STATA SURVIVAL ANALYSIS REFERENCE MANUAL
RELEASE 18

The suggested citation for this software is

StataCorp. 2023. *Stata 18*. Statistical software. StataCorp LLC.

The suggested citation for this manual is

StataCorp. 2023. *Stata 18 Survival Analysis Reference Manual*. College Station, TX: Stata Press.

www.stata.com

Contents

Cross-referencing the documentation

When reading this manual, you will find references to other Stata manuals, for example, [U] **27 Overview of Stata estimation commands**; [R] **regress**; and [D] **reshape**. The first example is a reference to chapter 27, *Overview of Stata estimation commands*, in the *User's Guide*; the second is a reference to the `regress` entry in the *Base Reference Manual*; and the third is a reference to the `reshape` entry in the *Data Management Reference Manual*.

All the manuals in the Stata Documentation have a shorthand notation:

[GSM]	*Getting Started with Stata for Mac*
[GSU]	*Getting Started with Stata for Unix*
[GSW]	*Getting Started with Stata for Windows*
[U]	*Stata User's Guide*
[R]	*Stata Base Reference Manual*
[ADAPT]	*Stata Adaptive Designs: Group Sequential Trials Reference Manual*
[BAYES]	*Stata Bayesian Analysis Reference Manual*
[BMA]	*Stata Bayesian Model Averaging Reference Manual*
[CAUSAL]	*Stata Causal Inference and Treatment-Effects Estimation Reference Manual*
[CM]	*Stata Choice Models Reference Manual*
[D]	*Stata Data Management Reference Manual*
[DSGE]	*Stata Dynamic Stochastic General Equilibrium Models Reference Manual*
[ERM]	*Stata Extended Regression Models Reference Manual*
[FMM]	*Stata Finite Mixture Models Reference Manual*
[FN]	*Stata Functions Reference Manual*
[G]	*Stata Graphics Reference Manual*
[IRT]	*Stata Item Response Theory Reference Manual*
[LASSO]	*Stata Lasso Reference Manual*
[XT]	*Stata Longitudinal-Data/Panel-Data Reference Manual*
[META]	*Stata Meta-Analysis Reference Manual*
[ME]	*Stata Multilevel Mixed-Effects Reference Manual*
[MI]	*Stata Multiple-Imputation Reference Manual*
[MV]	*Stata Multivariate Statistics Reference Manual*
[PSS]	*Stata Power, Precision, and Sample-Size Reference Manual*
[P]	*Stata Programming Reference Manual*
[RPT]	*Stata Reporting Reference Manual*
[SP]	*Stata Spatial Autoregressive Models Reference Manual*
[SEM]	*Stata Structural Equation Modeling Reference Manual*
[SVY]	*Stata Survey Data Reference Manual*
[ST]	*Stata Survival Analysis Reference Manual*
[TABLES]	*Stata Customizable Tables and Collected Results Reference Manual*
[TS]	*Stata Time-Series Reference Manual*
[I]	*Stata Index*
[M]	*Mata Reference Manual*

Title

> **Intro —** Introduction to survival analysis manual

Description

This manual documents commands for survival analysis and is referred to as [ST] in cross-references. Following this entry, [ST] **Survival analysis** provides an overview of the commands.

This manual is arranged alphabetically. If you are new to Stata's survival analysis, we recommend that you read the following sections first:

[ST] **Survival analysis**	Introduction to survival analysis commands
[ST] **st**	Survival-time data
[ST] **stset**	Set variables for survival data

Stata is continually being updated, and Stata users are always writing new commands. To find out about the latest survival analysis features, type search survival after installing the latest official updates; see [R] **update**.

Also see

[U] **1.3 What's new**

[R] **Intro** — Introduction to base reference manual

Title

> **Survival analysis —** Introduction to survival analysis commands

<p align="center"><a>Description   <a>Remarks and examples   <a>References   <a>Also see</p>

Description

Stata's survival analysis routines are used to compute sample size, power, and effect size and to declare, convert, manipulate, summarize, and analyze survival data. Survival data are time-to-event data, and survival analysis is full of jargon: truncation, censoring, hazard rates, etc. See the glossary in this manual. For a good Stata-specific introduction to survival analysis, see Cleves, Gould, and Marchenko (2016).

To learn how to effectively analyze survival analysis data using Stata, we recommend NetCourse 631, *Introduction to Survival Analysis Using Stata*; see https://www.stata.com/netcourse/nc631.html.

All the commands documented in this manual are listed below, and they are described in detail in their respective manual entries. While most commands for survival analysis are documented here, some are documented in other manuals. The commands for computing sample size, power, and effect size for survival analysis are documented in the *Stata Power, Precision, and Sample-Size Reference Manual* with the other power commands. The command for longitudinal or panel-data survival analysis is documented with the other panel-data commands in the *Stata Longitudinal-Data/Panel-Data Reference Manual*. The command for multilevel survival analysis is documented with the other multilevel commands in the *Stata Multilevel Mixed-Effects Reference Manual*. The commands for estimating treatment effects from observational survival-time data are documented in the *Stata Causal Inference and Treatment-Effects Estimation Reference Manual*. The commands for model selection and prediction using lasso and elastic net are documented in the *Stata Lasso Reference Manual*.

Declaring and converting count data

ctset	Declare data to be count-time data
cttost	Convert count-time data to survival-time data

Converting snapshot data

snapspan	Convert snapshot data to time-span data

Declaring and summarizing survival-time data

stset	Declare data to be survival-time data
stdescribe	Describe survival-time data
stsum	Summarize survival-time data

Manipulating survival-time data

stvary	Report variables that vary over time
stfill	Fill in by carrying forward values of covariates
stgen	Generate variables reflecting entire histories
stsplit	Split time-span records

<p align="center">2</p>

stjoin	Join time-span records
stbase	Form baseline dataset

Obtaining summary statistics, confidence intervals, tables, etc.

sts	Generate, graph, list, and test the survivor and related functions
stir	Report incidence-rate comparison
stci	Confidence intervals for means and percentiles of survival time
strate	Tabulate failure rate
stptime	Calculate person-time, incidence rates, and SMR
stmh	Calculate rate ratios with the Mantel–Haenszel method
stmc	Calculate rate ratios with the Mantel–Cox method
ltable	Display and graph life tables

Fitting regression models

stcox	Cox proportional hazards model
estat concordance	Compute the concordance probability
estat phtest	Test Cox proportional-hazards assumption
stphplot	Graphically assess the Cox proportional-hazards assumption
stcoxkm	Graphically assess the Cox proportional-hazards assumption
streg	Parametric survival models
stintreg	Parametric models for interval-censored survival-time data
estat gofplot	Graphically assess goodness of fit after streg, stcox, stintreg, or stintcox
stintcox	Cox proportional hazards model for interval-censored data
stintphplot	Graphically assess the Cox proportional-hazards assumption for interval-censored data
stintcoxnp	Graphically assess the Cox proportional-hazards assumption for interval-censored data
stcrreg	Competing-risks regression
xtstreg	Random-effects parametric survival models
mestreg	Multilevel mixed-effects parametric survival models
stcurve	Plot the survivor or related function after streg, stcox, and more
stteffects	Treatment-effects estimation for observational survival-time data
fmm: streg	Finite mixtures of parametric survival models
bayes: streg	Bayesian parametric survival models
bayes: mestreg	Bayesian multilevel parametric survival models

Prediction and model selection

lasso cox	Lasso selection of covariates in Cox proportional hazards models
elasticnet cox	Elastic net selection of covariates in Cox proportional hazards models

Sample size and power determination for survival analysis

power cox	Sample size, power, and effect size for the Cox proportional hazards model
power exponential	Sample size and power for the exponential test
power logrank	Sample size, power, and effect size for the log-rank test

Converting survival-time data

sttocc	Convert survival-time data to case–control data
sttoct	Convert survival-time data to count-time data

Programmer's utilities

st_*	Survival analysis subroutines for programmers

Remarks and examples

Remarks are presented under the following headings:

Introduction

All but one entry in this manual deals with the analysis of survival data, which is used to measure the time to an event of interest such as death or failure. Survival data can be organized in two ways. The first way is as *count data*, which refers to observations on populations, whether people or generators, with observations recording the number of units at a given time that failed or were lost because of censoring. The second way is as *survival-time*, or *time-span*, data. In survival-time data, the observations represent periods and typically contain three variables that record the start time of the period, the end time, and an indicator of whether failure or right-censoring occurred at the end of the period. The representation of the response of these three variables makes survival data unique in terms of implementing the statistical methods in the software. Such representation is specific to *right-censored survival-time data*. *Interval-censored survival-time data* are represented by two time variables that record the endpoints of time intervals in which failures are known to have occurred. Throughout the manual, when we refer to survival-time data, we will assume right-censored survival-time data. We will refer to interval-censored data explicitly.

Survival data may also be organized as *snapshot data* (a small variation of the survival-time format), in which observations depict an instance in time rather than an interval. When you have snapshot data, you simply use the snapspan command to convert it to survival-time data before proceeding.

Stata commands that begin with `ct` are used to convert count data to survival-time data. Survival-time data are analyzed using Stata commands that begin with `st`, known in our terminology as st commands. You can express all the information contained in count data in an equivalent survival-time dataset, but the converse is not true. Thus Stata commands are made to work with survival-time data because it is the more general representation.

All `st` commands, except `stintreg` and `stintcox`, are designed for right-censored survival-time data. The `stintreg` and `stintcox` commands analyze more general interval-censored survival-time (event-time) data.

Declaring and converting count data

Count data must first be converted to survival-time data before Stata's `st` commands can be used. Count data can be thought of as aggregated survival-time data. Rather than having observations that are specific to a subject and a period, you have data that, at each recorded time, record the number lost because of failure and, optionally, the number lost because of right-censoring.

`ctset` is used to tell Stata the names of the variables in your count data that record the time, the number failed, and the number censored. You `ctset` your data before typing `cttost` to convert it to survival-time data. Because you `ctset` your data, you can type `cttost` without any arguments to perform the conversion. Stata remembers how the data are `ctset`.

Converting snapshot data

Snapshot data are data in which each observation records the status of a given subject at a certain point in time. Usually you have multiple observations on each subject that chart the subject's progress through the study.

Before using Stata's survival analysis commands with snapshot data, you must first convert the data to survival-time data; that is, the observations in the data should represent intervals. When you convert snapshot data, the existing time variable in your data is used to record the end of a time span, and a new variable is created to record the beginning. Time spans are created using the recorded snapshot times as breakpoints at which new intervals are to be created. Before converting snapshot data to time-span data, you must understand the distinction between *enduring variables* and *instantaneous variables*. Enduring variables record characteristics of the subject that endure throughout the time span, such as sex or smoking status. Instantaneous variables describe events that occur at the end of a time span, such as failure or censoring. When you convert snapshots to intervals, enduring variables obtain their values from the previous recorded snapshot or are set to missing for the first interval. Instantaneous variables obtain their values from the current recorded snapshot because the existing time variable now records the end of the span.

Stata's `snapspan` makes this whole process easy. You specify an ID variable identifying your subjects, the snapshot time variable, the name of the new variable to hold the beginning times of the spans, and any variables that you want to treat as instantaneous variables. Stata does the rest for you.

Declaring and summarizing survival-time data

Stata does not automatically recognize survival-time data, so you must declare your survival-time data to Stata by using `stset`. Every st command, except `stintreg` and `stintcox`, relies on the information that is provided when you `stset` your data. Survival-time data come in different forms. For example, your time variables may be dates, time measured from a fixed date, or time measured from some other point unique to each subject, such as enrollment in the study. You can also consider

the following questions. What is the onset of risk for the subjects in your data? Is it time zero? Is it enrollment in the study or some other event, such as a heart transplant? Do you have censoring, and if so, which variable records it? What values does this variable record for censoring/failure? Do you have delayed entry? That is, were some subjects at risk of failure before you actually observed them? Do you have simple data and wish to treat everyone as entering and at risk at time zero?

Whatever the form of your data, you must first stset it before analyzing it, and so if you are new to Stata's st commands, we highly recommend that you take the time to learn about stset. It is really easy once you get the hang of it, and [ST] **stset** has many examples to help. For more discussion of stset, see Cleves, Gould, and Marchenko (2016, chap. 6).

Once you stset the data, you can use stdescribe to describe the aspects of your survival data. For example, you will see the number of subjects you were successful in declaring, the total number of records associated with these subjects, the total time at risk for these subjects, time gaps for any of these subjects, any delayed entry, etc. You can use stsum to summarize your survival data, for example, to obtain the total time at risk and the quantiles of time-to-failure in analysis-time units.

Manipulating survival-time data

Once your data have been stset, you may want to clean them up a bit before beginning your analysis. Suppose that you had an enduring variable and snapspan recorded it as missing for the interval leading up to the first recorded snapshot time. You can use stfill to fill in missing values of covariates, either by carrying forward the values from previous periods or by making the covariate equal to its earliest recorded (nonmissing) value for all time spans. You can use stvary to check for time-varying covariates or to confirm that certain variables, such as sex, are not time varying. You can use stgen to generate new covariates based on functions of the time spans for each given subject. For example, you can create a new variable called eversmoked that equals one for all of a subject's observations, if the variable smoke in your data is equal to one for *any* of the subject's time spans. Think of stgen as just a convenient way to do things that could be done using by *subject_id*: with survival-time data.

stsplit is useful for creating data that have multiple records per subject from data that have one record per subject. Suppose that you have already stset your data and wish to introduce a time-varying covariate. You would first need to stsplit your data so that separate time spans could be created for each subject, allowing the new covariate to assume different values over time within a subject. stjoin is the opposite of stsplit. Suppose that you have data with multiple records per subject but then realize that the data could be collapsed into single-subject records with no loss of information. Using stjoin would speed up any subsequent analysis using the st commands without changing the results.

stbase can be used to set every variable in your multiple-record st data to the value at baseline, defined as the earliest time at which each subject was observed. It can also be used to convert st data to cross-sectional data.

Obtaining summary statistics, confidence intervals, tables, etc.

Stata provides several commands for nonparametric analysis of survival data that can produce a wide array of summary statistics, inference, tables, and graphs. sts is a truly powerful command, used to obtain nonparametric estimates, inference, tests, and graphs of the survivor function, the cumulative hazard function, and the hazard function. You can compare estimates across groups, such as smoking versus nonsmoking, and you can adjust these estimates for the effects of other covariates in your data. sts can present these estimates as tables and graphs. sts can also be used to test the equality of survivor functions across groups.

`stir` is used to estimate incidence rates and to compare incidence rates across groups. `stci` is the survival-time data analog of `ci means` and is used to obtain confidence intervals for means and percentiles of time to failure. `strate` is used to tabulate failure rates. `stptime` is used to calculate person-time and standardized mortality/morbidity ratios (SMRs). `stmh` calculates rate ratios by using the Mantel–Haenszel method, and `stmc` calculates rate ratios by using the Mantel–Cox method.

`ltable` displays and graphs life tables for individual-level or aggregate data.

Fitting regression models

Stata has commands for fitting both semiparametric and parametric regression models to survival data. `stcox` fits the Cox proportional hazards model and `predict` after `stcox` can be used to retrieve estimates of the baseline survivor function, the baseline cumulative hazard function, and the baseline hazard contributions. `predict` after `stcox` can also calculate a myriad of Cox regression diagnostic quantities, such as martingale residuals, efficient score residuals, and Schoenfeld residuals. `stcox` has four options for handling tied failures. `stcox` can be used to fit stratified Cox models, where the baseline hazard is allowed to differ over the strata, and it can be used to model multivariate survival data by using a *shared-frailty* model, which can be thought of as a Cox model with random effects. After `stcox`, you can use `estat phtest` to test the proportional-hazards assumption or `estat concordance` to compute the concordance probability. With `stphplot` and `stcoxkm`, you can graphically assess the proportional-hazards assumption.

`stintcox` fits the Cox proportional hazards model for interval-censored data, and `predict` after `stintcox` can be used to obtain estimates of the baseline survivor function, the baseline cumulative hazard function, and the baseline hazard contributions. `predict` after `stintcox` can also calculate martingale-like residuals and Cox–Snell-like residuals. `stintcox` can be used to fit stratified Cox models, where the baseline hazard is allowed to differ over the strata. After `stintcox`, you can use `stintphplot` and `stintcoxnp` to graphically assess the proportional-hazards assumption.

Stata offers six parametric regression models for survival data: exponential, Weibull, lognormal, loglogistic, Gompertz, and generalized gamma. All six models are fit using `streg` for right-censored data and `stintreg` for interval-censored data, and you can specify the model you want with the `distribution()` option. All of these models, except for the exponential, have ancillary parameters that are estimated (along with the linear predictor) from the data. By default, these ancillary parameters are treated as constant, but you may optionally model the ancillary parameters as functions of a linear predictor. Stratified models may also be fit using `streg` and `stintreg`. You can also fit frailty models with `streg` and specify whether you want the frailties to be treated as spell-specific or shared across groups of observations.

`stcrreg` fits a semiparametric regression model for survival data in the presence of competing risks. Competing risks impede the failure event under study from occurring. An analysis of such competing-risks data focuses on the *cumulative incidence function*, the probability of failure in the presence of competing events that prevent that failure. `stcrreg` provides an analogue to `stcox` for such data. The baseline *subhazard function*—that which generates failures under competing risks—is left unspecified, and covariates act multiplicatively on the baseline subhazard.

You can also fit parametric survival models to clustered and hierarchical or multilevel data by using the `xtstreg` or `mestreg` command, respectively.

`xtstreg` fits random-intercept parametric survival models to clustered survival data. Random intercepts are assumed to be normally distributed. A random-intercept model with Gaussian intercepts can be viewed as a shared-frailty model with lognormal frailty. `xtstreg` supports five distributions: exponential, loglogistic, Weibull, lognormal, and gamma, which you can specify using the `distribution()` option. Several predictions, such as mean, median, or survivor or hazard functions, can be obtained by using `predict` after fitting a model with `xtstreg`.

`mestreg` fits multilevel mixed-effects parametric survival models. It supports five distributions: exponential, loglogistic, Weibull, lognormal, and gamma, which you can specify using the `distribution()` option. `mestreg` allows for multiple levels of random effects and for random coefficients. Marginal or conditional predictions for several statistics and functions of interest, such as mean, median, or survival or hazard functions, can be obtained by using `predict` after fitting a model with `mestreg`.

In addition, you can perform treatment-effects estimation for observational survival-time data by using `stteffects`. `stteffects` estimates average treatment effects, average treatment effects on the treated, and potential-outcome means using observational survival-time data. The available estimators are regression adjustment, inverse-probability weighting, and double-robust methods that combine regression adjustment and inverse-probability weighting; see [CAUSAL] **stteffects intro** for details.

`stcurve` plots the survivor, failure, hazard, or cumulative hazard function after `stcox`, `streg`, `stintreg`, `stintcox`, `stcrreg`, `mestreg`, or `xtstreg`. `stcurve` also plots the cumulative subhazard or cumulative incidence function after `stcrreg`. Covariates, by default, are held fixed at their mean values, but you can specify other values if you wish. `stcurve` is useful for comparing these functions across different levels of covariates.

`estat gofplot` creates a goodness of fit plot after `streg`, `stcox`, `stintreg`, or `stintcox`. This graph consists of the estimated cumulative hazard function for the Cox–Snell residuals plotted against the residuals themselves.

Prediction and model selection

Stata provides commands to select covariates and fit models using lasso and elastic net. Lasso is a solution to a penalized optimization problem, where the penalty is used to force some covariated to be omitted from the model. For more information on the lasso penalty, see [LASSO] **lasso**. Elastic net is similar to lasso; it uses a different penalty that performs better when groups of variables that are highly correlated. The results from lasso and elastic net are useful for prediction and model selection.

`lasso cox` selects covariates using lasso and fits a Cox proportional hazards model. After this command, predictions are available by using the standard `predict` postestimation command.

`elasticnet cox` selects covariates using elastic net and fits a Cox proportional hazards model. After this command, predictions are available by using the standard `predict` postestimation command.

Sample size and power determination for survival analysis

Stata has commands for computing sample size, power, and effect size for survival analysis using the log-rank test, the Cox proportional hazards model, and the exponential test comparing exponential hazard rates.

`power logrank` computes sample size, power, or effect size for survival analysis comparing survivor functions in two groups by using the log-rank test. The command supports unbalanced designs and provides options to account for administrative censoring, uniform accrual, and withdrawal of subjects from the study.

`power cox` computes sample size, power, or effect size for survival analyses that use Cox proportional hazards (PH) models. The results are obtained for the test of the effect of one covariate (binary or continuous) on time to failure adjusted for other predictors in a PH model. The command can account for the dependence between the covariate of interest and other model covariates, and it can adjust computations for censoring and for withdrawal of subjects for the study.

power exponential computes sample size or power for survival analysis comparing two exponential survivor functions by using parametric tests for the difference between hazards or, optionally, for the difference between log hazards. It accommodates unequal allocation between the two groups, flexible accrual of subjects into the study, and group-specific losses to follow-up. The accrual distribution may be chosen to be uniform or truncated exponential over a fixed accrual period.

The commands allow automated production of customizable tables and graphs; see [PSS-2] **power** for details.

Converting survival-time data

Stata has commands for converting survival-time data to case–control and count data. These commands are rarely used, because most of the analyses are performed using data in the survival-time format. sttocc is useful for converting survival data to case–control data suitable for estimation with clogit. sttoct is the opposite of cttost and will convert survival-time data to count data.

Programmer's utilities

Stata also provides routines for programmers interested in writing their own st commands. These are basically utilities for setting, accessing, and verifying the information saved by stset. For example, st_is verifies that the data have in fact been stset and gives the appropriate error if not. st_show is used to preface the output of a program with key information on the st variables used in the analysis. Programmers interested in writing st code should see [ST] **st_is**.

References

Bower, H., T. M.-L. Andersson, M. J. Crowther, and P. C. Lambert. 2022. Flexible parametric survival analysis with multiple timescales: Estimation and implementation using stmt. *Stata Journal* 22: 679–701.

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata Press.

Crowther, M. J. 2022. Simulating time-to-event data from parametric distributions, custom distributions, competing-risks models, and general multistate models. *Stata Journal* 22: 3–24.

Also see

[ST] **stset** — Declare data to be survival-time data

[ST] **Intro** — Introduction to survival analysis manual

Stata Power, Precision, and Sample-Size Reference Manual

Stata Meta-Analysis Reference Manual

Title

<div style="border:1px solid black; padding:10px;">

ct — Count-time data

</div>

Description   Also see

Description

The term *ct* refers to count-time data and the commands—all of which begin with the letters "ct"—for analyzing them. If you have data on populations, whether people or generators, with observations recording the number of units under test at time t (subjects alive) and the number of subjects that failed or were lost because of censoring, you have what we call count-time data.

If, on the other hand, you have data on individual subjects with observations recording that this subject came under observation at time t_0 and that later, at t_1, a failure or censoring was observed, you have what we call survival-time data. If you have survival-time data, see [ST] **st**.

Do not confuse count-time data with counting-process data, which can be analyzed using the st commands; see [ST] **st**.

There are two ct commands:

ctset	[ST] **ctset**	Declare data to be count-time data
cttost	[ST] **cttost**	Convert count-time data to survival-time data

The key is the cttost command. Once you have converted your count-time data to survival-time data, you can use the st commands to analyze the data. The entire process is as follows:

1. ctset your data so that Stata knows that they are count-time data; see [ST] **ctset**.

2. Type cttost to convert your data to survival-time data; see [ST] **cttost**.

3. Use the st commands; see [ST] **st**.

Also see

[ST] **ctset** — Declare data to be count-time data

[ST] **cttost** — Convert count-time data to survival-time data

[ST] **st** — Survival-time data

[ST] **Survival analysis** — Introduction to survival analysis commands

Title

> **ctset —** Declare data to be count-time data

Description

ct refers to count-time data and is described here and in [ST] **ct**. Do not confuse count-time data with counting-process data, which can be analyzed using the st commands; see [ST] **st**.

When specified with a *timevar* and *nfailvar*, ctset declares the data in memory to be ct data. When you ctset your data, ctset also checks that what you have declared makes sense.

ctset, noshow will suppress display of the identities of the key ct variables before the output of other ct commands. By default, this information is shown. If you type ctset, noshow and then wish to restore the default behavior, type ctset, show.

ctset, clear is used mostly by programmers and causes Stata to no longer consider the data to be ct data. The dataset itself remains unchanged. It is not necessary to type ctset, clear before doing another ctset.

ctset typed without arguments—which can be abbreviated ct—displays the identities of the key ct variables and reruns the checks on your data. Thus ct can remind you of what you have ctset (especially if you have ctset, noshow) and reverify your data if you make changes to the data.

Quick start

Declare count-time data with number of failures, fail, at each time in tvar
> ctset tvar fail

Same as above, and specify the number censored, cens, at each time
> ctset tvar fail cens

Same as above, and specify the number entering, enter, at each time
> ctset tvar fail cens enter

Specify that the number of failures and the number censored are recorded for groups identified by v1
> ctset tvar fail cens, by(v1)

Display previous ct settings, and verify that any changes to data correspond to settings
> ctset

Do not display information on variables specified in ctset when ct commands are run
> ctset, noshow

Menu

Statistics > Survival analysis > Setup and utilities > Declare data to be count-time data

Syntax

Declare data in memory to be count-time data and run checks on data

   ctset *timevar nfailvar* [*ncensvar* [*nentvar*]] [, by(*varlist*) no<u>s</u>how]

Specify whether to display identities of key ct variables

   ctset, { <u>s</u>how | <u>nos</u>how }

Clear ct setting

  ctset, clear

Display identity of key ct variables and rerun checks on data

   { ctset | ct }

where *timevar* refers to the time of failure, censoring, or entry. It should contain times ≥ 0.

nfailvar records the number failing at time *timevar*.

ncensvar records the number censored at time *timevar*.

nentvar records the number entering at time *timevar*.

Stata sequences events at the same time as

at *timevar*	*nfailvar* failures occurred,
then at *timevar* $+ 0$	*ncensvar* censorings occurred, and
finally at *timevar* $+ 0 + 0$	*nentvar* subjects entered the data.

Options

by(*varlist*) indicates that counts are provided by group. For instance, consider data containing records
  such as

t	fail	cens	sex	agecat
5	10	2	0	1
5	6	1	1	1
5	12	0	0	2

These data indicate that, in the category sex $= 0$ and agecat $= 1$, 10 failed and 2 were censored
at time 5; for sex $= 1$, 1 was censored and 6 failed; and so on.

The above data would be declared

   . ctset t fail cens, by(sex agecat)

The order of the records is not important, nor is it important that there be a record at every time
for every group or that there be only one record for a time and group. However, the data must
contain the full table of events.

show and noshow specify whether the identities of the key ct variables are to be displayed at the
start of every ct command. Some users find the report reassuring; others find it repetitive. In any
case, you can set and unset show, and you can always type ct to see the summary.

clear makes Stata no longer consider the data to be ct data.

Remarks and examples

Remarks are presented under the following headings:

Examples
Data errors flagged by ctset

Examples

About all you can do with ct data in Stata is convert it to survival-time (st) data so that you can use the survival analysis commands. To analyze count-time data with Stata,

```
. ctset ...
. cttost
. (now use any of the st commands)
```

▷ Example 1: Simple ct data

We have data on generators that are run until they fail:

```
. use https://www.stata-press.com/data/r18/ctset1
. list, sep(0)
```

	failtime	fail
1.	22	1
2.	30	1
3.	40	2
4.	52	1
5.	54	4
6.	55	2
7.	85	7
8.	97	1
9.	100	3
10.	122	2
11.	140	1

For instance, at time 54, four generators failed. To ctset these data, we could type

```
. ctset failtime fail
Count-time data settings
           Time: failtime
       Failures: fail
    Number lost: <none>
Number entered: All enter at time 0
```

It is not important that there be only 1 observation per failure time. For instance, according to our data, at time 85 there were seven failures. We could remove that observation and substitute two in its place—one stating that at time 85 there were five failures and another that at time 85 there were two more failures. ctset would interpret that data just as it did the previous data.

In more realistic examples, the generators might differ from one another. For instance, the following data show the number failing with old-style and new-style bearings:

```
. use https://www.stata-press.com/data/r18/ctset2
. list, sepby(bearings)
```

	bearings	failtime	fail
1.	Old-style	22	1
2.	Old-style	40	2
3.	Old-style	54	1
4.	Old-style	84	2
5.	Old-style	97	2
6.	Old-style	100	1
7.	New-style	30	1
8.	New-style	52	1
9.	New-style	55	1
10.	New-style	100	3
11.	New-style	122	2
12.	New-style	140	1

That the data are sorted on bearings is not important. The ctset command for these data is

```
. ctset failtime fail, by(bearings)
Count-time data settings
           Time: failtime
       Failures: fail
    Number lost: <none>
Number entered: All enter at time 0
Group variable: bearings
```

◁

▷ Example 2: ct data with censoring

In real data, not all units fail in the time allotted. Say that the generator experiment was stopped after 150 days. The data might be

```
. use https://www.stata-press.com/data/r18/ctset3
. list
```

	bearings	failtime	fail	censored
1.	Old-style	22	1	0
2.	Old-style	40	2	0
3.	Old-style	54	1	0
4.	Old-style	84	2	0
5.	New-style	97	2	0
6.	Old-style	100	1	0
7.	Old-style	150	0	2
8.	New-style	30	1	0
9.	New-style	52	1	0
10.	New-style	55	1	0
11.	New-style	122	2	0
12.	New-style	140	1	0
13.	New-style	150	0	3

The `ctset` command for these data is

```
. ctset failtime fail censored, by(bearings)
Count-time data settings
           Time: failtime
       Failures: fail
   Number lost: censored
Number entered: All enter at time 0
Group variable: bearings
```

In some other data, observations might also be censored along the way; that is, the value of censored would not be 0 before time 150. For instance, a record might read

```
        bearings    failtime        fail    censored
               0          84           2           1
```

This would mean that at time 84, two failed and one was lost because of censoring. The failure and censoring occurred at the same time, and when we analyze these data, Stata will assume that the censored observation could have failed, that is, that the censoring occurred after the two failures.

◁

▷ Example 3: ct data with delayed entry

Data on survival time of patients with a particular kind of cancer are collected. Time is measured as time since diagnosis. After data collection started, the sample was enriched with some patients from hospital records who had been previously diagnosed. Some of the data are

```
 time      die      cens       ent     other variables
    0        0         0        50
    1        0         0         5     ...
    .
    .
    .
   30        0         0         3     ...
   31        0         1         2     ...
   32        1         0         1     ...
    .
    .
    .
  100        1         1         0     ...
    .
    .
    .
```

Fifty patients entered at time 0 (time of diagnosis); five patients entered 1 day after diagnosis; and three, two, and one patients entered 30, 31, and 32 days after diagnosis, respectively. On the 32nd day, one of the previously entered patients died.

If the other variables are named `sex` and `agecat`, the `ctset` command for these data is

```
. ctset time die cens ent, by(sex agecat)
Count-time data settings
            Time: time
        Failures: die
    Number lost: cens
 Number entered: ent
Group variables: sex agecat
```

◁

The count-time format is an inferior way to record data like these—data in which every subject does not enter at time 0—because some information is already lost. When did the patient who died on the 32nd day enter? There is no way of telling.

For traditional survival analysis calculations, it does not matter. More modern methods of estimating standard errors, however, seek to identify each patient, and these data do not support using such methods.

This issue concerns the robust estimates of variance and the `vce(robust)` options on some of the st analysis commands. After converting the data, you must not use the `vce(robust)` option, even if an st command allows it, because the identities of the subjects—tying together when a subject starts and ceases to be at risk—are assigned randomly by `cttost` when you convert your ct to st data. When did the patient who died on the 32nd day enter? For conventional calculations, it does not matter, and `cttost` chooses a time randomly from the available entry times.

Data errors flagged by ctset

`ctset` requires only two things of your data: that the counts all be positive or zero and, if you specify an entry variable, that the entering and exiting subjects (failure + censored) balance.

If all subjects enter at time 0, we recommend that you do not specify a number-that-enter variable. `ctset` can determine for itself the number who enter at time 0 by summing the failures and censorings.

Also see

[ST] **ct** — Count-time data

[ST] **cttost** — Convert count-time data to survival-time data

Title

> **cttost** — Convert count-time data to survival-time data

Description

cttost converts count-time data to their survival-time format so that they can be analyzed with Stata. Do not confuse count-time data with counting-process data, which can also be analyzed with the st commands; see [ST] **ctset** for a definition and examples of count data.

Quick start

Convert count-time data to survival-time data using ctset data

    cttost

Same as above, but name the new weight variable mywvar instead of using the default name

    cttost, wvar(mywvar)

Menu

Statistics > Survival analysis > Setup and utilities > Convert count-time data to survival-time data

Syntax

> cttost [, *options*]

options	Description
t0(*t0var*)	name of entry-time variable
wvar(*wvar*)	name of frequency-weighted variable
clear	overwrite current data in memory
nopreserve	do not save the original data; programmer's command

You must ctset your data before using cttost; see [ST] **ctset**.

nopreserve does not appear in the dialog box.

Options

t0(*t0var*) specifies the name of the new variable to create that records entry time. (For most ct data, no entry-time variable is necessary because everyone enters at time 0.)

Even if an entry-time variable is necessary, you need not specify this option. cttost will, by default, choose t0, time0, or etime according to which name does not already exist in the data.

wvar(*wvar*) specifies the name of the new variable to be created that records the frequency weights for the new pseudo-observations. Count-time data are actually converted to frequency-weighted st data, and a variable is needed to record the weights. This sounds more complicated than it is. Understand that cttost needs a new variable name, which will become a permanent part of the st data.

If you do not specify wvar(), cttost will, by default, choose w, pop, weight, or wgt according to which name does not already exist in the data.

clear specifies that it is okay to proceed with the conversion, even though the current dataset has not been saved on disk.

The following option is available with cttost but is not shown in the dialog box:

nopreserve speeds the conversion by not saving the original data that can be restored should things go wrong or should you press *Break*. nopreserve is intended for use by programmers who use cttost as a subroutine. Programmers can specify this option if they have already preserved the original data. nopreserve does not affect the conversion.

Remarks and examples

Converting ct to st data is easy. We have some count-time data,

```
. use https://www.stata-press.com/data/r18/cttost
. ct
Count-time data settings
          Time: time
      Failures: ndead
   Number lost: ncens
Number entered: All enter at time 0
Group variables: agecat treat
```

```
. list in 1/5
```

	agecat	treat	time	ndead	ncens
1.	2	1	464	4	0
2.	3	0	268	3	1
3.	2	0	638	2	0
4.	1	0	803	1	4
5.	1	0	431	2	0

and to convert it, we type cttost:

```
. cttost
Count-time data settings
           Time: time
       Failures: ndead
    Number lost: ncens
 Number entered: All enter at time 0
Group variables: agecat treat

Converting count-time data to survival-time data ...

Survival-time data settings
         Failure event: ndead!=0 & ndead<.
Observed time interval: (0, time]
    Exit on or before: failure
               Weight: [fweight=w]
```

```
           33  total observations
            0  exclusions

           33  physical observations remaining, equal to
           82  weighted observations, representing
           39  failures in single-record/single-failure data
       48,726  total analysis time at risk and under observation
                                          At risk from t =          0
                                 Earliest observed entry t =          0
                                   Last observed exit t =      1,227
```

Now that it is converted, we can use any of the st commands:

```
. sts test treat, logrank
       Failure _d: ndead
 Analysis time _t: time
           Weight: [fweight=w]

Equality of survivor functions
Log-rank test
```

treat	Observed events	Expected events
0	22	17.05
1	17	21.95
Total	39	39.00

```
              chi2(1) =    2.73
              Pr>chi2 = 0.0986
```

Also see

[ST] **ct** — Count-time data

[ST] **ctset** — Declare data to be count-time data

Title

> **Discrete** — Discrete-time survival analysis

Description

As of the date that this manual was printed, Stata does not have a suite of built-in commands for discrete-time survival models matching the `st` suite for continuous-time models, but a good case could be made that it should. Instead, these models can be fit easily using other existing estimation commands and data manipulation tools.

Discrete-time survival analysis concerns analysis of time-to-event data whenever survival times are either a) intrinsically discrete (for example, numbers of machine cycles) or b) grouped into discrete intervals of time ("interval-censoring"). If intervals are of equal length, the same methods can be applied to both a) and b); survival times will be positive integers.

You can fit discrete-time survival models with the maximum likelihood method. Data may contain completed or right-censored spells, and late entry (left-truncation) can also be handled, as well as unobserved heterogeneity (also termed "frailty"). Estimation makes use of the property that the sample likelihood can be rewritten in a form identical to the likelihood for a binary dependent variable multiple regression model and applied to a specially organized dataset (Allison 2014, Jenkins 1995). For models without frailty, you can use, for example, `logistic` (or `logit`) to fit the discrete-time logistic hazard model or `cloglog` to fit the discrete-time proportional hazards model (Prentice and Gloeckler 1978). Models incorporating normal frailty may be fit using `xtlogit` and `xtcloglog`. A model with gamma frailty (Meyer 1990) may be fit using `pgmhaz` (Jenkins 1997).

Estimation consists of three steps:

1. *Data organization*: The dataset must be organized so that there is 1 observation for each period when a subject is at risk of experiencing the transition event. For example, if the original dataset contains one row for each subject, i, with information about their spell length, T_i, the new dataset requires T_i rows for each subject, one row for each period at risk. This may be accomplished using `expand` or `stsplit`. (This step is episode splitting at each and every interval.) The result is data of the same form as a discrete panel (`xt`) dataset with repeated observations on each panel (subject).

2. *Variable creation*: You must create at least three types of variables. First, you will need an interval identification variable, which is a sequence of positive integers $t = 1, \ldots, T_i$. For example,

   ```
   . sort subject_id
   . by subject_id: generate t = _n
   ```

   Second, you need a period-specific censoring indicator, d_i. If $d_i = 1$ if subject i's spell is complete and $d_i = 0$ if the spell is right-censored, the new indicator $d_{it}^* = 1$ if $d_i = 1$ and $t = T_i$, and $d_{it}^* = 0$ otherwise.

   Third, you must define variables (as functions of t) to summarize the pattern of duration dependence. These variables are entered as covariates in the regression. For example, for a duration dependence pattern analogous to that in the continuous-time Weibull model, you could define a new variable $x_1 = \log t$. For a quadratic specification, you define variables $x_1 = t$ and $x_2 = t^2$. We can achieve a piecewise constant specification by defining a set of dummy variables, with each group of periods sharing the same hazard rate, or a semiparametric model (analogous to the Cox regression model for continuous survival-time data) using separate dummy variables for each and every duration

interval. No duration variable need be defined if you want to fit a model with a constant hazard rate.

In addition to these three essentials, you may define other time-varying covariates.

3. *Estimation*: You fit a binary dependent variable multiple regression model, with d_{it}^* as the dependent variable and covariates, including the duration variables and any other covariates.

For estimation using spell data with late entry, the stages are the same as those outlined above, with one modification and one warning. To fit models without frailty, you must drop all intervals prior to each subject's entry to the study. For example, if entry is in period e_i, you drop it if $t < e_i$. If you want to fit frailty models on the basis of discrete-time data with late entry, then be aware that the estimation procedure outlined does not lead to correct estimates. (The sample likelihood in the reorganized data does not account for conditioning for late entry here. You will need to write your own likelihood function by using ml; see [R] **Maximize**.)

To derive predicted hazard rates, use the predict command. For example, after logistic or cloglog, use predict, pr. After xtlogit or xtcloglog, use predict, pu0 (which predicts the hazard assuming the individual effect is equal to the mean value). Estimates of the survivor function, S_{it}, can then be derived from the predicted hazard rates, p_{it}, because $S_{it} = (1 - p_{i1})(1 - p_{i2})(\cdots)(1 - p_{it})$.

Acknowledgment

References

Allison, P. D. 2014. *Event History and Survival Analysis*. 2nd ed. Newbury Park, CA: Sage.

Jenkins, S. P. 1995. Easy estimation methods for discrete-time duration models. *Oxford Bulletin of Economics and Statistics* 57: 129–138. https://doi.org/10.1111/j.1468-0084.1995.tb00031.x.

——. 1997. sbe17: Discrete time proportional hazards regression. *Stata Technical Bulletin* 39: 22–32. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 109–121. College Station, TX: Stata Press.

Meyer, B. D. 1990. Unemployment insurance and unemployment spells. *Econometrica* 58: 757–782. https://doi.org/10.2307/2938349.

Prentice, R. L., and L. A. Gloeckler. 1978. Regression analysis of grouped survival data with application to breast cancer data. *Biometrics* 34: 57–67. https://doi.org/10.2307/2529588.

Also see

[ST] **stcox** — Cox proportional hazards model

[ST] **stcrreg** — Competing-risks regression

[ST] **streg** — Parametric survival models

[D] **expand** — Duplicate observations

[R] **cloglog** — Complementary log–log regression

[R] **logistic** — Logistic regression, reporting odds ratios

[XT] **xtcloglog** — Random-effects and population-averaged cloglog models

[XT] **xtlogit** — Fixed-effects, random-effects, and population-averaged logit models

Title

> **estat gofplot —** Goodness-of-fit plots after streg, stcox, stintreg, or stintcox

Description

estat gofplot plots the estimated cumulative hazard function for the Cox−Snell residuals versus the residuals themselves to assess the goodness of fit of the model visually after streg, stcox, stintreg, or stintcox.

Quick start

Plot the default cumulative hazard function for the Cox−Snell residuals versus the residuals themselves

        estat gofplot

Plot the minus log of the Kaplan−Meier estimator for the Cox−Snell residuals versus the residuals themselves

        estat gofplot, km

Plot separate cumulative hazard functions for each group of x on the same graph

        estat gofplot, by(x)

Plot separate cumulative hazard functions for each group of x on different graphs

        estat gofplot, by(x) separate

For the stratified model, plot separate cumulative hazard functions for each stratum

        estat gofplot, stratify

Menu for estat

Statistics > Postestimation

Syntax

> estat gofplot [, *options*]

options	Description
na	calculate the cumulative hazard function of the Cox−Snell residuals using the Nelson−Aalen estimator; the default for and available only after streg and stcox
km	calculate the cumulative hazard function of the Cox−Snell residuals using the minus log of the Kaplan−Meier estimator; available only after streg and stcox
turnbull	calculate the cumulative hazard function of the Cox−Snell-like residuals using the Turnbull estimator; the estimator for stintreg and stintcox
by(*varlist*)	estimate and graph separate functions for each group formed by *varlist*
stratify	estimate and graph separate functions for each stratum defined by strata() in estimation
separate	show curves on separate graphs; default is to show curves one on top of another
<u>out</u>file(*filename* [, replace])	save values used to plot the goodness-of-fit graph
Plot	
connect_options	affect rendition of plotted cumulative hazard function
Reference line	
rlopts(*cline_options*)	affect rendition of the reference line
Add plots	
addplot(*plot*)	add other plots to the generated graph
Y axis, X axis, Titles, Legend, Overall	
twoway_options	any options other than by() documented in [G-3] *twoway_options*

estat gofplot is not appropriate with svy estimation results and is not supported after estimation with stcox, tvc().

Options

na specifies that the cumulative hazard function of the Cox−Snell residuals be calculated using the Nelson−Aalen estimator, which is the default after streg and stcox. na is not available after stintreg and stintcox.

km specifies that the cumulative hazard function of the Cox−Snell residuals be calculated using the minus log of the Kaplan−Meier estimator instead of the default Nelson−Aalen estimator after streg and stcox. km is not available after stintreg and stintcox.

turnbull specifies that the cumulative hazard function of the Cox−Snell-like residuals be calculated using the Turnbull estimator. turnbull is the only estimator after stintreg and stintcox, and it is not available after streg and stcox.

by(*varlist*) estimates a separate function for each by-group and plots all the functions on one graph. By-groups are identified by equal values of the variables in *varlist*. Up to five variables are allowed. by() may not be combined with stratify.

stratify requires that a stratified model has been previously fit using the strata() option; it estimates a separate function for each stratum and plots all the functions on one graph. stratify may not be combined with by().

separate is meaningful only with by() or stratify; it requests that each group be placed on its own graph rather than one on top of the other.

outfile(*filename*[, replace]) saves in *filename*.dta the values used to plot the goodness-of-fit graph.

Plot

connect_options affect the rendition of the plotted cumulative hazard function; see [G-3] *connect_options*.

Reference line

rlopts(*cline_options*) affects the rendition of the reference line; see [G-3] *cline_options*.

Add plots

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] *addplot_option*.

Y axis, X axis, Titles, Legend, Overall

twoway_options are any of the options documented in [G-3] *twoway_options*, excluding by(). These include options for titling the graph (see [G-3] *title_options*) and for saving the graph to disk (see [G-3] *saving_option*).

Remarks and examples

To assess the overall model fit, we can use the Cox–Snell residuals. If the survival regression model fits the data, these residuals should have a censored standard exponential distribution for right-censored data. Therefore, when we consider these residuals as failure (or censoring) times together with the original censoring variable, the hazard function should be constant and equal to 1, and the cumulative hazard should be a straight line with slope 1. estat gofplot allows us to verify the model's fit visually by calculating an empirical estimate of such a cumulative hazard function, which is based on either the Nelson–Aalen estimator or the Kaplan–Meier estimator for streg or stcox or is based on the Turnbull estimator for stintreg or stintcox. If the model fits the data, a plot of the cumulative hazard versus the residuals themselves should approximate a straight line with slope 1. See example 2 in [ST] **streg postestimation**, example 4 in [ST] **stcox postestimation**, example 4 in [ST] **stintreg postestimation**, and example 2 in [ST] **stintcox postestimation** for more detailed discussions. For interval-censored data, the Cox–Snell-like residuals are defined, and, under the correct model assumption, they are expected to approximate an interval-censored sample from the standard exponential distribution; see [ST] **stintcox postestimation**.

▷ Example 1: Goodness-of-fit plots for stratified model

Returning to the Stanford heart experiment data from example 8 in [ST] **stcox**, we refit our model stratified by year of diagnosis group.

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)
. generate pgroup = year
. recode pgroup min/69=1 70/72=2 73/max=3
(172 changes made to pgroup)
. stcox age posttran surg year, strata(pgroup) nolog
        Failure _d: died
  Analysis time _t: t1
       ID variable: id
Stratified Cox regression with Breslow method for ties
Strata variable: pgroup
No. of subjects =       103                      Number of obs =      172
No. of failures =        75
Time at risk    = 31,938.1
                                                 LR chi2(4)    =    20.67
Log likelihood = -213.35033                      Prob > chi2   =   0.0004
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.027406	.0150188	1.85	0.064	.9983874	1.057268
posttran	1.075476	.3354669	0.23	0.816	.583567	1.982034
surgery	.2222415	.1218386	-2.74	0.006	.0758882	.6508429
year	.5523966	.1132688	-2.89	0.004	.3695832	.825638

We can visually explore the goodness of fit for the stratified model by using estat gofplot. Here we will use the stratify option to plot the residuals separately for each stratum.

```
. estat gofplot, stratify
```


Goodness-of-fit plot for the stratified Cox model
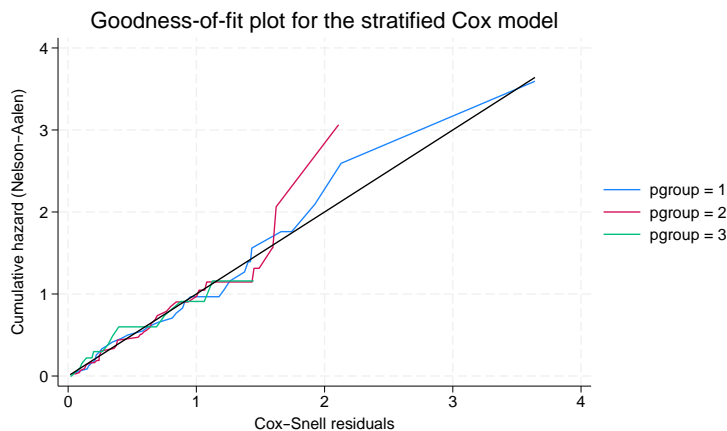
Even with perfect (artificially simulated) data, we expect to see departures from the diagonal at the right end of the curve, where values are based on only a few observations and greater fluctuation is observed. The above plot indicates the model fits well in all strata. To aid visual inspection of the plot, we can also add the `separate` option to produce separate graphs for each stratum.

```
. estat gofplot, stratify separate
```

Goodness-of-fit plot for the stratified Cox model

Graphs by pgroup

◁
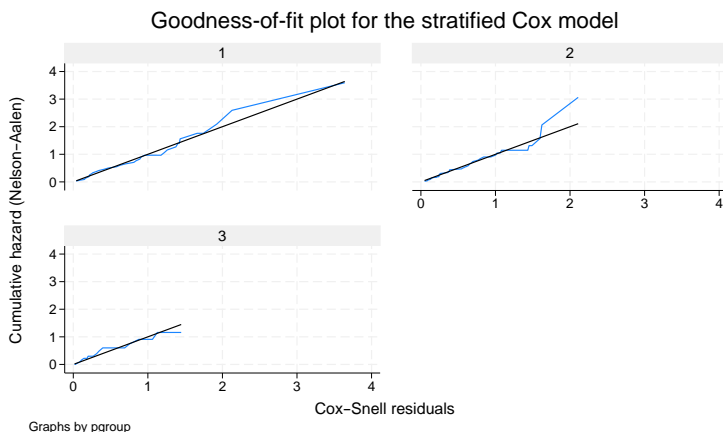
Methods and formulas

The Cox and Snell (1968) residual for the jth observation at time t_j is defined as the estimated cumulative hazard function, $\widehat{H}_j(t_j) = -\log\widehat{S}_j(t_j)$, from the fitted model (Klein and Moeschberger 2003). Cox and Snell argued that if the correct model has been fit to the data, these residuals are n observations from a censored standard exponential distribution for right-censored data. Thus, a plot of the cumulative hazard rate of the residuals against the residuals themselves should result in a straight line of slope 1. Cox–Snell residuals can never be negative and therefore are not symmetric about 0. In practice, we can calculate an empirical estimate of the cumulative hazard rate of the residuals.

The default method of calculating the cumulative hazard rate of the residuals after `streg` and `stcox` is to use the Nelson–Aalen estimator (Nelson 1972; Aalen 1978). Alternatively, we may use the minus log of the Kaplan and Meier (1958) estimator by specifying the `km` option. For multiple-record data, the overall Cox–Snell residual is used, and hence, the cumulative hazard function is evaluated at the subject level defined by `id()` in the `stset` command.

For interval-censored data, Cox–Snell-like residuals are intervals themselves for single-record-per-subject dataset. Farrington (2000) proposed to calculate the Cox–Snell-like residuals for both lower and upper endpoints of the time intervals, then to use those predicted Cox–Snell-like residual intervals as the new time intervals and to compute the cumulative hazard function using the Turnbull estimator (Turnbull 1976). For multiple-record-per-subject data fit by `stintcox`, the overall Cox–Snell-like residual is used, and the cumulative hazard function is evaluated at the subject level defined by the `id()` option of `stintcox`.

References

Aalen, O. O. 1978. Nonparametric inference for a family of counting processes. *Annals of Statistics* 6: 701–726. https://doi.org/10.1214/aos/1176344247.

Cox, D. R., and E. J. Snell. 1968. A general definition of residuals (with discussion). *Journal of the Royal Statistical Society, Series B* 30: 248–275. https://doi.org/10.1111/j.2517-6161.1968.tb00724.x.

Farrington, C. P. 2000. Residuals for proportional hazards models with interval-censored survival data. *Biometrics* 56: 473–482. https://doi.org/10.1111/j.0006-341X.2000.00473.x.

Kaplan, E. L., and P. Meier. 1958. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association* 53: 457–481. https://doi.org/10.2307/2281868.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data.* 2nd ed. New York: Springer.

Nelson, W. 1972. Theory and applications of hazard plotting for censored failure data. *Technometrics* 14: 945–966. https://doi.org/10.2307/1267144.

Turnbull, B. W. 1976. The empirical distribution function with arbitrarily grouped, censored and truncated data. *Journal of the Royal Statistical Society, Series B* 38: 290–295. https://doi.org/10.1111/j.2517-6161.1976.tb01597.x.

Also see

[ST] **stcox postestimation** — Postestimation tools for stcox

[ST] **stintcox postestimation** — Postestimation tools for stintcox

[ST] **stintreg postestimation** — Postestimation tools for stintreg

[ST] **streg postestimation** — Postestimation tools for streg

[U] **20 Estimation and postestimation commands**

Title

ltable — Life tables for survival data

Description

ltable displays and graphs life tables for individual-level or aggregate data and optionally presents the likelihood-ratio and log-rank tests for equivalence of groups. ltable also allows you to examine the empirical hazard function through aggregation.

Quick start

Life table for time variable tvar and death indicator died

    ltable tvar died

Same as above, but graph results with confidence intervals instead and suppress table

    ltable tvar died, graph ci notable

Life tables for each group defined by catvar with results saved to mydata.dta

    ltable tvar died, by(catvar) saving(mydata)

Aggregate time into thirty-day intervals, and suppress actuarial adjustment

    ltable tvar died, intervals(30) noadjust

Cumulative failure table for observations where catvar equals 1

    ltable tvar died if catvar==1, failure

Hazard table with frequency weights wvar

    ltable tvar died [fweight=wvar], hazard

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Life tables for survival data

Syntax

> ltable *timevar* [*deadvar*] [*if*] [*in*] [*weight*] [, *options*]

timevar specifies the time of failure or censoring. If *deadvar* is not specified, all values of *timevar* are interpreted as failure times. Observations with *timevar* equal to missing are ignored.

deadvar specifies how the time recorded in *timevar* is to be interpreted. Observations with *deadvar* equal to 0 are treated as censored and all other nonmissing values indicate that *timevar* should be interpreted as a failure time. Observations with *deadvar* equal to missing are ignored.

deadvar does not specify the number of failures. Specify frequency weights for aggregated data recording the number of failures.

options	Description
Main	
<u>su</u>rvival	display survival table; the default
<u>f</u>ailure	display cumulative failure table
<u>h</u>azard	display hazard table
by(*groupvar*)	produce separate tables (or graphs) for each value of *groupvar*
test	report χ^2 measure of differences between groups (2 tests)
overlay	overlay plots on the same graph
notable	display graph only; suppress display of table
graph	present the table graphically, as well as in tabular form
ci	graph confidence interval
level(*#*)	set confidence level; default is level(95)
<u>noadj</u>ust	suppress actuarial adjustment to the number at risk
tvid(*varname*)	subject ID variable to use with time-varying parameters
<u>i</u>ntervals(w | *numlist*)	time intervals in which data are to be aggregated for tables
saving(*filename* [, replace])	save the life-table data to *filename*; use replace to overwrite existing *filename*
Plot	
plotopts(*plot_options*)	affect rendition of the plotted line and plotted points
plot#opts(*plot_options*)	affect rendition of the #th plotted line and plotted points; available only with overlay
CI plot	
ciopts(*rspike_options*)	affect rendition of the confidence intervals
ci#opts(*rspike_options*)	affect rendition of the #th confidence interval; available only with overlay
Add plots	
addplot(*plot*)	add other plots to the generated graph
Y axis, X axis, Titles, Legend, Overall	
twoway_options	any options other than by() documented in [G-3] *twoway_options*
byopts(*byopts*)	how subgraphs are combined, labeled, etc.

plot_options	Description
connect_options	change look of lines or connecting method
marker_options	change look of markers (color, size, etc.)

fweights are allowed; see [U] **11.1.6 weight**.

Options

    �Main◻

survival, failure, and hazard indicate the table to be displayed. If none is specified, the default is the survival table. Specifying failure displays the cumulative failure table. Specifying survival failure would display both the survival and the cumulative failure table. If graph is specified, multiple tables may not be requested.

by(*groupvar*) creates separate tables (or graphs within the same image) for each value of *groupvar*. *groupvar* may be string or numeric.

test presents two χ^2 measures of the differences between groups, the likelihood-ratio test of homogeneity and the log-rank test for equality of survivor functions. The two groups are identified by the by() option, so by() must also be specified.

overlay causes the plot from each group identified in the by() option to be overlaid on the same graph. The default is to generate a separate graph (within the same image) for each group. This option requires the by() option.

notable suppresses displaying the table. This option is often used with graph.

graph requests that the table be presented graphically, as well as in tabular form; when notable is also specified, only the graph is presented. When you specify graph, only one table can be calculated and graphed at a time; see survival, failure, and hazard below.

    graph may not be specified with hazard. Use sts graph to graph estimates of the hazard function.

ci graphs the confidence intervals around survival, failure, or hazard.

level(#) specifies the confidence level, as a percentage, for confidence intervals. The default is level(95) or as set by set level; see [R] **level**.

noadjust suppresses the actuarial adjustment to the number at risk. The default is to consider the adjusted number at risk for each interval as the total at the start minus (the number of censored)/2. If noadjust is specified, the number at risk is simply the total at the start, corresponding to the standard Kaplan–Meier assumption. noadjust should be specified when using ltable to list results corresponding to those produced by sts list; see [ST] **sts list**.

tvid(*varname*) is for use with longitudinal data with time-varying parameters. Each subject appears in the data more than once, and equal values of *varname* identify observations referring to the same subject. When tvid() is specified, only the last observation on each subject is used in making the table. The order of the data does not matter, and *last* here means the last observation chronologically.

intervals(w | *numlist*) specifies the intervals into which the data are to be aggregated for tabular presentation. A numeric argument is interpreted as the width of the interval. For instance, interval(2) aggregates data into the intervals $0 \le t < 2$, $2 \le t < 4$, and so on. Not specifying interval() is equivalent to specifying interval(1). Because in most data, failure times are

recorded as integers, this amounts to no aggregation except that implied by the recording of the time variable, and so it produces Kaplan–Meier product-limit estimates of the survival curve (with an actuarial adjustment; see the `noadjust` option above). Also see [ST] **sts list**. Although it is possible to examine survival and failure without aggregation, some form of aggregation is almost always required to examine the hazard.

When more than one argument is specified, intervals are aggregated as specified. For instance, `interval(0,2,8,16)` aggregates data into the intervals $0 \le t < 2$, $2 \le t < 8$, and $8 \le t < 16$, and (if necessary) the open-ended interval $t \ge 16$.

`interval(w)` is equivalent to `interval(0,7,15,30,60,90,180,360,540,720)`, corresponding to 1 week, (roughly) 2 weeks, 1 month, 2 months, 3 months, 6 months, 1 year, 1.5 years, and 2 years when failure times are recorded in days. The `w` suggests widening intervals.

saving(*filename* [, replace]) creates a Stata data file (`.dta` file) containing the life table. This option will not save the graph to disk; see [G-2] **graph save** to save the resulting graph to disk.

> replace specifies that *filename* be overwritten if it exists. This option is not shown in the dialog box.

> ┌─ Plot ───

plotopts(*plot_options*) affects the rendition of the plotted line and plotted points; see [G-3] *connect_options* and [G-3] *marker_options*.

plot#opts(*plot_options*) affects the rendition of the #th plotted line and plotted points; see [G-3] *connect_options* and [G-3] *marker_options*. This option is valid only if `overlay` is specified.

> ┌─ CI plot ──

ciopts(*rspike_options*) affects the rendition of the confidence intervals for the graphed survival, failure, or hazard; see [G-3] *rspike_options*.

ci#opts(*rspike_options*) affects the rendition of the #th confidence interval for the graphed survival, failure, or hazard; see [G-3] *rspike_options*. This option is valid only if `overlay` is specified.

> ┌─ Add plots ──

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] *addplot_option*.

> ┌─ Y axis, X axis, Titles, Legend, and Overall ──────────────────────────────

twoway_options are any of the options documented in [G-3] *twoway_options*, excluding by(). These include options for titling the graph (see [G-3] *title_options*) and for saving the graph to disk (see [G-3] *saving_option*).

byopts(*byopts*) affects the appearance of the combined graph when by() is specified, including the overall graph title and the organization of subgraphs. See [G-3] *by_option*.

Remarks and examples

Life tables describe deathrates in a given population over time. Such tables date back to the 17th century. John Graunt (1662) and Edmund Halley (1693) are often credited with their development, but there is a dispute about how they shared in the development. `ltable` is for use with "cohort" data, and although one often thinks of such tables as monitoring a population from the "birth" of the first member to the "death" of the last, more generally, such tables can be thought of as a reasonable way to list any kind of survival data. For an introductory discussion of life tables, see Pagano and

Gauvreau (2022, 481–486) and Oliveira (2013); for an intermediate discussion, see Selvin (2004, 335–377); and for a more complete discussion, see Chiang (1984).

John Graunt (1620–1674) was born in London. He learned the haberdashery trade through an apprenticeship with his father and later managed his own business. He served as a councilman and as captain and major of the Trained Bands, a militia in charge of protecting England.

Graunt studied death records maintained by London parishes and published his findings in a book in 1662. In this book, he categorized deathrates by the cause of death and observed differences across regions and gender. Graunt was the first to point out that the differences in birthrates and deathrates across gender resulted in a population that had nearly equal proportions of males and females. He divided the population into age groups and listed their casualties and predicted life expectancy, thus creating the first life table known. His work motivated Charles II of England to nominate Graunt as a member of the Royal Society in 1662 and established Graunt as the founder of demography. His membership in this society stood out because he was a tradesman, not a scientist. In 1665, the third edition of his book was published.

Graunt later became governor of the company responsible for London's water supply, weeks after the Great Fire of London. The fire destroyed Graunt's business and over 13,000 homes. Graunt died of jaundice in 1674. His methodical analysis and awareness of the shortcomings of mortality data at that time will not be forgotten.

Edmond Halley (1656–1742) was born in London. He was a pioneer in astronomy, although he made many contributions in other fields as well. Halley attended the Queen's College in the University of Oxford but left in 1676 to Saint Helena, where he mapped 341 stars of the southern hemisphere. At 22 years old, he published his findings from the Saint Helena voyage, obtained his degree from Oxford, and was elected fellow of the Royal Society.

In 1693, he published his findings on mortality data for the city of Breslau, current day Wrocław. In this paper, Halley estimated the population size of Breslau, using the number of births and deaths in the city. He created a life table to calculate annuity prices, which would prove useful for future actuaries but did not impact the way annuities were priced at the time, independent of the buyer's age.

In 1705, Halley published a paper in which he applied his theory of elliptical orbits, predicting that the comet he had observed in 1682 would return in December 1758. He did not live to see it, but the comet was observed in December 1758 and consequently named Halley's comet. Some other major contributions include calculating the distance from the Earth to the sun, detecting the motion of stars, and using lunar observations to determine the longitude at sea. Halley also published the first meteorological chart, designed a diving bell, and introduced a root-finding algorithm, now called Halley's method.

▷ Example 1

In Pike (1966), two groups of rats were exposed to a carcinogen, and the number of days to death from vaginal cancer was recorded (reprinted in Kalbfleisch and Prentice 2002, 2):

Group 1	143	164	188	188	190	192	206	209	213	216
	220	227	230	234	246	265	304	216*	244*	
Group 2	142	156	163	198	205	232	232	233	233	233
	233	239	240	261	280	280	296	296	323	204*
	344*									

The '*' on a few of the entries indicates that the observation was censored—as of the recorded day, the rat had still not died because of vaginal cancer but was withdrawn from the experiment for other reasons.

Having entered these data into Stata, we see that the first few observations are

```
. use https://www.stata-press.com/data/r18/rat
. list in 1/5
```

	group	t	died
1.	1	143	1
2.	1	164	1
3.	1	188	1
4.	1	188	1
5.	1	190	1

For example, the first observation records a rat from group 1 that died on the 143rd day. The `died` variable records whether that rat died or was withdrawn (censored):

```
. list if died==0
```

	group	t	died
18.	1	216	0
19.	1	244	0
39.	2	204	0
40.	2	344	0

Four rats, two from each group, did not die but were withdrawn.

The life table for group 1 is

```
. ltable t died if group==1
```

Interval		Beg. total	Deaths	Lost	Survival	Std. error	[95% conf. int.]	
143	144	19	1	0	0.9474	0.0512	0.6812	0.9924
164	165	18	1	0	0.8947	0.0704	0.6408	0.9726
188	189	17	2	0	0.7895	0.0935	0.5319	0.9153
190	191	15	1	0	0.7368	0.1010	0.4789	0.8810
192	193	14	1	0	0.6842	0.1066	0.4279	0.8439
206	207	13	1	0	0.6316	0.1107	0.3790	0.8044
209	210	12	1	0	0.5789	0.1133	0.3321	0.7626
213	214	11	1	0	0.5263	0.1145	0.2872	0.7188
216	217	10	1	1	0.4709	0.1151	0.2410	0.6713
220	221	8	1	0	0.4120	0.1148	0.1937	0.6194
227	228	7	1	0	0.3532	0.1125	0.1502	0.5648
230	231	6	1	0	0.2943	0.1080	0.1105	0.5070
234	235	5	1	0	0.2355	0.1012	0.0751	0.4459
244	245	4	0	1	0.2355	0.1012	0.0751	0.4459
246	247	3	1	0	0.1570	0.0931	0.0312	0.3721
265	266	2	1	0	0.0785	0.0724	0.0056	0.2864
304	305	1	1	0	0.0000	.	.	.

The reported survival rates are the survival rates at the end of the interval. Thus, 94.7% of rats survived 144 days or more.

◁

❑ Technical note

If you compare the table just printed with the corresponding table in Kalbfleisch and Prentice (2002, 16), you will notice that the survival estimates differ beginning with the interval 216–217, which is the first interval containing a censored observation. ltable treats censored observations as if they were withdrawn halfway through the interval. The table printed in Kalbfleisch and Prentice treated censored observations as if they were withdrawn at the end of the interval, even though Kalbfleisch and Prentice (2002, 19) mention how results could be adjusted for censoring.

Here the same results as those printed in Kalbfleisch and Prentice could be obtained by incrementing the time of withdrawal by 1 for the four censored observations. We say "here" because there were no deaths on the incremented dates. For instance, one of the rats was withdrawn on the 216th day, a day on which there was also a real death. There were no deaths on day 217, however, so moving the withdrawal forward 1 day is equivalent to assuming that the withdrawal occurred at the end of the day 216–217 interval. If the adjustments are made and ltable is used to calculate survival in both groups, the results are the same as those printed in Kalbfleisch and Prentice, except that for group 2 in the interval 240–241, they report the survival as 0.345 when they mean 0.354.

In any case, the one-half adjustment for withdrawals is generally accepted, but it is only a crude adjustment that becomes cruder the wider the intervals.

❑

▷ Example 2: ltable with aggregated intervals

When you do not specify the intervals, ltable uses unit intervals. The only aggregation performed on the data was aggregation due to deaths or withdrawals occurring on the same "day". If we wanted to see the table aggregated into 30-day intervals, we would type

```
. ltable t died if group==1, interval(30)
```

Interval		Beg. total	Deaths	Lost	Survival	Std. error	[95% conf. int.]	
120	150	19	1	0	0.9474	0.0512	0.6812	0.9924
150	180	18	1	0	0.8947	0.0704	0.6408	0.9726
180	210	17	6	0	0.5789	0.1133	0.3321	0.7626
210	240	11	6	1	0.2481	0.1009	0.0847	0.4552
240	270	4	2	1	0.1063	0.0786	0.0139	0.3090
300	330	1	1	0	0.0000	.	.	.

The interval displayed as 120 150 indicates the interval including 120 and up to, but not including, 150. The reported survival rate is the survival rate just after the close of the interval.

When you specify more than one number as the argument to interval(), you specify the cutoff points, not the widths.

```
. ltable t died if group==1, interval(120,180,210,240,330)
```

Interval		Beg. total	Deaths	Lost	Survival	Std. error	[95% conf. int.]	
120	180	19	2	0	0.8947	0.0704	0.6408	0.9726
180	210	17	6	0	0.5789	0.1133	0.3321	0.7626
210	240	11	6	1	0.2481	0.1009	0.0847	0.4552
240	330	4	3	1	0.0354	0.0486	0.0006	0.2245

If any of the underlying failure or censoring times are larger than the last cutoff specified, then they are treated as being in the open-ended interval:

```
. ltable t died if group==1, interval(120,180,210,240)
```

Interval		Beg. total	Deaths	Lost	Survival	Std. error	[95% conf. int.]	
120	180	19	2	0	0.8947	0.0704	0.6408	0.9726
180	210	17	6	0	0.5789	0.1133	0.3321	0.7626
210	240	11	6	1	0.2481	0.1009	0.0847	0.4552
240	.	4	3	1	0.0354	0.0486	0.0006	0.2245

Whether the last interval is treated as open ended or not makes no difference for survival and failure tables, but it does affect hazard tables. If the interval is open ended, the hazard is not calculated for it.

◁

▷ Example 3: ltable with separate tables for each group

The by(*varname*) option specifies that separate tables be presented for each value of *varname*. Remember that our rat dataset contains two groups:

```
. ltable t died, by(group) interval(30)
```

Interval		Beg. total	Deaths	Lost	Survival	Std. error	[95% conf. int.]	
group = 1								
120	150	19	1	0	0.9474	0.0512	0.6812	0.9924
150	180	18	1	0	0.8947	0.0704	0.6408	0.9726
180	210	17	6	0	0.5789	0.1133	0.3321	0.7626
210	240	11	6	1	0.2481	0.1009	0.0847	0.4552
240	270	4	2	1	0.1063	0.0786	0.0139	0.3090
300	330	1	1	0	0.0000	.	.	.
group = 2								
120	150	21	1	0	0.9524	0.0465	0.7072	0.9932
150	180	20	2	0	0.8571	0.0764	0.6197	0.9516
180	210	18	2	1	0.7592	0.0939	0.5146	0.8920
210	240	15	7	0	0.4049	0.1099	0.1963	0.6053
240	270	8	2	0	0.3037	0.1031	0.1245	0.5057
270	300	6	4	0	0.1012	0.0678	0.0172	0.2749
300	330	2	1	0	0.0506	0.0493	0.0035	0.2073
330	360	1	0	1	0.0506	0.0493	0.0035	0.2073

◁

▷ Example 4: ltable for failure tables

A failure table is simply a different way of looking at a survival table; failure is $1 -$ survival:

```
. ltable t died if group==1, interval(30) failure
```

| | | Beg. | | | Cum. | Std. | | |
Interval		Total	Deaths	Lost	Failure	Error	[95% Conf. Int.]	
120	150	19	1	0	0.0526	0.0512	0.0076	0.3188
150	180	18	1	0	0.1053	0.0704	0.0274	0.3592
180	210	17	6	0	0.4211	0.1133	0.2374	0.6679
210	240	11	6	1	0.7519	0.1009	0.5448	0.9153
240	270	4	2	1	0.8937	0.0786	0.6910	0.9861
300	330	1	1	0	1.0000	.	.	.

◁

▷ Example 5: Survival rate at start of interval versus end of interval

Selvin (2004, 357) presents follow-up data from Cutler and Ederer (1958) on six cohorts of kidney cancer patients. The goal is to estimate the 5-year survival probability.

Year	Interval	Alive	Deaths	Lost	With-drawn	Year	Interval	Alive	Deaths	Lost	With-drawn
1946	0–1	9	4	1		1948	0–1	21	11	0	
	1–2	4	0	0			1–2	10	1	2	
	2–3	4	0	0			2–3	7	0	0	
	3–4	4	0	0			3–4	7	0	0	7
	4–5	4	0	0		1949	0–1	34	12	0	
	5–6	4	0	0	4		1–2	22	3	3	
1947	0–1	18	7	0			2–3	16	1	0	15
	1–2	11	0	0		1950	0–1	19	5	1	
	2–3	11	1	0			1–2	13	1	1	11
	3–4	10	2	2		1951	0–1	25	8	2	15
	4–5	6	0	0	6						

The following is the Stata dataset corresponding to the table:

```
. use https://www.stata-press.com/data/r18/selvin
. list
```

	year	t	died	pop
1.	1946	.5	1	4
2.	1946	.5	0	1
3.	1946	5.5	0	4
4.	1947	.5	1	7
5.	1947	2.5	1	1

(output omitted)

As summary data may often come in the form shown above, it is worth understanding exactly how the data were translated for use with `ltable`. `t` records the time of death or censoring (lost to follow-up or withdrawal). `died` contains 1 if the observation records a death and 0 if it instead records lost or withdrawn patients. `pop` records the number of patients in the category. The first line of the original table stated that, in the 1946 cohort, there were nine patients at the start of the interval $0-1$, and during the interval, four died and one was lost to follow-up. Thus we entered in observation 1 that at `t` $= 0.5$, four patients died and in observation 2 that at `t` $= 0.5$, one patient was censored. We ignored the information on the total population because `ltable` will figure that out for itself.

The second line of the table indicated that in the interval 1−2, four patients were still alive at the beginning of the interval, and during the interval, zero died or were lost to follow-up. Because no patients died or were censored, we entered nothing into our data. Similarly, we entered nothing for lines 3, 4, and 5 of the table. The last line for 1946 stated that, in the interval 5−6, four patients were alive at the beginning of the interval and that those four patients were withdrawn. In observation 3, we entered that there were four censorings at t = 5.5.

It does not matter that we chose to record the times of deaths or censoring as midpoints of intervals; we could just as well have recorded the times as 0.8 and 5.8. By default, ltable will form intervals 0−1, 1−2, and so on, and place observations into the intervals to which they belong. We suggest using 0.5 and 5.5 because those numbers correspond to the underlying assumptions made by ltable in making its calculations. Using midpoints reminds you of these assumptions.

To obtain the survival rates, we type

```
. ltable t died [fweight=pop]
                    Beg.                               Std.
    Interval       total   Deaths   Lost   Survival    error     [95% conf. int.]

       0     1       126      47     19     0.5966     0.0455    0.5017     0.6792
       1     2        60       5     17     0.5386     0.0479    0.4405     0.6269
       2     3        38       2     15     0.5033     0.0508    0.4002     0.5977
       3     4        21       2      9     0.4423     0.0602    0.3225     0.5554
       4     5        10       0      6     0.4423     0.0602    0.3225     0.5554
       5     6         4       0      4     0.4423     0.0602    0.3225     0.5554
```

We estimate the 5-year survival rate as 0.4423 and the 95% confidence interval as 0.3225 to 0.5554.

Selvin (2004, 361), in presenting these results, lists the survival in the interval 0−1 as 1, in 1−2 as 0.597, in 2−3 as 0.539, and so on. That is, relative to us, he shifted the rates down one row and inserted a 1 in the first row. In his table, the survival rate is the survival rate at the *start* of the interval. In our table, the survival rate is the survival rate at the *end* of the interval (or, equivalently, at the start of the next interval). This is, of course, simply a difference in the way the numbers are presented and not in the numbers themselves.

◁

▷ Example 6: ltable for hazard tables

The discrete hazard function is the rate of failure—the number of failures occurring within a time interval divided by the width of the interval (assuming that there are no censored observations). Although the survival and failure tables are meaningful at the "individual" level—with intervals so narrow that each contains only one failure—that is not true for the discrete hazard. If all intervals contained one death and if all intervals were of equal width, the hazard function would be $1/\Delta t$ and so appear to be a constant!

The empirically determined discrete hazard function can be revealed only by aggregation. Gross and Clark (1975, 37) print data on malignant melanoma at the University of Texas M. D. Anderson Tumor Clinic between 1944 and 1960. The interval is the time from initial diagnosis:

Interval (years)	Number lost to follow-up	Number with-drawn alive	Number dying
0–1	19	77	312
1–2	3	71	96
2–3	4	58	45
3–4	3	27	29
4–5	5	35	7
5–6	1	36	9
6–7	0	17	3
7–8	2	10	1
8–9	0	8	3
9+	0	0	32

For our statistical purposes, there is no difference between the number lost to follow-up (patients who disappeared) and the number withdrawn alive (patients dropped by the researchers)—both are censored. We have entered the data into Stata; here are a few of the data:

```
. use https://www.stata-press.com/data/r18/tumor
. list in 1/6, separator(0)
```

	t	d	pop
1.	.5	Death	312
2.	.5	Censoring	19
3.	.5	Censoring	77
4.	1.5	Death	96
5.	1.5	Censoring	3
6.	1.5	Censoring	71

We entered each group's time of death or censoring as the midpoint of the intervals and entered the numbers of the table, recording d as 1 for deaths and 0 for censoring. The hazard table is

```
. ltable t d [fweight=pop], hazard interval(0(1)9)
```

Interval		Beg. Total	Cum. Failure	Std. Error	Hazard	Std. Error	[95% Conf. Int.]	
0	1	913	0.3607	0.0163	0.4401	0.0243	0.3924	0.4877
1	2	505	0.4918	0.0176	0.2286	0.0232	0.1831	0.2740
2	3	335	0.5671	0.0182	0.1599	0.0238	0.1133	0.2064
3	4	228	0.6260	0.0188	0.1461	0.0271	0.0931	0.1991
4	5	169	0.6436	0.0190	0.0481	0.0182	0.0125	0.0837
5	6	122	0.6746	0.0200	0.0909	0.0303	0.0316	0.1502
6	7	76	0.6890	0.0208	0.0455	0.0262	0.0000	0.0969
7	8	56	0.6952	0.0213	0.0202	0.0202	0.0000	0.0598
8	9	43	0.7187	0.0235	0.0800	0.0462	0.0000	0.1705
9	.	32	1.0000	.	.	.	.	.

We specified the `interval()` option as we did—and not as `interval(1)` or omitting the option altogether—to force the last interval to be open ended. Had we not, and if we had recorded t as 9.5 for observations in that interval (as we did), `ltable` would have calculated a hazard rate for the "interval". Here the result of that calculation would have been 2, but no matter the result, it would have been meaningless because we do not know the width of the interval.

When dealing with the survivor or failure function, you are not limited to merely examining a column of numbers. With the `graph` option, you can see the result graphically:

```
. ltable t d [fweight=pop], i(0(1)9) graph notable ci xlab(0(2)10)
```
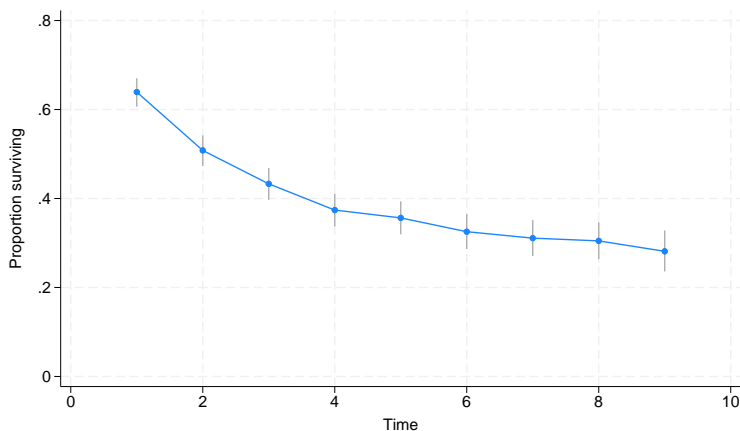

The vertical lines in the graph represent the 95% confidence intervals for the survivor function. Among the options we specified, although it is not required, is `notable`, which suppressed printing the table, saving us some paper. `xlab()` was passed through to the `graph` command (see [G-3] *twoway_options*) and was unnecessary but made the graph look better.

◁

❑ Technical note

Because many intervals can exist during which no failures occur (in which case the hazard estimate is zero), the estimated hazard is best graphically represented using a kernel smooth. Such an estimate is available in `sts graph`; see [ST] **sts graph**.

❑

Video example

How to construct life tables

Methods and formulas

Let τ_i be the individual failure or censoring times. The data are aggregated into intervals given by t_j, $j = 1, \ldots, J$, and $t_{J+1} = \infty$ with each interval containing counts for $t_j \leq \tau < t_{j+1}$. Let d_j be the number of failures during the interval, m_j be the censored observations during the interval, and N_j be the number alive at the start of the interval. Define $n_j = N_j - m_j/2$ as the adjusted number at risk at the start of the interval. If the `noadjust` option is specified, $n_j = N_j$.

The product-limit estimate of the survivor function is

$$S_j = \prod_{k=1}^{j} \frac{n_k - d_k}{n_k}$$

([Kalbfleisch and Prentice 2002](), 10, 15). Greenwood's formula for the asymptotic standard error of S_j is

$$s_j = S_j \sqrt{\sum_{k=1}^{j} \frac{d_k}{n_k(n_k - d_k)}}$$

([Greenwood 1926](); [Kalbfleisch and Prentice 2002](), 17). s_j is reported as the standard deviation of survival but is not used in generating the confidence intervals because it can produce intervals outside 0 and 1. The "natural" units for the survivor function are $\log(-\log S_j)$, and the asymptotic standard error of that quantity is

$$\widehat{s}_j = \sqrt{\frac{\sum d_k/\{n_k(n_k - d_k)\}}{\left[\sum \log\{(n_k - d_k)/n_k\}\right]^2}}$$

([Kalbfleisch and Prentice 2002](), 18). The corresponding confidence intervals are $S_j^{\exp(\pm z_{1-\alpha/2}\widehat{s}_j)}$.

The cumulative failure time is defined as $G_j = 1 - S_j$, and thus the variance is the same as for S_j and the confidence intervals are $1 - S_j^{\exp(\pm z_{1-\alpha/2}\widehat{s}_j)}$.

Both S_j and G_j are graphed against t_{j+1}.

Define the within-interval failure rate as $f_j = d_j/n_j$. The maximum likelihood estimate of the (within-interval) hazard is then

$$\lambda_j = \frac{f_j}{(1 - f_j/2)(t_{j+1} - t_j)}$$

The standard error of λ_j is

$$s_{\lambda_j} = \lambda_j \sqrt{\frac{1 - \{(t_{j+1} - t_j)\lambda_j/2\}^2}{d_j}}$$

from which a confidence interval is calculated.

If the `noadjust` option is specified, the estimate of the hazard is

$$\lambda_j = \frac{f_j}{t_{j+1} - t_j}$$

and its standard error is

$$s_{\lambda_j} = \frac{\lambda_j}{\sqrt{d_j}}$$

The confidence interval is

$$\left[\frac{\lambda_j}{2d_j}\chi^2_{2d_j,\alpha/2}, \ \frac{\lambda_j}{2d_j}\chi^2_{2d_j,1-\alpha/2} \right]$$

where $\chi^2_{2d_j,q}$ is the qth quantile of the χ^2 distribution with $2d_j$ degrees of freedom ([Cox and Oakes 1984](), 53–54, 38–40).

For the likelihood-ratio test for homogeneity, let d_g be the total number of deaths in the gth group. Define $T_g = \sum_{i \in g} \tau_i$, where i indexes the individual failure or censoring times. The χ^2 value with $G - 1$ degrees of freedom (where G is the total number of groups) is

$$\chi^2 = 2\left\{\left(\sum d_g\right) \log\left(\frac{\sum T_g}{\sum d_g}\right) - \sum d_g \log\left(\frac{T_g}{d_g}\right)\right\}$$

(Lawless 2003, 155).

The log-rank test for homogeneity is the test presented by `sts test`; see [ST] **sts**.

Acknowledgments

References

Chiang, C. L. 1984. *The Life Table and Its Applications*. Malabar, FL: Krieger.

Cox, D. R., and D. Oakes. 1984. *Analysis of Survival Data*. London: Chapman and Hall/CRC.

Cutler, S. J., and F. Ederer. 1958. Maximum utilization of the life table method in analyzing survival. *Journal of Chronic Diseases* 8: 699–712. https://doi.org/10.1016/0021-9681(58)90126-7.

Graunt, J. 1662. *Natural and Political Observations Mentioned in a following Index, and Made Upon the Bills of Mortality*. London: Thomas Roycroft, John Martyn, James Allestry, and Thomas Dicas.

Greenwood, M. 1926. The natural duration of cancer. *Reports on Public Health and Medical Subjects* 33: 1–26.

Gross, A. J., and V. A. Clark. 1975. *Survival Distribution: Reliability Applications in the Biomedical Sciences*. New York: Wiley.

Halley, E. 1693. An estimate of the degrees of the mortality of mankind, drawn from curious tables of the births and funerals at the city of Breslaw; with an attempt to ascertain the price of annuities upon lives. *Philosophical Transactions* 17: 596–610. https://doi.org/10.1098/rstl.1693.0007.

Kahn, H. A., and C. T. Sempos. 1989. *Statistical Methods in Epidemiology*. New York: Oxford University Press.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Krakauer, H., and J. Stewart. 1991. ssa1: Actuarial or life-table analysis of time-to-event data. *Stata Technical Bulletin* 1: 23–25. Reprinted in *Stata Technical Bulletin Reprints*, vol. 1, pp. 200–202. College Station, TX: Stata Press.

Lawless, J. F. 2003. *Statistical Models and Methods for Lifetime Data*. 2nd ed. New York: Wiley.

Muniz, J. O. 2020. Multistate life tables using Stata. *Stata Journal* 20: 721–745.

———. 2023. Iterative intercensal single-decrement life tables using Stata. *Stata Journal* 23: 813–834.

Oliveira, A. G. 2013. *Biostatistics Decoded*. Chichester, UK: Wiley.

Pagano, M., and K. Gauvreau. 2022. *Principles of Biostatistics*. 3rd ed. Boca Raton, FL: CRC Press.

Pike, M. C. 1966. A method of analysis of a certain class of experiments in carcinogenesis. *Biometrics* 22: 142–161. https://doi.org/10.2307/2528221.

Selvin, S. 2004. *Statistical Analysis of Epidemiologic Data*. 3rd ed. New York: Oxford University Press.

Wei, Y., and P. Royston. 2017. Reconstructing time-to-event data from published Kaplan–Meier curves. *Stata Journal* 17: 786–802.

Also see

[ST] **stcox** — Cox proportional hazards model

Title

> **snapspan —** Convert snapshot data to time-span data

Description

snapspan converts snapshot data for a given subject to time-span data required for use with survival analysis commands, such as stcox, streg, and stset. snapspan replaces the data in the specified variables. Transformed variables may be "events" that occur at the instant of the snapshot or retrospective variables that are to apply to the time span ending at the time of the current snapshot.

Quick start

Create a time-span dataset from data containing subject identifier id, event variable evar occurring at the time in tvar, and other variables measured at that time

    snapspan id tvar evar

Same as above, and create new variable time0 containing the entry time for each record

    snapspan id tvar evar, generate(time0)

Menu

Statistics > Survival analysis > Setup and utilities > Convert snapshot data to time-span data

Syntax

> snapspan *idvar* *timevar* *varlist* [, <u>g</u>enerate(*newt0var*) replace]

idvar records the subject ID and may be string or numeric.

timevar records the time of the snapshot; it must be numeric and may be recorded on any scale: date, hour, minute, second, etc.

varlist are the "event" variables, meaning that they occur at the instant of *timevar*. *varlist* can also include retrospective variables that are to apply to the time span ending at the time of the current snapshot. The other variables are assumed to be measured at the time of the snapshot and thus apply from the time of the snapshot forward. See *Specifying varlist* below.

snapspan does not allow alias variables; see [D] **frunalias** for advice on how to get around this restriction.

Options

<u>generate</u>(*newt0var*) adds *newt0var* to the dataset containing the entry time for each converted time-span record.

replace specifies that it is okay to change the data in memory, even though the dataset has not been saved on disk in its current form.

Remarks and examples

Remarks are presented under the following headings:

> *Snapshot and time-span datasets*
> *Specifying varlist*

Snapshot and time-span datasets

snapspan converts a snapshot dataset to a time-span dataset. A snapshot dataset records a subject *id*, a *time*, and then other variables measured at the *time*:

```
Snapshot datasets:

    idvar     timevar     x1    x2    ...

       47          12      5    27    ...
       47          42      5    18    ...
       47          55      5    19    ...

    idvar     datevar     x1    x2    ...

      122   14jul1998      5    27    ...
      122   12aug1998      5    18    ...
      122   08sep1998      5    19    ...

    idvar        year     x1    x2    ...

      122        1994      5    27    ...
      122        1995      5    18    ...
      122        1997      5    19    ...
```

A time-span dataset records a span of time (*time0*, *time1*]:

```
                                  some variables assumed
                                     to occur at time1
```

```
              |<── other variables assumed constant over span ──>|
              |                                                   |
   ───────────────────────────────────────────────────────────────────> time
            time0                                              time1
```
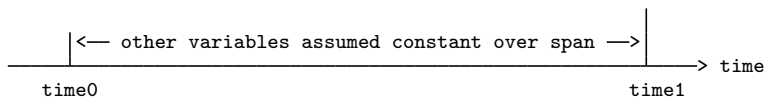
Time-span data are required, for instance, by `stset` and the st system. The variables assumed to occur at `time1` are the failure or event variables. All the other variables are assumed to be constant over the span.

```
        Time-span datasets:
            idvar       time0       time1     x1      x2    ...   event

               47           0          12      5      13    ...       0
               47          12          42      5      27    ...       0
               47          42          55      5      18    ...       1

            idvar       time0       time1     x1      x2    ...   event

              122  01jan1998  14jul1998        5      13    ...       0
              122  14jul1998  12aug1998        5      27    ...       0
              122  12aug1998  08sep1998        5      18    ...       1

            idvar       time0       time1     x1      x2    ...   event

              122        1993        1994      5      13    ...       0
              122        1994        1995      5      27    ...       0
              122        1995        1997      5      18    ...       1
```

To convert snapshot data to time-span data, you need to distinguish between event and nonevent variables. Event variables happen at an instant.

Say that you have a snapshot dataset containing variable `e` recording an event (`e` = 1 might record surgery, death, becoming unemployed, etc.) and the rest of the variables—call them `x1`, `x2`, etc.—recording characteristics (such as sex, birth date, blood pressure, or weekly wage). The same data, in snapshot and time-span form, would be

```
    In snapshot form:                In time-span form:
    id  time    x1   x2    e         id  time0  time     x1    x2     e

     1     5    a1   b1   e1          1     .      5      .     .     e1
     1     7    a2   b2   e2          1     5      7      a1    b1    e2
     1     9    a3   b3   e3          1     7      9      a2    b2    e3
     1    11    a4   b4   e4          1     9     11      a3    b3    e4
```

`snapspan` converts data from the form on the left to the form on the right:

```
            . snapspan id time e, generate(time0) replace
```

The form on the right is suitable for use by `stcox` and `stset` and the other survival analysis commands.

Specifying varlist

The *varlist*—the third variable on—specifies the "event" variables.

In fact, the *varlist* specifies the variables that apply to the time span ending at the time of the current snapshot. The other variables are assumed to be measured at the time of the snapshot and thus apply from the time of the snapshot forward.

Thus *varlist* should include retrospective variables.

For instance, say that the snapshot recorded bp, blood pressure; smokes, whether the patient smoked in the last 2 weeks; and event, a variable recording examination, surgery, etc. Then *varlist* should include smokes and event. The remaining variables, bp and the rest, would be assumed to apply from the time of the snapshot forward.

Suppose that the snapshot recorded ecs, employment change status (hired, fired, promoted, etc.); wage, the current hourly wage; and ms, current marital status. Then *varlist* should include esc and ms (assuming snapshot records are not generated for reason of ms change). The remaining variables, wage and the rest, would be assumed to apply from the time of the snapshot forward.

Also see

[ST] **stset** — Declare data to be survival-time data

[D] **frunalias** — Change storage type of alias variables

Title

st — Survival-time data

Description   Also see

Description

The term *st* refers to survival-time data and the commands—most of which begin with the letters st—for analyzing these data. If you have data on individual subjects with observations recording that a particular subject came under observation at time t_0 and that later, at t_1, a failure was observed, you have what we call uncensored survival-time data. If you have data on individual subjects with observations recording that a particular subject came under observation at time t_0 and that later, at t_1, a censoring was observed, you have right-censored survival-time data. If you have data on individual subjects with observations recording that a particular subject was observed at time t_0, but a failure already occurred by that time, you have left-censored survival-time data. If you have data on individual subjects with observations recording that a particular subject failed sometime between times t_l and t_u, you have interval-censored survival-time data. And, of course, you may have data that contain observations of all the above types.

If you have subject-specific data, with observations recording not a span of time, but measurements taken on the subject at that point in time, you have what we call a snapshot dataset; see [ST] **snapspan**.

If you have data on populations, with observations recording the number of units under test at time t (subjects alive) and the number of subjects that failed or were lost because of censoring, you have what we call count-time data; see [ST] **ct**.

st commands	Description
stset	Declare data to be survival-time data
stdescribe	Describe survival-time data
stsum	Summarize survival-time data
stvary	Report variables that vary over time
stfill	Fill in by carrying forward values of covariates
stgen	Generate variables reflecting entire histories
stsplit	Split time-span records
stjoin	Join time-span records
stbase	Form baseline dataset
sts	Generate, graph, list, and test the survivor and related functions
stir	Report incidence-rate comparison
stci	Confidence intervals for means and percentiles of survival time
strate	Tabulate failure rate
stptime	Calculate person-time, incidence rates, and SMR
stmh	Calculate rate ratios with the Mantel–Haenszel method
stmc	Calculate rate ratios with the Mantel–Cox method

stcox	Fit Cox proportional hazards model
estat concordance	Compute the concordance probability
estat phtest	Test Cox proportional-hazards assumption
stphplot	Graphically assess the Cox proportional-hazards assumption
stcoxkm	Graphically assess the Cox proportional-hazards assumption
streg	Fit parametric survival models
stintreg	Fit parametric survival models for interval-censored data
estat gofplot	Graphically assess goodness of fit after streg, stcox, and stintreg
stintcox	Fit Cox proportional hazards model for interval-censored data
stintphplot	Graphically assess the Cox proportional-hazards assumption for interval-censored data
stintcoxnp	Graphically assess the Cox proportional-hazards assumption for interval-censored data
stcrreg	Fit competing-risks regression models
xtstreg	Fit random-effects parametric survival models
mestreg	Fit mixed-effects parametric survival models
stcurve	Plot the survivor or related function after streg, stcox, and more
stteffects	Estimate treatment effects using observational data
sttocc	Convert survival-time data to case–control data
sttoct	Convert survival-time data to count-time data
st_*	Survival analysis subroutines for programmers
fmm: streg	Finite mixtures of parametric survival models
bayes: streg	Bayesian parametric survival models
bayes: mestreg	Bayesian multilevel parametric survival models

The st commands are used for analyzing time-to-absorbing-event (single-failure) data and for analyzing time-to-repeated-event (multiple-failure) data.

For uncensored and right-censored data, you begin an analysis by stsetting your data, which tells Stata the key survival-time variables; see [ST] stset. Once you have stset your data, you can use the other st commands. If you save your data after stsetting it, you will not have to stset it again in the future; Stata will remember.

The stintcox and stintreg commands are designed for the analysis of general interval-censored data, including right-, left-, and interval-censored observations. It does not require stsetting the data.

The subsequent st entries are printed in this manual in alphabetical order. You can skip around, but if you want to be an expert on all of Stata's survival analysis capabilities, we suggest the reading order listed above.

Also see

[ST] **ct** — Count-time data

[ST] **snapspan** — Convert snapshot data to time-span data

[ST] **stset** — Declare data to be survival-time data

[ST] **Survival analysis** — Introduction to survival analysis commands

[ST] **Glossary**

Title

> **st_is —** Survival analysis subroutines for programmers

Description

These commands are provided for programmers wishing to write new st commands.

st_is verifies that the data in memory are survival-time (st) data. If not, it issues the error message "data not st", r(119).

st is currently "release 2", meaning that this is the second design of the system. Programs written for the previous release continue to work. (The previous release of st corresponds to Stata 5.)

Modern programs code st_is 2 full or st_is 2 analysis. st_is 2 verifies that the dataset in memory is in release 2 format; if it is in the earlier format, it is converted to release 2 format. (Older programs simply code st_is. This verifies that no new features are stset about the data that would cause the old program to break.)

The full and analysis parts indicate whether the dataset may include past, future, or past and future data. Code st_is 2 full if the command is suitable for running on the analysis sample and the past and future data (many data management commands fall into this category). Code st_is 2 analysis if the command is suitable for use only with the analysis sample (most statistical commands fall into this category). See [ST] **stset** for the definitions of past and future.

st_show displays the summary of the survival-time variables or does nothing, depending on what you specify when stsetting the data. noshow requests that st_show display nothing.

st_ct is a low-level utility that provides risk-group summaries from survival-time data.

Syntax

Verify that data in memory are survival-time data

> st_is 2 {full | analysis}

Display or do not display summary of survival-time variables

> st_show [noshow]

Risk-group summaries

> st_ct "[*byvars*]" -> *newvar newpopvar newfailvar* [*newcensvar* [*newentvar*]]

You must have stset your data before using st_is, st_show, and st_ct; see [ST] **stset**.

Remarks and examples

Remarks are presented under the following headings:

Definitions of characteristics and st variables

From a programmer's perspective, st is a set of conventions that specify where certain pieces of information are stored and how that information should be interpreted, together with a few subroutines that make it easier to follow the conventions.

At the lowest level, st is nothing more than a set of Stata characteristics that programmers may access:

```
char _dta[_dta]       st (marks that the data are st)
char _dta[st_ver]     2 (version number)
char _dta[st_id]      varname or nothing; id() variable
char _dta[st_bt0]     varname or nothing; t0() variable
char _dta[st_bt]      varname; t variable from stset t, ...
char _dta[st_bd]      varname or nothing; failure() variable
char _dta[st_ev]      list of numbers or nothing; numlist from failure(varname[==numlist])
char _dta[st_enter]   contents of enter() or nothing; numlist expanded
char _dta[st_exit]    contents of exit() or nothing; numlist expanded
char _dta[st_orig]    contents of origin() or nothing; numlist expanded
char _dta[st_bs]      # or 1; scale() value
char _dta[st_o]       _origin or #
char _dta[st_s]       _scale or #
char _dta[st_ifexp]   exp or nothing; from stset ... if exp ...
char _dta[st_if]      exp or nothing; contents of if()
char _dta[st_ever]    exp or nothing; contents of ever()
char _dta[st_never]   exp or nothing; contents of never()
char _dta[st_after]   exp or nothing; contents of after()
char _dta[st_befor]   exp or nothing; contents of before()
char _dta[st_wt]      weight type or nothing; user-specified weight
char _dta[st_wv]      varname or nothing; user-specified weighting variable
char _dta[st_w]       [weighttype=weightvar] or nothing
char _dta[st_show]    noshow or nothing
char _dta[st_t]       _t (for compatibility with release 1)
char _dta[st_t0]      _t0 (for compatibility with release 1)
char _dta[st_d]       _d (for compatibility with release 1)
char _dta[st_n0]      # or nothing; number of st notes
char _dta[st_n1]      text of first note or nothing
char _dta[st_n2]      text of second note or nothing
char _dta[st_set]     text or nothing. If filled in, streset (see [ST] stset) will refuse
                      to execute and present this text as the reason
```

All st datasets also have the following four variables:

> _t0   Time of entry (in t units) into risk pool
> _t   Time of exit (in t units) from risk pool
> _d   1 if failure, 0 if censoring
> _st   1 if observation is to be used and 0 otherwise

Thus, in a program, you might code

```
display "the failure/censoring base time variable is _t"
display "and its mean in the uncensored subsample is"
summarize _t if _d
```

No matter how simple or complicated the data, these four variables exist and are filled in. For instance, in simple data, _t0 might contain 0 for every observation, and _d might always contain 1.

Some st datasets also contain the variables

> _origin   Evaluated value of origin()
> _scale   Evaluated value of scale()

The _dta[st_o] characteristic contains either the name _origin or a number, often 0. It contains a number when the origin does not vary across observations. _dta[st_s] works the same way with the scale() value. Thus the origin and scale are _dta[st_o] and _dta[st_s]. In fact, these characteristics are seldom used because variables _t and _t0 are already adjusted.

Some st datasets have an id() variable that clusters together records on the same subject. The name of the variable varies, and the name can be obtained from the _dta[st_id] characteristic. If there is no id() variable, the characteristic contains nothing.

Outline of an st command

If you are writing a new st command, place st_is near the top of your code to ensure that your command does not execute on inappropriate data. Also place st_show following the parsing of your command's syntax to display the key st variables. The minimal outline for an st command is

```
program st name
        version 18.0
        st_is 2 ...
        ... syntax command ...
        ... determined there are no syntax errors ...
        st_show
        ... guts of program ...
end
```

st_is 2 appears even before the input is parsed. This is to avoid irritating users when they type a command, get a syntax error, work hard to eliminate the error, and then learn that "data not st".

A fuller outline for an st command, particularly one that performs analysis on the data, is

```
program st name
        version 18.0
        st_is 2 ...
        syntax ... [, ... noSHow ... ]
        st_show 'show'
        marksample touse
        quietly replace 'touse' = 0 if _st==0
        ... guts of program ...
end
```

All calculations and actions are to be restricted, at the least, to observations for which $\_st \neq 0$. Observations with $\_st = 0$ are to be ignored.

Using the st_ct utility

st_ct converts the data in memory to observations containing summaries of risk groups. Consider the code

```
st_is 2 analysis
preserve
st_ct "" -> t pop die
```

Typing this would change the data in memory to contain something akin to count-time data. The transformed data would have observations containing

t	time
pop	population at risk at time t
die	number who fail at time t

There would be one record per time t, and the data would be sorted by t. The original data are discarded, which is why you should code preserve; see [P] **preserve**.

The above three lines of code could be used as the basis for calculating the Kaplan–Meier product-limit survivor-function estimate. The rest of the code is

```
keep if die
generate double hazard = die/pop
generate double km     = 1-hazard  if _n==1
replace    km     = (1-hazard)*km[_n-1] if _n>1
```

st_ct can be used to obtain risk groups separately for subgroups of the population. The code

```
st_is 2 analysis
preserve
st_ct "race sex" -> t pop die
```

would change the data in memory to contain

race	
sex	
t	time
pop	population at risk at time t
die	number who fail at time t

There would be one observation for each race–sex–t combination, and the data would be sorted by race sex t.

With this dataset, you could calculate the Kaplan–Meier product-limit survivor-function estimate for each race–sex group by coding

```
keep if die
generate double hazard = die/pop
by race sex: generate double km = 1-hazard  if _n==1
by race sex: replace          km = (1-hazard)*km[_n-1] if _n>1
```

st_ct is a convenient subroutine. The above code fragment works regardless of the complexity of the underlying survival-time data. It does not matter whether there is one record per subject, no censoring, and one failure per subject, or multiple records per subject, gaps, and recurring failures for the same subject. st_ct forms risk groups that summarize the events recorded by the data.

st_ct can provide the number of censored records and the number who enter the risk group. The code

```
st_ct "" -> t pop die cens ent
```

creates records containing

t	time
pop	population at risk at time t
die	number who fail at time t
cens	number who are censored at t (after the failures)
ent	number who enter at t (after the censorings)

As before,

```
st_ct "race sex" -> t pop die cens ent
```

would create a similar dataset with records for each race−sex group.

Comparison of st_ct with sttoct

sttoct—see [ST] **sttoct**—is related to st_ct, and in fact, sttoct is implemented in terms of st_ct. The differences between them are that

- sttoct creates ct data, meaning that the dataset is marked as being ct. st_ct merely creates a useful dataset; it does not ctset the data.

- st_ct creates a total population at-risk variable—which is useful in programming—but sttoct creates no such variable.

- sttoct eliminates thrashings—censorings and reentries of the same subject as covariates change—if there are no gaps, strata shifting, etc. st_ct does not do this. Thus, at a particular time, sttoct might show that there are two lost to censoring and none entered, whereas st_ct might show 12 censorings and 10 entries. This makes no difference in calculating the number at risk and the number who fail, which are the major ingredients in survival calculations.

- st_ct is faster.

Verifying data

As long as you code st_is at the top of your program, you need not verify the consistency of the data. That is, you need not verify that subjects do not fail before they enter, etc.

The dataset is verified when you stset it. If you make a substantive change to the data, you must rerun stset (which can be done by typing stset or streset without arguments) to reverify that all is well.

Converting data

If you write a program that converts the data from one form of st data to another, or from st data to something else, be sure to issue the appropriate stset command. For instance, a command we have written, stbase, converts the data from st to a simple cross-section in one instance. In our program, we coded stset, clear so that all other st commands would know that these are no longer st data and that making st calculations on them would be inappropriate.

Even if we had forgotten, other st programs would have found many of the key st variables missing and would have ended with a "[such-and-such] not found" error.

Also see

Title

> **stbase —** Form baseline dataset

Description Quick start Menu Syntax
Options Remarks and examples Also see

Description

stbase without the at() option converts multiple-record st data to st data with every variable set to its value at baseline, defined as the earliest time at which each subject was observed. stbase without at() does nothing to single-record st data.

stbase, at() converts single- or multiple-record st data to a cross-sectional dataset (not st data), recording the number of failures at the specified time. All variables are given their values at baseline—the earliest time at which each subject was observed. In this form, single-failure data could be analyzed by logistic regression and multiple-failure data by Poisson regression, for instance.

stbase can be used with single- or multiple-record or single- or multiple-failure st data.

Quick start

Set all variables to their values at the earliest time the subject was observed using stset data

    stbase

Create a dataset with one observation per subject, recording number of failures at time 10, with all variables set to the value at the earliest time the subject was observed

    stbase, at(10)

Menu

Statistics > Survival analysis > Setup and utilities > Form baseline dataset

56

Syntax

> stbase $\big[$ *if* $\big]$ $\big[$ *in* $\big]$ $\big[$, *options* $\big]$

options	Description
Main	
at(*#*)	convert single/multiple-record st data to cross-sectional dataset at time *#*
gap(*newvar*)	name of variable containing gap time; default is gap or gaptime
replace	overwrite current data in memory
no<u>show</u>	do not show st setting information
nopreserve	programmer's option; see *Options* below

You must stset your data before using stbase; see [ST] **stset**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

nopreserve does not appear in the dialog box.

Options

> [**Main**]

at(*#*) changes what stbase does. Without the at() option, stbase produces another related st dataset. With the at() option, stbase produces a related cross-sectional dataset.

gap(*newvar*) is allowed only with at(); it specifies the name of a new variable to be added to the data containing the amount of time the subject was not at risk after entering and before *#* as specified in at(). If gap() is not specified, the new variable will be named gap or gaptime, depending on which name does not already exist in the data.

replace specifies that it is okay to change the data in memory, even though the dataset has not been saved to disk in its current form.

noshow prevents stbase from showing the key st variables. This option is rarely used because most people type stset, show or stset, noshow to set once and for all whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

The following option is available with stbase but is not shown in the dialog box:

nopreserve is for use by programmers using stbase as a subroutine. It specifies that stbase not preserve the original dataset so that it can be restored should an error be detected or should the user press *Break*. Programmers would specify this option if, in their program, they had already preserved the original data.

Remarks and examples

Remarks are presented under the following headings:

> *stbase without the at() option*
> *stbase with the at() option*
> *Single-failure st data where all subjects enter at time 0*
> *Single-failure st data where some subjects enter after time 0*
> *Single-failure st data with gaps and perhaps delayed entry*
> *Multiple-failure st data*

stbase without the at() option

Once you type `stbase`, you may not `streset` your data, even though the data are st. `streset` will refuse to run because the data have changed, and if the original rules were reapplied, they might produce different, incorrect results. The st commands use four key variables:

_t0	the time at which the record came under observation
_t	the time at which the record left observation
_d	1 if the record left under failure, 0 otherwise
_st	whether the observation is to be used (contains 1 or 0)

These variables are adjusted by `stbase`. The _t0 and _t variables, in particular, are derived from your variables according to options you specified at the time you `stset` the data, which might include an `origin()` rule, an `entry()` rule, and the like. Once intervening observations are eliminated, those rules will not necessarily produce the same results that they did previously.

To illustrate how `stbase` works, consider multiple-record, time-varying st data, on which you have performed some analysis. You now wish to compare your results with a simpler, non-time-varying analysis. For instance, suppose that variables x1 and x2 measure blood pressure and weight, respectively, and that readings were taken at various times. Perhaps you fit the model

```
. use https://www.stata-press.com/data/r18/mfail

. stset
-> stset t, id(id) failure(d) exit(time .) noshow

Survival-time data settings

          ID variable: id
       Failure event: d!=0 & d<.
Observed time interval: (t[_n-1], t]
    Exit on or before: time .

─────────────────────────────────────────────────────────────────────
      1,734  total observations
          0  exclusions
─────────────────────────────────────────────────────────────────────
      1,734  observations remaining, representing
        926  subjects
        808  failures in multiple-failure-per-subject data
    435,855  total analysis time at risk and under observation
                                          At risk from t =          0
                                 Earliest observed entry t =          0
                                    Last observed exit t =        960
```

```
. stcox x1 x2

Iteration 0:   Log likelihood = -5034.9569
Iteration 1:   Log likelihood = -4978.4198
Iteration 2:   Log likelihood = -4978.1915
Iteration 3:   Log likelihood = -4978.1914
Refining estimates:
Iteration 0:   Log likelihood = -4978.1914

Cox regression with Breslow method for ties

No. of subjects =       926                          Number of obs =  1,734
No. of failures =       808
Time at risk    = 435,855
                                                     LR chi2(2)     = 113.53
Log likelihood = -4978.1914                          Prob > chi2    = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
x1	2.273456	.216537	8.62	0.000	1.886311	2.740059
x2	.329011	.0685638	-5.33	0.000	.2186883	.4949888

with these data. You now wish to fit that same model but this time use the values of x1 and x2 at baseline. You do this by typing

```
. stbase, replace

Converting multiple-record data to baseline data ...

Notes:
     1. No gaps.
     2. There were multiple failures or reentries after failures.
     3. Baseline data have multiple records per ID (id).
     4. All records have covariate values at baseline.

. stcox x1 x2

Iteration 0:   Log likelihood = -7886.9779
Iteration 1:   Log likelihood = -7863.9974
Iteration 2:   Log likelihood = -7863.9295
Iteration 3:   Log likelihood = -7863.9295
Refining estimates:
Iteration 0:   Log likelihood = -7863.9295

Cox regression with Breslow method for ties

No. of subjects =       926                          Number of obs =  1,734
No. of failures =     1,337
Time at risk    = 435,855
                                                     LR chi2(2)     =  46.10
Log likelihood = -7863.9295                          Prob > chi2    = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
x1	1.413195	.1107945	4.41	0.000	1.211903	1.647921
x2	.4566673	.0765272	-4.68	0.000	.3288196	.6342233

Another way you could perform the analysis is to type

```
. generate x1_0 = x1

. generate x2_0 = x2

. stfill x1_0 x2_0, baseline

. stcox x1 x2
```

See [ST] **stfill**. The method you use makes no difference, but if there were many explanatory variables, **stbase** would be easier.

stbase changes the data to record the same events but changes the values of all other variables to their values at the earliest time the subject was observed.
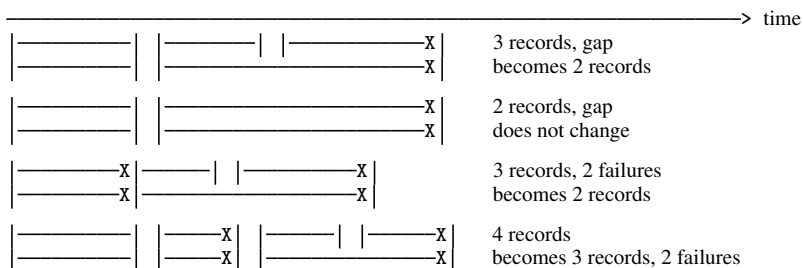
stbase also simplifies the st data where possible. Say that one of your subjects has three records in the original data and ends in a failure:

```
                                                              ──> time
|──| |──| |──X|
```
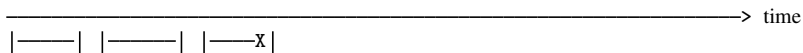
After running stbase, this subject would have one record in the data:

```
                                                              ──> time
|──| |──| |──X|
|──────────X|        <── becomes one record
```
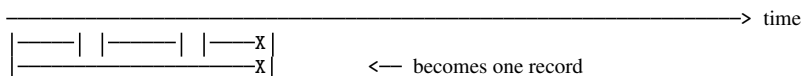
Here are some other examples of how stbase would process records with gaps and multiple failure events:

```
                                                              ──> time
|──────| |──────| |──────X|    3 records, gap
|──────────────────────X|       becomes 2 records

|──────| |──────────────X|      2 records, gap
|──────────────────────X|       does not change

|──────X|──| |──────X|          3 records, 2 failures
|──────X|──────────────X|        becomes 2 records

|──────| |────X| |──| |────X|   4 records
|──────| |────X|──────────X|     becomes 3 records, 2 failures
```

The following example shows numerically what is shown in the diagram above.

```
. use https://www.stata-press.com/data/r18/stbasexmpl, clear
. list, sepby(id)
```

	id	time0	time	wgt	death
1.	1	0	2	114	0
2.	1	3	5	110	0
3.	1	5	11	118	1
4.	2	0	2	120	0
5.	2	3	11	111	1
6.	3	0	2	108	1
7.	3	2	4	105	0
8.	3	4	7	113	1
9.	4	0	2	98	0
10.	4	3	4	101	1
11.	4	5	6	106	0
12.	4	6	11	104	1

```
. stset time, id(id) fail(death) time0(time0) exit(time .)
```

Survival-time data settings

```
             ID variable: id
          Failure event: death!=0 & death<.
Observed time interval: (time0, time]
     Exit on or before: time .
```
_____

```
          12  total observations
           0  exclusions
```
_____

```
          12  observations remaining, representing
           4  subjects
           6  failures in multiple-failure-per-subject data
          36  total analysis time at risk and under observation
                                                 At risk from t =          0
                                          Earliest observed entry t =          0
                                            Last observed exit t =         11
```

```
. list, sepby(id)
```

	id	time0	time	wgt	death	_st	_d	_t	_t0
1.	1	0	2	114	0	1	0	2	0
2.	1	3	5	110	0	1	0	5	3
3.	1	5	11	118	1	1	1	11	5
4.	2	0	2	120	0	1	0	2	0
5.	2	3	11	111	1	1	1	11	3
6.	3	0	2	108	1	1	1	2	0
7.	3	2	4	105	0	1	0	4	2
8.	3	4	7	113	1	1	1	7	4
9.	4	0	2	98	0	1	0	2	0
10.	4	3	4	101	1	1	1	4	3
11.	4	5	6	106	0	1	0	6	5
12.	4	6	11	104	1	1	1	11	6

```
. stbase, replace
```

```
         Failure _d: death
   Analysis time _t: time
  Exit on or before: time .
        ID variable: id
```

Converting multiple-record data to baseline data ...

Notes:
```
    1. There were gaps.
    2. There were multiple failures or reentries after failures.
    3. Baseline data have multiple records per ID (id).
    4. All records have covariate values at baseline.
```

```
. list, sepby(id)
```

	id	time0	time	wgt	death	_st	_d	_t	_t0
1.	1	0	2	114	0	1	0	2	0
2.	1	3	11	114	1	1	1	11	3
3.	2	0	2	120	0	1	0	2	0
4.	2	3	11	120	1	1	1	11	3
5.	3	0	2	108	1	1	1	2	0
6.	3	2	7	108	1	1	1	7	2
7.	4	0	2	98	0	1	0	2	0
8.	4	3	4	98	1	1	1	4	3
9.	4	5	11	98	1	1	1	11	5

stbase with the at() option

stbase, at() produces a cross-sectional dataset recording the status of each subject at the specified time. This new dataset is not st. Four "new" variables are created:

- the first entry time for the subject,
- the time on gap,
- the time at risk, and
- the number of failures during the time at risk.

The names given to those variables depend on how your data are stset. Pretend that your stset command was

```
. stset var1, failure(var2) time0(var3) ...
```

Then

the first entry time	will be named	*var3* or time0 or _t0
the time on gap	will be named	gap() or gap or gaptime
the time at risk	will be named	*var1*
the number of (or whether) failures	will be named	*var2* or failure or _d

The names may vary because, for instance, if you did not specify a *var2* variable when you stset your data, stbase, at() looks around for a name.

You need not memorize this; the names are obvious from the output produced by stbase, at().

Consider the actions of stbase, at() with some particular st datasets. Pretend that the command given is

```
. use https://www.stata-press.com/data/r18/stbasexmpl2, clear
. list, sepby(id)
```

	id	time0	time	wgt	death
1.	1	0	2	114	0
2.	1	2	8	110	0
3.	1	8	11	118	1
4.	2	0	1	120	0
5.	2	1	3	111	0
6.	2	3	8	108	0
7.	2	8	10	98	1

```
. stset time, id(id) fail(death) time0(time0)
Survival-time data settings

           ID variable: id
        Failure event: death!=0 & death<.
Observed time interval: (time0, time]
    Exit on or before: failure
_____
           7  total observations
           0  exclusions
_____
           7  observations remaining, representing
           2  subjects
           2  failures in single-failure-per-subject data
          21  total analysis time at risk and under observation
                                          At risk from t =         0
                                 Earliest observed entry t =         0
                                   Last observed exit t =        11
. list, sepby(id)
```

	id	time0	time	wgt	death	_st	_d	_t	_t0
1.	1	0	2	114	0	1	0	2	0
2.	1	2	8	110	0	1	0	8	2
3.	1	8	11	118	1	1	1	11	8
4.	2	0	1	120	0	1	0	1	0
5.	2	1	3	111	0	1	0	3	1
6.	2	3	8	108	0	1	0	8	3
7.	2	8	10	98	1	1	1	10	8

```
. stbase, at(5) replace
          Failure _d: death
   Analysis time _t: time
        ID variable: id

Converting multiple-record data to cross-sectional data ...

Cross-sectional data
  recording each subject status at time 5
     Variable │ Description
─────────────────┼──────────────────────────────────────────────────
           id │ Subject identifier
        time0 │ First entry time
          gap │ Time on gap
         time │ Time at risk
        death │ Number of failures during time at risk
     Variable │         Obs        Mean     Std. dev.         Min          Max
─────────────────┼──────────────────────────────────────────────────────────────
        time0 │           2           0           0            0            0
          gap │           2           0           0            0            0
         time │           2           5           0            5            5
        death │           2           0           0            0            0

. list
```

	id	wgt	death	time	time0	gap
1.	1	114	0	5	0	0
2.	2	120	0	5	0	0

thus producing a cross-section at analysis time 5.

Note that the value of time specified with the at() option must correspond to time in the analysis scale, that is, *t*. See [ST] **stset** for a definition of analysis time.

Single-failure st data where all subjects enter at time 0

The result of stbase, at(5) would be one record per subject. Any subject who was censored before time 5 would not appear in the data; the rest would. Those that failed after time 5 will be recorded as having been censored at time 5 (*failvar* = 0); those that failed at time 5 or earlier will have *failvar* = 1.

timevar will contain

for the failures:	
time of failure	if failed on or before time 5 or
5	if the subject has not failed yet
for the censored:	
5	if the subject has not failed yet

With such data, you could perform

- logistic regression of *failvar* on any of the characteristics or

- incidence-rate analysis, summing the failures (perhaps within strata) and the time at risk, *timevar*.

With these data, you could examine 5-year survival probabilities.

Single-failure st data where some subjects enter after time 0

The data produced by stbase, at(5) would be similar to the above, except

- persons who enter on or after time 5 would not be included in the data (because they have not entered yet) and
- the time at risk, *timevar*, would properly account for the time at which each patient entered.

timevar (the time at risk) will contain

for the failures:	
time of failure or less	if failed on or before time 5 (or less because the subject may not have entered at time 0); or
5 or less	if the subject has not failed yet (or less because the subject may not have entered at time 0)
for the censored:	
5 or less	if the subject has not failed yet (or less because the subject may not have entered at time 0)

Depending on the analysis you are performing, you may have to discard those that enter late. This is easy to do because t0 contains the first time of entry.

With these data, you could perform the following:

- Logistic regression of *failvar* on any of the characteristics, but only if you restricted the sample to if t0 == 0 because those who entered after time 0 have a lesser risk of failing over the fixed interval.
- Incidence-rate analysis, summing the failures (perhaps within stratum) and the time at risk, *timevar*. Here you would have to do nothing differently from what you did in the previous example. The time-at-risk variable already includes the time of entry for each patient.

Single-failure st data with gaps and perhaps delayed entry

These data will be similar to the delayed-entry, no-gap data, but gap will contain 0 only for those observations that have no gap.

If analyzing these data, you could perform

- logistic regression, but the sample must be restricted to if t0 == 0 & gap == 0, or
- incidence-rate analysis, and nothing would need to be done differently; the time at risk, *timevar*, accounts for late entry and gaps.

Multiple-failure st data

The multiple-failure case parallels the single-failure case, except that fail will not solely contain 0 and 1; it will contain 0, 1, 2, ..., depending on the number of failures observed. Regardless of late entry, gaps, etc., you could perform

- Poisson regression of fail, the number of events, but remember to specify exposure(*timevar*), and
- incidence-rate analysis.

Also see

Title

> **stci —** Confidence intervals for means and percentiles of survival time

Description

stci computes means and percentiles of survival time, standard errors, and confidence intervals. For multiple-event data, survival time is the time until a failure.

stci can be used with single- or multiple-record or single- or multiple-failure st data.

Quick start

Median survival time with standard error and 95% confidence interval using stset data

    stci

Also report medians with standard errors and confidence intervals for each level of v1

    stci, by(v1)

Same as above, but report 99% confidence intervals

    stci, by(v1) level(99)

Report the 75th percentile of survival times instead of the medians

    stci, by(v1) p(75)

Mean survival time, computed by exponentially extending curve to zero if last follow-up time is censored

    stci, emean

Same as above, and plot the extended survivor function

    stci, emean graph

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > CIs for means and percentiles of survival time

Syntax

> stci $\begin{bmatrix} if \end{bmatrix}$ $\begin{bmatrix} in \end{bmatrix}$ $\begin{bmatrix} , \ options \end{bmatrix}$

options	Description
[Main]	
by(*varlist*)	perform separate calculations for each group of *varlist*
median	calculate median survival times; the default
rmean	calculate mean survival time restricted to longest follow-up time
emean	calculate the mean survival time by exponentially extending the survival curve to zero
p(*#*)	compute the *#* percentile of survival times
ccorr	calculate the standard error for rmean using a continuity correction
noshow	do not show st setting information
dd(*#*)	set maximum number of decimal digits to report
level(*#*)	set confidence level; default is level(95)
graph	plot exponentially extended survivor function
tmax(*#*)	set maximum analysis time of *#* to be plotted
[Plot]	
cline_options	affect rendition of the plotted lines
Add plots	
addplot(*plot*)	add other plots to the generated graph
Y axis, X axis, Titles, Legend, Overall	
twoway_options	any options other than by() documented in [G-3] *twoway_options*

You must stset your data before using stci; see [ST] stset.
by and collect are allowed; see [U] **11.1.10 Prefix commands**.

Options

> [Main]

by(*varlist*) specifies that separate calculations be made for each group identified by equal values of
the variables in *varlist*, resulting in separate summaries and an overall total. *varlist* may contain
any number of variables, each of which may be string or numeric.

median specifies median survival times. This is the default.

rmean and emean specify mean survival times. If the longest follow-up time is censored, emean
(extended mean) computes the mean survival by exponentially extending the survival curve to zero,
and rmean (restricted mean) computes the mean survival time restricted to the longest follow-up
time. If the longest follow-up time is a failure, the restricted mean survival time and the extended
mean survival time are equal.

p(*#*) specifies the percentile of survival time to be computed. For example, p(25) will compute the
25th percentile of survival times, and p(75) will compute the 75th percentile of survival times.
Specifying p(50) is the same as specifying the median option.

ccorr specifies that the standard error for the restricted mean survival time be computed using a
continuity correction. ccorr is valid only with the rmean option.

noshow prevents stci from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

dd(*#*) specifies the maximum number of decimal digits to be reported for standard errors and confidence intervals. This option affects only how values are reported and not how they are calculated.

level(*#*) specifies the confidence level, as a percentage, for confidence intervals. The default is level(95) or as set by set level; see [U] **20.8 Specifying the width of confidence intervals**.

graph specifies that the exponentially extended survivor function be plotted. This option is valid only when the emean option is also specified and is not valid in conjunction with the by() option.

tmax(*#*) is for use with the graph option. It specifies the maximum analysis time to be plotted.

    ┃ Plot ┃

cline_options affect the rendition of the plotted lines; see [G-3] *cline_options*.

    ┃ Add plots ┃

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] *addplot_option*.

    ┃ Y axis, X axis, Titles, Legend, Overall ┃

twoway_options are any of the options documented in [G-3] *twoway_options*, excluding by(). These include options for titling the graph (see [G-3] *title_options*) and for saving the graph to disk (see [G-3] *saving_option*).

Remarks and examples

Remarks are presented under the following headings:

> *Single-failure data*
> *Multiple-failure data*

Single-failure data

Here is an example of stci with single-record survival data:

```
. use https://www.stata-press.com/data/r18/page2
. stset, noshow
. stci
```

	Number of subjects	50%	Std. err.	[95% conf. interval]	
Total	40	232	2.562933	213	239

```
. stci, by(group)
```

group	Number of subjects	50%	Std. err.	[95% conf. interval]	
1	19	216	7.661029	190	234
2	21	233	3.081611	232	280
Total	40	232	2.562933	213	239

In the example above, we obtained the median survival time, by default.

To obtain the 25th or any other percentile of survival time, specify the p(#) option.

```
. stci, p(25)
```

	Number of subjects	25%	Std. err.	[95% conf. interval]	
Total	40	198	10.76878	164	220

```
. stci, p(25) by(group)
```

group	Number of subjects	25%	Std. err.	[95% conf. interval]	
1	19	190	13.43601	143	213
2	21	232	19.42378	142	233
Total	40	198	10.76878	164	220

The p-percentile of survival time is the analysis time at which $p\%$ of subjects have failed and $1 - p\%$ have not. In the table above, 25% of subjects in group 1 failed by time 190, whereas 25% of subjects in group 2 failed by time 232, indicating a better survival experience for this group.

We can verify the quantities reported by `stci` by plotting and examining the Kaplan–Meier survival curves.

```
. sts graph, by(group)
```


Kaplan–Meier survival estimates
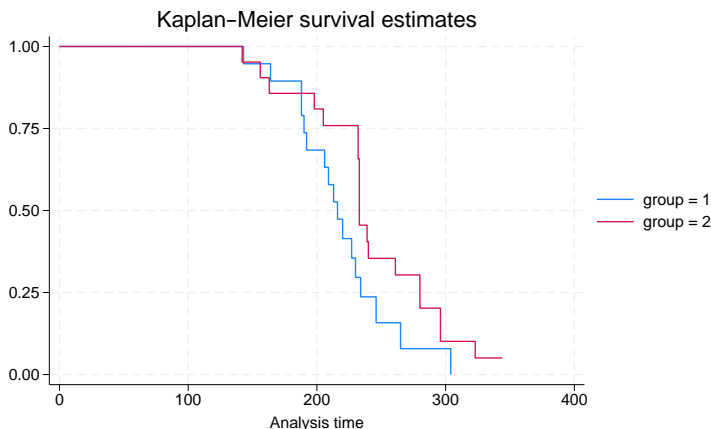
The mean survival time reported by `rmean` is calculated as the area under the Kaplan–Meier survivor function. If the observation with the largest analysis time is censored, the survivor function does not go to zero. Consequently, the area under the curve underestimates the mean survival time.

In the graph above, the survival probability for `group = 1` goes to 0 at analysis time 344, but the survivor function for `group = 2` never goes to 0. For these data, the mean survival time for `group = 1` will be properly estimated, but it will be underestimated for `group = 2`. When we specify the `rmean` option, Stata informs us if any of the mean survival times is underestimated.

```
. stci, rmean by(group)
```

group	Number of subjects	Restricted mean	Std. err.	[95% conf. interval]	
1	19	218.7566	9.122424	200.877	236.636
2	21	241.8571(*)	11.34728	219.617	264.097
Total	40	231.3522(*)	7.700819	216.259	246.446

(*) largest observed analysis time is censored, mean is underestimated

Stata flagged the mean for `group = 2` and the overall mean as being underestimated.

If the largest observed analysis time is censored, `stci`'s `emean` option extends the survivor function from the last observed time to zero by using an exponential function and computes the area under the entire curve.

```
. stci, emean
```

	Number of subjects	Extended mean
Total	40	234.2557

The resulting area must be evaluated with care because it is an ad hoc approximation that can at times be misleading. We recommend that you plot and examine the extended survivor function. This is facilitated by the use of `stci`'s `graph` option.

```
. stci, emean graph
```


Exponentially extended survivor function
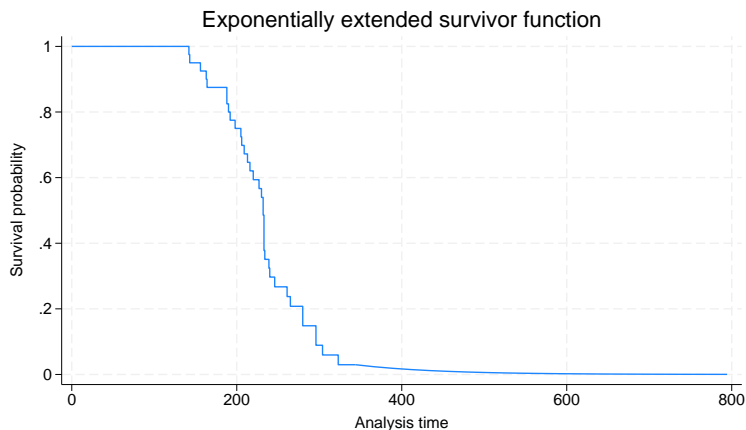
stci also works with multiple-record survival data. Here is a summary of the multiple-record Stanford heart transplant data introduced in [ST] **stset**:

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)
. stset, noshow
. stci
```

	Number of subjects	50%	Std. err.	[95% conf. interval]	
Total	103	100	38.64425	69	219

stci with the by() option may produce results with multiple-record data that you might think are in error:

```
. stci, by(posttran)
```

posttran	Number of subjects	50%	Std. err.	[95% conf. interval]	
0	103	149	43.81077	69	340
1	69	96	58.71712	45	285
Total	103	100	38.64425	69	219

For the number of subjects, $103 + 69 \neq 103$. The posttran variable is not constant for the subjects in this dataset:

```
. stvary posttran
```

	Subjects for whom the variable is				
Variable	constant	varying	never missing	always missing	sometimes missing
posttran	34	69	103	0	0

In this dataset, subjects have one or two records. All subjects were eligible for heart transplantation. They have one record if they die or are lost because of censoring before transplantation, and they have two records if the operation was performed. Then the first record records their survival up to transplantation, and the second records their subsequent survival. posttran is 0 in the first record and 1 in the second.

Therefore, all 103 subjects have records with posttran = 0, and when stci reported results for this group, it summarized the pretransplantation survival. The median survival time was 149 days.

The posttran = 1 line of stci's output summarizes the posttransplantation survival: 69 patients underwent transplantation, and the median survival time was 96 days. For these data, this is not 96 more days, but 96 days in total. That is, the clock was not reset on transplantation. Thus, without attributing cause, we can describe the differences between the groups as an increased hazard of death at early times followed by a decreased hazard later.

Multiple-failure data

If you simply type stci with multiple-failure data, the reported survival time is the survival time to the first failure, assuming that the hazard function is not indexed by number of failures.

Here we have some multiple-failure data:

```
. use https://www.stata-press.com/data/r18/mfail2
. st
-> stset t, id(id) failure(d) time0(t0) exit(time .) noshow

Survival-time data settings

           ID variable: id
        Failure event: d!=0 & d<.
Observed time interval: (t0, t]
    Exit on or before: time .

. stci
```

	Number of subjects	50%	Std. err.	[95% conf. interval]	
Total	926	420	13.42537	394	451

To understand this output, let's also obtain output for each failure separately:

```
. stgen nf = nfailures()
. stci, by(nf)
```

nf	Number of subjects	50%	Std. err.	[95% conf. interval]	
0	926	399	13.91796	381	430
1	529	503	28.53425	425	543
2	221	687	69.38412	549	817
3	58	.	.	.	.
Total	926	420	13.42537	394	451

The stgen command added, for each subject, a variable containing the number of previous failures. nf is 0 for a subject, up to and including the first failure. Then nf is 1 up to and including the second failure, and then it is 2, and so on; see [ST] stgen.

The first line, corresponding to nf = 0, states that among those who had experienced no failures yet, the median time to first failure is 399.

Similarly, the second line, corresponding to `nf = 1`, is for those who have already experienced one failure. The median time of second failures is 503.

When we simply typed `stci`, we obtained the same information shown as the total line of the more detailed output. The total survival time distribution is an estimate of the distribution of the time to first failure, assuming that the hazard function, $h(t)$, is the same across failures—that the second failure is no different from the first failure. This is an odd definition of *same* because the clock, t, is not reset in $h(t)$ upon failure. The hazard of a failure—any failure—at time t is $h(t)$.

Another definition of *same* would have it that the hazard of a failure is given by $h(\tau)$, where τ is the time since the last failure—that the process resets itself. These definitions are different unless $h()$ is a constant function of t.

Let's examine this multiple-failure data, assuming that the process repeats itself. The key variables in this st data are `id`, `t0`, `t`, and `d`:

```
. st
-> stset t, id(id) failure(d) time0(t0) exit(time .) noshow

Survival-time data settings

           ID variable: id
        Failure event: d!=0 & d<.
Observed time interval: (t0, t]
     Exit on or before: time .
```

Our goal, for each subject, is to reset `t0` and `t` to 0 after every failure event. We must trick Stata, or at least trick `stset` because it will not let us set data where the same subject has multiple records summarizing the overlapping periods. The trick is create a new `id` variable that is different for every `id`–`nf` combination (remember, `nf` is the variable we previously created that records the number of prior failures). Then each of the "new" subjects can have their clock start at time 0:

```
. egen newid = group(id nf)

. sort newid t

. by newid: replace t = t - t0[1]
(808 real changes made)

. by newid: gen newt0 = t0 - t0[1]

. stset t, failure(d) id(newid) time0(newt0)

Survival-time data settings

           ID variable: newid
        Failure event: d!=0 & d<.
Observed time interval: (newt0, t]
     Exit on or before: failure

─────────────────────────────────────────────────────────────────────

      1,734  total observations
          0  exclusions

─────────────────────────────────────────────────────────────────────

      1,734  observations remaining, representing
      1,734  subjects
        808  failures in single-failure-per-subject data
    435,444  total analysis time at risk and under observation
                                      At risk from t =            0
                               Earliest observed entry t =        0
                                   Last observed exit t =        797
```

`stset` no longer thinks that we have multiple-failure data. Whereas with `id`, subjects had multiple failures, `newid` gives a unique identity to each `id`–`nf` combination. Each "new" subject has at most one failure.

```
. stci, by(nf)
        Failure _d: d
  Analysis time _t: t
      ID variable: newid
```

nf	Number of subjects	50%	Std. err.	[95% conf. interval]	
0	926	399	13.91796	381	430
1	529	384	18.22987	359	431
2	221	444	29.80391	325	515
3	58	.	.	.	.
Total	1734	404	10.29992	386	430

Compare this table with the one we previously obtained. The number of subjects is the same, but the survival times differ because now we measure the times from one failure to the next, whereas previously we measured the time from a fixed point. The time between events in these data appears to be independent of event number.

Similarly, we can obtain the mean survival time for these data restricted to the longest follow-up time:

```
. stci, rmean by(nf)
        Failure _d: d
  Analysis time _t: t
      ID variable: newid
```

nf	Number of subjects	Restricted mean	Std. err.	[95% conf. interval]	
0	926	399.1802	8.872794	381.79	416.571
1	529	397.0077(*)	13.36058	370.821	423.194
2	221	397.8051(*)	25.78559	347.266	448.344
3	58	471(*)	0	471	471
Total	1734	404.7006	7.021657	390.938	418.463

(*) largest observed analysis time is censored, mean is underestimated

Stored results

stci stores the following in r():

Scalars

r(N_sub)	number of subjects	r(se)	standard error
r(p#)	#th percentile	r(lb)	lower bound of CI
r(rmean)	restricted mean	r(ub)	upper bound of CI
r(emean)	extended mean		

Methods and formulas

The percentiles of survival times are obtained from $S(t)$, the Kaplan–Meier product-limit estimate of the survivor function. The 25th percentile, for instance, is obtained as the minimum value of t such that $S(t) \leq 0.75$. The restricted mean is obtained as the area under the Kaplan–Meier product-limit survivor curve. The extended mean is obtained by extending the Kaplan–Meier product-limit survivor curve to zero by using an exponentially fitted curve and then computing the area under the entire curve. If the longest follow-up time ends in failure, the Kaplan–Meier product-limit survivor curve goes to zero, and the restricted mean and extended mean are identical.

The large-sample standard error for the pth percentile of the distribution is given by Collett (2015, 38) and Klein and Moeschberger (2003, 122) as

$$\frac{\sqrt{\widehat{\text{Var}}\{\widehat{S}(t_p)\}}}{\widehat{f}(t_p)}$$

where $\widehat{\text{Var}}\{\widehat{S}(t_p)\}$ is the Greenwood pointwise variance estimate for $\widehat{S}(t_p)$ and $\widehat{f}(t_p)$ is the estimated density function at the pth percentile.

Confidence intervals, however, are not calculated based on this standard error. For a given confidence level, the upper confidence limit for the pth percentile is defined as the first time at which the upper confidence limit for $S(t)$ (based on a $\ln\{-\ln S(t)\}$ transformation) is less than or equal to $1 - p/100$, and, similarly, the lower confidence limit is defined as the first time at which the lower confidence limit of $S(t)$ is less than or equal to $1 - p/100$.

The restricted mean is obtained as the area under the Kaplan–Meier product-limit survivor curve. The extended mean is obtained by extending the Kaplan–Meier product-limit survivor curve to zero by using an exponentially fitted curve and then computing the area under the entire curve. If the longest follow-up time ends in failure, the Kaplan–Meier product-limit survivor curve goes to zero, and the restricted mean and the extended mean are identical.

The standard error for the estimated restricted mean is computed as given by Klein and Moeschberger (2003, 118) and Collett (2015, 390):

$$\widehat{\text{SE}} = \sum_{i=1}^{D} \widehat{A}_i \sqrt{\frac{d_i}{R_i(R_i - d_i)}}$$

where the sum is over all distinct failure times, $\widehat{A}_i$ is the estimated area under the curve from time i to the maximum follow-up time, R_i is the number of subjects at risk at time i, and d_i is the number of failures at time i.

The $100(1 - \alpha)\%$ confidence interval for the estimated restricted mean is computed as

$$\widehat{A}_i \pm Z_{1-\alpha/2}\widehat{\text{SE}}$$

References

Collett, D. 2015. *Modelling Survival Data in Medical Research.* 3rd ed. Boca Raton, FL: Chapman and Hall/CRC.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data.* 2nd ed. New York: Springer.

Also see

Title

Description

stcox fits, via maximum likelihood, proportional hazards models on st data. stcox can be used with single- or multiple-record or single- or multiple-failure st data.

Quick start

Cox proportional hazards model with covariates x1 and x2 using stset data
    stcox x1 x2

Same as above, but using Efron method for tied failures
    stcox x1 x2, efron

Different baseline hazards for strata defined by levels of svar
    stcox x1 x2, strata(svar)

Adjust for complex survey design using svyset and stset data
    svy: stcox x1 x2

Menu

Statistics > Survival analysis > Regression models > Cox proportional hazards (PH) model

Syntax

stcox [*indepvars*] [*if*] [*in*] [, *options*]

options	Description
Model	
<u>estimate</u>	fit model without covariates
<u>strata</u>(*varnames*)	strata ID variables
<u>shared</u>(*varname*)	shared-frailty ID variable
<u>off</u>set(*varname*)	include *varname* in model with coefficient constrained to 1
breslow	use Breslow method to handle tied failures; the default
efron	use Efron method to handle tied failures
exactm	use exact marginal-likelihood method to handle tied failures
exactp	use exact partial-likelihood method to handle tied failures
Time varying	
tvc(*varlist*)	specify covariates to be interacted with a function of time
texp(*exp*)	specify a function of time; default is texp(_t)
SE/Robust	
vce(*vcetype*)	*vcetype* may be oim, <u>robust</u>, <u>cl</u>uster *clustvar*, <u>bootstrap</u>, or jackknife
<u>noadj</u>ust	do not use standard degree-of-freedom adjustment
Reporting	
<u>level</u>(*#*)	set confidence level; default is level(95)
nohr	report coefficients, not hazard ratios
noshow	do not show st setting information
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Maximization	
maximize_options	control the maximization process; seldom used
coeflegend	display legend instead of statistics

You must stset your data before using stcox; see [ST] **stset**.

varlist may contain factor variables; see [U] **11.4.3 Factor variables**.

bootstrap, by, collect, fp, jackknife, mfp, mi estimate, nestreg, statsby, stepwise, and svy are allowed; see [U] **11.1.10 Prefix commands**.

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix; see [MI] **mi estimate**.

estimate, shared(), efron, exactm, exactp, tvc(), texp(), vce(), and noadjust are not allowed with the svy prefix; see [SVY] **svy**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**. Weights are not supported with efron and exactp. Also weights may not be specified if you are using the bootstrap prefix with the stcox command.

coeflegend does not appear in the dialog box.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

estimate forces fitting of the null model. All Stata estimation commands redisplay results when the command name is typed without arguments. So does stcox. What if you wish to fit a Cox model on $\mathbf{x}_j\boldsymbol{\beta}$, where $\mathbf{x}_j\boldsymbol{\beta}$ is defined as 0? Logic says that you would type stcox. There are no explanatory variables, so there is nothing to type after the command. Unfortunately, this looks the same as stcox typed without arguments, which is a request to redisplay results.

To fit the null model, type stcox, estimate.

strata(*varnames*) specifies up to five strata variables. Observations with equal values of the strata variables are assumed to be in the same stratum. Stratified estimates (equal coefficients across strata but with a baseline hazard distinct for each stratum) are then obtained.

shared(*varname*) specifies that a Cox model with shared frailty be fit. Observations with equal value of *varname* are assumed to have shared (the same) frailty. Across groups, the frailties are assumed to be gamma-distributed latent random effects that affect the hazard multiplicatively, or, equivalently, the logarithm of the frailty enters the linear predictor as a random offset. Think of a shared-frailty model as a Cox model for panel data. *varname* is a variable in the data that identifies the groups. shared() is not allowed in the presence of delayed entries or gaps.

Shared-frailty models are discussed more in *Cox regression with shared frailty*.

offset(*varname*); see [R] **Estimation options**.

breslow, efron, exactm, and exactp specify the method for handling tied failures in the calculation of the log partial likelihood (and residuals). breslow is the default. Each method is described in *Treatment of tied failure times*. efron and the exact methods require substantially more computer time than the default breslow option. exactm and exactp may not be specified with tvc(), vce(robust), or vce(cluster *clustvar*).

Time varying

tvc(*varlist*) specifies the variables to be included in the model as an interaction with a function of time to form time-varying covariates. During estimation, these variables are interacted with analysis time or with a function of analysis time specified in the texp() option. This is a convenience option used to speed up calculations and to avoid having to stsplit (see [ST] **stsplit**) the data over many failure times.

Most predictions are not available after estimation with tvc(). These predictions require that the data be stsplit to generate the requested information; see help tvc note.

texp(*exp*) is used in conjunction with tvc(*varlist*) to specify the function of analysis time that should be used to multiply covariates specified in the tvc() option to include in the model time-varying covariates that are deterministic functions of time. For example, specifying texp(ln(_t)) would cause the covariates in option tvc() to be multiplied by the logarithm of analysis time. If tvc(*varlist*) is used without texp(*exp*), Stata understands that you mean texp(_t) and thus multiplies the covariates by the analysis time.

Both tvc(*varlist*) and texp(*exp*) are explained more in the section on *Cox regression with time-varying covariates using option tvc()* below.

SE/Robust

vce(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory (oim), that are robust to some kinds of misspecification (robust), that

allow for intragroup correlation (cluster *clustvar*), and that use bootstrap or jackknife methods (bootstrap, jackknife); see [R] *vce_option*.

noadjust is for use with vce(robust) or vce(cluster *clustvar*). noadjust prevents the estimated variance matrix from being multiplied by $N/(N-1)$ or $g/(g-1)$, where g is the number of clusters. The default adjustment is somewhat arbitrary because it is not always clear how to count observations or clusters. In such cases, however, the adjustment is likely to be biased toward 1, so we would still recommend making it.

┌─────────┐
│ Reporting │
└─────────┘

level(*#*); see [R] **Estimation options**.

nohr specifies that coefficients be displayed rather than exponentiated coefficients or hazard ratios. This option affects only how results are displayed and not how they are estimated. nohr may be specified at estimation time or when redisplaying previously estimated results (which you do by typing stcox without a variable list).

noshow prevents stcox from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(*#*), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

┌────────────┐
│ Maximization │
└────────────┘

maximize_options: iterate(*#*), [no]log, trace, tolerance(*#*), ltolerance(*#*), nrtolerance(*#*), and nonrtolerance; see [R] **Maximize**. These options are seldom used.

The following option is available with stcox but is not shown in the dialog box:

coeflegend; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

> *Cox regression with uncensored data*
> *Cox regression with censored data*
> *Treatment of tied failure times*
> *Cox regression with time-varying covariates in multiple-record data*
> *Cox regression with time-varying covariates using option tvc()*
> *Robust estimate of variance*
> *Cox regression with multiple-failure data*
> *Stratified estimation*
> *Cox regression as Poisson regression*
> *Cox regression with shared frailty*

What follows is a summary of what can be done with stcox. For a complete tutorial, see Cleves, Gould, and Marchenko (2016), which devotes three chapters to this topic.

In the Cox proportional hazards model (Cox 1972), the hazard is assumed to be

$$h(t) = h_0(t)\exp(\beta_1 x_1 + \cdots + \beta_k x_k)$$

The Cox model provides estimates of $\beta_1, \ldots, \beta_k$ but provides no direct estimate of $h_0(t)$—the baseline hazard. Formally, the function $h_0(t)$ is not directly estimated, but it is possible to recover an estimate of the baseline cumulative hazard $H_0(t)$ and, from that, an estimate of the baseline survivor function $S_0(t)$.

stcox fits the Cox proportional hazards model; that is, it provides estimates of β and its variance–covariance matrix. Estimates of $H_0(t)$, $S_0(t)$, and other predictions and diagnostics are obtained with predict after stcox; see [ST] **stcox postestimation**. For information on fitting a Cox model to survey data, see Cleves, Gould, and Marchenko (2016, sec. 9.5), and for information on handling missing data, see Cleves, Gould, and Marchenko (2016, sec. 9.6).

stcox with the strata() option will produce stratified Cox regression estimates. In the stratified estimator, the hazard at time t for a subject in group i is assumed to be

$$h_i(t) = h_{0i}(t)\exp(\beta_1 x_1 + \cdots + \beta_k x_k)$$

That is, the coefficients are assumed to be the same, regardless of group, but the baseline hazard can be group specific.

Regardless of whether you specify strata(), the default variance estimate is to calculate the conventional, inverse matrix of negative second derivatives. The theoretical justification for this estimator is based on likelihood theory. The vce(robust) option instead switches to the robust measure developed by Lin and Wei (1989). This variance estimator is a variant of the estimator discussed in [U] **20.22 Obtaining robust variance estimates**.

stcox with the shared() option fits a Cox model with shared frailty. A *frailty* is a group-specific latent random effect that multiplies into the hazard function. The distribution of the frailties is gamma with mean 1 and variance to be estimated from the data. Shared-frailty models are used to model within-group correlation. Observations within a group are correlated because they share the same frailty.

We give examples below with uncensored, censored, time-varying, and recurring failure data, but it does not matter in terms of what you type. Once you have stset your data, to fit a model you type stcox followed by the names of the explanatory variables. You do this whether your dataset has single or multiple records, includes censored observations or delayed entry, or even has single or multiple failures. You use stset to describe the properties of the data, and then that information is available to stcox—and all the other st commands—so that you do not have to specify it again.

Cox regression with uncensored data

▷ Example 1

We wish to analyze an experiment testing the ability of emergency generators with a new-style bearing to withstand overloads. For this experiment, the overload protection circuit was disabled, and the generators were run overloaded until they burned up. Here are our data:

```
. use https://www.stata-press.com/data/r18/kva
(Generator experiment)

. list
```

	failtime	load	bearings
1.	100	15	0
2.	140	15	1
3.	97	20	0
4.	122	20	1
5.	84	25	0
6.	100	25	1
7.	54	30	0
8.	52	30	1
9.	40	35	0
10.	55	35	1
11.	22	40	0
12.	30	40	1

Twelve generators, half with the new-style bearings and half with the old, were allocated to this destructive test. The first observation reflects an old-style generator (bearings = 0) under a 15-kVA overload. It stopped functioning after 100 hours. The second generator had new-style bearings (bearings = 1) and, under the same overload condition, lasted 140 hours. Paired experiments were also performed under overloads of 20, 25, 30, 35, and 40 kVA.

We wish to fit a Cox proportional hazards model in which the failure rate depends on the amount of overload and the style of the bearings. That is, we assume that bearings and load do not affect the shape of the overall hazard function, but they do affect the relative risk of failure. To fit this model, we type

```
. stset failtime
(output omitted )

. stcox load bearings

        Failure _d: 1 (meaning all fail)
  Analysis time _t: failtime
Iteration 0:  Log likelihood = -20.274897
Iteration 1:  Log likelihood = -10.515114
Iteration 2:  Log likelihood = -8.8700259
Iteration 3:  Log likelihood = -8.5915211
Iteration 4:  Log likelihood = -8.5778991
Iteration 5:  Log likelihood =  -8.577853
Refining estimates:
Iteration 0:  Log likelihood =  -8.577853
```

```
Cox regression with Breslow method for ties
No. of subjects =  12                                    Number of obs =      12
No. of failures =  12
Time at risk    = 896
                                                         LR chi2(2)    =  23.39
Log likelihood = -8.577853                               Prob > chi2   = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
load	1.52647	.2188172	2.95	0.003	1.152576	2.021653
bearings	.0636433	.0746609	-2.35	0.019	.0063855	.6343223

We find that after controlling for overload, the new-style bearings result in a lower hazard and therefore a longer survivor time.

Once an `stcox` model has been fit, typing `stcox` without arguments redisplays the previous results. Options that affect the display, such as `nohr`—which requests that coefficients rather than hazard ratios be displayed—can be specified upon estimation or when results are redisplayed:

```
. stcox, nohr
Cox regression with Breslow method for ties
No. of subjects =  12                                    Number of obs =      12
No. of failures =  12
Time at risk    = 896
                                                         LR chi2(2)    =  23.39
Log likelihood = -8.577853                               Prob > chi2   = 0.0000
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
load	.4229578	.1433485	2.95	0.003	.1419999	.7039157
bearings	-2.754461	1.173115	-2.35	0.019	-5.053723	-.4551981

◁

❑ Technical note

   `stcox`'s iteration log looks like a standard Stata iteration log up to where it says "Refining estimates". The Cox proportional-hazards likelihood function is indeed a difficult function, both conceptually and numerically. Until Stata says "Refining estimates", it maximizes the Cox likelihood in the standard way by using double-precision arithmetic. Then just to be sure that the answers are accurate, Stata switches to quad-precision routines (double double precision) and completes the maximization procedure from its current location on the likelihood.

❑

Cox regression with censored data

▷ Example 2

   We have data on 48 participants in a cancer drug trial. Of these 48, 28 receive treatment (`drug = 1`) and 20 receive a placebo (`drug = 0`). The participants range in age from 47 to 67 years. We wish to analyze time until death, measured in months. Our data include 1 observation for each patient. The variable `studytime` records either the month of their death or the last month that they were known to be alive. Some of the patients still live, so together with `studytime` is `died`, indicating their health status. Persons known to have died—"noncensored" in the jargon—have `died = 1`, whereas the patients who are still alive—"right-censored" in the jargon—have `died = 0`.

Here is an overview of our data:

```
. use https://www.stata-press.com/data/r18/drugtr
(Patient survival in drug trial)

. st
-> stset studytime, failure(died)

Survival-time data settings

         Failure event: died!=0 & died<.
Observed time interval: (0, studytime]
     Exit on or before: failure

. summarize
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
   studytime |         48        15.5    10.25629          1         39
        died |         48   .6458333    .4833211          0          1
        drug |         48   .5833333    .4982238          0          1
         age |         48      55.875    5.659205         47         67
         _st |         48           1           0          1          1
-------------+---------------------------------------------------------
          _d |         48   .6458333    .4833211          0          1
          _t |         48        15.5    10.25629          1         39
         _t0 |         48           0           0          0          0
```

We typed `stset studytime, failure(died)` previously; that is how `st` knew about this dataset. To fit the Cox model, we type

```
. stcox drug age

        Failure _d: died
  Analysis time _t: studytime

Iteration 0:   Log likelihood = -99.911448
Iteration 1:   Log likelihood = -83.551879
Iteration 2:   Log likelihood = -83.324009
Iteration 3:   Log likelihood = -83.323546
Refining estimates:
Iteration 0:   Log likelihood = -83.323546

Cox regression with Breslow method for ties

No. of subjects =  48                            Number of obs =       48
No. of failures =  31
Time at risk    = 744
                                                 LR chi2(2)    =    33.18
Log likelihood = -83.323546                      Prob > chi2   =   0.0000

------------------------------------------------------------------------------
          _t | Haz. ratio   Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
        drug |   .1048772   .0477017    -4.96   0.000     .0430057    .2557622
         age |   1.120325   .0417711     3.05   0.002     1.041375     1.20526
------------------------------------------------------------------------------
```

We find that the drug results in a lower hazard—and therefore a longer survivor time—controlling for age. Older patients are more likely to die. The model as a whole is statistically significant.

The hazard ratios reported correspond to a one-unit change in the corresponding variable. It is more typical to report relative risk for 5-year changes in age. To obtain such a hazard ratio, we create a new age variable such that a one-unit change indicates a 5-year change:

```
. replace age = age/5
variable age was byte now float
(48 real changes made)

. stcox drug age, nolog

        Failure _d: died
  Analysis time _t: studytime

Cox regression with Breslow method for ties

No. of subjects =  48                                  Number of obs =      48
No. of failures =  31
Time at risk     = 744
                                                       LR chi2(2)    =   33.18
Log likelihood = -83.323544                            Prob > chi2   =  0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
drug	.1048772	.0477017	-4.96	0.000	.0430057	.2557622
age	1.764898	.3290196	3.05	0.002	1.224715	2.543338

◁

Treatment of tied failure times

The proportional hazards model assumes that the hazard function is continuous and, thus, that there are no tied survival times. Because of the way that time is recorded, however, tied events do occur in survival data. In such cases, the partial likelihood must be modified. See *Methods and formulas* for more details on the methods described below.

Stata provides four methods for handling tied failures in calculating the Cox partial likelihood through the `breslow`, `efron`, `exactm`, and `exactp` options. If there are no ties in the data, the results are identical, regardless of the method selected.

Cox regression is a series of comparisons of those subjects who fail to those subjects at risk of failing; we refer to the latter set informally as a *risk pool*. When there are tied failure times, we must decide how to calculate the risk pools for these tied observations. Assume that there are 2 observations that fail in succession. In the calculation involving the second observation, the first observation is not in the risk pool because failure has already occurred. If the two observations have the same failure time, we must decide how to calculate the risk pool for the second observation and in which order to calculate the two observations.

There are two views of time. In the first, time is continuous, so ties should not occur. If they have occurred, the likelihood reflects the marginal probability that the tied-failure events occurred before the nonfailure events in the risk pool (the order that they occurred is not important). This is called the exact marginal likelihood (option `exactm`).

In the second view, time is discrete, so ties are expected. The likelihood is changed to reflect this discreteness and calculates the conditional probability that the observed failures are those that fail in the risk pool given the observed number of failures. This is called the exact partial likelihood (option `exactp`).

Let's assume that there are five subjects—e_1, e_2, e_3, e_4, and e_5—in the risk pool and that subjects e_1 and e_2 fail. Had we been able to observe the events at a better resolution, we might have seen that e_1 failed from risk pool $e_1 + e_2 + e_3 + e_4 + e_5$ and then e_2 failed from risk pool $e_2 + e_3 + e_4 + e_5$. Alternatively, e_2 might have failed first from risk pool $e_1 + e_2 + e_3 + e_4 + e_5$, and then e_1 failed from risk pool $e_1 + e_3 + e_4 + e_5$.

The Breslow method (option `breslow`) for handling tied values simply says that because we do not know the order, we will use the largest risk pool for each tied failure event. This method assumes that both e_1 and e_2 failed from risk pool $e_1 + e_2 + e_3 + e_4 + e_5$. This approximation is fast and is the default method for handling ties. If there are many ties in the dataset, this approximation will not be accurate because the risk pools include too many observations. The Breslow method is an approximation of the exact marginal likelihood.

The Efron method (option `efron`) for handling tied values assumes that the first risk pool is $e_1 + e_2 + e_3 + e_4 + e_5$ and the second risk pool is either $e_2 + e_3 + e_4 + e_5$ or $e_1 + e_3 + e_4 + e_5$. From this, Efron noted that the e_1 and e_2 terms were in the second risk pool with probability 1/2 and so used for the second risk pool $.5(e_1 + e_2) + e_3 + e_4 + e_5$. Efron's approximation is a more accurate approximation of the exact marginal likelihood than Breslow's but takes longer to calculate.

The exact marginal method (option `exactm`) is a misnomer in that the calculation performed is also an *approximation* of the exact marginal likelihood. It is an approximation because it evaluates the likelihood (and derivatives) by using 15-point Gauss–Laguerre quadrature. For small-to-moderate samples, this is slower than the Efron approximation, but the difference in execution time diminishes when samples become larger. You may want to consider the quadrature when deciding to use this method. If the number of tied deaths is large (on average), the quadrature approximation of the function is not well behaved. A little empirical checking suggests that if the number of tied deaths is larger (on average) than 30, the quadrature does not approximate the function well.

When we view time as discrete, the exact partial method (option `exactp`) is the final method available. This approach is equivalent to computing conditional logistic regression where the groups are defined by the risk sets and the outcome is given by the death variable. This is the slowest method to use and can take a significant amount of time if the number of tied failures and the risk sets are large.

Cox regression with time-varying covariates in multiple-record data

▷ Example 3

In [ST] **stset**, we introduce the Stanford heart transplant data in which there are one or two records per patient depending on whether they received a new heart.

This dataset (Crowley and Hu 1977) consists of 103 patients admitted to the Stanford Heart Transplantation Program. Patients were admitted to the program after review by a committee and then waited for an available donor heart. While waiting, some patients died or were transferred out of the program, but 67% received a transplant. The dataset includes the year the patient was accepted into the program along with the patient's age, whether the patient had other heart surgery previously, and whether the patient received a transplant.

In the data, `posttran` becomes 1 when a patient receives a new heart, so it is a time-varying covariate. That does not, however, affect what we type to fit the model:

```
. use https://www.stata-press.com/data/r18/stan3, clear
(Heart transplant data)
. stset t1, failure(died) id(id)
  (output omitted)
```

```
. stcox age posttran surg year

        Failure _d: died
  Analysis time _t: t1
      ID variable: id
Iteration 0:  Log likelihood = -298.31514
Iteration 1:  Log likelihood =  -289.7344
Iteration 2:  Log likelihood = -289.53498
Iteration 3:  Log likelihood = -289.53378
Iteration 4:  Log likelihood = -289.53378
Refining estimates:
Iteration 0:  Log likelihood = -289.53378

Cox regression with Breslow method for ties

No. of subjects =       103                          Number of obs =      172
No. of failures =        75
Time at risk    = 31,938.1
                                                     LR chi2(4)    =    17.56
Log likelihood = -289.53378                          Prob > chi2   =   0.0015
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.030224	.0143201	2.14	0.032	1.002536	1.058677
posttran	.9787243	.3032597	-0.07	0.945	.5332291	1.796416
surgery	.3738278	.163204	-2.25	0.024	.1588759	.8796
year	.8873107	.059808	-1.77	0.076	.7775022	1.012628

We find that older patients have higher hazards, that patients tend to do better over time, and that patients with prior surgery do better. Whether a patient ultimately receives a transplant does not seem to make much difference.

◁

Cox regression with time-varying covariates using option tvc()

The basic proportional hazards regression assumes the relationship

$$h(t) = h_0(t) \exp(\beta_1 x_1 + \cdots + \beta_k x_k)$$

where $h_0(t)$ is the baseline hazard function. For most purposes, this model is sufficient, but sometimes we may wish to introduce variables of the form $z_i(t) = z_i g(t)$, which vary continuously with time so that

$$h(t) = h_0(t) \exp\left\{\beta_1 x_1 + \cdots + \beta_k x_k + g(t)(\gamma_1 z_1 + \cdots + \gamma_m z_m)\right\} \tag{1}$$

where $z_1, \ldots, z_m$ are baseline (constant) covariates and where estimation has the net effect of estimating, say, a regression coefficient, γ_i, for a covariate, $g(t)z_i$, which is a function of the current time.

Variables $z_1, \ldots, z_m$ are specified by using the tvc(*varlist*) option, and $g(t)$ is specified by using the texp(*exp*) option, where t in $g(t)$ is analysis time. For example, if we want $g(t) = \log(t)$, we would use texp(log(_t)) because _t stores the analysis time once the data are stset.

Because the calculations in Cox regression are based on evaluations of the partial log likelihood at the times when failures occur, the above results could also be achieved by using stsplit to split the data at the observed failure times and manually generating the time-varying covariates. tvc() merely represents a more convenient way to accomplish this. However, for large datasets with many distinct failure times, using stsplit may produce datasets that are too large to fit in memory, and even if this were not so, the estimation would take far longer to complete. For these reasons, the tvc() and texp() options described above were introduced.

▷ Example 4

Consider a dataset consisting of 45 observations on recovery time from walking pneumonia. Recovery time (in days) is recorded in the variable `time`, and there are measurements on the covariates `age`, `drug1`, and `drug2`, where `drug1` and `drug2` interact a choice of treatment with initial dosage level. The study was terminated after 30 days, so those who had not recovered by that time were censored (`cured = 0`).

```
. use https://www.stata-press.com/data/r18/drugtr2
. list age drug1 drug2 time cured in 1/12, separator(0)
```

	age	drug1	drug2	time	cured
1.	36	0	50	20.6	1
2.	14	0	50	6.8	1
3.	43	0	125	8.6	1
4.	25	100	0	10	1
5.	50	100	0	30	0
6.	26	0	100	13.6	1
7.	21	150	0	5.4	1
8.	25	0	100	15.4	1
9.	32	125	0	8.6	1
10.	28	150	0	8.5	1
11.	34	0	100	30	0
12.	40	0	50	30	0

Patient 1 took 50 mg of drug number 2 and was cured after 20.6 days, whereas patient 5 took 100 mg of drug number 1 and had yet to recover when the study ended and so was censored at 30 days.

We run a standard Cox regression after `stset`ting the data:

```
. stset time, failure(cured)

Survival-time data settings

         Failure event: cured!=0 & cured<.
Observed time interval: (0, time]
     Exit on or before: failure

──────────────────────────────────────────────────────────────────────

        45  total observations
         0  exclusions

──────────────────────────────────────────────────────────────────────

        45  observations remaining, representing
        36  failures in single-record/single-failure data
     677.9  total analysis time at risk and under observation
                                          At risk from t =         0
                                Earliest observed entry t =         0
                                    Last observed exit t =        30
```

```
. stcox age drug1 drug2
          Failure _d: cured
    Analysis time _t: time
Iteration 0:  Log likelihood = -116.54385
Iteration 1:  Log likelihood = -102.77311
Iteration 2:  Log likelihood = -101.92794
Iteration 3:  Log likelihood = -101.92504
Iteration 4:  Log likelihood = -101.92504
Refining estimates:
Iteration 0:  Log likelihood = -101.92504
Cox regression with Breslow method for ties
No. of subjects =    45                            Number of obs =      45
No. of failures =    36
Time at risk    = 677.9
                                                   LR chi2(3)     =   29.24
Log likelihood = -101.92504                        Prob > chi2    =  0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	.8759449	.0253259	-4.58	0.000	.8276873	.9270162
drug1	1.008482	.0043249	1.97	0.049	1.000041	1.016994
drug2	1.00189	.0047971	0.39	0.693	.9925323	1.011337

The output includes p-values for the tests of the null hypotheses that each regression coefficient is 0 or, equivalently, that each hazard ratio is 1. That all hazard ratios are apparently close to 1 is a matter of scale; however, we can see that drug number 1 significantly increases the risk of being cured and so is an effective drug, whereas drug number 2 is ineffective (given the presence of age and drug number 1 in the model).

Suppose now that we wish to fit a model in which we account for the effect that as time goes by, the actual level of the drug remaining in the body diminishes, say, at an exponential rate. If it is known that the half-life of both drugs is close to 2 days, we can say that the actual concentration level of the drug in the patient's blood is proportional to the initial dosage times, $\exp(-0.35t)$, where t is analysis time. We now fit a model that reflects this change.

```
. stcox age, tvc(drug1 drug2) texp(exp(-0.35*_t)) nolog
          Failure _d: cured
    Analysis time _t: time
Cox regression with Breslow method for ties
No. of subjects =    45                            Number of obs =      45
No. of failures =    36
Time at risk    = 677.9
                                                   LR chi2(3)     =   36.98
Log likelihood = -98.052763                        Prob > chi2    =  0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
main						
age	.8614636	.028558	-4.50	0.000	.8072706	.9192948
tvc						
drug1	1.304744	.1135967	3.06	0.002	1.100059	1.547514
drug2	1.200613	.1113218	1.97	0.049	1.001103	1.439882

Note: Variables in **tvc** equation interacted with **exp(-0.35*_t)**.

The first equation, main, reports the results (hazard ratios) for the covariates that do not vary over time; the second equation, tvc, reports the results for the time-varying covariates.

As the level of drug in the blood system decreases, the drug's effectiveness diminishes. Accounting for this serves to unmask the effects of both drugs in that we now see increased effects on both. In fact, the effect on recovery time of drug number 2 now becomes significant.

❑ Technical note

The interpretation of hazard ratios requires careful consideration here. For the first model, the hazard ratio for, say, drug1 is interpreted as the proportional change in hazard when the dosage level of drug1 is increased by one unit. For the second model, the hazard ratio for drug1 is the proportional change in hazard when the blood concentration level—that is, drug1*exp$(-0.35t)$—increases by 1.
❑

Because the number of observations in our data is relatively small, for illustrative purposes we can stsplit the data at each recovery time, manually generate the blood concentration levels, and refit the second model.

```
. generate id=_n
. streset, id(id)
(output omitted )
. stsplit, at(failures)
(31 failure times)
(812 observations (episodes) created)
. generate drug1emt = drug1*exp(-0.35*_t)
. generate drug2emt = drug2*exp(-0.35*_t)
. stcox age drug1emt drug2emt

         Failure _d: cured
   Analysis time _t: time
         ID variable: id
Iteration 0:  Log likelihood = -116.54385
Iteration 1:  Log likelihood = -99.321912
Iteration 2:  Log likelihood =  -98.07369
Iteration 3:  Log likelihood =  -98.05277
Iteration 4:  Log likelihood = -98.052763
Refining estimates:
Iteration 0:  Log likelihood = -98.052763

Cox regression with Breslow method for ties

No. of subjects =     45                          Number of obs =     857
No. of failures =     36
Time at risk    = 677.9
                                                  LR chi2(3)    =   36.98
Log likelihood = -98.052763                       Prob > chi2   =  0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	.8614636	.028558	-4.50	0.000	.8072706	.9192948
drug1emt	1.304744	.1135967	3.06	0.002	1.100059	1.547514
drug2emt	1.200613	.1113218	1.97	0.049	1.001103	1.439882

We get the same answer. However, this required more work both for Stata and for you.
◁

Above we used tvc() and texp() to demonstrate fitting models with time-varying covariates, but these options can also be used to fit models with *time-varying coefficients*. For simplicity, consider a version of (1) that contains only one fixed covariate, x_1, and sets $z_1 = x_1$:

$$h(t) = h_0(t) \exp\left\{\beta_1 x_1 + g(t)\gamma_1 x_1\right\}$$

Rearranging terms results in

$$h(t) = h_0(t) \exp\left[\{\beta_1 + \gamma_1 g(t)\} x_1\right]$$

Given this new arrangement, we consider that $\beta_1 + \gamma_1 g(t)$ is a (possibly) time-varying coefficient on the covariate x_1, for some specified function of time $g(t)$. The coefficient has a time-invariant component, β_1, with γ_1 determining the magnitude of the time-dependent deviations from β_1. As such, a test of $\gamma_1 = 0$ is a test of time invariance for the coefficient on x_1.

Confirming that a coefficient is time invariant is one way of testing the proportional-hazards assumption. Proportional hazards implies that the relative hazard (that is, β) is fixed over time, and this assumption would be violated if a time interaction proved significant.

▷ Example 5

Returning to our cancer drug trial, we now include a time interaction on `age` as a way of testing the proportional-hazards assumption for that covariate:

```
. use https://www.stata-press.com/data/r18/drugtr, clear
(Patient survival in drug trial)

. stcox drug age, tvc(age)

        Failure _d: died
  Analysis time _t: studytime

Iteration 0:  Log likelihood = -99.911448
Iteration 1:  Log likelihood = -83.328648
Iteration 2:  Log likelihood = -83.095631
Iteration 3:  Log likelihood = -83.095036
Refining estimates:
Iteration 0:  Log likelihood = -83.095036

Cox regression with Breslow method for ties

No. of subjects =  48                          Number of obs =       48
No. of failures =  31
Time at risk    = 744
                                               LR chi2(3)    =    33.63
Log likelihood = -83.095036                    Prob > chi2   =   0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
main						
drug	.1059862	.0478178	-4.97	0.000	.0437737	.2566171
age	1.156977	.07018	2.40	0.016	1.027288	1.303037
tvc						
age	.9970966	.0042415	-0.68	0.494	.988818	1.005445

Note: Variables in `tvc` equation interacted with `_t`.

We used the default function of time, $g(t) = t$, although we could have specified otherwise with the `texp()` option. The estimation results are presented in terms of hazard ratios, and so 0.9971 is an estimate of $\exp(\gamma_{\text{age}})$. Tests of hypotheses, however, are in terms of the original metric, and so 0.494 is the significance for the test of $H_0: \gamma_{\text{age}} = 0$ versus the two-sided alternative. With respect to this specific form of misspecification, there is not much evidence to dispute the proportionality of hazards when it comes to age.

◁

Robust estimate of variance

By default, stcox produces the conventional estimate for the variance–covariance matrix of the coefficients (and hence the reported standard errors). If, however, you specify the vce(robust) option, stcox switches to the robust variance estimator (Lin and Wei 1989).

The key to the robust calculation is using the efficient score residual for each subject in the data for the variance calculation. Even in simple single-record, single-failure survival data, the same subjects appear repeatedly in the risk pools, and the robust calculation needs to account for that.

▷ Example 6

Refitting the Stanford heart transplant data model with robust standard errors, we obtain

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)
. stset t1, failure(died) id(id)
Survival-time data settings
           ID variable: id
        Failure event: died!=0 & died<.
Observed time interval: (t1[_n-1], t1]
    Exit on or before: failure
─────────────────────────────────────────────────────────────────
        172  total observations
          0  exclusions
─────────────────────────────────────────────────────────────────
        172  observations remaining, representing
        103  subjects
         75  failures in single-failure-per-subject data
   31,938.1  total analysis time at risk and under observation
                                             At risk from t =         0
                                  Earliest observed entry t =         0
                                      Last observed exit t =     1,799
```

```
. stcox age posttran surg year, vce(robust)

        Failure _d: died
  Analysis time _t: t1
      ID variable: id

Iteration 0:  Log pseudolikelihood = -298.31514
Iteration 1:  Log pseudolikelihood =  -289.7344
Iteration 2:  Log pseudolikelihood = -289.53498
Iteration 3:  Log pseudolikelihood = -289.53378
Iteration 4:  Log pseudolikelihood = -289.53378
Refining estimates:
Iteration 0:  Log pseudolikelihood = -289.53378

Cox regression with Breslow method for ties

No. of subjects =       103                          Number of obs =      172
No. of failures =        75
Time at risk    = 31,938.1
                                                     Wald chi2(4)  =   19.68
Log pseudolikelihood = -289.53378                    Prob > chi2   = 0.0006

                             (Std. err. adjusted for 103 clusters in id)
```

_t	Haz. ratio	Robust std. err.	z	P>|z|	[95% conf. interval]
age	1.030224	.0148771	2.06	0.039	1.001474	1.059799
posttran	.9787243	.2961736	-0.07	0.943	.5408498	1.771104
surgery	.3738278	.1304912	-2.82	0.005	.1886013	.7409665
year	.8873107	.0613176	-1.73	0.084	.7749139	1.01601

Note the word Robust above std. err. in the table and the phrase "Std. err. adjusted for 103 clusters in id" above the table.

The hazard ratio estimates are the same as before, but the standard errors are slightly different.

◁

❑ Technical note

In the previous example, stcox knew to specify vce(cluster id) for us when we specified vce(robust).

To see the importance of vce(cluster id), consider simple single-record, single-failure survival data, a piece of which is

```
       t0           t          died          x
        0           5             1          1
        0           9             0          1
        0           8             0          0
```

and then consider the absolutely equivalent multiple-record survival data:

```
       id          t0           t          died          x
        1           0           3             0          1
        1           3           5             1          1
        2           0           6             0          1
        2           6           9             0          1
        3           0           3             0          0
        3           3           8             0          0
```

Both datasets record the same underlying data, and so both should produce the same numerical results. This should be true regardless of whether vce(robust) is specified.

In the second dataset, were we to ignore id, it would appear that there are 6 observations on 6 subjects. The key ingredients in the robust calculation are the efficient score residuals, and viewing the data as 6 observations on 6 subjects produces different score residuals. Let's call the 6 score residuals s_1, s_2, ..., s_6 and the 3 score residuals that would be generated by the first dataset S_1, S_2, and S_3. $S_1 = s_1 + s_2$, $S_2 = s_3 + s_4$, and $S_3 = s_5 + s_6$.

That residuals sum is the key to understanding the vce(cluster *clustvar*) option. When you specify vce(cluster id), Stata makes the robust calculation based not on the overly detailed s_1, s_2, ..., s_6 but on $S_1 + S_2$, $S_3 + S_4$, and $S_5 + S_6$. That is, Stata sums residuals within clusters before entering them into subsequent calculations (where they are squared), so results estimated from the second dataset are equal to those estimated from the first. In more complicated datasets with time-varying regressors, delayed entry, and gaps, this action of summing within cluster, in effect, treats the cluster (which is typically a subject) as a unified whole.

Because we had stset an id() variable, stcox knew to specify vce(cluster id) for us when we specified vce(robust). You may, however, override the default clustering by specifying vce(cluster *clustvar*) with a different variable from the one you used in stset, id(). This is useful in analyzing multiple-failure data, where you need to stset a pseudo-ID establishing the time from the last failure as the onset of risk.

❑

Cox regression with multiple-failure data

▷ Example 7

In [ST] **stsum**, we introduce a multiple-failure dataset:

```
. use https://www.stata-press.com/data/r18/mfail
. stdescribe
```

			Per subject		
Category	Total	Mean	Min	Median	Max
Number of subjects	926				
Number of records	1734	1.87257	1	2	4
Entry time (first)		0	0	0	0
Exit time (final)		470.6857	1	477	960
Subjects with gap	0				
Time on gap	0	.	.	.	.
Time at risk	435855	470.6857	1	477	960
Failures	808	.8725702	0	1	3

This dataset contains two variables—x1 and x2—which we believe affect the hazard of failure.

If we simply want to analyze these multiple-failure data as if the baseline hazard remains unchanged as events occur (that is, the hazard may change with time, but time is measured from 0 and is independent of when the last failure occurred), we can type

```
. stcox x1 x2, vce(robust)

Iteration 0:   Log pseudolikelihood = -5034.9569
Iteration 1:   Log pseudolikelihood = -4978.4198
Iteration 2:   Log pseudolikelihood = -4978.1915
Iteration 3:   Log pseudolikelihood = -4978.1914
Refining estimates:
Iteration 0:   Log pseudolikelihood = -4978.1914

Cox regression with Breslow method for ties

No. of subjects =      926                           Number of obs =   1,734
No. of failures =      808
Time at risk    = 435,855
                                                     Wald chi2(2)  = 152.13
Log pseudolikelihood = -4978.1914                    Prob > chi2   = 0.0000

                               (Std. err. adjusted for 926 clusters in id)
```

_t	Haz. ratio	Robust std. err.	z	P>|z|	[95% conf. interval]	
x1	2.273456	.1868211	9.99	0.000	1.935259	2.670755
x2	.329011	.0523425	-6.99	0.000	.2408754	.4493951

We chose to fit this model with robust standard errors—we specified vce(robust)—but you can estimate conventional standard errors if you wish.

In [ST] **stsum**, we discuss analyzing this dataset as the time since last failure. We wished to assume that the hazard function remained unchanged with failure, except that one restarted the same hazard function. To that end, we made the following changes to our data:

```
. stgen nf = nfailures()
. egen newid = group(id nf)
. sort newid t
. by newid: replace t = t - t0[1]
(808 real changes made)
. by newid: gen newt0 = t0 - t0[1]
. stset t, id(newid) failure(d) time0(newt0) noshow
Survival-time data settings

           ID variable: newid
         Failure event: d!=0 & d<.
Observed time interval: (newt0, t]
     Exit on or before: failure

_____

      1,734  total observations
          0  exclusions

_____

      1,734  observations remaining, representing
      1,734  subjects
        808  failures in single-failure-per-subject data
    435,444  total analysis time at risk and under observation
                                         At risk from t =          0
                               Earliest observed entry t =          0
                                   Last observed exit t =        797
```

That is, we took each subject and made many `newid` subjects out of each, with each subject entering at time 0 (now meaning the time of the last failure). `id` still identifies a real subject, but Stata thinks the identifier variable is `newid` because we `stset, id(newid)`. If we were to fit a model with `vce(robust)`, we would get

```
. stcox x1 x2, vce(robust) nolog
Cox regression with Breslow method for ties
No. of subjects =    1,734                        Number of obs =   1,734
No. of failures =      808
Time at risk    = 435,444
                                                  Wald chi2(2)  =   88.51
Log pseudolikelihood = -5062.5815                 Prob > chi2   =  0.0000
                            (Std. err. adjusted for 1,734 clusters in newid)
```

		Robust				
_t	Haz. ratio	std. err.	z	P>|z|	[95% conf. interval]	
x1	2.002547	.1936906	7.18	0.000	1.656733	2.420542
x2	.2946263	.0569167	-6.33	0.000	.2017595	.4302382

Note carefully the message concerning the clustering: standard errors have been adjusted for clustering on `newid`. We, however, want the standard errors adjusted for clustering on `id`, so we must specify the `vce(cluster `*clustvar*`)` option:

```
. stcox x1 x2, vce(cluster id) nolog
Cox regression with Breslow method for ties
No. of subjects =   1,734                                Number of obs =  1,734
No. of failures =     808
Time at risk    = 435,444
                                                         Wald chi2(2)  =  93.66
Log pseudolikelihood = -5062.5815                        Prob > chi2   = 0.0000
                                (Std. err. adjusted for 926 clusters in id)
```

		Robust				
_t	Haz. ratio	std. err.	z	P>|z|	[95% conf. interval]	
x1	2.002547	.1920151	7.24	0.000	1.659452	2.416576
x2	.2946263	.0544625	-6.61	0.000	.2050806	.4232709

That is, if you are using vce(robust), you must remember to specify vce(cluster *clustvar*) for yourself when

1. you are analyzing multiple-failure data and

2. you have reset time to time since last failure, so what Stata considers the subjects are really subsubjects.

◁

Stratified estimation

When you type

```
. stcox xvars, strata(svars)
```

you are allowing the baseline hazard functions to differ for the groups identified by *svars*. This is equivalent to fitting separate Cox proportional hazards models under the constraint that the coefficients are equal but the baseline hazard functions are not.

▷ Example 8

Say that in the Stanford heart experiment data, there was a change in treatment for all patients, before and after transplant, in 1970 and then again in 1973. Further assume that the proportional-hazards assumption is not reasonable for these changes in treatment—perhaps the changes result in short-run benefit but little expected long-run benefit. Our interest in the data is not in the effect of these treatment changes but in the effect of transplantation, for which we still find the proportional-hazards assumption reasonable. We might fit our model to account for these fictional changes by typing

```
. use https://www.stata-press.com/data/r18/stan3, clear
(Heart transplant data)
. generate pgroup = year
. recode pgroup min/69=1 70/72=2 73/max=3
(172 changes made to pgroup)
```

```
. stcox age posttran surg year, strata(pgroup) nolog
        Failure _d: died
  Analysis time _t: t1
      ID variable: id
Stratified Cox regression with Breslow method for ties
Strata variable: pgroup
No. of subjects =      103                         Number of obs =      172
No. of failures =       75
Time at risk    = 31,938.1
                                                   LR chi2(4)     =   20.67
Log likelihood = -213.35033                        Prob > chi2    =  0.0004
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.027406	.0150188	1.85	0.064	.9983874	1.057268
posttran	1.075476	.3354669	0.23	0.816	.583567	1.982034
surgery	.2222415	.1218386	-2.74	0.006	.0758882	.6508429
year	.5523966	.1132688	-2.89	0.004	.3695832	.825638

Of course, we could obtain the robust estimate of variance by also including the vce(robust) option.

◁

Cox regression as Poisson regression

▷ Example 9

In example 2, we fit the following Cox model to data from a cancer drug trial with 48 participants:

```
. use https://www.stata-press.com/data/r18/drugtr, clear
(Patient survival in drug trial)
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
studytime	48	15.5	10.25629	1	39
died	48	.6458333	.4833211	0	1
drug	48	.5833333	.4982238	0	1
age	48	55.875	5.659205	47	67
_st	48	1	0	1	1
_d	48	.6458333	.4833211	0	1
_t	48	15.5	10.25629	1	39
_t0	48	0	0	0	0

```
. stcox drug age
  (output omitted )
Cox regression with Breslow method for ties
No. of subjects = 48                               Number of obs =       48
No. of failures = 31
Time at risk    = 744
                                                   LR chi2(2)     =   33.18
Log likelihood = -83.323546                        Prob > chi2    =  0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
drug	.1048772	.0477017	-4.96	0.000	.0430057	.2557622
age	1.120325	.0417711	3.05	0.002	1.041375	1.20526

In what follows, we discuss baseline hazard functions. Thus for clarity, we first fit the same model with an alternate `age` variable so that "baseline" reflects someone in the control group who is 50 years old and not a newborn; see *Making baseline reasonable* in [ST] **stcox postestimation** for more details.

```
. generate age50 = age - 50

. stcox drug age50

  (output omitted )

Cox regression with Breslow method for ties

No. of subjects = 48                              Number of obs =      48
No. of failures = 31
Time at risk    = 744
                                                  LR chi2(2)    = 33.18
Log likelihood = -83.323546                       Prob > chi2   = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
drug	.1048772	.0477017	-4.96	0.000	.0430057	.2557622
age50	1.120325	.0417711	3.05	0.002	1.041375	1.20526

Because `stcox` does not estimate a baseline hazard function, our model and hazard ratios remain unchanged.

Among others, Royston and Lambert (2011, sec. 4.5) show that you can obtain identical hazard ratios by fitting a Poisson model on the above data after splitting on all observed failure times.

Because these data have already been `stset`, variable `_t0` contains the beginning of the time span (which, for these simple data, is time zero for everyone), variable `_t` contains the end of the time span, and variable `_d` indicates failure (`_d == 1`) or censoring (`_d == 0`).

As we did in example 4, we can split these single-record observations at each observed failure time, thus creating a dataset with multiple records per subject. To do so, we must first create an ID variable that identifies each observation as a distinct patient:

```
. generate id = _n

. streset, id(id)
-> stset studytime, id(id) failure(died)

Survival-time data settings

         ID variable: id
      Failure event: died!=0 & died<.
Observed time interval: (studytime[_n-1], studytime]
    Exit on or before: failure

─────────────────────────────────────────────────────────────────
        48  total observations
         0  exclusions
─────────────────────────────────────────────────────────────────
        48  observations remaining, representing
        48  subjects
        31  failures in single-failure-per-subject data
       744  total analysis time at risk and under observation
                                     At risk from t =          0
                              Earliest observed entry t =        0
                                  Last observed exit t =         39
. stsplit, at(failures) riskset(interval)
(21 failure times)
(534 observations (episodes) created)
```

The output shows that we have 21 distinct failure times and that we created 534 new observations for a total of $48 + 534 = 582$ observations. Also created is the `interval` variable, which contains a value of 1 for those records that span from time zero to the first failure time, 2 for those records that span from the first failure time to the second failure time, all the way up to a value of 21 for those records that span from the 20th failure time to the 21st failure time. To see this requires a little bit of sorting and data manipulation:

```
. gsort _t -_d
. by _t: generate tolist = (_n==1) & _d
. list _t0 _t interval if tolist
```

	_t0	_t	interval
1.	0	1	1
49.	1	2	2
95.	2	3	3
140.	3	4	4
184.	4	5	5
226.	5	6	6
266.	6	7	7
303.	7	8	8
340.	8	10	9
371.	10	11	10
400.	11	12	11
426.	12	13	12
450.	13	15	13
473.	15	16	14
494.	16	17	15
517.	17	22	16
532.	22	23	17
545.	23	24	18
556.	24	25	19
566.	25	28	20
576.	28	33	21

Thus for example, interval 16 ranges from time 17 to time 22.

For this newly created multiple-record dataset, our Cox model fit will be identical because we have not added any information to the data. If you do not believe us, feel free to now try the following command:

```
. stcox drug age50
```

At this point, it would seem that making the dataset bigger is a needless waste of space, but what it grants us is the ability to directly estimate the baseline hazard function in addition to the hazard ratios we previously obtained. We accomplish this by using Poisson regression.

Poisson regression models event counts, and so we use our event counter for these data, the failure indicator _d, as the response variable. That _d is only valued as zero or one should not bother you—it is still a count variable. We need to treat time spanned as the amount of exposure a subject had toward failing; the longer the interval, the greater the exposure. As such, we create a variable that records the length of each time span and include it as an `exposure()` variable in our Poisson model. We also include indicator variables for each of the 21 time intervals, with no base level assumed; we use the `ibn.` factor-variable specification and the `noconstant` option:

```
. generate time_exposed = _t - _t0

. poisson _d ibn.interval drug age50, exposure(time_exposed) noconstant irr

Iteration 0:   Log likelihood = -1239.0595
Iteration 1:   Log likelihood = -114.23986
Iteration 2:   Log likelihood = -100.13556
Iteration 3:   Log likelihood = -99.938857
Iteration 4:   Log likelihood = -99.937354
Iteration 5:   Log likelihood = -99.937354
```

```
Poisson regression                                Number of obs =      573
                                                  Wald chi2(23) = 224.18
Log likelihood = -99.937354                       Prob > chi2   = 0.0000
```

_d	IRR	Std. err.	z	P>|z|	[95% conf. interval]	
interval						
1	.0360771	.0284092	-4.22	0.000	.0077081	.1688562
2	.0215286	.0225926	-3.66	0.000	.0027526	.1683778
3	.0228993	.0240269	-3.60	0.000	.0029289	.1790349
4	.0471539	.0366942	-3.92	0.000	.0102596	.2167234
5	.0596354	.045201	-3.72	0.000	.0134999	.2634375
6	.0749754	.0561057	-3.46	0.001	.017296	.3250055
7	.0396981	.0406826	-3.15	0.002	.0053267	.2958558
8	.1203377	.0744625	-3.42	0.001	.0357845	.4046762
9	.0276002	.0283969	-3.49	0.000	.003674	.207341
10	.1120012	.083727	-2.93	0.003	.0258763	.4847777
11	.1358135	.1024475	-2.65	0.008	.0309642	.5956972
12	.1007666	.1040271	-2.22	0.026	.0133221	.7621858
13	.0525547	.0540884	-2.86	0.004	.0069915	.395051
14	.1206462	.1250492	-2.04	0.041	.0158215	.919984
15	.1321868	.1357583	-1.97	0.049	.0176599	.9894363
16	.0670895	.0503478	-3.60	0.000	.0154122	.2920415
17	.5736017	.4415411	-0.72	0.470	.1268766	2.59322
18	.4636009	.5113227	-0.70	0.486	.0533731	4.026856
19	.5272168	.5810138	-0.58	0.561	.0608039	4.571377
20	.2074545	.2292209	-1.42	0.155	.023791	1.80898
21	.2101074	.2344194	-1.40	0.162	.0235909	1.871275
drug	.1048772	.0477017	-4.96	0.000	.0430057	.2557622
age50	1.120325	.0417711	3.05	0.002	1.041375	1.20526
ln(time_e~d)	1	(exposure)				

The incidence-rate ratios from `poisson` (obtained with the `irr` option) are identical to the hazard ratios we previously obtained. Additionally, the incidence-rate ratio for each of the 21 intervals is an estimate of the baseline hazard function for that time interval.

`poisson` gives us an estimated baseline hazard function (the hazard for someone aged 50 in the control group) as a piecewise-constant function. If we had continued to use `stcox`, estimating the baseline hazard function would have required that we apply a kernel smoother to the estimated baseline contributions; see example 3 of [ST] **stcox postestimation** for details. In other words, estimating a baseline hazard after `stcox` is not easy, and it requires choosing a kernel function and bandwidth. As such, the title of this section is technically a misnomer; the models are not exactly the same, only the "hazard ratios" are. Using `poisson` instead of `stcox` carries the added assumption that the baseline hazard is constant between observed failures. Making this assumption buys you the ability to directly estimate the baseline hazard.

There also exists a duality between the Poisson model and the exponential model as fit by `streg`; see [ST] **streg**. A defining property of the Poisson distribution is that waiting times between events are distributed as exponential. Thus we can fit the same piecewise-constant hazard model with

```
. streg ibn.interval drug age50, dist(exponential) noconstant
```

which we invite you to try.

Of course, if you are willing to assume the hazard is piecewise constant, then perhaps you do not need it to change over all 21 observed failure times, and thus perhaps you would want to collapse some intervals. Better still, why not just use `streg` without the indicator variables for `interval`, assume the baseline hazard is some smooth function, and reduce your 21 parameters to one or two estimated shape parameters? The advantages to this fully parametric approach are that you get a parsimonious model and smooth hazard functions that you can estimate at any time point. The disadvantage is that you now carry the stringent assumption that your hazard follows the chosen functional form. If you choose the wrong function, then your hazard ratios are, in essence, worthless.

The two extremes here are the model that makes no assumption about the baseline hazard (the Cox model) and the model that makes the strongest assumptions about the baseline hazard (the fully parametric model). Our piecewise-constant baseline hazard model represents a compromise between Cox regression and fully parametric regression. If you are interested in other ways you can compromise between Cox and parametric models, we recommend you read Royston and Lambert (2011), which is entirely devoted to that topic. There you will find information on (among other things) Royston–Parmar models (Royston and Parmar 2002; Lambert and Royston 2009), proportional odds models, scaled-probit models, the use of cubic splines and fractional polynomials, time-dependent effects, and models for relative survival.

◁

Cox regression with shared frailty

A shared-frailty model is the survival-data analog to regression models with random effects. A *frailty* is a latent random effect that enters multiplicatively on the hazard function. In a Cox model, the data are organized as $i = 1, \ldots, n$ groups with $j = 1, \ldots, n_i$ observations in group i. For the jth observation in the ith group, the hazard is

$$h_{ij}(t) = h_0(t)\alpha_i \exp(\mathbf{x}_{ij}\boldsymbol{\beta})$$

where α_i is the group-level frailty. The frailties are unobservable positive quantities and are assumed to have mean 1 and variance θ, to be estimated from the data. You can fit a Cox shared-frailty model by specifying `shared(`*varname*`)`, where *varname* defines the groups over which frailties are shared. `stcox, shared()` treats the frailties as being gamma distributed, but this is mainly an issue of computational convenience; see *Methods and formulas*. Theoretically, any distribution with positive support, mean 1, and finite variance may be used to model frailty.

Shared-frailty models are used to model within-group correlation; observations within a group are correlated because they share the same frailty. The estimate of θ is used to measure the degree of within-group correlation, and the shared-frailty model reduces to standard Cox when $\theta = 0$.

For $\nu_i = \log\alpha_i$, the hazard can also be expressed as

$$h_{ij}(t) = h_0(t) \exp(\mathbf{x}_{ij}\boldsymbol{\beta} + \nu_i)$$

and thus the log frailties, ν_i, are analogous to random effects in standard linear models.

▷ Example 10

Consider the data from a study of 38 kidney dialysis patients, as described in McGilchrist and Aisbett (1991). The study is concerned with the prevalence of infection at the catheter insertion point. Two recurrence times (in days) are measured for each patient, and each recorded time is the time from initial insertion (onset of risk) to infection or censoring:

```
. use https://www.stata-press.com/data/r18/catheter, clear
(Kidney data, McGilchrist and Aisbett, Biometrics, 1991)

. list patient time infect age female in 1/10
```

	patient	time	infect	age	female
1.	1	16	1	28	0
2.	1	8	1	28	0
3.	2	13	0	48	1
4.	2	23	1	48	1
5.	3	22	1	32	0
6.	3	28	1	32	0
7.	4	318	1	31.5	1
8.	4	447	1	31.5	1
9.	5	30	1	10	0
10.	5	12	1	10	0

Each patient (`patient`) has two recurrence times (`time`) recorded, with each catheter insertion resulting in either infection (`infect==1`) or right-censoring (`infect==0`). Among the covariates measured are `age` and sex (`female==1` if female, `female==0` if male).

One subtlety to note concerns the use of the generic term *subjects*. In this example, the subjects are taken to be the individual catheter insertions, not the patients themselves. This is a function of how the data were recorded—the onset of risk occurs at catheter insertion (of which there are two for each patient), and not, say, at the time of admission of the patient into the study. We therefore have two subjects (insertions) within each group (patient).

It is reasonable to assume independence of patients but unreasonable to assume that recurrence times within each patient are independent. One solution would be to fit a standard Cox model, adjusting the standard errors of the estimated hazard ratios to account for the possible correlation by specifying `vce(cluster patient)`.

We could instead model the correlation by assuming that the correlation is the result of a latent patient-level effect, or frailty. That is, rather than fitting a standard model and specifying `vce(cluster patient)`, we could fit a frailty model by specifying `shared(patient)`:

```
. stset time, fail(infect)
(output omitted)

. stcox age female, shared(patient)

        Failure _d: infect
  Analysis time _t: time

Fitting comparison Cox model ...

Estimating frailty variance:
Iteration 0:  Log profile likelihood = -182.06713
Iteration 1:  Log profile likelihood =  -181.9791
Iteration 2:  Log profile likelihood = -181.97453
Iteration 3:  Log profile likelihood = -181.97453

Fitting final Cox model:
Iteration 0:  Log likelihood = -199.05599
Iteration 1:  Log likelihood = -183.72296
Iteration 2:  Log likelihood = -181.99509
Iteration 3:  Log likelihood = -181.97455
Iteration 4:  Log likelihood = -181.97453
Refining estimates:
Iteration 0:  Log likelihood = -181.97453
```

```
Cox regression with Breslow method for ties
Gamma shared frailty                          Number of obs     =       76
Group variable: patient                       Number of groups  =       38
                                              Obs per group:
No. of subjects =    76                                    min =        2
No. of failures =    58                                    avg =        2
Time at risk    = 7,424                                    max =        2
                                              Wald chi2(2)      =    11.66
Log likelihood = -181.97453                   Prob > chi2       =   0.0029
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.006202	.0120965	0.51	0.607	.9827701	1.030192
female	.2068678	.095708	-3.41	0.001	.0835376	.5122756
theta	.4754497	.2673108				

```
LR test of theta=0: chibar2(01) = 6.27          Prob >= chibar2 = 0.006
```
Note: Standard errors of hazard ratios are conditional on theta.

From the output, we obtain $\widehat{\theta} = 0.475$, and given the standard error of $\widehat{\theta}$ and likelihood-ratio test of H_0: $\theta = 0$, we find a significant frailty effect, meaning that the correlation within patient cannot be ignored. Contrast this with the analysis of the same data in [ST] **streg**, which considered both Weibull and lognormal shared-frailty models. For Weibull, there was significant frailty; for lognormal, there was not.

The estimated ν_i are not displayed in the coefficient table but may be retrieved postestimation by using `predict` with the `effects` option; see [ST] **stcox postestimation** for an example.

◁

In shared-frailty Cox models, the estimation consists of two steps. In the first step, the optimization is in terms of θ only. For fixed θ, the second step consists of fitting a standard Cox model via penalized log likelihood, with the ν_i introduced as estimable coefficients of dummy variables identifying the groups. The penalty term in the penalized log likelihood is a function of θ; see *Methods and formulas*. The final estimate of θ is taken to be the one that maximizes the penalized log likelihood. Once the optimal θ is obtained, it is held fixed, and a final penalized Cox model is fit. As a result, the standard errors of the main regression parameters (or hazard ratios, if displayed as such) are treated as conditional on θ fixed at its optimal value.

With gamma-distributed frailty, hazard ratios decay over time in favor of the *frailty effect* and thus the displayed "Haz. ratio" in the above output is actually the hazard ratio only for $t = 0$. The degree of decay depends on θ. Should the estimated θ be close to 0, the hazard ratios do regain their usual interpretation; see Gutierrez (2002) for details.

❑ Technical note

The likelihood-ratio test of $\theta = 0$ is a boundary test and thus requires careful consideration concerning the calculation of its p-value. In particular, the null distribution of the likelihood-ratio test statistic is not the usual χ_1^2 but is rather a 50:50 mixture of a χ_0^2 (point mass at zero) and a χ_1^2, denoted as $\overline{\chi}_{01}^2$. See Gutierrez, Carter, and Drukker (2001) for more details.

◻

❑ Technical note

In [ST] **streg**, shared-frailty models are compared and contrasted with *unshared* frailty models. Unshared-frailty models are used to model heterogeneity, and the frailties are integrated out of the conditional survivor function to produce an unconditional survivor function, which serves as a basis for all likelihood calculations.

Given the nature of Cox regression (the baseline hazard remains unspecified), there is no Cox regression analog to the unshared parametric frailty model as fit using streg. That is not to say that you cannot fit a shared-frailty model with 1 observation per group; you can as long as you do not fit a null model.

◻

Stored results

stcox stores the following in e():

Scalars

e(N)	number of observations
e(N_sub)	number of subjects
e(N_fail)	number of failures
e(N_g)	number of groups
e(df_m)	model degrees of freedom
e(r2_p)	pseudo-R^2
e(ll)	log likelihood
e(ll_0)	log likelihood, constant-only model
e(ll_c)	log likelihood, comparison model
e(N_clust)	number of clusters
e(chi2)	χ^2
e(chi2_c)	χ^2, comparison test
e(risk)	total time at risk
e(g_min)	smallest group size
e(g_avg)	average group size
e(g_max)	largest group size
e(theta)	frailty parameter
e(se_theta)	standard error of θ
e(p_c)	p-value for comparison test
e(rank)	rank of e(V)
e(converged)	1 if converged, 0 otherwise

Macros
e(cmd)	cox or stcox_fr
e(cmd2)	stcox
e(cmdline)	command as typed
e(depvar)	_t
e(t0)	_t0
e(wtype)	weight type
e(wexp)	weight expression
e(texp)	function of time used for covariates from option tvc()
e(ties)	method used for handling ties
e(strata)	strata variables
e(shared)	frailty grouping variable
e(clustvar)	name of cluster variable
e(offset)	linear offset variable
e(chi2type)	Wald or LR; type of model χ^2 test
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(method)	requested estimation method
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(footnote)	program used to implement the footnote display
e(marginsnotok)	predictions disallowed by margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices
e(b)	coefficient vector
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance estimators

Functions
e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices
r(table)	matrix containing the coefficients with their standard errors, test statistics, *p*-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

The proportional hazards model with explanatory variables was first suggested by Cox (1972). For an introductory explanation, see Hosmer, Lemeshow, and May (2008, chap. 3, 4, and 7), Kahn and Sempos (1989, 193–198), and Selvin (2004, 412–442). For an introduction for the social scientist, see Box-Steffensmeier and Jones (2004, chap. 4). For a comprehensive review of the methods in this entry, see Klein and Moeschberger (2003). For a detailed development of these methods, see Kalbfleisch and Prentice (2002). For more Stata-specific insight, see Cleves, Gould, and Marchenko (2016), Dupont (2009), and Vittinghoff et al. (2012).

Let $\mathbf{x}_i$ be the row vector of covariates for the time interval $(t_{0i}, t_i]$ for the ith observation in the dataset $i = 1, \ldots, N$. stcox obtains parameter estimates, $\widehat{\boldsymbol{\beta}}$, by maximizing the partial log-likelihood function

$$\log L = \sum_{j=1}^{D} \left[\sum_{i \in D_j} \mathbf{x}_i \boldsymbol{\beta} - d_j \log \left\{ \sum_{k \in R_j} \exp(\mathbf{x}_k \boldsymbol{\beta}) \right\} \right]$$

where j indexes the ordered failure times $t_{(j)}$, $j = 1, \ldots, D$; D_j is the set of d_j observations that fail at $t_{(j)}$; d_j is the number of failures at $t_{(j)}$; and R_j is the set of observations k that are at risk at time $t_{(j)}$ (that is, all k such that $t_{0k} < t_{(j)} \leq t_k$). This formula for $\log L$ is for unweighted data and handles ties by using the Peto–Breslow approximation (Peto 1972; Breslow 1974), which is the default method of handling ties in `stcox`.

If `strata(`*varnames*`)` is specified, then the partial log likelihood is the sum of each stratum-specific partial log likelihood, obtained by forming the ordered failure times $t_{(j)}$, the failure sets D_j, and the risk sets R_j, using only those observations within that stratum.

The variance of $\widehat{\boldsymbol{\beta}}$ is estimated by the conventional inverse matrix of (negative) second derivatives of $\log L$, unless `vce(robust)` is specified, in which case the method of Lin and Wei (1989) is used. That method treats efficient score residuals as analogs to the log-likelihood scores one would find in fully parametric models; see *Methods and formulas* in [ST] **stcox postestimation** for how to calculate efficient score residuals. If `vce(cluster `*clustvar*`)` is specified, the efficient score residuals are summed within cluster before the sandwich (robust) estimator is applied.

Tied values are handled using one of four approaches. The log likelihoods corresponding to the four approaches are given with weights (`exactp` and `efron` do not allow weights) and offsets by

$$\log L_{\text{breslow}} = \sum_{j=1}^{D} \sum_{i \in D_j} \left[\widetilde{w}_i(\mathbf{x}_i\boldsymbol{\beta} + \text{offset}_i) - \widetilde{w}_i \log \left\{ \sum_{\ell \in R_j} \widetilde{w}_\ell \exp(\mathbf{x}_\ell\boldsymbol{\beta} + \text{offset}_\ell) \right\} \right]$$

$$\log L_{\text{efron}} = \sum_{j=1}^{D} \sum_{i \in D_j} \left[\mathbf{x}_i\boldsymbol{\beta} + \text{offset}_i - d_j^{-1} \sum_{k=0}^{d_j-1} \log \left\{ \sum_{\ell \in R_j} \exp(\mathbf{x}_\ell\boldsymbol{\beta} + \text{offset}_\ell) - kA_j \right\} \right]$$

$$A_j = d_j^{-1} \sum_{\ell \in D_j} \exp(\mathbf{x}_\ell\boldsymbol{\beta} + \text{offset}_\ell)$$

$$\log L_{\text{exactm}} = \sum_{j=1}^{D} \log \int_0^\infty \prod_{\ell \in D_j} \left\{ 1 - \exp\left(-\frac{e_\ell}{s}t\right) \right\}^{w_\ell} \exp(-t)dt$$

$$e_\ell = \exp(\mathbf{x}_\ell\boldsymbol{\beta} + \text{offset}_\ell)$$

$$s = \sum_{\substack{k \in R_j \\ k \notin D_j}} w_k \exp(\mathbf{x}_k\boldsymbol{\beta} + \text{offset}_k) = \text{sum of weighted nondeath risk scores}$$

$$\log L_{\text{exactp}} = \sum_{j=1}^{D} \left\{ \sum_{i \in R_j} \delta_{ij}(\mathbf{x}_i\boldsymbol{\beta} + \text{offset}_i) - \log f(r_j, d_j) \right\}$$

$$f(r, d) = f(r-1, d) + f(r-1, d-1)\exp(\mathbf{x}_k\boldsymbol{\beta} + \text{offset}_k)$$

$$k = r\text{th observation in the set } R_j$$

$$r_j = \text{cardinality of the set } R_j$$

$$f(r, d) = \begin{cases} 0 & \text{if } r < d \\ 1 & \text{if } d = 0 \end{cases}$$

where δ_{ij} is an indicator for failure of observation i at time $t_{(j)}$ and w_i are the weights. In the log likelihood for the Breslow method, $\widetilde{w}_i = w_i \times N / \sum w_i$ when the model is fit using probability weights, and $\widetilde{w}_i = w_i$ when the model is fit using frequency weights or importance weights.

Calculations for the exact marginal log likelihood (and associated derivatives) are obtained with 15-point Gauss–Laguerre quadrature. The `breslow` and `efron` options both provide approximations of the exact marginal log likelihood. The `efron` approximation is a better (closer) approximation, but the `breslow` approximation is faster. The choice of the approximation to use in a given situation should generally be driven by the proportion of ties in the data.

For shared-frailty models, the data are organized into G groups with the ith group consisting of n_i observations, $i = 1, \ldots, G$. From Therneau and Grambsch (2000, 253–255), estimation of θ takes place via maximum profile log likelihood. For fixed θ, estimates of $\boldsymbol{\beta}$ and $\nu_1, \ldots, \nu_G$ are obtained by maximizing

$$
\log L(\theta) = \log L_{\mathrm{Cox}}(\boldsymbol{\beta}, \nu_1, \ldots, \nu_G) + \sum_{i=1}^{G} \left[\frac{1}{\theta} \{ \nu_i - \exp(\nu_i) \} + \right.
$$

$$
\left. \left(\frac{1}{\theta} + D_i \right) \left\{ 1 - \log \left(\frac{1}{\theta} + D_i \right) \right\} - \frac{\log \theta}{\theta} + \log \Gamma \left(\frac{1}{\theta} + D_i \right) - \log \Gamma \left(\frac{1}{\theta} \right) \right]
$$

where D_i is the number of death events in group i, and $\log L_{\mathrm{Cox}}(\boldsymbol{\beta}, \nu_1, \ldots, \nu_G)$ is the standard Cox partial log likelihood, with the ν_i treated as the coefficients of indicator variables identifying the groups. That is, the jth observation in the ith group has log relative hazard $\mathbf{x}_{ij}\boldsymbol{\beta} + \nu_i$. The estimate of the frailty parameter, $\widehat{\theta}$, is chosen as that which maximizes $\log L(\theta)$. The final estimates of $\boldsymbol{\beta}$ are obtained by maximizing $\log L(\widehat{\theta})$ in $\boldsymbol{\beta}$ and the ν_i. The ν_i are not reported in the coefficient table but are available via `predict`; see [ST] **stcox postestimation**. The estimated variance–covariance matrix of $\widehat{\boldsymbol{\beta}}$ is obtained as the appropriate submatrix of the variance matrix of $(\widehat{\boldsymbol{\beta}}, \widehat{\nu}_1, \ldots, \widehat{\nu}_G)$, and that matrix is obtained as the inverse of the negative Hessian of $\log L(\widehat{\theta})$. Therefore, standard errors and inference based on $\widehat{\boldsymbol{\beta}}$ should be treated as conditional on $\theta = \widehat{\theta}$.

The likelihood-ratio test statistic for testing $H_0: \theta = 0$ is calculated as minus twice the difference between the log likelihood for a Cox model without shared frailty and $\log L(\widehat{\theta})$ evaluated at the final $(\widehat{\boldsymbol{\beta}}, \widehat{\nu}_1, \ldots, \widehat{\nu}_G)$.

David Roxbee Cox (1924–2022) was born in Birmingham, England. He earned master's and PhD degrees in mathematics and statistics from the universities of Cambridge and Leeds, and he worked at the Royal Aircraft Establishment, the Wool Industries Research Association, and the universities of Cambridge, London (Birkbeck and Imperial Colleges), and Oxford. He was knighted in 1985. Sir David has worked on a wide range of theoretical and applied statistical problems, with outstanding contributions in areas such as experimental design, stochastic processes, binary data, survival analysis, asymptotic techniques, and multivariate dependencies. In 2010, Sir David was awarded the Copley Medal, the Royal Society's highest honor. In 2017, he was the first recipient of the International Prize in Statistics.

Acknowledgment

We thank Peter Sasieni of King's College London for his statistical advice and guidance in implementing the robust variance estimator for this command.

References

Andersson, T. M.-L., and P. C. Lambert. 2012. Fitting and modeling cure in population-based cancer studies within the framework of flexible parametric survival models. *Stata Journal* 12: 623–638.

Bower, H., M. J. Crowther, and P. C. Lambert. 2016. strcs: A command for fitting flexible parametric survival models on the log-hazard scale. *Stata Journal* 16: 989–1012.

Box-Steffensmeier, J. M., and B. S. Jones. 2004. *Event History Modeling: A Guide for Social Scientists*. Cambridge: Cambridge University Press.

Breslow, N. E. 1974. Covariance analysis of censored survival data. *Biometrics* 30: 89–99. https://doi.org/10.2307/2529620.

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata Press.

Cox, D. R. 1972. Regression models and life-tables (with discussion). *Journal of the Royal Statistical Society, Series B* 34: 187–220. https://doi.org/10.1111/j.2517-6161.1972.tb00899.x.

———. 1975. Partial likelihood. *Biometrika* 62: 269–276. https://doi.org/10.1093/biomet/62.2.269.

Cox, D. R., and D. Oakes. 1984. *Analysis of Survival Data*. London: Chapman and Hall/CRC.

Cox, D. R., and E. J. Snell. 1968. A general definition of residuals (with discussion). *Journal of the Royal Statistical Society, Series B* 30: 248–275. https://doi.org/10.1111/j.2517-6161.1968.tb00724.x.

Cronin, A., L. Tian, and H. Uno. 2016. strmst2 and strmst2pw: New commands to compare survival curves using the restricted mean survival time. *Stata Journal* 16: 702–716.

Crowley, J., and M. Hu. 1977. Covariance analysis of heart transplant survival data. *Journal of the American Statistical Association* 72: 27–36. https://doi.org/10.2307/2286902.

Cui, J. 2005. Buckley–James method for analyzing censored data, with an application to a cardiovascular disease and an HIV/AIDS study. *Stata Journal* 5: 517–526.

Dupont, W. D. 2009. *Statistical Modeling for Biomedical Researchers: A Simple Introduction to the Analysis of Complex Data*. 2nd ed. Cambridge: Cambridge University Press.

Earnest, A. 2017. *Essentials of a Successful Biostatistical Collaboration*. Boca Raton, FL: CRC Press.

Fleming, T. R., and D. P. Harrington. 1991. *Counting Processes and Survival Analysis*. New York: Wiley.

Gutierrez, R. G. 2002. Parametric frailty and shared frailty survival models. *Stata Journal* 2: 22–44.

Gutierrez, R. G., S. L. Carter, and D. M. Drukker. 2001. sg160: On boundary-value likelihood-ratio tests. *Stata Technical Bulletin* 60: 15–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 269–273. College Station, TX: Stata Press.

Hinchliffe, S. R., D. A. Scott, and P. C. Lambert. 2013. Flexible parametric illness-death models. *Stata Journal* 13: 759–775.

Hosmer, D. W., Jr., S. A. Lemeshow, and S. May. 2008. *Applied Survival Analysis: Regression Modeling of Time to Event Data*. 2nd ed. New York: Wiley.

Kahn, H. A., and C. T. Sempos. 1989. *Statistical Methods in Epidemiology*. New York: Oxford University Press.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd ed. New York: Springer.

Lambert, P. C. 2017. The estimation and modeling of cause-specific cumulative incidence functions using time-dependent weights. *Stata Journal* 17: 181–207.

Lambert, P. C., and P. Royston. 2009. Further development of flexible parametric models for survival analysis. *Stata Journal* 9: 265–290.

Lin, D. Y., and L. J. Wei. 1989. The robust inference for the Cox proportional hazards model. *Journal of the American Statistical Association* 84: 1074–1078. https://doi.org/10.2307/2290085.

McGilchrist, C. A., and C. W. Aisbett. 1991. Regression with frailty in survival analysis. *Biometrics* 47: 461–466. https://doi.org/10.2307/2532138.

Newman, S. C. 2001. *Biostatistical Methods in Epidemiology*. New York: Wiley.

Overgaard, M., P. K. Andersen, and E. T. Parner. 2015. Regression analysis of censored data using pseudo-observations: An update. *Stata Journal* 15: 809–821.

Parner, E. T., and P. K. Andersen. 2010. Regression analysis of censored data using pseudo-observations. *Stata Journal* 10: 408–422.

Peto, R. 1972. Contribution to the discussion of paper by D. R. Cox. *Journal of the Royal Statistical Society, Series B* 34: 205–207. https://doi.org/10.1111/j.2517-6161.1972.tb00900.x.

Reid, N. M. 1994. A conversation with Sir David Cox. *Statistical Science* 9: 439–455. https://doi.org/10.1214/ss/1177010394.

Royston, P. 2001. Flexible parametric alternatives to the Cox model, and more. *Stata Journal* 1: 1–28.

———. 2006. Explained variation for survival models. *Stata Journal* 6: 83–96.

———. 2007. Profile likelihood for estimation and confidence intervals. *Stata Journal* 7: 376–387.

———. 2014. Tools for checking calibration of a Cox model in external validation: Approach based on individual event probabilities. *Stata Journal* 14: 738–755.

———. 2015. Tools for checking calibration of a Cox model in external validation: Prediction of population-averaged survival curves based on risk groups. *Stata Journal* 15: 275–291.

———. 2017. A combined test for a generalized treatment effect in clinical trials with a time-to-event outcome. *Stata Journal* 17: 405–421.

Royston, P., and P. C. Lambert. 2011. *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model*. College Station, TX: Stata Press.

Royston, P., and M. K. B. Parmar. 2002. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in Medicine* 21: 2175–2197. https://doi.org/10.1002/sim.1203.

Ruhe, C. 2019. Bootstrap pointwise confidence intervals for covariate-adjusted survivor functions in the Cox model. *Stata Journal* 19: 185–199.

Schoenfeld, D. A. 1982. Partial residuals for the proportional hazards regression model. *Biometrika* 69: 239–241. https://doi.org/10.2307/2335876.

Selvin, S. 2004. *Statistical Analysis of Epidemiologic Data*. 3rd ed. New York: Oxford University Press.

Sterne, J. A. C., and K. Tilling. 2002. G-estimation of causal effects, allowing for time-varying confounding. *Stata Journal* 2: 164–182.

Suárez, E. L., C. M. Pérez, G. M. Nogueras, and C. Moreno-Gorrin. 2016. *Biostatistics in Public Health Using Stata*. Boca Raton, FL: CRC Press.

Therneau, T. M., and P. M. Grambsch. 2000. *Modeling Survival Data: Extending the Cox Model*. New York: Springer.

Vittinghoff, E., D. V. Glidden, S. C. Shiboski, and C. E. McCulloch. 2012. *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. 2nd ed. New York: Springer.

Xu, Y., and Y. B. Cheung. 2015. Frailty models and frailty-mixture models for recurrent event times. *Stata Journal* 15: 135–154.

———. 2018. Frailty models and frailty-mixture models for recurrent event times: Update. *Stata Journal* 18: 477–484.

Also see

Title

> **stcox PH-assumption tests** — Tests of proportional-hazards assumption after stcox

Description

stphplot plots $-\ln\{-\ln(\text{survival})\}$ curves for each category of a nominal or ordinal covariate versus $\ln(\text{analysis time})$ for right-censored data. These are often referred to as "log–log" plots. Optionally, these estimates can be adjusted for covariates. The proportional-hazards assumption is not violated when the curves are parallel.

stcoxkm plots Kaplan–Meier observed survival curves and compares them with the Cox predicted curves for the same variable. The closer the observed values are to the predicted, the less likely it is that the proportional-hazards assumption has been violated.

estat phtest tests the proportional-hazards assumption on the basis of Schoenfeld residuals after fitting a model with stcox.

Quick start

Log–log plot of survival

Check for parallel lines in plot of $-\ln\{-\ln(\text{survival})\}$ versus $\ln(\text{analysis time})$ for each category of covariate a using stset data

    stphplot, by(a)

Same as above, but adjust for average values of covariates x1 and x2

    stphplot, by(a) adjustfor(x1 x2)

Same as above

    stphplot, by(a) adjustfor(x1 x2, atomeans)

Adjust for $x1 = 0$ and $x2 = 0$

    stphplot, by(a) adjustfor(x1 x2, atzeros)

Kaplan–Meier and predicted survival plot

Compare Kaplan–Meier survival curve with predicted survival from Cox model for each category of covariate a using stset data

    stcoxkm, by(a)

Same as above, but create separate plots for each level of a

    stcoxkm, by(a) separate

Test using Schoenfeld residuals

Test the proportional-hazards assumption after stcox x1 x2 x3

    estat phtest

Same as above, and report separate test for each covariate

    estat phtest, detail

Menu

stphplot

Statistics > Survival analysis > Regression models > Graphically assess PH assumption

stcoxkm

Statistics > Survival analysis > Regression models > Kaplan−Meier versus predicted survival

estat phtest

Statistics > Survival analysis > Regression models > Test PH assumption

Syntax

Check proportional-hazards assumption:

Log–log plot of survival

stphplot $\begin{bmatrix} if \end{bmatrix}$, $\{$by(*varname*)│strata(*varname*)$\}$ $\begin{bmatrix} stphplot\_options \end{bmatrix}$

Kaplan–Meier and predicted survival plot

stcoxkm $\begin{bmatrix} if \end{bmatrix}$, by(*varname*) $\begin{bmatrix} stcoxkm\_options \end{bmatrix}$

Using Schoenfeld residuals

estat phtest $\begin{bmatrix} , phtest\_options \end{bmatrix}$

stphplot_options	Description
Main	
* by(*varname*)	fit separate Cox models; the default
* strata(*varname*)	fit stratified Cox model; requires adjustfor()
adjustfor(*varlist*$\begin{bmatrix} , suboptions \end{bmatrix}$)	adjust the estimates to specific values of *varlist*;
	default is overall means
Options	
nonegative	plot ln$\{-\ln(\text{survival})\}$
nolntime	plot curves against analysis time
noshow	do not show st setting information
Plot	
plot#opts(*stphplot_plot_options*)	affect rendition of the #th connected line and #th plotted points
Add plots	
addplot(*plot*)	add other plots to the generated graph
Y axis, X axis, Titles, Legend, Overall	
twoway_options	any options other than by() documented in
	[G-3] *twoway_options*

*Either by(*varname*) or strata(*varname*) is required with stphplot.

stphplot_plot_options	Description
cline_options	change look of lines or connecting method
marker_options	change look of markers (color, size, etc.)

stcoxkm_options	Description
Main	
* by(*varname*)	report the nominal or ordinal covariate
ties(breslow)	use Breslow method to handle tied failures
ties(efron)	use Efron method to handle tied failures
ties(exactm)	use exact marginal-likelihood method to handle tied failures
ties(exactp)	use exact partial-likelihood method to handle tied failures
separate	draw separate plot for predicted and observed curves
noshow	do not show st setting information
Observed plot	
obsopts(*stcoxkm_plot_options*)	affect rendition of the observed curve
obs#opts(*stcoxkm_plot_options*)	affect rendition of the #th observed curve; not allowed with separate
Predicted plot	
predopts(*stcoxkm_plot_options*)	affect rendition of the predicted curve
pred#opts(*stcoxkm_plot_options*)	affect rendition of the #th predicted curve; not allowed with separate
Add plots	
addplot(*plot*)	add other plots to the generated graph
Y axis, X axis, Titles, Legend, Overall	
twoway_options	any options other than by() documented in [G-3] *twoway_options*
byopts(*byopts*)	how subgraphs are combined, labeled, etc.

 * by(*varname*) is required with stcoxkm.

stcoxkm_plot_options	Description
connect_options	change look of connecting method
marker_options	change look of markers (color, size, etc.)

You must stset your data before using stphplot and stcoxkm; see [ST] **stset**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

phtest_options	Description

[Main]
log	use natural logarithm time-scaling function
km	use $1 - $ KM product-limit estimate as the time-scaling function
rank	use rank of analysis time as the time-scaling function
time(*varname*)	use *varname* containing a monotone transformation of analysis time as the time-scaling function
plot(*varname*)	plot smoothed, scaled Schoenfeld residuals versus time
bwidth(*#*)	use bandwidth of *#*; default is bwidth(0.8)
detail	test proportional-hazards assumption separately for each covariate

Scatterplot
marker_options	change look of markers (color, size, etc.)
marker_label_options	add marker labels; change look or position

Smoothed line
lineopts(*cline_options*)	affect rendition of the smoothed line

Y axis, X axis, Titles, Legend, Overall
twoway_options	any options other than by() documented in [G-3] *twoway_options*

estat phtest is not appropriate with svy estimation results.

collect is allowed with estat phtest; see [U] **11.1.10 Prefix commands**.

Options

Options are presented under the following headings:

> *Options for stphplot*
> *Options for stcoxkm*
> *Options for estat phtest*

Options for stphplot

[Main]

by(*varname*) specifies the nominal or ordinal covariate. Either by() or strata() is required with
   stphplot.

strata(*varname*) is an alternative to by(). Rather than fitting separate Cox models for each value
   of *varname*, strata() fits one stratified Cox model. You must also specify adjustfor() with
   the strata() option; see [ST] **sts graph**.

adjustfor(*varlist* [, *suboptions*]) adjusts the estimates of the survivor function to specific values
   of *varlist*. The default is to adjust to overall mean values of covariates. adjustfor() can be
   specified with by(); it is required with strata().

   suboptions are atomeans (the default), atmeans, atzeros, atbase, and at(); see [ST] *adjust-
   for_option*.

nonegative specifies that $\ln\{-\ln(\text{survival})\}$ be plotted instead of $-\ln\{-\ln(\text{survival})\}$.

nolntime specifies that curves be plotted against analysis time instead of against ln(analysis time).

noshow prevents stphplot from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

Plot

plot#opts(*stphplot_plot_options*) affects the rendition of the #th connected line and #th plotted points; see [G-3] *cline_options* and [G-3] *marker_options*.

Add plots

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] *addplot_option*.

Y axis, X axis, Titles, Legend, Overall

twoway_options are any of the options documented in [G-3] *twoway_options*, excluding by(). These include options for titling the graph (see [G-3] *title_options*) and for saving the graph to disk (see [G-3] *saving_option*).

Options for stcoxkm

Main

by(*varname*) specifies the nominal or ordinal covariate. by() is required.

ties(breslow | efron | exactm | exactp) specifies one of the methods available to stcox for handling tied failures. If none is specified, ties(breslow) is assumed; see [ST] **stcox**.

separate produces separate plots of predicted and observed values for each value of the variable specified with by().

noshow prevents stcoxkm from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

Observed plot

obsopts(*stcoxkm_plot_options*) affects the rendition of the observed curve; see [G-3] *connect_options* and [G-3] *marker_options*.

obs#opts(*stcoxkm_plot_options*) affects the rendition of the #th observed curve; see [G-3] *connect_options* and [G-3] *marker_options*. This option is not allowed with separate.

Predicted plot

predopts(*stcoxkm_connect_options*) affects the rendition of the predicted curve; see [G-3] *connect_options* and [G-3] *marker_options*.

pred#opts(*stcoxkm_connect_options*) affects the rendition of the #th predicted curve; see [G-3] *connect_options* and [G-3] *marker_options*. This option is not allowed with separate.

Add plots

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] ***addplot_option***.

Y axis, X axis, Titles, Legend, Overall

twoway_options are any of the options documented in [G-3] ***twoway_options***, excluding by(). These include options for titling the graph (see [G-3] ***title_options***) and for saving the graph to disk (see [G-3] ***saving_option***).

byopts(*byopts*) affects the appearance of the combined graph when by() and separate are specified, including the overall graph title and the organization of subgraphs. See [G-3] ***by_option***.

Options for estat phtest

Main

log, km, rank, and time() are used to specify the time scaling function.

By default, estat phtest performs the tests using the identity function, that is, analysis time itself.

log specifies that the natural log of analysis time be used.

km specifies that 1 minus the Kaplan–Meier product-limit estimate be used.

rank specifies that the rank of analysis time be used.

time(*varname*) specifies a variable containing an arbitrary monotonic transformation of analysis time. You must ensure that *varname* is a monotonic transform.

plot(*varname*) specifies that a scatterplot and smoothed plot of scaled Schoenfeld residuals versus time be produced for the covariate specified by *varname*. By default, the smoothing is performed using the running-mean method implemented in lowess, mean noweight; see [R] **lowess**.

bwidth(#) specifies the bandwidth. Centered subsets of bwidth() $\times N$ observations are used for calculating smoothed values for each point in the data except for endpoints, where smaller, uncentered subsets are used. The greater the bwidth(), the greater the smoothing. The default is bwidth(0.8).

detail specifies that a separate test of the proportional-hazards assumption be produced for each covariate in the Cox model. By default, estat phtest produces only the global test.

Scatterplot

marker_options affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G-3] ***marker_options***.

marker_label_options specify if and how the markers are to be labeled; see [G-3] ***marker_label_options***.

Smoothed line

lineopts(*cline_options*) affects the rendition of the smoothed line; see [G-3] ***cline_options***.

Y axis, X axis, Titles, Legend, Overall

twoway_options are any of the options documented in [G-3] ***twoway_options***, excluding by(). These include options for titling the graph (see [G-3] ***title_options***) and for saving the graph to disk (see [G-3] ***saving_option***).

Remarks and examples

Cox proportional hazards models assume that the hazard ratio is constant over time. Suppose that a group of cancer patients on an experimental treatment is monitored for 10 years. If the hazard of dying for the nontreated group is twice the rate as that of the treated group (HR = 2.0), the proportional-hazards assumption implies that this ratio is the same at 1 year, at 2 years, or at any point on the time scale. Because the Cox model, by definition, is constrained to follow this assumption, it is important to evaluate its validity. If the assumption fails, alternative modeling choices would be more appropriate (for example, a stratified Cox model, time-varying covariates). For examples of testing the proportional-hazards assumption using Stata, see Allison (2014).

stphplot and stcoxkm provide graphical methods for assessing violations of the proportional-hazards assumption. Although using graphs to assess the validity of the assumption is subjective, it can be a helpful tool.

stphplot plots $-\ln\{-\ln(\text{survival})\}$ curves for each category of a nominal or ordinal covariate versus $\ln(\text{analysis time})$. These are often referred to as "log–log" plots. Optionally, these estimates can be adjusted for covariates. If the plotted lines are reasonably parallel, the proportional-hazards assumption has not been violated, and it would be appropriate to base the estimate for that variable on one baseline survivor function.

Another graphical method of evaluating the proportional-hazards assumption, though less common, is to plot the Kaplan–Meier observed survival curves and compare them with the Cox predicted curves for the same variable. This plot is produced with stcoxkm. When the predicted and observed curves are close together, the proportional-hazards assumption has not been violated. See Garrett (1997) for more details.

Many popular tests for proportional hazards are, in fact, tests of nonzero slope in a generalized linear regression of the scaled Schoenfeld residuals on time (see Grambsch and Therneau [1994]). The estat phtest command tests, for individual covariates and globally, the null hypothesis of zero slope, which is equivalent to testing that the log hazard-ratio function is constant over time. Thus rejection of the null hypothesis of a zero slope indicates deviation from the proportional-hazards assumption. The estat phtest command allows three common time-scaling options (log, km, and rank) and also allows you to specify a user-defined function of time through the time() option. When no option is specified, the tests are performed using analysis time without further transformation.

▷ Example 1

These examples use data from a leukemia remission study (Garrett 1997). The data consist of 42 patients who are monitored over time to see how long (weeks) it takes them to go out of remission (relapse: 1 = yes, 0 = no). Half the patients receive a new experimental drug, and the other half receive a standard drug (treatment1: 1 = drug A, 0 = standard). White blood cell count, a strong indicator of the presence of leukemia, is divided into three categories (wbc3cat: 1 = normal, 2 = moderate, 3 = high).

```
. use https://www.stata-press.com/data/r18/leukemia
(Leukemia remission study)

. describe

Contains data from https://www.stata-press.com/data/r18/leukemia.dta
 Observations:              42                  Leukemia remission study
    Variables:               8                  23 Mar 2022 10:39
─────────────────────────────────────────────────────────────────────────────
Variable        Storage   Display    Value
    name           type    format    label      Variable label
─────────────────────────────────────────────────────────────────────────────
weeks            byte     %8.0g                 Weeks in remission
relapse          byte     %8.0g      yesno      Relapse
treatment1       byte     %8.0g      trt1lbl    Treatment I
treatment2       byte     %8.0g      trt2lbl    Treatment II
wbc3cat          byte     %9.0g      wbclbl     White blood cell count
wbc1             byte     %8.0g                 wbc3cat==Normal
wbc2             byte     %8.0g                 wbc3cat==Moderate
wbc3             byte     %8.0g                 wbc3cat==High
─────────────────────────────────────────────────────────────────────────────
Sorted by: weeks

. stset weeks, failure(relapse)

Survival-time data settings

         Failure event: relapse!=0 & relapse<.
Observed time interval: (0, weeks]
     Exit on or before: failure

─────────────────────────────────────────────────────────────────────────────
        42  total observations
         0  exclusions
─────────────────────────────────────────────────────────────────────────────
        42  observations remaining, representing
        30  failures in single-record/single-failure data
       541  total analysis time at risk and under observation
                                            At risk from t =          0
                                 Earliest observed entry t =          0
                                    Last observed exit t =           35
```

In this example, we examine whether the proportional-hazards assumption holds for drug A versus the standard drug (treatment1). First, we will use stphplot, followed by stcoxkm.

```
. stphplot, by(treatment1)
        Failure _d: relapse
  Analysis time _t: weeks
```
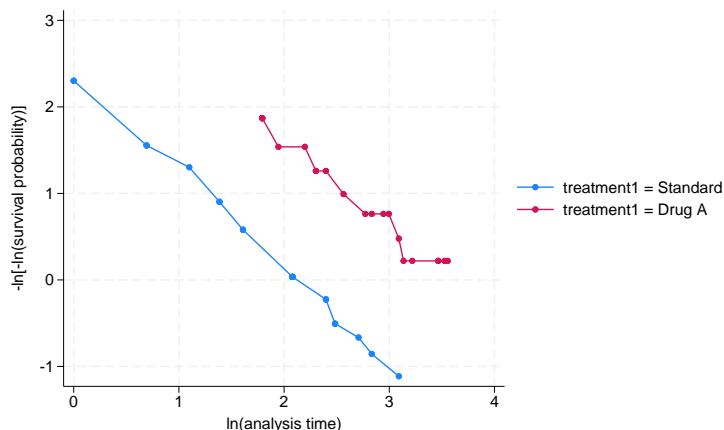

Figure 1.

```
. stcoxkm, by(treatment1) legend(pos(6) cols(2))
        Failure _d: relapse
  Analysis time _t: weeks
```
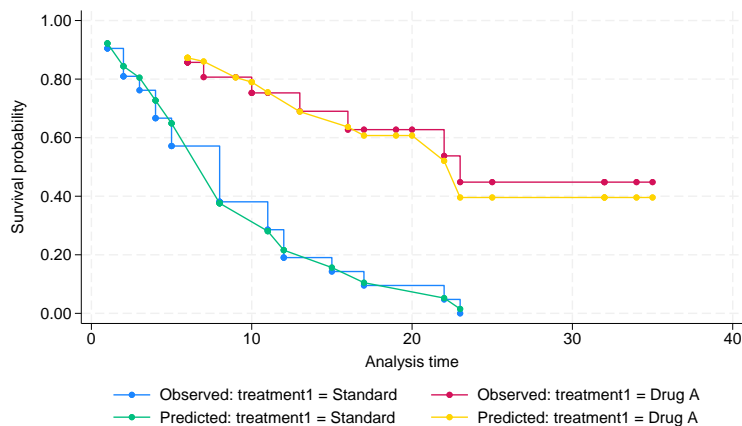

Figure 2.

Figure 1 (`stphplot`) displays lines that are parallel, implying that the proportional-hazards assumption for `treatment1` has not been violated. This is confirmed in figure 2 (`stcoxkm`), where the observed values and predicted values are close together.

The graph in figure 3 is the same as the one in figure 1, adjusted for white blood cell count. By default, this adjustment sets each level of `wbc3cat` to its overall mean. In other words, the results are adjusted based on the observed proportions of individuals having normal, moderate, and high white blood cell counts.

```
. stphplot, strata(treatment1) adjustfor(i.wbc3cat)
        Failure _d: relapse
  Analysis time _t: weeks
```
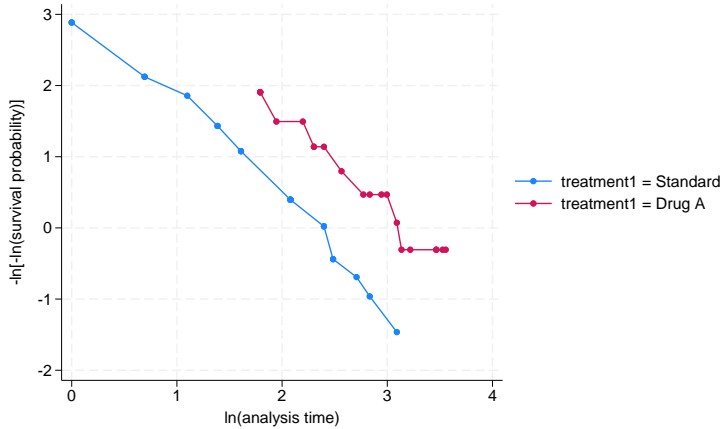

Figure 3.

The lines in figure 3 are still parallel, although they are somewhat closer together. Examining the proportional-hazards assumption on a variable without adjusting for covariates is usually adequate as a diagnostic tool before using the Cox model. However, if you know that adjustment for covariates in a final model is necessary, you may wish to reexamine whether the proportional-hazards assumption still holds.

If we wanted to adjust to the base level of the factor variable wbc3cat instead of the level-specific averages, we could have typed

```
. stphplot, strata(treatment1) adjustfor(i.wbc3cat, atbase)
```

Adjusting to a different value, however, would not affect our conclusion about the curves being parallel.

Another variable in this dataset measures a different drug (treatment2: $1 = $ drug B, $0 = $ standard). We wish to examine the proportional-hazards assumption for this variable.

```
. stphplot, by(treatment2)
          Failure _d: relapse
   Analysis time _t: weeks
```
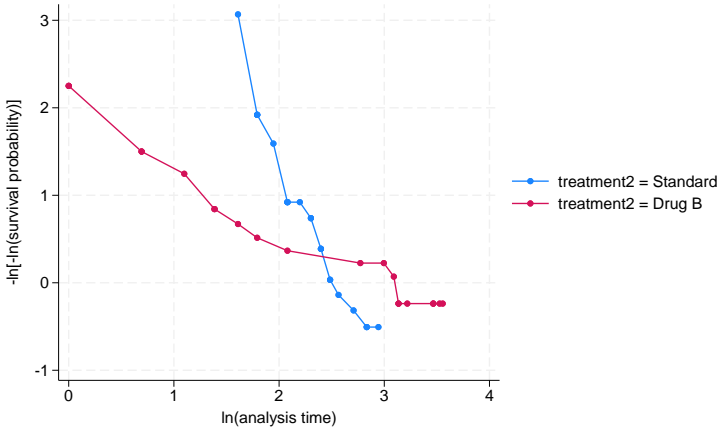

Figure 4.

```
. stcoxkm, by(treatment2) separate byopts(style(altleg))
          Failure _d: relapse
   Analysis time _t: weeks
```
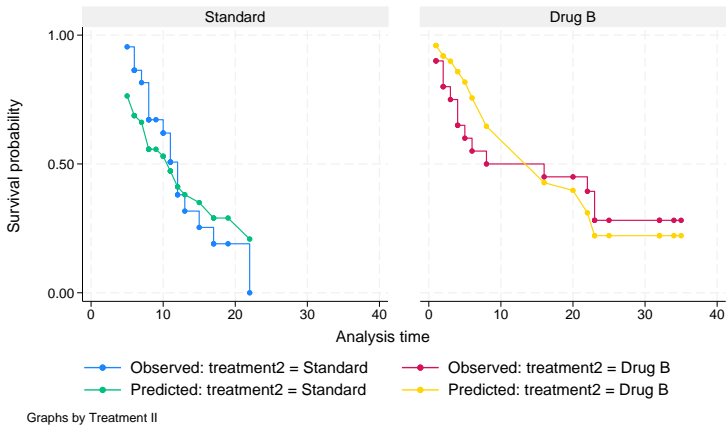

Figure 5.

This variable violates the proportional-hazards assumption. In figure 4, we see that the lines are not only nonparallel but also cross in the data region. In figure 5, we see that there are considerable differences between the observed and predicted values. We have overestimated the positive effect of drug B for the first half of the study and have underestimated it in the later weeks. One hazard ratio describing the effect of this drug would be inappropriate. We definitely would want to stratify on this variable in our Cox model.

◁

▷ Example 2: estat phtest

In this example, we use `estat phtest` to examine whether the proportional-hazards assumption holds for a model with covariates `treatment1` and `wbc3cat`. After `stsetting` the data, we first run `stcox` with these factor variables as regressors. Then we use `estat phtest`:

```
. stset weeks, failure(relapse)
Survival-time data settings

         Failure event: relapse!=0 & relapse<.
Observed time interval: (0, weeks]
     Exit on or before: failure

─────────────────────────────────────────────────────────────────────────────
         42  total observations
          0  exclusions
─────────────────────────────────────────────────────────────────────────────
         42  observations remaining, representing
         30  failures in single-record/single-failure data
        541  total analysis time at risk and under observation
                                            At risk from t =          0
                                 Earliest observed entry t =          0
                                   Last observed exit t =            35
```

```
. stcox i.treatment1 i.wbc3cat, nolog
        Failure _d: relapse
  Analysis time _t: weeks
Cox regression with Breslow method for ties
No. of subjects = 42                                  Number of obs =      42
No. of failures = 30
Time at risk    = 541
                                                      LR chi2(3)    =  33.02
Log likelihood = -77.476905                           Prob > chi2   = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
treatment1						
Drug A	.2834551	.1229874	-2.91	0.004	.1211042	.6634517
wbc3cat						
Moderate	3.637825	2.201306	2.13	0.033	1.111134	11.91015
High	10.92214	7.088783	3.68	0.000	3.06093	38.97284

```
. estat phtest, detail
Test of proportional-hazards assumption
Time function: Analysis time
```

	rho	chi2	df	Prob>chi2
0b.treatme~1	.	.	1	.
1.treatment1	-0.07019	0.15	1	0.6948
1b.wbc3cat	.	.	1	.
2.wbc3cat	-0.03223	0.03	1	0.8650
3.wbc3cat	0.01682	0.01	1	0.9237
Global test		0.33	3	0.9551

Because we specified the `detail` option with the `estat phtest` command, both covariate-specific and global tests were produced. In addition, the `rho` column reports the correlation between the scaled Schoenfeld residuals and the specified function of time. We can see that there is no evidence that the proportional-hazards assumption has been violated.

Another variable in this dataset measures a different drug (`treatment2`: 1 = drug B, 0 = standard). We now wish to examine the proportional-hazards assumption for the previous model by substituting `treatment2` for `treatment1`.

We fit a new Cox model and perform the test for proportional hazards:

```
. stcox i.treatment2 i.wbc3cat, nolog

        Failure _d: relapse
  Analysis time _t: weeks

Cox regression with Breslow method for ties

No. of subjects =   42                          Number of obs =      42
No. of failures =   30
Time at risk    = 541
                                                LR chi2(3)    =   23.93
Log likelihood = -82.019053                     Prob > chi2   =  0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
treatment2						
Drug B	.8483777	.3469054	-0.40	0.688	.3806529	1.890816
wbc3cat						
Moderate	3.409628	2.050784	2.04	0.041	1.048905	11.08353
High	14.0562	8.873693	4.19	0.000	4.078529	48.44314

```
. estat phtest, detail

Test of proportional-hazards assumption

Time function: Analysis time
```

	rho	chi2	df	Prob>chi2
0b.treatme~2	.	.	1	.
1.treatment2	-0.51672	10.19	1	0.0014
1b.wbc3cat	.	.	1	.
2.wbc3cat	-0.09860	0.29	1	0.5903
3.wbc3cat	-0.03559	0.04	1	0.8448
Global test		10.24	3	0.0166

`treatment2` violates the proportional-hazards assumption. A single hazard ratio describing the effect of this drug is inappropriate.

The test of the proportional-hazards assumption is based on the principle that, for a given regressor, the assumption restricts $\beta(t_j) = \beta$ for all t_j. This implies that a plot of $\beta(t_j)$ versus time will have a slope of zero. Grambsch and Therneau (1994) showed that $E(s_j^*) + \widehat{\beta} \approx \beta(t_j)$, where s_j^* is the scaled Schoenfeld residual at failure time t_j and $\widehat{\beta}$ is the estimated coefficient from the Cox model. Thus a plot of $s_j^* + \widehat{\beta}$ versus some function of time provides a graphical assessment of the assumption.

Continuing from above, if you type

```
. predict sch*, scaledsch
```

you obtain five variables—`sch1`, `sch2`, `sch3`, `sch4`, and `sch5`—corresponding to the regressors. Ignoring the base categories, `sch2` corresponds to `1.treatment2`, `sch4` corresponds to `2.wbc3cat`, and `sch5` corresponds to `3.wbc3cat`. Given the utility of $s_j^* + \widehat{\beta}$, what is stored in variable `sch2` is actually $s_{j2}^* + \widehat{\beta}_2$ and not just the scaled Schoenfeld residual for the `1.treatment2`, s_{j2}^*, itself. The

estimated coefficient, $\widehat{\beta}_2$, is added automatically. The same holds true for the variable representing the next regressor, $\text{sch4} = s_{j4}^* + \widehat{\beta}_4$, and so on.

As such, a graphical assessment of the proportional-hazards assumption for the first regressor is as simple as

```
. scatter sch2 _t || lfit sch2 _t
```

which plots a scatter of $s_{j2}^* + \widehat{\beta}_2$ versus analysis time, `_t`, and overlays a linear fit. Is the slope zero? The answer is no for `1.treatment2`, and that agrees with our results from `estat phtest`.

◁

❑ Technical note

The tests of the proportional-hazards assumption assume homogeneity of variance across risk sets. This allows the use of the estimated overall (pooled) variance–covariance matrix in the equations. Although these tests have been shown by Grambsch and Therneau (1994) to be fairly robust to departures from this assumption, exercise care where this assumption may not hold, particularly when performing a stratified Cox analysis. In such cases, we recommend that you check the proportional-hazards assumption separately for each stratum.

❑

Video example

How to fit a Cox proportional hazards model and check proportional-hazards assumption

Stored results

`estat phtest` stores the following in `r()`:

Scalars
    `r(df)`   global test degrees of freedom
    `r(chi2)`   global test χ^2
    `r(p)`   global test p-value

Matrices
    `r(phtest)`   separate tests for each covariate

Methods and formulas

For one covariate, x, the Cox proportional hazards model reduces to

$$h(t; x) = h_0(t) \exp(x\beta)$$

where $h_0(t)$ is the baseline hazard function from the Cox model. Let $S_0(t)$ and $H_0(t)$ be the corresponding Cox baseline survivor and baseline cumulative hazard functions, respectively.

The proportional-hazards assumption implies that

$$H(t) = H_0(t) \exp(x\beta)$$

or

$$\ln H(t) = \ln H_0(t) + x\beta$$

where $H(t)$ is the cumulative hazard function. Thus, under the proportional-hazards assumption, the logs of the cumulative hazard functions at each level of the covariate have equal slope. This is the basis for the method implemented in stphplot.

The proportional-hazards assumption also implies that

$$S(t) = S_0(t)^{\exp(x\beta)}$$

Let $\widehat{S}(t)$ be the estimated survivor function based on the Cox model. This function is a step function like the Kaplan–Meier estimate and, in fact, reduces to the Kaplan–Meier estimate when $x = 0$. Thus for each level of the covariate of interest, we can assess violations of the proportional-hazards assumption by comparing these survival estimates with estimates calculated independently of the model. See Kalbfleisch and Prentice (2002) or Hess (1995).

stcoxkm plots Kaplan–Meier estimated curves for each level of the covariate together with the Cox model predicted baseline survival curve. The closer the observed values are to the predicted values, the less likely it is that the proportional-hazards assumption has been violated.

Grambsch and Therneau (1994) presented a scaled adjustment for the Schoenfeld residuals that permits the interpretation of the smoothed residuals as a nonparametric estimate of the log hazard-ratio function. These scaled Schoenfeld residuals, $\mathbf{r}^*_{S_i}$, can be obtained directly with predict's scaledsch option; see [ST] **stcox postestimation**.

Scaled Schoenfeld residuals are centered at $\widehat{\beta}$ for each covariate and, when there is no violation of proportional hazards, should have slope zero when plotted against functions of time. The estat phtest command uses these residuals, tests the null hypothesis that the slope is equal to zero for each covariate in the model, and performs the global test proposed by Grambsch and Therneau (1994). The test of zero slope is equivalent to testing that the log hazard-ratio function is constant over time. With the detail option, estat phtest also reports the correlation between the scaled Schoenfeld residuals and the specified function of time.

For a specified function of time, $g(t)$, the statistic for testing the pth individual covariate is, for $\overline{g}(t) = d^{-1} \sum_{i=1}^N \delta_i g(t_i)$,

$$\chi^2_c = \frac{\left[\sum_{i=1}^N \left\{\delta_i g(t_i) - \overline{g}(t)\right\} r^*_{Spi}\right]^2}{d \operatorname{Var}(\widehat{\beta}_p) \sum_{i=1}^N \left\{\delta_i g(t_i) - \overline{g}(t)\right\}^2}$$

which is asymptotically distributed as χ^2 with 1 degree of freedom. r^*_{Spi} is the scaled Schoenfeld residual for observation i, and δ_i indicates failure for observation i, with $d = \sum \delta_i$.

The statistic for the global test is calculated as

$$\chi^2_g = \left[\sum_{i=1}^N \left\{\delta_i g(t_i) - \overline{g}(t)\right\} \mathbf{r}_{S_i}\right]' \left[\frac{d \operatorname{Var}(\widehat{\boldsymbol{\beta}})}{\sum_{i=1}^N \left\{\delta_i g(t_i) - \overline{g}(t)\right\}^2}\right] \left[\sum_{i=1}^N \left\{\delta_i g(t_i) - \overline{g}(t)\right\} \mathbf{r}_{S_i}\right]$$

for $\mathbf{r}_{S_i}$, a vector of the m (unscaled) Schoenfeld residuals for the ith observation; see [ST] **stcox postestimation**. The global test statistic is asymptotically distributed as χ^2 with m degrees of freedom.

The equations for the scaled Schoenfeld residuals and the two test statistics just described assume homogeneity of variance across risk sets. Although these tests are fairly robust to deviations from this assumption, care must be exercised, particularly when dealing with a stratified Cox model.

Acknowledgment

References

Allison, P. D. 2014. *Event History and Survival Analysis*. 2nd ed. Newbury Park, CA: Sage.

Barthel, F. M.-S., and P. Royston. 2006. Graphical representation of interactions. *Stata Journal* 6: 348–363.

Breslow, N. E. 1974. Covariance analysis of censored survival data. *Biometrics* 30: 89–99. https://doi.org/10.2307/2529620.

Cox, D. R. 1972. Regression models and life-tables (with discussion). *Journal of the Royal Statistical Society, Series B* 34: 187–220. https://doi.org/10.1111/j.2517-6161.1972.tb00899.x.

———. 1975. Partial likelihood. *Biometrika* 62: 269–276. https://doi.org/10.1093/biomet/62.2.269.

Cox, D. R., and D. Oakes. 1984. *Analysis of Survival Data*. London: Chapman and Hall/CRC.

Cox, D. R., and E. J. Snell. 1968. A general definition of residuals (with discussion). *Journal of the Royal Statistical Society, Series B* 30: 248–275. https://doi.org/10.1111/j.2517-6161.1968.tb00724.x.

Garrett, J. M. 1997. sbe14: Odds ratios and confidence intervals for logistic regression models with effect modification. *Stata Technical Bulletin* 36: 15–22. Reprinted in *Stata Technical Bulletin Reprints*, vol. 6, pp. 104–114. College Station, TX: Stata Press.

———. 1998. ssa12: Predicted survival curves for the Cox proportional hazards model. *Stata Technical Bulletin* 44: 37–41. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 285–290. College Station, TX: Stata Press.

Grambsch, P. M., and T. M. Therneau. 1994. Proportional hazards tests and diagnostics based on weighted residuals. *Biometrika* 81: 515–526. https://doi.org/10.2307/2337123.

Hess, K. R. 1995. Graphical methods for assessing violations of the proportional hazards assumption in Cox regression. *Statistics in Medicine* 14: 1707–1723. https://doi.org/10.1002/sim.4780141510.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Xiao, T., X. He, G. A. Whitemore, and M.-L. Ting Lee. 2012. Threshold regression for time-to-event analysis: The stthreg package. *Stata Journal* 12: 257–283.

Also see

[ST] **stcox** — Cox proportional hazards model

[ST] **sts** — Generate, graph, list, and test the survivor and related functions

[ST] **stset** — Declare data to be survival-time data

[ST] ***adjustfor_option*** — Adjust survivor and related functions for covariates at specific values

[U] **20 Estimation and postestimation commands**

Title

stcox postestimation — Postestimation tools for stcox

Postestimation commands

The following postestimation commands are of special interest after stcox:

Command	Description
*estat concordance	compute the concordance probability
estat phtest	test the proportional-hazards assumption
*estat gofplot	produce goodness-of-fit plot
stcoxkm	plot Kaplan–Meier observed survival and Cox predicted curves
stcurve	plot the survivor, failure, hazard, or cumulative hazard function
stphplot	plot $-\ln\{-\ln(\text{survival})\}$ curves
lassogof	calculate goodness-of-fit predictions

*estat concordance and estat gofplot are not appropriate with svy estimation results.

The following standard postestimation commands are also available:

Command	Description
contrast	contrasts and ANOVA-style joint tests of estimates
estat ic	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC)
estat summarize	summary statistics for the estimation sample
estat vce	variance–covariance matrix of the estimators (VCE)
estat (svy)	postestimation statistics for survey data
estimates	cataloging estimation results
etable	table of estimation results
*hausman	Hausman's specification test
lincom	point estimates, standard errors, testing, and inference for linear combinations of coefficients
linktest	link test for model specification
*lrtest	likelihood-ratio test
margins	marginal means, predictive margins, marginal effects, and average marginal effects
marginsplot	graph the results from margins (profile plots, interaction plots, etc.)
nlcom	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
predict	hazard ratios, survivor functions, influence statistics, residuals, etc.
predictnl	point estimates, standard errors, testing, and inference for generalized predictions
pwcompare	pairwise comparisons of estimates
test	Wald tests of simple and composite linear hypotheses
testnl	Wald tests of nonlinear hypotheses

*hausman and lrtest are not appropriate with svy estimation results.

predict

Description for predict

predict creates a new variable containing predictions such as hazard ratios; linear predictions; standard errors; baseline survivor, cumulative hazard, and hazard functions; martingale, Cox–Snell, deviance, efficient score, Schoenfeld, and scaled Schoenfeld residuals; likelihood displacement values; LMAX measures of influence; log frailties; and DFBETA measures of influence.

Menu for predict

Statistics > Postestimation

Syntax for predict

> predict [*type*] *newvar* [*if*] [*in*] [, *sv_statistic* <u>atfrail</u>ty [(*varname* | #)]
>
>   <u>nooff</u>set <u>partial</u>]

> predict [*type*] { *stub** | *newvarlist* } [*if*] [*in*], *mv_statistic* [<u>partial</u>]

sv_statistic	Description
Main	
hr	predicted hazard ratio, also known as the relative hazard; the default
xb	linear prediction $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$
stdp	standard error of the linear prediction; $\mathrm{SE}(\mathbf{x}_j\widehat{\boldsymbol{\beta}})$
* <u>basesurv</u>	baseline survivor function
* <u>basechazard</u>	baseline cumulative hazard function
* basehc	baseline hazard contributions
* <u>mgale</u>	martingale residuals
* <u>csnell</u>	Cox–Snell residuals
* <u>deviance</u>	deviance residuals
* <u>ldisplace</u>	likelihood displacement values
* <u>lmax</u>	LMAX measures of influence
* <u>effects</u>	log frailties

mv_statistic	Description
Main	
* <u>scores</u>	efficient score residuals
* esr	synonym for scores
* <u>dfbeta</u>	DFBETA measures of influence
* <u>schoenfeld</u>	Schoenfeld residuals
* <u>scaledsch</u>	scaled Schoenfeld residuals

Unstarred statistics are available both in and out of sample; type predict ... if e(sample) ... if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when e(sample) is not specified. nooffset is allowed only with unstarred statistics.

mgale, csnell, deviance, ldisplace, lmax, dfbeta, schoenfeld, and scaledsch are not allowed with svy estimation results.

Options for predict

    ┌─── Main ───

hr, the default, calculates the relative hazard (hazard ratio), that is, the exponentiated linear prediction, $\exp(\mathbf{x}_j\widehat{\boldsymbol{\beta}})$.

xb calculates the linear prediction from the fitted model. That is, you fit the model by estimating a set of parameters, β_1, β_2, ..., β_k, and the linear prediction is $\widehat{\beta}_1 x_{1j} + \widehat{\beta}_2 x_{2j} + \cdots + \widehat{\beta}_k x_{kj}$, often written in matrix notation as $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$.

The x_{1j}, x_{2j}, ..., x_{kj} used in the calculation are obtained from the data currently in memory and need not correspond to the data on the independent variables used in estimating $\boldsymbol{\beta}$.

stdp calculates the standard error of the prediction, that is, the standard error of $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$.

basesurv calculates the baseline survivor function. In the null model, this is equivalent to the Kaplan–Meier product-limit estimate. If stcox's strata() option was specified, baseline survivor functions for each stratum are provided.

basechazard calculates the cumulative baseline hazard. If stcox's strata() option was specified, cumulative baseline hazards for each stratum are provided.

basehc calculates the baseline hazard contributions. These are used to construct the product-limit type estimator for the baseline survivor function generated by basesurv. If stcox's strata() option was specified, baseline hazard contributions for each stratum are provided.

mgale calculates the martingale residuals. For multiple-record-per-subject data, by default only one value per subject is calculated, and it is placed on the last record for the subject.

Adding the partial option will produce partial martingale residuals, one for each record within subject; see partial below. Partial martingale residuals are the additive contributions to a subject's overall martingale residual. In single-record-per-subject data, the partial martingale residuals are the martingale residuals.

csnell calculates the Cox–Snell generalized residuals. For multiple-record data, by default only one value per subject is calculated, and it is placed on the last record for the subject.

Adding the partial option will produce partial Cox–Snell residuals, one for each record within subject; see partial below. Partial Cox–Snell residuals are the additive contributions to a subject's overall Cox–Snell residual. In single-record data, the partial Cox–Snell residuals are the Cox–Snell residuals.

deviance calculates the deviance residuals. Deviance residuals are martingale residuals that have been transformed to be more symmetric about zero. For multiple-record data, by default only one value per subject is calculated, and it is placed on the last record for the subject.

Adding the partial option will produce partial deviance residuals, one for each record within subject; see partial below. Partial deviance residuals are transformed partial martingale residuals. In single-record data, the partial deviance residuals are the deviance residuals.

ldisplace calculates the *likelihood displacement values*. A likelihood displacement value is an influence measure of the effect of deleting a subject on the overall coefficient vector. For multiple-record data, by default only one value per subject is calculated, and it is placed on the last record for the subject.

Adding the partial option will produce partial likelihood displacement values, one for each record within subject; see partial below. Partial displacement values are interpreted as effects due to deletion of individual records rather than deletion of individual subjects. In single-record data, the partial likelihood displacement values are the likelihood displacement values.

lmax calculates the LMAX measures of influence. LMAX values are related to likelihood displacement values because they also measure the effect of deleting a subject on the overall coefficient vector. For multiple-record data, by default only one LMAX value per subject is calculated, and it is placed on the last record for the subject.

Adding the partial option will produce partial LMAX values, one for each record within subject; see partial below. Partial LMAX values are interpreted as effects due to deletion of individual records rather than deletion of individual subjects. In single-record data, the partial LMAX values are the LMAX values.

effects is for use after stcox, shared() and provides estimates of the log frailty for each group. The log frailties are random group-specific offsets to the linear predictor that measure the group effect on the log relative-hazard.

scores calculates the efficient score residuals for each regressor in the model. For multiple-record data, by default only one score per subject is calculated, and it is placed on the last record for the subject.

Adding the partial option will produce partial efficient score residuals, one for each record within subject; see partial below. Partial efficient score residuals are the additive contributions to a subject's overall efficient score residual. In single-record data, the partial efficient score residuals are the efficient score residuals.

One efficient score residual variable is created for each regressor in the model; the first new variable corresponds to the first regressor, the second to the second, and so on.

esr is a synonym for scores.

dfbeta calculates the DFBETA measures of influence for each regressor in the model. The DFBETA value for a subject estimates the change in the regressor's coefficient due to deletion of that subject. For multiple-record data, by default only one value per subject is calculated, and it is placed on the last record for the subject.

Adding the partial option will produce partial DFBETAs, one for each record within subject; see partial below. Partial DFBETAs are interpreted as effects due to deletion of individual records rather than deletion of individual subjects. In single-record data, the partial DFBETAs are the DFBETAs.

One DFBETA variable is created for each regressor in the model; the first new variable corresponds to the first regressor, the second to the second, and so on.

schoenfeld calculates the Schoenfeld residuals. This option may not be used after stcox with the exactm or exactp option. Schoenfeld residuals are calculated and reported only at failure times.

One Schoenfeld residual variable is created for each regressor in the model; the first new variable corresponds to the first regressor, the second to the second, and so on.

scaledsch calculates the scaled Schoenfeld residuals. This option may not be used after stcox with the exactm or exactp option. Scaled Schoenfeld residuals are calculated and reported only at failure times.

One scaled Schoenfeld residual variable is created for each regressor in the model; the first new variable corresponds to the first regressor, the second to the second, and so on.

Note: The easiest way to use the preceding four options is, for example,

> `. predict double` *stub*`*, scores`

where *stub* is a short name of your choosing. Stata then creates variables *stub*1, *stub*2, etc. You may also specify each variable name explicitly, in which case there must be as many (and no more) variables specified as there are regressors in the model.

atfrailty or atfrailty(*varname* | #) is allowed only with basesurv, basechazard, and basehc and is relevant only if you specified shared(*varname*) for stcox. It modifies the computations of baseline functions so that values for the frailties are included in the computation.

Specifying atfrailty allows you to use the estimates of the frailty for each group, which are the exponentiation of the results calculated by predict, effects. atfrailty(*varname* | #) allows you to specify your own frailty values; # must be a positive number.

nooffset is allowed only with hr, xb, and stdp, and is relevant only if you specified offset(*varname*) for stcox. It modifies the calculations made by predict so that they ignore the offset variable; the linear prediction is treated as $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$ rather than $\mathbf{x}_j\widehat{\boldsymbol{\beta}} + \text{offset}_j$.

partial is relevant only for multiple-record data and is valid with mgale, csnell, deviance, ldisplace, lmax, scores, esr, and dfbeta. Specifying partial will produce "partial" versions of these statistics, where one value is calculated for each record instead of one for each subject. The subjects are determined by the [id()](id()) option to stset.

Specify partial if you wish to perform diagnostics on individual records rather than on individual subjects. For example, a partial DFBETA would be interpreted as the effect on a coefficient due to deletion of one record, rather than the effect due to deletion of all records for a given subject.

margins

Description for margins

margins estimates margins of response for hazard ratios and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

> margins [*marginlist*] [, *options*]
>
> margins [*marginlist*] , <u>predict</u>(*statistic ...*) [<u>predict</u>(*statistic ...*) ...] [*options*]

statistic	Description
hr	predicted hazard ratio, also known as the relative hazard; the default
xb	linear prediction $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$
<u>stdp</u>	not allowed with margins
<u>basesurv</u>	not allowed with margins
<u>basechazard</u>	not allowed with margins
basehc	not allowed with margins
<u>mgale</u>	not allowed with margins
<u>csnell</u>	not allowed with margins
deviance	not allowed with margins
<u>ldisplace</u>	not allowed with margins
<u>lmax</u>	not allowed with margins
<u>eff</u>ects	not allowed with margins
<u>sc</u>ores	not allowed with margins
esr	not allowed with margins
dfbeta	not allowed with margins
<u>sch</u>oenfeld	not allowed with margins
<u>sca</u>ledsch	not allowed with margins

Statistics not allowed with margins are functions of stochastic quantities other than e(b).

For the full syntax, see [R] **margins**.

estat

Description for estat

estat concordance calculates the concordance probability, which is defined as the probability that predictions and outcomes are concordant. estat concordance provides two measures of the concordance probability: Harrell's C and Gönen and Heller's K concordance coefficients. estat concordance also reports the Somers's D rank correlation, which is obtained by calculating $2C - 1$ or $2K - 1$.

Menu for estat

Statistics > Postestimation

Syntax for estat

estat <u>con</u>cordance $\begin{bmatrix} if \end{bmatrix}$ $\begin{bmatrix} in \end{bmatrix}$ $\begin{bmatrix} , concordance\_options \end{bmatrix}$

concordance_options	Description
Main	
<u>h</u>arrell	compute Harrell's C coefficient; the default
gheller	compute Gönen and Heller's concordance coefficient
se	compute asymptotic standard error of Gönen and Heller's coefficient
all	compute statistic for all observations in the data
<u>nosh</u>ow	do not show st setting information

collect is allowed; see [U] **11.1.10 Prefix commands**.

Options for estat

┌─ Main ───

harrell, the default, calculates Harrell's C coefficient, which is defined as the proportion of all usable subject pairs in which the predictions and outcomes are concordant.

gheller calculates Gönen and Heller's K concordance coefficient instead of Harrell's C coefficient. The harrell and gheller options may be specified together to obtain both concordance measures.

se calculates the smoothed version of Gönen and Heller's K concordance coefficient and its asymptotic standard error. The se option requires the gheller option.

all requests that the statistic be computed for all observations in the data. By default, estat concordance computes over the estimation subsample.

noshow prevents estat concordance from displaying the identities of the key st variables above its output.

Remarks and examples

Remarks are presented under the following headings:

Baseline functions

predict after stcox provides estimates of the baseline survivor and baseline cumulative hazard function, among other things. Here the term *baseline* means that these are the functions when all covariates are set to zero, that is, they reflect (perhaps hypothetical) individuals who have zero-valued measurements. When you specify predict's basechazard option, you obtain the baseline cumulative hazard. When you specify basesurv, you obtain the baseline survivor function. Additionally, when you specify predict's basehc option, you obtain estimates of the baseline hazard contribution at each failure time, which are factors used to develop the product-limit estimator for the survivor function generated by basesurv.

Although in theory $S_0(t) = \exp\{-H_0(t)\}$, where $S_0(t)$ is the baseline survivor function and $H_0(t)$ is the baseline cumulative hazard, the estimates produced by basechazard and basesurv do not exactly correspond in this manner, although they closely do. The reason is that predict after stcox uses different estimation schemes for each; the exact formulas are given in *Methods and formulas*.

When the Cox model is fit with the strata() option, you obtain estimates of the baseline functions for each stratum.

▷ Example 1: Baseline survivor function

Baseline functions refer to the values of the functions when all covariates are set to 0. Let's graph the survival curve for the Stanford heart transplant model that we fit in example 3 of [ST] **stcox**, and to make the baseline curve reasonable, let's do that at age $= 40$ and year $= 70$.

Thus we will begin by creating variables that, when 0, correspond to the baseline values we desire, and then we will fit our model with these variables instead. We then predict the baseline survivor function and graph it:

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)
. generate age40 = age - 40
. generate year70 = year - 70
```

```
. stcox age40 posttran surg year70, nolog

        Failure _d: died
  Analysis time _t: t1
        ID variable: id

Cox regression with Breslow method for ties

No. of subjects =        103                          Number of obs =        172
No. of failures =         75
Time at risk    = 31,938.1
                                                     LR chi2(4)      =   17.56
Log likelihood = -289.53378                          Prob > chi2     =  0.0015
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age40	1.030224	.0143201	2.14	0.032	1.002536	1.058677
posttran	.9787243	.3032597	-0.07	0.945	.5332291	1.796416
surgery	.3738278	.163204	-2.25	0.024	.1588759	.8796
year70	.8873107	.059808	-1.77	0.076	.7775022	1.012628

```
. predict s, basesurv

. summarize s
```

Variable	Obs	Mean	Std. dev.	Min	Max
s	172	.6291871	.2530009	.130666	.9908968

Our recentering of age and year did not affect the estimation, a fact you can verify by refitting the model with the original age and year variables.

To see how the values of the baseline survivor function are stored, we first sort according to analysis time and then list some observations.

```
. sort _t id

. list id _t0 _t _d s in 1/20
```

	id	_t0	_t	_d	s
1.	3	0	1	0	.9908968
2.	15	0	1	1	.9908968
3.	20	0	1	0	.9908968
4.	45	0	1	0	.9908968
5.	39	0	2	0	.9633915
6.	43	0	2	1	.9633915
7.	46	0	2	0	.9633915
8.	61	0	2	1	.9633915
9.	75	0	2	1	.9633915
10.	95	0	2	0	.9633915
11.	6	0	3	1	.9356873
12.	23	0	3	0	.9356873
13.	42	0	3	1	.9356873
14.	54	0	3	1	.9356873
15.	60	0	3	0	.9356873
16.	68	0	3	0	.9356873
17.	72	0	4	0	.9356873
18.	94	0	4	0	.9356873
19.	38	0	5	0	.9264087
20.	70	0	5	0	.9264087

At time $\_t = 2$, the baseline survivor function is 0.9634, or more precisely, $S_0(2 + \Delta t) = 0.9634$. What we mean by $S_0(t + \Delta t)$ is the probability of surviving just beyond t. This is done to clarify that the probability includes escaping failure at precisely time t.

The above also indicates that our estimate of $S_0(t)$ is a step function, and that the steps occur only at times when failure is observed—our estimated $S_0(t)$ does not change from $\_t = 3$ to $\_t = 4$ because no failure occurred at time 4. This behavior is analogous to that of the Kaplan–Meier estimate of the survivor function; see [ST] **sts**.

Here is a graph of the baseline survival curve:

```
. line s _t, sort c(J)
```
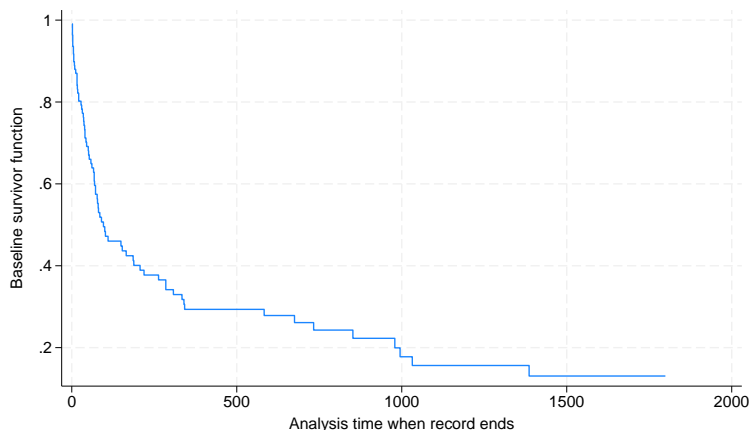

This graph was easy enough to produce because we wanted the survivor function at baseline. To graph survivor functions after `stcox` with covariates set to any value (baseline or otherwise), use `stcurve`; see [ST] **stcurve**.   ◁

The similarity to Kaplan–Meier is not limited to the fact that both are step functions that change only when failure occurs. They are also calculated in much the same way, with predicting `basesurv` after `stcox` having the added benefit that the result is automatically adjusted for all the covariates in your Cox model. When you have no covariates, both methods are equivalent. If you continue from the previous example, you will find that

```
. sts generate s1 = s
```

and

```
. stcox, estimate
. predict double s2, basesurv
```

produce the identical variables `s1` and `s2`, both containing estimates of the overall survivor function, unadjusted for covariates. We used type `double` for `s2` to precisely match `sts generate`, which gives results in double precision.

If we had fit a stratified model by using the `strata()` option, the recorded survivor-function estimate on each observation would be for the stratum of that observation. That is, what you get is one variable that holds not an overall survivor curve, but instead a set of stratum-specific curves.

▷ Example 2: Baseline cumulative hazard

Obtaining estimates of the baseline cumulative hazard, $H_0(t)$, is just as easy as obtaining the baseline survivor function. Using the same data as previously,

```
. use https://www.stata-press.com/data/r18/stan3, clear
(Heart transplant data)
. generate age40 = age - 40
. generate year70 = year - 70
. stcox age40 posttran surg year70
(output omitted)
. predict ch, basechazard
. line ch _t, sort c(J)
```
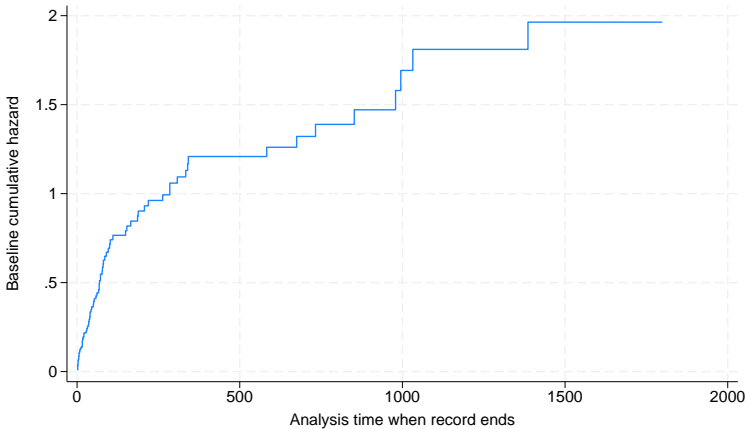

The estimated baseline cumulative hazard is also a step function with the steps occurring at the observed times of failure. When there are no covariates in your Cox model, what you obtain is equivalent to the Nelson–Aalen estimate of the cumulative hazard (see [ST] sts), but using predict, basechazard after stcox allows you to also adjust for covariates.

To obtain cumulative hazard curves at values other than baseline, you could either recenter your covariates—as we did previously with age and year—so that the values in which you are interested become baseline, or simply use stcurve; see [ST] stcurve.

◁

▷ Example 3: Baseline hazard contributions

Mathematically, a baseline hazard contribution, $h_i = (1 - \alpha_i)$ (see Kalbfleisch and Prentice 2002, 115), is defined at every analytic time t_i at which a failure occurs and is undefined at other times. Stata stores h_i in observations where a failure occurred and stores missing values in the other observations.

```
. use https://www.stata-press.com/data/r18/stan3, clear
(Heart transplant data)
. generate age40 = age - 40
. generate year70 = year - 70
. stcox age40 posttran surg year70
(output omitted)
. predict double h, basehc
(97 missing values generated)
```

```
. list id _t0 _t _d h in 1/10
```

	id	_t0	_t	_d	h
1.	1	0	50	1	.01503465
2.	2	0	6	1	.02035303
3.	3	0	1	0	.
4.	3	1	16	1	.03339642
5.	4	0	36	0	.
6.	4	36	39	1	.01365406
7.	5	0	18	1	.01167142
8.	6	0	3	1	.02875689
9.	7	0	51	0	.
10.	7	51	675	1	.06215003

At time $\_t = 50$, the hazard contribution h_1 is 0.0150. At time $\_t = 6$, the hazard contribution h_2 is 0.0204. In observation 3, no hazard contribution is stored. Observation 3 contains a missing value because observation 3 did not fail at time 1. We also see that values of the hazard contributions are stored only in observations that are marked as failing.

Hazard contributions by themselves have no substantive interpretation, and in particular they should *not* be interpreted as estimating the hazard function at time t. Hazard contributions are simply mass points that are used as components to calculate the survivor function; see *Methods and formulas*. You can also use hazard contributions to estimate the hazard, but because they are only mass points, they need to be smoothed first. This smoothing is done automatically with stcurve; see [ST] **stcurve**. In summary, hazard contributions in their raw form serve no purpose other than to help replicate calculations done by Stata, and we demonstrate this below simply for illustrative purposes.

When we created the new variable h for holding the hazard contributions, we used type double because we plan on using h in some further calculations below and we wish to be as precise as possible.

In contrast with the baseline hazard contributions, the baseline survivor function, $S_0(t)$, is defined at all values of t: its estimate changes its value when failures occur, and at times when no failures occur, the estimated $S_0(t)$ is equal to its value at the time of the last failure.

Continuing with our example, we now predict the baseline survivor function:

```
. predict double s, basesurv
. list id _t0 _t _d h s in 1/10
```

	id	_t0	_t	_d	h	s
1.	1	0	50	1	.01503465	.68100303
2.	2	0	6	1	.02035303	.89846438
3.	3	0	1	0	.	.99089681
4.	3	1	16	1	.03339642	.84087361
5.	4	0	36	0	.	.7527663
6.	4	36	39	1	.01365406	.73259264
7.	5	0	18	1	.01167142	.82144038
8.	6	0	3	1	.02875689	.93568733
9.	7	0	51	0	.	.6705895
10.	7	51	675	1	.06215003	.26115633

In the above, we sorted by `id`, but it is easier to see how `h` and `s` are related if we sort by `_t` and put the failures on top:

```
. gsort +_t -_d
. list id _t0 _t _d h s in 1/18
```

	id	_t0	_t	_d	h	s
1.	15	0	1	1	.00910319	.99089681
2.	3	0	1	0	.	.99089681
3.	20	0	1	0	.	.99089681
4.	45	0	1	0	.	.99089681
5.	43	0	2	1	.02775802	.96339147
6.	75	0	2	1	.02775802	.96339147
7.	61	0	2	1	.02775802	.96339147
8.	95	0	2	0	.	.96339147
9.	39	0	2	0	.	.96339147
10.	46	0	2	0	.	.96339147
11.	54	0	3	1	.02875689	.93568733
12.	6	0	3	1	.02875689	.93568733
13.	42	0	3	1	.02875689	.93568733
14.	60	0	3	0	.	.93568733
15.	23	0	3	0	.	.93568733
16.	68	0	3	0	.	.93568733
17.	94	0	4	0	.	.93568733
18.	72	0	4	0	.	.93568733

The baseline hazard contribution is stored on every failure record—if multiple failures occur at a given time, the value of the hazard contribution is repeated—and the baseline survivor is stored on every record. (More correctly, baseline values are stored on records that meet the criterion and that were used in estimation. If some observations are explicitly or implicitly excluded from the estimation, their baseline values will be set to missing, no matter what.)

With this listing, we can better understand how the hazard contributions are used to calculate the survivor function. Because the patient with $id = 15$ died at time $t_1 = 1$, the hazard contribution for that patient is $h_{15} = 0.00910319$. Because that was the only death at $t_1 = 1$, the estimated survivor function at this time is $S_0(1) = 1 - h_{15} = 1 - 0.00910319 = 0.99089681$. The next death occurs at time $t_1 = 2$, and the hazard contribution at this time for patient 43 (or patient 61 or patient 75, it does not matter) is $h_{43} = 0.02775802$. Multiplying the previous survivor function value by $1 - h_{43}$ gives the new survivor function at $t_1 = 2$ as $S_0(2) = 0.96339147$. The other survivor function values are then calculated in succession, using this method at each failure time. At times when no failures occur, the survivor function remains unchanged.

◁

❑ Technical note

Consider manually obtaining the estimate of $S_0(t)$ from the h_i:

```
. sort _t _d
. by _t: keep if _d & _n==_N
. generate double s2 = 1-h
. replace s2 = s2[_n-1]*s2 if _n>1
```

s2 will be equivalent to s as produced above. If you had obtained stratified estimates, the code would be

```
. sort group _t _d
. by group _t: keep if _d & _n==_N
. generate double s2 = 1-h
. by group: replace s2 = s2[_n-1]*s2 if _n>1
```

❑

Making baseline reasonable

When predicting with basesurv or basechazard, for numerical accuracy reasons, the baseline functions must correspond to something reasonable in your data. Remember, the baseline functions correspond to all covariates equal to 0 in your Cox model.

Consider, for instance, a Cox model that includes the variable calendar year among the covariates. Say that year varies between 1980 and 1996. The baseline functions would correspond to year 0, almost 2,000 years in the past. Say that the estimated coefficient on year is -0.2, meaning that the hazard ratio for one year to the next is a reasonable 0.82.

Think carefully about the contribution to the predicted log cumulative hazard: it would be approximately $-0.2 \times 2,000 = -400$. Now $e^{-400} \approx 10^{-173}$, which on a digital computer is so close to 0 that there is simply no hope that $H_0(t)e^{-400}$ will produce an accurate estimate of $H(t)$.

Even with less extreme numbers, problems arise, even in the calculation of the baseline survivor function. Baseline hazard contributions near 1 produce baseline survivor functions with steps differing by many orders of magnitude because the calculation of the survivor function is cumulative. Producing a meaningful graph of such a survivor function is hopeless, and adjusting the survivor function to other values of the covariates is too much work.

For these reasons, covariate values of 0 must be meaningful if you are going to specify the basechazard or basesurv option. As the baseline values move to absurdity, the first problem you will encounter is a baseline survivor function that is too hard to interpret, even though the baseline hazard contributions are estimated accurately. Further out, the procedure Stata uses to estimate the baseline hazard contributions will break down—it will produce results that are exactly 1. Hazard contributions that are exactly 1 produce survivor functions that are uniformly 0, and they will remain 0 even after adjusting for covariates.

This, in fact, occurs with the Stanford heart transplant data:

```
. use https://www.stata-press.com/data/r18/stan3, clear
(Heart transplant data)
. stcox age posttran surg year
  (output omitted)
. predict ch, basechazard
. predict s, basesurv
. summarize ch s
```

Variable	Obs	Mean	Std. dev.	Min	Max
ch	172	745.1134	682.8671	11.88239	2573.637
s	172	1.45e-07	9.43e-07	0	6.24e-06

The hint that there are problems is that the values of ch are huge and the values of s are close to 0. In this dataset, age (which ranges from 8 to 64 with a mean value of 45) and year (which ranges from 67 to 74) are the problems. The baseline functions correspond to a newborn at the turn of the century on the waiting list for a heart transplant!

To obtain accurate estimates of the baseline functions, type

```
. drop ch s
. generate age40 = age - 40
. generate year70 = year - 70
. stcox age40 posttran surg year70
  (output omitted)
. predict ch, basechazard
. predict s, basesurv
. summarize ch s
```

Variable	Obs	Mean	Std. dev.	Min	Max
ch	172	.5685743	.521076	.0090671	1.963868
s	172	.6291871	.2530009	.130666	.9908968

Adjusting the variables does not affect the coefficient (and, hence, hazard-ratio) estimates, but it changes the values at which the baseline functions are estimated to be within the range of the data.

❑ Technical note

Above we demonstrated what can happen to predicted baseline functions when baseline values represent a departure from what was observed in the data. In the above example, the Cox model fit was fine and only the baseline functions lacked accuracy. As baseline values move even further toward absurdity, the risk-set accumulations required to fit the Cox model will also break down. If you are having difficulty getting stcox to converge or you obtain missing coefficients, one possible solution is to recenter your covariates just as we did above.

❑

Residuals and diagnostic measures

Stata can calculate Cox–Snell residuals, martingale residuals, deviance residuals, efficient score residuals (esr), Schoenfeld residuals, scaled Schoenfeld residuals, likelihood displacement values, LMAX values, and DFBETA influence measures.

Although the uses of residuals vary and depend on the data and user preferences, traditional and suggested uses are the following: Cox–Snell residuals are useful in assessing overall model fit. Martingale residuals are useful in determining the functional form of covariates to be included in the model and are occasionally useful in identifying outliers. Deviance residuals are useful in examining model accuracy and identifying outliers. Schoenfeld and scaled Schoenfeld residuals are useful for checking and testing the proportional-hazards assumption. Likelihood displacement values and LMAX values are useful in identifying influential subjects. DFBETAs also measure influence, but they do so on a coefficient-by-coefficient basis. Likelihood displacement values, LMAX values, and DFBETAs are all based on efficient score residuals.

▷ Example 4: Cox–Snell residuals

Let's first examine the use of Cox–Snell residuals. Using the cancer data introduced in example 2 in [ST] stcox, we first perform a Cox regression and then predict the Cox–Snell residuals.

```
. use https://www.stata-press.com/data/r18/drugtr, clear
(Patient survival in drug trial)
. stset studytime, failure(died)
  (output omitted)
```

```
. stcox age drug, nolog

        Failure _d: died
  Analysis time _t: studytime
```

```
Cox regression with Breslow method for ties
No. of subjects =  48                                    Number of obs =       48
No. of failures =  31
Time at risk    = 744
                                                         LR chi2(2)     =   33.18
Log likelihood = -83.323546                              Prob > chi2    = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.120325	.0417711	3.05	0.002	1.041375	1.20526
drug	.1048772	.0477017	-4.96	0.000	.0430057	.2557622

```
. predict cs, csnell
```

The csnell option tells predict to output the Cox–Snell residuals to a new variable, cs. If the Cox regression model fits the data, these residuals should have a standard censored exponential distribution with hazard ratio 1. We can verify the model's fit by calculating—based, for example, on the Kaplan–Meier estimated survivor function or the Nelson–Aalen estimator—an empirical estimate of the cumulative hazard function, using the Cox–Snell residuals as the time variable and the data's original censoring variable. If the model fits the data, the plot of the cumulative hazard versus cs should approximate a straight line with slope 1.

To do this, we first re-stset the data, specifying cs as our new failure-time variable and died as the failure/censoring indicator. We then use the sts generate command to generate the H variable containing the Nelson–Aalen cumulative hazard estimates and plot it against cs.

```
. stset cs, failure(died)
  (output omitted )
. sts generate H = na
. line H cs cs, sort ytitle("") clstyle(. refline)
```
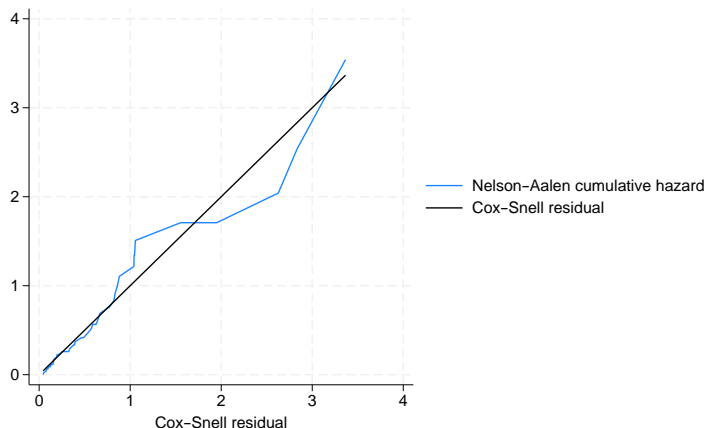
We specified `cs` twice in the graph command above so that a reference $45°$ line is plotted. Comparing the jagged line with the reference line, we observe that the Cox model does not fit these data too badly.

We could have used `estat gofplot` to automate the above steps, creating the goodness-of-fit plot for a Cox model with the following:

```
. quietly stset studytime, failure(died)
. quietly stcox age drug
. estat gofplot
```

◁

❑ Technical note

The statement that "if the Cox regression model fits the data, the Cox–Snell residuals have a standard censored exponential distribution with hazard ratio 1" holds only if the true parameters, β, and the true cumulative baseline hazard function, $H_0(t)$, are used in calculating the residuals. Because we use estimates $\widehat{\beta}$ and $\widehat{H}_0(t)$, deviations from the $45°$ line in the above plots could be due in part to uncertainty about these estimates. This is particularly important for small sample sizes and in the right-hand tail of the distribution, where the baseline hazard is more variable because of the reduced effective sample caused by prior failures and censoring.

❑

▷ Example 5: Martingale residuals

Let's now examine the martingale residuals. Martingale residuals are useful in assessing the functional form of a covariate to be entered into a Cox model. Sometimes the covariate may need transforming so that the transformed variable will satisfy the assumptions of the proportional hazards model. To find the appropriate functional form of a variable, we fit a Cox model excluding the variable and then plot a `lowess` smooth of the martingale residuals against some transformation of the variable in question. If the transformation is appropriate, then the smooth should be approximately linear.

We apply this procedure to our cancer data to find an appropriate transformation of `age` (or to verify that `age` need not be transformed).

```
. use https://www.stata-press.com/data/r18/drugtr, clear
(Patient survival in drug trial)
. stset studytime, failure(died)
  (output omitted)
. stcox drug
  (output omitted)
. predict mg, mgale
. lowess mg age, mean noweight title("") note("") m(o)
```
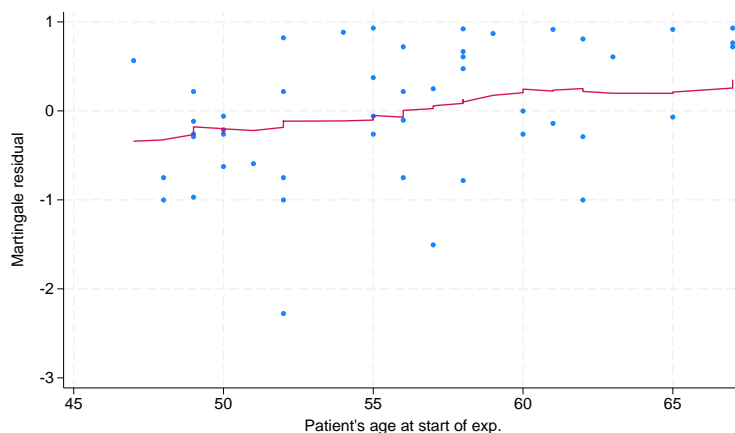

We used the `lowess` command with the `mean` and `noweight` options to obtain a plot of the running-mean smoother to ease interpretation. A lowess smoother or other smoother could also be used; see [R] **lowess**. The smooth appears nearly linear, supporting the inclusion of the untransformed version of `age` in our Cox model. Had the smooth not been linear, we would have tried smoothing the martingale residuals against various transformations of `age` until we found one that produced a near-linear smooth.

◁

Martingale residuals can also be interpreted as the difference over time of the observed number of failures minus the difference predicted by the model. Thus a plot of the martingale residuals versus the linear predictor may be used to detect outliers.

Plots of martingale residuals are sometimes difficult to interpret, however, because these residuals are skewed, taking values in $(-\infty, 1)$. For this reason, deviance residuals are preferred for examining model accuracy and identifying outliers.

Originally, "à la martingale" was a French expression meaning in the fashion of Martigues, a town in Provence. People from that town evidently had a reputation, no doubt unjustified, for their extravagance. Later the term was applied to a betting method in which a gambler doubles the stakes after each loss, which is not a strategy that StataCorp will endorse on your behalf. The current meaning in probability theory is more prosaic. In a fair game, knowing past events cannot help predict winnings in the future. By extension, a martingale is a stochastic process in time for which the expectation of the next value equals the present value, even given knowledge of all previous values. The original reference to fashion survives in equestrian and nautical terms referring to straps or stays.

▷ Example 6: Deviance residuals

Deviance residuals are a rescaling of the martingale residuals so that they are symmetric about 0 and thus are more like residuals obtained from linear regression. Plots of these residuals against the linear predictor, survival time, rank order of survival, or observation number can be useful in identifying aberrant observations and assessing model fit. We continue from the previous example, but we need to first refit the Cox model with `age` included:

```
. drop mg
. stcox drug age
  (output omitted )
. predict mg, mgale
. predict xb, xb
. scatter mg xb
```
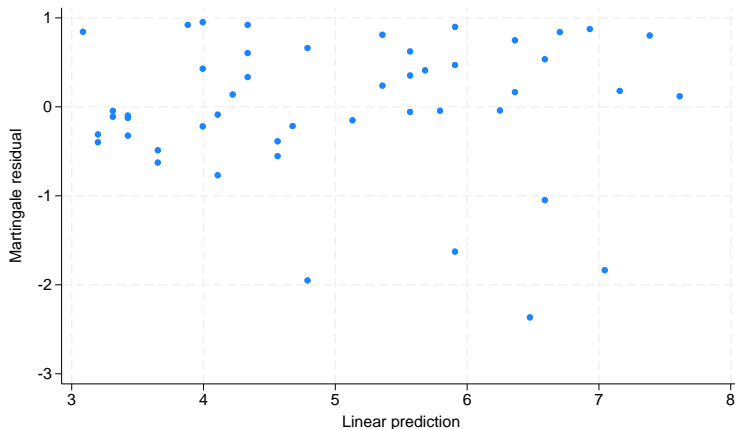
```
. predict dev, deviance
. scatter dev xb
```
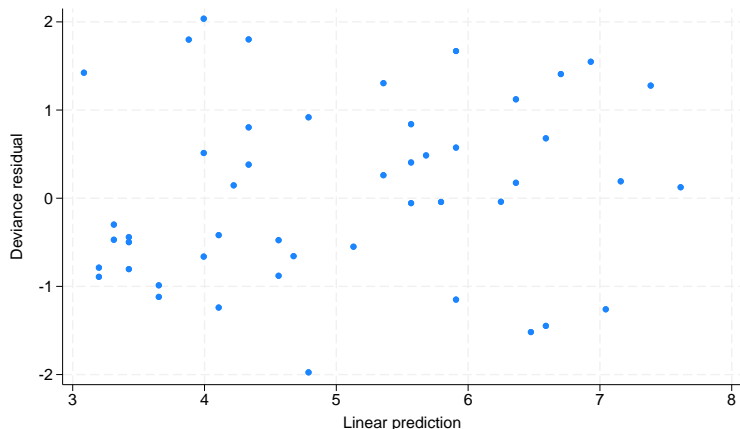

We first plotted the martingale residuals versus the linear predictor and then plotted the deviance residuals versus the linear predictor. Given their symmetry about 0, deviance residuals are easier to interpret, although both graphs yield the same information. With uncensored data, deviance residuals should resemble white noise if the fit is adequate. Censored observations would be represented as clumps of deviance residuals near 0 (Klein and Moeschberger 2003, 381). Given what we see above, there do not appear to be any outliers.

◁

In evaluating the adequacy of the fitted model, we must determine if any one subject has a disproportionate influence on the estimated parameters. This is known as influence or leverage analysis. The preferred method of performing influence or leverage analysis is to compare the estimated parameter, $\widehat{\beta}$, obtained from the full data, with estimated parameters $\widehat{\beta}_i$, obtained by fitting the model to the $N - 1$ subjects remaining after the ith subject is removed. If $\widehat{\beta} - \widehat{\beta}_i$ is close to 0, the ith subject has little influence on the estimate. The process is repeated for all subjects included in the original model. To compute these differences for a dataset with N subjects, we would have to execute `stcox` N additional times, which could be impractical for large datasets.

To avoid fitting N additional Cox models, an approximation to $\widehat{\beta} - \widehat{\beta}_i$ can be made based on the efficient score residuals; see *Methods and formulas*. The difference $\widehat{\beta} - \widehat{\beta}_i$ is commonly referred to as DFBETA in the literature; see [R] **regress postestimation**.

▷ Example 7: DFBETAs

You obtain DFBETAs by using `predict`'s `dfbeta` option:

```
. use https://www.stata-press.com/data/r18/drugtr, clear
(Patient survival in drug trial)
. stset studytime, failure(died)
(output omitted)
. stcox age drug
(output omitted)
. predict df*, dfbeta
```

The last command stores the estimates of DFBETA$_i = \widehat{\beta} - \widehat{\beta}_i$ for $i = 1, \ldots, N$ in the variables `df1` and `df2`. We can now plot these versus either time or subject (observation) number to identify

subjects with disproportionate influence. To maximize the available information, we plot versus time and label the points by their subject numbers.

```
. generate obs = _n
. scatter df1 studytime, yline(0) mlabel(obs)
```
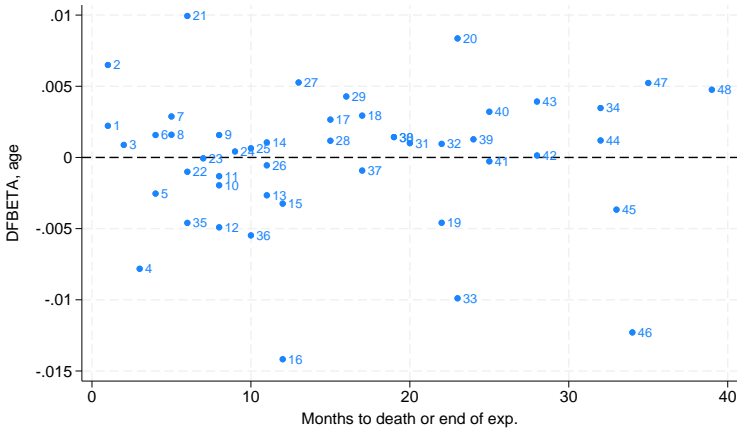


```
. scatter df2 studytime, yline(0) mlabel(obs)
```
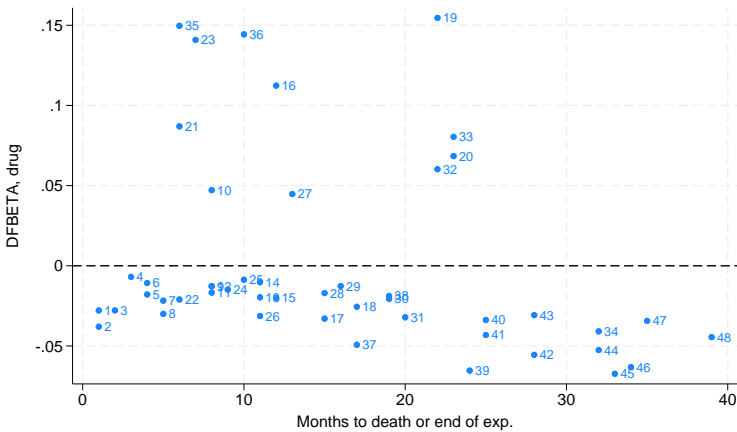

From the second graph we see that observation 35, if removed, would decrease the coefficient on `drug` by approximately 0.15 or, equivalently, decrease the hazard ratio for `drug` by a factor of approximately $\exp(-0.15) = 0.861$.

◁

DFBETAs as measures of influence have a straightforward interpretation. Their only disadvantage is that the number of values to examine grows both with sample size and with the number of regressors.

Two alternative measures of influence are *likelihood displacement* values and LMAX values, and both measure each subject's influence on the coefficient vector as a whole. Thus, for each, you have only one value per subject regardless of the number of regressors. As was the case with DFBETAs, likelihood displacement and LMAX calculations are also based on efficient score residuals; see *Methods and formulas*.

Likelihood displacement values measure influence by approximating what happens to the model log likelihood (more precisely, twice the log likelihood) when you omit subject i. Formally, the likelihood displacement value for subject i approximates the quantity

$$2 \left\{ \log L \left(\widehat{\boldsymbol{\beta}} \right) - \log L \left(\widehat{\boldsymbol{\beta}}_i \right) \right\}$$

where $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{\beta}}_i$ are defined as previously and $L(\cdot)$ is the partial likelihood for the Cox model estimated from all the data. In other words, when you calculate $L(\cdot)$, you use all the data, but you evaluate at the parameter estimates $\widehat{\boldsymbol{\beta}}_i$ obtained by omitting the ith subject. Note that because $\widehat{\boldsymbol{\beta}}$ represents an optimal solution, likelihood displacement values will always be nonnegative.

That likelihood displacements measure influence can be seen through the following logic: if subject i is influential, then the vector $\widehat{\boldsymbol{\beta}}_i$ will differ substantially from $\widehat{\boldsymbol{\beta}}$. When that occurs, evaluating the log likelihood at such a suboptimal solution will give you a very different log likelihood.

LMAX values are closely related to likelihood displacements and are derived from an eigensystem analysis of the matrix of efficient score residuals; see *Methods and formulas* for details.

Both likelihood displacement and LMAX values measure each subject's overall influence, but they are not directly comparable with each other. Likelihood displacement values should be compared only with other likelihood displacement values, and LMAX values only with other LMAX values.

▷ Example 8: Likelihood displacement and LMAX values

You obtain likelihood displacement values with predict's ldisplace option, and you obtain LMAX values with the lmax option. Continuing from the previous example:

```
. predict ld, ldisplace
. predict lmax, lmax
. list _t0 _t _d ld lmax in 1/10
```

	_t0	_t	_d	ld	lmax
1.	0	1	1	.0059511	.0735375
2.	0	1	1	.032366	.1124505
3.	0	2	1	.0038388	.0686295
4.	0	3	1	.0481942	.0113989
5.	0	4	1	.0078195	.0331513
6.	0	4	1	.0019887	.0308102
7.	0	5	1	.0069245	.0614247
8.	0	5	1	.0051647	.0763283
9.	0	8	1	.0021315	.0353402
10.	0	8	0	.0116187	.1179539

We can plot the likelihood displacement values versus time and label the points by observation number:

```
. scatter ld studytime, mlabel(obs)
```
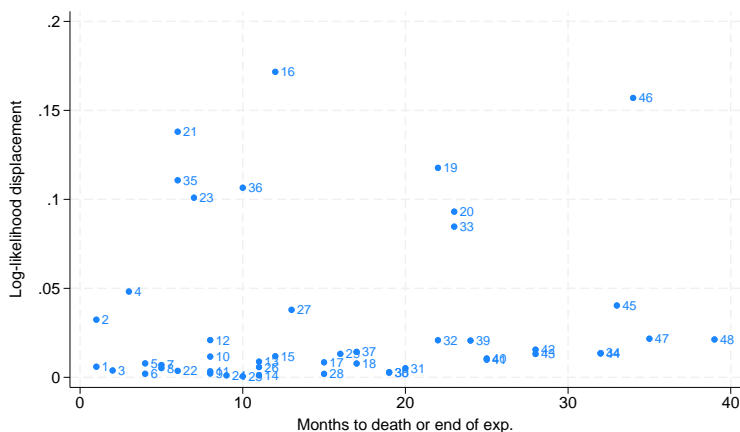

The above shows subjects 16 and 46 to be somewhat influential. A plot of LMAX values will show subject 16 as influential but not subject 46, a fact we leave to you to verify.

◁

Schoenfeld residuals and scaled Schoenfeld residuals are most often used to test the proportional-hazards assumption, as described in [ST] **stcox PH-assumption tests**.

Multiple records per subject

In the previous section, we analyzed data from a cancer study, and in doing so we were very loose in differentiating "observations" versus "subjects". In fact, we used both terms interchangeably. We were able to get away with that because in that dataset each subject (patient) was represented by only one observation—the subjects were the observations.

Oftentimes, however, subjects need representation by multiple observations, or records. For example, if a patient leaves the study for some time only to return later, at least one additional record will be needed to denote the subject's return to the study and the gap in their history. If the covariates of interest for a subject change during the study (for example, transitioning from smoking to nonsmoking), then this will also require representation by multiple records.

Multiple records per subject are not a problem for Stata; you simply specify an `id()` variable when `stsetting` your data, and this `id()` variable tells Stata which records belong to which subjects. The other commands in Stata's `st` suite know how to then incorporate this information into your analysis.

For `predict` after `stcox`, by default Stata handles diagnostic measures as always being at the *subject level*, regardless of whether that subject comprises one observation or multiple ones.

▷ Example 9: Stanford heart transplant data

As an example, consider, as we did previously, data from the Stanford heart transplant study:

```
. use https://www.stata-press.com/data/r18/stan3, clear
(Heart transplant data)
. stset
-> stset t1, id(id) failure(died)

Survival-time data settings

            ID variable: id
         Failure event: died!=0 & died<.
Observed time interval: (t1[_n-1], t1]
     Exit on or before: failure

_____

          172  total observations
            0  exclusions

_____

          172  observations remaining, representing
          103  subjects
           75  failures in single-failure-per-subject data
     31,938.1  total analysis time at risk and under observation
                                               At risk from t =          0
                                     Earliest observed entry t =          0
                                        Last observed exit t =      1,799
. list id _t0 _t _d age posttran surgery year in 1/10
```

	id	_t0	_t	_d	age	posttran	surgery	year
1.	1	0	50	1	30	0	0	67
2.	2	0	6	1	51	0	0	68
3.	3	0	1	0	54	0	0	68
4.	3	1	16	1	54	1	0	68
5.	4	0	36	0	40	0	0	68
6.	4	36	39	1	40	1	0	68
7.	5	0	18	1	20	0	0	68
8.	6	0	3	1	54	0	0	68
9.	7	0	51	0	50	0	0	68
10.	7	51	675	1	50	1	0	68

The data come to us already stset, and we type stset without arguments to examine the current settings. We verify that the id variable has been set as the patient id. We also see that we have 172 records representing 103 subjects, implying multiple records for some subjects. From our listing, we see that multiple records are necessary to accommodate changes in patients' heart-transplant status (pretransplant versus posttransplant).

Residuals and other diagnostic measures, where applicable, will by default take place at the subject level, meaning that (for example) there will be 103 likelihood displacement values for detecting influential subjects (not observations, but subjects).

```
. stcox age posttran surg year
 (output omitted )
. predict ld, ldisplace
(69 missing values generated)
. list id _t0 _t _d age posttran surgery year ld in 1/10
```

	id	_t0	_t	_d	age	posttran	surgery	year	ld
1.	1	0	50	1	30	0	0	67	.0596877
2.	2	0	6	1	51	0	0	68	.0154667
3.	3	0	1	0	54	0	0	68	.
4.	3	1	16	1	54	1	0	68	.0298421
5.	4	0	36	0	40	0	0	68	.
6.	4	36	39	1	40	1	0	68	.0359712
7.	5	0	18	1	20	0	0	68	.1260891
8.	6	0	3	1	54	0	0	68	.0199614
9.	7	0	51	0	50	0	0	68	.
10.	7	51	675	1	50	1	0	68	.0659499

Because here we are not interested in predicting any baseline functions, it is perfectly safe to leave age and year uncentered. The "(69 missing values generated)" message after predict tells us that only 103 out of the 172 observations of ld were filled in; that is, we received only one likelihood displacement per subject. Regardless of the current sorting of the data, the ld value for a subject is stored in the last chronological record for that subject as determined by analysis time, _t.

Patient 4 has two records in the data, one pretransplant and one posttransplant. As such, the ld value for that patient is interpreted as the change in twice the log likelihood due to deletion of both of these observations, that is, the deletion of patient 4 from the study. The interpretation is at the patient level, not the record level.

◁

If, instead, you want likelihood displacement values that you can interpret at the observation level (that is, changes in twice the log likelihood due to deleting one record), you simply add the partial option to the predict command above:

```
. predict ld, ldisplace partial
```

We do not think these kinds of observation-level diagnostics are generally what you would want, but they are available.

In the above, we discussed likelihood displacement values, but the same issue concerning subject-level versus observation-level interpretation also exists with Cox–Snell residuals, martingale residuals, deviance residuals, efficient score residuals, LMAX values, and DFBETAs. Regardless of which diagnostic you examine, this issue of interpretation is the same.

There is one situation where you do want to use the partial option. If you are using martingale residuals to determine functional form and the variable you are thinking of adding varies within subject, then you want to graph the partial martingale residuals against that new variable. Because the variable changes within subject, the martingale residuals should also change accordingly.

Predictions after stcox with the tvc() option

The residuals and diagnostics discussed previously are not available after estimation with `stcox` with the `tvc()` option, which is a convenience option for handling time-varying covariates:

```
. use https://www.stata-press.com/data/r18/drugtr, clear
(Patient survival in drug trial)

. stcox drug age, tvc(age) nolog

        Failure _d: died
  Analysis time _t: studytime

Cox regression with Breslow method for ties

No. of subjects =  48                          Number of obs =        48
No. of failures =  31
Time at risk    = 744
                                               LR chi2(3)    =    33.63
Log likelihood = -83.095036                    Prob > chi2   =   0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
main						
drug	.1059862	.0478178	-4.97	0.000	.0437737	.2566171
age	1.156977	.07018	2.40	0.016	1.027288	1.303037
tvc						
age	.9970966	.0042415	-0.68	0.494	.988818	1.005445

```
Note: Variables in tvc equation interacted with _t.
. predict dev, deviance
this prediction is not allowed after estimation with tvc();
see tvc note for an alternative to the tvc() option
r(198);
```

The above fits a Cox model to the cancer data and includes an interaction of `age` with analysis time, `_t`. Such interactions are useful for testing the proportional-hazards assumption: significant interactions are violations of the proportional-hazards assumption for the variable being interacted with analysis time (or some function of analysis time). That is not the situation here.

In any case, models with `tvc()` interactions do not allow predicting the residuals and diagnostics discussed thus far. The solution in such situations is to forgo the use of `tvc()`, expand the data, and use factor variables to specify the interaction:

```
. generate id = _n

. streset, id(id)
 (output omitted)

. stsplit, at(failures)
(21 failure times)
(534 observations (episodes) created)

. stcox drug age c.age#c._t, nolog
        Failure _d: died
  Analysis time _t: studytime
        ID variable: id

Cox regression with Breslow method for ties

No. of subjects =  48                          Number of obs =      582
No. of failures =  31
Time at risk    = 744
                                               LR chi2(3)    =   33.63
Log likelihood = -83.095036                    Prob > chi2   = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
drug	.1059862	.0478178	-4.97	0.000	.0437737	.2566171
age	1.156977	.07018	2.40	0.016	1.027288	1.303037
c.age#c._t	.9970966	.0042415	-0.68	0.494	.988818	1.005445

```
. predict dev, deviance
(534 missing values generated)

. summarize dev
```

Variable	Obs	Mean	Std. dev.	Min	Max
dev	48	.0658485	1.020993	-1.804876	2.065424

We split the observations, currently one per subject, so that the interaction term is allowed to vary over time. Splitting the observations requires that we first establish a subject id variable. Once that is done, we split the observations with stsplit and the at(failures) option, which splits the records only at the observed failure times. This amount of splitting is the minimal amount required to reproduce our previous Cox model. We then include the interaction term c.age#c._t in our model, verify that our Cox model is the same as before, and obtain our 48 deviance residuals, one for each subject.

Predictions after stcox with the shared() option

A Cox shared frailty model is a Cox model with added group-level random effects such that

$$h_{ij}(t) = h_0(t) \exp(\mathbf{x}_{ij}\boldsymbol{\beta} + \nu_i)$$

with ν_i representing the added effect due to being in group i; see *Cox regression with shared frailty* in [ST] **stcox** for more details. You fit this kind of model by specifying the shared(*varname*) option with stcox, where *varname* identifies the groups. stcox will produce an estimate of $\boldsymbol{\beta}$, its covariance matrix, and an estimate of the variance of the ν_i. What it will not produce are estimates of the ν_i themselves. These you can obtain postestimation with predict.

▷ Example 10: Shared frailty models

In example 10 of [ST] **stcox**, we fit a shared frailty model to data from 38 kidney dialysis patients, measuring the time to infection at the catheter insertion point. Two recurrence times (in days) were measured for each patient.

The estimated ν_i are not displayed in the stcox coefficient table but may be retrieved postestimation by using predict with the effects option:

```
. use https://www.stata-press.com/data/r18/catheter, clear
(Kidney data, McGilchrist and Aisbett, Biometrics, 1991)
. quietly stcox age female, shared(patient)
. predict nu, effects
. sort nu
. list patient nu in 1/2
```

	patient	nu
1.	21	-2.448707
2.	21	-2.448707

```
. list patient nu in 75/L
```

	patient	nu
75.	7	.5187159
76.	7	.5187159

From the results above, we estimate that the least frail patient is patient 21, with $\hat{\nu}_{21} = -2.45$, and that the frailest patient is patient 7, with $\hat{\nu}_7 = 0.52$.

◁

❏ Technical note

When used with shared-frailty models, predict's basehc, basesurv, and basechazard options produce estimates of baseline quantities that are based on the last-step penalized Cox model fit. Therefore, without option atfrailty or atfrailty(), the term "baseline" means that not only the covariates are set to 0 but also the ν_i. If the atfrailty option is specified, all the covariates are set to 0, but the frailties are set to $\exp(\nu_i)$.

Other predictions, such as martingale residuals, are conditional on the estimated frailty variance being fixed and known at the onset.

❏

estat concordance

estat concordance calculates the concordance probability, which is defined as the probability that predictions and outcomes are concordant. estat concordance provides two measures of the concordance probability: Harrell's C and Gönen and Heller's K concordance coefficients. Harrell's C, which is defined as the proportion of all usable subject pairs in which the predictions and outcomes are concordant, is computed by default. Gönen and Heller (2005) propose an alternative measure of concordance, computed when the gheller option is specified, that is not sensitive to the degree of

censoring, unlike Harrell's C coefficient. This estimator is not dependent on the observed event or the censoring time and is a function of only the regression parameters and the covariate distribution, which leads to the asymptotic unbiasedness. estat concordance also reports the Somers's D rank correlation, which is derived by calculating $2C - 1$ for Harrell's C and $2K - 1$ for Gönen and Heller's K.

estat concordance may not be used after a Cox regression model with time-varying covariates and may not be applied to weighted data or to data with delayed entries. The computation of Gönen and Heller's K coefficient is not supported for shared-frailty models, stratified estimation, or multiple-record data.

▷ Example 11: Harrell's C

Using our cancer data, we wish to evaluate the predictive value of the measurement of drug and age. After fitting a Cox regression model, we use estat concordance to calculate Harrell's C index.

```
. use https://www.stata-press.com/data/r18/drugtr, clear
(Patient survival in drug trial)

. stcox drug age

        Failure _d: died
  Analysis time _t: studytime

Iteration 0:  Log likelihood = -99.911448
Iteration 1:  Log likelihood = -83.551879
Iteration 2:  Log likelihood = -83.324009
Iteration 3:  Log likelihood = -83.323546
Refining estimates:
Iteration 0:  Log likelihood = -83.323546

Cox regression with Breslow method for ties

No. of subjects =  48                              Number of obs =      48
No. of failures =  31
Time at risk    = 744
                                                   LR chi2(2)    =  33.18
Log likelihood = -83.323546                        Prob > chi2   = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
drug	.1048772	.0477017	-4.96	0.000	.0430057	.2557622
age	1.120325	.0417711	3.05	0.002	1.041375	1.20526

```
. estat concordance, noshow

Harrell's C concordance statistic

  Number of subjects (N)              =       48
  Number of comparison pairs (P)      =      849
  Number of orderings as expected (E) =      679
  Number of tied predictions (T)      =       15

          Harrell's C = (E + T/2) / P =   0.8086
                          Somers' D =   0.6172
```

The result of stcox shows that the drug results in a lower hazard and therefore a longer survival time, controlling for age and older patients being more likely to die. The value of Harrell's C is 0.8086, which indicates that we can correctly order survival times for pairs of patients 81% of the time on the basis of measurement of drug and age. See *Methods and formulas* for the full definition of concordance.

◁

❑ Technical note

estat concordance does not work after a Cox regression model with time-varying covariates. When the covariates are varying with time, the prognostic score, $\text{PS} = \mathbf{x}\boldsymbol{\beta}$, will not capture or condense the information in given measurements, in which case it does not make sense to calculate the rank correlation between PS and survival time.

❑

▷ Example 12: Gönen and Heller's K

Alternatively, we can obtain Gönen and Heller's estimate of the concordance probability, K. To do so, we specify the gheller option with estat concordance:

```
. estat concordance, noshow gheller

Gonen and Heller's K concordance statistic

  Number of subjects (N)       =       48
          Gonen and Heller's K =    0.7748
                     Somers' D =    0.5496
```

Gönen and Heller's concordance coefficient may be preferred to Harrell's C when censoring is present because Harrell's C can be biased. Because 17 of our 48 subjects are censored, we prefer Gönen and Heller's concordance to Harrell's C.

◁

Stored results

estat concordance stores the following in r():

Scalars

r(N)	number of observations	r(K)	Gönen and Heller's K coefficient
r(n_P)	number of comparison pairs	r(K_s)	smoothed Gönen and Heller's K coefficient
r(n_E)	number of orderings as expected	r(K_s_se)	standard error of the smoothed K coefficient
r(n_T)	number of tied predictions	r(D)	Somers's D coefficient for Harrell's C
r(C)	Harrell's C coefficient	r(D_K)	Somers's D coefficient for Gönen and Heller's K

r(n_P), r(n_E), and r(n_T) are returned only when strata are not specified.

Methods and formulas

Let $\mathbf{x}_i$ be the row vector of covariates for the time interval $(t_{0i}, t_i]$ for the ith observation in the dataset $(i = 1, \ldots, N)$. The Cox partial log-likelihood function, using the default Peto–Breslow method for tied failures is

$$\log L_{\text{breslow}} = \sum_{j=1}^{D} \sum_{i \in D_j} \left[w_i(\mathbf{x}_i\boldsymbol{\beta} + \text{offset}_i) - w_i \log \left\{ \sum_{\ell \in R_j} w_\ell \exp(\mathbf{x}_\ell\boldsymbol{\beta} + \text{offset}_\ell) \right\} \right]$$

where j indexes the ordered failure times t_j $(j = 1, \ldots, D)$, D_j is the set of d_j observations that fail at t_j, d_j is the number of failures at t_j, and R_j is the set of observations k that are at risk at time t_j (that is, all k such that $t_{0k} < t_j \leq t_k$). w_i and offset$_i$ are, respectively, the weight and linear offset for observation i, if specified.

If the Efron method for ties is specified at estimation, the partial log likelihood is

$$
\log L_{\text{efron}} = \sum_{j=1}^{D} \sum_{i \in D_j} \left[\mathbf{x}_i \boldsymbol{\beta} + \text{offset}_i - d_j^{-1} \sum_{k=0}^{d_j-1} \log \left\{ \sum_{\ell \in R_j} \exp(\mathbf{x}_\ell \boldsymbol{\beta} + \text{offset}_\ell) - kA_j \right\} \right]
$$

for $A_j = d_j^{-1} \sum_{\ell \in D_j} \exp(\mathbf{x}_\ell \boldsymbol{\beta} + \text{offset}_\ell)$. Weights are not supported with the Efron method.

At estimation, Stata also supports the exact marginal and exact partial methods for handling ties, but only the Peto–Breslow and Efron methods are supported in regard to the calculation of residuals, diagnostics, and other predictions. As such, only the partial log-likelihood formulas for those two methods are presented above, for easier reference in what follows.

If you specified `efron` at estimation, all predictions are carried out using the Efron method; that is, the handling of tied failures is done analogously to the way it was done when calculating $\log L_{\text{efron}}$. If you specified `breslow` (or nothing, because `breslow` is the default), `exactm`, or `exactp`, all predictions are carried out using the Peto–Breslow method. That is not to say that if you specify `exactm` at estimation, your predictions will be the same as if you had specified `breslow`. The formulas used will be the same, but the parameter estimates at which they are evaluated will differ because those were based on different ways of handling ties.

Define $z_i = \mathbf{x}_i \widehat{\boldsymbol{\beta}} + \text{offset}_i$. Schoenfeld residuals for the pth variable using the Peto–Breslow method are given by

$$
r_{S_{pi}} = \delta_i \left(x_{pi} - a_{pi} \right)
$$

where

$$
a_{pi} = \frac{\sum_{\ell \in R_i} w_\ell x_{p\ell} \exp(z_\ell)}{\sum_{\ell \in R_i} w_\ell \exp(z_\ell)}
$$

δ_i indicates failure for observation i, and x_{pi} is the pth element of $\mathbf{x}_i$. For the Efron method, Schoenfeld residuals are

$$
r_{S_{pi}} = \delta_i \left(x_{pi} - b_{pi} \right)
$$

where

$$
b_{pi} = d_i^{-1} \sum_{k=0}^{d_i-1} \frac{\sum_{\ell \in R_i} x_{p\ell} \exp(z_\ell) - kd_i^{-1} \sum_{\ell \in D_i} x_{p\ell} \exp(z_\ell)}{\sum_{\ell \in R_i} \exp(z_\ell) - kd_i^{-1} \sum_{\ell \in D_i} \exp(z_\ell)}
$$

Schoenfeld residuals are derived from the first derivative of the log likelihood, with

$$
\left. \frac{\partial \log L}{\partial \beta_p} \right|_{\widehat{\boldsymbol{\beta}}} = \sum_{i=1}^{N} r_{S_{pi}} = 0
$$

and only those observations that fail ($\delta_i = 1$) contribute a Schoenfeld residual to the derivative.

For censored observations, Stata stores a missing value for the Schoenfeld residual even though the above implies a value of 0. This is to emphasize that no calculation takes place when the observation is censored.

Scaled Schoenfeld residuals are given by

$$
\mathbf{r}_{S_i}^{*} = \widehat{\boldsymbol{\beta}} + d \ \text{Var}(\widehat{\boldsymbol{\beta}}) \mathbf{r}_{S_i}
$$

where $\mathbf{r}_{S_i} = (r_{S_{1i}}, \ldots, r_{S_{mi}})'$, m is the number of regressors, and d is the total number of failures.

In what follows, we assume the Peto–Breslow method for handling ties. Formulas for the Efron method, while tedious, can be obtained by applying similar principles of averaging across risk sets, as demonstrated above with Schoenfeld residuals.

Efficient score residuals are obtained by

$$r_{E_{pi}} = r_{S_{pi}} - \exp(z_i) \sum_{j:t_{0i}<t_j\leq t_i} \frac{\delta_j w_j (x_{pi} - a_{pj})}{\sum_{\ell \in R_j} w_\ell \exp(z_\ell)}$$

Like Schoenfeld residuals, efficient score residuals are also additive components of the first derivative of the log likelihood. Whereas Schoenfeld residuals are the contributions of each failure, efficient score residuals are the contributions of each observation. Censored observations contribute to the log likelihood (and its derivative) because they belong to risk sets at times when other observations fail. As such, an observation's contribution is twofold: 1) If the observation ends in failure, a risk assessment is triggered, that is, a term in the log likelihood is computed. 2) Whether failed or censored, an observation contributes to risk sets for other observations that do fail. Efficient score residuals reflect both contributions.

The above computes efficient score residuals at the observation level. If you have multiple records per subject and do not specify the `partial` option, then the efficient score residual for a given subject is calculated by summing the efficient scores over the observations within that subject.

Martingale residuals are

$$r_{M_i} = \delta_i - \exp(z_i) \sum_{j:t_{0i}<t_j\leq t_i} \frac{w_j \delta_j}{\sum_{\ell \in R_j} w_\ell \exp(z_\ell)}$$

The above computes martingale residuals at the observation level. If you have multiple records per subject and do not specify the `partial` option, then the martingale residual for a given subject is calculated by summing r_{M_i} over the observations within that subject.

Martingale residuals are in the range $(-\infty, 1)$. Deviance residuals are transformations of martingale residuals designed to have a distribution that is more symmetric about zero. Deviance residuals are calculated using

$$r_{D_i} = \text{sign}(r_{M_i}) \left[-2 \left\{ r_{M_i} + \delta_i \log(\delta_i - r_{M_i}) \right\} \right]^{1/2}$$

These residuals are expected to be symmetric about zero but do not necessarily sum to zero.

The above computes deviance residuals at the observation level. If you have multiple records per subject and do not specify the `partial` option, then the deviance residual for a given subject is calculated by applying the above transformation to the *subject-level* martingale residual.

The estimated baseline hazard contribution is obtained at each failure time as $h_j = 1 - \widehat{\alpha}_j$, where $\widehat{\alpha}_j$ is the solution to

$$\sum_{k \in D_j} \frac{\exp(z_k)}{1 - \widehat{\alpha}_j^{\exp(z_k)}} = \sum_{\ell \in R_j} \exp(z_\ell)$$

(Kalbfleisch and Prentice 2002, eq. 4.34, 115).

The estimated baseline survivor function is

$$\widehat{S}_0(t) = \prod_{j:t_j \leq t} \widehat{\alpha}_j$$

When estimated with no covariates, $\widehat{S}_0(t)$ is the Kaplan–Meier estimate of the survivor function.

The estimated baseline cumulative hazard function, if requested, is related to the baseline survivor function calculation, yet the values of $\widehat{\alpha}_j$ are set at their starting values and are not iterated. Equivalently,

$$\widehat{H}_0(t) = \sum_{j:t_j \leq t} \frac{d_j}{\sum_{\ell \in R_j} \exp(z_\ell)}$$

When estimated with no covariates, $\widehat{H}_0(t)$ is the Nelson–Aalen estimate of the cumulative hazard.

Cox–Snell residuals are calculated with

$$r_{C_i} = \delta_i - r_{M_i}$$

where r_{M_i} are the martingale residuals. Equivalently, Cox–Snell residuals can be obtained with

$$r_{C_i} = \exp(z_i)\widehat{H}_0(t_i)$$

The above computes Cox–Snell residuals at the observation level. If you have multiple records per subject and do not specify the `partial` option, then the Cox–Snell residual for a given subject is calculated by summing r_{C_i} over the observations within that subject.

DFBETAs are calculated with

$$\text{DFBETA}_i = \mathbf{r}_{E_i} \widetilde{\text{Var}}(\widehat{\boldsymbol{\beta}})$$

where $\mathbf{r}_{E_i} = (r_{E_{1i}}, \ldots, r_{E_{mi}})$ is a row vector of efficient score residuals with one entry for each regressor, and $\widetilde{\text{Var}}(\widehat{\boldsymbol{\beta}})$ is the model-based variance matrix of $\widehat{\boldsymbol{\beta}}$.

Likelihood displacement values are calculated with

$$\text{LD}_i = \mathbf{r}_{E_i} \text{Var}(\widehat{\boldsymbol{\beta}})\mathbf{r}'_{E_i}$$

(Collett 2015, 156). In both of the above, $\mathbf{r}_{E_i}$ can represent either one observation or, in multiple-record data, the cumulative efficient score for an entire subject. For the former, the interpretation is that due to deletion of one record; for the latter, the interpretation is that due to deletion of all of a subject's records.

Following Collett (2015, 156), LMAX values are obtained from an eigensystem analysis of

$$\mathbf{B} = \boldsymbol{\Theta} \text{ Var}(\widehat{\boldsymbol{\beta}}) \boldsymbol{\Theta}'$$

where $\boldsymbol{\Theta}$ is the $N \times m$ matrix of efficient score residuals, with element (i, j) representing the jth regressor and the ith observation (or subject). LMAX values are then the absolute values of the elements of the unit-length eigenvector associated with the largest eigenvalue of the $N \times N$ matrix $\mathbf{B}$.

For shared-frailty models, the data are organized into G groups, with the ith group consisting of n_i observations, $i = 1, \ldots, G$. From Therneau and Grambsch (2000, 253–255), for fixed θ, estimates of $\boldsymbol{\beta}$ and $\nu_1, \ldots, \nu_G$ are obtained by maximizing

$$\log L(\theta) = \log L_{\text{Cox}}(\boldsymbol{\beta}, \nu_1, \ldots, \nu_G) + \sum_{i=1}^{G} \left[\frac{1}{\theta} \left\{ \nu_i - \exp(\nu_i) \right\} + \right.$$

$$\left. \left(\frac{1}{\theta} + D_i \right) \left\{ 1 - \log \left(\frac{1}{\theta} + D_i \right) \right\} - \frac{\log \theta}{\theta} + \log \Gamma \left(\frac{1}{\theta} + D_i \right) - \log \Gamma \left(\frac{1}{\theta} \right) \right]$$

where D_i is the number of death events in group i, and $\log L_{\text{Cox}}(\boldsymbol{\beta}, \nu_1, \ldots, \nu_G)$ is the standard Cox partial log likelihood, with the ν_i treated as the coefficients of indicator variables identifying the groups. That is, the jth observation in the ith group has log relative-hazard $\mathbf{x}_{ij}\boldsymbol{\beta} + \nu_i$.

You obtain the estimates of $\nu_1, \ldots, \nu_G$ with `predict`'s `effects` option after `stcox, shared()`.

estat concordance

Harrell's C was proposed by Harrell et al. (1982) and was developed to evaluate the results of a medical test. The C index is defined as the proportion of all usable subject pairs in which the predictions and outcomes are concordant. The C index may be applied to ordinary continuous outcomes, dichotomous diagnostic outcomes, ordinal outcomes, and censored time-until-event response variables.

In predicting the time until death, C is calculated by considering all comparable patient pairs. A pair of patients is comparable if either 1) the two have different values on the time variable, and the one with the lowest value presents a failure, or 2) the two have the same value on the time variable, and exactly one of them presents a failure. If the predicted survival time is larger for the patient who lived longer, the predictions for the pair are said to be concordant with the outcomes. From Fibrinogen Studies Collaboration (2009), Harrell's C is defined as $\sum_k (E_k + T_k/2) / \sum_k (D_k)$, where D_k is the total number of pairs usable for comparison in stratum k, E_k is the number of pairs for which the predictions are concordant with the outcomes and the predictions are not identical in stratum k, and T_k is the number of usable pairs for which the predictions are identical in stratum k. If there are no strata specified, then the formula for Harrell's C reduces to $(E + T/2)/D$.

For a Cox proportional hazards model, the probability that the patient survives past time t is given by $S_0(t)$ raised to the $\exp(\mathbf{x}\boldsymbol{\beta})$ power, where $S_0(t)$ is the baseline survivor function, $\mathbf{x}$ denotes a set of measurements for the patient, and $\boldsymbol{\beta}$ is the vector of coefficients. A Cox regression model is fit by the `stcox` command. The hazard ratio, $\exp(\mathbf{x}\boldsymbol{\beta})$, is obtained by `predict` after `stcox`. Because the predicted survival time and the predicted survivor function are one-to-one functions of each other, the predicted survivor function can be used to calculate C instead of the predicted survival time. The predicted survivor function decreases when the predicted hazard ratio increases; therefore, Harrell's C can be calculated by computing E, T, and D, based on the observed outcomes and the predicted hazard ratios.

C takes a value between 0 and 1. A value of 0.5 indicates no predictive discrimination, and values of 0 or 1.0 indicate perfect separation of subjects with different outcomes. See Harrell, Lee, and Mark (1996) for more details. Somers's D rank correlation is calculated by $2C - 1$; see Newson (2002) for a discussion of Somers's D.

In the presence of censoring, Harrell's C coefficient tends to be biased. An alternative measure of concordance that is asymptotically unbiased with censored data was proposed by Gönen and Heller (2005). This estimator does not depend on observed time directly and is a function of only the regression parameters and the covariate distribution, which leads to its asymptotic unbiasedness and thus robustness to the degree of censoring.

Let $\Delta\mathbf{x}_{ij}$ be the pairwise difference $\mathbf{x}_i - \mathbf{x}_j$. Then Gönen and Heller's concordance probability estimator is given by

$$K \equiv K_N(\widehat{\boldsymbol{\beta}}) = \frac{2}{N(N-1)} \sum_{i<j} \sum \left\{ \frac{I(\Delta\mathbf{x}_{ji}\widehat{\boldsymbol{\beta}} \leq 0)}{1 + \exp(\Delta\mathbf{x}_{ji}\widehat{\boldsymbol{\beta}})} + \frac{I(\Delta\mathbf{x}_{ij}\widehat{\boldsymbol{\beta}} < 0)}{1 + \exp(\Delta\mathbf{x}_{ij}\widehat{\boldsymbol{\beta}})} \right\} \tag{1}$$

where $I(\cdot)$ is the indicator function. Somers's D rank correlation is calculated by $2K - 1$.

The concordance probability estimator (1) involves indicator functions and thus is a nonsmooth function for which the asymptotic standard error cannot be computed directly. To obtain the standard error, a smooth approximation to this estimator is considered:

$$\widetilde{K} \equiv \widetilde{K}_N(\widehat{\boldsymbol{\beta}}) = \frac{2}{N(N-1)} \sum_{i<j} \sum \left\{ \frac{\Phi(-\Delta\mathbf{x}_{ji}\widehat{\boldsymbol{\beta}}/h)}{1 + \exp(\Delta\mathbf{x}_{ji}\widehat{\boldsymbol{\beta}})} + \frac{\Phi(-\Delta\mathbf{x}_{ij}\widehat{\boldsymbol{\beta}}/h)}{1 + \exp(\Delta\mathbf{x}_{ij}\widehat{\boldsymbol{\beta}})} \right\} \tag{2}$$

where $\Phi(\cdot)$ is a standard normal distribution function, $h = 0.5\widehat{\sigma}N^{-1/3}$ is a smoothing bandwidth, and $\widehat{\sigma}$ is the estimated standard deviation of the subject-specific linear predictors $\mathbf{x}_i\widehat{\boldsymbol{\beta}}$.

The asymptotic standard error is then computed using a first-order Taylor series expansion of (2) around the true parameter $\boldsymbol{\beta}$; see Gönen and Heller (2005) for computational details.

References

Cattaneo, M. D., P. Malighetti, and D. Spinelli. 2017. Estimating receiver operative characteristic curves for time-dependent outcomes: The stroccurve package. *Stata Journal* 17: 1015–1023.

Cefalu, M. S. 2011. Pointwise confidence intervals for the covariate-adjusted survivor function in the Cox model. *Stata Journal* 11: 64–81.

Collett, D. 2015. *Modelling Survival Data in Medical Research*. 3rd ed. Boca Raton, FL: Chapman and Hall/CRC.

Fibrinogen Studies Collaboration. 2009. Measures to assess the prognostic ability of the stratified Cox proportional hazards model. *Statistics in Medicine* 28: 389–411. https://doi.org/10.1002/sim.3378.

Gönen, M., and G. Heller. 2005. Concordance probability and discriminatory power in proportional hazards regression. *Biometrika* 92: 965–970. https://doi.org/10.1093/biomet/92.4.965.

Harrell, F. E., Jr., R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati. 1982. Evaluating the yield of medical tests. *Journal of the American Medical Association* 247: 2543–2546. https://doi.org/10.1001/jama.1982.03320430047030.

Harrell, F. E., Jr., K. L. Lee, and D. B. Mark. 1996. Multivariable prognostic models: Issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine* 15: 361–387. https://doi.org/10.1002/(SICI)1097-0258(19960229)15:4⟨361::AID-SIM168⟩3.0.CO;2-4.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd ed. New York: Springer.

Mansuy, R. 2009. The origins of the word "martingale". *Electronic Journal for History of Probability and Statistics* 5: 1–10. http://www.jehps.net/juin2009/Mansuy.pdf.

Mazliak, L., and G. Shafer. 2009. The splendors and miseries of martingales. *Electronic Journal for History of Probability and Statistics* 5: 1–5. http://www.jehps.net/juin2009/MazliakShafer.pdf.

Metzger, S. K., and B. T. Jones. 2018. mstatecox: A package for simulating transition probabilities from semiparametric multistate survival models. *Stata Journal* 18: 533–563.

———. 2021. Properly calculating estat phtest in the presence of stratified hazards. *Stata Journal* 21: 1028–1033.

Newson, R. B. 2002. Parameters behind "nonparametric" statistics: Kendall's tau, Somers' D and median differences. *Stata Journal* 2: 45–64.

——. 2006. Confidence intervals for rank statistics: Somers' D and extensions. *Stata Journal* 6: 309–334.

——. 2010. Comparing the predictive powers of survival models using Harrell's C or Somers' D. *Stata Journal* 10: 339–358.

Schoenfeld, D. A. 1982. Partial residuals for the proportional hazards regression model. *Biometrika* 69: 239–241. https://doi.org/10.2307/2335876.

Schwartzman, S. 1994. *The Words of Mathematics: An Etymological Dictionary of Mathematical Terms Used in English.* Washington, DC: Mathematical Association of America.

Therneau, T. M., and P. M. Grambsch. 2000. *Modeling Survival Data: Extending the Cox Model.* New York: Springer.

Wei, Y., and P. Royston. 2017. Reconstructing time-to-event data from published Kaplan–Meier curves. *Stata Journal* 17: 786–802.

Also see

[ST] **stcox** — Cox proportional hazards model

[ST] **estat gofplot** — Goodness-of-fit plots after streg, stcox, stintreg, or stintcox

[ST] **stcox PH-assumption tests** — Tests of proportional-hazards assumption after stcox

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[LASSO] **lassogof** — Goodness of fit after lasso for prediction

[U] **20 Estimation and postestimation commands**

Title

Description

stcrreg fits, via maximum likelihood, competing-risks regression models on st data, according to the method of Fine and Gray (1999). Competing-risks regression posits a model for the subhazard function of a failure event of primary interest. In the presence of competing failure events that impede the event of interest, a standard analysis using Cox regression (see [ST] **stcox**) is able to produce incidence-rate curves that either 1) are appropriate only for a hypothetical universe where competing events do not occur or 2) are appropriate for the data at hand, yet the effects of covariates on these curves are not easily quantified. Competing-risks regression, as performed using stcrreg, provides an alternative model that can produce incidence curves that represent the observed data and for which describing covariate effects is straightforward.

stcrreg can be used with single- or multiple-record data. stcrreg cannot be used when you have multiple failures per subject.

Quick start

Competing-risks regression with covariates x1 and x2 and competing event defined by $fvar = 2$
   using data that are stset with failure $fvar = 1$

    stcrreg x1 x2, compete(fvar==2)

Same as above, but report coefficients instead of subhazard ratios

    stcrreg x1 x2, compete(fvar==2) noshr

With cluster–robust standard errors for clustering by levels of cvar

    stcrreg x1 x2, compete(fvar==2) vce(cluster cvar)

Competing events defined by $fvar = 2$, $fvar = 3$, and $fvar = 4$

    stcrreg x1 x2, compete(fvar==2 3 4)

Specify indicator variable compvar identifying competing events

    stcrreg x1 x2, compete(compvar)

Menu

Statistics > Survival analysis > Regression models > Competing-risks regression

Syntax

> <u>stcrreg</u> [*indepvars*] [*if*] [*in*], <u>compete</u>(*crvar*[==*numlist*]) [*options*]

options	Description
__Model__	
* <u>compete</u>(*crvar*[==*numlist*])	specify competing-risks event(s)
<u>offset</u>(*varname*)	include *varname* in model with coefficient constrained to 1
<u>constraints</u>(*constraints*)	apply specified linear constraints
__Time varying__	
tvc(*tvarlist*)	specify covariates to be interacted with time
texp(*exp*)	specify a function of time; default is texp(_t)
__SE/Robust__	
vce(*vcetype*)	*vcetype* may be <u>robust</u>, <u>cl</u>uster *clustvar*, <u>boot</u>strap, or <u>jack</u>knife
<u>noadjust</u>	do not use standard degree-of-freedom adjustment
__Reporting__	
<u>level</u>(#)	set confidence level; default is level(95)
noshr	report coefficients, not subhazard ratios
noshow	do not show st setting information
<u>nohead</u>er	suppress header from coefficient table
<u>notable</u>	suppress coefficient table
<u>nodisplay</u>	suppress output; iteration log is still displayed
<u>nocnsreport</u>	do not display constraints
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
__Maximization__	
maximize_options	control the maximization process; seldom used
<u>collinear</u>	keep collinear variables
<u>coeflegend</u>	display legend instead of statistics

* <u>compete</u>(*crvar*[==*numlist*]) is required.

You must stset your data before using stcrreg; see [ST] **stset**.

varlist and *tvarlist* may contain factor variables; see [U] **11.4.3 Factor variables**.

bootstrap, by, collect, fp, jackknife, mfp, mi estimate, nestreg, statsby, and stepwise are allowed; see [U] **11.1.10 Prefix commands**.

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix; see [MI] **mi estimate**.

Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**. In multiple-record data, weights are applied to subjects as a whole, not to individual observations. iweights are treated as fweights that can be noninteger, but not negative.

collinear and coeflegend do not appear in the dialog box.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

    ⎡ Model ⎤

compete(*crvar*[==*numlist*]) is required and specifies the events that are associated with failure due
    to competing risks.

    If compete(*crvar*) is specified, *crvar* is interpreted as an indicator variable; any nonzero, nonmissing
    values are interpreted as representing competing events.

    If compete(*crvar*==*numlist*) is specified, records with *crvar* taking on any of the values in *numlist*
    are assumed to be competing events.

    The syntax for compete() is the same as that for stset's failure() option. Use stset,
    failure() to specify the failure event of interest, that is, the failure event you wish to model
    using stcox, streg, stcrreg, or whatever. Use stcrreg, compete() to specify the event or
    events that compete with the failure event of interest. Competing events, because they are not the
    failure event of primary interest, must be stset as censored.

    If you have multiple records per subject, only the value of *crvar* for the last chronological record
    for each subject is used to determine the event type for that subject.

offset(*varname*), constraints(*constraints*); see [R] **Estimation options**.

    ⎡ Time varying ⎤

tvc(*tvarlist*) specifies the variables to be included in the model as an interaction with a function
    of time to form time-varying covariates. During estimation, these variables are interacted with
    analysis time or with a function of analysis time specified in the texp() option.

texp(*exp*) is used in conjunction with tvc(*tvarlist*) to specify the function of analysis time that
    should be used to multiply covariates specified in the tvc() option to include in the model time-
    varying covariates that are deterministic functions of time. For example, specifying texp(ln(_t))
    would cause the covariates in option tvc() to be multiplied by the logarithm of analysis time. If
    tvc(*tvarlist*) is used without texp(*exp*), Stata understands that you mean texp(_t) and thus
    multiplies the covariates by the analysis time.

    Both tvc(*tvarlist*) and texp(*exp*) are explained more in *Option tvc() and testing the proportional-
    subhazards assumption* below.

    ⎡ SE/Robust ⎤

vce(*vcetype*) specifies the type of standard error reported, which includes types that are robust to some
    kinds of misspecification (robust), that allow for intragroup correlation (cluster *clustvar*), and that
    use bootstrap or jackknife methods (bootstrap, jackknife); see [R] *vce_option*. vce(robust)
    is the default in single-record-per-subject st data. For multiple-record st data, vce(cluster *idvar*)
    is the default, where *idvar* is the ID variable previously stset.

    Standard Hessian-based standard errors—*vcetype* oim—are not statistically appropriate for this
    model and thus are not allowed.

noadjust is for use with vce(robust) or vce(cluster *clustvar*). noadjust prevents the estimated
    variance matrix from being multiplied by $N/(N-1)$ or $g/(g-1)$, where g is the number of
    clusters. The default adjustment is somewhat arbitrary because it is not always clear how to count
    observations or clusters. In such cases, however, the adjustment is likely to be biased toward 1,
    so we would still recommend making it.

___Reporting___

`level(`*#*`)`; see [R] **Estimation options**.

`noshr` specifies that coefficients be displayed rather than exponentiated coefficients or subhazard ratios. This option affects only how results are displayed and not how they are estimated. `noshr` may be specified at estimation time or when redisplaying previously estimated results (which you do by typing `stcrreg` without a variable list).

`noshow` prevents `stcrreg` from showing the key st variables. This option is seldom used because most people type `stset, show` or `stset, noshow` to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

`noheader` suppresses the header information from the output. The coefficient table is still displayed. `noheader` may be specified at estimation time or when redisplaying previously estimated results.

`notable` suppresses the table of coefficients from the output. The header information is still displayed. `notable` may be specified at estimation time or when redisplaying previously estimated results.

`nodisplay` suppresses the output. The iteration log is still displayed.

`nocnsreport`; see [R] **Estimation options**.

display_options: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(`*#*`)`, `fvwrapon(`*style*`)`, `cformat(`%*fmt*`)`, `pformat(`%*fmt*`)`, `sformat(`%*fmt*`)`, and `nolstretch`; see [R] **Estimation options**.

___Maximization___

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(`*#*`)`, [`no`]`log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(`*#*`)`, `ltolerance(`*#*`)`, `nrtolerance(`*#*`)`, `nonrtolerance`, and `from(`*init_specs*`)`; see [R] **Maximize**. These options are seldom used.

The following options are available with `stcrreg` but are not shown in the dialog box:

`collinear`, `coeflegend`; see [R] **Estimation options**.

Remarks and examples

This section provides a summary of what can be done with `stcrreg`. For a more general tutorial on competing-risks analysis, see Cleves, Gould, and Marchenko (2016, chap. 17).

Remarks are presented under the following headings:

The case for competing-risks regression

In this section, we provide a brief history and literature review of competing-risks analysis, and provide the motivation behind the `stcrreg` model. If you know you want to use `stcrreg` and are anxious to get started, you can safely skip this section.

Based on the method of Fine and Gray (1999), competing-risks regression provides a useful alternative to Cox regression (Cox 1972) for survival data in the presence of competing risks. Consider the usual survival analysis where one measures time-to-failure as a function of experimental or observed factors. For example, we may be interested in measuring time from initial treatment to recurrence of breast cancer in relation to factors such as treatment type and smoking status. The term *competing risk* refers to the chance that instead of cancer recurrence, you will observe a *competing event*, for example, death. The competing event, death, impedes the occurrence of the *event of interest*, breast cancer. This is not to be confused with the usual right-censoring found in survival data, such as censoring due to loss to follow-up. When subjects are lost to follow-up, they are still considered at risk of recurrent breast cancer—it is just that the researcher is not in a position to record the precise time that it happens. In contrast, death is a permanent condition that prevents future breast cancer. While censoring merely obstructs you from observing the event of interest, a competing event prevents the event of interest from occurring altogether. Because competing events are distinct from standard censorings, a competing-risks analysis requires some new methodology and some caution when interpreting the results from the old methodology.

Putter, Fiocco, and Geskus (2007) and Gichangi and Vach (2005) provide excellent tutorials covering the problem of competing risks, nonparametric estimators and tests, competing-risks regression, and the more general multistate models. Textbook treatments of competing-risks analysis can be found within Andersen et al. (1993), Klein and Moeschberger (2003), Therneau and Grambsch (2000), and Marubini and Valsecchi (1995). The texts by Crowder (2001) and Pintilie (2006) are devoted entirely to the topic. In what follows, we assume that you are familiar with the basic concepts of survival analysis, for example, hazard functions and Kaplan–Meier curves. For such an introduction to survival analysis aimed at Stata users, see Cleves, Gould, and Marchenko (2016).

Without loss of generality, assume a situation where there is only one event that competes with the failure event of interest. Before analyzing the problem posed by competing-risks data—the problem `stcrreg` proposes to solve—we first formalize the mechanism behind it. Ignoring censoring for the moment, recording a failure time in a competing-risks scenario can be represented as observing the minimum of two potential failure times: the time to the event of interest, T_1, and the time to the competing event, T_2. The problem of competing risks then becomes one of understanding the nature of the bivariate distribution of (T_1, T_2), and in particular the correlation therein. Although conceptually simple, unfortunately this joint distribution cannot be identified by the data (Pepe and Mori 1993; Tsiatis 1975; Gail 1975). If you get to observe only the minimum, you are getting only half the picture.

An alternate representation of the competing-risks scenario that relies on quantities that are data-identifiable is described by Beyersman et al. (2009). In that formulation, we consider the hazard for the event of interest, $h_1(t)$, and that for the competing event, $h_2(t)$. Both hazards can be estimated from available data and when combined form a total hazard that any event will occur equal to $h(t) = h_1(t) + h_2(t)$. As risk accumulates according to $h(t)$, event times T are observed. Whether these events turn out to be failures of interest (type 1) or competing events (type 2) is determined by the two component hazards at that precise time. The event will be a failure of interest with probability $h_1(T)/\{h_1(T) + h_2(T)\}$, or a competing event with probability one minus that.

Instead of focusing on the survivor function for the event of interest, $P(T > t$ and event type 1), when competing risks are present you want to focus on the failure function, $P(T \le t$ and event type 1), also known as the *cumulative incidence function* (CIF). That is because you will not know what type of event will occur until after it has occurred. It makes more sense to ask "What is the probability of breast cancer within 5 months?" than to ask "What is probability that nothing happens before 5 months, and that when something does happen, it will be breast cancer and not death?"

Much of the literature on competing risks focuses on the inadequacy of the Kaplan–Meier (1958) estimator (which we refer to as KM) as a measure of prevalence for the event of interest. Among

others, Gooley et al. (1999) point out that $1{-}$KM is a biased estimate of the CIF. The bias results from KM treating competing events as if they were censored. That is, subjects that experience competing events are treated as if they could later experience the event of interest, even though that is impossible. Although you could interpret $1{-}$KM as the probability of a type 1 failure in a hypothetical setting where type 2 failures do not occur, this requires you to assume that $h_1(t)$ remains unchanged given that $h_2(t) = 0$, a rather strong and untestable assumption. Regardless of whether the independence assumption holds, $1{-}$KM is still not representative of the data at hand, under which competing events do take place.

As such, $1{-}$KM should be rejected in favor of the *cumulative incidence estimator* of the CIF; see Coviello and Boggess (2004) for a Stata-specific presentation. The cumulative incidence estimator is superior to $1{-}$KM because it acknowledges that cumulative incidence is a function of both cause-specific hazards, $h_1(t)$ and $h_2(t)$. Conversely, $1{-}$KM treats the CIF as a function solely of $h_1(t)$.

When you have covariates, you can use `stcox` to perform regression on $h_1(t)$ by treating failures of type 2 as censored, on $h_2(t)$ by treating failures of type 1 as censored, or on $h_1(t)$ and $h_2(t)$ simultaneously by using the method of data duplication described by Lunn and McNeil (1995) and Cleves (1999). Because cause-specific hazards are identified by the data, all three of the above analyses are suitable for estimating how covariates affect the mechanism behind a given type of failure. For example, if you are interested in how smoking affects breast cancer in general terms (competing death notwithstanding), then a Cox model for $h_1(t)$ that treats death as censored is perfectly valid; see Pintilie (2007).

If you are interested in the incidence of breast cancer, however, you want to use a Cox model that models both $h_1(t)$ and $h_2(t)$, because the CIF for breast cancer will likely depend on both. Based on the fitted model, you will have a hard time spotting the effects of covariates on cumulative incidence, because the covariates can affect $h_1(t)$ and $h_2(t)$ differently, and the CIF is a nonlinear function of these effects and of the baseline hazards. Whether increasing a covariate increases or decreases the cumulative incidence depends on time and on the nominal value of that covariate, as well as on the values of the other covariates. There is no way to determine the full effects of the covariates by just looking at the model coefficients. You would have to estimate and graph the CIF for various sets of covariate values, and this requires a bit of programming; see example 4.

An alternative model for the CIF that does make it easy to see the effects of covariates is that due to Fine and Gray (1999). They specify a model for the *hazard of the subdistribution* (Gray 1988), formally defined for failure type 1 as

$$\overline{h}_1(t) = \lim_{\delta \to 0} \left\{ \frac{P(t < T \le t + \delta \text{ and event type 1}) \mid T > t \text{ or } (T \le t \text{ and not event type 1})}{\delta} \right\}$$

Less formally, think of this hazard as that which generates failure events of interest while keeping subjects who experience competing events "at risk" so that they can be adequately counted as not having any chance of failing. The advantage of modeling the subdistribution hazard, or *subhazard*, is that you can readily calculate the CIF from it;

$$\text{CIF}_1(t) = 1 - \exp\{-\overline{H}_1(t)\}$$

where $\overline{H}_1(t) = \int_0^t \overline{h}_1(t)dt$ is the *cumulative subhazard*.

Competing-risks regression performed in this manner using `stcrreg` is quite similar to Cox regression performed using `stcox`. The model is semiparametric in that the baseline subhazard $\overline{h}_{1,0}(t)$ (that for covariates set to zero) is left unspecified, while the effects of the covariates x are assumed to be proportional:

$$\overline{h}_1(t|\mathbf{x}) = \overline{h}_{1,0}(t) \exp(\mathbf{x}\boldsymbol{\beta})$$

Estimation with stcrreg will produce estimates of β, or exponentiated coefficients known as *subhazard ratios*. A positive (negative) coefficient means that the effect of increasing that covariate is to increase (decrease) the subhazard and thus increase (decrease) the CIF across the board.

Estimates of the baseline cumulative subhazard and of the baseline CIF are available via predict after stcrreg; see [ST] **stcrreg postestimation**. Because proportionality holds for cumulative subhazards as well, adjusting the baseline cumulative hazard and baseline CIF for a given set of covariate values is quite easy and, in fact, done automatically for you by stcurve; see [ST] **stcurve**.

Using stcrreg

If you have used stcox before, stcrreg will look very familiar.

▷ Example 1: Cervical cancer study

Pintilie (2006, sec. 1.6.2) describes data from 109 cervical cancer patients that were treated at a cancer center between 1994 and 2000. The patients were treated and then the time in years until relapse or loss to follow-up was recorded. Relapses were recorded as either "local" if cancer relapsed in the pelvis, or "distant" if cancer recurred elsewhere but not in the pelvis. Patients who did not respond to the initial treatment were considered to have relapsed locally after one day.

```
. use https://www.stata-press.com/data/r18/hypoxia
(Hypoxia study)

. describe

Contains data from https://www.stata-press.com/data/r18/hypoxia.dta
 Observations:             109                    Hypoxia study
    Variables:              16                    7 Apr 2022 09:44
                                                  (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
stnum	int	%8.0g		Patient ID
age	byte	%8.0g		Age (years)
hgb	int	%8.0g		Hemoglobin (g/l)
tumsize	float	%9.0g		Tumor size (cm)
ifp	float	%9.0g		Interstitial fluid pressure (marker, mmHg)
hp5	float	%9.0g		Hypoxia marker (percentage of meas. < 5 mmHg)
pelvicln	str1	%9s		Pelvic node involvement: N=Negative, E=Equivocal, Y=Positive
resp	str2	%9s		Response after treatment: CR=Complete response, NR=No response
pelrec	byte	%9.0g	yesno	Pelvic disease observed
disrec	byte	%9.0g	yesno	Distant disease observed
survtime	float	%9.0g		Time from diagnosis to death or last follow-up time (yrs)
stat	byte	%8.0g		Status at last follow-up: 0=Alive, 1=Dead
dftime	float	%9.0g		Time from diagnosis to first failure or last follow-up (yrs)
dfcens	byte	%8.0g		Censoring variable: 1=Failure, 0=Censored
failtype	byte	%8.0g		Failure type: 1 if pelrec, 2 if disrec & not pelrec, 0 otherwise
pelnode	byte	%8.0g		1 if pelvic nodes negative or equivocal

```
Sorted by:
```

The dftime variable records analysis time in years and the failtype variable records the type of event observed: 0 for loss to follow-up (censored), 1 for a local relapse, and 2 for a distant relapse. Among the covariates used in the analysis were a hypoxia marker (hp5) that measures the degree of oxygenation in the tumor, interstitial fluid pressure (ifp), tumor size (tumsize), and an indicator of pelvic node involvement (pelnode == 0 if positive involvement and pelnode == 1 otherwise). The main goal of the study was to determine whether ifp and hp5 influence the outcome, controlling for the other covariates. Following Pintilie (2006), we focus on ifp and not on hp5. For more details regarding this study and the process behind the measured data, see Fyles et al. (2002) and Milosevic et al. (2001).

We wish to fit a competing-risks model that treats a local relapse as the event of interest and a distant relapse as the competing event. Although a distant relapse does not strictly prevent a future local relapse, presumably, the treatment protocol changed based on which event was first observed. As such, both events can be treated as competing with one another because the conditions of the study ended once any relapse was observed. Because no deaths occurred before first relapse, death is not considered a competing event in this analysis.

To fit the model, we first `stset` the data and specify that a local relapse, `failtype == 1`, is the event of interest. We then specify to `stcrreg` the covariates and that a distant relapse (`failtype == 2`) is a competing event.

```
. stset dftime, failure(failtype == 1)
 (output omitted )

. stcrreg ifp tumsize pelnode, compete(failtype == 2)

        Failure _d: failtype==1
  Analysis time _t: dftime

Iteration 0:  Log pseudolikelihood = -138.67925
Iteration 1:  Log pseudolikelihood = -138.53082
Iteration 2:  Log pseudolikelihood =  -138.5308
Iteration 3:  Log pseudolikelihood =  -138.5308

Competing-risks regression                     No. of obs      =        109
                                               No. of subjects =        109
Failure event:    failtype == 1               No. failed      =         33
Competing event: failtype == 2                No. competing   =         17
                                               No. censored    =         59

                                               Wald chi2(3)    =      33.21
Log pseudolikelihood =  -138.5308              Prob > chi2     =     0.0000
```

_t	SHR	Robust std. err.	z	P>|z|	[95% conf. interval]	
ifp	1.033206	.0178938	1.89	0.059	.9987231	1.068879
tumsize	1.297332	.1271191	2.66	0.008	1.070646	1.572013
pelnode	.4588123	.1972067	-1.81	0.070	.1975931	1.065365

From the above we point out the following:

- When we `stset` the data, distant relapses were set as censored because they are not the event of interest and any standard, noncompeting-risks analysis would want to treat them as censored. `stcrreg` option `compete()` tells Stata which of these "censored" events are actually competing events that require special consideration in a competing-risks regression. Because competing events are not the event of interest, `stcrreg` will issue an error if competing events are not `stset` as censored.

- `stcrreg` lists the event code(s) for the event of interest under "`Failure event(s):`" and the competing event code(s) under "`Competing event(s):`". The syntax for `stset` and `stcrreg` allows you to have multiple codes for both. For competing events, multiple event codes can be devoted entirely to one competing event type, many competing event types, or some combination of both. The methodology behind `stcrreg` extends to more than one competing event type and is concerned only with whether events are competing events, not with their exact type. The focus is on the event of interest.

- We see that out of the 109 patients, 33 experienced a local relapse, 17 experienced a distant relapse, and the remaining 59 were lost to follow-up before any relapse.

- In the column labeled "SHR" are the estimated subhazard ratios, and you interpret these similarly to hazard ratios in Cox regression. Because the estimated subhazard ratio for `ifp` is greater than 1, higher interstitial fluid pressures are associated with higher incidence of local relapses controlling for tumor size, pelvic node involvement, and the fact that distant relapses can also occur. However, this effect is not highly significant.

- To see the estimated coefficients instead of subhazard ratios, use the `noshr` option either when fitting the model or when replaying results.

- Standard errors are listed as "Robust", even though we did not specify any sampling weights, vce(robust), or vce(cluster *clustvar*). As mentioned in the previous section, competing-risks regression works by keeping subjects who experience competing events at risk so that they can be adequately counted as having no chance of failing. Doing so requires a form of sample weighting that invalidates the usual model-based standard errors; see *Methods and formulas*. Robust standard errors are conventional in stcrreg.

- The output lists a "log pseudolikelihood" rather than the standard log likelihood. This is also a consequence of the inherent sample weighting explained in the previous bullet. The log pseudolikelihood is used as a maximization criterion to obtain parameter estimates, but is not representative of the distribution of the data. For this reason, likelihood-ratio (LR) tests (the lrtest command) are not valid after stcrreg. Use Wald tests (the test command) instead.

As mentioned above, you can use the noshr option to obtain coefficients instead of subhazard ratios.

```
. stcrreg, noshr
Competing-risks regression                       No. of obs       =          109
                                                 No. of subjects  =          109
Failure event:   failtype == 1                   No. failed       =           33
Competing event: failtype == 2                   No. competing    =           17
                                                 No. censored     =           59
                                                 Wald chi2(3)     =        33.21
Log pseudolikelihood =  -138.5308                Prob > chi2      =       0.0000
```

_t	Coefficient	Robust std. err.	z	P>|z|	[95% conf. interval]	
ifp	.0326664	.0173188	1.89	0.059	-.0012777	.0666105
tumsize	.2603096	.0979851	2.66	0.008	.0682623	.4523568
pelnode	-.779114	.4298199	-1.81	0.070	-1.621546	.0633175

Just as with stcox, this model has no constant term. It is absorbed as part of the baseline subhazard, which is not directly estimated.

◁

▷ Example 2: CIF curves after stcrreg

In the above analysis, we stated that with increased interstitial fluid pressure comes an increase in the incidence of local relapses in the presence of possible distant relapses. To demonstrate this visually, we use stcurve to compare two CIF curves: one for ifp == 5 and one for ifp == 20. For both curves, we assume positive pelvic node involvement (pelnode==0) and tumor size set at the mean over the data.

```
. stcurve, cif at(ifp=(5 20) pelnode=0)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```
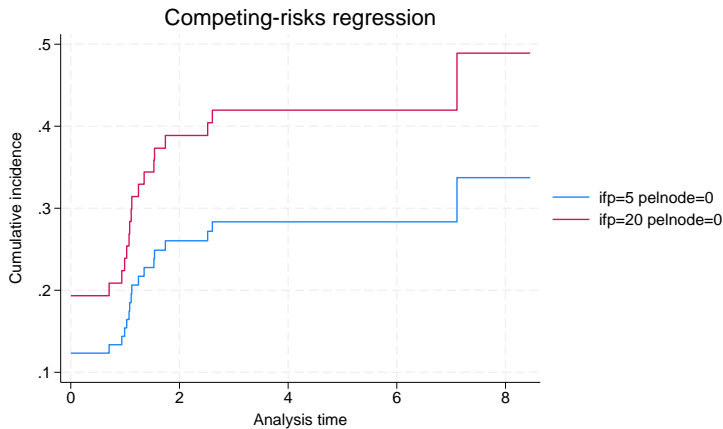

For positive pelvic node involvement and mean tumor size, the probability of local relapse within 2 years is roughly 26% when the interstitial fluid pressure is 5 mmHg and near 40% when this is increased to 20 mmHg. Both probabilities take into account the possibility that a distant relapse could occur instead.

◁

Multiple competing-event types

Competing-risks regression generalizes to the case where more than one type of event competes with the event of interest. If you have such data, after you stset the failure event of interest, you can lump together all competing event codes into the compete() option of stcrreg. It does not matter whether multiple codes represent the same competing-event type, or if they represent multiple types. The results will be the same.

▷ Example 3: UDCA in patients with PBC

Therneau and Grambsch (2000, sec. 8.4.3) analyze data from patients with primary biliary cirrhosis (PBC), a chronic liver disease characterized by progressive destruction of the bile ducts. Data were obtained from 170 patients in a randomized double-blind trial conducted at the Mayo Clinic from 1988 to 1992. The trial was for a new treatment, ursodeoxycholic acid (UDCA; Lindor et al. [1994]).

```
. use https://www.stata-press.com/data/r18/udca, clear
(Randomized trial of UDCA in PBC)

. describe

Contains data from https://www.stata-press.com/data/r18/udca.dta
 Observations:              188                   Randomized trial of UDCA in PBC
    Variables:                8                   3 Apr 2022 09:37
                                                  (_dta has notes)

Variable       Storage    Display    Value
    name          type     format    label     Variable label

id              int       %9.0g                Patient ID
entry           int       %td                  Date of enrollment
eventtime       float     %td                  Date of first event or loss to
                                                  follow-up
treat           byte      %9.0g                0=placebo 1=UDCA
stage           byte      %9.0g                Histologic stage: 0=stage 1/2 at
                                                  entry 1=stage 3/4
lbili           float     %9.0g                log(bilirubin value)
etype           byte      %9.0g      event     Event type (see notes)
wt              double    %4.2f                Observation weight

Sorted by: id
```

The etype variable is coded as any of eight distinct event types (or no event) according to table 1.

Table 1. Event codes for the etype variable

Event code	Event type
0	No event (censored)
1	Death
2	Transplant
3	Histologic progression
4	Development of varices
5	Development of ascites
6	Development of encephalopathy
7	Doubling of bilirubin
8	Worsening of symptoms

Cleves (1999) analyzed these data by estimating the cause-specific hazards for each of the eight events. In the version of the data used there, the time at which any adverse event occurred was recorded, but here we record only the time of the first adverse event for each patient. We do so because we wish to perform a competing-risks analysis where we are interested in the time to the first adverse event and the type of that event. The events compete because only one can be first.

We are interested in whether treatment will decrease the incidence of histologic progression (etype == 3) as the first adverse outcome, in reference to treatment (treat), the logarithm of bilirubin level (lbili), and histologic stage at entry (stage). Because the patients entered the study at different times (entry), when stsetting the data we must specify this variable as the origin, or onset of risk.

The competing-risks analysis described above could thus proceed as follows:

```
. stset eventtime, failure(etype == 3) origin(entry)
. stcrreg treat lbili stage, compete(etype == 1 2 4 5 6 7 8)
```

except for one minor complication. Some patients experienced multiple "first events", and thus ties exist. For example, consider patient 8 who experienced four adverse events at the same time:

```
. list if id == 8
```

	id	entry	eventtime	treat	stage	lbili	etype	wt
8.	8	25may1988	02jul1990	0	1	1.629241	Ascites	0.25
9.	8	25may1988	02jul1990	0	1	1.629241	Ence	0.25
10.	8	25may1988	02jul1990	0	1	1.629241	Bili_2	0.25
11.	8	25may1988	02jul1990	0	1	1.629241	Worse	0.25

While most patients are represented by one record each, patients with multiple first events are represented by multiple records. Rather than break ties arbitrarily, we take advantage of how importance weights (iweights) are handled by stcrreg. Importance weights are treated like frequency weights, but they are allowed to be noninteger. As such, we define the weight variable (wt) to equal one for single-record patients and to equal one divided by the number of tied events for multiple-record patients. In this way, each patient contributes a total weight of one observation.

The only further modification we need is to specify vce(cluster id) so that our standard errors account for the correlation within multiple records on the same patient.

```
. stset eventtime [iw=wt], failure(etype == 3) origin(entry)
 (output omitted )
. stcrreg treat lbili stage, compete(etype == 1 2 4 5 6 7 8) vce(cluster id)
        Failure _d: etype==3
  Analysis time _t: (eventtime-origin)
            Origin: time entry
            Weight: [iweight=wt]
Iteration 0:  Log pseudolikelihood = -62.158461
Iteration 1:  Log pseudolikelihood = -61.671367
Iteration 2:  Log pseudolikelihood = -61.669225
Iteration 3:  Log pseudolikelihood = -61.669225
```

```
Competing-risks regression                     No. of obs      =         170
                                               No. of subjects =         170
Failure event:    etype == 3                   No. failed      =          13
Competing events: etype == 1 2 4 5 6 7 8       No. competing   =          59
                                               No. censored    =          98

                                               Wald chi2(3)    =        1.89
Log pseudolikelihood = -61.669225              Prob > chi2     =      0.5955
```

(Std. err. adjusted for 170 clusters in id)

_t	SHR	Robust std. err.	z	P>|z|	[95% conf. interval]
treat	.5785214	.3238038	-0.98	0.328	.1931497	1.732786
lbili	1.012415	.367095	0.03	0.973	.4974143	2.060623
stage	.5537101	.3305371	-0.99	0.322	.1718534	1.78405

In the above, we clustered on id but we did not stset it as an id() variable. That was because we wanted stcrreg to treat each observation within patient as its own distinct spell, not as a set of overlapping spells.

Treatment with UDCA seems to decrease the incidence of histologic progression as a first adverse event. However, the effect is not significant, most likely as a result of observing so few failures.

◁

stcrreg as an alternative to stcox

In this section, we demonstrate that you may also use `stcox` to perform a cumulative-incidence analysis, and we compare that approach with one that uses `stcrreg`.

▷ Example 4: HIV and SI as competing events

Geskus (2000) and Putter, Fiocco, and Geskus (2007) analyzed data from 324 homosexual men from the Amsterdam Cohort Studies on HIV infection and AIDS. During the course of infection, the syncytium inducing (SI) HIV phenotype appeared in many of these individuals. The appearance of the SI phenotype worsens prognosis. Thus the time to SI appearance in the absence of an AIDS diagnosis is of interest. In this context, a diagnosis of AIDS acts as a competing event.

```
. use https://www.stata-press.com/data/r18/hiv_si
(HIV and SI as competing risks)
. describe
Contains data from https://www.stata-press.com/data/r18/hiv_si.dta
 Observations:           324                  HIV and SI as competing risks
    Variables:             4                  3 Apr 2022 13:40
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
patnr	int	%8.0g		ID
time	float	%9.0g		Years from HIV infection
status	byte	%10.0g	stat	Status
ccr5	byte	%9.0g	ccr5	WM (deletion in C-C chemokine receptor 5 gene)

```
Sorted by:
```

In what follows, we re-create the analysis performed by Putter, Fiocco, and Geskus (2007), treating AIDS and SI as competing events and modeling cumulative incidence in relation to covariate `ccr5`. `ccr5` equals 1 if a specific deletion in the C-C chemokine receptor 5 gene is present and equals zero otherwise (wild type).

We can model the cumulative incidence of SI on `ccr5` directly with `stcrreg`:

```
. stset time, failure(status == 2)        // SI is the event of interest
  (output omitted)
. stcrreg ccr5, compete(status == 1)      // AIDS is the competing event
  (output omitted)
Competing-risks regression                      No. of obs      =         324
                                                No. of subjects =         324
Failure event:   status == 2                    No. failed      =         107
Competing event: status == 1                    No. competing   =         113
                                                No. censored    =         104
                                                Wald chi2(1)    =        0.01
Log pseudolikelihood = -579.06241               Prob > chi2     =      0.9172
```

_t	SHR	Robust std. err.	z	P>|z|	[95% conf. interval]
ccr5	1.023865	.2324119	0.10	0.917	.6561827 1.597574

It seems that this particular genetic mutation has little relation with the incidence of SI, a point we emphasize further with a graph:

```
. stcurve, cif at(ccr5=(0 1)) title(SI) range(0 13) yscale(range(0 0.5))
note: function evaluated at specified covariate values.
```
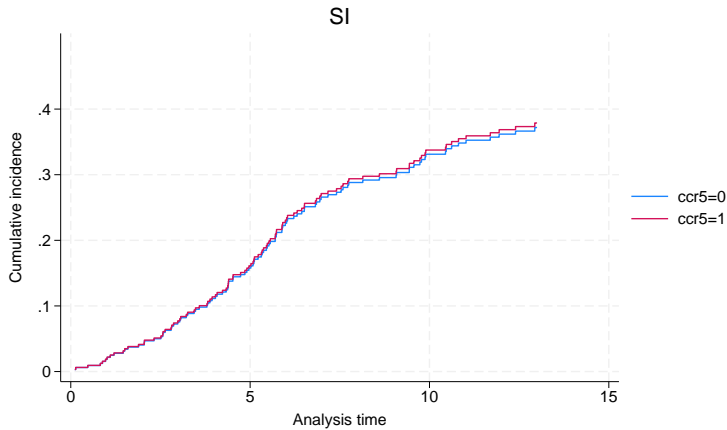

The above analysis compared SI incidence curves under the assumption that the subhazard for SI, that which generates SI events in the presence of AIDS, was proportional with respect to `ccr5`. Because we modeled the subhazard and not the cause-specific hazard, obtaining estimates of cumulative incidence was straightforward and depended only on the subhazard for SI and not on that for AIDS.

As explained in *The case for competing-risks regression*, the cumulative incidence of SI is a function of both the cause-specific hazard for SI, $h_1(t)$, and that for AIDS, $h_2(t)$, because SI and AIDS are competing events. Suppose for the moment that we are not interested in the incidence of SI in the presence of AIDS, but instead in the biological mechanism that causes SI in general. We can model this mechanism with `stcox` by treating AIDS events as censored.

```
. stcox ccr5

  (output omitted )

Cox regression with no ties

No. of subjects =       324                          Number of obs =      324
No. of failures =       107
Time at risk    = 2,261.96

                                                     LR chi2(1)    =     1.19
Log likelihood = -549.73443                          Prob > chi2   =   0.2748
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]
ccr5	.7755334	.1846031	-1.07	0.286	.4863914 1.23656

Because we initially `stset` our data with SI as the event of interest, AIDS events are treated as censored by `stcox` (but not by `stcrreg`). In any case, the `ccr5` mutation somewhat decreases the risk of SI, but this effect is not significant.

We make the above interpretation with no regard to AIDS as a competing risk because we are interested only in the biological mechanism behind SI. To estimate the cumulative incidence of SI, we first need to make a choice. Either we can pretend a diagnosis of AIDS does not exist as a competing risk and use `stcurve` to plot survivor curves for SI based on the Cox model above, or we can acknowledge AIDS as a competing risk and model that cause-specific hazard also.

We choose the latter. Before fitting the model, however, we need to re-`stset` the data with AIDS as the event of interest.

```
. stset time, failure(status == 1)           // AIDS is the event of interest
  (output omitted )

. stcox ccr5

  (output omitted )

Cox regression with Breslow method for ties

No. of subjects =      324                          Number of obs =      324
No. of failures =      113
Time at risk    = 2,261.96
                                                    LR chi2(1)    =   21.98
Log likelihood = -555.37301                         Prob > chi2   = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]
ccr5	.2906087	.0892503	-4.02	0.000	.1591812	.530549

Patients with the `ccr5` mutation have a significantly lower risk of AIDS.

We have now modeled both cause-specific hazards separately. Cleves (1999); Lunn and Mc-Neil (1995); and Putter, Fiocco, and Geskus (2007) (among others) describe an approach based on data duplication where both hazards can be modeled simultaneously. Such an approach has the advantage of being able to set the effects of `ccr5` on both hazards as equal and to test that hypothesis. Also, you can model the baseline hazards as proportional rather than entirely distinct. However, for the least parsimonious model with event-specific covariate effects and event-specific baseline hazards, the data duplication method is no different than fitting separate models for each event type, just as we have done above. Because data duplication will reveal no simpler model for these data, we do not describe it further.

We can derive estimates of cumulative incidence for SI based on the above cause-specific hazard models, but the process is a bit more complicated than before. The cumulative incidence of SI (event type 1) in the presence of AIDS (event type 2) is calculated as

$$\widehat{\text{CIF}}_1(t) = \sum_{j:t_j \leq t} \widehat{h}_1(t_j)\widehat{S}(t_{j-1})$$

with

$$\widehat{S}(t) = \prod_{j:t_j \leq t} \left\{1 - \widehat{h}_1(t_j) - \widehat{h}_2(t_j)\right\}$$

The t_j index the times at which events (of any type) occur, and $\widehat{h}_1(t_j)$ and $\widehat{h}_2(t_j)$ are the cause-specific *hazard contributions* for SI and AIDS respectively. Baseline hazard contributions can be obtained with `predict` after `stcox`, and they can be transformed to hazard contributions for any covariate pattern by multiplying them by the exponentiated linear predictor for that pattern. Hazard contributions represent the increments of the cumulative hazards at each event time. $\widehat{S}(t)$ estimates the probability that you are event free at time t.

We begin by refitting both models and predicting the hazard contributions.

```
. stset time, failure(status == 2)           // SI
  (output omitted )
. stcox ccr5
  (output omitted )
. predict h_si_0, basehc
(217 missing values generated)
. generate h_si_1 = h_si_0*exp(_b[ccr5])
(217 missing values generated)
. stset time, failure(status == 1)           // AIDS
  (output omitted )
. stcox ccr5
  (output omitted )
. predict h_aids_0, basehc
(211 missing values generated)
. gsort _t -_d
. by _t: replace h_aids_0 = . if _n > 1
(1 real change made, 1 to missing)
. generate h_aids_1 = h_aids_0*exp(_b[ccr5])
(212 missing values generated)
```

Variables h_si_0 and h_aids_0 hold the baseline hazard contributions, those for ccr5 == 0. Variables h_si_1 and h_aids_1 hold the hazard contributions for ccr5 == 1, and they were obtained by multiplying the baseline contributions by the exponentiated coefficient for ccr5. When we ran stcox with AIDS as the event of interest, the output indicated that we had tied failure times (the analysis for SI had no ties). As such, this required the extra step of setting any duplicated hazard contributions to missing. As it turned out, this affected only one observation.

Hazard contributions are generated only at times when events are observed and are set to missing otherwise. Because we will be summing and multiplying over event times, we next drop the observations that contribute nothing and then replace missing with zero for those observations that have some hazard contributions missing and some nonmissing.

```
. drop if missing(h_si_0) & missing(h_aids_0)
(105 observations deleted)
. replace h_aids_0 = 0 if missing(h_aids_0)
(107 real changes made)
. replace h_aids_1 = 0 if missing(h_aids_1)
(107 real changes made)
. replace h_si_0 = 0 if missing(h_si_0)
(112 real changes made)
. replace h_si_1 = 0 if missing(h_si_1)
(112 real changes made)
```

We can now sort by analysis time and calculate the estimated event-free survivor functions. Recall that you can express a product as an exponentiated sum of logarithms, which allows us to take advantage of Stata's sum() function for obtaining running sums.

```
. sort _t
. generate S_0 = exp(sum(log(1- h_aids_0 - h_si_0)))
. generate S_1 = exp(sum(log(1- h_aids_1 - h_si_1)))
```

Finally, we calculate the estimated CIFs and graph:

```
. generate cif_si_0 = sum(S_0[_n-1]*h_si_0)
. label var cif_si_0 "ccr5=0"
```

```
. generate cif_si_1 = sum(S_1[_n-1]*h_si_1)

. label var cif_si_1 "ccr5=1"

. twoway line cif_si* _t if _t<13, connect(J J) sort yscale(range(0 0.5))
> title(SI) ytitle(Cumulative incidence) xtitle(Analysis time)
```
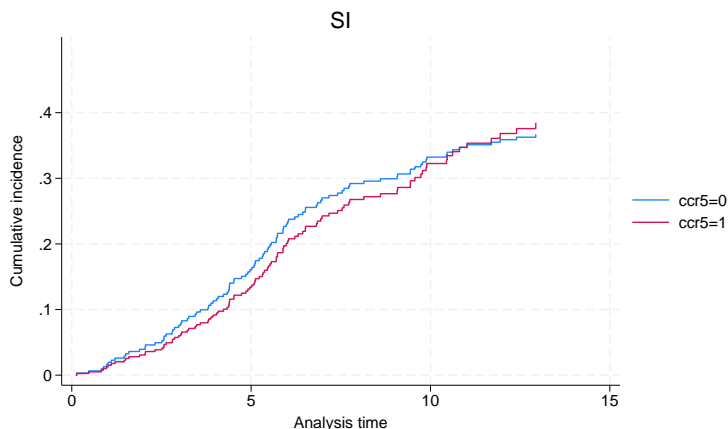

This model formulation shows `ccr5` to have more of an effect on the incidence of SI, although the effect is still small. Note that under this formulation, the effect of `ccr5` is not constrained to be overall increasing or overall decreasing. In fact, when $t > 11$ years or so, those with the `ccr5` mutation actually have an increased SI incidence. That is due to time-accumulated reduced competition from AIDS, the risk of which is significantly lower when the `ccr5` mutation is present.

Putter, Fiocco, and Geskus (2007) also performed the same analysis using AIDS as the event of interest, something we leave to you as an exercise.

◁

We have described two different modeling approaches for estimating the cumulative incidence of SI. Although you may prefer the `stcrreg` approach because it is much simpler, that does not mean it is a better model than the one based on `stcox`. The better model is the one whose assumptions more closely fit the data. The `stcrreg` model assumes that the effect of `ccr5` is proportional on the subhazard for SI. The `stcox` model assumes proportionality on the cause-specific hazards for both SI and AIDS. Because our analysis uses only one binary covariate, we can compare both models with a nonparametric estimator of the CIF to see which fits the data more closely; see [ST] **stcrreg postestimation**.

Multiple records per subject

`stcrreg` can be used with data where you have multiple records per subject, as long as 1) you `stset` an ID variable that identifies the subjects and 2) you carefully consider the role played by time-varying covariates in subjects who fail because of competing events. We explain both issues below.

Stata's `st` suite of commands allows for multiple records per subject. Having multiple records allows you to record gaps in subjects' histories and to keep track of time-varying covariates. If you have multiple records per subject, you identify which records belong to which subjects by specifying an ID variable to `stset` option `id()`.

Consider the sample data listed below:

```
. list if id == 18
```

	id	_t0	_t	_d	x
1.	18	3	5	0	5.1
2.	18	5	8	0	7.8
3.	18	11	12	0	6.7
4.	18	12	20	1	8.9

These data reflect the following:

- Subject 18 first became at risk at analysis time 3 (delayed entry) with covariate value x equal to 5.1.

- At time 5, subject 18's x value changed to 7.8.

- Subject 18 left the study at time 8 only to return at time 11 (gap), with x equal to 6.7 at that time.

- At time 12, x changed to 8.9.

- Subject 18 failed at time 20 with x equal to 8.9 at that time.

An analysis of these data with Cox regression using stcox is capable of processing all of this information. Intermittent records are treated as censored (_d==0), and either failure or censoring occurs on the last record (here failure with _d==1). When subjects are not under observation, they are simply not considered at risk of failure. Time-varying covariates are also processed correctly. For example, if some other subject failed at time 7, then the risk calculations would count subject 18 at risk with x equal to 7.8 at that time.

stcox will give the same results for the above data whether or not you stset the ID variable, id. Whether you treat the above data as four distinct subjects (three censored and one failed) or as one subject with a four-record history is immaterial. The only difference you may encounter concerns robust and replication-based standard errors, in which case if you stset an ID variable, then stcox will automatically cluster on this variable.

Such a distinction, however, is of vital importance to stcrreg. While stcox is concerned only about detecting one type of failure, stcrreg relies on precise accounting of the number of subjects who fail because of the event of interest, those who fail because of competing events, and those who are censored. In particular, the weighting mechanism behind stcrreg depends on an accurate estimate of the probability a subject will be censored; see *Methods and formulas*. As such, it makes a difference whether you want to treat the above as four distinct subjects or as one subject. If you have multiple records per subject, you must stset your ID variable before using stcrreg. When counting the number failed, number competing, and number censored, stcrreg only considers what happened at the end of a subject's history. Intermittent records are treated simply as temporary entries to and exits from the analysis, and the exits are not counted as censored in the strict sense.

Furthermore, when using stcrreg with covariates that change over multiple records (time-varying covariates), you need to carefully consider what happens when subjects experience competing failures. For the above sample data, subject 18 failed because of the event of interest (_d==1). Consider, however, what would have happened had this subject failed because of a competing event instead. Competing-risks regression keeps such subjects "at risk" of failure from the event of interest even after they fail from competing events; see *Methods and formulas*. Because these subjects will be used in future risk calculations for which they have no data, stcrreg will use the last available covariate values for these calculations. For the above example, if subject 18 experiences a competing event at time 20, then the last available value of x, 8.9, will be used in all subsequent risk calculations. If

the last available values are as good a guess as any as to what future values would have been—for example, a binary covariate recording pretransplant versus posttransplant status—then this is not an issue. If, however, you have reason to believe that a subject's covariates would have been much different had the subject remained under observation, then the results from `stcrreg` could be biased.

▷ Example 5: Hospital-acquired pneumonia

Consider the following simulated data from a competing-risks analysis studying the effects of pneumonia.

```
. use https://www.stata-press.com/data/r18/pneumonia, clear
(Hospital-acquired pneumonia)

. describe

Contains data from https://www.stata-press.com/data/r18/pneumonia.dta
 Observations:            957                   Hospital-acquired pneumonia
    Variables:              7                   7 Apr 2022 15:35
```

Variable name	Storage type	Display format	Value label	Variable label
id	int	%9.0g		Patient ID
age	byte	%9.0g		Age at admission
ndays	int	%9.0g		Days in ICU
died	byte	%9.0g		1 if died
censored	byte	%9.0g		1 if alive and in ICU at the end of the study
discharged	byte	%9.0g		1 if discharged
pneumonia	byte	%9.0g		1 if pneumonia

```
Sorted by: id
```

The above data are for 855 ICU patients. One hundred twenty-three patients contracted pneumonia, of which 21 did before admission and 102 during their stay. Those patients who contracted pneumonia during their stay are represented by two records with the time-varying covariate `pneumonia` recording the change in status.

We perform a competing-risks regression for the cumulative incidence of death during ICU stay with `age` and `pneumonia` as covariates. We also treat hospital discharge as a competing event.

```
. stset ndays, id(id) failure(died)
  (output omitted)

. stcrreg age pneumonia, compete(discharged) noshow nolog

Competing-risks regression                        No. of obs       =        957
                                                  No. of subjects  =        855
Failure events:   died nonzero, nonmissing        No. failed       =        178
Competing events: discharged nonzero, nonmissing  No. competing    =        641
                                                  No. censored     =         36

                                                  Wald chi2(2)     =     121.21
Log pseudolikelihood = -1128.6096                 Prob > chi2      =     0.0000

                          (Std. err. adjusted for 855 clusters in id)
```

_t	SHR	Robust std. err.	z	P>|z|	[95% conf. interval]	
age	1.021612	.0076443	2.86	0.004	1.006739	1.036705
pneumonia	5.587052	.9641271	9.97	0.000	3.983782	7.835558

Both increased age and contracting pneumonia are associated with an increased incidence of death in the ICU.

◁

Option tvc() and testing the proportional-subhazards assumption

In the previous section, we considered data with multiple records per subject. Such data make it possible to record discretely time-varying covariates, those whose values change at discrete points in time. Each change is captured by a new record.

Consider instead what happens when you have covariates that vary continuously with respect to time. Competing-risks regression assumes the following relationship between subhazard and baseline subhazard

$$\overline{h}_1(t) = \overline{h}_{1,0}(t) \exp(\beta_1 x_1 + \cdots + \beta_k x_k)$$

where $\overline{h}_{1,0}(t)$ is the baseline subhazard function. For most purposes, this model is sufficient, but sometimes we may wish to introduce variables of the form $z_i(t) = z_i g(t)$, which vary continuously with time so that

$$\overline{h}_1(t) = \overline{h}_{1,0}(t) \exp \left\{ \beta_1 x_1 + \cdots + \beta_k x_k + g(t)(\gamma_1 z_1 + \cdots + \gamma_m z_m) \right\} \tag{1}$$

where $(z_1, \ldots, z_m)$ are the baseline (constant) covariates. Fitting this model has the net effect of estimating the regression coefficient, γ_i, for the covariate $g(t)z_i$, which is a function of analysis time.

The covariates $(z_1, \ldots, z_m)$ are specified using the tvc(*tvarlist*) option, and $g(t)$ is specified using the texp(*exp*) option, where t in $g(t)$ is analysis time. For example, if we want $g(t) = \log(t)$, we would use texp(log(_t)) because _t stores the analysis time once the data are stset.

When subjects fail because of competing events, covariate values for these subjects continue to be used in subsequent risk calculations; see the previous section for details. When this occurs, any covariates specified using tvc() will continue to respect their time interactions even after these subjects fail. Because such behavior is unlikely to reflect any real data situation, we do not recommend using tvc() for this purpose.

We do, however, recommend using tvc() to model *time-varying coefficients*, because these can be used to test the proportionality assumption behind competing-risks regression. Consider a version of (1) that contains only one fixed covariate, x_1, and sets $z_1 = x_1$:

$$\overline{h}_1(t) = \overline{h}_{1,0}(t) \exp \left[\left\{ \beta_1 + \gamma_1 g(t) \right\} x_1 \right]$$

Given this new arrangement, we consider that $\beta_1 + \gamma_1 g(t)$ is a (possibly) time-varying coefficient on the covariate x_1, for some specified function of time $g(t)$. The coefficient has a time-invariant component β_1, with γ_1 determining the magnitude of the time-dependent deviations from β_1. As such, a test of $\gamma_1 = 0$ is a test of time invariance for the coefficient on x_1.

Confirming that a coefficient is time invariant is one way of testing the proportional-subhazards assumption. Proportional subhazards implies that the relative subhazard (that is, β) is fixed over time, and this assumption would be violated if a time interaction proved significant.

▷ Example 6: Testing proportionality of subhazards

Returning to our cervical cancer study (example 1), we now include time interactions on all three covariates as a way of testing the proportional-subhazards assumption for each:

```
. use https://www.stata-press.com/data/r18/hypoxia
(Hypoxia study)
. stset dftime, failure(failtype == 1)
 (output omitted )
. stcrreg ifp tumsize pelnode, compete(failtype == 2) tvc(ifp tumsize pelnode)
> noshr
 (output omitted )
```

Competing-risks regression				No. of obs	=	109
				No. of subjects =		109
Failure event: failtype == 1				No. failed	=	33
Competing event: failtype == 2				No. competing	=	17
				No. censored	=	59
				Wald chi2(6)	=	44.93
Log pseudolikelihood = -136.79				Prob > chi2	=	0.0000

_t	Coefficient	Robust std. err.	z	P>|z|	[95% conf. interval]	
main						
ifp	.0262093	.0174458	1.50	0.133	-.0079838	.0604025
tumsize	.37897	.1096628	3.46	0.001	.1640348	.5939052
pelnode	-.766362	.473674	-1.62	0.106	-1.694746	.162022
tvc						
ifp	.0055901	.0081809	0.68	0.494	-.0104441	.0216243
tumsize	-.1415204	.0908955	-1.56	0.119	-.3196722	.0366314
pelnode	.0610457	.5676173	0.11	0.914	-1.051464	1.173555

Note: Variables in tvc equation interacted with _t.

We used the default function of time $g(t) = t$, although we could have specified otherwise with the texp() option. After looking at the significance levels in the equation labeled "tvc", we find no indication that the proportionality assumption has been violated.

◁

When you use tvc() in this manner, there is no issue of postfailure covariate values for subjects who fail from competing events. The covariate values are assumed constant—the *coefficients* change with time.

Stored results

stcrreg stores the following in e():

Scalars
e(N)	number of observations
e(N_sub)	number of subjects
e(N_fail)	number of failures
e(N_compete)	number of competing events
e(N_censor)	number of censored subjects
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_dv)	number of dependent variables
e(df_m)	model degrees of freedom
e(ll)	log pseudolikelihood
e(N_clust)	number of clusters
e(chi2)	χ^2
e(p)	p-value for model test
e(rank)	rank of e(V)
e(fmult)	1 if $>$ 1 failure events, 0 otherwise
e(crmult)	1 if $>$ 1 competing events, 0 otherwise
e(fnz)	1 if nonzero indicates failure, 0 otherwise
e(crnz)	1 if nonzero indicates competing, 0 otherwise
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros
e(cmd)	stcrreg
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(mainvars)	variables in main equation
e(tvc)	covariates interacted with time from option tvc()
e(texp)	function of time used for covariates from option tvc()
e(fevent)	failure event(s) in estimation output
e(crevent)	competing event(s) in estimation output
e(compete)	competing event(s) as typed
e(wtype)	weight type
e(wexp)	weight expression
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(offset1)	offset
e(chi2type)	Wald; type of model χ^2 test
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(properties)	b V
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices
    e(b)   coefficient vector
    e(Cns)   constraints matrix
    e(ilog)   iteration log
    e(gradient)   gradient vector
    e(V)   variance–covariance matrix of the estimators
    e(V_modelbased) model-based variance

Functions
    e(sample)   marks estimation sample

In addition to the above, the following is stored in `r()`:

Matrices
    r(table)   matrix containing the coefficients with their standard errors, test statistics, p-values,
                             and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

In what follows, we assume single-record data and time-invariant covariates or coefficients. Extensions to both multiple-record data and time-varying covariates that are functions of time are achieved by treating the mechanisms that generate censorings, competing events, and failure events of interest as counting processes; see Fine and Gray (1999) and Andersen et al. (1993) for further details.

Let $\mathbf{x}_i$ be the row vector of m covariates for the time interval $(t_{0i}, t_i]$ for the ith observation in the dataset ($i = 1, \dots, n$). stcrreg obtains parameter estimates $\widehat{\boldsymbol{\beta}}$ by maximizing the log-pseudolikelihood function

$$\log L = \sum_{i=1}^{n} \delta_i w_i \left[\mathbf{x}_i \boldsymbol{\beta} + \text{offset}_i - \log \left\{ \sum_{j \in R_i} w_j \pi_{ji} \exp(\mathbf{x}_j \boldsymbol{\beta} + \text{offset}_j) \right\} \right]$$

where δ_i indicates a failure of interest for observation i and R_i is the set of observations, j, that are at risk at time t_i (that is, all j such that $t_{0j} < t_i \le t_j$). w_i and offset$_i$ are the usual observation weights and linear offsets, if specified.

The log likelihood given above is identical to that for standard Cox regression (Breslow method for ties) with the exception of the weights π_{ji}. These weights are used to keep subjects who have failed because of competing events in subsequent risk sets and to decrease their weight over time as their likelihood of being otherwise censored increases.

Formally, extend R_i above not only to include those at risk of failure at time t_i, but also to include those subjects already having experienced a competing-risks event. Also, define

$$\pi_{ji} = \frac{\widehat{S}_c(t_i)}{\widehat{S}_c\{\min(t_j, t_i)\}}$$

if subject j experiences a competing event; $\pi_{ji} = 1$ otherwise. $\widehat{S}_c(t)$ is the Kaplan–Meier estimate of the survivor function for the censoring distribution—that which treats censorings as the events of interest—evaluated at time t, and t_j is the time at which subject j experienced his or her competing-failure event. As a matter of convention, $\widehat{S}_c(t)$ is treated as the probability of being censored up to *but not including* time t.

Because of the sample weighting inherent to this estimator, the standard Hessian-based estimate of variance is not statistically appropriate and is thus rejected in favor of a robust, sandwich-type estimator, as derived by Fine and Gray (1999).

Define $z_i = \mathbf{x}_i \widehat{\boldsymbol{\beta}} + \text{offset}_i$. (Pseudo)likelihood scores are given by

$$\widehat{\mathbf{u}}_i = \widehat{\boldsymbol{\eta}}_i + \widehat{\boldsymbol{\psi}}_i$$

where $\widehat{\boldsymbol{\eta}}_i = (\widehat{\eta}_{1i}, \dots, \widehat{\eta}_{mi})'$, and

$$\widehat{\eta}_{ki} = \delta_i \left(x_{ki} - a_{ki} \right) - \exp(z_i) \sum_{j:t_{0i}<t_j \leq t_i} \frac{\delta_j w_j \pi_{ij}(x_{ki} - a_{kj})}{\sum_{\ell \in R_j} w_\ell \pi_{\ell j} \exp(z_\ell)}$$

for

$$a_{ki} = \frac{\sum_{\ell \in R_i} w_\ell \pi_{\ell i} x_{k\ell} \exp(z_\ell)}{\sum_{\ell \in R_i} w_\ell \pi_{\ell i} \exp(z_\ell)}$$

The $\widehat{\boldsymbol{\psi}}_i$ are variance contributions due to data estimation of the weights π_{ji}, with

$$\widehat{\boldsymbol{\psi}}_i = \frac{\gamma_i \widehat{\mathbf{q}}(t_i)}{r(t_i)} - \sum_{j:t_{0i}<t_j \leq t_i} \frac{\gamma_j \widehat{h}_c(t_j) \widehat{\mathbf{q}}(t_j)}{r(t_j)}$$

γ_i indicates censoring for observation i, $r(t)$ is the number at risk of failure (or censoring) at time t,

$$\widehat{h}_c(t) = \frac{\sum_{i=1}^n \gamma_i I(t_i = t)}{r(t)}$$

and the kth component of $\widehat{\mathbf{q}}(t)$ is

$$\widehat{q}_k(t) = \sum_{i \in C(t)} w_i \exp(z_i) \sum_{j:t_{0i}<t_j \leq t_i} \frac{\delta_j w_j \pi_{ij}(x_{ki} - a_{kj})}{\sum_{\ell \in R_j} w_\ell \pi_{\ell j} \exp(z_\ell)} I(t_j \geq t)$$

where $C(t)$ is the set of observations that experienced a competing event prior to time t.

By default, stcrreg calculates the Huber/White/sandwich estimator of the variance and calculates its clustered version if either the vce(cluster *clustvar*) option is specified or an ID variable has been stset. See *Maximum likelihood estimators* and *Methods and formulas* in [P] _robust for details on how the pseudolikelihood scores defined above are used to calculate this variance estimator.

Acknowledgment

References

Andersen, P. K., Ø. Borgan, R. D. Gill, and N. Keiding. 1993. *Statistical Models Based on Counting Processes*. New York: Springer.

Beyersman, J., A. Latouche, A. Buchholz, and M. Schumacher. 2009. Simulating competing risks data in survival analysis. *Statistics in Medicine* 28: 956–971. https://doi.org/10.1002/sim.3516.

Beyersman, J., and M. Schumacher. 2008. Time-dependent covariates in the proportional subdistribution hazards model for competing risks. *Biostatistics* 9: 765–776. https://doi.org/10.1093/biostatistics/kxn009.

Bottai, M. 2022. Estimating the risk of events with stprisk. *Stata Journal* 22: 969–974.

Cleves, M. A. 1999. ssa13: Analysis of multiple failure-time data with Stata. *Stata Technical Bulletin* 49: 30–39. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 338–349. College Station, TX: Stata Press.

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata Press.

Coviello, V., and M. M. Boggess. 2004. Cumulative incidence estimation in the presence of competing risks. *Stata Journal* 4: 103–112.

Cox, D. R. 1972. Regression models and life-tables (with discussion). *Journal of the Royal Statistical Society, Series B* 34: 187–220. https://doi.org/10.1111/j.2517-6161.1972.tb00899.x.

Crowder, M. J. 2001. *Classical Competing Risks*. Boca Raton, FL: Chapman and Hall/CRC.

Fine, J. P., and R. J. Gray. 1999. A proportional hazards model for the subdistribution of a competing risk. *Journal of the American Statistical Association* 94: 496–509. https://doi.org/10.1080/01621459.1999.10474144.

Fyles, A., M. Milosevic, D. Hedley, M. Pintilie, W. Levin, L. Manchul, and R. P. Hill. 2002. Tumor hypoxia has independent predictor impact only in patients with node-negative cervix cancer. *Journal of Clinical Oncology* 20: 680–687. https://doi.org/10.1200/JCO.2002.20.3.680.

Gail, M. H. 1975. A review and critique of some models used in competing risk analysis. *Biometrics* 31: 209–222. https://doi.org/10.2307/2529721.

Geskus, R. B. 2000. On the inclusion of prevalent cases in HIV/AIDS natural history studies through a marker-based estimate of time since seroconversion. *Statistics in Medicine* 19: 1753–1769. https://doi.org/10.1002/1097-0258(20000715)19:13⟨1753::AID-SIM487⟩3.0.CO;2-F.

Gichangi, A., and W. Vach. 2005. The analysis of competing risks data: A guided tour. Preprint series, Department of Statistics, University of Southern Denmark.

Gooley, T. A., W. Leisenring, J. Crowley, and B. E. Storer. 1999. Estimation of failure probabilities in the presence of competing risks: New representations of old estimators. *Statistics in Medicine* 18: 695–706. https://doi.org/10.1002/(sici)1097-0258(19990330)18:6⟨695::aid-sim60⟩3.0.co;2-o.

Gray, R. J. 1988. A class of k-sample tests for comparing the cumulative incidence of a competing risk. *Annals of Statistics* 16: 1141–1154. https://doi.org/10.1214/aos/1176350951.

Hinchliffe, S. R., and P. C. Lambert. 2013. Extending the flexible parametric survival model for competing risks. *Stata Journal* 13: 344–355.

Hinchliffe, S. R., D. A. Scott, and P. C. Lambert. 2013. Flexible parametric illness-death models. *Stata Journal* 13: 759–775.

Kaplan, E. L., and P. Meier. 1958. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association* 53: 457–481. https://doi.org/10.2307/2281868.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd ed. New York: Springer.

Lambert, P. C. 2007. Modeling of the cure fraction in survival studies. *Stata Journal* 7: 351–375.

———. 2017. The estimation and modeling of cause-specific cumulative incidence functions using time-dependent weights. *Stata Journal* 17: 181–207.

Lin, D. Y., and L. J. Wei. 1989. The robust inference for the Cox proportional hazards model. *Journal of the American Statistical Association* 84: 1074–1078. https://doi.org/10.2307/2290085.

Lindor, K. D., E. R. Dickson, W. P. Baldus, R. A. Jorgensen, J. Ludwig, P. A. Murtaugh, J. M. Harrison, R. H. Wiesner, M. L. Anderson, S. M. Lange, G. LeSage, S. S. Rossi, and A. F. Hofman. 1994. Ursodeoxycholic acid in the treatment of primary biliary cirrhosis. *Gastroenterology* 106: 1284–1290. https://doi.org/10.1016/0016-5085(94)90021-3.

Lunn, M., and D. McNeil. 1995. Applying Cox regression to competing risks. *Biometrics* 51: 524–532. https://doi.org/10.2307/2532940.

Marubini, E., and M. G. Valsecchi. 1995. *Analysing Survival Data from Clinical Trials and Observational Studies.* New York: Wiley.

Milosevic, M., A. Fyles, D. Hedley, M. Pintilie, W. Levin, L. Manchul, and R. P. Hill. 2001. Interstitial fluid pressure predicts survival in patients with cervix cancer independent of clinical prognostic factors and tumor oxygen measurements. *Cancer Research* 61: 6400–6405.

Mozumder, S. I., M. J. Rutherford, and P. C. Lambert. 2017. A flexible parametric competing-risks model using a direct likelihood approach for the cause-specific cumulative incidence function. *Stata Journal* 17: 462–489.

Pepe, M. S., and M. Mori. 1993. Kaplan–Meier, marginal or conditional probability curves in summarizing competing risks failure time data? *Statistics in Medicine* 12: 737–751. https://doi.org/10.1002/sim.4780120803.

Pintilie, M. 2006. *Competing Risks: A Practical Perspective.* Chichester, UK: Wiley.

———. 2007. Analysing and interpreting competing risk data. *Statistics in Medicine* 26: 1360–1367. https://doi.org/10.1002/sim.2655.

Putter, H., M. Fiocco, and R. B. Geskus. 2007. Tutorial in biostatistics: Competing risks and multi-state models. *Statistics in Medicine* 26: 2389–2430. https://doi.org/10.1002/sim.2712.

Therneau, T. M., and P. M. Grambsch. 2000. *Modeling Survival Data: Extending the Cox Model.* New York: Springer.

Tsiatis, A. A. 1975. A nonidentifiability aspect of the problem of competing risks. *Proceedings of the National Academy of Sciences* 72: 20–22. https://doi.org/10.1073/pnas.72.1.20.

Also see

[ST] **stcrreg postestimation** — Postestimation tools for stcrreg

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[ST] **stcox** — Cox proportional hazards model

[ST] **stcox PH-assumption tests** — Tests of proportional-hazards assumption after stcox

[ST] **stcox postestimation** — Postestimation tools for stcox

[ST] **stintcox** — Cox proportional hazards model for interval-censored survival-time data

[ST] **stintreg** — Parametric models for interval-censored survival-time data

[ST] **streg** — Parametric survival models

[ST] **sts** — Generate, graph, list, and test the survivor and related functions

[ST] **stset** — Declare data to be survival-time data

[MI] **Estimation** — Estimation commands for use with mi estimate

[U] **20 Estimation and postestimation commands**

Title

> **stcrreg postestimation** — Postestimation tools for stcrreg

Postestimation commands

The following postestimation command is of special interest after `stcrreg`:

Command	Description
stcurve	plot the cumulative subhazard and cumulative incidence functions

The following standard postestimation commands are also available:

Command	Description
contrast	contrasts and ANOVA-style joint tests of estimates
estat ic	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC)
estat summarize	summary statistics for the estimation sample
estat vce	variance–covariance matrix of the estimators (VCE)
estimates	cataloging estimation results
etable	table of estimation results
hausman	Hausman's specification test
lincom	point estimates, standard errors, testing, and inference for linear combinations of coefficients
margins	marginal means, predictive margins, marginal effects, and average marginal effects
marginsplot	graph the results from margins (profile plots, interaction plots, etc.)
nlcom	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
predict	subhazard ratios, linear predictions and their SEs, etc.
predictnl	point estimates, standard errors, testing, and inference for generalized predictions
pwcompare	pairwise comparisons of estimates
test	Wald tests of simple and composite linear hypotheses
testnl	Wald tests of nonlinear hypotheses

predict

Description for predict

predict creates a new variable containing predictions such as subhazard ratios, linear predictions, standard errors, baseline cumulative incidence and subhazard functions, Kaplan–Meier survivor curves, pseudolikelihood scores, efficient score and Schoenfeld residuals, and DFBETA measures of influence.

Menu for predict

Statistics > Postestimation

Syntax for predict

> predict [*type*] *newvar* [*if*] [*in*] [, *sv_statistic* <u>nooff</u>set]

> predict [*type*] { *stub** | *newvarlist* } [*if*] [*in*], *mv_statistic* [<u>part</u>ial]

sv_statistic	Description
Main	
shr	predicted subhazard ratio, also known as the relative subhazard; the default
xb	linear prediction $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$
stdp	standard error of the linear prediction; $\text{SE}(\mathbf{x}_j\widehat{\boldsymbol{\beta}})$
* basecif	baseline cumulative incidence function (CIF)
* basecshazard	baseline cumulative subhazard function
* kmcensor	Kaplan–Meier survivor curve for the censoring distribution

mv_statistic	Description
Main	
* scores	pseudolikelihood scores
* esr	efficient score residuals
* dfbeta	DFBETA measures of influence
* schoenfeld	Schoenfeld residuals

Unstarred statistics are available both in and out of sample; type predict ... if e(sample) ... if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when if e(sample) is not specified.

nooffset is allowed only with unstarred statistics.

Options for predict

$\boxed{\text{Main}}$

shr, the default, calculates the relative subhazard (subhazard ratio), that is, the exponentiated linear prediction, $\exp(\mathbf{x}_j\widehat{\boldsymbol{\beta}})$.

xb calculates the linear prediction from the fitted model. That is, you fit the model by estimating a set of parameters, $\beta_1, \beta_2, \ldots, \beta_k$, and the linear prediction is $\widehat{\beta}_1 x_{1j} + \widehat{\beta}_2 x_{2j} + \cdots + \widehat{\beta}_k x_{kj}$, often written in matrix notation as $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$.

The x_{1j}, x_{2j}, ..., x_{kj} used in the calculation are obtained from the data currently in memory and need not correspond to the data on the independent variables used in estimating $\boldsymbol{\beta}$.

stdp calculates the standard error of the prediction, that is, the standard error of $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$.

basecif calculates the baseline CIF. This is the CIF of the subdistribution for the cause-specific failure process.

basecshazard calculates the baseline cumulative subhazard function. This is the cumulative hazard function of the subdistribution for the cause-specific failure process.

kmcensor calculates the Kaplan–Meier survivor function for the censoring distribution. These estimates are used to weight within risk pools observations that have experienced a competing event. As such, these values are not predictions or diagnostics in the strict sense, but are provided for those who wish to reproduce the pseudolikelihood calculations performed by stcrreg; see [ST] **stcrreg**.

nooffset is allowed only with shr, xb, and stdp, and is relevant only if you specified offset(*varname*) for stcrreg. It modifies the calculations made by predict so that they ignore the offset variable; the linear prediction is treated as $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$ rather than $\mathbf{x}_j\widehat{\boldsymbol{\beta}} + \text{offset}_j$.

scores calculates the pseudolikelihood scores for each regressor in the model. These scores are components of the robust estimate of variance. For multiple-record data, by default only one score per subject is calculated and it is placed on the last record for the subject.

Adding the partial option will produce partial scores, one for each record within subject; see partial below. Partial pseudolikelihood scores are the additive contributions to a subject's overall pseudolikelihood score. In single-record data, the partial pseudolikelihood scores are the pseudolikelihood scores.

One score variable is created for each regressor in the model; the first new variable corresponds to the first regressor, the second to the second, and so on.

esr calculates the efficient score residuals for each regressor in the model. Efficient score residuals are diagnostic measures equivalent to pseudolikelihood scores, with the exception that efficient score residuals treat the censoring distribution (that used for weighting) as known rather than estimated. For multiple-record data, by default only one score per subject is calculated and it is placed on the last record for the subject.

Adding the partial option will produce partial efficient score residuals, one for each record within subject; see partial below. Partial efficient score residuals are the additive contributions to a subject's overall efficient score residual. In single-record data, the partial efficient scores are the efficient scores.

One efficient score variable is created for each regressor in the model; the first new variable corresponds to the first regressor, the second to the second, and so on.

dfbeta calculates the DFBETA measures of influence for each regressor of in the model. The DFBETA value for a subject estimates the change in the regressor's coefficient due to deletion of that subject.

For multiple-record data, by default only one value per subject is calculated and it is placed on the last record for the subject.

Adding the `partial` option will produce partial DFBETAs, one for each record within subject; see [partial](#) below. Partial DFBETAs are interpreted as effects due to deletion of individual records rather than deletion of individual subjects. In single-record data, the partial DFBETAs are the DFBETAs.

One DFBETA variable is created for each regressor in the model; the first new variable corresponds to the first regressor, the second to the second, and so on.

schoenfeld calculates the Schoenfeld-like residuals. Schoenfeld-like residuals are diagnostic measures analogous to Schoenfeld residuals in Cox regression. They compare a failed observation's covariate values to the (weighted) average covariate values for all of those at risk at the time of failure. Schoenfeld-like residuals are calculated only for those observations that end in failure; missing values are produced otherwise.

One Schoenfeld residual variable is created for each regressor in the model; the first new variable corresponds to the first regressor, the second to the second, and so on.

Note: The easiest way to use the preceding four options is, for example,

```
. predict double stub*, scores
```

where *stub* is a short name of your choosing. Stata then creates variables *stub*1, *stub*2, etc. You may also specify each variable name explicitly, in which case there must be as many (and no more) variables specified as there are regressors in the model.

partial is relevant only for multiple-record data and is valid with scores, esr, and dfbeta. Specifying partial will produce "partial" versions of these statistics, where one value is calculated for each record instead of one for each subject. The subjects are determined by the [id()](#) option to stset.

Specify partial if you wish to perform diagnostics on individual records rather than on individual subjects. For example, a partial DFBETA would be interpreted as the effect on a coefficient due to deletion of one record, rather than the effect due to deletion of all records for a given subject.

margins

Description for margins

margins estimates margins of response for subhazard ratios and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

margins [*marginlist*] [, *options*]

margins [*marginlist*] , <u>predict</u>(*statistic* ...) [<u>predict</u>(*statistic* ...) ...] [*options*]

statistic	Description
shr	predicted subhazard ratio, also known as the relative subhazard; the default
xb	linear prediction $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$
stdp	not allowed with margins
basecif	not allowed with margins
<u>basecsh</u>azard	not allowed with margins
<u>km</u>censor	not allowed with margins
<u>sc</u>ores	not allowed with margins
esr	not allowed with margins
<u>df</u>beta	not allowed with margins
<u>schoenf</u>eld	not allowed with margins

Statistics not allowed with margins are functions of stochastic quantities other than e(b).

For the full syntax, see [R] **margins**.

Remarks and examples

Remarks are presented under the following headings:

> *Baseline functions*
> *Null models*
> *Measures of influence*

Baseline functions

▷ Example 1: Cervical cancer study

In example 1 of [ST] **stcrreg**, we fit a proportional subhazards model on data from a cervical cancer study.

```
. use https://www.stata-press.com/data/r18/hypoxia
(Hypoxia study)
. stset dftime, failure(failtype == 1)
 (output omitted )
. stcrreg ifp tumsize pelnode, compete(failtype == 2)

 (output omitted )
```

Competing-risks regression				No. of obs	=	109
				No. of subjects	=	109
Failure event: failtype == 1				No. failed	=	33
Competing event: failtype == 2				No. competing	=	17
				No. censored	=	59
				Wald chi2(3)	=	33.21
Log pseudolikelihood = -138.5308				Prob > chi2	=	0.0000

_t	SHR	Robust std. err.	z	P>|z|	[95% conf. interval]	
ifp	1.033206	.0178938	1.89	0.059	.9987231	1.068879
tumsize	1.297332	.1271191	2.66	0.008	1.070646	1.572013
pelnode	.4588123	.1972067	-1.81	0.070	.1975931	1.065365

After fitting the model, we can predict the baseline cumulative subhazard, $\overline{H}_{1,0}(t)$, and the baseline CIF, $CIF_{1,0}(t)$:

```
. predict bch, basecsh
. predict bcif, basecif
. list dftime failtype ifp tumsize pelnode bch bcif in 1/15
```

	dftime	failtype	ifp	tumsize	pelnode	bch	bcif
1.	6.152	0	8	7	1	.0658792	.063756
2.	8.008	0	8.2	2	1	.0813224	.0781036
3.	.003	1	8.6	10	1	.0260186	.025683
4.	1.073	1	3.3	8	1	.0379107	.0372011
5.	.003	1	18.5	8	0	.0260186	.025683
6.	7.929	0	20	8	1	.0813224	.0781036
7.	8.454	0	21.8	4	1	.0813224	.0781036
8.	7.107	1	31.6	5	1	.0813224	.0781036
9.	8.378	0	16.5	5	1	.0813224	.0781036
10.	8.178	0	31.5	3	1	.0813224	.0781036
11.	3.395	0	18.5	4	1	.0658792	.063756
12.	.003	1	12.8	5	0	.0260186	.025683
13.	1.35	1	18.4	4	1	.051079	.0497964
14.	.003	1	18.5	8	1	.0260186	.025683
15.	.512	2	21	10	0	.0260186	.025683

The baseline functions are for subjects who have zero-valued covariates, which in this example are not representative of the data. If baseline is an extreme departure from the covariate patterns in your data, then we recommend recentering your covariates to avoid numerical overflows when predicting baseline functions; see *Making baseline reasonable* in [ST] **stcox postestimation** for more details.

For our data, baseline is close enough to not cause any numerical problems, but far enough to not be of scientific interest (zero tumor size?). You can transform the baseline functions to those for other covariate patterns according to the relationships

$$\overline{H}_1(t) = \exp(\mathbf{x}\boldsymbol{\beta})\overline{H}_{1,0}(t)$$

and

$$\mathrm{CIF}_1(t) = 1 - \exp\{-\exp(\mathbf{x}\boldsymbol{\beta})\overline{H}_{1,0}(t)\}$$

but it is rare that you will ever have to do that. `stcurve` will predict, transform, and graph these functions for you. When you use `stcurve`, you specify the covariate settings, and any you leave unspecified are set at the mean over the data used in the estimation.

```
. stcurve, cif at(ifp=(5 20) pelnode=0)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```
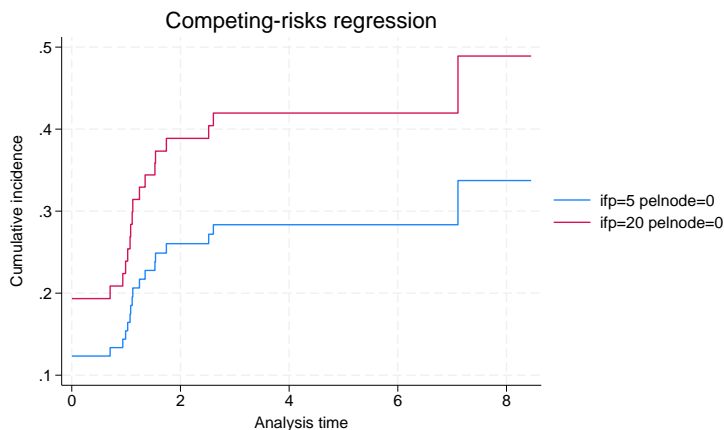

Because they were left unspecified, the cumulative incidence curves are for mean tumor size. If you wish to graph cumulative subhazards instead of CIFs, use the `stcurve` option `cumhaz` in place of `cif`.

◁

Null models

Predicting baseline functions after fitting a null model (one without covariates) yields nonparametric estimates of the cumulative subhazard and the CIF.

▷ Example 2: HIV and SI as competing events

In example 4 of [ST] **stcrreg**, we analyzed the incidence of appearance of the SI HIV phenotype, where a diagnosis of AIDS is a competing event. We modeled SI incidence in reference to a genetic mutation indicated by the covariate ccr5. We compared two approaches: one that used stcrreg and assumed that the subhazard of SI was proportional with respect to ccr5 versus one that used stcox and assumed that the cause-specific hazards for both SI and AIDS were each proportional with respect to ccr5. For both approaches, we produced cumulative incidence curves for SI comparing those who did not have the mutation (ccr5==0) to those who did (ccr5==1).

To see which approach better fits these data, we now produce cumulative incidence curves that make no model assumption about the effect of ccr5. We do this by fitting null models on the two subsets of the data defined by ccr5 and predicting the baseline CIF for each. Because the models have no covariates, the estimated baseline CIFs are nonparametric estimators.

```
. use https://www.stata-press.com/data/r18/hiv_si, clear
(HIV and SI as competing risks)
. stset time, failure(status == 2)            // SI is the event of interest
 (output omitted )
. stcrreg if !ccr5, compete(status == 1) noshow  // AIDS is the competing event
```

Competing-risks regression	No. of obs	=	259
	No. of subjects =		259
Failure event: status == 2	No. failed	=	84
Competing event: status == 1	No. competing	=	101
	No. censored	=	74
	Wald chi2(0)	=	0.00
Log pseudolikelihood = -435.80148	Prob > chi2	=	.

_t	SHR	Robust std. err.	z	P>|z|	[95% conf. interval]

```
. predict cif_si_0, basecif
(65 missing values generated)
. label var cif_si_0 "ccr5=0"
. stcrreg if ccr5, compete(status == 1) noshow
```

Competing-risks regression	No. of obs	=	65
	No. of subjects =		65
Failure event: status == 2	No. failed	=	23
Competing event: status == 1	No. competing	=	12
	No. censored	=	30
	Wald chi2(0)	=	0.00
Log pseudolikelihood = -88.306665	Prob > chi2	=	.

_t	SHR	Robust std. err.	z	P>|z|	[95% conf. interval]

```
. predict cif_si_1, basecif
(259 missing values generated)
. label var cif_si_1 "ccr5=1"
```

```
. twoway line cif_si* _t if _t<13, connect(J J) sort yscale(range(0 0.5))
> title(SI) ytitle(Cumulative incidence) xtitle(analysis time)
```
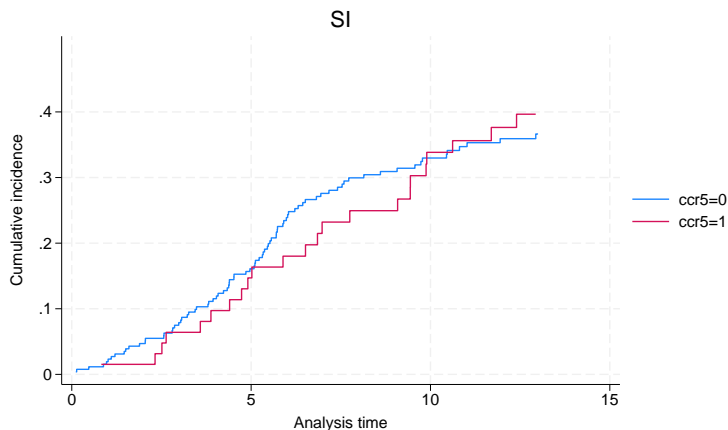

After comparing with the graphs produced in [ST] **stcrreg**, we find that the nonparametric analysis favors the stcox approach over the stcrreg approach.

◁

❑ Technical note

Predicting the baseline CIF after fitting a null model with stcrreg produces a nonparametric CIF estimator that is asymptotically equivalent, but not exactly equal, to an alternate estimator that is often used; see Coviello and Boggess (2004) for the details of that estimator. The estimator used by predict after stcrreg is a competing-risks extension of the Nelson–Aalen estimator (Nelson 1972; Aalen 1978); see *Methods and formulas*. The other is a competing-risks extension of the Kaplan–Meier (1958) estimator.

In large samples with many failures, the difference is negligible.

❑

Measures of influence

With predict after stcrreg, you can obtain pseudolikelihood scores that are used to calculate robust estimates of variance, Schoenfeld residuals that reflect each failure's contribution to the gradient of the log pseudolikelihood, efficient score residuals that represent each subject's (observation's) contribution to the gradient, and DFBETAs that measure the change in coefficients due to deletion of a subject or observation.

▷ Example 3: DFBETAs

Returning to our cervical cancer study, we obtain DFBETAs for each of the three coefficients in the model and graph those for the first with respect to analysis time.

```
. use https://www.stata-press.com/data/r18/hypoxia, clear
(Hypoxia study)
. stset dftime, failure(failtype == 1)
(output omitted)
. stcrreg ifp tumsize pelnode, compete(failtype == 2)
(output omitted)
. predict df*, dfbeta
. generate obs = _n
. twoway scatter df1 dftime, yline(0) mlabel(obs)
```
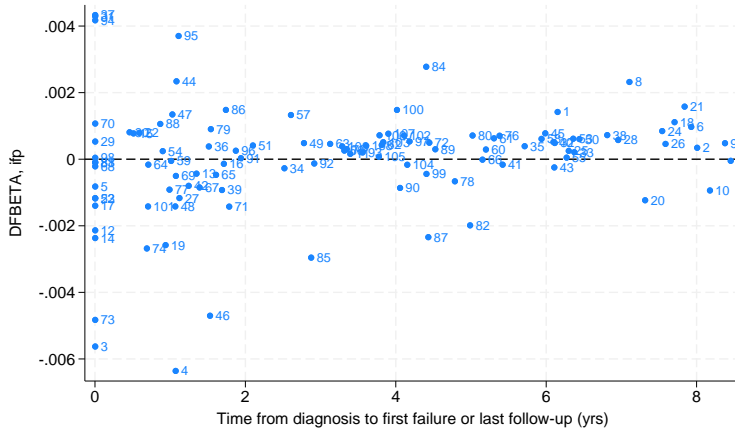

predict created the variables df1, df2, and df3, holding DFBETA values for variables ifp, tumsize, and pelnode, respectively. Based on the graph, we see that subject 4 is the most influential on the coefficient for ifp, the first covariate in the model.

◁

In the previous example, we had single-record data. If you have data with multiple records per subject, then by default DFBETAs will be calculated at the subject level, with one value representing each subject and measuring the effect of deleting all records for that subject. If you instead want record-level DFBETAs that measure the change due to deleting single records within subjects, add the partial option; see [ST] stcox postestimation for further details.

Methods and formulas

Continuing the discussion from *Methods and formulas* in [ST] **stcrreg**, the baseline cumulative subhazard function is calculated as

$$\widehat{H}_{1,0}(t) = \sum_{j:t_j \leq t} \frac{\delta_j}{\sum_{\ell \in R_j} w_\ell \pi_{\ell j} \exp(z_\ell)}$$

The baseline CIF is $\widehat{\text{CIF}}_{1,0}(t) = 1 - \exp\{-\widehat{H}_{1,0}(t)\}$.

The Kaplan–Meier survivor curve for the censoring distribution is

$$\widehat{S}_c(t) = \prod_{t_{(j)} < t} \left\{ 1 - \frac{\sum_i \gamma_i I(t_i = t_{(j)})}{r(t_{(j)})} \right\}$$

where $t_{(j)}$ indexes the times at which censorings occur.

Both the pseudolikelihood scores, $\widehat{\mathbf{u}}_i$, and the efficient score residuals, $\widehat{\boldsymbol{\eta}}_i$, are as defined previously. DFBETAs are calculated according to Collett (2015):

$$\text{DFBETA}_i = \widehat{\boldsymbol{\eta}}_i' \text{Var}^*(\widehat{\boldsymbol{\beta}})$$

where $\text{Var}^*(\widehat{\boldsymbol{\beta}})$ is the model-based variance estimator, that is, the inverse of the negative Hessian.

Schoenfeld residuals are $\mathbf{r}_i = (\widehat{r}_{1i}, \ldots, \widehat{r}_{mi})$ with

$$\widehat{r}_{ki} = \delta_i (x_{ki} - a_{ki})$$

References

Aalen, O. O. 1978. Nonparametric inference for a family of counting processes. *Annals of Statistics* 6: 701–726. https://doi.org/10.1214/aos/1176344247.

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata Press.

Collett, D. 2015. *Modelling Survival Data in Medical Research*. 3rd ed. Boca Raton, FL: Chapman and Hall/CRC.

Coviello, V., and M. M. Boggess. 2004. Cumulative incidence estimation in the presence of competing risks. *Stata Journal* 4: 103–112.

Fyles, A., M. Milosevic, D. Hedley, M. Pintilie, W. Levin, L. Manchul, and R. P. Hill. 2002. Tumor hypoxia has independent predictor impact only in patients with node-negative cervix cancer. *Journal of Clinical Oncology* 20: 680–687. https://doi.org/10.1200/JCO.2002.20.3.680.

Geskus, R. B. 2000. On the inclusion of prevalent cases in HIV/AIDS natural history studies through a marker-based estimate of time since seroconversion. *Statistics in Medicine* 19: 1753–1769. https://doi.org/10.1002/1097-0258(20000715)19:13⟨1753::AID-SIM487⟩3.0.CO;2-F.

Kaplan, E. L., and P. Meier. 1958. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association* 53: 457–481. https://doi.org/10.2307/2281868.

Milosevic, M., A. Fyles, D. Hedley, M. Pintilie, W. Levin, L. Manchul, and R. P. Hill. 2001. Interstitial fluid pressure predicts survival in patients with cervix cancer independent of clinical prognostic factors and tumor oxygen measurements. *Cancer Research* 61: 6400–6405.

Nelson, W. 1972. Theory and applications of hazard plotting for censored failure data. *Technometrics* 14: 945–966. https://doi.org/10.2307/1267144.

Pintilie, M. 2006. *Competing Risks: A Practical Perspective*. Chichester, UK: Wiley.

Putter, H., M. Fiocco, and R. B. Geskus. 2007. Tutorial in biostatistics: Competing risks and multi-state models. *Statistics in Medicine* 26: 2389–2430. https://doi.org/10.1002/sim.2712.

Also see

[ST] **stcrreg** — Competing-risks regression

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[U] **20 Estimation and postestimation commands**

Title

> **stcurve** — Plot the survivor or related function after streg, stcox, and more

Description

stcurve plots the survivor, failure, hazard, or cumulative hazard function after stcox, streg, stintreg, stintcox, mestreg, xtstreg, lasso cox, or elasticnet cox. stcurve also plots the cumulative subhazard or cumulative incidence function (CIF) after stcrreg.

Quick start

Plot the survivor function with covariates at their means after stcox, streg, stintreg, stintcox, mestreg, xtstreg, lasso cox, or elasticnet cox

    stcurve, survival

Same as above, but plot separate survivor functions for covariate x set to 1, 2, and 3

    stcurve, survival at1(x=1) at2(x=2) at3(x=3)

Same as above, but specify a numlist for x in at()

    stcurve, survival at(x=(1 2 3))

Same as above, but specify a different pattern for each line

    stcurve, survival at(x=(1 2 3)) lpattern(solid dash dot)

Same as above, and save the graph as mygraph.gph

    stcurve, survival at(x=(1 2 3)) saving(mygraph)

Plot the estimated hazard function after stcox, streg, stintreg, stintcox, mestreg, xtstreg, lasso cox, or elasticnet cox

    stcurve, hazard

Smooth the estimated hazard contributions using the Gaussian kernel function for the kernel density estimate after stcox or stintcox, and set x to 1

    stcurve, hazard kernel(gaussian) at(x=1)

Plot the cumulative hazard function after stcox, streg, stintreg, stintcox, mestreg, xtstreg, lasso cox, or elasticnet cox

    stcurve, cumhaz

Plot the cumulative subhazard function after stcrreg

    stcurve, cumhaz

Plot the cumulative incidence function after stcrreg

    stcurve, cif

Same as above, but set x to 0

    stcurve, cif at(x=0)

Menu

Statistics > Survival analysis > Regression models > Plot survivor or related function

Syntax

stcurve [, *options*]

options	Description
[Main]	
* <u>surv</u>ival	plot survivor function
* <u>f</u>ailure	plot failure function
* <u>haz</u>ard	plot hazard function
* <u>cumh</u>az	plot cumulative hazard function
* cif	plot cumulative incidence function
atomeans	evaluate function at overall means; the default
attmeans	evaluate function at time-specific means; available only after stintcox
at(*atspec*)	values of the specified covariates and means of
[at1(*atspec1*) [at2(*atspec2*) [...]]]	unspecified covariates
atframe(*framename*)	use covariate values from frame; available only after stintcox
[Options]	
alpha1	conditional frailty model
fixedonly	set all random effects to zero
unconditional	unconditional frailty model or random-effects model
marginal	synonym for unconditional
range(*# #*)	range of analysis time
outfile(*filename* [, replace])	save values used to plot the curves
width(*#*)	override "optimal" width; use with hazard
kernel(*kernel*)	kernel function; use with hazard
noboundary	no boundary correction; use with hazard
[Plot]	
connect_options	affect rendition of plotted survivor, failure, hazard, or cumulative hazard function
[Add plots]	
addplot(*plot*)	add other plots to the generated graph
[Y axis, X axis, Titles, Legend, Overall]	
twoway_options	any options other than by() documented in [G-3] *twoway_options*

---

*One of survival, failure, hazard, cumhaz, or cif must be specified.

survival, failure, and hazard are not allowed after estimation with stcrreg; see [ST] **stcrreg**

cif is allowed only after estimation with stcrreg; see [ST] **stcrreg**.

stcurve is not supported after stratified estimation.

For the stcurve syntax following lasso cox and elasticnet cox, see [LASSO] **lasso postestimation**.

Options

survival specifies that the survivor function be plotted. survival is not allowed after estimation
with stcrreg.

failure specifies that the failure function be plotted. failure is not allowed after estimation with
stcrreg.

hazard specifies that the hazard function be plotted. hazard is not allowed after estimation with
stcrreg.

cumhaz specifies that the cumulative hazard function be plotted when used after stcox, streg,
stintreg, stintcox, mestreg, or xtstreg and specifies that the cumulative subhazard function
be plotted when used after stcrreg.

cif specifies that the cumulative incidence function be plotted. This option is available only after
estimation with stcrreg.

atomeans specifies that the estimates of the survivor or other function be evaluated at the overall
means of covariates. This is the default.

attmeans is supported after stintcox in a multiple-record-per-subject format. It specifies that the
estimates of the survivor or other function be evaluated at the time-specific means of covariates.
This option is useful to incorporate time profiles for time-varying covariates present in the dataset.
Also see option atframe().

at(*atspec*) specifies that the estimates of the survivor or other function be evaluated at specific
covariate values. By default, stcurve evaluates the function by setting each covariate to its overall
mean value. This option causes the function to be evaluated at the values of the covariates listed in
at() and at the overall means of all unlisted covariates. If option attmeans is also specified, the
unlisted covariates are evaluated at time-specific means. This option can be repeated to produce
multiple curves, or you can specify multiple values for a set of covariates in one at() option;
see *Syntax of at()* in [ST] *adjustfor_option* for details. at() may not be combined with at1(),
at2(), and so on.

at1(*atspec1*), at2(*atspec2*), ..., at20(*atspec20*) are the alternatives to the repeated use of
at(). They specify that multiple curves (up to 20) be plotted on the same graph. at1(), at2(),
..., at20() work similarly to the at() option. at1() specifies the values of the covariates for
the first curve, at2() specifies the values of the covariates for the second curve, and so on. But,
unlike at(), at#() cannot be repeated and may not be combined with at(). *atspec1*, *atspec2*,
and so on follow the same syntax as *atspec*, except they do not allow *numlist*s or multiple values
for the same covariate.

atframe(*framename*) is supported after stintcox in a multiple-record-per-subject format. It specifies
that the estimates of the survivor or other function be evaluated using the values of variables
specified in the *framename* frame. The frame must contain a time variable with the same name
as the examination time variable specified in stintcox's time() option. It must also include at
least one covariate as specified with stintcox or in its tvc() option. atframe() may not be
combined with at().

alpha1, when used after fitting a frailty model, plots curves that are conditional on a frailty value
of one. This is the default for shared-frailty models.

fixedonly specifies that all random effects be set to zero, which is equivalent to using only the fixed portion of the model, when plotting results for random-effects models. This option is allowed only after xtstreg or mestreg; it is the default after xtstreg.

unconditional and marginal, when used after fitting a frailty model or a random-effects model, plot curves that are unconditional on the frailty or on the random effects. That is, the curve is "averaged" over the frailty distribution or over the random-effects distributions. This is the default for unshared-frailty models and for random-effects models. This option is not allowed after stintreg, stintcox, or xtstreg.

range(# #) specifies the range of the time axis to be plotted. If this option is not specified, stcurve plots the desired curve on an interval expanding from the earliest to the latest time in the data.

outfile(*filename* [, replace]) saves in *filename*.dta the values used to plot the curve(s).

width(#) is for use with hazard (and applies only after stcox) and is used to specify the bandwidth to be used in the kernel smooth used to plot the estimated hazard function. If left unspecified, a default bandwidth is used, as described in [R] **kdensity**.

kernel(*kernel*) is for use with hazard and is for use only after stcox or stintcox because, for Cox regression, an estimate of the hazard function is obtained by smoothing the estimated hazard contributions. kernel() specifies the kernel function for use in calculating the weighted kernel-density estimate required to produce a smoothed hazard-function estimator. The default is kernel(epanechnikov), yet *kernel* may be any of the kernels supported by kdensity; see [R] **kdensity**.

noboundary is for use with hazard and applies only to the plotting of smoothed hazard functions after stcox or stintcox. It specifies that no boundary-bias adjustments are to be made when calculating the smoothed hazard-function estimator. By default, the smoothed hazards are adjusted near the boundaries; see [ST] **sts graph**. If the epan2, biweight, or rectangular kernel is used after estimation using stcox, the bias correction near the boundary is performed using boundary kernels. For other kernels, the plotted range of the smoothed hazard function is restricted to be inside of one bandwidth from each endpoint. For these other kernels, specifying noboundary merely removes this range restriction. After estimation using stintcox, the boundary adjustments correspond to simply restricting the plotted range of the function for all kernels.

┌─ Plot ───

connect_options affect the rendition of the plotted survivor, failure, hazard, or cumulative hazard function; see [G-3] *connect_options*.

┌─ Add plots ──

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] *addplot_option*.

┌─ Y axis, X axis, Titles, Legend, Overall ──────────────────────────────────

twoway_options are any of the options documented in [G-3] *twoway_options*, excluding by(). These include options for titling the graph (see [G-3] *title_options*) and for saving the graph to disk (see [G-3] *saving_option*).

Remarks and examples

Remarks are presented under the following headings:

> *stcurve after stcox*
> *stcurve after streg*
> *stcurve after stcrreg*
> *stcurve after stintreg and stintcox*
> *Using at() with stcurve*

For examples of stcurve after xtstreg and mestreg, see [XT] **xtstreg postestimation** and [ME] **mestreg postestimation**, respectively.

stcurve after stcox

After fitting a Cox model, stcurve can be used to plot the estimated survivor, failure, hazard, or cumulative hazard function.

▷ Example 1

```
. use https://www.stata-press.com/data/r18/drugtr
(Patient survival in drug trial)
. stcox age drug
  (output omitted)
. stcurve, survival
note: function evaluated at overall means of covariates.
```


Cox proportional hazards regression
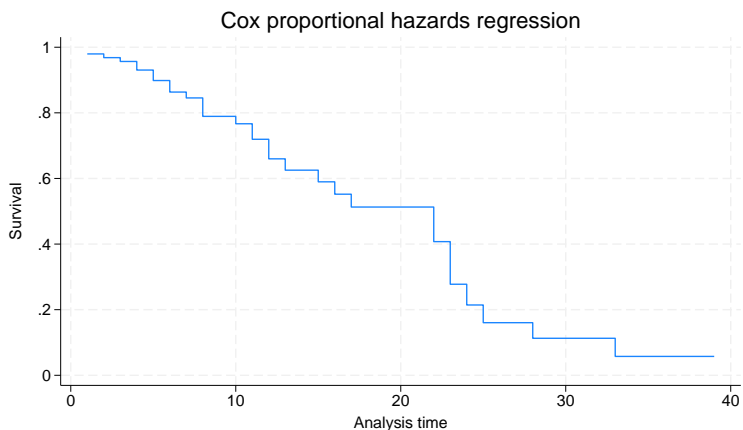
By default, the curve is evaluated at the mean values of all the predictors, but we can specify other values if we wish.
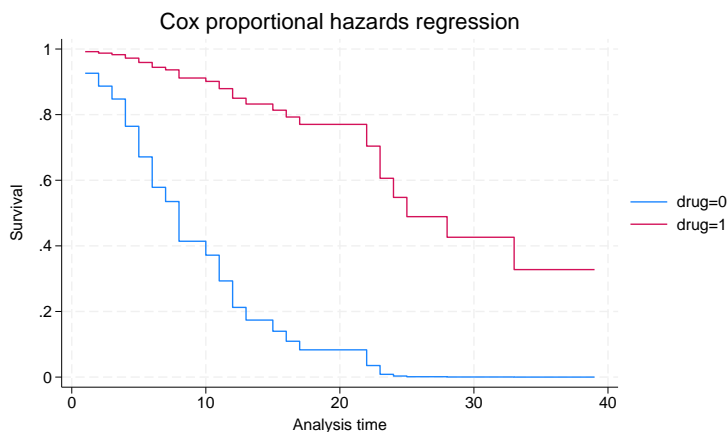
```
. stcurve, survival at1(drug=0) at2(drug=1)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```


Alternatively, you can obtain the same plot by typing the following:

```
. stcurve, survival at(drug=(0 1))
```

In this example, we asked for two plots, one for the placebo group and one for the treatment group. For both groups, the value of `age` was held at its mean value for the overall estimation sample.

See Cefalu (2011) for a Stata command to plot the survivor or cumulative hazard function with pointwise confidence intervals.

◁

▷ Example 2

stcurve can also be used to plot estimated hazard functions. The hazard function is estimated by a kernel smooth of the estimated hazard contributions; see [ST] **sts graph** for details. We can thus customize the smooth as we would any other; see [R] **kdensity** for details.

```
. stcurve, hazard at(drug=(0 1)) kernel(gauss) yscale(log)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```
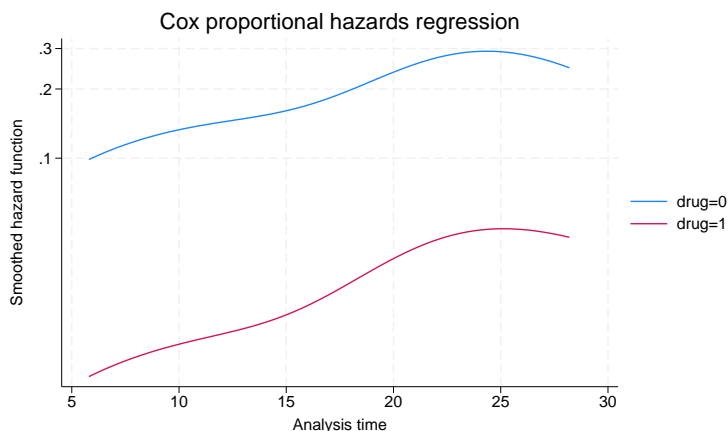

For the hazard plot, we plotted on a log scale to demonstrate the proportionality of hazards under this model; see the technical note below on smoothed hazards.

◁

❑ Technical note

For survivor or cumulative hazard estimation, stcurve works by first estimating the baseline function and then modifying it to adhere to the specified (or by default, mean) covariate patterns. As mentioned previously, *baseline* (when all covariates are equal to zero) must correspond to something that is meaningful and preferably in the range of your data. Otherwise, stcurve could encounter numerical difficulties. We ignored our own advice above and left age unchanged. Had we encountered numerical problems, or funny-looking graphs, we would have known to try shifting age so that age==0 was in the range of our data.

For hazard estimation, stcurve works by first transforming the estimated hazard contributions to adhere to the necessary covariate pattern and then applying the smooth. When you plot multiple curves, each is smoothed independently, although the same bandwidth is used for each.

The smoothing takes place in the hazard scale and not in the log hazard-scale. As a result, the resulting curves will look nearly, but not exactly, parallel when plotted on a log scale. This inexactitude is a product of the smoothing and should not be interpreted as a deviation from the proportional-hazards assumption; stcurve (after stcox) assumes proportionality of hazards and will reflect this in the produced plots. If smoothing were a perfect science, the curves would be parallel when plotted on a log scale. If you encounter estimated hazards exhibiting severe disproportionality, this may signal a numerical problem as described above. Try recentering your covariates so that baseline is more reasonable.

❑

stcurve after streg

stcurve is used after streg to plot the fitted survivor, failure, hazard, or cumulative hazard function. By default, stcurve computes the means of the covariates and evaluates the functions at each time in the data, censored or uncensored. The resulting plot is therefore the survival experience of a subject with a covariate pattern equal to the average covariate pattern in the study. You can produce the plot at other values of the covariates by using the at() option or specify a time range by using the range() option.

▷ Example 3

We pick up where example 6 of [ST] **streg** left off. The cancer dataset we are using has three values for variable drug: 1 corresponds to placebo, and 2 and 3 correspond to two alternative treatments. Using the cancer data with drug remapped to form an indicator of treatment, let's fit a loglogistic regression model and plot its survival curves. We can perform a loglogistic regression by issuing the following commands:

```
. use https://www.stata-press.com/data/r18/cancer
(Patient survival in drug trial)
. replace drug = drug==2 | drug==3              // 0, placebo : 1, nonplacebo
(48 real changes made)
. stset studytime, failure(died)
 (output omitted )
. streg age drug, distribution(llogistic) nolog

        Failure _d: died
  Analysis time _t: studytime

Loglogistic AFT regression

No. of subjects =  48                                    Number of obs =      48
No. of failures =  31
Time at risk    = 744
                                                        LR chi2(2)    =   35.14
Log likelihood = -43.21698                              Prob > chi2   =  0.0000
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
age	-.0803289	.0221598	-3.62	0.000	-.1237614	-.0368964
drug	1.420237	.2502148	5.68	0.000	.9298251	1.910649
_cons	6.446711	1.231914	5.23	0.000	4.032204	8.861218
/lngamma	-.8456552	.1479337	-5.72	0.000	-1.1356	-.5557105
gamma	.429276	.0635044			.3212293	.5736646

Now, we wish to plot the survivor and the hazard functions:

```
. stcurve, survival ylabels(0 .5 1)
note: function evaluated at overall means of covariates.
```
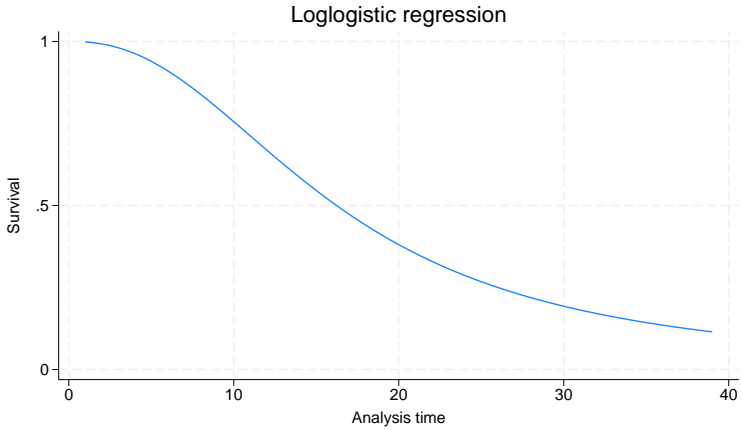

Figure 3. Loglogistic survival distribution at mean value of all covariates

```
. stcurve, hazard
note: function evaluated at overall means of covariates.
```
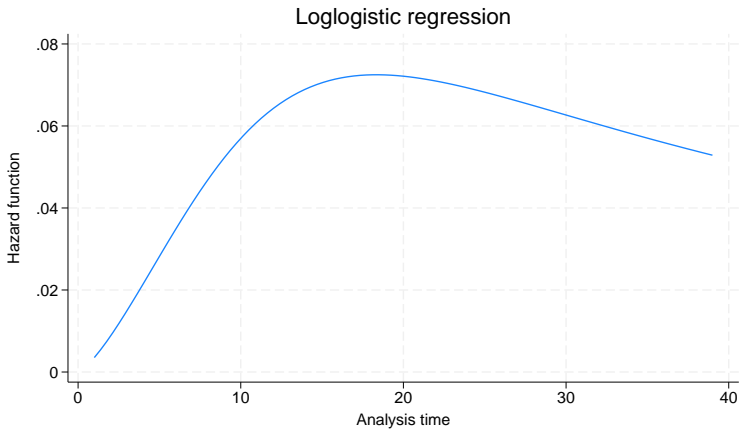

Figure 4. Loglogistic hazard distribution at mean value of all covariates

These plots show the fitted survivor and hazard functions evaluated for a cancer patient of average age receiving the average drug. Of course, the "average drug" has no meaning here because `drug` is an indicator variable. It makes more sense to plot the curves at a fixed value (level) of the drug. We can do this with the `at` option. For example, we may want to compare the average-age patient's survival curve under placebo (`drug==0`) and under treatment (`drug==1`).

We can plot both curves on the same graph:

```
. stcurve, survival at(drug=(0 1)) ylabels(0 .5 1)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```


Figure 5. Loglogistic survival distribution at mean age for placebo

In the plot, we can see from the loglogistic model that the survival experience of an average-age patient receiving the placebo is worse than the survival experience of that same patient receiving treatment. We can also see the accelerated-failure-time feature of the loglogistic model. The survivor function for treatment is a time-decelerated (stretched-out) version of the survivor function for placebo.

◁

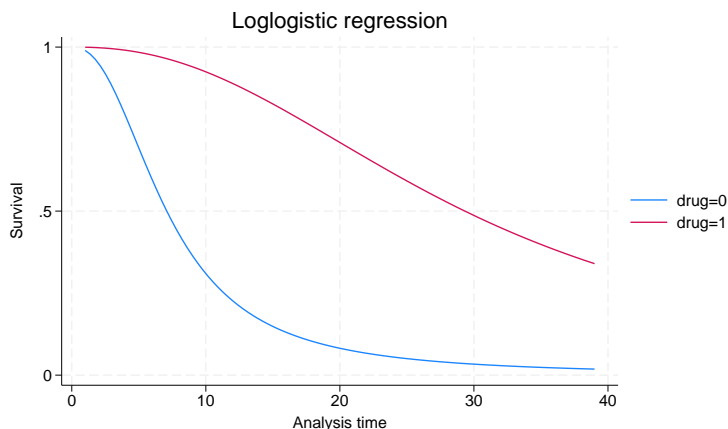▷ Example 4

In our discussion of frailty models in [ST] **streg**, we emphasize the distinction between the individual hazard (or survivor) function and the hazard (survivor) function for the population. When significant frailty is present, the population hazard will tend to begin falling past a certain point, regardless of the shape of the individual hazard. This is due to the frailty effect—as time passes, the frailer individuals will fail, leaving a more homogeneous population comprising only the most robust individuals.

The frailty effect may be demonstrated using `stcurve` to plot the estimated hazard (both individual and population) after fitting a frailty model. Use the `alpha1` option to specify the individual hazard ($\alpha = 1$) and the `unconditional` option to specify the population hazard. Applying this to the Weibull/inverse-Gaussian shared-frailty model on the kidney data of example 11 of [ST] **streg**,

```
. use https://www.stata-press.com/data/r18/catheter, clear
(Kidney data, McGilchrist and Aisbett, Biometrics, 1991)
. stset time infect
  (output omitted )
. quietly streg age female, distribution(weibull) frailty(invgauss) shared(patient)
```

```
. stcurve, hazard at(female=1) alpha1
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```
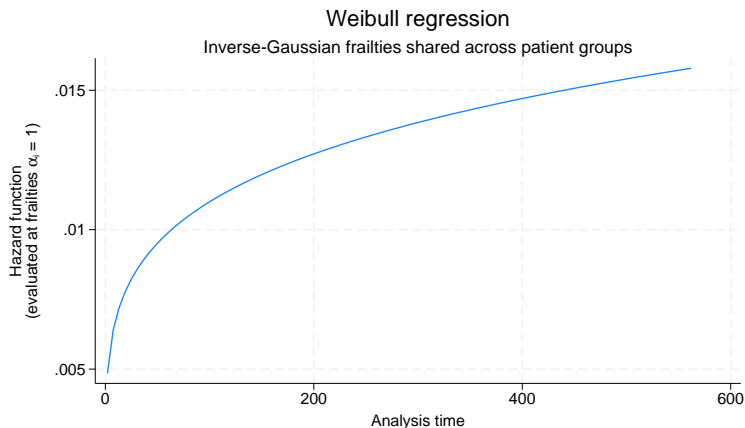

Figure 6. Individual hazard for females at mean age

Compare with

```
. stcurve, hazard at(female=1) unconditional
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```
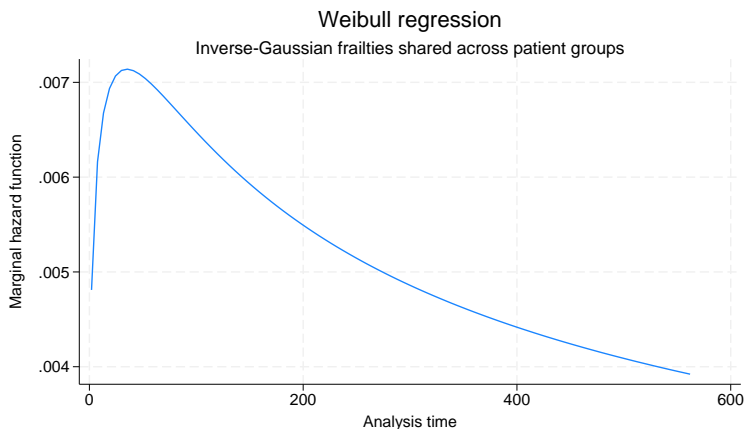

Figure 7. Population hazard for females at mean age

stcurve after stcrreg

▷ Example 5

In [ST] **stcrreg**, we analyzed data from 109 patients with primary cervical cancer, treated at a cancer center between 1994 and 2000. We fit a competing-risks regression model where local relapse was the failure event of interest (`failtype == 1`), distant relapse with no local relapse was the competing risk event (`failtype == 2`), and we were interested primarily in the effect of interstitial fluid pressure (`ifp`) while controlling for tumor size and pelvic node involvement.

After fitting the competing-risks regression model, we can use `stcurve` to plot the estimated cumulative incidence of local relapses in the presence of the competing risk. We wish to compare the cumulative incidence curves for `ifp == 5` versus `ifp == 20`, assuming positive pelvic node involvement (`pelnode == 0`) and a tumor size that is the average over the data.

```
. use https://www.stata-press.com/data/r18/hypoxia
(Hypoxia study)
. stset dftime, fail(failtype==1)
(output omitted)
. stcrreg ifp tumsize pelnode, compete(failtype==2)
(output omitted)
. stcurve, cif at(ifp=(5 20) pelnode=0)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```


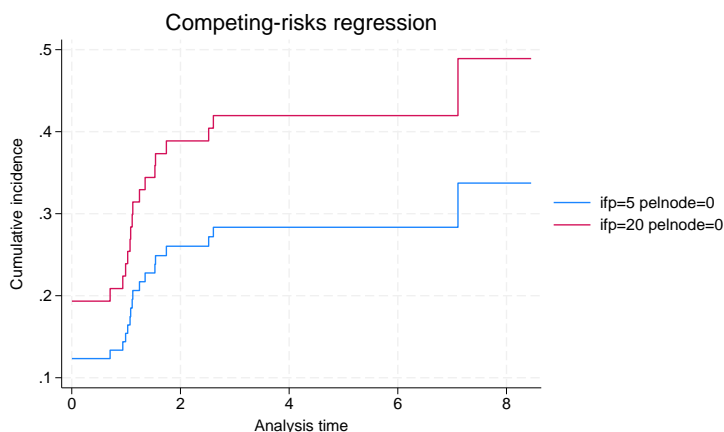Figure 8. Comparative cumulative incidence functions

Specifying `ifp=(5 20)` in the `at()` option is the same as specifying the following `at#()` options:

```
. stcurve, cif at1(ifp=5 pelnode=0) at2(ifp=20 pelnode=0)
```

◁

stcurve after stintreg and stintcox

stcurve can be used after stintreg or stintcox to plot the fitted survivor, failure, hazard, or cumulative hazard function. For single-record interval-censored data, these functions can be evaluated at a lower or upper time endpoint of time intervals. stcurve after stintreg uses the lower and upper time endpoints to determine the range for the plotted functions. stcurve after stintcox plots the functions at the distinct time points formed by combining the lower and upper time endpoints. By default, without the at() option, stcurve computes the overall means of the covariates and evaluates the function at the overall means and at each time in the data, censored or uncensored. The resulting plot is therefore the survival experience of a subject with a covariate pattern equal to the average covariate pattern in the study. You can produce the plot at other values of the covariates by using the at() option or specify a time range by using the range() option. stcurve after stintcox can also be used to plot functions that allow covariates to vary over time; see *Remarks and examples* in [ST] **stintcox postestimation**.

▷ Example 6

We continue with example 1 of [ST] **stintreg**, which studies the effect of treatment on breast retraction for breast cancer patients. In that example, we compared the cosmetic effects of two cancer treatments, radiotherapy alone versus radiotherapy plus adjuvant chemotherapy, by fitting a Weibull proportional hazards model:

```
. use https://www.stata-press.com/data/r18/cosmesis, clear
(Cosmetic deterioration of breast cancer patients)

. stintreg i.treat, interval(ltime rtime) distribution(weibull)
 (iteration log omitted)
```

Weibull PH regression

Number of obs	=	94			
Uncensored	=	0			
Left-censored	=	5			
Right-censored	=	38			
Interval-cens.	=	51			

LR chi2(1) = 10.93
Log likelihood = −143.19228   Prob > chi2 = 0.0009

	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]
treat					
Radio+Chemo	2.498526	.7069467	3.24	0.001	1.434961 4.350383
_cons	.0018503	.0013452	−8.66	0.000	.000445 .007693
/ln_p	.4785786	.1198972	3.99	0.000	.2435844 .7135729
p	1.613779	.1934876			1.275814 2.041271
1/p	.6196635	.0742959			.4898907 .7838134

Note: _cons estimates baseline hazard.

Now, we wish to compare the average patient's survival curve under radiotherapy only (`treat == 0`) and under radiotherapy plus chemotherapy (`treat == 1`):

```
. stcurve, survival at(treat=(0 1))
note: function evaluated at specified covariate values.
```
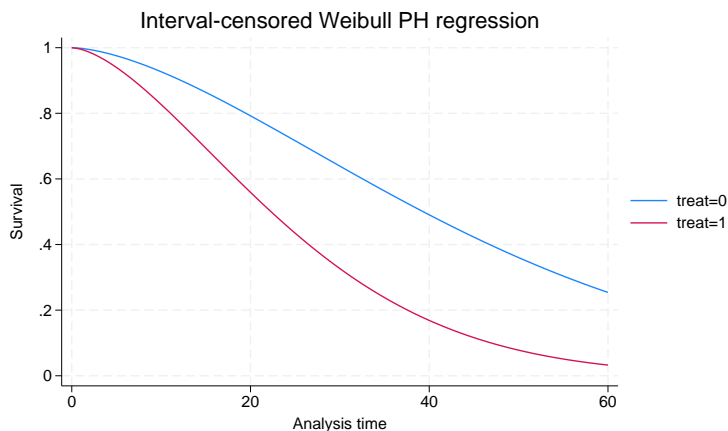

Figure 9. Treatment-specific survivor functions for Weibull proportional hazards model

From figure 9, we see that the risk of developing breast retraction for an average patient receiving the radiotherapy-plus-chemotherapy treatment is higher than that for the same patient receiving radiotherapy-only treatment. In other words, the adjuvant chemotherapy increases the risk of breast retraction.

Let's now use `stintcox` to fit a semiparametric Cox model that relaxes the distributional assumption about the event-time distribution. To speed up execution, we will use the `favorspeed` option in this demonstration.

```
. stintcox i.treat, interval(ltime rtime) favorspeed
note: using fixed step size with a multiplier of 5 to compute derivatives.
note: using EM and VCE tolerances of 0.0001.
note: option noemhsgtolerance assumed.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -150.52924
Iteration 36:  Log likelihood = -133.02071

Computing standard errors: ... done

Interval-censored Cox regression                Number of obs     =      94
Baseline hazard: Reduced intervals                   Uncensored =       0
                                                    Left-censored =       5
Event-time interval:                               Right-censored =      38
  Lower endpoint: ltime                            Interval-cens. =      51
  Upper endpoint: rtime
                                                   Wald chi2(1)      =    8.34
Log likelihood = -133.02071                        Prob > chi2       =  0.0039
```

		OPG				
	Haz. ratio	std. err.	z	P>|z|	[95% conf. interval]	
treat						
Radio+Chemo	2.229089	.6188939	2.89	0.004	1.293589	3.841127

Note: Standard error estimates may be more variable for small datasets and datasets with low proportions of interval-censored observations.

And we now compare the survivor functions of the two treatment groups:

```
. stcurve, survival at(treat=(0 1))
note: function evaluated at specified covariate values.
```
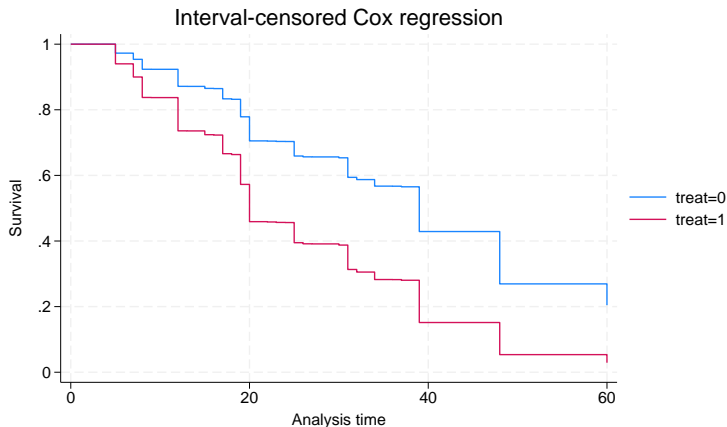

Figure 10. Treatment-specific survivor functions for Cox proportional hazards model

The survivor functions for the semiparametric Cox model are step functions but they look similar to the Weibull survivor functions from figure 9.

◁

Using at() with stcurve

stcurve, by default, evaluates the function by setting each covariate to its mean value. The at() option specifies that the function be evaluated at the values of the covariates listed in at() and at the means of all unlisted covariates. You can repeat the at() option to produce multiple curves corresponding to different sets of covariate values.

▷ Example 7

Let's return to example 1. Suppose that we want to compare the survival curves for patients at ages 20, 30, 40, and 50 of the treatment group. The easiest way to do this is to specify multiple values for `age` in `at(numlist)`.

```
. use https://www.stata-press.com/data/r18/drugtr
(Patient survival in drug trial)
. quietly stcox age drug
. stcurve, survival at(age=(20(10)50) drug=1)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```
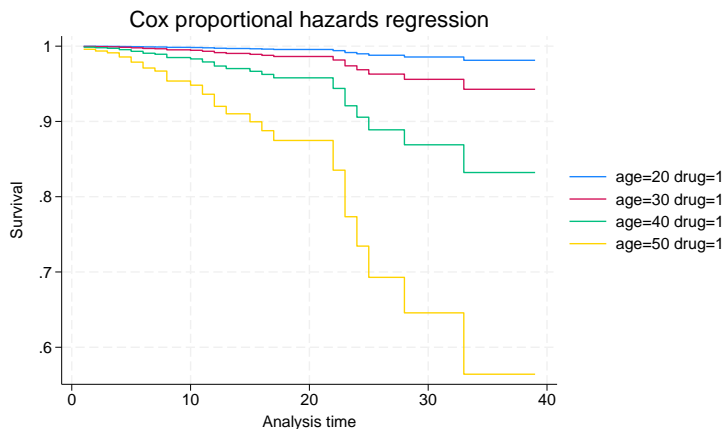

We could have obtained the same plot by specifying the `at#()` options but with more typing:

```
stcurve, survival at1(age=20 drug=1) at2(age=30 drug=1) ///
    at3(age=40 drug=1) at4(age=50 drug=1)
```

The `at()` option provides many other flexible specifications. For example, if we would like to plot the baseline survivor function, we do not need to set every covariate to zero in `at()`. We can set all covariates to zero at once as follows:

```
. stcurve, survival at((zero) _all)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```
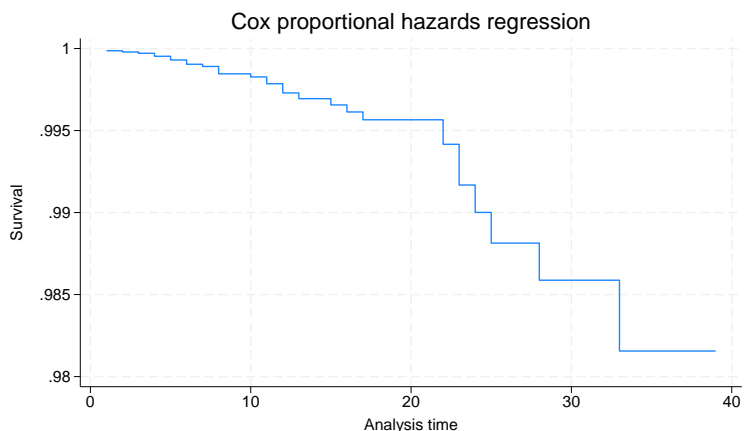

For details about the `at()` option, see *Syntax of at()* in [ST] **adjustfor_option**.

◁

References

Cefalu, M. S. 2011. Pointwise confidence intervals for the covariate-adjusted survivor function in the Cox model. *Stata Journal* 11: 64–81.

Ruhe, C. 2016. Estimating survival functions after stcox with time-varying coefficients. *Stata Journal* 16: 867–879.

Also see

[ST] **stcox** — Cox proportional hazards model

[ST] **stcox postestimation** — Postestimation tools for stcox

[ST] **stcrreg** — Competing-risks regression

[ST] **stcrreg postestimation** — Postestimation tools for stcrreg

[ST] **stintcox** — Cox proportional hazards model for interval-censored survival-time data

[ST] **stintcox postestimation** — Postestimation tools for stintcox

[ST] **stintreg** — Parametric models for interval-censored survival-time data

[ST] **stintreg postestimation** — Postestimation tools for stintreg

[ST] **streg** — Parametric survival models

[ST] **streg postestimation** — Postestimation tools for streg

[ST] **sts** — Generate, graph, list, and test the survivor and related functions

[ST] **stset** — Declare data to be survival-time data

[ST] *adjustfor_option* — Adjust survivor and related functions for covariates at specific values

[ME] **mestreg** — Multilevel mixed-effects parametric survival models

[ME] **mestreg postestimation** — Postestimation tools for mestreg

[XT] **xtstreg** — Random-effects parametric survival models

[XT] **xtstreg postestimation** — Postestimation tools for xtstreg

Title

stdescribe — Describe survival-time data

Description

stdescribe reports the characteristics of a survival-time dataset. The report includes the number of subjects and per-subject summary statistics related to the number of records, entry and exit times, gaps in the data, time at risk, and number of failures.

stdescribe can be used with single- or multiple-record and single- or multiple-failure st data.

Quick start

Report characteristics of a survival-time dataset using stset data

    stdescribe

Describe only data with $v1 = 1$

    stdescribe if v1==1

Compute weighted statistics using the weight specified in stset

    stdescribe, weight

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Describe survival-time data

Syntax

> stdescribe [_if_] [_in_] [, <u>weight</u> <u>nosh</u>ow]

You must stset your data before using stdescribe; see [ST] stset.

by and collect are allowed; see [U] **11.1.10 Prefix commands**.

fweights, iweights, and pweights may be specified using stset; see [ST] stset.

Options

> [Main]

weight specifies that the summary use weighted rather than unweighted statistics. weight does
nothing unless you specified a weight when you stset the data. The weight option and the
ability to ignore weights are unique to stdescribe. The purpose of stdescribe is to describe
the data in a computer sense—the number of records, etc.—and for that purpose, the weights are
best ignored.

noshow prevents stdescribe from showing the key st variables. This option is seldom used because
most people type stset, show or stset, noshow to set whether they want to see these variables
mentioned at the top of the output of every st command; see [ST] stset.

Remarks and examples

Here is an example of stdescribe with single-record survival data:

```
. use https://www.stata-press.com/data/r18/page2

. stdescribe

        Failure _d: dead
  Analysis time _t: time
```

			Per subject		
Category	Total	Mean	Min	Median	Max
Number of subjects	40				
Number of records	40	1	1	1	1
Entry time (first)		0	0	0	0
Exit time (final)		227.95	142	231	344
Subjects with gap	0				
Time on gap	0				
Time at risk	9118	227.95	142	231	344
Failures	36	.9	0	1	1

There is one record per subject. The purpose of this summary is not analysis—it is to describe how
the data are arranged. We can quickly see that there is one record per subject (the number of subjects
equals the number of records, but if there is any doubt, the minimum and maximum number of
records per subject is 1), that all the subjects entered at time 0, that the subjects exited between times
142 and 344 (median 231), that there are no gaps (as there could not be if there is only one record
per subject), that the total time at risk is 9,118 (distributed reasonably evenly across the subjects),
and that the total number of failures is 36 (with a maximum of 1 failure per subject).

Here is a description of the multiple-record Stanford heart transplant data that we introduced in
[ST] **stset**:

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)

. stdescribe

        Failure _d: died
  Analysis time _t: t1
       ID variable: id
```

Category	Total	Mean	Min	Median	Max
Number of subjects	103				
Number of records	172	1.669903	1	2	2
Entry time (first)		0	0	0	0
Exit time (final)		310.0786	1	90	1799
Subjects with gap	0				
Time on gap	0	.	.	.	.
Time at risk	31938.1	310.0786	1	90	1799
Failures	75	.7281553	0	1	1

Here patients have one or two records. Although this is not revealed by the output, a patient has one
record if the patient never received a heart transplant and two if the patient did receive a transplant;
the first reflects the patient's survival up to the time of transplantation and the second their subsequent
survival:

```
. stset, noshow             /* to not show the st marker variables */

. stdescribe if !transplant
```

Category	Total	Mean	Min	Median	Max
Number of subjects	34				
Number of records	34	1	1	1	1
Entry time (first)		0	0	0	0
Exit time (final)		96.61765	1	21	1400
Subjects with gap	0				
Time on gap	0	.	.	.	.
Time at risk	3285	96.61765	1	21	1400
Failures	30	.8823529	0	1	1

```
. stdescribe if transplant
```

Category	Total	Mean	Min	Median	Max
Number of subjects	69				
Number of records	138	2	2	2	2
Entry time (first)		0	0	0	0
Exit time (final)		415.2623	5.1	207	1799
Subjects with gap	0				
Time on gap	0	.	.	.	.
Time at risk	28653.1	415.2623	5.1	207	1799
Failures	45	.6521739	0	1	1

Finally, here are the results of stdescribe from multiple-failure data:

```
. use https://www.stata-press.com/data/r18/mfail2

. stdescribe
```

Category	Total	Mean	Min	Median	Max
			Per subject		
Number of subjects	926				
Number of records	1734	1.87257	1	2	4
Entry time (first)		0	0	0	0
Exit time (final)		470.6857	1	477	960
Subjects with gap	6				
Time on gap	411	68.5	16	57.5	133
Time at risk	435444	470.2419	1	477	960
Failures	808	.8725702	0	1	3

The maximum number of failures per subject observed is three, although 50% had just one failure, and six subjects have gaps in their histories.

Video example

[How to describe and summarize survival data](#)

Stored results

stdescribe stores the following in r():

Scalars

r(N_sub)	number of subjects	r(gap)	total gap, if gap
r(N_total)	number of records	r(gap_min)	minimum gap, if gap
r(N_min)	minimum number of records	r(gap_mean)	mean gap, if gap
r(N_mean)	mean number of records	r(gap_med)	median gap, if gap
r(N_med)	median number of records	r(gap_max)	maximum gap, if gap
r(N_max)	maximum number of records	r(tr)	total time at risk
r(t0_min)	minimum first entry time	r(tr_min)	minimum time at risk
r(t0_mean)	mean first entry time	r(tr_mean)	mean time at risk
r(t0_med)	median first entry time	r(tr_med)	median time at risk
r(t0_max)	maximum first entry time	r(tr_max)	maximum time at risk
r(t1_min)	minimum final exit time	r(N_fail)	number of failures
r(t1_mean)	mean final exit time	r(f_min)	minimum number of failures
r(t1_med)	median final exit time	r(f_mean)	mean number of failures
r(t1_max)	maximum final exit time	r(f_med)	median number of failures
r(N_gap)	number of subjects with gap	r(f_max)	maximum number of failures

Reference

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata Press.

Also see

[ST] **stset** — Declare data to be survival-time data

[ST] **stsum** — Summarize survival-time data

[ST] **stvary** — Report variables that vary over time

Title

> **stfill —** Fill in by carrying forward values of covariates

Description

stfill is intended for use with multiple-record st data for which id() has been stset. stfill may be used with single-record data, but it does nothing. That is, stfill can be used with multiple-record or single- or multiple-failure st data.

stfill, baseline changes variables to contain the value at the earliest time each subject was observed, making the variable constant over time. stfill, baseline changes all subsequent values of the specified variables to equal the first value, whether they originally contained missing or not.

stfill, forward fills in missing values of each variable with that of the most recent time at which the variable was last observed. stfill, forward changes only missing values.

You must specify either the baseline or the forward option.

if *exp* and in *range* operate slightly differently from their usual definitions to work as you would expect. if and in restrict where changes can be made to the data, but no matter what, all stset observations are used to provide the values to be carried forward.

Quick start

Replace values of x1 with the value of x1 at the earliest time the subject was observed using multiple-record [stset](#) data

    stfill x1, baseline

Replace missing values in x1 and x2 with the most recently observed value of the variable for the subject

    stfill x1 x2, forward

Menu

Statistics > Survival analysis > Setup and utilities > Fill forward with values of covariates

Syntax

> stfill *varlist* [*if*] [*in*] , { <u>ba</u>seline | <u>f</u>orward } [*options*]

options	Description
Main	
* <u>ba</u>seline	replace with values at baseline
* <u>f</u>orward	carry forward values
<u>nosh</u>ow	do not show st setting information

* Either baseline or forward is required.

You must stset your data before using stfill; see [ST] **stset**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

Options

> [Main]

baseline specifies that values be replaced with the values at baseline, the earliest time at which the subject was observed. All values of the specified variables are replaced, missing and nonmissing.

forward specifies that values be carried forward and that previously observed, nonmissing values be used to fill in later values that are missing in the specified variables.

noshow prevents stfill from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

Remarks and examples

stfill assists in fixing data errors and makes baseline analyses easier.

▷ Example 1

Let's begin by repairing broken data.

You have a multiple-record st dataset that, because of how it was constructed, has a problem with the gender variable:

```
. use https://www.stata-press.com/data/r18/mrecord

. stvary sex
        Failure _d: myopic
    Analysis time _t: t
        ID variable: id

            Subjects for whom the variable is
                                              never    always sometimes
    Variable    constant     varying        missing   missing   missing

         sex         131           1             22         0       110
```

For 110 subjects, sex is sometimes missing, and for one more subject, the value of sex changes over time! The sex change is an error, but the missing values occurred because sometimes the subject's sex was not filled in on the revisit forms. We will assume that you have checked the changing-sex subject and determined that the baseline record is correct in that case, too.

```
. stfill sex, baseline
        Failure _d: myopic
   Analysis time _t: t
        ID variable: id
Replacing all values with value at earliest observed time:
             sex: 221 real changes made
. stvary sex
        Failure _d: myopic
   Analysis time _t: t
        ID variable: id
              Subjects for whom the variable is
                                         never    always sometimes
     Variable |   constant     varying    missing   missing   missing
   ────────────┼──────────────────────────────────────────────────────
          sex |        132           0        132         0         0
```

The sex variable is now completely filled in.

In this same dataset, there is another variable—bp, blood pressure—that is not always filled in because readings were not always taken.

```
. stvary bp
        Failure _d: myopic
   Analysis time _t: t
        ID variable: id
              Subjects for whom the variable is
                                         never    always sometimes
     Variable |   constant     varying    missing   missing   missing
   ────────────┼──────────────────────────────────────────────────────
           bp |         18         114          9         0       123
```

(bp is constant for 18 patients because it was taken only once—at baseline.) Anyway, you decide that it will be good enough when bp is missing to use the previous value of bp:

```
. stfill bp, forward noshow
Replacing missing values with previously observed values:
              bp: 263 real changes made
. stvary bp, noshow
              Subjects for whom the variable is
                                         never    always sometimes
     Variable |   constant     varying    missing   missing   missing
   ────────────┼──────────────────────────────────────────────────────
           bp |         18         114        132         0         0
```

So much for data repair and fabrication.

◁

▷ Example 2

Much later, deep in analysis, you are concerned about the bp variable and decide to compare results with a model that simply includes blood pressure at baseline. You are undecided on the issue and want to have both variables in your data:

```
. stset, noshow
. gen bp0 = bp
. stfill bp0, baseline
Replacing all values with value at earliest observed time:
          bp0: 406 real changes made
. stvary bp bp0
```

| | | | | never | always | sometimes |
Variable	constant	varying		missing	missing	missing
bp	18	114		132	0	0
bp0	132	0		132	0	0

◁

Also see

[ST] **stbase** — Form baseline dataset

[ST] **stgen** — Generate variables reflecting entire histories

[ST] **stset** — Declare data to be survival-time data

[ST] **stvary** — Report variables that vary over time

Title

> **stgen** — Generate variables reflecting entire histories

Description

stgen provides a convenient way to generate new variables reflecting entire histories. These functions are intended for use with multiple-record survival data but may be used with single-record data. With single-record data, each function reduces to one generate, and generate would be a more natural way to approach the problem.

stgen can be used with single- or multiple-failure st data.

If you want to generate calculated values, such as the survivor function, see [ST] **sts**.

Quick start

Create binary indicator newv1 equal to 1 in all records for a subject if $v1 = 1$ at any time using multiple-record stset data

```
stgen newv1 = ever(v1==1)
```

Create newv2 containing the time when v2 is first greater than 5 for the subject

```
stgen newv2 = when(v2>5)
```

Same as above, but assume $v2 > 5$ becomes true at the beginning instead of at the end of the corresponding record

```
stgen newv2 = when0(v2>5)
```

Create newv3 containing the cumulative number of records with $v1 = 1$ for the subject

```
stgen newv3 = count(v1==1)
```

Same as above, but assume $v1 = 1$ becomes true at the beginning instead of at the end of the corresponding record

```
stgen newv3 = count0(v1==1)
```

Create newv4 containing the cumulative number of gaps for the subject

```
stgen newv4 = ngaps()
```

Menu

Statistics > Survival analysis > Setup and utilities > Generate variable reflecting entire histories

Syntax

> stgen [*type*] *newvar* = *function*

where *function* is

> ever(*exp*)
> never(*exp*)
> always(*exp*)
> min(*exp*)
> max(*exp*)
> when(*exp*)
> when0(*exp*)
> count(*exp*)
> count0(*exp*)
> minage(*exp*)
> maxage(*exp*)
> avgage(*exp*)
> nfailures()
> ngaps()
> gaplen()
> hasgap()

> You must stset your data before using stgen; see [ST] stset.

Functions

In the description of the functions below, time units refer to the same units as *timevar* from stset *timevar*, ... For instance, if *timevar* is the number of days since 01 January 1960 (a Stata date), time units are days. If *timevar* is in years—years since 1960, years since diagnosis, or whatever—time units are years.

When we say variable X records a "time", we mean a variable that records when something occurred in the same units and with the same base as *timevar*. If *timevar* is a Stata date, "time" is correspondingly a Stata date.

t units, or analysis-time units, refer to a variable in the units *timevar*/scale() from stset *timevar*, scale(...) . . . . If you did not specify a scale(), t units are the same as time units. Alternatively, say that *timevar* is recorded as a Stata date and you specified scale(365.25). Then t units are years. If you specified a nonconstant scale—scale(myvar), where myvar varies from subject to subject—t units are different for every subject.

"An analysis time" refers to the time something occurred, recorded in the units (*timevar*−origin()))/scale(). We speak about analysis time only in terms of the beginning and end of each time-span record.

Although in *Description* above we said that stgen creates variables reflecting entire histories, stgen restricts itself to the stset observations, so "entire history" means the entire history as it is currently stset. If you really want to use entire histories as recorded in the data, type streset, past or streset, past future before using stgen. Then type streset to reset to the original analysis sample.

The following functions are available:

ever(*exp*) creates *newvar* containing 1 (true) if the expression is ever true (nonzero) and 0 otherwise. For instance,

```
. stgen everlow = ever(bp<100)
```

would create everlow containing, for each subject, uniformly 1 or 0. Every record for a subject would contain everlow = 1 if, on any stset record for the subject, bp < 100; otherwise, everlow would be 0.

never(*exp*) is the reverse of ever(); it creates *newvar* containing 1 (true) if the expression is always false (0) and 0 otherwise. For instance,

```
. stgen neverlow = never(bp<100)
```

would create neverlow containing, for each subject, uniformly 1 or 0. Every record for a subject would contain neverlow = 1 if, on every stset record for the subject, bp < 100 is false.

always(*exp*) creates *newvar* containing 1 (true) if the expression is always true (nonzero) and 0 otherwise. For instance,

```
. stgen lowlow = always(bp<100)
```

would create lowlow containing, for each subject, uniformly 1 or 0. Every record for a subject would contain lowlow = 1 if, on every stset record for a subject, bp < 100.

min(*exp*) and max(*exp*) create *newvar* containing the minimum or maximum nonmissing value of *exp* within id(). min() and max() are often used with variables recording a time (see [definition](#) above), such as min(visitdat).

when(*exp*) and when0(*exp*) create *newvar* containing the time when *exp* first became true within the previously stset id(). The result is in time, not t units; see the [definition](#) above.

when() and when0() differ about when the *exp* became true. Records record time spans (*time0*, *time1*]. when() assumes that the expression became true at the end of the time span, *time1*. when0() assumes that the expression became true at the beginning of the time span, *time0*.

Assume that you previously stset myt, failure(*eventvar* =...) .... when() would be appropriate for use with *eventvar*, and, presumably, when0() would be appropriate for use with the remaining variables.

count(*exp*) and count0(*exp*) create *newvar* containing the number of occurrences when *exp* is true within id().

count() and count0() differ in when they assume that *exp* occurs. count() assumes that *exp* corresponds to the end of the time-span record. Thus even if *exp* is true in this record, the count would remain unchanged until the next record.

count0() assumes that *exp* corresponds to the beginning of the time-span record. Thus if *exp* is true in this record, the count is immediately updated.

For example, assume that you previously stset myt, failure(*eventvar*=...) .... count() would be appropriate for use with *eventvar*, and, presumably, count0() would be appropriate for use with the remaining variables.

minage(*exp*), maxage(*exp*), and avgage(*exp*) return the elapsed time, in time units, because *exp* is at the beginning, end, or middle of the record, respectively. *exp* is expected to evaluate to a time in time units. minage(), maxage(), and avgage() would be appropriate for use with the result of when(), when0(), min(), and max(), for instance.

Also see [ST] **stsplit**; stsplit will divide the time-span records into new time-span records that record specified intervals of ages.

nfailures() creates *newvar* containing the cumulative number of failures for each subject as of the entry time for the observation. nfailures() is intended for use with multiple-failure data; with single-failure data, nfailures() is always 0. In multiple-failure data,

    . stgen nfail = nfailures()

might create, for a particular subject, the following:

id	time0	time1	fail	x	nfail
93	0	20	0	1	0
93	20	30	1	1	0
93	30	40	1	2	1
93	40	60	0	1	2
93	60	70	0	2	2
93	70	80	1	1	2

The total number of failures for this subject is 3, and yet the maximum of the new variable nfail is 2. At time 70, the beginning of the last record, there had been two failures previously, and there were two failures up to but not including time 80.

ngaps() creates *newvar* containing the cumulative number of gaps for each subject as of the entry time for the record. Delayed entry (an opening gap) is not considered a gap. For example,

    . stgen ngap = ngaps()

might create, for a particular subject, the following:

id	time0	time1	fail	x	ngap
94	10	30	0	1	0
94	30	40	0	2	0
94	50	60	0	1	1
94	60	70	0	2	1
94	82	90	1	1	2

gaplen() creates *newvar* containing the time on gap, measured in analysis-time units, for each subject as of the entry time for the observation. Delayed entry (an opening gap) is not considered a gap. Continuing with the previous example,

    . stgen gl = gaplen()

would produce

id	time0	time1	fail	x	ngap	gl
94	10	30	0	1	0	0
94	30	40	0	2	0	0
94	50	60	0	1	1	10
94	60	70	0	2	1	0
94	82	90	1	1	2	12

hasgap() creates *newvar* containing uniformly 1 if the subject ever has a gap and 0 otherwise. Delayed entry (an opening gap) is not considered a gap.

Remarks and examples

stgen does nothing you cannot do in other ways, but it is convenient.

Consider how you would obtain results like those created by stgen should you need something that stgen will not create for you. Say that we have an st dataset for which we have previously

    . stset t, failure(d) id(id)

Assume that these are some of the data:

```
id          t          d          bp
27          30         0          90
27          50         0         110
27          60         1          85
28          11         0         120
28          40         1         130
```

If we were to type

```
. stgen everlow = ever(bp<100)
```

the new variable, everlow, would contain for these two subjects

```
id          t          d          bp    everlow
27          30         0          90          1
27          50         0         110          1
27          60         1          85          1
28          11         0         120          0
28          40         1         130          0
```

Variable everlow is 1 for subject 27 because, in two of the three observations, bp < 100, and everlow is 0 for subject 28 because everlow is never less than 100 in either observation.

Here is one way we could have created everlow for ourselves:

```
. generate islow = bp<100
. sort id
. by id: generate sumislow = sum(islow)
. by id: generate everlow = sumislow[_N]>0
. drop islow sumislow
```

The generic term for code like this is explicit subscripting; see [U] **13.7 Explicit subscripting**.

Anyway, that is what stgen did for us, although, internally, stgen used denser code that was equivalent to

```
. by id, sort: generate everlow=sum(bp<100)
. by id: replace everlow = everlow[_N]>0
```

Obtaining things like the time on gap is no more difficult. When we stset the data, stset created variable _t0 to record the entry time. stgen's gaplen() function is equivalent to

```
. sort id _t
. by id: generate gaplen = _t0-_t[_n-1]
. by id: replace gaplen = 0 if _n == 1
```

Seeing this, you should realize that if all you wanted was the cumulative length of the gap before the current record, you could type

```
. sort id _t
. by id: generate curgap = sum(_t0-_t[_n-1])
```

If, instead, you wanted a variable that was 1 if there were a gap just before this record and 0 otherwise, you could type

```
. sort id _t
. by id: generate iscurgap = (_t0-_t[_n-1])>0
```

▷ Example 1

Let's use the `stgen` commands to real effect. We have a multiple-record, multiple-failure dataset.

```
. use https://www.stata-press.com/data/r18/mrmf, clear
. st
-> stset t, id(id) failure(d) time0(t0) exit(time .) noshow
Survival-time data settings
           ID variable: id
        Failure event: d!=0 & d<.
Observed time interval: (t0, t]
    Exit on or before: time .
. stdescribe
```

Category	Total	Mean	Min	Median	Max
			Per subject		
Number of subjects	926				
Number of records	1734	1.87257	1	2	4
Entry time (first)		0	0	0	0
Exit time (final)		470.6857	1	477	960
Subjects with gap	6				
Time on gap	411	68.5	16	57.5	133
Time at risk	435444	470.2419	1	477	960
Failures	808	.8725702	0	1	3

Also in this dataset are two covariates, `x1` and `x2`. We wish to fit a Cox model on these data but wish to assume that the baseline hazard for first failures is different from that for second and later failures.

Our data contain six subjects with gaps. Because failures might have occurred during the gap, we begin by dropping those six subjects:

```
. stgen hg = hasgap()
. drop if hg
(14 observations deleted)
```

The six subjects had 14 records among them. We can now create variable `nf` containing the number of failures and, from that, create variable `group`, which will be 0 when subjects have experienced no previous failures and 1 thereafter:

```
. stgen nf = nfailures()
. generate byte group = nf>0
```

We can now fit our stratified model:

```
. stcox x1 x2, strata(group) vce(robust)

Iteration 0:  Log pseudolikelihood = -4499.9966
Iteration 1:  Log pseudolikelihood = -4444.7797
Iteration 2:  Log pseudolikelihood = -4444.4596
Iteration 3:  Log pseudolikelihood = -4444.4596
Refining estimates:
Iteration 0:  Log pseudolikelihood = -4444.4596

Stratified Cox regression with Breslow method for ties
Strata variable: group
No. of subjects =      920                          Number of obs =  1,720
No. of failures =      800
Time at risk    = 432,153
                                                    Wald chi2(2)  = 102.78
Log pseudolikelihood = -4444.4596                   Prob > chi2   = 0.0000

                            (Std. err. adjusted for 920 clusters in id)
```

_t	Haz. ratio	Robust std. err.	z	P>|z|	[95% conf. interval]	
x1	2.087903	.1961725	7.84	0.000	1.736738	2.510074
x2	.2765613	.052277	-6.80	0.000	.1909383	.4005806

◁

Also see

Title

> **stintcox** — Cox proportional hazards model for interval-censored survival-time data

Description

stintcox fits semiparametric Cox proportional hazards models to interval-censored survival-time data or, more precisely, event-time data, which may contain right-censored, left-censored, and interval-censored observations. stintcox can be used with single- or multiple-record [interval-censored](#) data. With interval-censored data, the [event-time variables](#) are specified with the stintcox command instead of using [stset](#). All st settings are ignored by stintcox.

Quick start

Single-record-per-subject interval-censored data

Cox proportional hazards model with covariates x1 and x2 fit to interval-censored data with lower and upper interval endpoints t1 and t2
    stintcox x1 x2, interval(t1 t2)

Same as above, but estimate the baseline hazard function using all observed intervals instead of the default reduced set
    stintcox x1 x2, interval(t1 t2) full

Use less stringent convergence criteria to explore initial results more quickly
    stintcox x1 x2, interval(t1 t2) favorspeed

Fit a stratified Cox model with strata defined by levels of svar
    stintcox x1 x2, interval(t1 t2) strata(svar)

Include a time-varying covariate by interacting x2 with the logarithm of analysis time
    stintcox x1 x2, interval(t1 t2) tvc(x2) texp(ln(_t))

Report the log-likelihood model test instead of the default Wald model test, and report regression coefficients instead of hazard ratios
    stintcox x1 x2, interval(t1 t2) lrmodel nohr

Report OIM standard errors instead of the default OPG standard errors
    stintcox x1 x2, interval(t1 t2) vce(oim)

After estimation, report regression coefficients instead of hazard ratios
    stintcox, nohr

After estimation, report OPG standard errors using fixed step size instead of the default adaptive step size
    stintcox, vce(opg, stepsize(fixed))

After estimation, save estimated baseline hazard contributions to basehc.dta, and store estimation results as intcox

```
stintcox, saving(basehc)
estimates store intcox
```

Multiple-record-per-subject interval-censored data

Cox proportional hazards model with baseline (time-invariant) covariate x1 and time-varying covariate x2 fit to multiple-record-per-subject interval-censored data with subject identifier idvar, examination time tvar, and event status indicator status

```
stintcox x1 x2, id(idvar) time(tvar) status(status)
```

Same as above, but use the covariate values at the nearest examination time on the right instead of the default nearest examination time on the left to impute values of time-varying covariate x2 between two examination times

```
stintcox x1 x2, id(idvar) time(tvar) status(status)   ///
    tvcovimpute(nearright)
```

Report robust standard errors instead of the default OPG standard errors

```
stintcox x1 x2, id(idvar) time(tvar) status(status) vce(robust)
```

Report cluster–robust standard errors with the cluster identifier clustvar

```
stintcox x1 x2, id(idvar) time(tvar) status(status)   ///
    vce(cluster clustvar)
```

Menu

Statistics > Survival analysis > Regression models > Interval-censored Cox PH model

Syntax

Single-record-per-subject interval-censored data

stintcox $\big[$ *indepvars* $\big]$ $\big[$ *if* $\big]$ $\big[$ *in* $\big]$, <u>int</u>erval(t_l t_u) $\big[$ *single_options* $\big]$

Multiple-record-per-subject interval-censored data

stintcox $\big[$ *indepvars* $\big]$ $\big[$ *if* $\big]$ $\big[$ *in* $\big]$, id(*idvar*) time(*timevar*) status(*statusvar*)

$\big[$ *multiple_options* $\big]$

single_options	Description
Model	
* <u>int</u>erval(t_l t_u)	specify lower and upper endpoints for the [event-time interval](#)
[options](#)	options for both single- and multiple-record interval-censored data

**interval(t_l t_u) is required for single-record interval-censored data and cannot be combined with option id(), time(), status(), or tvcovimpute().

multiple_options	Description
Model	
† id(*idvar*)	specify multiple-record ID variable
† time(*timevar*)	specify [examination time](#) variable
† status(*statusvar*)	specify event status indicator variable
<u>tvcovimpute</u>(*[type](#)*)	specify how to impute unobserved covariate values between examination times for time-varying covariates; default is nearleft
statussysmissok	retain the observations that contain system missing values (.)
[options](#)	options for both single- and multiple-record interval-censored data

†id(), time(), and status() are required for multiple-record interval-censored data and cannot be combined with option interval().

options	Description
[Model]	
strata(*varlist*)	specify strata variables
reduced	estimate baseline hazard function using a reduced set of time intervals; the default
full	estimate baseline hazard function using all time intervals
<u>favoraccuracy</u>	favor accuracy of results over speed; the default
<u>favorspeed</u>	favor speed possibly over accuracy of results
[Time varying]	
tvc(*varlist*$_t$)	specify covariates to be interacted with a function of time
texp(*exp*)	specify a function of time; default is texp(_t)
lrphtest	perform the likelihood-ratio test for covariates interacted with time; default is to perform Wald test
[SE/Robust]	
vce(*vcetype*)	*vcetype* may be one of opg (the default), oim, robust, or cluster *clustvar*; may be specified on replay of results
[Reporting]	
<u>level</u>(*#*)	set confidence level; default is level(95)
lrmodel	perform the likelihood-ratio model test instead of the default Wald model test
<u>saving</u>(*filename* [, replace])	save estimates of baseline hazard contributions to *filename*; use replace to overwrite existing *filename*
nohr	report regression coefficients, not hazard ratios
<u>nohead</u>er	suppress header from coefficient table
[no]log	display or suppress EM and VCE iteration logs; default is log
dots	display all EM and VCE iterations as dots
[no]vcedots[(*#*)]	display or suppress VCE iteration dots; default is to display a dot every iteration, meaning vcedots or vcedots(1)
[no]emlog[(*#*)]	display or suppress EM iteration log; default is emlog(100), which displays the log-likelihood value every 100 iterations
[no]emdots[(*#*)]	display or suppress EM iteration dots; default is noemdots
[display_options]	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling

EM options

<u>emiterate</u>(#)	maximum number of EM iterations; default is emiterate(5000)
<u>emtol</u>erance(#)	tolerance for the coefficient vector; default is emtolerance(1e-6)
<u>emltol</u>erance(#)	tolerance for the log likelihood; default is emltolerance(1e-7)
<u>emhsgtol</u>erance(#)	tolerance for the scaled gradient; default is emhsgtolerance(1e-5)
<u>noemhsgtol</u>erance	do not perform the scale-gradient convergence check
from(*init_specs*)	initial values for the regression coefficients
<u>coefl</u>egend	display legend instead of statistics

indepvars and *varlist$_t$* may contain factor variables; see [U] **11.4.3 Factor variables**.

by, collect, and statsby are allowed; see [U] **11.1.10 Prefix commands**.

coeflegend does not appear in the dialog box.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

vcetype	Description
<u>opg</u>[, *vce_options*]	outer product of the gradient (OPG) vectors; the default
<u>oim</u>[, *vce_options*]	observed information matrix (OIM)
<u>robust</u>[, *vce_options*]	Huber/White/sandwich estimator
<u>cl</u>uster *clustvar*[, *vce_options*]	clustered sandwich estimator

vce_options	Description
<u>stepsize</u>(adaptive | fixed [#])	use adaptive or fixed step size to compute VCE; default is adaptive step size
derivopts	options to control computation of numerical derivatives when adaptive step size is used
<u>iter</u>ate(#)	maximum number of iterations to compute VCE; default is iterate(5000)
<u>tol</u>erance(#)	profile log-likelihood tolerance to compute VCE; default is tolerance(1e-6)
[no]dots[(#)]	synonym for vcedots, vcedots(), and novcedots
post	replace the current e(V) with the specified VCE type; can be used only on replay with opg, oim, robust, or cluster

dots, dots(), nodots, and post do not appear in the dialog box.

Options

$\boxed{\text{Model}}$

`interval(`t_l t_u`)` is required with single-record-per-subject interval-censored data; see *Single- versus multiple-record interval-censored data formats* in *Remarks and examples*. It specifies two time variables that contain the endpoints of the event-time interval. t_l represents the lower endpoint, and t_u represents the upper endpoint. `interval()` may not be combined with option `id()`, `time()`, `status()`, or `tvcovimpute()`.

The interval time variables t_l and t_u should have the following form:

Type of observations		t_l	t_u
interval-censored	$(\,a, b\,]$	a	b
left-censored	$(\,0, b\,]$	.	b
left-censored	$(\,0, b\,]$	0	b
right-censored	$(\,a, +\infty\,)$	a	.
missing		.	.
missing		0	.

In the table, a and b satisfy $0 < a < b < \infty$. Also note that $t_l = t_u$ is not allowed with left-censored or interval-censored observations.

`id(`*idvar*`)` is required with multiple-record-per-subject interval-censored data; see *Single- versus multiple-record interval-censored data formats* in *Remarks and examples*. It specifies the subject-ID variable; observations with equal, nonmissing values of *idvar* are assumed to belong to the same subject. Observations for which *idvar* is missing are ignored.

When `id()` is not specified, each observation is assumed to represent a different subject and thus constitutes a single-record-per-subject dataset.

When you specify `id()`, the data are said to be multiple-record-per-subject data, even if it turns out that there is only one record per subject. Multiple-record-per-subject data can be used to accommodate time-varying covariates that exist in the dataset.

If you specify `id()`, `stintcox` requires that you also specify options `time()` and `status()`. `id()` may not be combined with the `interval()` option.

`time(`*timevar)* is required with multiple-record-per-subject interval-censored data; see *Single- versus multiple-record interval-censored data formats* in *Remarks and examples*. It specifies the examination times for the event of interest and may not be combined with the `interval()` option.

`status(`*statusvar)* is required with multiple-record-per-subject interval-censored data; see *Single- versus multiple-record interval-censored data formats* in *Remarks and examples*. It specifies a binary status indicator for the event of interest. For each examination time, *statusvar* indicates, by the value of 1 versus 0, whether the event has occurred between previous and current examination times. `status()` may not be combined with the `interval()` option.

In combination with option `statussysmissok`, observations with system missing values (.) in the *statusvar* variable will be used during estimation but will not be used to determine the event-time information for the corresponding subjects; see the description of option `statussysmissok`.

Recurrent events are not allowed. After the first event occurs, the subject is removed from the analysis, even if the subject has subsequent records in the data.

Options `time()` and `status()` together define the lower and upper endpoints of the event-time intervals (t_l and t_u) and the censoring types in a multiple-record-per-subject interval-censored dataset. If the event of interest occurs before the first examination time for a subject, the subject is left-censored. In the first record for this subject, time is the first examination time, and the event-status indicator is 1. The corresponding event-time interval for this subject has 0 as the left lower endpoint, t_l, and the first examination time as the right upper endpoint, t_u. If the event occurs between two examination times, the subject is interval-censored. The event-time interval for this subject has the last examination time where the status indicator is 0 as t_l and the first examination time where the status indicator is 1 as t_u. If the event does not occur during the study, the subject is right-censored. The event-time interval for this subject has the last examination time as t_l and missing time (.) as t_u.

`strata(`*varlist*`)` specifies the stratification variables. Observations with equal values of the strata variables are assumed to be in the same stratum. Stratified estimates (equal regression coefficients across strata but with a baseline hazard unique to each stratum) are then obtained.

`reduced`, the default, specifies that the baseline hazard function be estimated using a reduced (innermost) set of time intervals. This allows the estimator of the cumulative baseline hazard function to change its values only at the endpoints of the innermost time intervals, which were originally used by Turnbull (1976) to estimate the survivor function in the one-sample case. This option may not be combined with `full`.

`full` specifies that the baseline hazard function be estimated using all observed time intervals. In this case, the estimator of the cumulative baseline hazard function can potentially change its values at the endpoints of all the observed time intervals. This is the approach used by Zeng, Mao, and Lin (2016). It is more time consuming, but it may provide more accurate results. `full` may not be combined with `reduced`.

`favoraccuracy`, the default, and `favorspeed` control the tradeoff between accuracy of the results and the execution speed. `favoraccuracy` specifies that the command run longer to obtain more accurate results. `favorspeed` specifies that the command run faster at the possible expense of reduced accuracy of the results. You can use `favorspeed` for a quick initial exploration of the results and `favoraccuracy` for final reporting of the results.

When you specify `favorspeed`, `stintcox` uses less stringent convergence criteria to obtain the results. Specifically, it assumes lower EM coefficient, likelihood, and VCE tolerances of 0.0001 and implies option `noemhsgtolerance`. In addition, it uses a fixed step size with a multiplier of 5 instead of an adaptive step size when computing VCE. That is, specifying `favorspeed` is equivalent to specifying `emtolerance(0.0001)`, `emltolerance(0.0001)`, `noemhsgtolerance`, and `vce(, tolerance(0.0001) stepsize(fixed))`.

`tvcovimpute(`*type*`)` is used with multiple-record-per-subject interval-censored data and relevant only to variables included in the model whose values vary over time in the dataset. It specifies how to impute unobserved covariate values between two examination times for time-varying covariates in the dataset for each subject. *type* is one of `nearleft`, the default, `nearright`, `nearest`, or `first`. `tvcovimpute()` may not be combined with the `interval()` option.

`stintcox` requires covariate values for each subject at all distinct analysis times, but the data typically record covariates only at subject-specific examination times, and the covariate values at other analysis times need to be imputed. `stintcox` offers the following imputation methods.

`nearleft`, the default, uses covariate values at the nearest examination time on the left to impute covariate values at observation times that fall between two examination times for a given subject. This method is often preferred in practice because it does not use future covariate values to fill in the current values. This is also the method used with right-censored survival-time data.

nearright uses covariate values at the nearest examination time on the right to impute covariate values at analysis times that fall between two examination times for a given subject.

nearest uses covariate values at the nearest examination time, left or right, to impute covariate values at analysis times that fall between two examination times for a given subject.

For all three imputation methods above, if a subject is not examined at baseline (time 0), then the covariate value at the first examination time is used for all analysis times before the first examination time for this subject. And the covariate value at the last examination time is used for all observation times after the last examination time for a subject.

first replaces all covariate values for a subject with those at the first examination time, which is the same as using the baseline covariate values for all time-varying variables during estimation.

See *Methods and formulas* for details.

statussysmissok is used with multiple-record-per-subject format. It specifies that, during estimation, the observations that contain system missing values (.) in the event status variable specified in option status() be retained. These observations will not be used to determine the event-time intervals and censoring information for the corresponding subjects, but examination times and covariate values in these observations will be used during estimation. Without this option, stintcox omits system missing observations in status() from estimation, like any other missing value in any of the specified variables. Observations that contain extended missing values (.a through .z) are always omitted during estimation. Option statussysmissok is useful to create time-varying covariates in a multiple-record-per-subject format; see *Single- versus multiple-record interval-censored data formats* in *Remarks and examples*.

> ⌐ Time varying ⌐

tvc($varlist_t$) specifies the variables to be included in the model as an interaction with a function of time to form time-varying covariates. During estimation, these variables are interacted with analysis time or with a function of analysis time specified in the texp() option. This is a convenience option to include time-varying covariates that are deterministic functions of time. Using this option speeds up calculations and avoids having to split the data over many analysis times. tvc() in conjunction with texp() is also useful for testing the proportional-hazards assumption; see *Testing the proportional-hazards assumption using option tvc()* in *Remarks and examples*.

texp(exp) is used in conjunction with tvc($varlist_t$)) to specify the function of time that should be used to multiply covariates specified in the tvc() option to include in the model time-varying covariates that are deterministic functions of time. For example, specifying texp(ln(_t)) would cause the covariates in option tvc() to be multiplied by the logarithm of analysis time. If tvc($varlist_t$) is used without texp(exp), Stata understands that you mean texp(_t) and thus multiplies the covariates by the analysis time, denoted as _t here.

lrphtest is used in conjunction with tvc($varlist_t$). It performs the likelihood-ratio test between the full model and the model without covariates interacted with time, that is, without specifying option tvc(). By default, the Wald test of coefficients on time-varying covariates equal to zero is reported when option tvc($varlist_t$) is specified.

Options tvc($varlist_t$), texp(exp), and lrphtest are explained more in *Testing the proportional-hazards assumption using option tvc()* in *Remarks and examples*.

SE/Robust

vce(*vcetype*) specifies the type of standard error estimate reported. vce() may be specified at the time of estimation or when replaying results. If specified when replaying results, vce()-related stored results are not updated unless the post suboption is specified. *vcetype* may be one of the following:

vce(opg[, *vce_options*]) uses the sum of the OPG vectors based on the profile log likelihood; see *Methods and formulas*. vce(opg) is the default.

vce(oim[, *vce_options*]) uses the sum of the OIM vectors based on the profile log likelihood; see *Methods and formulas*.

vce(robust[, *vce_options*]) uses the robust or sandwich estimator of variance based on the profile log likelihood; see *Methods and formulas*.

vce(cluster *clustvar* [, *vce_options*]) specifies that the standard errors allow for intragroup correlation, relaxing the usual requirement that the observations be independent. That is, the observations are independent across groups (clusters) but not necessarily within groups. *clustvar* specifies to which group each observation belongs. The clustered sandwich estimator of variance is based on the profile log likelihood; see *Methods and formulas*.

vce_options may be stepsize(), *derivopts* with adaptive step size, iterate(#), tolerance(#), dots, dots(), nodots, and post.

    stepsize(adaptive | fixed [#]) specifies the step size for computing numerical derivatives with methods opg, oim, robust, or cluster. The default is stepsize(adaptive), which uses adaptive step size in computations; see [M-5] **deriv()**. stepsize(fixed) uses a fixed step size equal to $\delta_n = 5n^{-1/2}$, where n is the number of subjects or the number of clusters. stepsize(fixed #) uses a fixed step size equal to $\# \times n^{-1/2}$.

    derivopts are allowed only with stepsize(adaptive) and may be search(), h(), scale(), and bounds().

        search(*search_type*) specifies the approach used to search for an optimal step size for computing the numerical derivatives; three approaches are offered: bracket, interpolate, and off; see deriv_init_search() in [M-5] **deriv()**. The default is search(interpolate). In some cases, such as when factor variables have highly unbalanced levels, the search may lead to the step size that is too small or too large, which may lead to the error message that the estimates of baseline hazard contributions cannot be computed because the VCE matrix is close to being singular. Trying search(bracket) may be helpful in this case.

        h(# | *matname*) specifies the h values, which are multipliers for step size used to compute numerical derivatives; see deriv_init_h() in [M-5] **deriv()**. You can specify the same h value, #, for all parameters or parameter-specific h values as a Stata matrix (vector) *matname*.

        scale(# | *matname*) specifies the starting scale values used to compute numerical derivatives; see deriv_init_scale() in [M-5] **deriv()**. You can specify the same initial scale value, #, for all parameters or parameter-specific initial scale values as a Stata matrix (vector) *matname*.

        bounds($\#_1$ $\#_2$) specifies the minimum and maximum values used to search for optimal scale values; see deriv_init_bounds() in [M-5] **deriv()**. The default is bounds(1e-6 1e-5).

    iterate(#) specifies the maximum number of iterations to compute the VCE based on the profile log likelihood. The default is iterate(5000).

tolerance(#) specifies the tolerance for the profile log likelihood used to compute the VCE. The default is tolerance(1e-6).

dots, dots(#), and nodots display or suppress iteration dots showing the progress of the variance estimation. The dots are displayed by default, but you can use nodots to suppress them. By default, the dot is displayed every iteration, but you can change this by specifying dots(#).

When a fixed step size is used, an iteration corresponds to one derivative computation with respect to a regression coefficient. When an adaptive step size is used, an iteration corresponds to one call of the Mata deriv() function, which may be called multiple times to compute one derivative with respect to one regression coefficient. Thus, you will typically see more iteration dots with VCE estimation using an adaptive step size than using a fixed step size.

These options do not appear in the dialog box.

post can be used only on replay with vce(opg), vce(oim), vce(robust), or vce(cluster *clustvar*). It replaces the current e(V) with the specified *vcetype*. When vce(*vcetype*) is used on replay without post, the coefficient table will display the standard error of the specified *vcetype*, but e(V) will remain unchanged. This option does not appear in the dialog box.

┌─────────┐
│ Reporting │
└─────────┘

level(#), lrmodel; see [R] **Estimation options**.

saving(*filename* [, replace]) saves the estimated baseline hazard contributions in *filename*.dta. The replace option specifies to overwrite *filename*.dta if it exists. If option saving() is not specified, stintcox saves estimation results in a temporary file for later access by postestimation commands. This temporary file will be overridden every time stintcox is run and will also be erased if the current estimation results are cleared. saving() may be specified during estimation or on replay.

Because the file containing the baseline hazard contributions is considered to be part of estimation results, you must use option saving() before storing or saving your estimation results using estimates store or estimates save.

nohr specifies that regression coefficients be displayed rather than exponentiated regression coefficients or hazard ratios. This option affects only how results are displayed and not how they are estimated. nohr may be specified at estimation time or when replaying results.

noheader suppresses the output header, either at estimation or upon replay.

log and nolog display or suppress stintcox's iteration log, which includes both the EM and VCE iterations. The EM iteration log displays the log-likelihood value every 100 iterations (option emlog). The VCE iteration log displays iterations as dots (option vcedots). log, the default, implies emlog and vcedots. Use nolog to suppress both EM and VCE iteration logs, which is equivalent to specifying noemlog and novcedots. If log or nolog is specified, any other options that control an iteration log are ignored. log and nolog may not be combined with dots.

dots implies option emdots and vcedots to display both the EM and VCE iteration logs as dots. The VCE iteration log is always displayed as dots. However, the EM iteration log, by default, displays the log-likelihood value every 100 iterations (option emlog). To display the EM iterations as dots, you can specify dots or emdots. If dots is specified, any other options that control an iteration log are ignored. dots may not be combined with log and nolog.

vcedots, vcedots(#), and novcedots are synonyms for vce(, dots), vce(, dots(#)), and vce(, nodots), respectively.

emlog, emlog(#), and noemlog display or suppress an iteration log showing the progress of the EM algorithm. The log is displayed by default, and noemlog suppresses it; see set iterlog in [R] *set iter*. emlog, the default, displays the log-likelihood value every 100 iterations and is equivalent to emlog(100). emlog(#) displays the log-likelihood value every #th iterations.

noemdots, emdots(#), and emdots control the display of the EM iteration log as dots. By default, the EM iteration log displays the log-likelihood value every 100 iterations; that is, noemdots is implied. Instead, you can specify emdots to display every 100 iterations as a dot or emdots(#) to display every # iterations as a dot. This is a useful alternative for long EM iteration logs.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(*%fmt*), pformat(*%fmt*), sformat(*%fmt*), and nolstretch; see [R] **Estimation options**.

⌐ EM options ¬

emiterate(#), emtolerance(#), emltolerance(#), emhsgtolerance(#), noemhsgtolerance, and from(); see iterate(), tolerance(), ltolerance(), nrtolerance(), nonrtolerance(), and from() in [R] **Maximize**. These options control the EM optimization process. The defaults are emiterate(5000), emtolerance(1e-6), emltolerance(1e-7), and emhsgtolerance(1e-5).

The following option is available with stintcox but is not shown in the dialog box:

coeflegend; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

> *Introduction*
> *Single- versus multiple-record interval-censored data formats*
> *Case II interval-censored data*
> *Time-varying covariates*
> *Standard error estimation with interval-censored data*
> *Current status or case I interval-censored data*
> *Testing the proportional-hazards assumption using option tvc()*

Introduction

stintcox fits the Cox proportional hazards model to interval-censored survival-time data. In the context of interval-censored data, the term "failure-time data" or "event-time data" is more appropriate, so we will use it in that context.

Interval-censoring occurs when the failure time or the event time of interest is not exactly observed but is known only to lie within some interval. See *Introduction* in [ST] **stintreg** for details about interval-censored data. If you have right-censored data, see [ST] **stcox**. See [ST] **stintreg** for fitting parametric models to interval-censored data.

The Cox proportional hazards model was first introduced by Cox (1972) for right-censored survival data. For an introduction to interval-censored data, see Finkelstein and Wolfe (1985), Odell, Anderson, and D'Agostino (1992), Rabinowitz, Tsiatis, and Aragon (1995), Huang and Wellner (1997), Lindsey (1998), Lindsey and Ryan (1998), Sun (2006), and Sun and Li (2014).

The Cox proportional hazards model specifies that the hazard function of the event time conditional on a p-vector of baseline (time-invariant or time-independent) covariates $\mathbf{x} = (x_1, \ldots, x_p)$ takes the form

$$h(t; \mathbf{x}) = h_0(t) \exp(\beta_1 x_1 + \cdots + \beta_p x_p)$$

where $\beta_1, \ldots, \beta_p$ are unknown regression coefficients and $h_0(t)$ is an arbitrary baseline hazard function. Under the proportional-hazards assumption, the hazard ratios, or exponentiated regression coefficients $\exp(\beta_1), \ldots, \exp(\beta_p)$, are constant over time. As with right-censored data, the Cox proportional hazards model is appealing for interval-censored data because it does not require parameterization of the baseline hazard function and, for low event rates, the exponentiated regression parameters approximate the relative risks.

The partial-likelihood approach (Cox 1972, 1975) is used to estimate parameters of the Cox model with right-censored data, in which some of the event times are observed exactly, while others are known to be longer than the duration of follow-up. Under interval-censoring, however, none of the event times are observed exactly. Thus, it is much more challenging to deal with interval-censored data than right-censored data, both theoretically and computationally. In particular, the traditional partial-likelihood approach is not applicable.

Several authors (Cai and Betensky 2003; Zhang, Hua, and Huang 2010; Wang et al. 2016) have proposed spline methods to fit the Cox proportional hazards model to interval-censored data. Spline methods have limitations, however. First, the choices for the type of spline and the number and positions of knots are arbitrary, and different choices may yield conflicting results. Second, the analysis will be biased if the event-time distribution is not well approximated by the chosen spline function. Finally, the variance estimation is difficult given the data-dependent choices of spline functions (Zhang, Hua, and Huang 2010).

Direct maximum-likelihood optimization for the Cox model with interval-censored data using, for instance, the Newton–Raphson algorithm is highly unstable (Sun 2006; Finkelstein 1986).

Zeng, Mao, and Lin (2016) developed a novel EM algorithm for efficient nonparametric maximum-likelihood estimation (NPMLE) of the Cox proportional hazards model with interval-censored data. It allows a completely arbitrary event-time distribution and results in consistent, asymptotically normal, and asymptotically efficient estimators of the regression parameters. And it reduces to the classical maximum partial-likelihood estimation in the special case of right-censored data. For more details about this method, see *Methods and formulas*.

Unlike with right-censored data, the estimation of regression coefficients must be performed jointly with estimation of the baseline cumulative hazard function for interval-censored data. Stata provides two ways to estimate the baseline cumulative hazard function. One is to use all distinct lower and upper interval endpoints as time points for estimating the baseline cumulative hazard function. This is available by specifying the `full` option.

For large datasets with many distinct time points, this approach may become time consuming. An alternative is to estimate the baseline cumulative hazard at fewer time points. Turnbull (1976) proposed a method for estimating the one-sample survivor function at a subset of time intervals, known as Turnbull's intervals, or innermost intervals, or regions of the maximal cliques. Thus, one can allow the baseline cumulative hazard function to change its values only at the endpoints of those time intervals and set baseline hazard contributions to zero for the other times. This is available via the `reduced` option and, for computational reasons, is the default in `stintcox`.

As mentioned above, NPMLE is a computationally intensive approach, so `stintcox` may take some time to run, especially for large datasets. The speed of the command depends on the desired accuracy of the computations, among other things. The higher the accuracy, the more iterations are needed to achieve that accuracy, and thus the longer the command runs. It is important to have high accuracy for the final reporting of the results, but the speed may become an issue during the exploratory stage of the project. Thus, you may consider using the `favorspeed` option to expedite

the command execution. When you specify this option, `stintcox` uses less stringent convergence criteria to produce the results more quickly; see the description of option `favorspeed` for details.

Unlike many `st` commands, `stintcox` requires that you specify event-time ("survival") information directly with the command instead of using `stset`. Typing `stset` is unnecessary, and `stintcox` will ignore any settings of `stset` for the usual trivariate response variable (t_0, t, d). Event-time information can be specified with `stintcox` in two different ways depending on the storage format of the interval-censored event-time data. The two different storage formats are single-record-per-subject format (or time-intervals format) and multiple-record-per-subject format (or examination-times format). With single-record-per-subject data, you must specify two variables containing time intervals for each subject in `stintcox`'s `interval()` option. With multiple-record-per-subject data, you must specify subjects' identifiers in option `id()`, examination times in option `time()`, and event-status indicators at each examination time in option `status()`. See *Single- versus multiple-record interval-censored data formats* for details.

`stintcox` supports time-varying (time-dependent) covariates. You can use option `tvc()` to more easily include time-varying covariates that are formed by multiplying covariates specified in `tvc()` with a deterministic function of time specified in option `texp()`. In a multiple-record-per-subject format, you can include more general time-varying covariates; see *Single- versus multiple-record interval-censored data formats* and *Time-varying covariates*. In the presence of general time-varying covariates, `stintcox` offers several methods for imputing the values of these covariates between the examination times; see the description of option `tvcovimpute()` for details.

`stintcox` does not support data exhibiting delayed entry, gaps, and multiple failures.

Single- versus multiple-record interval-censored data formats

Interval-censored event-time data can be recorded in two different formats. In one format, the event-time information is recorded as interval data, with one record per subject containing lower and upper endpoints of the event-time interval. We call data stored in this format "single-record-per-subject interval-censored event-time data" or "single-record interval-censored data" for short.

In the other format, the event-time information is recorded by a pair of an examination time and an event status at that time. The dataset typically contains multiple records (multiple examination times) for a subject. We call data stored in this format "multiple-record-per-subject interval-censored event-time data" or "multiple-record interval-censored data". This format is common for what is called "case I interval-censored data or current status data" and is often used to accommodate time-varying covariates.

`stintcox` supports both formats.

Single-record interval-censored data. Consider the following dataset:

```
  id   ltime   rtime     x1     x2     x3
 101       0       6     17     22      0
 102       4       9     12     22      1
 103      13       .     13     22      0
```

Here variables `ltime` and `rtime` record the respective lower, t_l, and upper, t_u, endpoints of the event-time interval for each subject. `stintcox` requires that we specify these variables in the `interval()` option:

```
. stintcox x1 x2 x3, interval(ltime rtime) ...
```

In this format, if the data are left-censored, the lower endpoint is zero and may be represented in t_l by either a missing value (.) or zero. If the data are right-censored, the upper endpoint is $+\infty$ and is represented in t_u by a missing value. Uncensored data are represented by the two endpoints that are

equal. If $0 < t_l < t_u < \infty$, the data are interval-censored. Truly missing values must be represented by missing values in both t_l and t_u or by a 0 in t_l and a missing value in t_u. Uncensored observations, with $t_l = t_u$, are not allowed in the presence of left-censored or interval-censored observations.

In our example, subject 101 is left-censored, subject 102 is interval-censored, and subject 103 is right-censored.

This format is convenient for storing data containing interval-censored observations and covariates that are constant over time.

Multiple-record interval-censored data. We can provide the same event-time information as in the previous dataset in the following format:

```
id   time status    x1     x2     x3
101      6      1    17     22      0
102      4      0    12     22      1
102      9      1    12     22      1
103     13      0    13     22      0
```

Here variable `time` records examination times, and variable `status` records the event status indicator for each examination time. And subjects may have multiple examination times. In this format, `stintcox` requires that we specify the subject identifier (`id`) in option `id()`, examination time (`time`) in option `time()`, and the event status indicator (`status`) in option `status()`.

```
. stintcox x1 x2 x3, id(id) time(time) status(status) ...
```

Examination times and event status at each examination time can be used to determine the censoring type and the event-time interval for each subject. In our example, subject 101 has only one examination time, 6, and the event has already occurred by that time. Subject 101 is thus left-censored with the time interval $(t_l, t_u] = (0, 6]$. Subject 102 has two examination times, 4 and 9, and the event occurred by time 9. Subject 102 is interval-censored with the time interval $(4, 9]$. Subject 103 has one examination at time 13, and the event has not occurred by that time. This subject is right-censored with the time interval $(13, +\infty)$. Uncensored observations are not allowed in this format.

In this format, we can easily record time-varying covariates. For instance, suppose that subject 102 had an additional examination at time 6 with a status of 0 and the value for covariate `x3` was recorded to be 0. `x3` is no longer constant within subject 102 but varies discretely with time. (The event-time interval for this subject also changed to $(6, 9]$! More about this below.) We can accommodate varying `x3` by adding an extra record for subject 102 at time 6:

```
id   time status    x1     x2     x3
101      6      1    17     22      0
102      4      0    12     22      1
102      6      0    12     22      0
102      9      1    12     22      1
103     13      0    13     22      0
```

Potential caveat when creating time-varying covariates with interval-censored data. In the above, when we added an extra record for subject 102 at time 6 with status equal 0, we modified the actual event-time interval for this subject from the original $(4, 9]$ to $(6, 9]$. This is not a problem if this new record we added corresponds to the actual examination time for this subject. Sometimes, however, we may need to add extra records to specify varying covariate values at intermediate times without changing the event-time information for subjects. This is useful, for instance, when we create more complicated functions of time and covariates to assess the proportional-hazards assumption.

Returning to our previous example, how can we incorporate the varying values of covariate `x3` without changing the event-time interval for subject 102? We can replace the zero status with a system missing value (`.`) in the new observation and use the `statussysmissok` option with `stintcox` to

specify that this observation should be included in the estimation but should not be used to determine the subject's event-time interval and censoring information:

```
id    time status    x1    x2    x3
101      6      1    17    22     0
102      4      0    12    22     1
102      6      .    12    22     0
102      9      1    12    22     1
103     13      0    13    22     0
```

and

```
. stintcox x1 x2 x3, id(id) time(time) status(status) statussysmissok ...
```

Without the `statussysmissok` option, the record with a system missing status value would have been omitted from the analysis entirely, like any missing value in any of the specified variables. If you need to omit any values in the `status()` variable from your analysis in the presence of `statussysmissok`, you should record them as extended system missing values (`.a` through `.z`).

For an example of fitting `stintcox` to multiple-record interval-censored data, see *Time-varying covariates*. The multiple-record-per-subject format is also convenient for fitting current status data; see *Current status or case I interval-censored data*.

Case II interval-censored data

Case II interval-censored data arise when there are potentially two or more examination times for each study subject. With baseline covariates, case II interval-censored data are typically recorded in a single-record-per-subject format. In the presence of time-varying covariates in the dataset, a multiple-record-per-subject format is used; see *Time-varying covariates*.

In a single-record-per-subject format, the interval that brackets the event time of interest, the event-time interval, is recorded for each subject. The event of interest may occur before the first examination time, resulting in a left-censored observation; after the last examination time, resulting in a right-censored observation; or between two examination times, resulting in a truly interval-censored observation.

▷ Example 1: Single-record-per-subject case II interval-censored data

Zeng, Mao, and Lin (2016) considered a cohort study of injecting drug users in Thailand. Subjects were initially seronegative for the HIV-1 virus. They were followed and assessed for HIV-1 seropositivity through blood tests approximately every four months. The event of interest was time to HIV-1 seropositivity. Because the subjects were tested approximately every four months, the exact time of HIV-1 seropositivity was not observed but was known to fall only in the interval between blood tests.

The data used in this example are the data provided in supplementary materials of Zeng, Mao, and Lin (2016), which are based on the study described above. The dataset contains 1,124 subjects: 76 are females and 1,048 are males. We wish to identify the factors that influence HIV-1 infection. The covariates that we are interested in are age at recruitment (`age`), sex (`male`), history of needle sharing (`needle`), history of drug injection before recruitment (`inject`), and whether a subject has been in jail at the time of recruitment (`jail`). The dataset also contains two variables, `ltime` and `rtime`, that record, respectively, the last time of blood test when the HIV-1 was seronegative and the first time of blood test when the HIV-1 was seropositive.

```
. use https://www.stata-press.com/data/r18/idu
(Modified Bangkok IDU Preparatory Study)

. describe

Contains data from https://www.stata-press.com/data/r18/idu.dta
 Observations:           1,124                  Modified Bangkok IDU Preparatory
                                                  Study
     Variables:              8                  15 Dec 2022 13:34
                                                (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
age	byte	%8.0g		Age (in years)
male	byte	%8.0g	yesno	Male
needle	byte	%8.0g	yesno	Shared needles
jail	byte	%8.0g	yesno	Imprisoned at recruitment
inject	byte	%8.0g	yesno	Injected drugs before recruitment
ltime	double	%10.0g		Last time seronegative for HIV–1
rtime	double	%10.0g		First time seropositive for HIV–1
age_mean	double	%10.0g		Centered age (in years)

```
Sorted by:
```

We want to use stintcox to fit the Cox proportional hazards model in which the time to HIV-1 infection depends on age, male, needle, inject, and jail. To make the interpretation of the baseline hazard function more meaningful, we will use the centered age variable, age_mean. (Remember that a baseline hazard function corresponds to all covariates equal to zero, and age of zero would not make sense for our sample of subjects. See *Making baseline reasonable* in [ST] **stcox postestimation** for more details.)

Unlike stcox's specification, in which the survival variables are set using stset and do not appear in the command, for single-record-per-subject, the interval time variables ltime and rtime must be specified in the stintcox's option interval(). Also, recall that the command implements an NPMLE method, which is computationally intensive, so it may take a little longer to run on our dataset of 1,124 observations:

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -1086.2564
Iteration 100: Log likelihood = -597.65634
Iteration 200: Log likelihood = -597.57555
Iteration 295: Log likelihood = -597.56443

Computing standard errors: .......................... done
```

Interval-censored Cox regression					Number of obs	=	1,124
Baseline hazard: Reduced intervals					Uncensored =		0
					Left-censored =		41
Event-time interval:					Right-censored =		991
Lower endpoint: ltime					Interval-cens. =		92
Upper endpoint: rtime							
					Wald chi2(5)	=	17.10
Log likelihood = -597.56443					Prob > chi2	=	0.0043

	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf. interval]	
age_mean	.9684341	.0126552	-2.45	0.014	.9439452	.9935582
male						
Yes	.6846949	.1855907	-1.40	0.162	.4025073	1.164717
needle						
Yes	1.275912	.2279038	1.36	0.173	.8990401	1.810768
inject						
Yes	1.250154	.2414221	1.16	0.248	.8562184	1.825334
jail						
Yes	1.567244	.3473972	2.03	0.043	1.014982	2.419998

Note: Standard error estimates may be more variable for small datasets and
      datasets with low proportions of interval-censored observations.

The header above the coefficient table summarizes censored observations. Because this is a single-record-per-subject dataset, the number of observations equals the number of subjects. There are 991 subjects who did not test positive for HIV-1 by the last visit, resulting in right-censored observations. There are 41 subjects who tested positive for HIV-1 at their first follow-up, resulting in left-censored observations. The remaining 92 subjects are interval-censored.

As seen in the table, age is associated with lower risk of HIV-1 infection, and being in jail at enrollment is associated with higher risk of HIV-1 infection. Being a female, needle sharing, and history of drug injection are associated with the higher risk of HIV-1 infection, although we do not have strong evidence that the hazard ratios for these variables are different from 1.

The command displays a note following the command that an adaptive step size is used to compute derivatives during VCE computation. The command also displays a note following the output table about potential variability of the standard error estimates. We address all of this in more detail in *Standard error estimation with interval-censored data*.

stintcox uses the EM algorithm to estimate parameters. EM algorithms are known to require many iterations. Thus, by default, `stintcox` displays log-likelihood values every 100 iterations. You can change how often to display the iteration log by specifying the `emlog(#)` option. For example, `emlog(1)` will display every iteration. You can also use the `noemlog` option to suppress the iteration log.

Similarly, the progress of the VCE computation is displayed with a dot for each iteration. With more regression coefficients and with larger datasets, you may see many more dots. In this case, you may consider displaying a dot every # iterations by specifying vcedots(#). To suppress the dots, use novcedots.

◁

▷ Example 2: Speed versus accuracy

By default, stintcox uses Stata's standard convergence rules for estimation of parameters. For instance, the parameter tolerance of 1e–6 is used as a stopping rule, and the Hessian scale-gradient tolerance of 1e–5 is used to check for convergence. More stringent criteria typically require more iterations and thus lead to longer execution times.

Although high accuracy of the results is important for final reporting, it may be reasonable to consider less stringent criteria during exploratory analysis in favor of speed. stintcox provides the favorspeed option for this.

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
> favorspeed
note: using fixed step size with a multiplier of 5 to compute derivatives.
note: using EM and VCE tolerances of 0.0001.
note: option noemhsgtolerance assumed.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -1086.2564
Iteration 32:  Log likelihood = -598.60293

Computing standard errors: ........... done
```

Interval-censored Cox regression	Number of obs	=	1,124
Baseline hazard: Reduced intervals	Uncensored =		0
	Left-censored =		41
	Right-censored =		991
Event-time interval:	Interval-cens. =		92
Lower endpoint: ltime			
Upper endpoint: rtime			
	Wald chi2(5)	=	16.95
Log likelihood = -598.60293	Prob > chi2	=	0.0046

	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf. interval]	
age_mean	.9684228	.0126481	-2.46	0.014	.9439476	.9935326
male						
Yes	.6853044	.1873617	-1.38	0.167	.4010197	1.17112
needle						
Yes	1.275045	.2272609	1.36	0.173	.899103	1.80818
inject						
Yes	1.251637	.2414583	1.16	0.245	.8575707	1.826784
jail						
Yes	1.566873	.3479667	2.02	0.043	1.013914	2.421398

Note: Standard error estimates may be more variable for small datasets and datasets with low proportions of interval-censored observations.

The above takes only a few seconds to run. And the results happen to be very similar to the results from the previous two examples.

Following the command specification, we now see additional notes about the updated convergence criteria. The EM and VCE tolerances are now 0.0001 compared with the defaults of 1e–6. The tolerance check for the Hessian scaled gradient is suppressed. And a fixed step size with the multiplier of 5 is used to compute derivatives during the VCE computation.

◁

▷ Example 3: Full versus reduced sets of time intervals

By default, stintcox uses a reduced set of intervals (option reduced) to estimate the baseline hazard function. You can specify the full option to estimate the baseline hazard using all time intervals. This approach may be much more time consuming, especially for large datasets. For demonstration purposes, we will also use the favorspeed option from example 2 to speed up execution.

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime) full
> favorspeed
note: using fixed step size with a multiplier of 5 to compute derivatives.
note: using EM and VCE tolerances of 0.0001.
note: option noemhsgtolerance assumed.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -951.11659
Iteration 42:  Log likelihood = -599.05659

Computing standard errors: ........... done
```

```
Interval-censored Cox regression              Number of obs    =    1,124
Baseline hazard: All intervals                      Uncensored =        0
                                                Left-censored =       41
Event-time interval:                            Right-censored =      991
  Lower endpoint: ltime                         Interval-cens. =       92
  Upper endpoint: rtime
                                                Wald chi2(5)     =    16.94
Log likelihood = -599.05659                     Prob > chi2      =   0.0046
```

	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf. interval]	
age_mean	.9685754	.0126454	-2.45	0.014	.9441053	.9936797
male Yes	.6845339	.1871941	-1.39	0.166	.4005194	1.169947
needle Yes	1.276357	.2276023	1.37	0.171	.8998792	1.810339
inject Yes	1.252501	.2416295	1.17	0.243	.8581566	1.828058
jail Yes	1.566614	.3479196	2.02	0.043	1.013734	2.42103

Note: Standard error estimates may be more variable for small datasets and
      datasets with low proportions of interval-censored observations.

The results are very similar between the two ways of estimating the baseline hazard in this example. And EM optimization iterations change from 32 to 42 when the full option is specified.

◁

Time-varying covariates

There are two ways to incorporate time-varying covariates with `stintcox`. If a time-varying covariate can be represented as an interaction between a baseline covariate and a deterministic function of time, you can use the `tvc()` and `texp()` options to include such a covariate in the model. You can do this with either single-record or multiple-record interval-censored data. When you specify `tvc()`, `stintcox` provides a more memory-efficient computation and automatically incorporates the specified function of time in various calculations during estimation and postestimation.

But you may already have data in which some variables vary with time. These data will have multiple examination times for some subjects. You can use `stintcox`'s multiple-record-per-subject syntax to analyze data with existing time-varying covariates. And you can still use the `tvc()` option to additionally include time-varying covariates that are deterministic functions of time by specifying baseline or existing time-varying variables in that option.

▷ Example 4: Multiple-record-per-subject case II interval-censored data

The dataset used in this example is an extended version of the dataset described in example 1; it has the imprisonment indicator (`jail_vary`) that varies with time. The data are recorded in the multiple-record-per-subject format, where each observation records a subject identifier (`id`), the examination time of the blood test (`time`), whether the blood test changes to positive for HIV-1 since last examination time (`is_seropos`), and whether the subject has been imprisoned since the last clinic visit (`jail_vary`), which is a time-varying covariate. Each observation also records baseline (time-invariant) factors, such as centered age at recruitment (`age_mean`) and sex (`male`), as described in example 1.

Here is a subset of the dataset:

```
. use https://www.stata-press.com/data/r18/idu2
(Modified Bangkok IDU Preparatory Study with time-varying variable jail_vary)

. format time age_mean %6.2f

. list id time is_seropos age_mean male needle inject jail_vary
> if id >= 271 & id <= 274, sepby(id) noobs compress abbreviate(10)
```

id	time	is_seropos	age_mean	male	needle	inject	jail_vary
271	4.89	No	-6.46	Yes	Yes	No	No
271	9.31	No	-6.46	Yes	Yes	No	No
271	13.38	No	-6.46	Yes	Yes	No	Yes
271	17.97	No	-6.46	Yes	Yes	No	Yes
271	22.00	No	-6.46	Yes	Yes	No	No
272	3.80	No	8.54	No	No	No	Yes
272	9.41	Yes	8.54	No	No	No	No
273	3.93	No	-11.46	Yes	Yes	No	No
273	8.00	No	-11.46	Yes	Yes	No	No
273	12.07	No	-11.46	Yes	Yes	No	Yes
273	15.97	No	-11.46	Yes	Yes	No	Yes
273	20.66	No	-11.46	Yes	Yes	No	Yes
274	3.87	Yes	-4.46	Yes	Yes	Yes	Yes

There are subjects with multiple records. The examination `time` and event status `is_seropos` define subjects' event-time intervals and censoring types. For example, subject 271 never tested positive for HIV-1 through all his examinations. He is right-censored with an event-time interval $(22, +\infty)$.

Subject 272 tested negative at her first clinic visit but then tested positive at her second clinic visit. She is interval-censored with an event-time interval $(3.80, 9.41]$. Like subject 271, subject 273 is right-censored with an event-time interval $(20.66, +\infty]$. Finally, subject 274 tested positive at his first clinic visit, so he is left-censored with an event-time interval $(0, 3.87]$.

We use stintcox to fit a Cox proportional hazards model in which the time to HIV-1 infection depends on baseline covariates age_mean, male, needle, inject, and time-varying covariate jail_vary. In a multiple-record-per-subject format, we must specify options id(), time(), and status() with stintcox.

```
. stintcox age_mean i.male i.needle i.inject i.jail_vary, id(id) time(time)
> status(is_seropos)
note: time-varying covariates detected in the data; using method nearleft to
      impute their values between examination times.
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -1086.2564
Iteration 100: Log likelihood = -598.45375
Iteration 200: Log likelihood = -598.35872
Iteration 285: Log likelihood = -598.34887

Computing standard errors: ...................................... done

Interval-censored Cox regression              Number of obs     =   6,453
Baseline hazard: Reduced intervals            Number of subjects =  1,124
                                                   Uncensored =        0
ID variable: id                                 Left-censored =       41
Examination time: time                         Right-censored =      991
Status indicator: is_seropos                   Interval-cens. =       92

                                              Wald chi2(5)      =   17.03
Log likelihood = -598.34887                   Prob > chi2       =  0.0044
```

time	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf. interval]	
age_mean	.9714605	.012757	-2.20	0.027	.9467762	.9967884
male						
Yes	.6678044	.1816576	-1.48	0.138	.3918353	1.138138
needle						
Yes	1.271409	.2275426	1.34	0.180	.8952546	1.805609
inject						
Yes	1.370672	.2575405	1.68	0.093	.9484142	1.980928
jail_vary						
Yes	1.440966	.2916178	1.81	0.071	.9691488	2.142481

```
Time varying: jail_vary
Note: Standard error estimates may be more variable for small datasets and
      datasets with low proportions of interval-censored observations.
```

There are 6,453 observations on 1,124 subjects, but the censoring information (as expected) is the same as in example 1. stintcox identified one time-varying covariate in this dataset—jail_vary.

Compared with example 1, after we account for time-varying imprisonment, the hazard ratio for drug injection inject increases from 1.25 to 1.37, but the effect of imprisonment decreases from 1.57 for baseline jail to 1.44 for time-varying jail_vary.

◁

Standard error estimation with interval-censored data

With interval-censored data, the NPMLE approach estimates regression coefficients jointly with the contributions to the baseline cumulative hazard function, which is infinite dimensional. The inverse of the entire information matrix for all the parameters, which is typically used to estimate standard errors, does not provide a valid estimator of the variance–covariance matrix in this case.

Murphy and van der Vaart (2000) and Zeng, Mao, and Lin (2016) propose to estimate the VCE for regression coefficients using the profile log likelihood, which is obtained by maximizing the likelihood by holding the regression coefficients fixed. Four different methods based on this profile log likelihood are offered to calculate the VCE for regression coefficients. The vce(opg) method, the default, uses the first-order numerical derivatives, whereas the vce(oim) method uses the second-order numerical derivatives. When there are sufficient data to estimate the second-order derivatives reliably, the OIM method will generally provide more accurate results. With small samples, however, Zeng, Gao, and Lin (2017) found that the OIM method may lead to a negative definite matrix of second-order derivatives, which is not invertible. In addition, vce(robust) provides a VCE that is robust to some types of misspecification, and vce(cluster *clustvar*) provides a VCE that allows for observations to be correlated within groups defined by *clustvar*.

For some datasets, the choice of the step size used to compute numerical derivatives may also affect the estimates. By default, stintcox uses an adaptive step size, stepsize(adaptive), which is calculated from the data and updated during the computation of numerical derivatives. Zeng, Mao, and Lin (2016) and Zeng, Gao, and Lin (2017) found a fixed ad hoc step size of $\delta_n = 5n^{-1/2}$ to work well in the examples they considered. You can specify the stepsize(fixed) option to use that step size. And you can also provide your own multiplier instead of the above 5 by specifying stepsize(fixed #).

When an adaptive step size is used, stintcox searches for an optimal step size to use to compute the numerical derivatives. In some cases, such as in the presence of covariates of different magnitudes, the search may lead to step sizes that are too large or too small such that the VCE matrix becomes close to being singular. In that case, you may consider trying a different search method, for example, vce(, search(bracket)), or using a fixed step size, vce(, stepsize(fixed)).

For small datasets or datasets with low proportions of interval-censored observations, the standard error estimates may be more variable between different estimation methods. In that case, you may want to compare several VCE estimation methods. stintcox provides the OPG, OIM, robust, and cluster–robust methods on replay so that you do not need to rerun the estimation command.

▷ Example 5: Variance estimation

Continuing with example 1, we note that the dataset contains only 92 interval-censored observations out of a total of 1,124 observations. Let's compare the OPG and OIM methods using both an adaptive and a fixed step size.

Let's refit our model using the default settings and save the estimation results for later comparisons.

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
(output omitted)
```

To save estimation results after stintcox, we must save the estimated baseline hazard contributions in a dataset before using estimates store because they are an integral part of the estimation results. This can be done after estimation, on replay, or we could have specified the saving() option during estimation above.

```
. stintcox, saving(basehc, replace)
note: file basehc.dta not found; file saved.
. estimates store opg_adapt
```

We do not need to refit the model to compute OPG and OIM VCEs. To compute OIM estimates using an adaptive step size, we simply type

```
. stintcox, vce(oim, post)
note: using adaptive step size to compute derivatives.
Computing standard errors: ............................................. done
Interval-censored Cox regression              Number of obs    =   1,124
Baseline hazard: Reduced intervals                 Uncensored  =       0
                                                 Left-censored  =      41
Event-time interval:                            Right-censored  =     991
  Lower endpoint: ltime                         Interval-cens.  =      92
  Upper endpoint: rtime
                                              Wald chi2(5)      =   17.10
Log likelihood = -597.56443                   Prob > chi2       =  0.0043
```

	Haz. ratio	OIM std. err.	z	P>|z|	[95% conf. interval]
age_mean	.9684341	.0142537	-2.18	0.029	.9408965	.9967776
male Yes	.6846949	.2175931	-1.19	0.233	.3672728	1.276455
needle Yes	1.275912	.2688546	1.16	0.248	.8442274	1.928334
inject Yes	1.250154	.3363611	0.83	0.407	.7378069	2.118284
jail Yes	1.567244	.4679699	1.50	0.132	.8729166	2.813847

Note: Standard error estimates may be more variable for small datasets and
      datasets with low proportions of interval-censored observations.

```
. estimates store oim_adapt
```

We also specified the post suboption with vce(oim) to store the oim estimates in e(V) and stored the updated estimation results.

We repeat the same steps using a fixed step size for oim,

```
. stintcox, vce(oim, stepsize(fixed) post)
(output omitted )
. estimates store oim_fixed
```

and for opg,

```
. stintcox, vce(opg, stepsize(fixed) post)
(output omitted )
. estimates store opg_fixed
```

We compare the results using `estimates table`:

```
. estimates table opg* oim*, b(%9.4f) se(%9.4f) t p
```

Variable	opg_adapt	opg_fixed	oim_adapt	oim_fixed
age_mean	-0.0321	-0.0321	-0.0321	-0.0321
	0.0131	0.0131	0.0147	0.0120
	-2.45	-2.46	-2.18	-2.67
	0.0141	0.0141	0.0293	0.0077
male				
Yes	-0.3788	-0.3788	-0.3788	-0.3788
	0.2711	0.2736	0.3178	0.2994
	-1.40	-1.38	-1.19	-1.27
	0.1623	0.1662	0.2333	0.2058
needle				
Yes	0.2437	0.2437	0.2437	0.2437
	0.1786	0.1784	0.2107	0.1824
	1.36	1.37	1.16	1.34
	0.1725	0.1719	0.2475	0.1817
inject				
Yes	0.2233	0.2233	0.2233	0.2233
	0.1931	0.1933	0.2691	0.1961
	1.16	1.16	0.83	1.14
	0.2476	0.2480	0.4066	0.2548
jail				
Yes	0.4493	0.4493	0.4493	0.4493
	0.2217	0.2221	0.2986	0.2379
	2.03	2.02	1.50	1.89
	0.0427	0.0431	0.1324	0.0589

Legend: b/se/t/p

As expected, the regression coefficient estimates are the same for all VCE methods. The standard error estimates are fairly similar across all methods but a little more variable for `oim_adapt`. This is not surprising because the OIM method is based on the second-order derivatives, which are estimated numerically and thus require higher tolerances to produce more accurate estimates. For instance, we can specify a slightly lower tolerance (1e–7 instead of the default 1e–6) for the `oim` method using an adaptive step size:

```
. stintcox, vce(oim, tolerance(1e-7)) nohr
note: using adaptive step size to compute derivatives.
Computing standard errors: ....................................................
> ..... done
```

Interval-censored Cox regression   Number of obs   =   1,124
Baseline hazard: Reduced intervals   Uncensored =   0
                                                     Left-censored =   41
Event-time interval: Right-censored =   991
  Lower endpoint: ltime   Interval-cens. =   92
  Upper endpoint: rtime
                                                    Wald chi2(5)   =   17.10
Log likelihood = -597.56443   Prob > chi2   = 0.0043

	Coefficient	OIM std. err.	z	P>|z|	[95% conf. interval]	
age_mean	-.0320749	.012257	-2.62	0.009	-.0560982	-.0080516
male Yes	-.378782	.288278	-1.31	0.189	-.9437965	.1862326
needle Yes	.2436616	.1848383	1.32	0.187	-.1186149	.6059381
inject Yes	.2232666	.1981754	1.13	0.260	-.1651501	.6116832
jail Yes	.4493186	.2299995	1.95	0.051	-.0014721	.9001094

Note: Standard error estimates may be more variable for small datasets and
      datasets with low proportions of interval-censored observations.

The standard errors are now closer to those of the other methods.

◁

Current status or case I interval-censored data

Current status data or case I interval-censored data arise when each study subject is examined only once, such that the event of interest is known to occur before or after the examination time, resulting in a left- or right-censored observation. These data can be viewed as a special case of interval-censored data without truly interval-censored observations. Current status data can be recorded in either single-record-per-subject format or multiple-record-per-subject format, but the latter is more common.

▷ Example 6: Case I interval-censoring

Sun (2006) investigated a dataset about calcification of the hydrogel intraocular lenses (IOL), a rarely reported complication of cataract treatment. The dataset contains 379 patients who had IOL implantation and were examined by an ophthalmologist. The status variable indicates the degree of severity of IOL calcification: 0 means no or little calcification, and 1 means mild or serious calcification. The study contains 237 females and 142 males. We want to test whether the IOL calcification is different between males and females.

Current status data usually contain two variables: one that records the examination time (time) and one that records the status of the event of interest (status). So we can analyze these data using the multiple-record-per-subject format, even though each subject has only one observation.

Alternatively, we can create two interval time variables based on examination times and event statuses and analyze the data using the single-record-per-subject format. We already have the respective interval variables, ltime and rtime, in our dataset, but see example 3 of [ST] **stintreg** on how to generate these variables.

In the following, we will demonstrate how to analyze current status data using both formats.

Let us fit a Cox proportional hazards model on gender using the multiple-record-per-subject format first:

```
. use https://www.stata-press.com/data/r18/iol
(Hydrogel Intraocular Lenses (IOL) Study)

. stintcox i.gender, id(id) time(time) status(status) nohr
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood =  -191.3847
Iteration 100: Log likelihood = -137.70653
Iteration 200: Log likelihood = -137.65739
Iteration 300: Log likelihood = -137.64799
Iteration 400: Log likelihood = -137.64509
Iteration 425: Log likelihood = -137.64472

Computing standard errors: ........... done
```

Interval-censored Cox regression				Number of obs	=	379
Baseline hazard: Reduced intervals				Number of subjects =		379
				Uncensored =		0
ID variable: id				Left-censored =		48
Examination time: time				Right-censored =		331
Status indicator: status				Interval-cens. =		0
				Wald chi2(1)	=	0.52
Log likelihood = -137.64472				Prob > chi2	=	0.4719

		OPG				
time	Coefficient	std. err.	z	P>|z|	[95% conf. interval]	
gender						
Male	-.2242553	.3117387	-0.72	0.472	-.835252	.3867415

Note: Standard error estimates may be more variable for small datasets and datasets with low proportions of interval-censored observations.

We specified the nohr option to report the regression coefficient estimates instead of the hazard ratios for the independent variables to more easily compare results from the literature.

The above table reports $\widehat{\beta} = -0.2243$ and its estimated standard error of 0.312. This yields a test of $\beta = 0$ with a p-value of 0.47, which suggests that there is no difference between males and females in terms of the time to IOL calcification.

Sun (2006) reports $\widehat{\beta} = -0.2241$ with its standard error of 0.295, which are obtained by direct maximum-likelihood optimization. Our EM-based estimates are comparable.

Let's fit the same model but now use the single-record-per-subject format, where we specify the event-time intervals ltime and rtime in the interval() option:

```
. stintcox i.gender, interval(ltime rtime) nohr
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):
Iteration 0:    Log likelihood =  -191.3847
Iteration 100: Log likelihood = -137.70653
Iteration 200: Log likelihood = -137.65739
Iteration 300: Log likelihood = -137.64799
Iteration 400: Log likelihood = -137.64509
Iteration 425: Log likelihood = -137.64472

Computing standard errors: .......... done
```

```
Interval-censored Cox regression              Number of obs     =      379
Baseline hazard: Reduced intervals                 Uncensored =        0
                                                Left-censored =       48
Event-time interval:                           Right-censored =      331
  Lower endpoint: ltime                         Interval-cens. =        0
  Upper endpoint: rtime
                                                Wald chi2(1)      =     0.52
Log likelihood = -137.64472                     Prob > chi2       =   0.4719
```

	Coefficient	OPG std. err.	z	P>|z|	[95% conf. interval]
gender					
Male	-.2242553	.3117387	-0.72	0.472	-.835252 .3867415

Note: Standard error estimates may be more variable for small datasets and
      datasets with low proportions of interval-censored observations.

We obtained the same results in the multiple-record-per-subject format.

◁

Testing the proportional-hazards assumption using option tvc()

One way of testing the proportional-hazards assumption for a covariate is to test whether the coefficient associated with that covariate is time invariant. This can be accomplished by including an interaction between that covariate and a function of time in the model and testing whether the corresponding coefficient equals zero. For instance, consider a model with one baseline covariate x_1. We wish to include an interaction between the covariate and a function of time $g(t)$:

$$h(t) = h_0(t) \exp\{\beta_1 x_1 + g(t)\gamma_1 x_1\}$$

Rearranging terms results in

$$h(t) = h_0(t) \exp\left[\{\beta_1 + \gamma_1 g(t)\} x_1\right]$$

Given this new arrangement, we consider that $\beta_1 + \gamma_1 g(t)$ is a (possibly) time-varying coefficient on the covariate x_1 for some specified function of time $g(t)$. The coefficient has a time-invariant component, β_1, with γ_1 determining the magnitude of the time-varying deviations from β_1. Thus, a test of $\gamma_1 = 0$ is a test of time invariance for the coefficient on x_1. Proportional hazards imply that the relative hazard is fixed over time, and this assumption would be violated if a time interaction proved significant.

The above can be easily accomplished by using the tvc() option with stintcox, where you specify the covariates suspected to violate the proportional-hazards assumption. By default, $g(t) = t$ is used, but you can use the texp() option to specify other functions of time.

▷ Example 7: Testing the proportional-hazards assumption

Continuing with example 1, we now include time interaction terms for all covariates to test the proportional-hazards assumption for individual covariates and globally:

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
> tvc(age_mean i.male i.needle i.inject i.jail) nohr
note: using adaptive step size to compute derivatives.
Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -1086.2564
Iteration 100: Log likelihood = -590.53655
Iteration 200: Log likelihood = -590.45163
Iteration 300: Log likelihood = -590.43665
Iteration 340: Log likelihood = -590.43386
Computing standard errors: ............................................. done
```

Interval-censored Cox regression	Number of obs	=	1,124
Baseline hazard: Reduced intervals	Uncensored =		0
	Left-censored =		41
Event-time interval:	Right-censored =		991
Lower endpoint: ltime	Interval-cens. =		92
Upper endpoint: rtime			
	Wald chi2(10)	=	31.99
Log likelihood = -590.43386	Prob > chi2	=	0.0004

	Coefficient	OPG std. err.	z	P>|z|	[95% conf. interval]	
main						
age_mean	-.0310177	.0233817	-1.33	0.185	-.076845	.0148097
male						
Yes	-1.271583	.4604788	-2.76	0.006	-2.174105	-.3690615
needle						
Yes	-.1819587	.3297493	-0.55	0.581	-.8282554	.464338
inject						
Yes	.6852961	.3431924	2.00	0.046	.0126513	1.357941
jail						
Yes	-.529615	.4021087	-1.32	0.188	-1.317734	.2585036
tvc						
age_mean	-.000129	.0017099	-0.08	0.940	-.0034804	.0032224
male						
Yes	.0884102	.042994	2.06	0.040	.0041434	.1726769
needle						
Yes	.0358545	.0238562	1.50	0.133	-.0109027	.0826118
inject						
Yes	-.0361192	.0228754	-1.58	0.114	-.0809541	.0087157
jail						
Yes	.0916036	.0348915	2.63	0.009	.0232176	.1599896

Notes: Standard error estimates may be more variable for small datasets and
       datasets with low proportions of interval-censored observations.
       Variables in **tvc** equation interacted with _t.

Wald test that [**tvc**] = 0: chi2(5) = 13.3282   Prob > chi2 = 0.0205

The first equation, `main`, reports the coefficients for the covariates that do not vary over time; the second equation, `tvc`, reports the results for covariates interacted with time. We used the default function of time, $g(t) = t$, although we could have specified other functions with the `texp()` option.

Without the `nohr` option, the table would present results as hazard ratios for the two equations; exponentiated coefficients can be interpreted as hazard ratios when we are actually modeling time-varying covariates. However, in this case, we are using the `tvc()` option to incorporate time-varying coefficients; in our model, the coefficient for covariate k is expressed as $b_k + \gamma_k \times t$, where b_k corresponds to the parameter in the `main` equation and γ_k corresponds to the parameter in the `tvc` equation. The proportional-hazards assumption does not hold unless $\gamma_k = 0$.

There is little evidence to dispute the proportionality of hazards for the covariates `age_mean`, `needle`, and `inject` with respect to this specific form of misspecification. But the proportional-hazards assumption appears to be violated for covariates `male` and `jail`.

At the bottom of the table, `stintcox` displays results for a Wald test for the global proportional-hazards assumption (that is, the null hypothesis is that all the coefficients in the `tvc` equation are equal to zero). Alternatively, you can specify the `lrphtest` option to perform the likelihood-ratio test between the full model and the model without the time interaction terms:

```
. stintcox age_mean i.male i.needle i.inject i.jail, interval(ltime rtime)
> tvc(age_mean i.male i.needle i.inject i.jail) lrphtest nohr
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):

Fitting main model:
Iteration 0:    Log likelihood = -1086.2564
Iteration 100: Log likelihood = -597.65634
Iteration 200: Log likelihood = -597.57555
Iteration 295: Log likelihood = -597.56443

Fitting full model:
Iteration 0:    Log likelihood = -1086.2564
Iteration 100: Log likelihood = -590.53655
Iteration 200: Log likelihood = -590.45163
Iteration 300: Log likelihood = -590.43665
Iteration 340: Log likelihood = -590.43386

Computing standard errors: ............................................. done
```

```
Interval-censored Cox regression                    Number of obs     =   1,124
Baseline hazard: Reduced intervals                      Uncensored =       0
                                                     Left-censored =      41
Event-time interval:                                Right-censored =     991
  Lower endpoint: ltime                              Interval-cens. =      92
  Upper endpoint: rtime
                                                     Wald chi2(10)     =   31.99
Log likelihood = -590.43386                          Prob > chi2       =  0.0004
```

	Coefficient	OPG std. err.	z	P>|z|	[95% conf. interval]	
main						
age_mean	-.0310177	.0233817	-1.33	0.185	-.076845	.0148097
male						
Yes	-1.271583	.4604788	-2.76	0.006	-2.174105	-.3690615
needle						
Yes	-.1819587	.3297493	-0.55	0.581	-.8282554	.464338
inject						
Yes	.6852961	.3431924	2.00	0.046	.0126513	1.357941
jail						
Yes	-.529615	.4021087	-1.32	0.188	-1.317734	.2585036
tvc						
age_mean	-.000129	.0017099	-0.08	0.940	-.0034804	.0032224
male						
Yes	.0884102	.042994	2.06	0.040	.0041434	.1726769
needle						
Yes	.0358545	.0238562	1.50	0.133	-.0109027	.0826118
inject						
Yes	-.0361192	.0228754	-1.58	0.114	-.0809541	.0087157
jail						
Yes	.0916036	.0348915	2.63	0.009	.0232176	.1599896

```
Notes: Standard error estimates may be more variable for small datasets and
       datasets with low proportions of interval-censored observations.
       Variables in tvc equation interacted with _t.
LR test that [tvc] = 0: chi2(5) = 14.2611                Prob > chi2 = 0.0140
```

The results of the likelihood-ratio test agree with the previous Wald test—the proportional-hazards assumption has been violated globally. The likelihood-ratio test may be more stable for sample sizes and when the sampling distribution of regression coefficients is not symmetric.

Specifying the lrphtest option can substantially increase computational time because both the full model and the reduced main model must be fit.

◁

Stored results

stintcox stores the following in e():

Scalars

e(N)	number of observations
e(N_sub)	number of subjects
e(N_unc)	number of uncensored subjects
e(N_lc)	number of left-censored subjects
e(N_rc)	number of right-censored subjects
e(N_int)	number of interval-censored subjects
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_dv)	number of dependent variables
e(df_m)	model degrees of freedom
e(df_tvc)	degrees of freedom for proportional-hazards test
e(ll)	log likelihood
e(ll_0)	log likelihood, constant-only model
e(ll_c)	log likelihood, main model
e(N_clust)	number of clusters
e(chi2)	χ^2
e(chi2_tvc)	χ^2 for proportional-hazards test
e(p)	p-value for model test
e(p_tvc)	p-value for proportional-hazards test
e(rank)	rank of e(V)
e(ic)	number of iterations
e(converged)	1 if converged, 0 otherwise
e(delta)	multiplier used with fixed step size
e(delta_n)	fixed step size
e(emiterate)	maximum EM iterations
e(emtolerance)	EM coefficient tolerance
e(emltolerance)	EM log-likelihood tolerance
e(emhsgtolerance)	EM scaled-gradient tolerance
e(noemhsgtolerance)	1 if noemhsgtolerance, 0 otherwise
e(vceiterate)	maximum VCE iterations
e(vcetolerance)	VCE log-likelihood tolerance

Macros

e(cmd)	stintcox
e(cmdline)	command as typed
e(depvar)	names of time interval variables specified in interval() or name of examination time variable specified in time()
e(id)	id variable specified in id()
e(status)	name of status variable specified in status()
e(strata)	strata variables
e(clustvar)	name of cluster variable
e(title)	title in estimation output
e(title2)	secondary title in estimation output
e(chi2type)	Wald or LR; type of model χ^2 test
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(tvc)	covariates interacted with time from option tvc()
e(texp)	function of time used for covariates from tvc()
e(phtest)	Wald or LR; type of proportional-hazards assumption test
e(tvvariables)	time-varying variables detected in the data
e(tvcovimpute)	imputation method used for variables in e(tvvariables)
e(intervals)	reduced or full

e(filename)	name of the file with estimated baseline hazard contributions
e(stepsize)	adaptive or fixed

Matrices

e(b)	coefficient vector
e(V)	variance–covariance matrix of the estimators

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

If the vce() option is specified on replay, the following estimation results will be updated only if vce()'s post suboption is specified: e(vce), e(vcetype), e(stepsize), e(delta) and e(delta_n) (with fixed step size) and e(chi2_tvc) and e(p_tvc) (with the tvc() option).

Methods and formulas

Methods and formulas are presented under the following headings:

> *Data and model*
> *EM algorithm for computing parameter estimates*
> *Variance estimation using the profile log-likelihood function*
> *Stratified estimation*
> *Option tvc()*

Data and model

For a comprehensive review of the methods in this entry, see Zeng, Mao, and Lin (2016).

Let T denote the event time, and let $\mathbf{x}(\cdot)$ denote a $1 \times p$ vector of covariates that can potentially depend on time. Under the Cox proportional hazards model, the hazard function of T conditional on $\mathbf{x}(\cdot)$ is

$$h(t; \mathbf{x}) = h_0(t) \exp\{\mathbf{x}(t)\boldsymbol{\beta}\}$$

where $\boldsymbol{\beta}$ is a $p \times 1$ vector of unknown regression parameters and $h_0(t)$ is an arbitrary baseline hazard function. Let $H_0(t) = \int_0^t h_0(s)ds$, which is the baseline cumulative hazard function.

The occurrence of an asymptomatic event can be detected only through periodic examinations. Let $(T_l, T_u]$ denote the shortest time interval that brackets T, with $T_l < T_u$. Left-censoring is indicated by $T_l = 0$, and right-censoring by $T_u = \infty$.

Consider a study with n subjects and N observations. In a single-record-per-subject format, $n = N$ and the data consist of $(t_{li}, t_{ui}, \mathbf{x}_i)$ for $i = 1, \ldots, n$, where $\mathbf{x}_i$ records covariate values for subject i and t_{li} and t_{ui} define the observed time interval.

In a multiple-record-per-subject format, $n < N$ (typically) and the data consist of $(t_{ij}, \delta_{ij}, \mathbf{x}_{ij})$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n_i$, where $N = \sum_{i=1}^n n_i$, t_{ij} is the examination time j for subject i, δ_{ij} denotes whether the event of interest occurs between $t_{i(j-1)}$ and t_{ij} ($t_{i(j-1)} = 0$ if $j = 1$), and $\mathbf{x}_{ij}$ records covariate values at t_{ij}. Here $\mathbf{x}_{ij}$ includes both baseline and time-varying covariates, where for baseline covariates $\mathbf{x}_{ij} = \mathbf{x}_i$. Pairs (t_{ij}, δ_{ij})'s are used to form the event-time

interval endpoints t_{li}'s and t_{ui}'s as follows. If an event occurs before the first examination time t_{i1} ($\delta_{i1} = 1$), then $t_{li} = 0$, $t_{ui} = t_{i1}$, and subject i is left-censored. If an event does not occur by the last examination time t_{in_i} ($\delta_{in_i} = 0$), then $t_{li} = t_{in_i}$, $t_{ui} = +\infty$, and subject i is right-censored. If, for $1 < j < n_i$, an event occurs between two examination times $t_{i(j-1)}$ and t_{ij} ($\delta_{ij} = 1$), then $t_{li} = t_{i(j-1)}$, $t_{ui} = t_{ij}$, and subject i is interval-censored.

The observed-data likelihood function for $\boldsymbol{\beta}$ and $H_0(t)$ is

$$L_n(\boldsymbol{\beta}, H_0) = \prod_{i=1}^n \left(\exp\left[-\int_0^{t_{li}} \exp\{\mathbf{x}_i(s)\boldsymbol{\beta}\}dH_0(s) \right] - \exp\left[-\int_0^{t_{ui}} \exp\{\mathbf{x}_i(s)\boldsymbol{\beta}\}dH_0(s) \right] \right)$$

where the integral is ∞ if $t_{ui} = \infty$ and $\mathbf{x}_i(\cdot)$ is a $1 \times p$ vector of potentially time-varying covariates for subject i. Under the NPMLE approach, H_0 is regarded as a step function with nonnegative jumps $h_1, \ldots, h_m$ at $t_1, \ldots, t_m$, respectively, where $t_1 < \cdots < t_m$ are the distinct time points for all $t_{li} > 0$ and $t_{ui} < \infty$ for $i = 1, \ldots, n$. Thus, we need to maximize the function

$$L_n(\boldsymbol{\beta}, \{h_k\}) = \prod_{i=1}^n \exp\left\{ -\sum_{t_k \leq t_{li}} h_k \exp(\mathbf{x}_{ik}^*\boldsymbol{\beta}) \right\} \left[1 - \exp\left\{ -\sum_{t_{li} < t_k \leq t_{ui}} h_k \exp(\mathbf{x}_{ik}^*\boldsymbol{\beta}) \right\} \right]^{I(t_{ui}<\infty)} \quad (1)$$

where $\mathbf{x}_{ik}^* = \mathbf{x}_i(t_k)$ are the covariate values for subject i at time t_k. For baseline covariates, $\mathbf{x}_{ik}^* = \mathbf{x}_i$. For time-varying covariates, if $t_k = t_{ij}$ is one of the examination times for subject i, then $\mathbf{x}_{ik}^* = \mathbf{x}_{ij}$. Otherwise, $\mathbf{x}_{ik}^*$ is imputed as described in the `tvcovimpute()` option in *Options*.

Direct maximization of (1) is difficult because of the lack of an analytic expression for the parameters h_k ($k = 1, \ldots, m$). And an even greater challenge is that not all t_{li} and t_{ui} are informative about the event times, so many h_k's are zeros and thus lie on the boundary of the parameter space.

To address these challenges, Zeng, Mao, and Lin (2016) construct some latent Poisson variables that yield the same observed-data likelihood as (1). They propose the EM algorithm, in which the E-step involves simple calculations and the M-step amounts to the maximization of a weighted sum of Poisson log-likelihood functions that is strictly concave and has a closed-form solution for h_k's.

Turnbull (1976) showed that the NPMLE of the survival distribution is unique only up to a set of intervals, which is called Turnbull's intervals (the innermost intervals or the regions of the maximal cliques). Therefore, for computational reasons, by default or if the `reduced` option is specified, we estimate the baseline cumulative hazard function at the endpoints of Turnbull's reduced set of intervals. Alternatively, you can specify the `full` option to estimate the baseline cumulative hazard function at the endpoints of all observed time intervals, which corresponds to the approach of Zeng, Mao, and Lin (2016).

EM algorithm for computing parameter estimates

Let W_{ik} ($i = 1, \ldots, n; k = 1, \ldots, m$) be independent latent Poisson random variables with means $h_k \exp(\mathbf{x}_{ik}^*\boldsymbol{\beta})$. Define $A_i = \sum_{t_k \leq t_{li}} W_{ik}$ and $B_i = I(t_{ui} < \infty) \sum_{t_{li} < t_k \leq t_{ui}} W_{ik}$. The likelihood for the observed data $\{t_{li}, t_{ui}, \mathbf{x}_i(\cdot), A_i = 0, B_i > 0\}$ ($i = 1, \ldots, n$) is

$$\prod_{i=1}^n \prod_{t_k \leq t_{li}} \Pr(W_{ik} = 0) \left\{ 1 - \Pr\left(\sum_{t_{li} < t_k \leq t_{ui}} W_{ik} = 0 \right) \right\}^{I(t_{ui}<\infty)} \quad (2)$$

which is exactly equal to (1). Thus, we can maximize (2) through an EM algorithm treating W_{ik} as missing data.

The complete-data log likelihood is

$$
\sum_{i=1}^{n} \sum_{k=1}^{m} I(t_k \leq t_{ui}^*) \Big[W_{ik} \log\{ h_k \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta}) \} - h_k \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta}) - \log W_{ik}! \Big]
$$

where $t_{ui}^* = I(t_{ui} < \infty) t_{ui} + I(t_{ui} = \infty) t_{li}$. In the E-step, we evaluate the posterior means of W_{ik} as

$$
w_{ik} = \begin{cases} \dfrac{h_k \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta})}{1 - \exp\{ -\sum_{t_{li} < t_j \leq t_{ui}} h_j \exp(\mathbf{x}_{ij}^* \boldsymbol{\beta}) \}} & \text{if } t_{li} < t_k \leq t_{ui} < \infty \\ 0 & \text{otherwise} \end{cases}
$$

In the M-step, we update $\boldsymbol{\beta}$ by solving the following equation via the one-step Newton–Raphson method,

$$
\sum_{i=1}^{n} \sum_{k=1}^{m} I(t_k \leq t_{ui}^*) \, w_{ik} \left\{ \mathbf{x}_{ik}^* - \frac{\sum_{j=1}^{n} I(t_k \leq t_{uj}^*) \exp(\mathbf{x}_{jk}^* \boldsymbol{\beta}) \mathbf{x}_{jk}^*}{\sum_{j=1}^{n} I(t_k \leq t_{uj}^*) \exp(\mathbf{x}_{jk}^* \boldsymbol{\beta})} \right\} = 0
$$

and then update h_k $(k = 1, \ldots, m)$ using

$$
h_k = \frac{\sum_{i=1}^{n} I(t_k \leq t_{ui}^*) w_{ik}}{\sum_{i=1}^{n} I(t_k \leq t_{ui}^*) \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta})} \tag{3}
$$

Setting the initial values of $\boldsymbol{\beta}$ to zeros and the initial values of h_k's to $1/m$, we iterate between the E- and M-steps until the desired convergence criteria are achieved. The convergence tolerances are described in detail in [R] **Maximize**, where `emtolerance()` is analogous to `tolerance()`, `emltolerance()` to `ltolerance()`, `hsgtolerance()` to `nrtolerance()`, and `nonrtolerance` to `nohsgtolerance`.

The observed-data log-likelihood function is calculated as

$$
\log L = \sum_{i=1}^{n} \log \Big\{ I(t_{li} < t_{ui})(S_{li} - S_{ui}) + I(t_{li} = t_{ui}) \sum_{k=1}^{m} I(t_{li} = t_k) \, h_k \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta}) S_{li} \Big\}
$$

where $S_{li} = \exp\{ -\sum_{t_k \leq t_{li}} h_k \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta}) \}$ and $S_{ui} = \exp\{ -\sum_{t_k \leq t_{ui}} h_k \exp(\mathbf{x}_{ik}^* \boldsymbol{\beta}) \} I(t_{ui} < \infty)$.

When there are no covariates, the above algorithm becomes iteratively updating h_k as

$$
h_k = \frac{\sum_{i=1}^{n} \big\{ h_k I(t_{li} < t_k \leq t_{ui} < \infty) S_{li}/(S_{li} - S_{ui}) + I(t_{li} = t_{ui} = t_k) \big\}}{\sum_{i=1}^{n} I(t_k \leq t_{ui}^*)}
$$

where $S_{li} = \exp\{ -\sum_{t_k \leq t_{li}} h_k \}$ and $S_{ui} = \exp\{ -\sum_{t_k \leq t_{ui}} h_k \} I(t_{ui} < \infty)$.

With multiple-record interval-censored data, because uncensored observations are not supported, the terms with $t_{li} = t_{ui}$ are ignored in the formulas above.

Variance estimation using the profile log-likelihood function

Denote the estimators of β and h_k's by $\widehat{\beta}$ and $\widehat{h}_k$'s, respectively, for $k = 1, \ldots, m$. The NPMLEs $\widehat{\beta}$ and $\widehat{H}_0(t) = \sum_{t_k \le t} \widehat{h}_k$ are strongly consistent, and $\widehat{\beta}$ is asymptotically normal and asymptotically efficient (Zeng, Mao, and Lin 2016). The covariance matrix of $\widehat{\beta}$ can be estimated using profile likelihood (Murphy and van der Vaart 2000; Zeng, Mao, and Lin 2016; Zeng, Gao, and Lin 2017).

The profile log-likelihood function for β takes the form

$$
pl(\beta) = \sum_{i=1}^{n} \log\left[\exp\left\{ -\sum_{t_k \le t_{li}} \widetilde{h}_k \exp(\mathbf{x}_{ik}^* \beta) \right\} - I(t_{iu} < \infty) \exp\left\{ -\sum_{t_k \le t_{ui}} \widetilde{h}_k \exp(\mathbf{x}_{ik}^* \beta) \right\} \right]
$$

where $\widetilde{h}_k$ $(k = 1, \ldots, m)$ are the maximizers of (1) for the given β. These maximizers are obtained from the EM algorithm described in the previous section with $\widehat{h}_k$ $(k = 1, \ldots, m)$ as the initial values and with β fixed over the iterations. Specifically, we apply the same EM algorithm but hold β fixed during the iterations. Thus, the only steps in the EM algorithm are to explicitly evaluate w_{ik} and to update h_k using (3).

Two likelihood-based methods, vce(opg) and vce(oim), are available to estimate the covariance matrix of $\widehat{\beta}$.

The oim method estimates the covariance matrix of $\widehat{\beta}$ by the negative inverse of the Hessian matrix, $-\{D^2 pl(\widehat{\beta})\}^{-1}$, where D^2 is the second-order numerical derivative and its (j, k)th element is

$$
\left[D^2 pl(\widehat{\beta}) \right]_{j,k} = \frac{pl(\widehat{\beta}) - pl(\widehat{\beta} + \delta_n \mathbf{e}_k) - pl(\widehat{\beta} + \delta_n \mathbf{e}_j) + pl(\widehat{\beta} + \delta_n \mathbf{e}_k + \delta_n \mathbf{e}_j)}{\delta_n^2}
$$

where $\mathbf{e}_j$ is the jth canonical vector in the space of β and δ_n is the step size chosen for numerical difference. Zeng, Gao, and Lin (2017) found that the above estimated matrix may be negative definite, especially in small samples. Therefore, they proposed the opg method, which estimates the covariance matrix using $\left[\sum_{i=1}^{n} \{Dpl_i(\widehat{\beta})\}^{\otimes 2} \right]^{-1}$, where D is the first-order numerical derivative and $\mathbf{a}^{\otimes 2} = \mathbf{a}\mathbf{a}'$. Specifically, the jth element of D is

$$
\left[Dpl_i(\widehat{\beta}) \right]_j = \frac{pl_i(\widehat{\beta} + \delta_n \mathbf{e}_j) - pl_i(\widehat{\beta})}{\delta_n}
$$

where $pl_i(\beta)$ is the contribution of subject i to $pl(\beta)$. The resulting covariance matrix estimator is guaranteed to be positive semidefinite and more robust with respect to the choice of the step size than the estimator based on the second-order numerical difference.

The stepsize() suboption of the vce() option offers different ways for you to choose the step size with the opg and oim methods when computing numerical derivatives. stepsize(adaptive), the default, uses an adaptive step size; see [M-5] **deriv( )**. stepsize(fixed) uses a fixed step size $\delta_n = 5n^{-1/2}$ (Zeng, Mao, and Lin 2016; Zeng, Gao, and Lin 2017). And stepsize(fixed #) uses a fixed step size equal to $\# \times n^{-1/2}$.

This command also supports the Huber/White/sandwich estimator of the variance and its clustered version using vce(robust) and vce(cluster *clustvar*), respectively. The sandwich estimator is calculated as $\{D^2 pl(\widehat{\beta})\}^{-1} \sum_{m=1}^{M} Dpl_m(\widehat{\beta})^{\otimes 2} \{D^2 pl(\widehat{\beta})\}^{-1}$, where $pl_m(\beta)$ is the contribution of the mth cluster to $pl(\beta)$ and M is the number of clusters. For datasets where option id(*idvar*) has been specified, option vce(robust) has been implemented as equivalent to vce(cluster *idvar*).

Stratified estimation

stintcox with the strata() option will produce a stratified proportional hazards model for interval-censored data. Assume that within stratum q $(q = 1, \ldots, Q)$, the hazard function of T is

$$h_q(t; \mathbf{x}) = h_{q0}(t) \exp\{\mathbf{x}(t)\boldsymbol{\beta}\}$$

where $h_{q0}(t)$ is an unknown baseline hazard function for stratum q.

Let S_i be the stratum for subject i, which can take value from $\{1, \ldots, Q\}$. Analogously to (1), we maximize the function

$$L_n(\boldsymbol{\beta}, \{h_{qk}\}) = \prod_{i=1}^{n} \prod_{q=1}^{Q} \exp\Big\{ -\sum_{t_{qk} \leq t_{li}} h_{qk} \exp(\mathbf{x}_{iqk}^* \boldsymbol{\beta}) \Big\}$$
$$\Big[1 - \exp\Big\{ -\sum_{t_{li} < t_{qk} \leq t_{ui}} h_{qk} \exp(\mathbf{x}_{iqk}^* \boldsymbol{\beta}) \Big\} \Big]^{I(t_{ui} < \infty)}$$

where $\mathbf{x}_{iqk}^* = \mathbf{x}_i(t_{qk})$. Within the NPMLE framework, a stratum-specific baseline cumulative hazard function $H_{q0}(t) = \int_0^t h_{q0}(s) \, ds$ for $q = 1, \ldots, Q$ is regarded as a step function, $\sum_{t_{qk} \leq t} h_{qk}$, with nonnegative jumps h_{qk} at the distinct values t_{qk}, ordered from smallest to largest, of the observed interval endpoints from all subjects in stratum q.

Option tvc()

Let $\mathbf{z}(\cdot)$ be a $1 \times q$ vector of potentially time-varying covariates specified in the tvc() option. Let $g(\cdot)$ be the function of time specified in the texp() option. By default, $g(t) = t$. Let $\mathbf{z}_i$'s and $\mathbf{z}_{ij}$'s be the observed covariate values for $\mathbf{z}(\cdot)$ in the respective single-record-per-subject and multiple-record-per-subject formats.

When the tvc() option is specified, we replace $\mathbf{x}_{ik}^* \boldsymbol{\beta}$ in all formulas above with

$$\mathbf{x}_{ik}^* \boldsymbol{\beta} + g(t_k) \mathbf{z}_{ik}^* \boldsymbol{\gamma}$$

where $\boldsymbol{\gamma}$ is a $q \times 1$ vector of unknown regression parameters for $\mathbf{z}(\cdot)$ and $\mathbf{z}_{ik}^* = \mathbf{z}_i(t_k)$ are the covariate values of the tvc() variables for subject i at time t_k. For baseline covariates, $\mathbf{z}_{ik}^* = \mathbf{z}_i$. For time-varying covariates, if $t_k = t_{ij}$ is one of the examination times for subject i, then $\mathbf{z}_{ik}^* = \mathbf{z}_{ij}$. Otherwise, $\mathbf{z}_{ik}^*$ is imputed as described in the tvcovimpute() option in *Options*.

Acknowledgments

References

Cai, T., and R. A. Betensky. 2003. Hazard regression for interval-censored data with penalized spline. *Biometrics* 59: 570–579. https://www.jstor.org/stable/3695433.

Cox, D. R. 1972. Regression models and life-tables (with discussion). *Journal of the Royal Statistical Society, Series B* 34: 187–220. https://doi.org/10.1111/j.2517-6161.1972.tb00899.x.

———. 1975. Partial likelihood. *Biometrika* 62: 269–276. https://doi.org/10.1093/biomet/62.2.269.

Finkelstein, D. M. 1986. A proportional hazards model for interval-censored failure time data. *Biometrics* 42: 845–854. https://doi.org/10.2307/2530698.

Finkelstein, D. M., and R. A. Wolfe. 1985. A semiparametric model for regression analysis of interval-censored failure time data. *Biometrics* 41: 933–945. https://doi.org/10.2307/2530965.

Huang, J., and J. A. Wellner. 1997. Interval censored survival data: A review of recent progress. In *Proceedings of the First Seattle Symposium in Biostatistics: Survival Analysis*, ed. D. Y. Lin and T. R. Fleming, 123–169. New York: Springer.

Lindsey, J. C., and L. M. Ryan. 1998. Methods for interval-censored data. *Statistics in Medicine* 17: 219–238. https://doi.org/10.1002/(SICI)1097-0258(19980130)17:2⟨219::AID-SIM735⟩3.0.CO;2-O.

Lindsey, J. K. 1998. A study of interval censoring in parametric regression models. *Lifetime Data Analysis* 4: 329–354. https://doi.org/10.1023/A:1009681919084.

Murphy, S. A., and A. W. van der Vaart. 2000. On profile likelihood. *Journal of the American Statistical Association* 95: 449–465. https://doi.org/10.2307/2669386.

Odell, P. M., K. M. Anderson, and R. B. D'Agostino. 1992. Maximum likelihood estimation for interval-censored data using a Weibull-based accelerated failure time model. *Biometrics* 48: 951–959. https://doi.org/10.2307/2532360.

Rabinowitz, D., A. A. Tsiatis, and J. Aragon. 1995. Regression with interval-censored data. *Biometrika* 82: 501–513. https://doi.org/10.2307/2337529.

Sun, J. 2006. *The Statistical Analysis of Interval-Censored Failure Time Data*. New York: Springer.

Sun, J., and J. Li. 2014. Interval censoring. In *Handbook of Survival Analysis*, ed. J. P. Klein, H. C. van Houwelingen, J. G. Ibrahim, and T. H. Scheike, 369–390. Boca Raton, FL: CRC Press.

Turnbull, B. W. 1976. The empirical distribution function with arbitrarily grouped, censored and truncated data. *Journal of the Royal Statistical Society, Series B* 38: 290–295. https://doi.org/10.1111/j.2517-6161.1976.tb01597.x.

Wang, L., C. S. McMahan, M. G. Hudgens, and Z. P. Qureshi. 2016. A flexible, computationally efficient method for fitting the proportional hazards model to interval-censored data. *Biometrics* 72: 222–231. https://doi.org/10.1111/biom.12389.

Zeng, D., F. Gao, and D. Y. Lin. 2017. Maximum likelihood estimation for semiparametric regression models with multivariate interval-censored data. *Biometrika* 104: 505–525. https://doi.org/10.1093/biomet/asx029.

Zeng, D., L. Mao, and D. Y. Lin. 2016. Maximum likelihood estimation for semiparametric transformation models with interval-censored data. *Biometrika* 103: 253–271. https://doi.org/10.1093/biomet/asw013.

Zhang, Y., L. Hua, and J. Huang. 2010. A spline-based semiparametric maximum likelihood estimation method for the Cox model with interval-censored data. *Scandinavian Journal of Statistics* 37: 338–354. https://doi.org/10.1111/j.1467-9469.2009.00680.x.

Also see

Title

> **stintcox PH-assumption plots** — Plots of proportional-hazards assumption after stintcox

Description

stintphplot plots $-\ln\{-\ln(\text{survival})\}$ curves for each level of a nominal or ordinal covariate specified in option by() versus $\ln(\text{analysis time})$ for interval-censored data. These plots are often referred to as "log–log" survival plots. Optionally, the estimates can be adjusted for covariates. The proportional-hazards assumption is satisfied when the curves are parallel.

stintcoxnp plots Turnbull's nonparametric and Cox predicted survival curves for each level of a categorical covariate specified in option by() for interval-censored data. The Turnbull nonparametric curve for interval-censored data is analogous to the Kaplan–Meier nonparametric curve for right-censored data. The closer the nonparametric estimates are to the Cox estimates, the less likely it is that the proportional-hazards assumption has been violated.

Quick start

Log–log plot of survival

Check for parallel lines in plot of $-\ln\{-\ln(\text{survival})\}$ versus $\ln(\text{analysis time})$ for each category of covariate a for interval-censored event-time data with interval endpoints ltime and rtime

    stintphplot, interval(ltime rtime) by(a)

Same as above, but adjust for average values of covariates x1 and x2

    stintphplot, interval(ltime rtime) by(a) adjustfor(x1 x2)

Same as above

    stintphplot, interval(ltime rtime) by(a) adjustfor(x1 x2, atmeans)

Adjust for $x1 = 0$ and $x2 = 0$

    stintphplot, interval(ltime rtime) by(a) adjustfor(x1 x2, atzeros)

Turnbull's nonparametric and Cox predicted survival plots

Compare Turnbull's nonparametric curve with predicted survival from the Cox model for each category of covariate a for interval-censored event-time data with interval endpoints ltime and rtime

    stintcoxnp, interval(ltime rtime) by(a)

Same as above, but create separate plots for each level of a

    stintcoxnp, interval(ltime rtime) by(a) separate

Menu

stintphplot

Statistics > Survival analysis > Regression models > Assess PH assumption for interval-censored data

stintcoxnp

Statistics > Survival analysis > Regression models > Nonparametric versus Cox predicted survival

Syntax

Check proportional-hazards assumption:

Log–log plot of survival

stintphplot $\big[$ *if* $\big]$, <u>interval</u>(t_l t_u) $\big\{$by(*varname*) | strata(*varname*)$\big\}$

$\big[$ *phplot_options* $\big]$

Nonparametric and Cox predicted survival plots

stintcoxnp $\big[$ *if* $\big]$, <u>interval</u>(t_l t_u) by(*varname*) $\big[$ *stintcoxnp_options* $\big]$

phplot_options	Description
Main	
* <u>interval</u>(t_l t_u)	lower and upper endpoints for the event-time interval
* by(*varname*)	fit separate Cox models for levels of *varname*
* strata(*varname*)	fit stratified Cox model; requires adjustfor()
<u>adjustfor</u>(*varlist*$\big[$, *suboptions*$\big]$)	adjust the estimates to specific values of *varlist*; default is overall means
<u>noneg</u>ative	plot $\ln\{-\ln(\text{survival})\}$ instead of $-\ln\{-\ln(\text{survival})\}$
<u>nolntime</u>	plot curves against analysis time
Plot	
plot#opts(*phplot_plot_options*)	affect rendition of the #th connected line and #th plotted points
Add plots	
addplot(*plot*)	add other plots to the generated graph
Y axis, X axis, Titles, Legend, Overall	
twoway_options	any options other than by() documented in [G-3] *twoway_options*

* interval() is required with stintphplot.

* Either by(*varname*) or strata(*varname*) and adjustfor(*varlist*) is required with stintphplot.

phplot_plot_options	Description
cline_options	change look of lines or connecting method
marker_options	change look of markers (color, size, etc.)

stintcoxnp_options	Description
Main	
* <u>interval</u>(t_l t_u)	lower and upper endpoints for the event-time interval
* by(*varname*)	specify a categorical covariate
separate	draw separate plot for nonparametric and Cox predicted curves
Nonparametric plot	
npopts(*stintcoxnp_plot_options*)	affect rendition of the nonparametric curve
np#opts(*stintcoxnp_plot_options*)	affect rendition of the #th nonparametric curve; not allowed with separate
Cox predicted plot	
coxopts(*stintcoxnp_plot_options*)	affect rendition of the Cox predicted curve
<u>cox</u>#opts(*stintcoxnp_plot_options*)	affect rendition of the #th Cox predicted curve; not allowed with separate
Add plots	
addplot(*plot*)	add other plots to the generated graph
Y axis, X axis, Titles, Legend, Overall	
twoway_options	any options other than by() documented in [G-3] *twoway_options*
By options	
byopts(*byopts*)	how subgraphs are combined, labeled, etc.

*interval() and by() are required with stintcoxnp.

stintcoxnp_plot_options	Description
connect_options	change look of connecting method
marker_options	change look of markers (color, size, etc.)

Options

Options are presented under the following headings:

> Options for stintphplot
> Options for stintcoxnp

Options for stintphplot

Main

interval($t_l\ t_u$); see [ST] **stintcox**. interval() is required.

by(*varname*) specifies a categorical covariate for which the proportional-hazards assumption is to be checked. Without option adjustfor(), stintphplot computes nonparametric estimates of the survivor functions for each category of *varname* and produces log–log plots. With adjustfor(), the command fits a separate Cox model for each category of *varname* to produce log–log plots. When the plotted curves are parallel for all categories of *varname*, the proportional-hazards assumption is considered to be met for *varname*. Either by() or strata() is required with stintphplot.

strata(*varname*) is an alternative to by(). Rather than fitting separate Cox models for each value of *varname*, strata() fits one stratified Cox model. You must also specify adjustfor() with the strata() option.

adjustfor(*varlist*[, *suboptions*]) adjusts the estimates of the survivor function to specific values of *varlist*. The default is to adjust to overall mean values of covariates. adjustfor() can be specified with by(); it is required with strata().

> *suboptions* are atomeans (the default), atmeans, atzeros, atbase, and at(); see [ST] *adjustfor_option*.

nonegative specifies that ln{−ln(survival)} be plotted instead of −ln{−ln(survival)}.

nolntime specifies that curves be plotted against analysis time instead of against ln(analysis time).

Plot

plot#opts(*phplot_plot_options*) affects the rendition of the #th connected line and #th plotted points; see [G-3] *cline_options* and [G-3] *marker_options*.

Add plots

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] *addplot_option*.

Y axis, X axis, Titles, Legend, Overall

twoway_options are any of the options documented in [G-3] *twoway_options*, excluding by(). These include options for titling the graph (see [G-3] *title_options*) and for saving the graph to disk (see [G-3] *saving_option*).

Options for stintcoxnp

Main

interval($t_l\ t_u$); see [ST] **stintcox**. interval() is required.

by(*varname*) specifies a categorical covariate *varname* for which the agreement between the Cox predicted and nonparametric survivor functions is to be explored. The nonparametric and Cox predicted survivor functions are plotted for each level of *varname*. The agreement between the two survival curves across all levels indicates that the proportional-hazards assumption for *varname* is reasonable. by() is required.

separate produces separate plots of Cox predicted and nonparametric survivor functions for each value of the variable specified with by().

⎡ Nonparametric plot ⎤

npopts(*stintcoxnp_plot_options*) affects the rendition of the nonparametric curve; see [G-3] *connect_options* and [G-3] *marker_options*.

np#opts(*stintcoxnp_plot_options*) affects the rendition of the #th nonparametric curve; see [G-3] *connect_options* and [G-3] *marker_options*. This option is not allowed with separate.

⎡ Cox predicted plot ⎤

coxopts(*stintcoxnp_connect_options*) affects the rendition of the Cox predicted curve; see [G-3] *connect_options* and [G-3] *marker_options*.

cox#opts(*stintcoxnp_connect_options*) affects the rendition of the #th Cox predicted curve; see [G-3] *connect_options* and [G-3] *marker_options*. This option is not allowed with separate.

⎡ Add plots ⎤

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] *addplot_option*.

⎡ Y axis, X axis, Titles, Legend, Overall ⎤

twoway_options are any of the options documented in [G-3] *twoway_options*, excluding by(). These include options for titling the graph (see [G-3] *title_options*) and for saving the graph to disk (see [G-3] *saving_option*).

⎡ By options ⎤

byopts(*byopts*) affects the appearance of the combined graph when by() and separate are specified, including the overall graph title and the organization of subgraphs. See [G-3] *by_option*.

Remarks and examples

The underlying assumption for the Cox proportional hazards model is that the hazard ratio is constant over time. Therefore, it is important to evaluate the validity of this assumption. stintphplot and stintcoxnp are graphical tools for assessing violations of the proportional-hazards assumption for interval-censored data. For proportional-hazards-assumption plots for right-censored data, see [ST] **stcox PH-assumption tests**.

stintphplot plots $-\ln\{-\ln(\text{survival})\}$ curves for each category of a nominal or ordinal covariate versus $\ln(\text{analysis time})$. These are often referred to as "log–log" plots. Optionally, these estimates can be adjusted for covariates. If the plotted lines are reasonably parallel, the proportional-hazards assumption has not been violated.

Another graphical method of evaluating the proportional-hazards assumption is to plot the non-parametric maximum-likelihood estimation (NPMLE) survival curves proposed by Turnbull (1976) and compare them with the Cox predicted curves for each level of a categorical variable. This plot is produced by stintcoxnp. When the two curves are close together, the proportional-hazards assumption has not been violated.

For a single categorical covariate in a Cox model, you can use stintphplot and stintcoxnp to check the proportional-hazards assumption. In the presence of multiple covariates, you can use only stintphplot, and you should adjust for covariates by including all but the tested covariate in the adjustfor() option.

To test the proportional-hazards assumption, you do not need to fit a Cox proportional hazards model using stintcox before issuing stintphplot or stintcoxnp.

With interval-censored data, the analysis time is formed by the unique values of lower and upper endpoints t_l and t_u, which must be specified in the interval() option.

▷ Example 1: Proportional-hazards assumption holds for a model with a single covariate

Consider the dataset described in example 1 of [ST] **stintcox**. Suppose we want to check whether the proportional-hazards assumption holds for a single binary variable inject. We will use stintphplot first and then stintcoxnp:

```
. use https://www.stata-press.com/data/r18/idu
(Modified Bangkok IDU Preparatory Study)
. stintphplot, interval(ltime rtime) by(inject)
Computing nonparametric estimates for inject = No ...
Computing nonparametric estimates for inject = Yes ...
```


Nonparametric estimates for interval-censored data
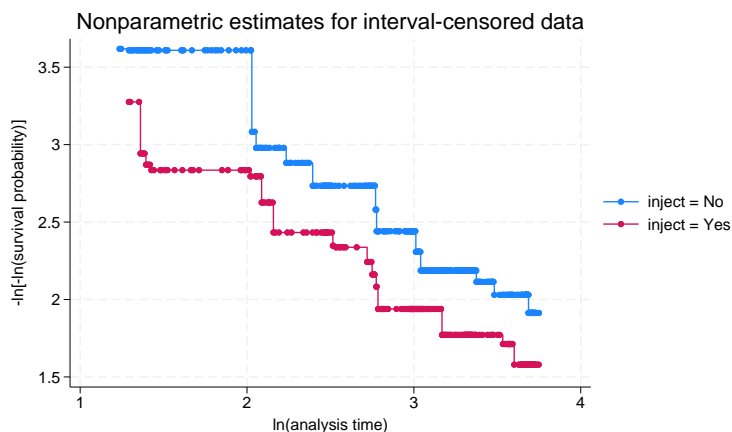
```
. stintcoxnp, interval(ltime rtime) by(inject) byopts(style(altleg)) separate
Computing nonparametric estimates ...
Computing Cox estimates ...
```


Interval-censored data
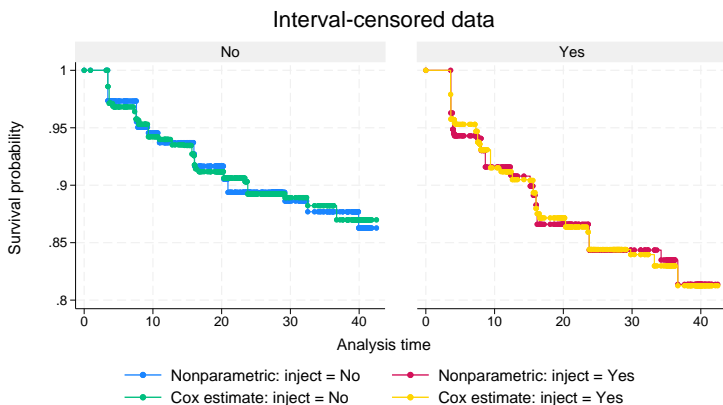
Graphs by Injected drugs before recruitment

The `stintphplot` plot shows that the two lines are parallel, implying that the proportional-hazards assumption holds for `inject`. This is confirmed in the `stintcoxnp` plot, where the Turnbull nonparametric estimates and Cox estimates are close together.

◁

▷ Example 2: Proportional-hazards assumption does not hold for a model with a single covariate

Continuing with `idu.dta`, suppose we want to examine the proportional-hazards assumption for a single binary variable `male`:

```
. stintphplot, interval(ltime rtime) by(male)
Computing nonparametric estimates for male = No ...
Computing nonparametric estimates for male = Yes ...
```


Nonparametric estimates for interval-censored data
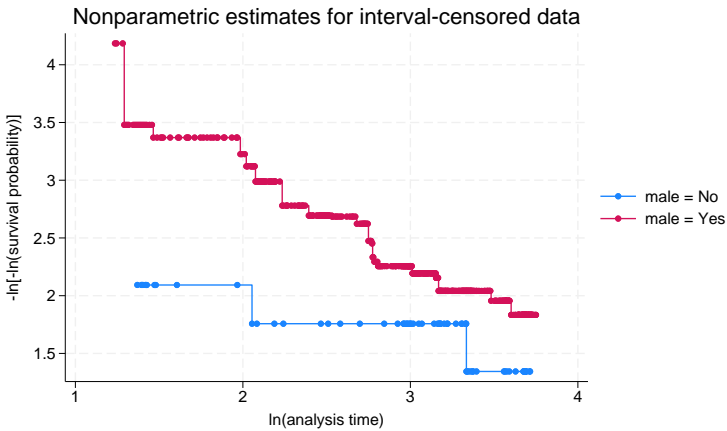
```
. stintcoxnp, interval(ltime rtime) by(male) byopts(style(altleg)) separate
Computing nonparametric estimates ...
Computing Cox estimates ...
```


Interval-censored data
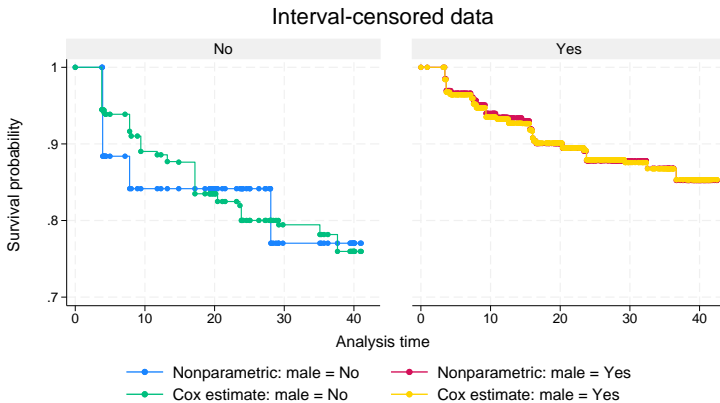
The "log–log" plot displays two lines that are not parallel, which indicates that the proportional-hazards assumption is violated for `male`. If we take a closer look at the separate `stintcoxnp` plots for both males and females, we can see that there are considerable differences between the NPMLE estimates and the Cox estimates, especially for females. Therefore, using this variable in the Cox model may not be appropriate.

◁

▷ Example 3: Checking proportional-hazards assumption for a model with multiple covariates

When a Cox model contains multiple covariates, as in example 1 of [ST] **stintcox**, we should use the `adjustfor()` option to adjust for covariates.

To check the proportional-hazards assumption for `inject` for the Cox model in that example, we specify all the remaining covariates in the `adjustfor()` option.

```
. stintphplot, interval(ltime rtime) by(inject)
> adjustfor(age_mean i.male i.needle i.jail)
Fitting Cox model with covariates from option adjustfor()
for inject = No ...
Fitting Cox model with covariates from option adjustfor()
for inject = Yes ...
```
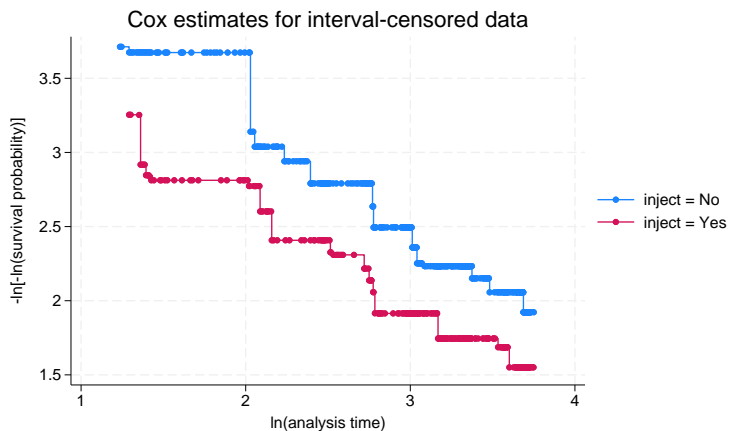

In this case, a separate Cox model, which contains all covariates from the `adjustfor()` option, is fit for each level of `inject`.

To check the proportional-hazards assumption for `male`, we include all covariates except `male` in the `adjustfor()` option.

```
. stintphplot, interval(ltime rtime) by(male)
> adjustfor(age_mean i.needle i.inject i.jail)
Fitting Cox model with covariates from option adjustfor()
for male = No ...
Fitting Cox model with covariates from option adjustfor()
for male = Yes ...
```
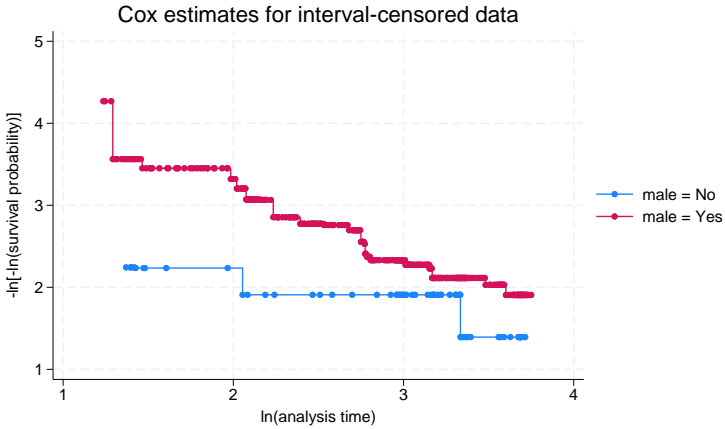

After adjusting for covariates, we see our conclusions from example 1 and example 2 remain the same. The proportional-hazards assumption appears to be met for the `inject` variable but not for the `male` variable.

It may not always be feasible to fit a separate Cox model for all levels of a tested covariate when some levels have only a few observations. In this case, you may consider using the strata() option instead of by() to fit a Cox model stratified on the tested covariate. For instance, you could type

```
. stintphplot, interval(ltime rtime) strata(male)
> adjustfor(age_mean i.needle i.inject i.jail)
Fitting Cox model stratified on male with covariates from option adjustfor() ...
```

Stratified Cox estimates for interval-censored data
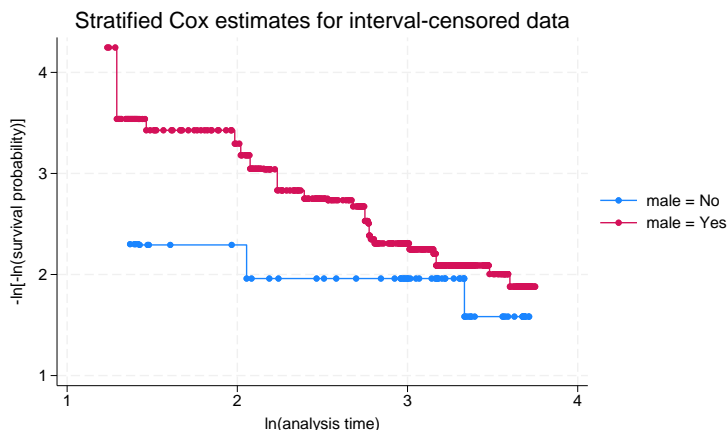
Note that the stintcoxnp command is not appropriate for testing the proportional-hazards assumption in the presence of multiple covariates in a Cox model.

◁

Methods and formulas

For one covariate, x, the Cox proportional hazards model reduces to

$$h(t; x) = h_0(t) \exp(x\beta)$$

where $h_0(t)$ is the baseline hazard function from the Cox model. Let $S_0(t)$ and $H_0(t)$ be the corresponding Cox baseline survivor and baseline cumulative hazard functions, respectively.

The proportional-hazards assumption implies that

$$H(t; x) = H_0(t) \exp(x\beta)$$

or

$$\ln\{H(t; x)\} = \ln\{H_0(t)\} + x\beta$$

where $H(t; x)$ is the cumulative hazard function. Thus, under the proportional-hazards assumption, the logs of the cumulative hazard functions at each level of the covariate have equal slope. This is the basis for the method implemented in `stintphplot`.

The proportional-hazards assumption also implies that

$$S(t; x) = S_0(t)^{\exp(x\beta)} = \exp\{-H_0(t)\}^{\exp(x\beta)}$$

so the plot is referred to as a "log–log" survival plot because $\ln[-\ln\{S(t; x)\}] = \ln\{H(t; x)\}$.

For the Cox model, let $\widehat{H}(t; x) = \widehat{H}_0(t) \exp(x\widehat{\beta})$, where $\widehat{H}_0(t) = \sum_{t_k \leq t} \widehat{h}_k$, $\widehat{h}_k$'s ($k = 1, \ldots, m$) are estimated baseline hazard contributions and t_k's are the unique time points, ordered from smallest to largest, of the observed interval endpoints t_l and t_u. Let $\widehat{S}(t; x) = \exp\{-\widehat{H}(t; x)\}$ be the estimated survivor function based on the Cox model. This function is a step function. Thus, for each level of the covariate of interest, we can assess violations of the proportional-hazards assumption by comparing these survival estimates with nonparametric estimates from Turnbull (1976). See Kalbfleisch and Prentice (2002) or Hess (1995).

`stintcoxnp` plots Turnbull's (1976) NPMLE curves for each level of the covariate together with the survival curves predicted by the Cox model. The closer the Turnbull estimates are to the Cox estimates, the less likely it is that the proportional-hazards assumption has been violated.

References

Hess, K. R. 1995. Graphical methods for assessing violations of the proportional hazards assumption in Cox regression. *Statistics in Medicine* 14: 1707–1723. https://doi.org/10.1002/sim.4780141510.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Turnbull, B. W. 1976. The empirical distribution function with arbitrarily grouped, censored and truncated data. *Journal of the Royal Statistical Society, Series B* 38: 290–295. https://doi.org/10.1111/j.2517-6161.1976.tb01597.x.

Also see

Title

> **stintcox postestimation** — Postestimation tools for stintcox

Postestimation commands

The following postestimation commands are of special interest after `stintcox`:

Command	Description
estat gofplot	produce goodness-of-fit plot
stcurve	plot the survivor, failure, hazard, or cumulative hazard function
stintcoxnp	plot nonparametric curves and Cox predicted curves
stintphplot	plot $-\ln\{-\ln(\text{survival})\}$ curves

The following standard postestimation commands are also available:

Command	Description
contrast	contrasts and ANOVA-style joint tests of estimates
estat ic	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC)
estat summarize	summary statistics for the estimation sample
estat vce	variance–covariance matrix of the estimators (VCE)
estimates	cataloging estimation results
etable	table of estimation results
hausman	Hausman's specification test
lincom	point estimates, standard errors, testing, and inference for linear combinations of coefficients
lrtest	likelihood-ratio test
margins	marginal means, predictive margins, marginal effects, and average marginal effects
marginsplot	graph the results from margins (profile plots, interaction plots, etc.)
nlcom	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
predict	hazard ratios, survivor functions, influence statistics, residuals, etc.
predictnl	point estimates, standard errors, testing, and inference for generalized predictions
pwcompare	pairwise comparisons of estimates
test	Wald tests of simple and composite linear hypotheses
testnl	Wald tests of nonlinear hypotheses

predict

Description for predict

predict creates new variables containing predictions such as hazard ratios, linear predictions, standard errors, and baseline survivor and baseline cumulative hazard functions.

Menu for predict

Statistics > Postestimation

Syntax for predict

Single-record-per-subject interval-censored data with baseline covariates

> predict [*type*] *newvar* [*if*] [*in*] [, *statistic*]

> predict [*type*] { *stub** | *newvar_l* *newvar_u* } [*if*] [*in*] , *statistic2*

Single-record-per-subject interval-censored data with time-varying covariates

> predict [*type*] *newvar* [*if*] [*in*] [, mgale]

> predict [*type*] { *stub** | *newvar_l* *newvar_u* } [*if*] [*in*] [, *statistic statistic2*]

Multiple-record-per-subject interval-censored data

> predict [*type*] *newvar* [*if*] [*in*] [, *statistic statistic2*]

For single-record-per-subject data, predictions that depend on time, *statistic2*, are calculated for both the lower endpoint t_l and the upper endpoint t_u of the time interval specified in stintcox's interval() option and thus are stored in two new variables. This also applies to each *statistic* except mgale in the presence of time-varying covariates. For multiple-record-per-subject data, predictions are calculated for the examination times specified in stintcox's time() option and are always stored in a single variable.

statistic	Description
Main	
hr	predicted hazard ratio, also known as the relative hazard; the default
xb	linear prediction $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$
stdp	standard error of the linear prediction, $\text{SE}(\mathbf{x}_j\widehat{\boldsymbol{\beta}})$
*mgale	martingale-like residuals

statistic2	Description
Main	
*basesurv	baseline survivor function
*basechazard	baseline cumulative hazard function
*basehc	baseline hazard contributions
*csnell	Cox–Snell-like residuals

Unstarred statistics are available both in and out of sample; type predict ... if e(sample) ... if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample.

Options for predict

> Main

hr, the default, calculates the relative hazard (hazard ratio), that is, the exponentiated linear prediction, $\exp(\mathbf{x}_j\widehat{\boldsymbol{\beta}})$. After stintcox with option tvc(), it calculates $\exp\left\{\mathbf{x}_j\widehat{\boldsymbol{\beta}} + g(t)\mathbf{z}_j\widehat{\boldsymbol{\gamma}}\right\}$, where $g(t)$ is a function of time as specified in stintcox's texp() option.

xb calculates the linear prediction from the fitted model. That is, you fit the model by estimating a set of parameters, β_1, β_2, ..., β_p, and the linear prediction is $\widehat{\beta}_1 x_{1j} + \widehat{\beta}_2 x_{2j} + \cdots + \widehat{\beta}_p x_{pj}$, often written in matrix notation as $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$.

The x_{1j}, x_{2j}, ..., x_{pj} used in the calculation are obtained from the data currently in memory and need not correspond to the data on the independent variables used in estimating $\boldsymbol{\beta}$.

After stintcox with option tvc(), it calculates $\mathbf{x}_j\widehat{\boldsymbol{\beta}} + g(t)\mathbf{z}_j\widehat{\boldsymbol{\gamma}}$, where $g(t)$ is a function of time as specified in stintcox's texp() option.

stdp calculates the standard error of the linear prediction xb.

basesurv calculates the baseline survivor function.

basechazard calculates the baseline cumulative hazard.

basehc calculates the baseline hazard contributions.

mgale calculates interval-censored martingale-like residuals, which are an interval-censored version of martingale residuals for right-censored data.

csnell calculates the Cox–Snell-like residuals, which are the estimates of the cumulative hazard function obtained from the fitted model.

margins

Description for margins

margins estimates margins of response for hazard ratios and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

margins [*marginlist*] [, *options*]

margins [*marginlist*] , <u>predict</u>(*statistic ...*) [<u>predict</u>(*statistic ...*) ...] [*options*]

statistic	Description
hr	hazard ratio, also known as the relative hazard
xb	linear prediction
<u>stdp</u>	not allowed with margins
<u>basesurv</u>	not allowed with margins
<u>basechazard</u>	not allowed with margins
basehc	not allowed with margins
mgale	not allowed with margins
<u>csnell</u>	not allowed with margins

Statistics not allowed with margins are functions of stochastic quantities other than e(b).

For the full syntax, see [R] **margins**.

Remarks and examples

Remarks are presented under the following headings:

> *Baseline functions*
> *Residuals and diagnostic measures*
> *Postestimation after stintcox with option tvc()*
> *Survivor curves for multiple-record-per-subject data with time-varying covariates*

Baseline functions

predict after stintcox is used to generate a new variable or variables containing predicted values or residuals. predict can generate predicted hazard ratios, linear predictions, and standard errors of the linear predictions. It can also predict the baseline survivor function, baseline cumulative hazard function, or baseline hazard contributions. Baseline functions refer to the values of the functions when all covariates are set to 0. If the dataset is single-record-per-subject data, it calculates those statistics for both interval endpoints t_l and t_u specified in the interval() option with stintcox.

▷ Example 1: Baseline survivor function

We use data from example 1 of [ST] **stintcox** and fit the Cox proportional hazards model in which the time to HIV infection depends on centered age variable (age_mean) and whether a subject has been in jail at the time of recruitment (jail):

```
. use https://www.stata-press.com/data/r18/idu
(Modified Bangkok IDU Preparatory Study)

. stintcox age_mean i.jail, interval(ltime rtime)
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):
Iteration 0:   Log likelihood = -1086.2564
Iteration 100: Log likelihood = -600.55818
Iteration 200: Log likelihood = -600.47678
Iteration 295: Log likelihood = -600.46551

Computing standard errors: ................ done

Interval-censored Cox regression                Number of obs   =   1,124
Baseline hazard: Reduced intervals                   Uncensored =       0
                                                  Left-censored =      41
Event-time interval:                             Right-censored =     991
  Lower endpoint: ltime                          Interval-cens. =      92
  Upper endpoint: rtime
                                                 Wald chi2(2)    =   14.07
Log likelihood = -600.46551                      Prob > chi2     =  0.0009
```

	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf. interval]
age_mean	.9647236	.0122201	-2.84	0.005	.9410676	.9889743
jail						
Yes	1.819721	.3804566	2.86	0.004	1.207927	2.741379

Note: Standard error estimates may be more variable for small datasets and
      datasets with low proportions of interval-censored observations.

We can now estimate, for instance, the baseline survivor function. For this dataset, estimates of baseline survivor function, as well as baseline cumulative hazard and baseline hazard contributions, are intervals. So, to compute these statistics, we can either specify two new variable names with predict or specify a stub *stub** that will automatically create two new variable names *stub1* and *stub2*; the first one will contain statistics computed using the lower time endpoint, and the second one will contain statistics computed using the upper time endpoint.

```
. predict bs_l bs_u, basesurv
```

```
. list bs_l bs_u ltime rtime age jail in 300/310
```

	bs_l	bs_u	ltime	rtime	age	jail
300.	.8989196	0	40.00	.	36	Yes
301.	.9547279	.9450011	11.97	15.84	21	Yes
302.	.9647679	.9506636	8.20	15.18	36	Yes
303.	.8989196	0	39.93	.	40	Yes
304.	.8989196	0	39.48	.	25	Yes
305.	.8989196	0	36.72	.	40	Yes
306.	.8989196	0	39.93	.	40	Yes
307.	.9767599	0	4.30	.	34	Yes
308.	.8989196	0	39.74	.	42	No
309.	.8989196	0	37.61	.	30	Yes
310.	.8989196	0	39.97	.	28	Yes

Listed above are the baseline survivor functions, `bs_l` and `bs_u`, evaluated at the lower and upper time endpoints `ltime` and `rtime`, for subjects 300 to 310.

To graph the baseline survival curve, we can use `stcurve` with all covariates set to 0. We can also graph survivor functions after `stintcox` with covariates set to any value. See [ST] **stcurve** for more information.

```
. stcurve, survival at(age_mean=0 jail=0)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```
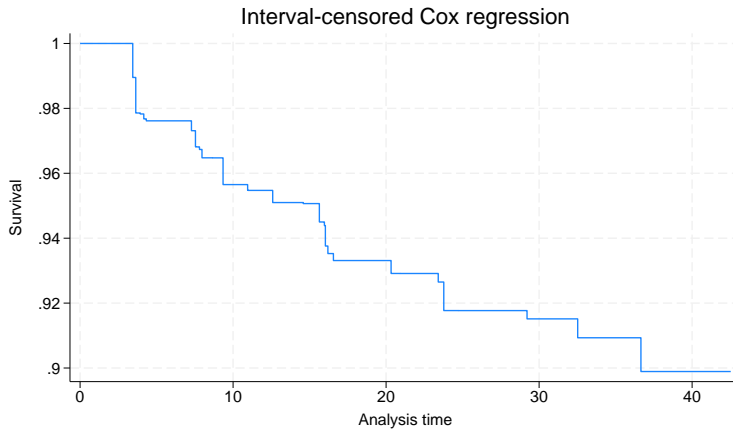
Residuals and diagnostic measures

For right-censored data, several types of residuals have been introduced to assess the appropriateness of the Cox proportional hazards model; see *Residuals and diagnostic measures* in [ST] **stcox postestimation** for details.

Farrington (2000) proposed extensions of those residuals for interval-censored data; see the reference for applications and a discussion of limitations of the residuals for interval-censored data. Here we offer martingale-like residuals and Cox−Snell-like residuals for visual model checking. Martingale-like residuals are useful in determining the functional form of covariates to be included in the model. They are also useful in assessing whether some covariates are needed in the model and for identifying outliers. Cox−Snell-like residuals are useful in assessing the overall model fit. If the model fits the data, those residuals should approximate an interval-censored sample from the standard exponential distribution. These residuals can be plotted by using `estat gofplot`.

▷ Example 2: Assess overall model fit

To visually assess the overall model fit, we can use the Cox−Snell-like residuals. For right-censored data, Cox and Snell (1968) argued that if the correct model has been fit to the data, these residuals should have a censored standard exponential distribution. With interval-censored, Cox−Snell-like residuals approximate an interval-censored sample from this distribution.

`estat gofplot` calculates an empirical estimate of the cumulative hazard function based on the Cox−Snell-like residuals and plots the resulting cumulative hazard rate against the residuals themselves. If the model fits the data, the plot is expected to approximate a straight line with slope 1.

Continuing with example 1, let's produce the goodness-of-fit plot.

```
. estat gofplot
```


Goodness-of-fit plot for interval-censored Cox model
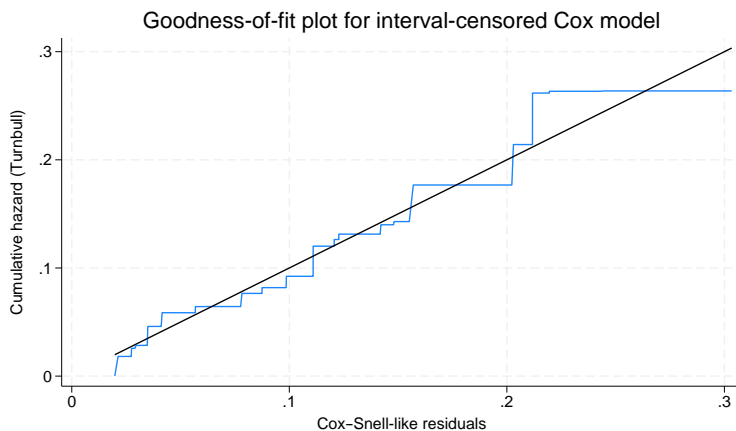
The goodness-of-fit plot shows that the jagged line stays very close to the 45° reference line. The Cox proportional hazards model appears to fit the data well.

◁

▷ Example 3: Martingale-like residuals

Martingale-like residuals may be used as a diagnostic tool to assess the functional form of a covariate to be included in a Cox model. To find the appropriate functional form of a variable, we fit a Cox model without the variable of interest and then plot a `lowess` smooth of the martingale-like residuals against some transformation of that variable. If the transformation is appropriate, then the smooth should be approximately linear.

Continuing with example 1, we can apply this procedure to check whether the functional form of the covariate `age_mean` is appropriate. First, we refit the Cox proportional hazards model on whether a subject has been in jail at the time of recruitment (`jail`). Then, we obtain the martingale-like residuals `mg` using `predict` with the `mgale` option. Last, we plot a `lowess` smooth of `mg` against variable `age_mean`.

```
. stintcox i.jail, interval(ltime rtime) nolog
note: using adaptive step size to compute derivatives.

Interval-censored Cox regression                Number of obs    =   1,124
Baseline hazard: Reduced intervals                   Uncensored =       0
                                                  Left-censored =      41
Event-time interval:                             Right-censored =     991
  Lower endpoint: ltime                          Interval-cens. =      92
  Upper endpoint: rtime
                                                 Wald chi2(1)     =    6.24
Log likelihood = -605.20197                      Prob > chi2      =  0.0125
```

	Haz. ratio	OPG std. err.	z	P>|z|	[95% conf. interval]
jail					
Yes	1.675924	.3464929	2.50	0.013	1.11756	2.513264

Note: Standard error estimates may be more variable for small datasets and datasets with low proportions of interval-censored observations.

```
. predict mg, mgale
. lowess mg age_mean, mean noweight title("") note("") m(o)
```
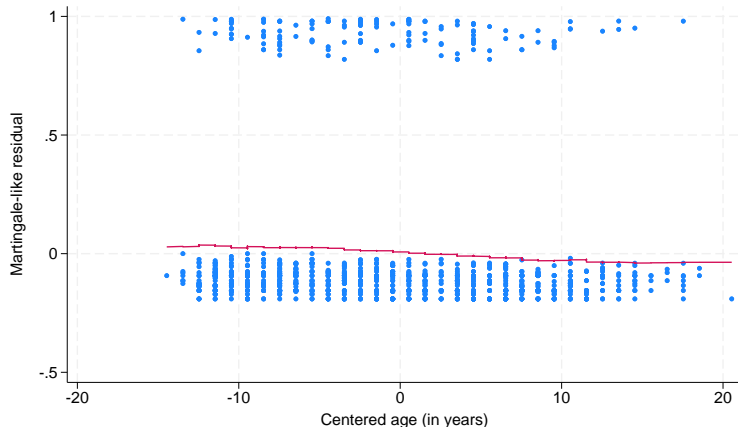

We used the `lowess` command with the `mean` and `noweight` options to obtain a plot of the running-mean smoother to ease interpretation. The smooth appears nearly linear, supporting the inclusion of the untransformed version of `age_mean` in our Cox model. Had the smooth not been linear, we would

have tried smoothing the martingale residuals against various transformations of `age_mean` until we found one that produced a near-linear smooth.

◁

Postestimation after stintcox with option tvc()

When the `tvc()` option is used with `stintcox`, the model includes the time-varying covariates formed by covariates in the `tvc()` option multiplied by the deterministic function of time specified in the `texp()` option. In this case, predictions that depend on the covariate pattern, such as a linear predictor and relative hazard, also depend on time. For instance, for single-record-per-subject data, `predict, xb` will now produce two new variables: one containing the linear predictor computed using the lower endpoint of the time interval and the other one computed using the upper endpoint.

After `stintcox, tvc()`, postestimation features such as `predict` and `stcurve` incorporate both fixed and time-varying covariates. For instance, the linear predictor will include the effects of both baseline covariates and time-varying covariates produced by the `tvc()` option.

Predictions of baseline functions and martingale-like residuals do not depend on covariates and are thus unaffected by `stintcox, tvc()`.

▷ Example 4: Predictions after stintcox with option tvc()

In *Testing the proportional-hazards assumption using option tvc()* in [ST] **stintcox**, we discovered that `jail` was one of the variables that did not satisfy the proportional-hazards assumption.

Let's fit the following model with a time-varying covariate $\mathtt{jail} \times t$:

$$h(t) = h_0(t) \exp\{\beta_1 \times \mathtt{age\_mean} + \beta_2 \times \mathtt{jail} + \gamma_1 \times (\mathtt{jail} \times t)\}$$

We use the `tvc()` option to create the time-varying covariate $jail \times t$:

```
. stintcox age_mean i.jail, interval(ltime rtime) tvc(i.jail) nohr
note: using adaptive step size to compute derivatives.

Performing EM optimization (showing every 100 iterations):
Iteration 0:    Log likelihood = -1086.2564
Iteration 100: Log likelihood =  -597.6072
Iteration 200: Log likelihood = -597.52358
Iteration 295: Log likelihood = -597.51247

Computing standard errors: ........................ done

Interval-censored Cox regression                 Number of obs    =   1,124
Baseline hazard: Reduced intervals                  Uncensored =        0
                                                 Left-censored =        41
Event-time interval:                            Right-censored =       991
  Lower endpoint: ltime                          Interval-cens. =       92
  Upper endpoint: rtime
                                                 Wald chi2(3)     =   16.87
Log likelihood = -597.51247                      Prob > chi2      =  0.0008
```

	Coefficient	OPG std. err.	z	P>|z|	[95% conf. interval]
main						
age_mean	-.0360376	.0126271	-2.85	0.004	-.0607864	-.0112889
jail						
Yes	-.1593675	.3777957	-0.42	0.673	-.8998335	.5810986
tvc						
jail						
Yes	.0732193	.0342545	2.14	0.033	.0060818	.1403569

```
Notes: Standard error estimates may be more variable for small datasets and
       datasets with low proportions of interval-censored observations.
       Variables in tvc equation interacted with _t.
Wald test that [tvc] = 0: chi2(1) = 4.5690          Prob > chi2 = 0.0326
```

Let's use `predict, xb` to calculate the linear prediction. Because the linear prediction now depends on time and there are two time interval endpoints in a single-record-per-subject data, `predict` requires two new variable names. Below, we use a stub notation xb* that automatically creates two new variables, xb1 and xb2, to store linear predictions. To demonstrate, we list the results for subjects 30 and 95.

```
. predict xb*, xb
(991 missing values generated)
note: prediction includes time-varying covariates formed by stintcox, tvc().

. list age_mean jail ltime rtime xb1 xb2 if _n==30 | _n==95
```

	age_mean	jail	ltime	rtime	xb1	xb2
30.	.53825623	Yes	7.147541	13.213115	.3445733	.7886905
95.	-.46174377	No	4.1311474	8.0327873	.0166402	.0166402

For subject 30, `jail` is equal to 1, and the linear predictions in xb1 are computed as $\widehat{\beta}_1 \times age\_mean + \widehat{\beta}_2 + \widehat{\gamma}_1 \times ltime$ and in xb2 as $\widehat{\beta}_1 \times age\_mean + \widehat{\beta}_2 + \widehat{\gamma}_1 \times rtime$. For subject 95, $jail = 0$, and thus both x1 and x2 contain $\widehat{\beta}_1 \times age\_mean$.

◁

▷ Example 5: stcurve after stintcox with option tvc()

stcurve can be used after stintcox to plot the estimated survivor, failure, hazard, or cumulative hazard function. When the tvc() option is specified with stintcox, stcurve incorporates the time-varying nature of the covariates. By default, a function is evaluated at the overall means of covariates in the model. For covariates specified in the tvc() option, their overall mean values are multiplied by the function of time, as specified in stintcox's texp() option, during a function evaluation.

```
. stcurve, survival
note: function evaluated at overall means of covariates, multiplied by _t for
      covariates in stintcox's tvc() option.
```
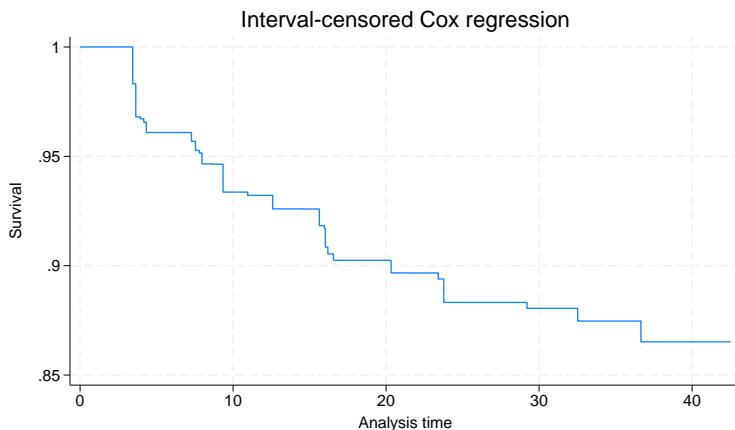

Here jail is binary, so we may want to use the at() option to evaluate the survivor function for jail = 0 and jail = 1. The overall mean is used for any covariate not listed in at() (age_mean in our example).

```
. stcurve, survival at(jail=(0 1))
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any), multiplied by _t for
      covariates in stintcox's tvc() option.
```
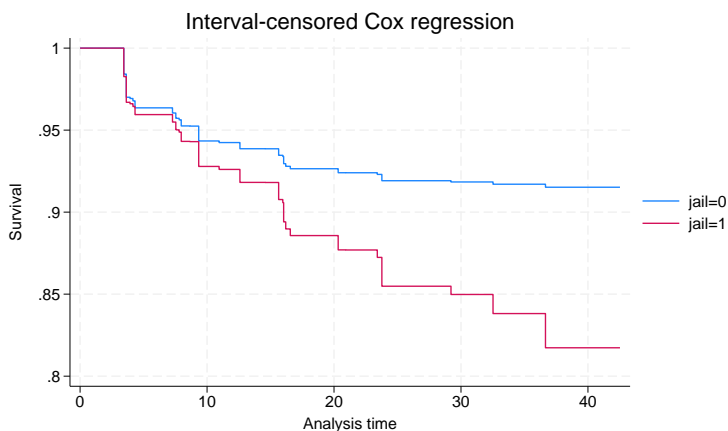

The survival of those who have not been imprisoned is higher than those who have.

◁

Survivor curves for multiple-record-per-subject data with time-varying covariates

By default, stcurve evaluates the survivor and other functions at the overall means of covariates. We can also use the at() option to evaluate functions at specific covariate values. Both of these options work well for baseline covariates. But multiple-record-per-subject data often include time-varying covariates. In that case, we may want to incorporate the time-varying nature of these covariates when plotting the survivor function or other functions. We can use the attmeans option to evaluate the function at time-specific means, which will vary with time for time-varying covariates. We can also specify our own time-varying values for all or some covariates by using the atframe() option.

In general, care should be taken when interpreting the survivor function in the presence of time-varying covariates. The interpretation is valid only for external time-varying covariates, which are covariates that are external to an individual under study, such as air temperature. (Any covariate that is formed as an interaction of a baseline covariate and a deterministic function of time is an external covariate by construction.) For an internal time-varying covariate, which is generated by an individual in a study, such as wellness status, the survivor function is not well defined. See, for instance, Kalbfleisch and Prentice (2002) for details.

▷ Example 6: stcurve with option attmeans

In example 4 in [ST] stintcox, we fit a model that included the time-varying jail_vary variable. In this example, jail_vary is an internal time-varying covariate because it can be observed for a subject only until a subject "survives" and is not censored. Although it would not be sensible to proceed with this example in practice, we will use it here solely to demonstrate the software.

So, continuing with example 4 in [ST] stintcox, after fitting the model with time-varying jail_vary, we use stcurve with the attmeans option to plot the survivor curve that uses the time-specific mean values for covariates at each time point, hence accounting for the average changes of the covariates, specifically jail_vary, over time:

```
. use https://www.stata-press.com/data/r18/idu2, clear
(Modified Bangkok IDU Preparatory Study with time-varying variable jail_vary)
. stintcox age_mean i.male i.needle i.inject i.jail_vary, id(id) time(time)
> status(is_seropos)
  (output omitted)
. stcurve, survival attmeans
note: function evaluated at time-specific means of covariates.
```
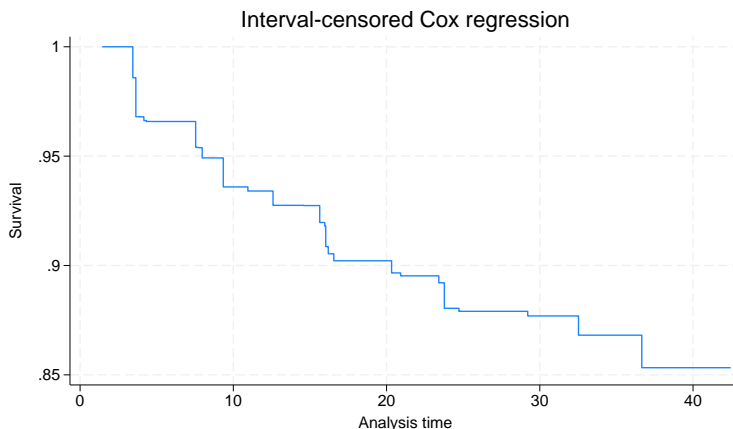
▷ Example 7: stcurve with option atframe()

We can use the `atframe()` option to specify time-varying covariate values to be used to evaluate the survivor function. We first need to create a frame that includes all or a subset of the covariates and the time variable in your model and fill in those variables with the values we want to use for prediction and the corresponding time points. If a subset of the covariates is specified in `atframe()`, the unspecified covariates are evaluated, by default, at their overall means or, if the `attmeans` option is specified, at their time-specific means.

Suppose we want to plot the survivor curve for an individual with the same covariate pattern as subject 2 in our dataset. We start by creating a new frame called `id2` and use `frame put` to copy the relevant information for subject 2 in this new frame. Then we `list` the data we just saved in frame `id2`.

```
. frame put time age_mean male needle inject jail_vary if id==2, into(id2)
. frame id2: list
```

	time	age_mean	male	needle	inject	jail_v~y
1.	4.1311475	-6.4617438	Yes	No	Yes	Yes
2.	8.2622951	-6.4617438	Yes	No	Yes	No
3.	12.295082	-6.4617438	Yes	No	Yes	No
4.	16.065574	-6.4617438	Yes	No	Yes	No
5.	20.098361	-6.4617438	Yes	No	Yes	No
6.	24.262295	-6.4617438	Yes	No	Yes	No

In frame `id2`, six examination times are recorded, and all covariate values for each examination time are filled in. Variables `age_mean`, `male`, `needle`, and `inject` are constant, and variable `jail_vary` changes at the second examination time.

We can graph the survivor curve for this particular profile by typing

```
. stcurve, survival atframe(id2)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
note: covariate values from frame id2 used to evaluate function.
```
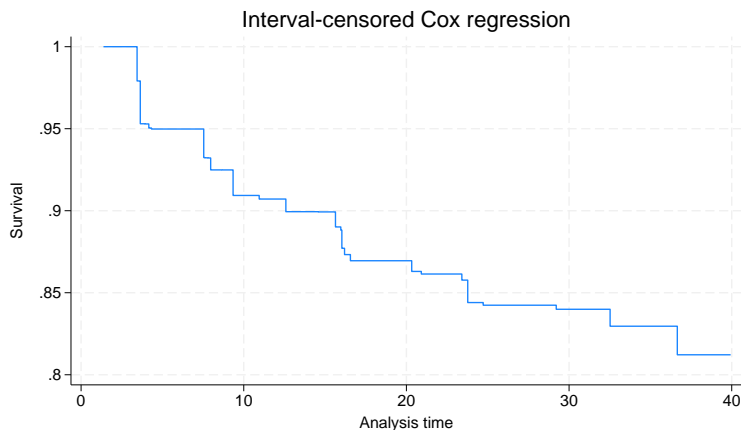

Unless you provide covariate values for all distinct observed times used during estimation, `stcurve` will use the same imputation method as specified in the `tvcovimpute()` option with `stintcox`

([ST] **stintcox**) to impute the unobserved covariate values for time points that are not recorded in the frame.

◁

Methods and formulas

Methods and formulas are presented under the following headings:

> Predictions for single-record interval-censored data
> Predictions for multiple-record interval-censored data
> Survivor curves for interval-censored data

Predictions for single-record interval-censored data

For single-record interval-censored data, each subject i contains one observation $(t_{li}, t_{ui}, \mathbf{x}_i)$, where t_{li} and t_{ui} define the observed time interval and $\mathbf{x}_i$ is a $1 \times p$ vector of covariate values. Let $\widehat{\boldsymbol{\beta}}$ be a $p \times 1$ vector of estimates of regression coefficients for $\mathbf{x}_i$.

Case of baseline covariates. The survivor function for subject i at time t can be estimated as

$$\widehat{S}_i(t) = \exp\Big\{ -\sum_{t_k \leq t} \widehat{h}_k \exp(\mathbf{x}_i \widehat{\boldsymbol{\beta}}) \Big\}$$

where $\widehat{h}_k$ $(k = 1, \ldots, m)$ are the estimates of the baseline hazard jumps at $t_1, \ldots, t_m$ and $t_1 < \cdots < t_m$ are the distinct time points for all $t_{li} > 0$ and $t_{ui} < \infty$.

predict *newvar*, *statistic* may be used after stintcox to predict various quantities, according to each following *statistic*:

hr:
$$newvar_i = \exp(\mathbf{x}_i \widehat{\boldsymbol{\beta}})$$

xb:
$$newvar_i = \mathbf{x}_i \widehat{\boldsymbol{\beta}}$$

stdp:
$$newvar_i = \widehat{se}(\mathbf{x}_i \widehat{\boldsymbol{\beta}})$$

mgale:
$$newvar_i = \frac{\widehat{S}_i(t_{li}) \log \widehat{S}_i(t_{li}) - \widehat{S}_i(t_{ui}) \log \widehat{S}_i(t_{ui})}{\widehat{S}_i(t_{li}) - \widehat{S}_i(t_{ui})}$$

For right-censored data, martingale residuals can be defined as the scores of the regression parameters. This property can carry over to the interval-censored data for martingale-like residuals mgale. Therefore, these residuals are expected to have mean zero and to be asymptotically uncorrelated. Furthermore, these residuals are orthogonal to variables included in the model. Thus, we can use them to assess the need to include other covariates in the model. See Farrington (2000) for details.

predict *newvar_l* *newvar_u*, *statistic* may be used after stintcox to predict a pair of quantities for each observation for both the lower and upper endpoints of the time interval (t_{li}, t_{ui}), according to each following *statistic*:

basehc:

$$newvar_{li} = I(t_k = t_{li})\widehat{h}_k$$

$$newvar_{ui} = I(t_k = t_{ui})\widehat{h}_k$$

basechazard:

$$newvar_{li} = \widehat{H}_0(t_{li}) = \sum_{k:t_k \le t_{li}} \widehat{h}_k$$

$$newvar_{ui} = \widehat{H}_0(t_{ui}) = \sum_{k:t_k \le t_{ui}} \widehat{h}_k$$

basesurv:

$$newvar_{li} = \widehat{S}_0(t_{li}) = \exp\left\{-\widehat{H}_0(t_{li})\right\}$$

$$newvar_{ui} = \widehat{S}_0(t_{ui}) = \exp\left\{-\widehat{H}_0(t_{ui})\right\}$$

csnell:

$$newvar_{li} = -\log \widehat{S}_i(t_{li})$$

$$newvar_{ui} = -\log \widehat{S}_i(t_{ui})$$

The Cox–Snell-like residuals are the estimates of the cumulative hazard function obtained from the fitted model. They are computed separately for each of the two interval endpoints. For interval-censored data, under the correct model assumption, these residuals are expected to approximate an interval-censored sample from the standard exponential distribution. Therefore, they can be used for checking the overall model fit. Cox–Snell-like residuals can never be negative and therefore are not symmetric about zero. See Farrington (2000) for details.

Case of time-varying covariates with option tvc(). When the tvc() option is specified with stintcox, predict modifies its calculations so that they include the time-varying components. Let $\mathbf{z}_i$ denote a $1 \times q$ vector of covariates specified in the tvc() option of stintcox, and let $g(t)$ denote the function of time specified in the texp() option of stintcox. Let a $q \times 1$ vector $\widehat{\gamma}$ contain the estimates of regression coefficients for $\mathbf{z}_i$. The survivor function for subject i at time t can be estimated as

$$\widehat{S}_i(t) = \exp\left[-\sum_{t_k \le t} \widehat{h}_k \exp\{\mathbf{x}_i\widehat{\boldsymbol{\beta}} + g(t_k)\mathbf{z}_i\widehat{\boldsymbol{\gamma}}\}\right]$$

Also, each following *statistic* changes to predict a pair of quantities for each observation for both the lower and upper endpoints of the time interval (t_{li}, t_{ui}):

hr:

$$newvar_{li} = \exp\{\mathbf{x}_i\widehat{\boldsymbol{\beta}} + g(t_{li})\mathbf{z}_i\widehat{\boldsymbol{\gamma}}\}$$

$$newvar_{ui} = \exp\{\mathbf{x}_i\widehat{\boldsymbol{\beta}} + g(t_{ui})\mathbf{z}_i\widehat{\boldsymbol{\gamma}}\}$$

`xb`:

$$newvar_{li} = \mathbf{x}_i\widehat{\boldsymbol\beta} + g(t_{li})\mathbf{z}_i\widehat{\boldsymbol\gamma}$$

$$newvar_{ui} = \mathbf{x}_i\widehat{\boldsymbol\beta} + g(t_{ui})\mathbf{z}_i\widehat{\boldsymbol\gamma}$$

`stdp`:

$$newvar_{li} = \widehat{\mathrm{se}}\big\{\mathbf{x}_i\widehat{\boldsymbol\beta} + g(t_{li})\mathbf{z}_i\widehat{\boldsymbol\gamma}\big\} = \big\{\mathbf{x}_i\widehat{\boldsymbol\Sigma}_{\boldsymbol\beta}\mathbf{x}_i' + 2g(t_{li})\mathbf{z}_i\widehat{\boldsymbol\Sigma}_{\boldsymbol\beta,\boldsymbol\gamma}\mathbf{x}_i' + g(t_{li})^2\mathbf{z}_i\widehat{\boldsymbol\Sigma}_{\boldsymbol\gamma}\mathbf{z}_i'\big\}^{\frac12}$$

$$newvar_{li} = \widehat{\mathrm{se}}\big\{\mathbf{x}_i\widehat{\boldsymbol\beta} + g(t_{ui})\mathbf{z}_i\widehat{\boldsymbol\gamma}\big\} = \big\{\mathbf{x}_i\widehat{\boldsymbol\Sigma}_{\boldsymbol\beta}\mathbf{x}_i' + 2g(t_{ui})\mathbf{z}_i\widehat{\boldsymbol\Sigma}_{\boldsymbol\beta,\boldsymbol\gamma}\mathbf{x}_i' + g(t_{ui})^2\mathbf{z}_i\widehat{\boldsymbol\Sigma}_{\boldsymbol\gamma}\mathbf{z}_i'\big\}^{\frac12}$$

where $\widehat{\boldsymbol\Sigma}_{\boldsymbol\beta}$ and $\widehat{\boldsymbol\Sigma}_{\boldsymbol\gamma}$ are the estimated covariance matrices for $\widehat{\boldsymbol\beta}$ and $\widehat{\boldsymbol\gamma}$, respectively, and $\widehat{\boldsymbol\Sigma}_{\boldsymbol\beta,\boldsymbol\gamma}$ is the estimated covariance matrix between $\widehat{\boldsymbol\beta}$ and $\widehat{\boldsymbol\gamma}$.

Predictions for multiple-record interval-censored data

For multiple-record interval-censored data, subject $i = 1, \ldots, n$ may contain n_i observations. And each observation $j = 1, \ldots, n_i$ for subject i consists of $(t_{ij}, \delta_{ij}, \mathbf{x}_{ij})$, where t_{ij} is the examination time, δ_{ij} denotes whether the event of interest occurs between $t_{i(j-1)}$ and t_{ij} ($t_{i(j-1)} = 0$ if $j = 1$), and $\mathbf{x}_{ij}$ records covariate values at t_{ij}. Here $\mathbf{x}_{ij}$ includes both baseline and time-varying covariates. For baseline covariates, $\mathbf{x}_{ij} = \mathbf{x}_i$. Without loss of generality, we assume that the examination times are sorted within the same subject. Let $t_1 < \cdots < t_m$ be the distinct observed time points for all $t_{li} > 0$ and $t_{ui} < \infty$, where t_{li} and t_{ui} define the observed time interval, based on (t_{ij}, δ_{ij}), that bracket the event of interest for subject i. The survivor function for subject i at the examination time t_{ij} can be estimated as

$$\widehat{S}_i(t_{ij}) = \exp\big\{ -\sum_{t_k \le t_{ij}} \widehat{h}_k \exp(\mathbf{x}_{ik}^*\widehat{\boldsymbol\beta}) \big\}$$

where $\widehat{h}_k$ ($k = 1, \ldots, m$) are the estimates of the baseline hazard jumps at $t_1, \ldots, t_m$ and $\mathbf{x}_{ik}^*$ are the covariates' values for subject i at time t_k. For baseline covariates, $\mathbf{x}_{ik}^* = \mathbf{x}_{ij} = \mathbf{x}_i$. For time-varying covariates, if t_k happens to be t_{ij}, then $\mathbf{x}_{ik}^* = \mathbf{x}_{ij}$; otherwise, $\mathbf{x}_{ik}^*$ is imputed using the imputation rule specified in the `tvcovimpute()` option of `stintcox`.

`predict` *newvar*, *statistic* may be used after `stintcox` to predict various quantities for each observation, according to each following *statistic*:

`hr`:

$$newvar_{ij} = \exp(\mathbf{x}_{ij}\widehat{\boldsymbol\beta})$$

`xb`:

$$newvar_{ij} = \mathbf{x}_{ij}\widehat{\boldsymbol\beta}$$

`stdp`:

$$newvar_{ij} = \widehat{\mathrm{se}}(\mathbf{x}_{ij}\widehat{\boldsymbol\beta})$$

`mgale`:

$$newvar_{ij} = \frac{\widehat{S}_i(t_{i(j-1)}) \log \widehat{S}_i(t_{i(j-1)}) - \widehat{S}_i(t_{ij}) \log \widehat{S}_i(t_{ij})}{\widehat{S}_i(t_{i(j-1)}) - \widehat{S}_i(t_{ij})}$$

`basehc`:

$$newvar_{ij} = I(t_k = t_{ij})\widehat{h}_k$$

`basechazard`:

$$newvar_{ij} = \widehat{H}_0(t_{ij}) = \sum_{k:t_k \leq t_{ij}} \widehat{h}_k$$

`basesurv`:

$$newvar_{ij} = \widehat{S}_0(t_{ij}) = \exp\left\{-\widehat{H}_0(t_{ij})\right\}$$

`csnell`:

$$newvar_{ij} = -\log \widehat{S}_i(t_{ij})$$

When the `tvc()` option is specified with `stintcox`, `predict` computes the linear prediction xb as $\mathbf{x}_{ij}\widehat{\boldsymbol{\beta}} + g(t_{ij})\mathbf{z}_{ij}\widehat{\boldsymbol{\gamma}}$ instead of $\mathbf{x}_{ij}\widehat{\boldsymbol{\beta}}$. Here $g(t)$ denotes the function of time specified in the `texp()` option of `stintcox`, and z denotes the covariates specified in the `tvc()` option of `stintcox`. The survivor function is then estimated as

$$\widehat{S}_i(t_{ij}) = \exp\left[-\sum_{t_k \leq t_{ij}} \widehat{h}_k \exp\{\mathbf{x}_{ik}^*\widehat{\boldsymbol{\beta}} + g(t_k)\mathbf{z}_{ik}^*\widehat{\boldsymbol{\gamma}}\}\right]$$

where $\mathbf{z}_{ik}^*$ are defined similarly to $\mathbf{x}_{ik}^*$ above but using the covariates $\mathbf{z}_{ij}$'s.

Survivor curves for interval-censored data

You can use the `stcurve` command to plot the survivor function after `stintcox`. `stcurve` can also plot the failure, hazard, or cumulative hazard function.

Single-record interval-censored data. By default, `stcurve` plots the estimated survivor function with covariates set to their mean values,

$$\widehat{S}(t) = \exp\left\{-\sum_{t_k \leq t} \widehat{h}_k \exp(\overline{\mathbf{x}}\widehat{\boldsymbol{\beta}})\right\} = \widehat{S}_0(t)^{\exp(\overline{\mathbf{x}}\widehat{\boldsymbol{\beta}})}$$

where $\overline{\mathbf{x}}$ is a $1 \times p$ mean vector of x. When the `tvc()` option is specified with `stintcox`, the plotted survivor function becomes

$$\widehat{S}(t) = \exp\left[-\sum_{t_k \leq t} \widehat{h}_k \exp\{\overline{\mathbf{x}}\widehat{\boldsymbol{\beta}} + g(t_k)\overline{\mathbf{z}}\widehat{\boldsymbol{\gamma}}\}\right]$$

where $\overline{\mathbf{z}}$ is a $1 \times q$ mean vector of z. Alternatively, you can specify your own values in `stcurve`'s `at()` option. For covariates that are not specified in the `at()` option, the mean values are used. For details about the `at()` option, see *Syntax of at()* in [ST] **adjustfor_option**.

Multiple-record interval-censored data. By default, stcurve plots the estimated survivor function with covariates set to their overall mean values:

$$\widehat{S}(t) = \exp\Big\{ -\sum_{t_k \leq t} \widehat{h}_k \exp(\overline{\mathbf{x}}\widehat{\boldsymbol{\beta}}) \Big\} = \widehat{S}_0(t)^{\exp(\overline{\mathbf{x}}\widehat{\boldsymbol{\beta}})}$$

And with the tvc() option of stintcox, the estimated survivor function is computed by default as

$$\widehat{S}(t) = \exp\Big[-\sum_{t_k \leq t} \widehat{h}_k \exp\{\overline{\mathbf{x}}\widehat{\boldsymbol{\beta}} + g(t_k)\overline{\mathbf{z}}\widehat{\boldsymbol{\gamma}}\} \Big]$$

Let $\overline{\mathbf{x}}_{t_k}$ be a $1 \times p$ mean vector of $\mathbf{x}_{ik}^*$ over all subjects at time t_k and $\overline{\mathbf{z}}_{t_k}$ be a $1 \times q$ mean vector of $\mathbf{z}_{ik}^*$ over all subjects at time t_k (if the tvc() option is specified with stintcox). When the attmeans option is specified, stcurve plots the following estimated survivor function:

$$\widehat{S}(t) = \exp\Big[-\sum_{t_k \leq t} \widehat{h}_k \exp\{\overline{\mathbf{x}}_{t_k}\widehat{\boldsymbol{\beta}} + g(t_k)\overline{\mathbf{z}}_{t_k}\widehat{\boldsymbol{\gamma}}\} \Big]$$

If stcurve's atframe() option is specified, $\overline{\mathbf{x}}_{t_k}$ and $\overline{\mathbf{z}}_{t_k}$ in the above will be replaced with $\mathbf{x}_{t_k}^{\text{new}}$ and $\mathbf{z}_{t_k}^{\text{new}}$, where $\mathbf{x}_{t_k}^{\text{new}}$ and $\mathbf{z}_{t_k}^{\text{new}}$ are user-specified values if time t_k and the corresponding covariate values are recorded in the frame. Otherwise, $\mathbf{x}_{t_k}^{\text{new}}$ and $\mathbf{z}_{t_k}^{\text{new}}$ are imputed using the imputation rule specified in the tvcovimpute() option of stintcox. Then the survivor function can be estimated as

$$\widehat{S}(t) = \exp\Big[-\sum_{t_k \leq t} \widehat{h}_k \exp\{\mathbf{x}_{t_k}^{\text{new}}\widehat{\boldsymbol{\beta}} + g(t_k)\mathbf{z}_{t_k}^{\text{new}}\widehat{\boldsymbol{\gamma}}\} \Big]$$

The failure, hazard, and cumulative hazard functions are calculated based on the survivor function above.

References

Cox, D. R., and E. J. Snell. 1968. A general definition of residuals (with discussion). *Journal of the Royal Statistical Society, Series B* 30: 248–275. https://doi.org/10.1111/j.2517-6161.1968.tb00724.x.

Farrington, C. P. 2000. Residuals for proportional hazards models with interval-censored survival data. *Biometrics* 56: 473–482. https://doi.org/10.1111/j.0006-341X.2000.00473.x.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Also see

[ST] **stintcox** — Cox proportional hazards model for interval-censored survival-time data

[ST] **stintcox PH-assumption plots** — Plots of proportional-hazards assumption after stintcox

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[U] **20 Estimation and postestimation commands**

Title

stintreg — Parametric models for interval-censored survival-time data

Description	Quick start	Menu	Syntax
Options	Remarks and examples	Stored results	Methods and formulas
References	Also see		

Description

stintreg fits parametric models to survival-time data or, more precisely, event-time data that can be uncensored, right-censored, left-censored, or interval-censored. These models are generalizations of the models fit by streg to support interval-censored data. The supported survival models are exponential, Weibull, Gompertz, lognormal, loglogistic, and generalized gamma. Proportional-hazards (PH) and accelerated failure-time (AFT) parameterizations are provided.

With interval-censored data, the event-time variables are specified with the stintreg command instead of using stset. Any st settings are ignored by stintreg.

Quick start

Weibull survival model with covariates x1 and x2 fit to interval-censored event-time data with lower and upper endpoints t1 and t2

    stintreg x1 x2, interval(t1 t2) distribution(weibull)

Use AFT metric instead of PH metric

    stintreg x1 x2, interval(t1 t2) distribution(weibull) time

Different intercepts and ancillary parameters for strata identified by svar

    stintreg x1 x2, interval(t1 t2) distribution(weibull) strata(svar)

Lognormal survival model

    stintreg x1 x2, interval(t1 t2) distribution(lognormal)

Same as above, but also model the logarithm of ancillary parameter as the linear combination of covariates z1 and z2

    stintreg x1 x2, interval(t1 t2) distribution(lognormal) ///
        ancillary(z1 z2)

Menu

Statistics > Survival analysis > Regression models > Interval-censored parametric survival models

Syntax

> stintreg [*indepvars*] [*if*] [*in*] [*weight*] , <u>inter</u>val(t_l t_u) <u>dist</u>ribution(*distname*)
>
> [*options*]

options	Description
Model	
* <u>inter</u>val(t_l t_u)	lower and upper endpoints for the [event-time interval]
<u>nocons</u>tant	suppress constant term
* <u>dist</u>ribution(*distname*)	specify survival distribution
time	use accelerated failure-time metric
Model 2	
strata(*varname*)	strata ID variable
<u>off</u>set(*varname*)	include *varname* in model with coefficient constrained to 1
ancillary(*varlist*)	use *varlist* to model the first ancillary parameter
anc2(*varlist*)	use *varlist* to model the second ancillary parameter
constraints(*constraints*)	apply specified linear constraints
epsilon(#)	tolerance to treat observations as uncensored; default is epsilon(1e-6)
SE/Robust	
vce(*vcetype*)	*vcetype* may be oim, robust, <u>cl</u>uster *clustvar*, opg, <u>boot</u>strap, or <u>jack</u>knife
Reporting	
level(#)	set confidence level; default is level(95)
nohr	do not report hazard ratios
<u>tr</u>atio	report time ratios
<u>nohe</u>ader	suppress header from coefficient table
<u>nocns</u>report	do not display constraints
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Maximization	
maximize_options	control the maximization process; seldom used
collinear	keep collinear variables
<u>coefl</u>egend	display legend instead of statistics

*interval(t_l t_u) and distribution(*distname*) are required.

distname	Description
exponential	exponential survival distribution
gompertz	Gompertz survival distribution
loglogistic	loglogistic survival distribution
llogistic	synonym for loglogistic
weibull	Weibull survival distribution
lognormal	lognormal survival distribution
lnormal	synonym for lognormal
ggamma	generalized gamma survival distribution

varlist may contain factor variables; see [U] **11.4.3 Factor variables**.

bootstrap, by, collect, fp, jackknife, nestreg, statsby, stepwise, and svy are allowed; see [U] **11.1.10 Prefix commands**.

Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.

vce(), noheader, and weights are not allowed with the svy prefix; see [SVY] **svy**.

fweights, iweights, and pweights are allowed; see [U] **11.1.6 weight**.

collinear and coeflegend do not appear in the dialog box.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

> [Model]

interval(t_l t_u) specifies two time variables that contain the endpoints of the event-time interval. t_l represents the lower endpoint, and t_u represents the upper endpoint. interval() is required.

The interval time variables t_l and t_u should have the following form:

Type of observations		t_l	t_u
uncensored	$a = [\,a, a\,]$	a	a
interval-censored	$(\,a, b\,]$	a	b
left-censored	$(\,0, b\,]$	.	b
left-censored	$(\,0, b\,]$	0	b
right-censored	$(\,a, +\infty\,)$	a	.
missing		.	.
missing		0	.

noconstant; see [R] **Estimation options**.

distribution(*distname*) specifies the survival model to be fit. distribution() is required.

time specifies that the model be fit in the accelerated failure-time metric rather than in the log relative-hazard metric or proportional hazards metric. This option is valid only for the exponential and Weibull models, because these are the only models that have both a proportional hazards and an accelerated failure-time parameterization. Regardless of metric, the likelihood function is the same, and models are equally appropriate viewed in either metric; it is just a matter of changing the interpretation.

Model 2

strata(*varname*) specifies the stratification ID variable. Observations with equal values of the variable are assumed to be in the same stratum. Stratified estimates (with equal coefficients across strata but intercepts and ancillary parameters distinct for each stratum) are then obtained. *varname* may be a factor variable; see [U] **11.4.3 Factor variables**.

offset(*varname*); see [R] **Estimation options**.

ancillary(*varlist*) specifies that the ancillary parameter for the Weibull, lognormal, Gompertz, and loglogistic distributions and that the first ancillary parameter (σ) of the generalized log-gamma distribution be estimated as a linear combination of *varlist*.

When an ancillary parameter is constrained to be strictly positive, the logarithm of the ancillary parameter is modeled as a linear combination of *varlist*.

anc2(*varlist*) specifies that the second ancillary parameter (κ) for the generalized log-gamma distribution be estimated as a linear combination of *varlist*.

constraints(*constraints*); see [R] **Estimation options**.

epsilon(*#*) specifies that observations with $t_u - t_l < \#$ be treated as uncensored. The default is epsilon(1e-6).

SE/Robust

vce(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory (oim, opg), that are robust to some kinds of misspecification (robust), that allow for intragroup correlation (cluster *clustvar*), and that use bootstrap or jackknife methods (bootstrap, jackknife); see [R] *vce_option*.

Reporting

level(*#*); see [R] **Estimation options**.

nohr, which may be specified at estimation or upon redisplaying results, specifies that coefficients rather than exponentiated coefficients be displayed, that is, that coefficients rather than hazard ratios be displayed. This option affects only how coefficients are displayed, not how they are estimated.

This option is valid only for models with a natural proportional hazards parameterization: exponential, Weibull, and Gompertz. These three models, by default, report hazard ratios (exponentiated coefficients).

tratio specifies that exponentiated coefficients, which are interpreted as time ratios, be displayed. tratio is appropriate only for the loglogistic, lognormal, and generalized gamma models, or for the exponential and Weibull models when fit in the accelerated failure-time metric.

tratio may be specified at estimation or upon replay.

noheader suppresses the output header, either at estimation or upon replay.

nocnsreport; see [R] **Estimation options**.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(*#*), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

maximize_options: <u>difficult</u>, <u>technique</u>(*algorithm_spec*), <u>iterate</u>(*#*), [no]<u>log</u>, <u>trace</u>,
   <u>gradient</u>, <u>showstep</u>, <u>hessian</u>, <u>showtolerance</u>, <u>tolerance</u>(*#*), <u>ltolerance</u>(*#*),
   <u>nrtole</u>rance(*#*), <u>nonrtol</u>erance, and <u>from</u>(*init_specs*); see [R] **Maximize**. These options are
   seldom used.

   Setting the optimization type to technique(bhhh) resets the default *vcetype* to vce(opg).

The following options are available with stintreg but are not shown in the dialog box:

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

   Remarks are presented under the following headings:

Introduction

   stintreg fits parametric models to survival-time data, which can be uncensored, right-censored,
left-censored, or interval-censored. These models are generalizations of the models fit by streg,
because they extend the censoring mechanism beyond right-censoring. In the context of interval-
censored data, the term "failure-time" or, more generally, "event-time data" is more appropriate than
"survival-time data", so we will use it from now on.

   In survival analysis, we find different types of censored data. Among them, right-censored data have
been studied extensively and can be analyzed using all of Stata's survival commands, including streg
and stcox. Research on interval-censored data has also been popular; see, for example, Finkelstein
and Wolfe (1985), Odell, Anderson, and D'Agostino (1992), Rabinowitz, Tsiatis, and Aragon (1995),
Huang and Wellner (1997), Lindsey (1998), Lindsey and Ryan (1998), Sun (2006), and Sun and
Li (2014). Interval-censoring occurs when the failure time of interest is not exactly observed but is
known only to lie within some interval (for example, Kalbfleisch and Prentice 2002). Uncensored,
right-censored, and left-censored data are special cases of interval-censored data. In these cases, the
interval reduces to a single point, is unbounded on the right, or is bounded by zero on the left.

   Interval-censored event-time data arise in many areas including medical, epidemiological, financial,
and sociological studies. A study may lead to event-time data with different types of censoring.
Consider a medical study that involves periodic follow-ups with patients who had breast cancer. In
this case, patients are tested on a regular basis, but the time to the recurrence of the cancer may not
be measured exactly. If cancer recurs before the first visit, the observation is called left-censored. If
cancer recurs between two visits, the observation is called interval-censored. If there is no recurrence
by the last visit, the observation is right-censored. To analyze such data, you may fit parametric
survival models using stintreg.

   Regardless of the type of censoring, stintreg requires the survival outcome to be stored in the
dataset as interval data. That is, two time variables, t_l and t_u, that contain the endpoints of the
event-time interval must be specified in the interval() option. If the data are left-censored, the
lower endpoint is zero and may be represented in t_l by either a missing value (.) or zero. If the

data are right-censored, the upper endpoint is $+\infty$ and is represented in t_u by a missing value. Uncensored data are represented by the two endpoints that are equal. If $0 < t_l < t_u < \infty$, the data are interval-censored. Truly missing values must be represented by missing values in both t_l and t_u or by a 0 in t_l and a missing value in t_u. Typing stset is unnecessary, and stintreg will ignore any settings of stset for the usual trivariate response variable (t_0, t, d). stintreg does not support data exhibiting delayed entry, gaps, time-varying covariates, and multiple failures.

Two often-used parametric models for adjusting survivor functions for the effects of covariates are the AFT models and the multiplicative or PH models. The survival models supported by stintreg are exponential, Weibull, Gompertz, lognormal, loglogistic, and generalized gamma. The lognormal, loglogistic, and generalized gamma models are implemented as AFT models. The exponential and Weibull models are implemented as both AFT and PH models, and the Gompertz model is implemented only in the PH metric. See *Remarks and examples* in [ST] **streg** for more details about the supported models and distributions.

Types of interval-censoring

Interval-censoring can occur in different forms, and each form represents one type of interval-censored event-time data. stintreg accommodates two important types of interval-censored data that are commonly used in practice: case II interval-censored data and case I interval-censored data. Case II interval-censored data are also referred to as general interval-censored data, and case I interval-censored data are also referred to as current status data. We describe each censoring type in detail below. Also see Sun (2006) for more information about different types of interval censoring.

Case II interval-censored data

The most general case of interval-censoring is case II interval-censoring. This type of interval-censoring occurs when we do not know the exact failure time t, but only know that the failure happened within a random time interval $(t_l, t_u]$, or before the right endpoint of the time interval t_u, or after the left endpoint of the time interval t_l. The following is an example of case II interval-censored data, which contain left-, right-, and interval-censored observations.

▷ Example 1: Case II interval-censoring

Sun (2006) presented parametric analysis of a retrospective study of early breast cancer patients, originally from Finkelstein and Wolfe (1985), that compared the cosmetic effects of two cancer treatments: radiotherapy alone versus radiotherapy plus adjuvant chemotherapy. There were 46 radiotherapy-only patients and 48 radiation-plus-chemotherapy patients who were observed every four to six months. Patients had different visit times and durations between visits. At each visit, the physician recorded a measure of breast retraction. The event of interest was breast retraction. Because patients were observed at random follow-up times, the exact time of breast retraction was not observed and was known only to fall in the interval between visits. The data consist of two interval variables, ltime and rtime, that represent the last clinic visit time when breast retraction had not yet occurred and the first clinic visit time when breast retraction was detected.

To study the effect of treatment on breast retraction, we fit a Weibull model of time to breast retraction on treatment treat using stintreg. Unlike streg, in which the survival variables are set using stset and do not appear in the command, the interval variables ltime and rtime are required for stintreg and are specified in the interval() option:

```
. use https://www.stata-press.com/data/r18/cosmesis
(Cosmetic deterioration of breast cancer patients)

. stintreg i.treat, interval(ltime rtime) distribution(weibull)

Fitting constant-only model:
Iteration 0:   Log likelihood = -200.17506
Iteration 1:   Log likelihood = -175.09443
Iteration 2:   Log likelihood = -155.03621
Iteration 3:   Log likelihood = -149.07347
Iteration 4:   Log likelihood =  -148.7602
Iteration 5:   Log likelihood = -148.65639
Iteration 6:   Log likelihood = -148.65584
Iteration 7:   Log likelihood = -148.65584

Fitting full model:
Iteration 0:   Log likelihood = -148.65584
Iteration 1:   Log likelihood = -143.53903
Iteration 2:   Log likelihood = -143.19322
Iteration 3:   Log likelihood = -143.19228
Iteration 4:   Log likelihood = -143.19228
```

Weibull PH regression

					Number of obs	=	94
					Uncensored	=	0
					Left-censored	=	5
					Right-censored	=	38
					Interval-cens.	=	51
					LR chi2(1)	=	10.93

Log likelihood = -143.19228   Prob > chi2   = 0.0009

	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
treat						
Radio+Chemo	2.498526	.7069467	3.24	0.001	1.434961	4.350383
_cons	.0018503	.0013452	-8.66	0.000	.000445	.007693
/ln_p	.4785786	.1198972	3.99	0.000	.2435844	.7135729
p	1.613779	.1934876			1.275814	2.041271
1/p	.6196635	.0742959			.4898907	.7838134

Note: _cons estimates baseline hazard.

The header above the coefficient table summarizes censored and uncensored observations. There are 38 patients who did not experience breast retraction by the last visit, resulting in right-censored observations. There are 5 patients who had breast retraction before their first follow-up, resulting in left-censored observations. There are no uncensored observations, so the remaining 51 observations are interval-censored.

By default, the hazard ratios are reported instead of the natural coefficients. The estimated hazard ratio of the radiotherapy plus chemotherapy is approximately 2.5 with a 95% confidence interval of $[1.435, 4.350]$, indicating significantly higher risk to develop breast retraction using this treatment than radiotherapy only. In other words, the adjuvant chemotherapy increases the risk of breast retraction. The shape parameter is estimated as $\ln(p)$, but p and $1/p = \sigma$ are also reported. The estimated p is greater than 1, indicating that the hazard of breast retraction increases with time.

By default, `stintreg` uses the PH parameterization for the Weibull model, but we can specify the `time` option to request the AFT parameterization.

```
. stintreg i.treat, interval(ltime rtime) distribution(weibull) time
```

(iteration log omitted)

```
Weibull AFT regression                        Number of obs     =       94
                                                 Uncensored =        0
                                              Left-censored =        5
                                             Right-censored =       38
                                             Interval-cens. =       51
                                              LR chi2(1)        =    10.93
Log likelihood = -143.19228                   Prob > chi2       =   0.0009
```

	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
treat						
Radio+Chemo	-.5674261	.175814	-3.23	0.001	-.9120151	-.2228371
_cons	3.899163	.1405986	27.73	0.000	3.623595	4.174731
/ln_p	.4785789	.119897	3.99	0.000	.2435851	.7135726
p	1.613779	.1934873			1.275815	2.041271
1/p	.6196634	.0742958			.4898909	.7838128

With the AFT parameterization, coefficients are reported by default, but we can use the `tratio` option to display time ratios.

◁

▷ Example 2: Comparing distributions

To compare different models, let's fit the model from example 1 but use the generalized gamma distribution instead. The hazard function of the generalized gamma distribution is extremely flexible, allowing for many different shapes. Weibull, exponential, and lognormal distributions are all special cases of the generalized gamma distribution. Therefore, we can use the generalized gamma model to evaluate and select an appropriate parametric model for the data. When $\kappa = 0$, the generalized gamma model reduces to the lognormal model. When $\kappa = 1$, the generalized gamma model reduces to the Weibull model.

```
. stintreg i.treat, interval(ltime rtime) distribution(ggamma)
```
(*iteration log omitted*)

```
Generalized gamma AFT regression                  Number of obs   =       94
                                                       Uncensored =        0
                                                   Left-censored =        5
                                                  Right-censored =       38
                                                  Interval-cens. =       51

                                                  LR chi2(1)      =    11.26
Log likelihood = -142.71767                       Prob > chi2     =   0.0008
```

	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
treat						
Radio+Chemo	-.5696386	.1686355	-3.38	0.001	-.9001581	-.2391192
_cons	4.009316	.1721275	23.29	0.000	3.671952	4.346679
/lnsigma	-.7016455	.2793936	-2.51	0.012	-1.249247	-.1540441
/kappa	1.532208	.6176602	2.48	0.013	.321616	2.7428
sigma	.4957689	.1385146			.2867207	.8572342

The Wald test of $H_0: \kappa = 0$ is reported in the output above. The p-value is 0.013, indicating that the lognormal model is not appropriate. We can test the hypothesis that $\kappa = 1$ using the `test` command:

```
. test /kappa = 1
 ( 1)   [/]kappa = 1

           chi2(  1) =     0.74
         Prob > chi2 =    0.3889
```

The above Wald test of $H_0: \kappa = 1$ has a p-value of 0.39, suggesting that the Weibull model may be appropriate for these data.

◁

Case I interval-censored data

Case I interval-censored data arise when the only survival information available is whether the event of interest occurred before or after the observed time, leading to data in which an observation is either left-censored or right-censored. As such, case I interval-censored data can be viewed as a special case of case II interval-censored data without uncensored and interval-censored on $(a, b]$ observations. Case I interval-censored data occur when subjects are observed only once, and thus we can only know whether the event had already happened before we observed them. Such data are common in demographical studies, where they are also known as current status data. In addition to demographical studies, case I interval-censored data occur in other fields including epidemiological studies, cross-sectional studies, and tumorigenicity experiments. See Huang and Wellner (1997) and Sun (2006) for more information.

The `stintreg` command requires that case I interval-censored data are recorded by two interval time variables that identify which observations are left-censored and which observations are right-censored.

▷ Example 3: Case I interval-censoring

We consider the data from Hoel and Walburg (1972) on nonlethal lung tumors for 144 male mice. The death time of each mouse (`death`) and an indicator of whether the lung tumor was present by the time of death (`status`) were reported. The type of environment (`group`) in which those mice lived, either conventional environment (`CE`) or germ-free environment (`GE`), was also reported. The goal of this study was to test whether different types of environment had influence on the time of tumor onset for those mice. The lung tumor was known to be nonlethal for the mice. Therefore, the tumor onset time could not be directly observed. The only available information was the observed death time and whether or not the tumor was detected at the time of death.

```
. use https://www.stata-press.com/data/r18/lungtumor
(Lung tumor data for mice)
. table group status
```

	Tumor status		
	No tumor	With tumor	Total
Environment			
CE	69	27	96
GE	13	35	48
Total	82	62	144

```
. list in 26/30
```

	group	status	death
26.	CE	With tumor	811
27.	CE	With tumor	839
28.	CE	No tumor	45
29.	CE	No tumor	198
30.	CE	No tumor	215

Case I interval-censored data are often stored as shown above: each subject has one variable that represents the observation time and one variable that represents the status of the event of interest. To use `stintreg`, we must create two time variables to contain the lower and upper endpoints of the intervals. Because case I interval-censored data are either left-censored or right-censored, we first create two new variables, `ltime` and `rtime`, that are both equal to the observation time, `death`, then replace the lower endpoint, `ltime`, with a missing value if the tumor was detected (`status` $= 1$) and replace the upper endpoint, `rtime`, with a missing value if the tumor was not detected (`status` $= 0$).

```
. generate ltime = death
. generate rtime = death
. replace ltime = . if status == 1
(62 real changes made, 62 to missing)
. replace rtime = . if status == 0
(82 real changes made, 82 to missing)
. list in 26/30
```

	group	status	death	ltime	rtime
26.	CE	With tumor	811	.	811
27.	CE	With tumor	839	.	839
28.	CE	No tumor	45	45	.
29.	CE	No tumor	198	198	.
30.	CE	No tumor	215	215	.

Now, we fit the model using the exponential distribution.

```
. stintreg i.group, interval(ltime rtime) distribution(exponential)
(iteration log omitted)
```

Exponential PH regression

```
Number of obs     =      144
       Uncensored =        0
    Left-censored =       62
   Right-censored =       82
   Interval-cens. =        0
```

```
Log likelihood = -81.325875
```

```
LR chi2(1)        =    16.09
Prob > chi2       =   0.0001
```

	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
group						
GE	2.90202	.7728318	4.00	0.000	1.721942	4.890828
_cons	.0005664	.0001096	-38.63	0.000	.0003876	.0008277

Note: **_cons** estimates baseline hazard.

The estimated hazard for the mice in the germ-free environment is approximately three times the hazard for those in the conventional environment. In other words, the mice in the germ-free environment had higher lung tumor incidence than those in the conventional environment.

◁

Parameterization of ancillary parameters

stintreg's ancillary() and anc2() options allow us to parameterize ancillary parameters in terms of covariates. By default, all ancillary parameters are estimated as being constant. By specifying, for example,

```
. stintreg x1 x2, interval(ltime rtime) distribution(weibull) ancillary(z1 z2)
```

the logarithm of the ancillary parameter p is modeled using the linear predictor of z1 and z2. The anc2() option models the second ancillary parameter κ for the generalized log-gamma distribution.

▷ Example 4: Modeling the ancillary parameters

Consider the data described in table 2.3 of Sun (2006) (originally from Richman, Grimes, and Lagakos 1990) on times to resistance to the drug zidovudine for AIDS patients. Covariates of interest are the stage of the disease, stage (0 = early stage, 1 = late stage) and the dose level of the treatment, dose (0 = low dose, 1 = high dose). The time intervals, in months, are stored in variables ltime and rtime.

To investigate whether stage has any effect on time to drug resistance, we fit the Weibull model using stintreg. To later compare results with another model that reports coefficients, we use the nohr option here to display the untransformed coefficients.

```
. use https://www.stata-press.com/data/r18/aids, clear
(Time to zidovudine resistance)

. stintreg i.stage, interval(ltime rtime) distribution(weibull) nohr

  (iteration log omitted)
```

```
Weibull PH regression                          Number of obs    =       31
                                                    Uncensored =        0
                                                 Left-censored =       15
                                                Right-censored =       13
                                                Interval-cens. =        3
                                               LR chi2(1)       =    10.02
Log likelihood = -13.27946                     Prob > chi2      =   0.0016
```

	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
1.stage	1.910652	.6604416	2.89	0.004	.6162103	3.205094
_cons	-7.952868	3.000563	-2.65	0.008	-13.83386	-2.071872
/ln_p	1.036662	.397829	2.61	0.009	.2569319	1.816393
p	2.81979	1.121794			1.292957	6.149635
1/p	.3546364	.1410846			.1626113	.7734209

Out of the 31 patients, 13 patients are right-censored, 15 patients are left-censored, and only 3 patients are interval-censored. The estimated coefficient for patients in their late stage of the disease is 1.91; their hazard of resisting zidovudine is approximately $\exp(1.91) = 6.75$ times the hazard for patients in their early stage.

Suppose we believe that the hazards for different dose levels have different shape parameters. We can accommodate this by specifying the ancillary() option.

```
. stintreg i.stage, interval(ltime rtime) distribution(weibull) ancillary(i.dose)
note: option nohr is implied if option strata() or ancillary() is specified.

  (iteration log omitted)
```

```
Weibull PH regression                          Number of obs    =       31
                                                    Uncensored =        0
                                                 Left-censored =       15
                                                Right-censored =       13
                                                Interval-cens. =        3
                                               LR chi2(1)       =    12.20
Log likelihood = -11.214877                    Prob > chi2      =   0.0005
```

	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
ltime						
1.stage	2.795073	1.1675	2.39	0.017	.5068147	5.083331
_cons	-10.8462	4.233062	-2.56	0.010	-19.14285	-2.549553
ln_p						
1.dose	.1655302	.0874501	1.89	0.058	-.0058689	.3369292
_cons	1.252361	.4143254	3.02	0.003	.4402978	2.064424

With the `ancillary()` option, results are displayed as coefficients by default; see the technical note below. From the above results, $\widehat{\ln(p)}_{\text{low}} = 1.25$ for patients with low dose and $\widehat{\ln(p)}_{\text{high}} = 1.25 + 0.17 = 1.42$ for patients with high dose. Thus, $\widehat{p}_{\text{low}} = 3.49$ and $\widehat{p}_{\text{high}} = 4.14$. When we combine this with the main equation in the model, the estimated hazards are

$$
\widehat{h}(t_j|\mathbf{x}_j) = \begin{cases} 3.49 \times t_j^{3.49-1} \times \exp\left(-10.85 + 2.80 \times \texttt{stage}_j\right) & \text{if dose} = 0 \\ 4.14 \times t_j^{4.14-1} \times \exp\left(-10.85 + 2.80 \times \texttt{stage}_j\right) & \text{if dose} = 1 \end{cases}
$$

◁

❑ Technical note

When fitting PH models, `stintreg`, by default, displays hazard ratios. If the `strata()` option or the `ancillary()` option (as in our previous example) is specified, `stintreg` reports coefficients instead. If either of these options is specified, ancillary parameters are no longer constant and are modeled as a function of covariates specified in those options. If any of the covariates from the main equation are used to model ancillary parameters, hazard ratios lose their interpretation. As a precaution, `stintreg` always displays results as coefficients when those options are used. If we want to compare results with PH models with constant ancillary parameters, we can use the `nohr` option to display coefficients.

The above argument also applies to time ratios when fitting AFT models. For this reason, the `tratio` option is not allowed with AFT models whenever `strata()`, `ancillary()`, or `anc2()` is specified.

❑

Stratified estimation

We can fit a stratified model by specifying the `strata(`*varname*`)` option. A stratified model means that the coefficients on the covariates are the same across strata, but the intercept and ancillary parameters are allowed to vary for each level of the strata variable.

▷ Example 5: Fitting a stratified model

Continuing with example 4, suppose that we believe that `dose` affects both the scale and shape of the hazard, and the effect of `stage` is the same for each level of `dose`. We refit the Weibull model, but now we also stratify on `dose`:

```
. stintreg i.stage, interval(ltime rtime) distribution(weibull) strata(dose)
note: option nohr is implied if option strata() or ancillary() is specified.
```

  (*iteration log omitted*)

```
Weibull PH regression                          Number of obs     =       31
                                                    Uncensored =        0
                                                 Left-censored =       15
                                                Right-censored =       13
                                                Interval-cens. =        3
                                               LR chi2(2)        =    12.40
Log likelihood = -11.115197                    Prob > chi2       =   0.0020
```

	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
ltime						
1.stage	2.711532	1.084146	2.50	0.012	.5866441	4.83642
1.dose	-2.661873	5.883975	-0.45	0.651	-14.19425	8.870506
_cons	-9.143	4.930796	-1.85	0.064	-18.80718	.5211833
ln_p						
1.dose	.4538942	.6700991	0.68	0.498	-.8594759	1.767264
_cons	1.051935	.6190549	1.70	0.089	-.1613906	2.26526

The indicator for level 1 of `dose` is added to the main equation and to the ancillary equation; level 0 is the baseline and is modeled by the constant terms.

  Note that the specification above is the same as fitting the following model:

```
stintreg i.stage i.dose, interval(ltime rtime) distribution(weibull) ///
    ancillary(i.dose)
```

◁

  By using `ancillary()` or `strata()`, we may fit a wide variety of models; see *Stratified estimation* in [ST] **streg** for details. These models may be compared using Wald or likelihood-ratio tests when the models in question are nested or by using the AIC for nonnested models. Modeling of ancillary parameters and stratification is also available for AFT models.

Stored results

  `stintreg` stores the following in `e()`:

Scalars
e(N)	number of observations
e(N_unc)	number of uncensored observations
e(N_lc)	number of left-censored observations
e(N_rc)	number of right-censored observations
e(N_int)	number of interval-censored observations
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_aux)	number of auxiliary parameters
e(k_dv)	number of dependent variables
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(ll_0)	log likelihood, constant-only model
e(N_clust)	number of clusters
e(chi2)	χ^2
e(aux_p)	ancillary parameter (weibull)
e(gamma)	ancillary parameter (gompertz, loglogistic)

e(sigma)	ancillary parameter (ggamma, lnormal)
e(kappa)	ancillary parameter (ggamma)
e(epsilon)	tolerance for uncensored observations
e(p)	p-value for model test
e(rank)	rank of e(V)
e(rank0)	rank of e(V), constant-only model
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros
e(cmd)	model or regression name
e(cmd2)	stintreg
e(cmdline)	command as typed
e(depvar)	names of time interval variables specified in interval()
e(distribution)	distribution
e(strata)	stratum variable
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(wtype)	weight type
e(wexp)	weight expression
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(frm2)	hazard or time
e(chi2type)	Wald or LR; type of model χ^2 test
e(offset1)	offset for main equation
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(predict_sub)	predict subprogram
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices
e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions
e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices
r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Methods and formulas are presented under the following headings:

> Introduction
> Distributions and parameterizations
> Parameter estimation using interval-censored data

Introduction

Consider event-time data that consists of n independent observations. Let t_j represent the survival time for the event of interest for observation j, $j = 1, \ldots, n$.

For a given survivor function, $S(t)$, the density function is obtained as

$$f(t) = -\frac{d}{dt} S(t)$$

and the hazard function (the instantaneous rate of failure) is obtained as

$$h(t) = \frac{f(t)}{S(t)} = -\frac{\log S(t)}{dt}$$

Let $\mathbf{x}_j$ denote a vector of covariates for observation j, and let $\boldsymbol{\beta}$ denote a vector of regression coefficients. Let $S_j(t) = S(t|\mathbf{x} = \mathbf{x}_j)$ be the covariate-adjusted survivor function and similarly define $h_j(t)$ and $f_j(t)$.

stintreg supports six survival distributions: exponential, Weibull, Gompertz, lognormal, loglogistic, and generalized gamma; and two parameterizations for the effects of covariates: PH and AFT. The parameterization and ancillary parameters for each distribution are summarized in table 1 below.

Distributions and parameterizations

The PH model assumes that the hazard function has the form

$$h_j(t) = h_0(t) \exp\left(\mathbf{x}_j \boldsymbol{\beta}\right)$$

for some baseline hazard function $h_0(t)$. For the **stintreg** command, $h_0(t)$ is assumed to be parametric and the supported distributions are exponential, Weibull, and Gompertz. This model specifies that the covariates have a multiplicative effect on the hazard function. The covariate-adjusted survivor function $S_j(t)$ is obtained as

$$S_j(t) = \{S_0(t)\}^{\exp(\mathbf{x}_j \boldsymbol{\beta})}$$

where the baseline survivor function $S_0(t) = \exp\{-\int_0^t h_0(s)ds\}$.

In the AFT model, the natural logarithm of the survival time, $\log t$, is expressed as a linear function of the covariates, yielding the linear model

$$\log t_j = \mathbf{x}_j \boldsymbol{\beta} + z_j$$

where z_j is the error term with density $f()$. The distributional form of the error term determines the regression model. If we let $f()$ be the normal density, the lognormal regression model for t_j is obtained. Similarly, by letting $f()$ be the logistic density, the loglogistic regression is obtained. Setting $f()$ equal to the extreme-value density yields the exponential and the Weibull regression models. The effect of covariates is also multiplicative, but on time t_j, by a factor of $\exp(-\mathbf{x}_j\boldsymbol{\beta})$. Depending on whether this factor is greater or less than one, time is either accelerated or decelerated.

Table 1 below describes the supported survival models, their parameterizations, and the corresponding ancillary parameters.

Table 1. Parametric survival distributions supported by `stintreg`

Distribution	Metric	Survivor function	Parameterization	Ancillary parameters
Exponential	PH	$\exp(-\lambda_j t_j)$	$\lambda_j = \exp(\mathbf{x}_j\boldsymbol{\beta})$	
Exponential	AFT	$\exp(-\lambda_j t_j)$	$\lambda_j = \exp(-\mathbf{x}_j\boldsymbol{\beta})$	
Weibull	PH	$\exp(-\lambda_j t_j^p)$	$\lambda_j = \exp(\mathbf{x}_j\boldsymbol{\beta})$	p
Weibull	AFT	$\exp(-\lambda_j t_j^p)$	$\lambda_j = \exp(-p\mathbf{x}_j\boldsymbol{\beta})$	p
Gompertz	PH	$\exp\{-\lambda_j\gamma^{-1}(e^{\gamma t_j}-1)\}$	$\lambda_j = \exp(\mathbf{x}_j\boldsymbol{\beta})$	γ
Lognormal	AFT	$1-\Phi\left\{\frac{\log(t_j)-\mu_j}{\sigma}\right\}$	$\mu_j = \mathbf{x}_j\boldsymbol{\beta}$	σ
Loglogistic	AFT	$\{1+(\lambda_j t_j)^{1/\gamma}\}^{-1}$	$\lambda_j = \exp(-\mathbf{x}_j\boldsymbol{\beta})$	γ
Generalized gamma				
if $\kappa > 0$	AFT	$1-I(\gamma,u)$	$\mu_j = \mathbf{x}_j\boldsymbol{\beta}$	σ, κ
if $\kappa = 0$	AFT	$1-\Phi(z)$	$\mu_j = \mathbf{x}_j\boldsymbol{\beta}$	σ, κ
if $\kappa < 0$	AFT	$I(\gamma,u)$	$\mu_j = \mathbf{x}_j\boldsymbol{\beta}$	σ, κ

where $\Phi(z)$ is the standard normal cumulative distribution. For the generalized gamma, $\gamma = |\kappa|^{-2}$, $u = \gamma\exp(|\kappa|z)$, $I(a,x)$ is the incomplete gamma function, and $z = \text{sign}(\kappa)\{\log(t_j)-\mu_j\}/\sigma$.

Parameter estimation using interval-censored data

Suppose that t_j is not observed and that only the lower and upper endpoints of the time interval, t_{lj} and t_{uj}, where $t_j \in (t_{lj}, t_{uj}]$, are observed. `stintreg` estimates $\boldsymbol{\beta}$ and the ancillary parameters via maximum likelihood. For interval-censored observations, the log likelihood is given by

$$\log L = \sum_{j=1}^{n} \log\{S_j(t_{lj}) - S_j(t_{uj})\}$$

Implicit in the above log-likelihood expression are the regression parameters, $\boldsymbol{\beta}$, and the ancillary parameters, because both are components of the chosen $S_j(t)$; see table 1.

For case II interval-censored data, the log likelihood can be written as

$$
\log L = \sum_{j \in UC} \log f_j(t_{lj}) + \sum_{j \in RC} \log S_j(t_{lj}) + \sum_{j \in LC} \log \{1 - S_j(t_{uj})\}
$$
$$
+ \sum_{j \in IC} \log \{S_j(t_{lj}) - S_j(t_{uj})\}
$$

where the set UC contains uncensored observations, RC contains right-censored observations, LC contains left-censored observations, and IC contains interval-censored observations.

For case I interval-censored data, with only right-censored and left-censored observations, the log likelihood reduces to

$$
\log L = \sum_{j \in RC} \log S_j(t_{lj}) + \sum_{j \in LC} \log \{1 - S_j(t_{uj})\}
$$

Specifying `ancillary()`, `anc2()`, or `strata()` will parameterize the ancillary parameter(s) by using the linear predictor, $\mathbf{z}_j \boldsymbol{\alpha}_z$, where the covariates, $\mathbf{z}_j$, need not be distinct from $\mathbf{x}_j$. Here `stintreg` will report estimates of $\boldsymbol{\alpha}_z$ in addition to estimates of $\boldsymbol{\beta}$. The log likelihood here is simply the log likelihood given above, with $\mathbf{z}_j \boldsymbol{\alpha}_z$ substituted for the ancillary parameter. If the ancillary parameter is constrained to be strictly positive, its logarithm is parameterized instead; that is, we substitute the linear predictor for the logarithm of the ancillary parameter in the above log likelihood. The gamma model has two ancillary parameters, σ and κ; we parameterize σ by using `ancillary()` and κ by using `anc2()`, and the linear predictors used for each may be distinct. Specifying `strata()` includes indicator variables for the strata in the main equation, and uses them to parameterize any ancillary parameters that exist for the chosen model.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster` *clustvar*`)`, respectively. See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*. If the assumption of independence of the observations is highly questionable, this means that the conventional estimate of variance is not appropriate. We strongly advise that you use the `vce(robust)` and `vce(cluster` *clustvar*`)` options here.

`stintreg` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] **Variance estimation**.

References

Finkelstein, D. M., and R. A. Wolfe. 1985. A semiparametric model for regression analysis of interval-censored failure time data. *Biometrics* 41: 933–945. https://doi.org/10.2307/2530965.

Hoel, D. G., and H. E. Walburg, Jr. 1972. Statistical analysis of survival experiments. *Journal of the National Cancer Institute* 49: 361–372. https://doi.org/10.1093/jnci/49.2.361.

Huang, J., and J. A. Wellner. 1997. Interval censored survival data: A review of recent progress. In *Proceedings of the First Seattle Symposium in Biostatistics: Survival Analysis*, ed. D. Y. Lin and T. R. Fleming, 123–169. New York: Springer.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Lindsey, J. C., and L. M. Ryan. 1998. Methods for interval-censored data. *Statistics in Medicine* 17: 219–238. https://doi.org/10.1002/(SICI)1097-0258(19980130)17:2⟨219::AID-SIM735⟩3.0.CO;2-O.

Lindsey, J. K. 1998. A study of interval censoring in parametric regression models. *Lifetime Data Analysis* 4: 329–354. https://doi.org/10.1023/A:1009681919084.

Odell, P. M., K. M. Anderson, and R. B. D'Agostino. 1992. Maximum likelihood estimation for interval-censored data using a Weibull-based accelerated failure time model. *Biometrics* 48: 951–959. https://doi.org/10.2307/2532360.

Rabinowitz, D., A. A. Tsiatis, and J. Aragon. 1995. Regression with interval-censored data. *Biometrika* 82: 501–513. https://doi.org/10.2307/2337529.

Richman, D. D., J. M. Grimes, and S. W. Lagakos. 1990. Effect of stage of disease and drug dose on zidovudine susceptibilities of isolates of human immunodeficiency virus. *Journal of Acquired Immune Deficiency Syndromes* 3: 743–746.

Sun, J. 2006. *The Statistical Analysis of Interval-Censored Failure Time Data.* New York: Springer.

Sun, J., and J. Li. 2014. Interval censoring. In *Handbook of Survival Analysis*, ed. J. P. Klein, H. C. van Houwelingen, J. G. Ibrahim, and T. H. Scheike, 369–390. Boca Raton, FL: CRC Press.

Also see

[ST] **stintreg postestimation** — Postestimation tools for stintreg

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[ST] **stcox** — Cox proportional hazards model

[ST] **stintcox** — Cox proportional hazards model for interval-censored survival-time data

[ST] **streg** — Parametric survival models

[SVY] **svy estimation** — Estimation commands for survey data

[U] **20 Estimation and postestimation commands**

Title

> **stintreg postestimation** — Postestimation tools for stintreg

Postestimation commands   predict   margins   Remarks and examples
Methods and formulas   References   Also see

Postestimation commands

The following postestimation commands are of special interest after `stintreg`:

Command	Description
*`estat gofplot`	produce goodness-of-fit plot
`stcurve`	plot the survivor, failure, hazard, or cumulative hazard function

*`estat gofplot` is not appropriate with `svy` estimation results.

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of estimates
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
*`hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of coefficients
*`lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
`predict`	hazard ratios, survivor functions, influence statistics, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of estimates
`suest`	seemingly unrelated estimation
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

predict creates a new variable containing predictions such as median and mean survival times, hazards, hazard ratios, linear predictions, standard errors, probabilities, and Cox–Snell-like and martingale-like residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

> predict [*type*] *newvar* [*if*] [*in*] [, *statistic* *options*]
>
> predict [*type*] *newvar_l* *newvar_u* [*if*] [*in*] , *statistic2* [*options*]
>
> predict [*type*] *stub** [*if*] [*in*], <u>sc</u>ores

statistic	Description
Main	
<u>medi</u>an time	median survival time; the default
<u>medi</u>an <u>lnt</u>ime	median ln(survival time)
mean time	mean survival time
mean <u>lnt</u>ime	mean ln(survival time)
hr	hazard ratio, also known as the relative hazard
xb	linear prediction $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$
stdp	standard error of the linear prediction; $\text{SE}(\mathbf{x}_j\widehat{\boldsymbol{\beta}})$
* <u>mg</u>ale	martingale-like residuals

statistic2	Description
Main	
<u>ha</u>zard	hazard for interval endpoints t_l and t_u
surv	survivor probability for interval endpoints t_l and t_u
* <u>cs</u>nell	Cox–Snell-like residuals for interval endpoints t_l and t_u

options	Description
Main	
<u>noo</u>ffset	ignore the offset() variable specified in stintreg
oos	make *statistic* and *statistic2* available in and out of sample

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample. Starred statistics are calculated for the estimation sample by default, but the `oos` option makes them available both in and out of sample.

The predicted hazard ratio, option `hr`, is available only for the exponential, Weibull, and Gompertz models. The `mean time` and `mean lntime` options are not available for the Gompertz model.

`csnell` and `mgale` are not allowed with `svy` estimation results.

Options for predict

        ┌─────────┐
        │ Main │

`median time` calculates the predicted median survival time in analysis-time units. When no options are specified with `predict`, the predicted median survival time is calculated for all models.

`median lntime` calculates the natural logarithm of what `median time` produces.

`mean time` calculates the predicted mean survival time in analysis-time units. This option is not available for Gompertz regression.

`mean lntime` predicts the mean of the natural logarithm of `time`. This option is not available for Gompertz regression.

`hazard` calculates the predicted hazard for both the lower endpoint t_l and the upper endpoint t_u of the time interval.

`hr` calculates the hazard ratio. This option is valid only for models having a proportional-hazards parameterization.

`xb` calculates the linear prediction from the fitted model. That is, you fit the model by estimating a set of parameters, $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$, and the linear prediction is $\hat{y}_j = \hat{\beta}_0 + \hat{\beta}_1 x_{1j} + \hat{\beta}_2 x_{2j} + \cdots + \hat{\beta}_k x_{kj}$, often written in matrix notation as $\hat{y}_j = \mathbf{x}_j \hat{\boldsymbol{\beta}}$.

The $x_{1j}, x_{2j}, \ldots, x_{kj}$ used in the calculation are obtained from the data currently in memory and need not correspond to the data on the independent variables used in estimating $\boldsymbol{\beta}$.

`stdp` calculates the standard error of the linear prediction, that is, the standard error of $\hat{y}_j$.

`surv` calculates each observation's predicted survivor probabilities for both the lower endpoint t_l and the upper endpoint t_u of the time interval.

`csnell` calculates the Cox–Snell-like residuals for both the lower endpoint t_l and the upper endpoint t_u of the time interval.

`mgale` calculates interval-censored martingale-like residuals, which are an interval-censored version of martingale residuals for right-censored data.

`nooffset` is relevant only if you specified `offset(`*varname*`)` with `stintreg`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as $\mathbf{x}\boldsymbol{\beta}$ rather than $\mathbf{x}\boldsymbol{\beta} + \text{offset}$.

`oos` makes `csnell` and `mgale` available both in and out of sample. `oos` also dictates that summations and other accumulations take place over the sample as defined by `if` and `in`. By default, the summations are taken over the estimation sample, with `if` and `in` merely determining which values of *newvar*, *newvar*$_l$, and *newvar*$_u$ are to be filled in once the calculation is finished.

`scores` calculates equation-level score variables. The number of score variables created depends upon the chosen distribution.

The first new variable will always contain $\partial \ln L / \partial(\mathbf{x}_j \boldsymbol{\beta})$.

The subsequent new variables will contain the partial derivative of the log likelihood with respect to the ancillary parameters.

margins

Description for margins

margins estimates margins of response for median and mean survival times, hazard ratios, and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

margins [*marginlist*] [, *options*]

margins [*marginlist*] , <u>predict</u>(*statistic ...*) [<u>predict</u>(*statistic ...*) ...] [*options*]

statistic	Description
<u>median</u> time	median survival time; the default
<u>median</u> <u>lnt</u>ime	median ln(survival time)
mean time	mean survival time
mean <u>lnt</u>ime	mean ln(survival time)
hr	hazard ratio, also known as the relative hazard
xb	linear prediction $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$
stdp	not allowed with margins
<u>hazard</u>	not allowed with margins
<u>surv</u>	not allowed with margins
<u>csnell</u>	not allowed with margins
<u>mgale</u>	not allowed with margins

Hazard estimation is not allowed because it produces interval estimates.

Statistics not allowed with margins are functions of stochastic quantities other than e(b).

For the full syntax, see [R] **margins**.

Remarks and examples

Remarks are presented under the following headings:

> *Predicted values*
> *Residuals and diagnostic measures*

Predicted values

`predict` after `stintreg` is used to generate a new variable or variables containing predicted values or residuals.

Regardless of the metric used, `predict` can generate predicted median survival times and median log survival-times for all models and predicted mean times and mean log survival-times where available. Predicted survival, hazard, and residuals are also available for all models. The predicted hazard ratio can be calculated only for models with a proportional-hazards parameterization, that is, the Weibull, exponential, and Gompertz models. However, the estimation need not take place in the log-hazard metric. You can perform, for example, a Weibull regression specifying the `time` option and then ask that hazard ratios be predicted.

▷ Example 1: Obtaining predictions

Continuing with example 1 of [ST] **stintreg**, we refit a proportional-hazards Weibull model for the effect of treatment on breast retraction for breast cancer patients:

```
. use https://www.stata-press.com/data/r18/cosmesis
(Cosmetic deterioration of breast cancer patients)
. stintreg i.treat, interval(ltime rtime) distribution(weibull)
  (output omitted )
```

We can predict, for example, the median survival time and the log-median survival time for each observation by specifying the `median time` and `median lntime` options, respectively.

```
. predict time, median time
. predict lntime, median lntime
. tabulate treat, summarize(time) means freq
```

Treatment	Summary of Predicted median for (ltime,rtime] Mean	Freq.
Radio	39.332397	46
Radio+Che	22.300791	48
Total	30.635407	94

```
. tabulate treat, summarize(lntime) means freq
```

	Summary of Predicted median log for (ltime,rtime]	
Treatment	Mean	Freq.
Radio	3.6720486	46
Radio+Che	3.1046221	48
Total	3.3822989	94

From the `tabulate` command, the expected mean of the predicted median survival time for patients with radiotherapy only is approximately 39 months, and the expected mean of the predicted median survival time for patients with both radiotherapy and chemotherapy is 22 months. We can also obtain the same results by using `margins`.

```
. margins treat, predict(median time)
```

Adjusted predictions   Number of obs = 94
Model VCE: OIM

Expression: Predicted median for (ltime,rtime], predict(median time)

	Margin	Delta-method std. err.	z	P>|z|	[95% conf. interval]	
treat						
Radio	39.3324	5.342494	7.36	0.000	28.8613	49.80349
Radio+Chemo	22.30079	2.436642	9.15	0.000	17.52506	27.07652

```
. margins treat, predict(median lntime)
```

Adjusted predictions   Number of obs = 94
Model VCE: OIM

Expression: Predicted median log for (ltime,rtime], predict(median lntime)

	Margin	Delta-method std. err.	z	P>|z|	[95% conf. interval]	
treat						
Radio	3.672049	.1358293	27.03	0.000	3.405828	3.938269
Radio+Chemo	3.104622	.1092626	28.41	0.000	2.890471	3.318773

Because the `median` option is the default, we could have omitted it in the above specifications of `predict` and `margins`.

◁

▷ Example 2: Obtaining survivor probabilities

Continuing with the example above, we can compute observation-specific survivor probabilities. As with `predict` after [ST] **streg**, we will use `predict`'s `surv` option. For interval-censored data, however, estimates of survivor probabilities, as well as hazard estimates and Cox–Snell-like residuals, are intervals. So, to compute these statistics, we must specify two new variable names with `predict` instead of one; one variable will contain statistics computed using the lower time endpoint, and the other will contain statistics computed using the upper time endpoint.

```
. predict surv_l surv_u, surv
(38 missing values generated)

. list ltime rtime treat surv_l surv_u in 1/10
```

	ltime	rtime	treat	surv_l	surv_u
1.	0	7	Radio	1	.95814
2.	0	8	Radio	1	.948338
3.	0	5	Radio	1	.9754614
4.	4	11	Radio	.9828176	.9151379
5.	5	12	Radio	.9754614	.9029849
6.	5	11	Radio	.9754614	.9151379
7.	6	10	Radio	.967206	.9267811
8.	7	16	Radio	.95814	.8501493
9.	7	14	Radio	.95814	.8773297
10.	11	15	Radio	.9151379	.8639108

Listed above are the survivor probabilities, surv_l and surv_u, evaluated at the lower and upper time endpoints ltime and rtime, for the first 10 subjects, all of whom happen to be in the radiotherapy-only group.

◁

Residuals and diagnostic measures

For uncensored or right-censored data, several types of residuals have been introduced to assess the appropriateness of the fitted parametric survival models; see *Remarks and examples* in [ST] **streg postestimation** for details. Farrington (2000) proposed extensions of those residuals, including Cox–Snell-like residuals and martingale-like residuals, to interval-censored data; see the reference for applications and a discussion of limitations of the residuals for interval-censored data.

Cox–Snell-like residuals are used with interval-censored event-time data in assessing the overall model fit. If the model fits the data, those residuals should have the standard exponential distribution. To use them for checking the goodness of fit, we can estimate the cumulative hazard function corresponding to these residuals and plot them against the values at which the hazard is evaluated. If the model fits the data, the plot should be a straight line with a slope of 1 through the origin.

As with right-censored data, martingale-like residuals for interval-censored data do not arise naturally from martingale theory for parametric survival models as they do for the Cox proportional hazards model. For right-censored data, martingale residuals are defined using Cox–Snell residuals. For interval-censored data, Cox–Snell-like residuals are intervals themselves. So Farrington (2000) proposed a single measure, called adjusted Cox–Snell residuals, which are expectations of the interval residuals under the standard exponential distribution. Then, following Lagakos's (1981) definition of martingale residuals for right-censored data, an interval-censored version of martingale residuals is defined as one minus the adjusted Cox–Snell residuals. These martingale-like residuals are commonly used to examine the functional form of covariates. You could also use them to assess whether some covariates are needed in the model. Or you could plot them against observation numbers to identify outliers.

▷ Example 3: Check whether additional covariates should be included in the model

Martingale-like residuals may be used as a diagnostic tool to assess the need of including some other covariates in the model. If the model fits well without the covariate of interest, the plot of martingale residuals against that covariate should not show any trend.

Continuing with example 1, suppose that we want to check whether the patient's age (`age`) should be included in our model. We can specify the `mgale` option with `predict` to obtain the martingale-like residuals from the current model and store them in the `mg` variable. We then produce a scatterplot of `mg` against `age`.

```
. predict mg, mgale
. scatter mg age
```
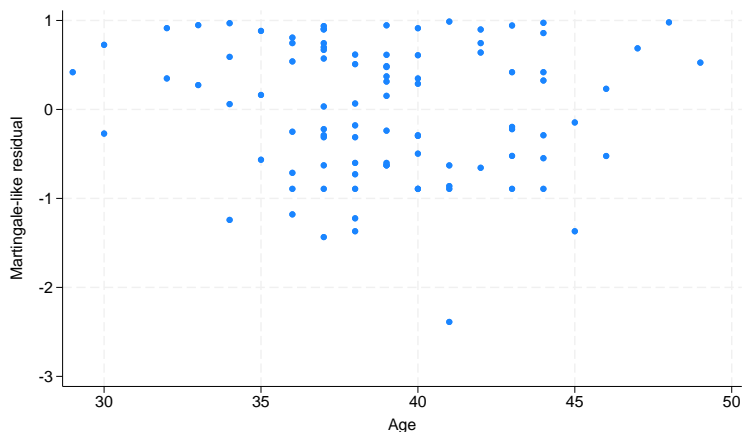

The figure does not show any systematic trend, suggesting that `age` is not needed in the model. In fact, if we included `age` in our Weibull model in the first place, we would have found that `age` is not statistically significant. You can verify this by typing

```
. stintreg i.treat age, interval(ltime rtime) distribution(weibull)
  (output omitted)
```

We can produce scatterplots of `mg` against other variables of interest to identify potential omitted predictors.

◁

▷ Example 4: Assess overall model fit

Returning to example 1, suppose that we instead want to fit the model with an exponential distribution and visually assess the overall model fit. We type

```
. quietly stintreg i.treat, interval(ltime rtime) distribution(exponential)
. estat gofplot
```

`estat gofplot` plots the estimated cumulative hazards for Cox–Snell-like residuals against the residuals themselves. The estimated cumulative hazards are calculated using the algorithm proposed by Turnbull (1976). The Cox–Snell-like residuals plotted against themselves form the $45°$ reference line. If the model fits the data well, the estimated cumulative hazards plotted against the Cox–Snell-like residuals should be close to the reference line. Comparing the jagged line with the reference line in

figure 1, we observe that the estimated cumulative hazards deviate from the reference line. So the exponential model does not appear to fit these data well.
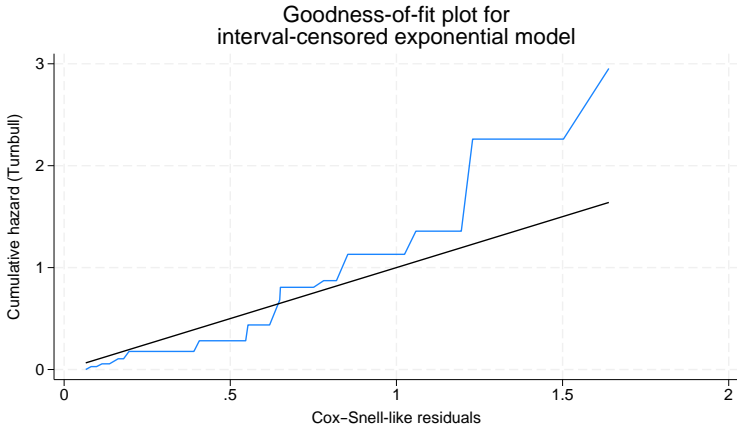
Figure 1. Goodness-of-fit plot for the exponential model

Let's refit this model using our original Weibull distribution and obtain the goodness-of-fit plot.

```
. quietly stintreg i.treat, interval(ltime rtime) distribution(weibull)
. estat gofplot
```
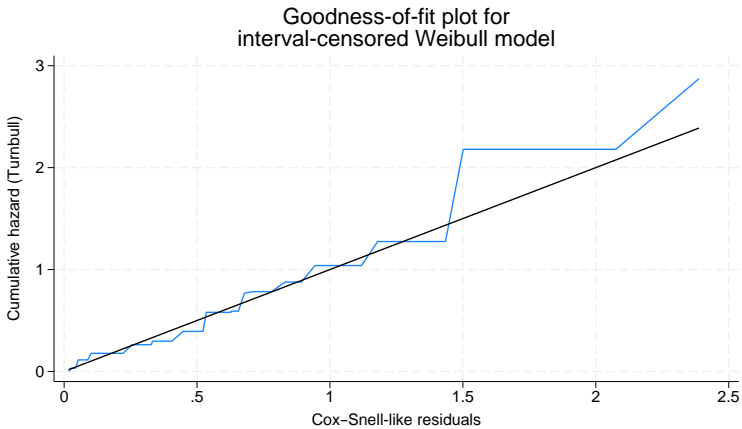

Figure 2. Goodness-of-fit plot for the Weibull model

The goodness-of-fit plot above shows that the jagged line stays very close to the $45°$ reference line. Therefore, we conclude that the Weibull model fits the data better than the exponential model.

◁

Methods and formulas

predict *newvar*, *statistic* may be used after stintreg to predict various quantities, according to the following *statistic*:

median time:

$$newvar_j = \{t : \widehat{S}_j(t) = 1/2\}$$

where $\widehat{S}_j(t)$ is $S_j(t)$ for observation j with the parameter estimates "plugged in" and $S_j(t)$ is defined in table 1 of [ST] **stintreg**.

median lntime:

$$newvar_j = \left\{y : \widehat{S}_j(e^y) = 1/2\right\}$$

mean time:

$$newvar_j = \int_0^\infty \widehat{S}_j(t)dt$$

mean lntime:

$$newvar_j = \int_{-\infty}^\infty ye^y \widehat{f}_j(e^y)dy$$

where $\widehat{f}_j(t)$ is $f_j(t)$ with the parameter estimates plugged in and $f_j(t) = -(d/dt)S_j(t)$.

hr (proportional hazards models only):

$$newvar_j = \exp(\mathbf{x}_j^* \widehat{\boldsymbol{\beta}^*})$$

where $\widehat{\boldsymbol{\beta}^*}$ does not contain the constant and $\mathbf{x}_j^*$ does not contain the coefficient of 1 corresponding to the constant.

xb:

$$newvar_j = \mathbf{x}_j \widehat{\boldsymbol{\beta}}$$

stdp:

$$newvar_j = \widehat{se}(\mathbf{x}_j \widehat{\boldsymbol{\beta}})$$

mgale:

$$newvar_j = \frac{\widehat{S}_j(t_{lj}) \log \widehat{S}_j(t_{lj}) - \widehat{S}_j(t_{uj}) \log \widehat{S}_j(t_{uj})}{\widehat{S}_j(t_{lj}) - \widehat{S}_j(t_{uj})}$$

For right-censored data, martingale residuals can be defined as the scores of the regression parameters. This property can carry over to the interval-censored data to define martingale-like residuals. Therefore, these residuals are expected to have mean zero and are uncorrelated asymptotically. Furthermore, these residuals are orthogonal to variables included in the model. Thus, we can use it to assess the need of including some other covariates in the model. See Farrington (2000) for details.

These residuals take values between $-\infty$ and 1 and have an expected value of 0, although like the Cox–Snell-like residuals, they are not symmetric about 0, making them difficult to interpret.

predict *newvar_l* *newvar_u* , *statistic2* may be used after stintreg to predict a pair of quantities for each observation for both the lower and upper endpoints of the time interval (t_{lj}, t_{uj}), according to the following *statistic2*:

hazard:

$$newvar_{lj} = \widehat{f}_j(t_{lj})/\widehat{S}_j(t_{lj})$$

$$newvar_{uj} = \widehat{f}_j(t_{uj})/\widehat{S}_j(t_{uj})$$

surv:

$$newvar_{lj} = \widehat{S}_j(t_{lj})$$

$$newvar_{uj} = \widehat{S}_j(t_{uj})$$

csnell:

$$newvar_{lj} = -\log \widehat{S}_j(t_{lj})$$

$$newvar_{uj} = -\log \widehat{S}_j(t_{uj})$$

The Cox–Snell-like residuals are the estimates of the cumulative hazard function obtained from the fitted model. They are computed separately for each of the two interval endpoints. Under the correct model assumption, the Cox–Snell-like residuals are expected to approximate an interval-censored sample from the standard exponential distribution. Therefore, they can be used for checking the overall model fit. Cox–Snell-like residuals can never be negative and therefore are not symmetric about zero. See Farrington (2000) for details.

References

Farrington, C. P. 2000. Residuals for proportional hazards models with interval-censored survival data. *Biometrics* 56: 473–482. https://doi.org/10.1111/j.0006-341X.2000.00473.x.

Lagakos, S. W. 1981. The graphical evaluation of explanatory variables in proportional hazard regression models. *Biometrika* 68: 93–98. https://doi.org/10.2307/2335809.

Turnbull, B. W. 1976. The empirical distribution function with arbitrarily grouped, censored and truncated data. *Journal of the Royal Statistical Society, Series B* 38: 290–295. https://doi.org/10.1111/j.2517-6161.1976.tb01597.x.

Also see

[ST] **stintreg** — Parametric models for interval-censored survival-time data

[ST] **estat gofplot** — Goodness-of-fit plots after streg, stcox, stintreg, or stintcox

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[U] **20 Estimation and postestimation commands**

Title

stir — Report incidence-rate comparison

Description

stir reports point estimates and confidence intervals for the incidence-rate ratio (IRR) and incidence-rate difference (IRD). Stratified IRRs may be standardized to produce standardized mortality ratios.

stir can be used with single- or multiple-record and single- or multiple-failure st data.

Quick start

IRR and IRD with confidence intervals for exposure indicator exposed using stset data

        stir exposed

Same as above, but display exact p-values calculated without the mid-p adjustment

        stir exposed, exact

Crude and Mantel–Haenszel combined IRRs with test of homogeneity for strata defined by svar

        stir exposed, strata(svar)

Same as above, and standardize the IRRs by weighting variable wvar

        stir exposed, strata(svar) standard(wvar)

Same as above, but standardize using time at risk for the unexposed group as weights

        stir exposed, strata(svar) estandard

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Report incidence-rate comparison

Syntax

> stir *exposedvar* $\begin{bmatrix} if \end{bmatrix}$ $\begin{bmatrix} in \end{bmatrix}$ $\begin{bmatrix} , options \end{bmatrix}$

options	Description
Options	
strata(*varname*)	stratify on *varname*
estandard	combine external weights with within-stratum statistics
istandard	combine internal weights with within-stratum statistics
standard(*varname*)	combine user-specified weights with within-stratum statistics
pool	display pooled estimate
nocrude	do not display crude estimate
nohom	do not display homogeneity test
ird	calculate standardized IRD
midp	display *p*-values calculated using mid-*p* adjustment (unstratified only); the default
exact	display exact *p*-values without mid-*p* adjustment (unstratified only)
level(#)	set confidence level; default is level(95)
noshow	do not show st setting information

You must stset your data before using stir; see [ST] **stset**.

by and collect are allowed; see [U] **11.1.10 Prefix commands**.

fweights and iweights may be specified using stset; see [ST] **stset**. stir may not be used with pweighted data.

Options

> Options

strata(*varname*) specifies that the calculation be stratified on *varname*, which may be a numeric or string variable. Within-stratum statistics are shown and then combined with Mantel–Haenszel weights. Also see the by() option in [R] **Epitab**.

estandard, istandard, standard(*varname*), pool, nocrude, nohom, and ird are relevant only if strata() is specified; see [R] **Epitab**.

midp and exact are relevant only if strata() is not specified; see [R] **Epitab**.

level(#) is relevant in all cases; see [R] **Epitab**.

noshow is relevant in all cases; it prevents stir from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

Remarks and examples

stir examines the incidence rate and time at risk.

```
. use https://www.stata-press.com/data/r18/page2
. stir group
        Failure _d: dead
  Analysis time _t: time
Incidence-rate comparison
Exposed:   group = 2
Unexposed: group = 1
```

	group Exposed	Unexposed	Total
Failures	19	17	36
Time	5023	4095	9118
Incidence rate	.0037826	.0041514	.0039482

	Point estimate	[95% conf. interval]		
Inc. rate diff.	−.0003688	−.002974	.0022364	
Inc. rate ratio	.9111616	.4484366	1.866047	(exact)
Prev. frac. ex.	.0888384	−.8660469	.5515634	(exact)
Prev. frac. pop	.04894			

```
Mid-p-values for tests of incidence-rate difference:
  Adj Pr(Exposed failures <= 19) = 0.3900 (lower one-sided)
  Adj Pr(Exposed failures >= 19) = 0.6100 (upper one-sided)
              Two-sided p-value = 0.7799
```

Specifying the exact option displays p-values for the tests of IRD calculated without using the mid-p adjustment. The noshow option suppresses the display of st variables.

```
. stir group, exact noshow
Incidence-rate comparison
Exposed:   group = 2
Unexposed: group = 1
```

	group Exposed	Unexposed	Total
Failures	19	17	36
Time	5023	4095	9118
Incidence rate	.0037826	.0041514	.0039482

	Point estimate	[95% conf. interval]		
Inc. rate diff.	−.0003688	−.002974	.0022364	
Inc. rate ratio	.9111616	.4484366	1.866047	(exact)
Prev. frac. ex.	.0888384	−.8660469	.5515634	(exact)
Prev. frac. pop	.04894			

```
Exact p-values for tests of incidence-rate difference:
  Pr(Exposed failures <= 19) = 0.4536 (lower one-sided)
  Pr(Exposed failures >= 19) = 0.6737 (upper one-sided)
          Two-sided p-value = 0.9071
```

See [R] **Epitab** for details about the exact option and other stir options.

Video example

[How to calculate incidence rates and incidence-rate ratios](#)

Stored results

stir (without strata()) stores the following in r():

Scalars

r(ird)	IRD
r(lb_ird)	lower CI bound for IRD
r(ub_ird)	upper CI bound for IRD
r(irr)	IRR
r(lb_irr)	lower CI bound for IRR
r(ub_irr)	upper CI bound for IRR
r(afe)	attributable fraction among the exposed
r(lb_afe)	lower CI bound for attributable fraction among the exposed
r(ub_afe)	upper CI bound for attributable fraction among the exposed
r(afp)	attributable fraction for the population
r(p_lower_midp)	lower one-sided p-value with mid-p adjustment
r(p_upper_midp)	upper one-sided p-value with mid-p adjustment
r(p_twosided_midp)	two-sided p-value with mid-p adjustment
r(p_lower_exact)	lower one-sided exact p-value
r(p_upper_exact)	upper one-sided exact p-value
r(p_twosided_exact)	two-sided exact p-value

stir, strata() stores the following in r():

Scalars

r(irr)	Mantel–Haenszel IRR, if option ird is not specified
r(lb_irr)	lower CI bound for Mantel–Haenszel IRR
r(ub_irr)	upper CI bound for Mantel–Haenszel IRR
r(ird)	Mantel–Haenszel IRD, if option ird is specified
r(lb_ird)	lower CI bound for Mantel–Haenszel IRD
r(ub_ird)	upper CI bound for Mantel–Haenszel IRD
r(crude)	crude IRR or, if option ird is specified, crude IRD
r(lb_crude)	lower CI bound for the crude IRR or IRD
r(ub_crude)	upper CI bound for the crude IRR or IRD
r(pooled)	pooled IRR or, if option ird is specified, pooled IRD
r(lb_pooled)	lower CI bound for pooled IRR or IRD
r(ub_pooled)	upper CI bound for pooled IRR or IRD
r(df)	degrees of freedom for homogeneity χ^2 test
r(chi2_mh)	Mantel–Haenszel homogeneity χ^2
r(chi2_p)	pooled homogeneity χ^2, if option pool is specified

Methods and formulas

stir simply accumulates numbers of failures and time at risk by exposed and unexposed (by strata, if necessary) and passes the calculation to ir; see [R] **Epitab**.

Reference

Dupont, W. D. 2009. *Statistical Modeling for Biomedical Researchers: A Simple Introduction to the Analysis of Complex Data*. 2nd ed. Cambridge: Cambridge University Press.

Also see

Title

> **stmc —** Calculate rate ratios with the Mantel–Cox method

Description

stmc calculates rate ratios that are stratified finely by time by using the Mantel–Cox method. The corresponding significance test (the log-rank test) is also calculated.

You can also use stmc to carry out a trend test for a metric explanatory variable.

Quick start

Failure-rate ratio, stratified by time, comparing category 1 with 0 in binary variable a; computed using the Mantel–Cox method

    stmc a

Same as above, but comparing 4 with 3 in multivalued b

    stmc b, compare(4,3)

Same as above, but controlling for values of catvar

    stmc b catvar, compare(4,3)

Test for a trend of failure rates with x1 controlling for time and catvar

    stmc x1 catvar

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Tabulate Mantel–Cox rate ratios

Syntax

> stmc *varname* [*varlist*] [*if*] [*in*] [, *options*]

options	Description
Main	
by(*varlist*)	tabulate rate ratio on *varlist*
<u>com</u>pare(*num1*,*den2*)	compare categories of exposure variable
<u>m</u>issing	include missing values as extra categories
<u>l</u>evel(*#*)	set confidence level; default is level(95)

You must stset your data before using stmc; see [ST] **stset**.

by and collect are allowed; see [U] **11.1.10 Prefix commands**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

Options

> **Main**

by(*varlist*) specifies categorical variables by which the rate ratio is to be tabulated.

> A separate rate ratio is produced for each category or combination of categories of *varlist*, and a test for unequal rate ratios (effect modification) is displayed.

compare(*num1*,*den2*) specifies the categories of the exposure variable to be compared. The first code defines the numerator categories, and the second code defines the denominator categories.

> When compare() is not specified and there are only two categories, the larger category is compared with the smaller one; when compare() is not specified and there are more than two categories, stmc analyzes the log-linear trend.

missing specifies that missing values of the explanatory variables be treated as extra categories. The default is to exclude such observations.

level(*#*) specifies the confidence level, as a percentage, for confidence intervals. The default is level(95) or as set by set level; see [U] **20.8 Specifying the width of confidence intervals**.

Remarks and examples

The stmc (Mantel–Cox) command is used to control for variation of rates on a time scale by breaking up time into short intervals, or *clicks*.

Usually, this approach is used only to calculate significance tests, but the rate ratio estimated remains just as useful as in the coarsely stratified analysis from [ST] **stmh**. The method may be viewed as an approximate form of Cox regression.

In its simplest specification, stmc *varname*, stmc will compute a failure-rate ratio, stratified by time, comparing the two categories of the explanatory variable *varname*. If *varname* has more than two categories, you can specify which two categories should be compared with the compare() option. If *varname* has more than two categories and you don't specify the compare() option, stmc will carry out a trend test. For trend tests, a one-step Newton approximation to the log-linear Poisson regression coefficient is also computed.

Any additional variables specified before the comma are categorical variables that are to be "controlled for" by using stratification. Strata are defined by cross-classification of these variables.

You can also specify categorical variables in the by() option. With this specification, the rate ratio is controlled for analysis time separately for each level of the variables specified with by() and then combined to give a rate ratio controlled for both time and the by() variables.

▷ Example 1

Below, we use the diet data (Clayton and Hills 1993) described in example 1 of [ST] **stsplit**. In this dataset, the presence of coronary heart disease (CHD) has been coded as fail = 1, 3, or 13. The variable hienergy is coded 1 if the total energy consumption is more than 2.75 Mcal and 0 otherwise. We want to obtain the effect of high energy controlled for age by stratifying finely. First, we stset the data, specifying the date of birth, dob, as the origin (so analysis time is age), and then we use stmc:

```
. use https://www.stata-press.com/data/r18/diet
(Diet data with dates)

. stset dox, origin(time dob) enter(time doe) id(id)
> scale(365.25) fail(fail==1 3 13)
Survival-time data settings

           ID variable: id
        Failure event: fail==1 3 13
Observed time interval: (dox[_n-1], dox]
    Enter on or after: time doe
    Exit on or before: failure
    Time for analysis: (time-origin)/365.25
               Origin: time dob

─────────────────────────────────────────────────────────────────
       337  total observations
         0  exclusions

─────────────────────────────────────────────────────────────────
       337  observations remaining, representing
       337  subjects
        46  failures in single-failure-per-subject data
 4,603.669  total analysis time at risk and under observation
                                      At risk from t =         0
                              Earliest observed entry t =  30.07529
                                 Last observed exit t =  69.99863


. stmc hienergy

         Failure _d: fail==1 3 13
   Analysis time _t: (dox-origin)/365.25
             Origin: time dob
  Enter on or after: time doe
         ID variable: id
Mantel-Cox estimate of the rate ratio
  comparing hienergy==1 vs. hienergy==0
  controlling for time (clicks)
```

Rate ratio	chi2	P>chi2	[95% conf. interval]	
0.537	4.20	0.0403	0.293	0.982

The rate ratio of 0.537 is close to that obtained with stmh when controlling for age by using 10-year age bands; see example 1 of [ST] **stmh**.

◁

Stored results

stmc stores the following in r():

Scalars
r(rratio)	overall rate ratio
r(chi2)	χ^2
r(p)	p-value
r(level)	confidence level
r(lb)	lower bound of confidence interval
r(ub)	upper bound of confidence interval
r(chi2_unequal)	χ^2 for test of unequal rate ratios with by()
r(p_unequal)	p-value for test of unequal rate ratios with by()

Macros
r(expvar)	explanatory variable
r(explevels)	levels of binary explanatory variable
r(controlvars)	control variables
r(byvars)	by() variables
r(test)	type of test

Matrices
r(table)	group-specific rate ratios

Acknowledgments

The original version of stmc was written by David Clayton (retired) of the Cambridge Institute for Medical Research and Michael Hills (1934–2021) of the London School of Hygiene and Tropical Medicine.

Reference

Clayton, D. G., and M. Hills. 1993. *Statistical Models in Epidemiology*. Oxford: Oxford University Press.

Also see

[ST] **stci** — Confidence intervals for means and percentiles of survival time

[ST] **stir** — Report incidence-rate comparison

[ST] **stmh** — Calculate rate ratios with the Mantel–Haenszel method

[ST] **stptime** — Calculate person-time, incidence rates, and SMR

[ST] **strate** — Tabulate failure rates and rate ratios

[ST] **stset** — Declare data to be survival-time data

Title

> **stmh —** Calculate rate ratios with the Mantel–Haenszel method

Description

stmh calculates stratified rate ratios and significance tests by using a Mantel–Haenszel-type method. You can also use stmh to carry out a trend test for a metric explanatory variable.

Quick start

Failure-rate ratio comparing category 1 with 0 in binary variable a, computed using the Mantel–Haenszel method

    stmh a

Same as above, but controlling for catvar

    stmh a catvar

Failure-rate ratios for each level of catvar, with test for unequal rate ratios

    stmh a, by(catvar)

Same as above, but compare 4 with 3 in multivalued b at each level of catvar

    stmh b, compare(4,3) by(catvar)

Test for a trend of failure rates with x1 controlling for b

    stmh x1 b

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Tabulate Mantel–Haenszel rate ratios

Syntax

> stmh *varname* [*varlist*] [*if*] [*in*] [, *options*]

options	Description
Main	
<u>b</u>y(*varlist*)	tabulate rate ratio on *varlist*
<u>c</u>ompare(*num1*,*den2*)	compare categories of exposure variable
<u>m</u>issing	include missing values as extra categories
<u>l</u>evel(*#*)	set confidence level; default is level(95)

You must stset your data before using stmh; see [ST] **stset**.
by and collect are allowed; see [U] **11.1.10 Prefix commands**.
fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

Options

> [Main]

by(*varlist*) specifies categorical variables by which the rate ratio is to be tabulated.

> A separate rate ratio is produced for each category or combination of categories of *varlist*, and a test for unequal rate ratios (effect modification) is displayed.

compare(*num1*,*den2*) specifies the categories of the exposure variable to be compared. The first code defines the numerator categories, and the second code defines the denominator categories.

> When compare() is not specified and there are only two categories, the larger category is compared with the smaller one; when compare() is not specified and there are more than two categories, stmh analyzes log-linear trend.

missing specifies that missing values of the explanatory variables be treated as extra categories. The default is to exclude such observations.

level(*#*) specifies the confidence level, as a percentage, for confidence intervals. The default is level(95) or as set by set level; see [U] **20.8 Specifying the width of confidence intervals**.

Remarks and examples

The stmh command is used for estimating rate ratios, controlled for confounding, using stratification. You can use it to estimate the ratio of the rates of failure for two categories of the explanatory variable *varname*. Categories to be compared may be defined by specifying the codes of the levels with compare(). Rate ratios and significance tests are calculated using a Mantel–Haenszel-type method.

The first variable listed on the command line after stmh is the explanatory variable used in comparing rates, and the remaining variables, if any, are categorical variables that are to be "controlled for" by using stratification. Strata are defined by cross-classification of these variables.

stmh may also be used to carry out trend tests for a metric explanatory variable. A one-step Newton approximation to the log-linear Poisson regression coefficient is also computed.

▷ Example 1: Stratified rate ratios

We want to examine the rate ratio of coronary heart disease (CHD) for individuals with high and low levels of caloric intake, while controlling for age. We will use data from a heart disease and diet survey that was analyzed in Clayton and Hills (1993). In this dataset, events such as cancer and CHD indicate failure, but we were interested only in CHD, which has been coded as fail = 1, 3, or 13. The variable hienergy is coded 1 if the total energy consumption is more than 2.75 Mcal and 0 otherwise.

First, we stset the data: origin is set to date of birth, making age the analysis time; and the scale is set to 365.25, so analysis time is measured in years. We also specify the failure codes for CHD.

```
. use https://www.stata-press.com/data/r18/diet, clear
(Diet data with dates)
. stset dox, origin(time dob) enter(time doe) id(id) scale(365.25)
> fail(fail==1 3 13)
  (output omitted)
```

Below, we stsplit the data into 10-year age bands. The new variable, ageband, identifies each observation's age group. We then use stmh to compare the rate for hienergy level 1 with the rate for level 0, controlled for ageband.

```
. stsplit ageband, at(40(10)70) after(time=dob) trim
(26 observations trimmed because of lower bound)
(418 observations (episodes) created)

. stmh hienergy ageband

        Failure _d: fail==1 3 13
  Analysis time _t: (dox-origin)/365.25
            Origin: time dob
  Enter on or after: time doe
        ID variable: id
              Note: ageband<=40 trimmed

Mantel-Haenszel estimate of the rate ratio
  comparing hienergy==1 vs. hienergy==0
  controlling for ageband
```

Rate ratio	chi2	P>chi2	[95% conf. interval]	
0.534	4.36	0.0369	0.293	0.972

We see that the rate of CHD for individuals with high levels of caloric intake is about half the rate for individuals with low levels of caloric intake, controlling for age.

We can also compare the effect of `hienergy` between jobs, controlling for `ageband`.

```
. stmh hienergy ageband, by(job)
          Failure _d: fail==1 3 13
    Analysis time _t: (dox-origin)/365.25
              Origin: time dob
   Enter on or after: time doe
         ID variable: id
                Note: ageband<=40 trimmed
```

Mantel–Henszel estimates of the rate ratio
comparing hienergy==1 vs. hienergy==0
controlling for **ageband**
by **job**

job	Rate ratio	Lower	Upper
0	0.42	0.13	1.33
1	0.64	0.22	1.87
2	0.51	0.21	1.26

Note: Lower and Upper are bounds of 95% confidence intervals.

Overall Mantel–Henszel estimate, controlling for **ageband** and **job**

Rate ratio	chi2	P>chi2	[95% conf. interval]	
0.521	4.88	0.0271	0.289	0.939

```
Approx. test for unequal RRs (effect modification): chi2(2)  =    0.28
                                                    Pr>chi2  =  0.8695
```

Because the rate-ratio estimates are approximate, the test for unequal rate ratios is also approximate.

◁

▷ Example 2: Log-linear trend test for metric explanatory variables

`diet.dta` contains the height for each patient recorded in the variable `height`. We can test for a trend of heart disease rates with `height` controlling for `ageband` by typing

```
. stmh height ageband
          Failure _d: fail==1 3 13
    Analysis time _t: (dox-origin)/365.25
              Origin: time dob
   Enter on or after: time doe
         ID variable: id
                Note: ageband<=40 trimmed
```

Score test for trend of rates with height

Overall Mantel–Henszel estimate, controlling for **ageband**

Rate ratio	chi2	P>chi2	[95% conf. interval]	
0.906	18.60	0.0000	0.866	0.948

Note: The Rate ratio estimate is an approximation to the rate ratio
      for a one-unit increase in height.

`stmh` tested for trend of heart disease rates with height within age bands and provided a rough estimate of the rate ratio for a 1 centimeter increase in height—this estimate is a one-step Newton approximation to the maximum likelihood estimate. It is not consistent, but it does provide a useful indication of the size of the effect.

The rate ratio is less than 1, so there is evidence for a decreasing rate with increasing height (about a 9% decrease in rate per centimeter increase in height).

◁

Stored results

stmh stores the following in r():

Scalars
r(rratio)	overall rate ratio
r(chi2)	χ^2
r(p)	p-value
r(level)	confidence level
r(lb)	lower bound of confidence interval
r(ub)	upper bound of confidence interval
r(chi2_unequal)	χ^2 for test of unequal rate ratios with by()
r(p_unequal)	p-value for test of unequal rate ratios with by()

Macros
r(expvar)	explanatory variable
r(explevels)	levels of binary explanatory variable
r(controlvars)	control variables
r(byvars)	by() variables
r(test)	type of test

Matrices
r(table)	group-specific rate ratios

Acknowledgments

Nathan Mantel (1919–2002) was an American biostatistician who grew up in New York. He served in the U.S. Army from 1942 to 1946 and then joined the National Cancer Institute as a statistical consultant in 1947. He would remain there for over 26 years. During this time, he was presented with problems from many fields, which led to his diverse publications and contributions. He is remembered for his outstanding problem-solving skills and brilliance.

Mantel is best known for his 1959 collaboration with William M. Haenszel, which introduced the Mantel–Haenszel method for obtaining odds ratios for stratified data and a test of equality across strata. In 1966, Mantel developed the log-rank test and, through a collaboration in 1974, extended this to data involving transient states. In another collaboration, he helped develop a procedure for testing carcinogenic agents; this paper defined a safe dosage level that was temporarily adopted by the Food and Drug Administration. Mantel also contributed to the development of the polychotomous logistic regression model and methods used to explore temporal and spatial clustering of diseases.

Mantel received many honors recognizing his contributions. He was elected an honorary fellow of the Royal Statistical Society and received the Superior Service Award from the National Institutes of Health. He was also a fellow of the American Statistical Association, which in 2002 created a lifetime achievement award in his name.

William M. Haenszel (1910–1998) was an American biostatistician and epidemiologist. In 1932, he obtained a master's degree in statistics from the University of Buffalo. He then worked at several leading institutions of health, making a lasting impact worldwide.

From 1934 to 1947, he worked for the New York State Department of Health, after which he became the director of the Bureau of Vital Statistics in the Connecticut State Health Department. In 1952, he joined the National Institutes of Health, where he would serve as the chief of the Biometry Branch and head of the Biometric Section. During his time there, Haenszel collaborated with Nathan Mantel to develop the Mantel–Haenszel method. He also initiated a series of studies of migrant populations that explored the causes of diseases. Another major contribution was the instrumental role he played in developing SEER, the first national system for tracking cases of cancer and the potential causes.

Haenszel was a fellow of the American Public Health Association, the American Association for the Advancement of Science, and the American Statistical Association. He also served as chair of the Biometrics Section of the American Statistical Association. Toward the end of his career, Haenszel joined the Illinois Cancer Council and provided consulting services to the World Health Organization. Despite having Parkinson's disease, Haenszel helped tutor a research group that was developing a brain tumor registry.

References

Clayton, D. G., and M. Hills. 1993. *Statistical Models in Epidemiology*. Oxford: Oxford University Press.

Gail, M. H. 1997. A conversation with Nathan Mantel. *Statistical Science* 12: 88–97. https://doi.org/10.1214/ss/1029963424.

Hankey, B. 1997. A conversation with William M. Haenszel. *Statistical Science* 12: 108–112. https://doi.org/10.1214/ss/1029963427.

Also see

Title

> **stptime —** Calculate person-time, incidence rates, and SMR

Description

stptime calculates person-time and incidence rates. stptime computes standardized mortality/morbidity ratios (SMRs) after merging the data with a suitable file of standard rates specified with the using() option.

Quick start

Person-time and incidence rate using [stset](#) data

    stptime

Same as above, but tabulate in ten-year intervals from 20 to 50

    stptime, at(20(10)50)

Same as above, but exclude observations less than or equal to 20 or greater than 50

    stptime, at(20(10)50) trim

Same as above, but report rate per 1,000 person-years with two decimal places

    stptime, at(20(10)50) trim per(1000) dd(2)

Person-time and incidence rates for each level of v1

    stptime, by(v1)

Standardized mortality ratios in 10-year intervals from 20 to 50 from reference rates rvar for lower
    end-points lower, defining each cohort saved in mydata.dta

    stptime, at(20(10)50) smr(lower rvar) using(mydata)

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Person-time, incidence rates, and SMR

Syntax

> stptime $\bigl[$ *if* $\bigr]$ $\bigl[$, *options* $\bigr]$

options	Description
Main	
at(*numlist*)	compute person-time at specified intervals; default is to compute overall person-time and incidence rates
trim	exclude observations $\leq$ minimum or $>$ maximum of at()
by(*varname*)	compute incidence rates or SMRs by *varname*
Options	
per(*#*)	units to be used in reported rates
dd(*#*)	number of decimal digits to be displayed
smr(*groupvar ratevar*)	use *groupvar* and *ratevar* in using() dataset to calculate SMRs
using(*filename*)	specify filename to merge that contains smr() variables
level(*#*)	set confidence level; default is level(95)
noshow	do not show st setting information
Advanced	
jackknife	jackknife confidence intervals
title(*string*)	label output table with *string*
output(*filename* $\bigl[$, replace $\bigr]$)	save summary dataset as *filename*; use replace to overwrite existing *filename*

You must stset your data before using stptime; see [ST] **stset**.

by and collect are allowed; see [U] **11.1.10 Prefix commands**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

Options

> [**Main**]

at(*numlist*) specifies intervals at which person-time is to be computed. The intervals are specified in analysis time t units. If at() is not specified, overall person-time and incidence rates are computed.

If, for example, you specify at(5(5)20) and the trim option is not specified, person-time is reported for the intervals $t = (0 - 5]$, $t = (5 - 10]$, $t = (10 - 15]$, and $t = (15 - 20]$.

trim specifies that observations less than or equal to the minimum or greater than the maximum value listed in at() be excluded from the computations.

by(*varname*) specifies a categorical variable by which incidence rates or SMRs are to be computed.

> [**Options**]

per(*#*) specifies the units to be used in reported rates. For example, if the analysis time is in years, specifying per(1000) results in rates per 1,000 person-years.

dd(*#*) specifies the maximum number of decimal digits to be reported for rates, ratios, and confidence intervals. This option affects only how values are displayed, not how they are calculated.

smr(*groupvar ratevar*) specifies two variables in the using() dataset. The *groupvar* identifies the age-group or calendar-period variable used to match the data in memory and the using() dataset. The *ratevar* variable contains the appropriate reference rates. stptime then calculates SMRs rather than incidence rates.

using(*filename*) specifies the filename that contains a file of standard rates that is to be merged with the data so that SMRs can be calculated.

level(*#*) specifies the confidence level, as a percentage, for confidence intervals. The default is level(95) or as set by set level; see [U] **20.8 Specifying the width of confidence intervals**.

noshow prevents stptime from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

> ⎡ Advanced ⎤

jackknife specifies that jackknife confidence intervals be produced. This is the default if pweights or iweights were specified when the dataset was stset.

title(*string*) replaces the default "Person-time" label on the output table with *string*.

output(*filename* [, replace]) saves a summary dataset in *filename*. The file contains counts of failures and person-time, incidence rates (or SMRs), confidence limits, and categorical variables identifying the time intervals. This dataset could be used for further calculations or simply as input to the table command.

> replace specifies that *filename* be overwritten if it exists. This option is not shown in the dialog box.

Remarks and examples

stptime computes and tabulates the person-time and incidence rate (formed from the number of failures divided by the person-time). If you use the by() option, this will be calculated by different levels of one or more categorical explanatory variables specified by *varname*. Confidence intervals for the rate are also given. By default, the confidence intervals are calculated using the quadratic approximation to the Poisson log likelihood for the log-rate parameter. However, whenever the Poisson assumption is questionable, such as when pweights or iweights are used, jackknife confidence intervals can also be calculated.

stptime can also calculate and report SMRs if the data have been merged with a suitable file of reference rates.

If pweights or iweights were specified when the dataset was stset, stptime calculates jackknife confidence intervals by default.

The summary dataset can be saved to a file specified with the output() option for further analysis or a more elaborate graphical display.

▷ Example 1

We begin with a simple fictitious example from Clayton and Hills (1993, 42). Thirty subjects were monitored until the development of a particular disease. Here are the data for the first five subjects:

```
. use https://www.stata-press.com/data/r18/stptime
. list in 1/5
```

	id	year	fail
1.	1	19.6	1
2.	2	10.8	1
3.	3	14.1	1
4.	4	3.5	1
5.	5	4.8	1

The id variable identifies the subject, year records the time to failure in years, and fail is the failure indicator, which is 1 for all 30 subjects in the data. To use stptime, we must first stset the data.

```
. stset year, fail(fail) id(id)
Survival-time data settings

           ID variable: id
        Failure event: fail!=0 & fail<.
Observed time interval: (year[_n-1], year]
     Exit on or before: failure
─────────────────────────────────────────────────────────────────────
         30  total observations
          0  exclusions
─────────────────────────────────────────────────────────────────────
         30  observations remaining, representing
         30  subjects
         30  failures in single-failure-per-subject data
      261.9  total analysis time at risk and under observation
                                            At risk from t =         0
                                 Earliest observed entry t =         0
                                     Last observed exit t =      36.5
```

We can use stptime to obtain the overall person-time of observation and disease incidence rate.

```
. stptime, title(Person-years)
        Failure _d: fail
  Analysis time _t: year
       ID variable: id
Estimated person-time and incidence rate
```

Cohort	Person-years	Failures	Rate	[95% conf. interval]	
Total	261.9	30	.11454754	.08009	.1638299

The total 261.9 person-years reported by stptime matches what stset reported as total analysis time at risk. stptime computed an incidence rate of 0.11454754 per person-year. In epidemiology, incidence rates are often presented per 1,000 person-years. We can do this by specifying per(1000).

```
. stptime, title(Person-years) per(1000)

        Failure _d: fail
   Analysis time _t: year
       ID variable: id

Estimated person-time and incidence rate
     Cohort | Person-years   Failures         Rate    [95% conf. interval]
    --------+----------------------------------------------------------------
      Total |        261.9         30    114.54754     80.09001     163.8299
```

More interesting would be to compare incidence rates at 10-year intervals. We will specify dd(4) to display rates to four decimal places.

```
. stptime, per(1000) at(0(10)40) dd(4)

        Failure _d: fail
   Analysis time _t: year
       ID variable: id

Estimated person-time and incidence rates
     Cohort | Person-time   Failures         Rate    [95% conf. interval]
    --------+----------------------------------------------------------------
(0   -  10] |     188.8000         18      95.3390      60.0676     151.3215
(10  -  20] |      55.1000         10     181.4882      97.6506     337.3044
(20  -  30] |      11.5000          1      86.9565      12.2490     617.3106
     >  30  |       6.5000          1     153.8462      21.6713    1092.1648
    --------+----------------------------------------------------------------
      Total |     261.9000         30     114.5475      80.0900     163.8299
```

◁

▷ Example 2

Using the diet data (Clayton and Hills 1993) described in example 1 of [ST] **stsplit**, we will use stptime to tabulate age-specific person-years and coronary heart disease (CHD) incidence rates. In this dataset, CHD has been coded as fail = 1, 3, or 13.

We first `stset` the data: failure codes for CHD are specified; origin is set to date of birth, making age the analysis time; and the scale is set to 365.25, so analysis time is measured in years.

```
. use https://www.stata-press.com/data/r18/diet
(Diet data with dates)
. stset dox, origin(time dob) enter(time doe) id(id) scale(365.25)
> fail(fail==1 3 13)

Survival-time data settings

           ID variable: id
         Failure event: fail==1 3 13
Observed time interval: (dox[_n-1], dox]
     Enter on or after: time doe
     Exit on or before: failure
     Time for analysis: (time-origin)/365.25
                Origin: time dob

──────────────────────────────────────────────────────────────────
        337  total observations
          0  exclusions
──────────────────────────────────────────────────────────────────
        337  observations remaining, representing
        337  subjects
         46  failures in single-failure-per-subject data
  4,603.669  total analysis time at risk and under observation
                                       At risk from t =          0
                              Earliest observed entry t =   30.07529
                                 Last observed exit t =   69.99863
```

The incidence of CHD per 1,000 person-years can be tabulated in 10-year intervals.

```
. stptime, per(1000) at(40(10)70) trim

         Failure _d: fail==1 3 13
   Analysis time _t: (dox-origin)/365.25
             Origin: time dob
  Enter on or after: time doe
         ID variable: id
               Note: _group<=40 trimmed

Estimated person-time and incidence rates
```

Cohort	Person-time	Failures	Rate	[95% conf. interval]	
(40 - 50]	907.00616	6	6.6151701	2.971936	14.72457
(50 - 60]	2107.0418	18	8.5427828	5.382317	13.55906
(60 - 70]	1493.2923	22	14.732548	9.700656	22.37457
Total	4507.3402	46	10.205575	7.644246	13.62512

◁

The SMR for a cohort is the ratio of the total number of observed deaths to the number expected from age-specific reference rates. This expected number can be found by multiplying the person-time in each cohort by the reference rate for that cohort. Using the `smr` option to define the cohort variable and reference rate variable in the `using()` dataset, `stptime` calculates SMRs and confidence intervals. You must specify the `per()` option. For example, if the reference rates were per 100,000, you would specify `per(100000)`.

▷ Example 3

In `smrchd.dta`, we have age-specific CHD rates per 1,000 person-years for a reference population. We can merge these data with our current data and use `stptime` to obtain SMRs and confidence intervals.

```
. stptime, smr(ageband rate) using(https://www.stata-press.com/data/r18/smrchd)
> per(1000) at(40(10)70) trim
          Failure _d: fail==1 3 13
    Analysis time _t: (dox-origin)/365.25
              Origin: time dob
   Enter on or after: time doe
         ID variable: id
                Note: _group<=40 trimmed
Estimated person-time and standardized mortality ratios
```

	Person-time	Observed failures	Expected failures	SMR	[95% conf. interval]	
Cohort						
(40 - 50]	907.00616	6	5.62344	1.067	.4793445	2.374931
(50 - 60]	2107.0418	18	18.7527	.95986	.6047547	1.52349
(60 - 70]	1493.2923	22	22.8474	.96291	.6340298	1.46239
Total	4507.3402	46	47.2235	.97409	.7296205	1.300477

The `stptime` command can also calculate person-time and incidence rates or SMRs by categories of the explanatory variable. In our diet data, the variable `hienergy` is coded 1 if the total energy consumption is more than 2.75 Mcal and 0 otherwise. We want to compute the person-years and incidence rates for these two levels of `hienergy`.

```
. stptime, by(hienergy) per(1000)
          Failure _d: fail==1 3 13
    Analysis time _t: (dox-origin)/365.25
              Origin: time dob
   Enter on or after: time doe
         ID variable: id
Estimated person-time and incidence rates
```

hienergy	Person-time	Failures	Rate	[95% conf. interval]	
0	2059.4305	28	13.595992	9.387478	19.69123
1	2544.2382	18	7.0748093	4.457431	11.2291
Total	4603.6687	46	9.9920309	7.484296	13.34002

We can also compute the incidence rate for the two levels of `hienergy` and the three previously defined age cohorts:

```
. stptime, by(hienergy) per(1000) at(40(10)70) trim
           Failure _d: fail==1 3 13
     Analysis time _t: (dox-origin)/365.25
                Origin: time dob
    Enter on or after: time doe
           ID variable: id
```

Estimated person-time and incidence rates

hienergy	Person-time	Failures	Rate	[95% conf. interval]	
0					
(40 – 50]	346.87474	2	5.76577	1.442006	23.05407
(50 – 60]	979.34018	12	12.253148	6.958681	21.57587
> 60	699.13758	14	20.024671	11.85966	33.81104
1					
(40 – 50]	560.13142	4	7.1411813	2.680213	19.02702
(50 – 60]	1127.7016	6	5.3205566	2.390317	11.84292
> 60	794.15469	8	10.073604	5.037786	20.14327
Total	4507.3402	46	10.205575	7.644246	13.62512

Or we can compute the corresponding SMR:

```
. stptime, smr(ageband rate) using(https://www.stata-press.com/data/r18/smrchd)
> by(hienergy) per(1000) at(40(10)70) trim
           Failure _d: fail==1 3 13
     Analysis time _t: (dox-origin)/365.25
                Origin: time dob
    Enter on or after: time doe
           ID variable: id
```

Estimated person-time and standardized mortality ratios

hienergy	Person-time	Observed failures	Expected failures	SMR	[95% conf. interval]	
0						
(40 – 50]	346.87474	2	2.15062	.9299629	.2325815	3.718399
(50 – 60]	979.34018	12	8.71613	1.376758	.7818743	2.424256
> 60	699.13758	14	10.6968	1.308802	.7751411	2.209872
1						
(40 – 50]	560.13142	4	3.47281	1.151803	.4322924	3.068875
(50 – 60]	1127.7016	6	10.0365	.5978154	.2685749	1.330665
> 60	794.15469	8	12.1506	.6584055	.329267	1.316554
Total	4507.3402	46	47.2235	.9740917	.7296205	1.300477

◁

Video example

How to calculate incidence rates and incidence-rate ratios

Stored results

stptime stores the following in r():

Scalars
    r(ptime)   person-time
    r(failures)   observed failures
    r(rate)   failure rate
    r(expected)   expected number of failures
    r(smr)   standardized mortality ratio
    r(lb)   lower bound for SMR
    r(ub)   upper bound for SMR

References

Clayton, D. G., and M. Hills. 1993. *Statistical Models in Epidemiology*. Oxford: Oxford University Press.

Rutherford, M. J., P. C. Lambert, and J. Thompson. 2010. Age–period–cohort modeling. *Stata Journal* 10: 606–627.

Also see

[ST] **stci** — Confidence intervals for means and percentiles of survival time

[ST] **stir** — Report incidence-rate comparison

[ST] **strate** — Tabulate failure rates and rate ratios

[ST] **stset** — Declare data to be survival-time data

[ST] **stsplit** — Split and join time-span records

[R] **Epitab** — Tables for epidemiologists

Title

> **strate —** Tabulate failure rates and rate ratios

Description

strate tabulates rates by one or more categorical variables declared in *varlist*. You can also save an optional summary dataset, which includes event counts and rate denominators, for further analysis or display. The combination of the commands stsplit and strate implements most of, if not all, the functions of the special-purpose person-years programs in widespread use in epidemiology; see [ST] **stsplit**.

Quick start

Table of failure rates using stset data

    strate

Same as above, but calculate failure rates at each level of categorical variable catvar

    strate catvar

Graph rates against catvar

    strate catvar, graph

Table of SMRs per 1,000 with reference rates stored in variable rvar

    strate catvar, per(1000) smr(rvar)

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Tabulate failure rates and rate ratios

Syntax

> strate [*varlist*] [*if*] [*in*] [, *options*]

options	Description
Main	
per(*#*)	units to be used in reported rates
smr(*varname*)	use *varname* as reference-rate variable to calculate SMRs
cluster(*varname*)	cluster variable to be used by the jackknife
jackknife	report jackknife confidence intervals
missing	include missing values as extra categories
graph	graph rates against exposure category
nowhisker	omit confidence intervals from the graph
level(*#*)	set confidence level; default is level(95)
output(*filename*[, replace])	save summary dataset as *filename*; use replace to overwrite existing *filename*
nolist	suppress listed output
Plot	
marker_options	change look of markers (color, size, etc.)
marker_label_options	add marker labels; change look or position
cline_options	affect rendition of the plotted points
CI plot	
ciopts(*rspike_options*)	affect rendition of the confidence intervals (whiskers)
Add plots	
addplot(*plot*)	add other plots to the generated graph
Y axis, X axis, Titles, Legend, Overall	
twoway_options	any options other than by() documented in [G-3] *twoway_options*

You must stset your data before using strate; see [ST] **stset**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

Options

> **Main**

per(*#*) specifies the units to be used in reported rates. For example, if the analysis time is in years, specifying per(1000) results in rates per 1,000 person-years.

smr(*varname*) specifies a reference-rate variable. strate then calculates SMRs rather than rates. This option will usually follow stsplit to separate the follow-up records by age bands and possibly calendar periods.

cluster(*varname*) defines a categorical variable that indicates clusters of data to be used by the jackknife. If the jackknife option is selected and this option is not specified, the cluster variable is taken as the id variable defined in the st data. Specifying cluster() implies jackknife.

jackknife specifies that jackknife confidence intervals be produced. This is the default if weights were specified when the dataset was stset.

missing specifies that missing values of the explanatory variables be treated as extra categories. The default is to exclude such observations.

graph produces a graph of the rate against the numerical code used for the categories of *varname*.

nowhisker omits the confidence intervals from the graph.

level(*#*) specifies the confidence level, as a percentage, for confidence intervals. The default is level(95) or as set by set level; see [U] **20.8 Specifying the width of confidence intervals**.

output(*filename* [, replace]) saves a summary dataset in *filename*. The file contains counts of failures and person-time, rates (or SMRs), confidence limits, and all the categorical variables in the *varlist*. This dataset could be used for further calculations or simply as input to the table command; see [R] **table**.

> replace specifies that *filename* be overwritten if it exists. This option is not shown in the dialog box.

nolist suppresses the output. This is used only when saving results to a file specified by output().

Plot
---

marker_options affect the rendition of markers drawn at the plotted points, including their shape, size, color, and outline; see [G-3] ***marker_options***.

marker_label_options specify if and how the markers are to be labeled; see [G-3] ***marker_label_options***.

cline_options affect whether lines connect the plotted points and the rendition of those lines; see [G-3] ***cline_options***.

CI plot
---

ciopts(*rspike_options*) affects the rendition of the confidence intervals (whiskers); see [G-3] ***rspike_options***.

Add plots
---

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] ***addplot_option***.

Y axis, X axis, Titles, Legend, Overall
---

twoway_options are any of the options documented in [G-3] ***twoway_options***, excluding by(). These include options for titling the graph (see [G-3] ***title_options***) and for saving the graph to disk (see [G-3] ***saving_option***).

Remarks and examples

strate tabulates the rate, formed from the number of failures divided by the person-time, by different levels of one or more categorical explanatory variables specified by *varlist*. Confidence intervals for the rate are also given. By default, the confidence intervals are calculated using the quadratic approximation to the Poisson log likelihood for the log-rate parameter. However, whenever the Poisson assumption is questionable, jackknife confidence intervals can also be calculated. The jackknife option also allows for multiple records for the same cluster (usually subject).

strate can also calculate and report SMRs if the data have been merged with a suitable file of reference rates.

The summary dataset can be saved to a file specified with the output() option for further analysis or more elaborate graphical display.

   If weights were specified when the dataset was `stset`, `strate` calculates jackknife confidence intervals by default.

▷ Example 1: Tabulation of failure rates

   Using the diet data (Clayton and Hills 1993) described in example 1 of [ST] **stsplit**, we will use `strate` to tabulate age-specific coronary heart disease (CHD). In this dataset, CHD has been coded as `fail` = 1, 3, or 13.

   We first `stset` the data: failure codes for CHD are specified; origin is set to date of birth, making age the analysis time; and the scale is set to 365.25, so analysis time is measured in years.

```
. use https://www.stata-press.com/data/r18/diet
(Diet data with dates)
. stset dox, origin(time dob) enter(time doe) id(id) scale(365.25)
> fail(fail==1 3 13)
Survival-time data settings

           ID variable: id
        Failure event: fail==1 3 13
Observed time interval: (dox[_n-1], dox]
     Enter on or after: time doe
     Exit on or before: failure
     Time for analysis: (time-origin)/365.25
                Origin: time dob

───────────────────────────────────────────────────────────────────
        337  total observations
          0  exclusions
───────────────────────────────────────────────────────────────────
        337  observations remaining, representing
        337  subjects
         46  failures in single-failure-per-subject data
  4,603.669  total analysis time at risk and under observation
                                          At risk from t =         0
                                Earliest observed entry t =  30.07529
                                  Last observed exit t =  69.99863
```

Now we `stsplit` the data into 10-year age bands.

```
. stsplit ageband, at(40(10)70) after(time=dob) trim
(26 observations trimmed because of lower bound)
(418 observations (episodes) created)
```

`stsplit` added 418 observations to the dataset in memory and generated a new variable, `ageband`, that identifies each observation's age group.

The CHD rate per 1,000 person-years can now be tabulated for categories of `ageband`:

```
. strate ageband, per(1000) graph
            Failure _d: fail==1 3 13
      Analysis time _t: (dox-origin)/365.25
                Origin: time dob
      Enter on or after: time doe
            ID variable: id
                  Note: ageband<=40 trimmed
  Estimated failure rates
  Number of records = 729
```

ageband	D	Y	Rate	Lower	Upper
40	6	0.9070	6.6152	2.9719	14.7246
50	18	2.1070	8.5428	5.3823	13.5591
60	22	1.4933	14.7325	9.7007	22.3746

```
  Notes: Rate = D/Y = failures/person-time (per 1000).
         Lower and Upper are bounds of 95% confidence intervals.
```
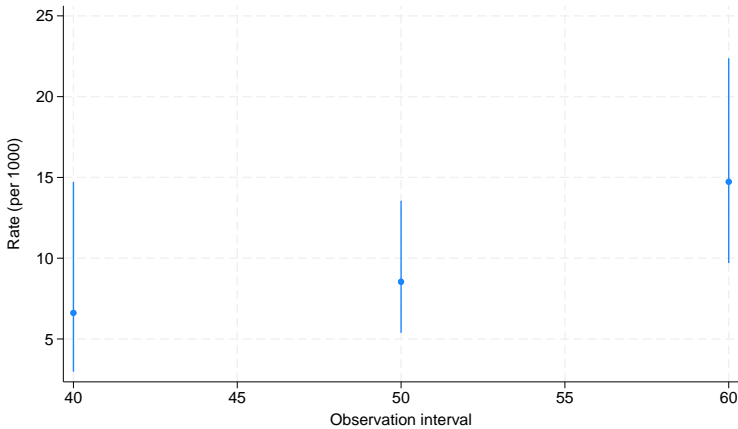

Because we specified the `graph` option, `strate` also generated a plot of the estimated rates and confidence intervals.

◁

The SMR for a cohort is the ratio of the total number of observed deaths to the number expected from age-specific reference rates. This expected number can be found by first expanding on age, using `stsplit`, and then multiplying the person-years in each age band by the reference rate for that band. `merge` (see [D] **merge**) can be used to add the reference rates to the dataset. Using the `smr` option to define the variable containing the reference rates, `strate` calculates SMRs and confidence intervals. You must specify the `per()` option. For example, if the reference rates were per 100,000 person-years, you would specify `per(100000)`. When reference rates are available by age and calendar period, you must call `stsplit` twice to expand on both time scales before merging the data with the reference-rate file.

▷ Example 2: Tabulation of SMRs

In `smrchd.dta`, we have age-specific CHD rates per 1,000 person-years for a reference population. We can merge these data with our current data and use `strate` to obtain SMRs and confidence intervals.

```
. sort ageband
. merge m:1 ageband using https://www.stata-press.com/data/r18/smrchd
(variable ageband was byte, now float to accommodate using data's values)

    Result                       Number of obs
    ─────────────────────────────────────────────
    Not matched                             26
        from master                         26  (_merge==1)
        from using                           0  (_merge==2)
    Matched                                729  (_merge==3)
    ─────────────────────────────────────────────

. strate ageband, per(1000) smr(rate)
          Failure _d: fail==1 3 13
    Analysis time _t: (dox-origin)/365.25
              Origin: time dob
  Enter on or after: time doe
         ID variable: id
                Note: ageband<=40 trimmed
Estimated standardized mortality rates
Reference-rate variable: rate

Number of records = 729

    ┌──────────────────────────────────────────────────────┐
    │ ageband    D      E      SMR    Lower    Upper  │
    │──────────────────────────────────────────────────────│
    │     40     6    5.62   1.0670  0.4793   2.3749 │
    │     50    18   18.75   0.9599  0.6048   1.5235 │
    │     60    22   22.85   0.9629  0.6340   1.4624 │
    └──────────────────────────────────────────────────────┘

  Notes: SMR = D/E = failures/expected failures (per 1000).
         Lower and Upper are bounds of 95% confidence intervals.
```

◁

Acknowledgments

Reference

Clayton, D. G., and M. Hills. 1993. *Statistical Models in Epidemiology*. Oxford: Oxford University Press.

Also see

Title

> **streg —** Parametric survival models

Description

streg performs maximum likelihood estimation for parametric regression survival-time models. streg can be used with single- or multiple-record or single- or multiple-failure st data. Survival models currently supported are exponential, Weibull, Gompertz, lognormal, loglogistic, and generalized gamma. Parametric frailty models and shared-frailty models are also fit using streg.

Also see [ST] **stcox** for proportional hazards models.

Quick start

Weibull survival model with covariates x1 and x2 using stset data
       streg x1 x2, distribution(weibull)

Use accelerated failure-time metric instead of proportional-hazards parameterization
       streg x1 x2, distribution(weibull) time

Different intercepts and ancillary parameters for strata identified by svar
       streg x1 x2, distribution(weibull) strata(svar)

Lognormal survival model
       streg x1 x2, distribution(lognormal)

Same as above, but also model frailty using the gamma distribution
       streg x1 x2, distribution(lognormal) frailty(gamma)

Specify shared frailty within groups identified by gvar
       streg x1 x2, distribution(lognormal) frailty(gamma) shared(gvar)

Menu

Statistics > Survival analysis > Regression models > Parametric survival models

Syntax

> streg [*indepvars*] [*if*] [*in*] [, *options*]

options	Description
[Model]	
<u>noc</u>onstant	suppress constant term
<u>d</u>istribution(<u>exp</u>onential)	exponential survival distribution
<u>d</u>istribution(<u>gom</u>pertz)	Gompertz survival distribution
<u>d</u>istribution(<u>loglog</u>istic)	loglogistic survival distribution
<u>d</u>istribution(<u>llog</u>istic)	synonym for distribution(loglogistic)
<u>d</u>istribution(<u>w</u>eibull)	Weibull survival distribution
<u>d</u>istribution(<u>logn</u>ormal)	lognormal survival distribution
<u>d</u>istribution(<u>lnorm</u>al)	synonym for distribution(lognormal)
<u>d</u>istribution(<u>gg</u>amma)	generalized gamma survival distribution
<u>fr</u>ailty(<u>gam</u>ma)	gamma frailty distribution
<u>fr</u>ailty(<u>inv</u>gaussian)	inverse-Gaussian distribution
time	use accelerated failure-time metric
[Model 2]	
<u>str</u>ata(*varname*)	strata ID variable
<u>off</u>set(*varname*)	include *varname* in model with coefficient constrained to 1
<u>sh</u>ared(*varname*)	shared frailty ID variable
<u>anc</u>illary(*varlist*)	use *varlist* to model the first ancillary parameter
anc2(*varlist*)	use *varlist* to model the second ancillary parameter
<u>c</u>onstraints(*constraints*)	apply specified linear constraints
[SE/Robust]	
vce(*vcetype*)	*vcetype* may be oim, <u>r</u>obust, <u>cl</u>uster *clustvar*, opg, <u>boot</u>strap, or <u>jack</u>knife
[Reporting]	
<u>l</u>evel(*#*)	set confidence level; default is level(95)
nohr	do not report hazard ratios
<u>tr</u>atio	report time ratios
<u>nos</u>how	do not show st setting information
<u>noh</u>eader	suppress header from coefficient table
<u>nolr</u>test	do not perform likelihood-ratio test
<u>nocns</u>report	do not display constraints
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
[Maximization]	
maximize_options	control the maximization process; seldom used
<u>coll</u>inear	keep collinear variables
<u>coef</u>legend	display legend instead of statistics

You must stset your data before using streg; see [ST] **stset**.

varlist may contain factor variables; see [U] **11.4.3 Factor variables**.

bayes, bootstrap, by, collect, fmm, fp, jackknife, mfp, mi estimate, nestreg, statsby, stepwise, and svy
  are allowed; see [U] **11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: streg** and [FMM] **fmm: streg**.

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix; see [MI] **mi estimate**.

shared(), vce(), and noheader are not allowed with the svy prefix; see [SVY] **svy**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**. However, weights may not be
  specified if you are using the bootstrap prefix with the streg command.

collinear and coeflegend do not appear in the dialog box.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

> ┌─── Model ───

noconstant; see [R] **Estimation options**.

distribution(*distname*) specifies the survival model to be fit. A specified distribution() is
  remembered from one estimation to the next when distribution() is not specified.

  For instance, typing streg x1 x2, distribution(weibull) fits a Weibull model. Subsequently,
  you do not need to specify distribution(weibull) to fit other Weibull regression models.

  All Stata estimation commands, including streg, redisplay results when you type the command
  name without arguments. To fit a model with no explanatory variables, type streg, distribu-
  tion(*distname*)....

frailty(gamma | invgaussian) specifies the assumed distribution of the frailty, or heterogeneity.
  The estimation results, in addition to the standard parameter estimates, will contain an estimate of
  the variance of the frailties and a likelihood-ratio test of the null hypothesis that this variance is
  zero. When this null hypothesis is true, the model reduces to the model with frailty(*distname*)
  not specified.

  A specified frailty() is remembered from one estimation to the next when distribution()
  is not specified. When you specify distribution(), the previously remembered specification of
  frailty() is forgotten.

time specifies that the model be fit in the accelerated failure-time metric rather than in the log
  relative-hazard metric. This option is valid only for the exponential and Weibull models because
  these are the only models that have both a proportional hazards and an accelerated failure-time
  parameterization. Regardless of metric, the likelihood function is the same, and models are equally
  appropriate viewed in either metric; it is just a matter of changing the interpretation.

  time must be specified at estimation.

> ┌─── Model 2 ───

strata(*varname*) specifies the stratification ID variable. Observations with equal values of the
  variable are assumed to be in the same stratum. Stratified estimates (with equal coefficients across
  strata but intercepts and ancillary parameters distinct for each stratum) are then obtained. This
  option is not available if frailty(*distname*) is specified.

offset(*varname*); see [R] **Estimation options**.

shared(*varname*) is valid with frailty() and specifies a variable defining those groups over which
  the frailty is shared, analogous to a random-effects model for panel data where *varname* defines the
  panels. frailty() specified without shared() treats the frailties as occurring at the observation
  level.

A specified shared() is remembered from one estimation to the next when distribution() is not specified. When you specify distribution(), the previously remembered specification of shared() is forgotten.

shared() may not be used with distribution(ggamma), vce(robust), vce(cluster *clustvar*), vce(opg), the svy prefix, or in the presence of delayed entries or gaps.

If shared() is specified without frailty() and there is no remembered frailty() from the previous estimation, frailty(gamma) is assumed to provide behavior analogous to stcox; see [ST] **stcox**.

ancillary(*varlist*) specifies that the ancillary parameter for the Weibull, lognormal, Gompertz, and loglogistic distributions and that the first ancillary parameter (sigma) of the generalized log-gamma distribution be estimated as a linear combination of *varlist*. This option may not be used with frailty(*distname*).

When an ancillary parameter is constrained to be strictly positive, the logarithm of the ancillary parameter is modeled as a linear combination of *varlist*.

anc2(*varlist*) specifies that the second ancillary parameter (kappa) for the generalized log-gamma distribution be estimated as a linear combination of *varlist*. This option may not be used with frailty(*distname*).

constraints(*constraints*); see [R] **Estimation options**.

> SE/Robust

vce(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory (oim, opg), that are robust to some kinds of misspecification (robust), that allow for intragroup correlation (cluster *clustvar*), and that use bootstrap or jackknife methods (bootstrap, jackknife); see [R] *vce_option*.

> Reporting

level(*#*); see [R] **Estimation options**.

nohr, which may be specified at estimation or upon redisplaying results, specifies that coefficients rather than exponentiated coefficients be displayed, that is, that coefficients rather than hazard ratios be displayed. This option affects only how coefficients are displayed, not how they are estimated.

This option is valid only for models with a natural proportional-hazards parameterization: exponential, Weibull, and Gompertz. These three models, by default, report hazard ratios (exponentiated coefficients).

tratio specifies that exponentiated coefficients, which are interpreted as time ratios, be displayed. tratio is appropriate only for the loglogistic, lognormal, and generalized gamma models, or for the exponential and Weibull models when fit in the accelerated failure-time metric.

tratio may be specified at estimation or upon replay.

noshow prevents streg from showing the key st variables. This option is rarely used because most people type stset, show or stset, noshow to set once and for all whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

noheader suppresses the output header, either at estimation or upon replay.

nolrtest is valid only with frailty models, in which case it suppresses the likelihood-ratio test for significant frailty.

nocnsreport; see [R] **Estimation options**.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels,
allbaselevels, nofvlabel, fvwrap(*#*), fvwrapon(*style*), cformat(*%fmt*), pformat(*%fmt*),
sformat(*%fmt*), and nolstretch; see [R] **Estimation options**.

┌─────────────┐
│ Maximization │
└─────────────┘

maximize_options: difficult, technique(*algorithm_spec*), iterate(*#*), [no]log, trace,
gradient, showstep, hessian, showtolerance, tolerance(*#*), ltolerance(*#*),
nrtolerance(*#*), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. These options are
seldom used.

Setting the optimization type to technique(bhhh) resets the default *vcetype* to vce(opg).

The following options are available with streg but are not shown in the dialog box:

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

> *Introduction*
> *Distributions*
> > *Weibull and exponential models*
> > *Gompertz model*
> > *Lognormal and loglogistic models*
> > *Generalized gamma model*
> *Examples*
> *Parameterization of ancillary parameters*
> *Stratified estimation*
> *(Unshared-) frailty models*
> *Shared-frailty models*

Introduction

What follows is a brief summary of what you can do with streg. For a complete tutorial, see
Cleves, Gould, and Marchenko (2016), which devotes four chapters to this topic.

Two often-used models for adjusting survivor functions for the effects of covariates are the
accelerated failure-time (AFT) model and the multiplicative or proportional hazards (PH) model. In
the AFT model, the natural logarithm of the survival time, $\log t$, is expressed as a linear function of
the covariates, yielding the linear model

$$\log t_j = \mathbf{x}_j \boldsymbol{\beta} + z_j$$

where $\mathbf{x}_j$ is a vector of covariates, $\boldsymbol{\beta}$ is a vector of regression coefficients, and z_j is the error with
density $f(\cdot)$. The distributional form of the error term determines the regression model. If we let $f(\cdot)$
be the normal density, the lognormal regression model is obtained. Similarly, by letting $f(\cdot)$ be the
logistic density, the loglogistic regression is obtained. Setting $f(\cdot)$ equal to the extreme-value density
yields the exponential and the Weibull regression models.

The effect of the AFT model is to change the time scale by a factor of $\exp(-\mathbf{x}_j\boldsymbol{\beta})$. Depending
on whether this factor is greater or less than 1, time is either accelerated or decelerated (degraded).
That is, if a subject at baseline experiences a probability of survival past time t equal to $S(t)$, then a
subject with covariates $\mathbf{x}_j$ would have probability of survival past time t equal to $S(\cdot)$ evaluated at

the point $\exp(-\mathbf{x}_j\boldsymbol{\beta})t$, instead. Thus accelerated failure time does not imply a positive acceleration of time with the increase of a covariate but instead implies a deceleration of time or, equivalently, an increase in the expected waiting time for failure.

In the PH model, the concomitant covariates have a multiplicative effect on the hazard function

$$h(t_j) = h_0(t)g(\mathbf{x}_j)$$

for some $h_0(t)$, and for $g(\mathbf{x}_j)$, a nonnegative function of the covariates. A popular choice, and the one adopted here, is to let $g(\mathbf{x}_j) = \exp(\mathbf{x}_j\boldsymbol{\beta})$. The function $h_0(t)$ may either be left unspecified, yielding the Cox proportional hazards model (see [ST] **stcox**), or take a specific parametric form. For the `streg` command, $h_0(t)$ is assumed to be parametric. Three regression models are currently implemented as PH models: the exponential, Weibull, and Gompertz models. The exponential and Weibull models are implemented as both AFT and PH models, and the Gompertz model is implemented only in the PH metric.

The above model allows for the presence of an intercept term, β_0, within $\mathbf{x}_j\boldsymbol{\beta}$. Thus what is commonly referred to as the *baseline hazard function*—the hazard when all covariates are zero—is actually equal to $h_0(t)\exp(\beta_0)$. That is, the intercept term serves to scale the baseline hazard. Of course, specifying `noconstant` suppresses the intercept or equivalently constrains β_0 to equal zero.

`streg` is suitable only for data that have been `stset`. By `stset`ting your data, you define the variables `_t0`, `_t`, and `_d`, which serve as the trivariate response variable (t_0, t, d). Each response corresponds to a period under observation, $(t_0, t]$, resulting in either failure $(d = 1)$ or right-censoring $(d = 0)$ at time t. As a result, `streg` is appropriate for data exhibiting delayed entry, gaps, time-varying covariates, and even multiple-failure data.

Distributions

Six parametric survival distributions are currently supported by `streg`. The parameterization and ancillary parameters for each distribution are summarized in table 1:

Table 1. Parametric survival distributions supported by `streg`

Distribution	Metric	Survivor function	Parameterization	Ancillary parameters
Exponential	PH	$\exp(-\lambda_j t_j)$	$\lambda_j = \exp(\mathbf{x}_j\boldsymbol{\beta})$	
Exponential	AFT	$\exp(-\lambda_j t_j)$	$\lambda_j = \exp(-\mathbf{x}_j\boldsymbol{\beta})$	
Weibull	PH	$\exp(-\lambda_j t_j^p)$	$\lambda_j = \exp(\mathbf{x}_j\boldsymbol{\beta})$	p
Weibull	AFT	$\exp(-\lambda_j t_j^p)$	$\lambda_j = \exp(-p\mathbf{x}_j\boldsymbol{\beta})$	p
Gompertz	PH	$\exp\{-\lambda_j\gamma^{-1}(e^{\gamma t_j} - 1)\}$	$\lambda_j = \exp(\mathbf{x}_j\boldsymbol{\beta})$	γ
Lognormal	AFT	$1 - \Phi\left\{\frac{\log(t_j)-\mu_j}{\sigma}\right\}$	$\mu_j = \mathbf{x}_j\boldsymbol{\beta}$	σ
Loglogistic	AFT	$\{1 + (\lambda_j t_j)^{1/\gamma}\}^{-1}$	$\lambda_j = \exp(-\mathbf{x}_j\boldsymbol{\beta})$	γ
Generalized gamma				
if $\kappa > 0$	AFT	$1 - I(\gamma, u)$	$\mu_j = \mathbf{x}_j\boldsymbol{\beta}$	σ, κ
if $\kappa = 0$	AFT	$1 - \Phi(z)$	$\mu_j = \mathbf{x}_j\boldsymbol{\beta}$	σ, κ
if $\kappa < 0$	AFT	$I(\gamma, u)$	$\mu_j = \mathbf{x}_j\boldsymbol{\beta}$	σ, κ

where PH = proportional hazards, AFT = accelerated failure time, and $\Phi(z)$ is the standard normal cumulative distribution. For the generalized gamma, $\gamma = |\kappa|^{-2}$, $u = \gamma \exp(|\kappa| z)$, $I(a, x)$ is the incomplete gamma function, and $z = \text{sign}(\kappa)\{\log(t_j) - \mu_j\}/\sigma$.

Plotted in figure 1 are example hazard functions for five of the six distributions. The exponential hazard (not separately plotted) is a special case of the Weibull hazard when the Weibull ancillary parameter $p = 1$. The generalized gamma (not plotted) is extremely flexible and therefore can take many shapes.
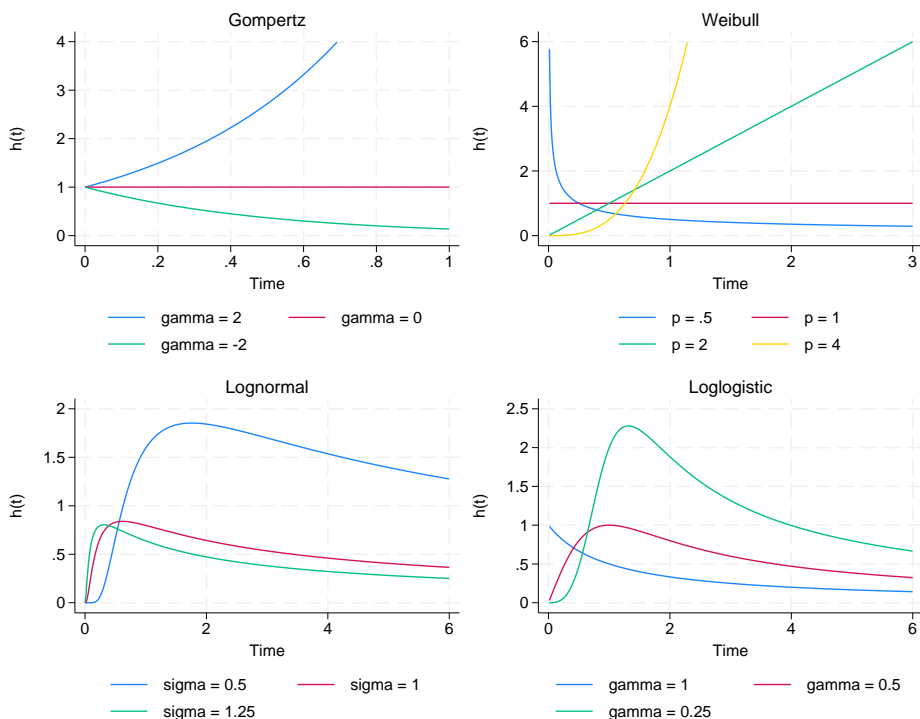
Figure 1. Example plots of hazard functions

Weibull and exponential models

The Weibull and exponential models are parameterized as both PH and AFT models. The Weibull distribution is suitable for modeling data with monotone hazard rates that either increase or decrease exponentially with time, whereas the exponential distribution is suitable for modeling data with constant hazard (see figure 1).

For the PH model, $h_0(t) = 1$ for exponential regression, and $h_0(t) = p\, t^{p-1}$ for Weibull regression, where p is the shape parameter to be estimated from the data. Some authors refer not to p but to $\sigma = 1/p$.

The AFT model is written as

$$\log(t_j) = \mathbf{x}_j\boldsymbol{\beta}^* + z_j$$

where z_j has an extreme-value distribution scaled by σ. Let $\boldsymbol{\beta}$ be the vector of regression coefficients derived from the PH model so that $\boldsymbol{\beta}^* = -\sigma\boldsymbol{\beta}$. This relationship holds only if the ancillary parameter, p, is a constant; it does not hold when the ancillary parameter is parameterized in terms of covariates.

streg uses, by default, for the exponential and Weibull models, the proportional-hazards metric simply because it eases comparison with those results produced by stcox (see [ST] **stcox**). You can, however, specify the time option to choose the accelerated failure-time parameterization.

The Weibull hazard and survivor functions are

$$h(t) = p\lambda t^{p-1}$$

$$S(t) = \exp(-\lambda t^p)$$

where λ is parameterized as described in table 1. If $p = 1$, these functions reduce to those of the exponential.

Gompertz model

The Gompertz regression is parameterized only as a PH model. First described in 1825, this model has been extensively used by medical researchers and biologists modeling mortality data. The Gompertz distribution implemented is the two-parameter function as described in Lee and Wang (2013), with the following hazard and survivor functions:

$$h(t) = \lambda \exp(\gamma t)$$

$$S(t) = \exp\{-\lambda\gamma^{-1}(e^{\gamma t} - 1)\}$$

The model is implemented by parameterizing $\lambda_j = \exp(\mathbf{x}_j\boldsymbol{\beta})$, implying that $h_0(t) = \exp(\gamma t)$, where γ is an ancillary parameter to be estimated from the data.

This distribution is suitable for modeling data with monotone hazard rates that either increase or decrease exponentially with time (see figure 1).

When γ is positive, the hazard function increases with time; when γ is negative, the hazard function decreases with time; and when γ is zero, the hazard function is equal to λ for all t, so the model reduces to an exponential.

Some recent survival analysis texts, such as Klein and Moeschberger (2003), restrict γ to be strictly positive. If $\gamma < 0$, then as t goes to infinity, the survivor function, $S(t)$, exponentially decreases to a nonzero constant, implying that there is a nonzero probability of never failing (living forever). That is, there is always a nonzero hazard rate, yet it decreases exponentially. By restricting γ to be positive, we know that the survivor function always goes to zero as t tends to infinity.

Although the above argument may be desirable from a mathematical perspective, in Stata's implementation, we took the more traditional approach of not restricting γ. We did this because, in survival studies, subjects are not monitored forever—there is a date when the study ends, and in many investigations, specifically in medical research, an exponentially decreasing hazard rate is clinically appealing.

Lognormal and loglogistic models

The lognormal and loglogistic models are implemented only in the AFT form. These two distributions are similar and tend to produce comparable results. For the lognormal distribution, the natural logarithm of time follows a normal distribution; for the loglogistic distribution, the natural logarithm of time follows a logistic distribution.

The lognormal survivor and density functions are

$$S(t) = 1 - \Phi\left\{\frac{\log(t) - \mu}{\sigma}\right\}$$

$$f(t) = \frac{1}{t\sigma\sqrt{2\pi}}\exp\left[\frac{-1}{2\sigma^2}\left\{\log(t) - \mu\right\}^2\right]$$

where $\Phi(z)$ is the standard normal cumulative distribution function.

The lognormal regression is implemented by setting $\mu_j = \mathbf{x}_j\boldsymbol{\beta}$ and treating the standard deviation, σ, as an ancillary parameter to be estimated from the data.

The loglogistic regression is obtained if z_j has a logistic density. The loglogistic survivor and density functions are

$$S(t) = \left\{1 + (\lambda t)^{1/\gamma}\right\}^{-1}$$

$$f(t) = \frac{\lambda^{1/\gamma}t^{1/\gamma-1}}{\gamma\left\{1 + (\lambda t)^{1/\gamma}\right\}^2}$$

This model is implemented by parameterizing $\lambda_j = \exp(-\mathbf{x}_j\boldsymbol{\beta})$ and treating the scale parameter γ as an ancillary parameter to be estimated from the data.

Unlike the exponential, Weibull, and Gompertz distributions, the lognormal and the loglogistic distributions are indicated for data exhibiting nonmonotonic hazard rates, specifically initially increasing and then decreasing rates (figure 1).

Thus far we have considered the exponential, Weibull, lognormal, and loglogistic models. These models are sufficiently flexible for many datasets, but further flexibility can be obtained with the generalized gamma model, described below. Alternatively, you might consider using a Royston–Parmar model (Royston and Parmar 2002; Lambert and Royston 2009). Royston–Parmar models are highly flexible alternatives to the exponential, Weibull, lognormal, and loglogistic models that allow extension from proportional hazards to proportional odds and to scaled probit models. Additional flexibility can be obtained with restricted cubic spline functions as alternatives to the linear functions of log time considered in *Introduction*. See Royston and Lambert (2011) for a thorough treatment of this topic.

Generalized gamma model

The generalized gamma model is implemented only in the AFT form. The three-parameter generalized gamma survivor and density functions are

$$
S(t) = \begin{cases} 1 - I\big(\gamma, u\big) & \text{if } \kappa > 0 \\ 1 - \Phi(z) & \text{if } \kappa = 0 \\ I\big(\gamma, u\big) & \text{if } \kappa < 0 \end{cases}
$$

$$
f(t) = \begin{cases} \frac{\gamma^{\gamma}}{\sigma t \sqrt{\gamma} \Gamma(\gamma)} \exp\big(z\sqrt{\gamma} - u\big) & \text{if } \kappa \neq 0 \\ \frac{1}{\sigma t \sqrt{2\pi}} \exp(-z^2/2) & \text{if } \kappa = 0 \end{cases}
$$

where $\gamma = |\kappa|^{-2}$, $z = \text{sign}(\kappa)\{\log(t) - \mu\}/\sigma$, $u = \gamma \exp(|\kappa|z)$, $\Phi(z)$ is the standard normal cumulative distribution function, and $I(a, x)$ is the incomplete gamma function. See the gammap(a,x) entry in [FN] **Statistical functions** to see how the incomplete gamma function is implemented in Stata.

This model is implemented by parameterizing $\mu_j = \mathbf{x}_j\boldsymbol{\beta}$ and treating the parameters κ and σ as ancillary parameters to be estimated from the data.

The hazard function of the generalized gamma distribution is extremely flexible, allowing for many possible shapes, including as special cases the Weibull distribution when $\kappa = 1$, the exponential when $\kappa = 1$ and $\sigma = 1$, and the lognormal distribution when $\kappa = 0$. The generalized gamma model is, therefore, commonly used for evaluating and selecting an appropriate parametric model for the data. The Wald or likelihood-ratio test can be used to test the hypotheses that $\kappa = 1$ or that $\kappa = 0$.

❑ Technical note

Prior to Stata 14, streg's option distribution(gamma) was used to fit generalized gamma models. As of Stata 14, the new option for fitting these models is distribution(ggamma). The old option continues to work under version control. This option was renamed to avoid confusion with mestreg's option distribution(gamma) for fitting mixed-effects survival gamma models; see [ME] **mestreg**.

❑

Examples

▷ Example 1

The Weibull distribution provides a good illustration of streg because this distribution is parameterized as both AFT and PH and serves to compare and contrast the two approaches.

We wish to analyze an experiment testing the ability of emergency generators with new-style bearings to withstand overloads. This dataset is described in [ST] **stcox**. This time, we wish to fit a Weibull model:

```
. use https://www.stata-press.com/data/r18/kva
(Generator experiment)
. stset failtime
  (output omitted)
. streg load bearings, distribution(weibull)

        Failure _d: 1 (meaning all fail)
  Analysis time _t: failtime

Fitting constant-only model:
Iteration 0:  Log likelihood = -13.666193
Iteration 1:  Log likelihood = -9.7427276
Iteration 2:  Log likelihood = -9.4421169
Iteration 3:  Log likelihood = -9.4408287
Iteration 4:  Log likelihood = -9.4408286

Fitting full model:
Iteration 0:  Log likelihood = -9.4408286
Iteration 1:  Log likelihood =  -2.078323
Iteration 2:  Log likelihood =  5.2226016
Iteration 3:  Log likelihood =  5.6745808
Iteration 4:  Log likelihood =  5.6934031
Iteration 5:  Log likelihood =  5.6934189
Iteration 6:  Log likelihood =  5.6934189

Weibull PH regression
```

No. of subjects = 12				Number of obs =		12
No. of failures = 12						
Time at risk = 896						
				LR chi2(2)	=	30.27
Log likelihood = 5.6934189				Prob > chi2	=	0.0000

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
load	1.599315	.1883807	3.99	0.000	1.269616	2.014631
bearings	.1887995	.1312109	-2.40	0.016	.0483546	.7371644
_cons	2.51e-20	2.66e-19	-4.26	0.000	2.35e-29	2.68e-11
/ln_p	2.051552	.2317074	8.85	0.000	1.597414	2.505691
p	7.779969	1.802677			4.940241	12.25202
1/p	.1285352	.0297826			.0816192	.2024193

Note: _cons estimates baseline hazard.

Because we did not specify otherwise, the estimation took place in the hazard metric, which is the default for distribution(weibull). The estimates are directly comparable to those produced by stcox: stcox estimated a hazard ratio of 1.526 for load and 0.0636 for bearings.

However, we estimated the baseline hazard function as well, assuming that it is Weibull. The estimates are the full maximum-likelihood estimates. The shape parameter is fit as $\ln p$, but streg then reports p and $1/p = \sigma$ so that you can think about the parameter however you wish.

We find that p is greater than 1, which means that the hazard of failure increases with time and, here, increases dramatically. After 100 hours, the bearings are more than 1 million times more likely to fail per second than after 10 hours (or, to be precise, $(100/10)^{7.78-1}$). From our knowledge of generators, we would expect this; it is the accumulation of heat due to friction that causes bearings to expand and seize.

◁

❑ Technical note

Regression results are often presented in a metric other than the natural regression coefficients, that is, as hazard ratios, relative risk ratios, odds ratios, etc. In those cases, standard errors are calculated using the delta method.

However, the Z test and p-values given are calculated from the natural regression coefficients and standard errors. Although a test based on, say, a hazard ratio and its standard error would be asymptotically equivalent to that based on a regression coefficient, in real samples a hazard ratio will tend to have a more skewed distribution because it is an exponentiated regression coefficient. Also, it is more natural to think of these tests as testing whether a regression coefficient is nonzero, rather than testing whether a transformed regression coefficient is unequal to some nonzero value (one for a hazard ratio).

Finally, the confidence intervals given are obtained by transforming the endpoints of the corresponding confidence interval for the untransformed regression coefficient. This ensures that, say, strictly positive quantities such as hazard ratios have confidence intervals that do not overlap zero.

❑

▷ Example 2

The previous estimation took place in the PH metric, and exponentiated coefficients—hazard ratios—were reported. If we want to see the unexponentiated coefficients, we could redisplay results and specify the nohr option:

```
. streg, nohr

Weibull PH regression

No. of subjects =   12                              Number of obs =       12
No. of failures =   12
Time at risk    =  896
                                                    LR chi2(2)    =    30.27
Log likelihood = 5.6934189                          Prob > chi2   =   0.0000
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
load	.4695753	.1177884	3.99	0.000	.2387143	.7004363
bearings	-1.667069	.6949745	-2.40	0.016	-3.029194	-.3049443
_cons	-45.13191	10.60663	-4.26	0.000	-65.92053	-24.34329
/ln_p	2.051552	.2317074	8.85	0.000	1.597414	2.505691
p	7.779969	1.802677			4.940241	12.25202
1/p	.1285352	.0297826			.0816192	.2024193

◁

▷ Example 3

We could just as well have fit this model in the AFT metric:

```
. streg load bearings, distribution(weibull) time nolog
      Failure _d: 1 (meaning all fail)
  Analysis time _t: failtime

Weibull AFT regression

No. of subjects =   12                              Number of obs =       12
No. of failures =   12
Time at risk    =  896
                                                    LR chi2(2)    =   30.27
Log likelihood = 5.6934189                          Prob > chi2   =  0.0000
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]
load	-.060357	.0062214	-9.70	0.000	-.0725507 -.0481632
bearings	.2142771	.0746451	2.87	0.004	.0679753 .3605789
_cons	5.80104	.1752301	33.11	0.000	5.457595 6.144485
/ln_p	2.051552	.2317074	8.85	0.000	1.597414 2.505691
p	7.779969	1.802677			4.940241 12.25202
1/p	.1285352	.0297826			.0816192 .2024193

This is the same model we previously fit, but it is presented in a different metric. Calling the previous coefficients b, these coefficients are $-\sigma b = -b/p$. For instance, in the previous example, the coefficient on load was reported as roughly 0.47, and $-0.47/7.78 = -0.06$.

◁

▷ Example 4

streg may also be applied to more complicated data. Below we have multiple records per subject on a failure that can occur repeatedly:

```
. use https://www.stata-press.com/data/r18/mfail3
. stdescribe
```

		Per subject			
Category	Total	Mean	Min	Median	Max
Number of subjects	926				
Number of records	1734	1.87257	1	2	4
Entry time (first)		0	0	0	0
Exit time (final)		470.6857	1	477	960
Subjects with gap	6				
Time on gap	411	68.5	16	57.5	133
Time at risk	435444	470.2419	1	477	960
Failures	808	.8725702	0	1	3

In this dataset, subjects have up to four records (most have two) and have up to three failures (most have one) and, although you cannot tell from the above output, the data have time-varying covariates, as well. There are even six subjects with gaps in their histories, meaning that, for a while, they went unobserved. Although we could estimate in the AFT metric, it is easier to interpret results in the PH metric (or the log relative-hazard metric, as it is also known):

```
. streg x1 x2, distribution(weibull) vce(robust)

Fitting constant-only model:
Iteration 0:  Log pseudolikelihood = -1398.2504
Iteration 1:  Log pseudolikelihood = -1382.8224
Iteration 2:  Log pseudolikelihood = -1382.7457
Iteration 3:  Log pseudolikelihood = -1382.7457

Fitting full model:
Iteration 0:  Log pseudolikelihood = -1382.7457
Iteration 1:  Log pseudolikelihood = -1328.4186
Iteration 2:  Log pseudolikelihood = -1326.4483
Iteration 3:  Log pseudolikelihood = -1326.4449
Iteration 4:  Log pseudolikelihood = -1326.4449

Weibull PH regression
```

No. of subjects =   926   Number of obs = 1,734
No. of failures =   808
Time at risk   = 435,444
                                                           Wald chi2(2) = 154.45
Log pseudolikelihood = -1326.4449   Prob > chi2   = 0.0000

                                  (Std. err. adjusted for 926 clusters in id)

	Haz. ratio	Robust std. err.	z	P>|z|	[95% conf. interval]	
x1	2.240069	.1812848	9.97	0.000	1.911504	2.625111
x2	.3206515	.0504626	-7.23	0.000	.2355458	.436507
_cons	.0006962	.0001792	-28.25	0.000	.0004204	.001153
/ln_p	.1771265	.0310111	5.71	0.000	.1163458	.2379071
p	1.193782	.0370205			1.123384	1.268591
1/p	.8376738	.0259772			.7882759	.8901674

Note: **_cons** estimates baseline hazard.

A one-unit change in x1 approximately doubles the hazard of failure, whereas a one-unit change in x2 cuts the hazard to one-third its previous value. We also see that these data are close to being exponentially distributed; p is nearly 1.

Above we mentioned that interpreting results in the PH metric is easier, though regression coefficients are not difficult to interpret in the AFT metric. A positive coefficient means that time is decelerated by a unit increase in the covariate in question. This may seem awkward, but think of this instead as a unit increase in the covariate causing a delay in failure and thus *increasing* the expected time until failure.

The difficulty that arises with the AFT metric is merely that it places an emphasis on log(time-to-failure) rather than risk (hazard) of failure. With this emphasis usually comes a desire to predict the time to failure, and therein lies the difficulty with complex survival data. Predicting the log(time to failure) with `predict` assumes that the subject is at risk from time 0 until failure and has a fixed covariate pattern over this period. With these data, such assumptions produce predictions having little to do with the test subjects, who exhibit not only time-varying covariates but also multiple failures.

Predicting time to failure with complex survival data is difficult regardless of the metric under which estimation took place. Those who estimate in the PH metric are probably used to dealing with results from Cox regression, of which predicted time to failure is typically not the focus.

◁

▷ Example 5

The multiple-failure data above are close enough to exponentially distributed that we will reestimate using exponential regression:

```
. streg x1 x2, distribution(exp) vce(robust)

Iteration 0:   Log pseudolikelihood = -1398.2504
Iteration 1:   Log pseudolikelihood = -1343.6083
Iteration 2:   Log pseudolikelihood = -1341.5932
Iteration 3:   Log pseudolikelihood = -1341.5893
Iteration 4:   Log pseudolikelihood = -1341.5893

Exponential PH regression

No. of subjects =       926                          Number of obs =    1,734
No. of failures =       808
Time at risk    = 435,444
                                                     Wald chi2(2)  = 166.92
Log pseudolikelihood = -1341.5893                    Prob > chi2   = 0.0000

                            (Std. err. adjusted for 926 clusters in id)
```

_t	Haz. ratio	Robust std. err.	z	P>|z|	[95% conf. interval]
x1	2.19065	.1684399	10.20	0.000	1.884186	2.54696
x2	.3037259	.0462489	−7.83	0.000	.2253552	.4093511
_cons	.0024536	.0001535	−96.05	0.000	.0021704	.0027738

Note: **_cons** estimates baseline hazard.

◁

❏ Technical note

For our "complex" survival data, we specified vce(robust) when fitting the Weibull and exponential models. This was because these data were stset with an id() variable, and given the time-varying covariates and multiple failures, it is important not to assume that the observations within each subject are independent. When we specified vce(robust), it was implicit that we were "clustering" on the groups defined by the id() variable.

You might sometimes have multiple observations per subject, which exist merely as a result of the data-organization mechanism and are not used to record gaps, time-varying covariates, or multiple failures. Such data could be collapsed into single-observation-per-subject data with no loss of information. In these cases, we refer to splitting the observations to form multiple observations per subject as *noninformative*. When the episode-splitting is noninformative, the model-based (nonrobust) standard errors produced will be the same as those produced when the data are collapsed into single records per subject. Thus, for these type of data, clustering of these multiple observations that results from specifying vce(robust) is not critical.

❏

▷ Example 6

A reasonable question to ask is, "Given that we have several possible parametric models, how can we select one?" When parametric models are nested, the likelihood-ratio or Wald test can be used to discriminate between them. This can certainly be done for Weibull versus exponential or gamma versus Weibull or lognormal. When models are not nested, however, these tests are inappropriate, and the task of discriminating between models becomes more difficult. A common approach to this

problem is to use the Akaike information criterion (AIC). Akaike (1974) proposed penalizing each log likelihood to reflect the number of parameters being estimated in a particular model and then comparing them. Here the AIC can be defined as

$$\text{AIC} = -2(\text{log likelihood}) + 2(c + p + 1)$$

where c is the number of model covariates and p is the number of model-specific ancillary parameters listed in table 1. Although the best-fitting model is the one with the largest log likelihood, the preferred model is the one with the smallest AIC value. The AIC value may be obtained by using the estat ic postestimation command; see [R] **estat ic**.

Using cancer.dta distributed with Stata, let's first fit a generalized gamma model and test the hypothesis that $\kappa = 0$ (test for the appropriateness of the lognormal) and then test the hypothesis that $\kappa = 1$ (test for the appropriateness of the Weibull).

```
. use https://www.stata-press.com/data/r18/cancer
(Patient survival in drug trial)
. stset studytime, failure(died)
  (output omitted)
. replace drug = drug==2 | drug==3  // 0, placebo : 1, nonplacebo
(48 real changes made)
. streg drug age, distribution(ggamma) nolog
        Failure _d: died
  Analysis time _t: studytime
Generalized gamma AFT regression
No. of subjects =   48                         Number of obs =       48
No. of failures =   31
Time at risk    = 744
                                               LR chi2(2)    =   36.07
Log likelihood = -42.452006                    Prob > chi2   =  0.0000
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
drug	1.394658	.2557198	5.45	0.000	.893456	1.895859
age	-.0780416	.0227978	-3.42	0.001	-.1227245	-.0333587
_cons	6.456091	1.238457	5.21	0.000	4.02876	8.883421
/lnsigma	-.3793632	.183707	-2.07	0.039	-.7394222	-.0193041
/kappa	.4669252	.5419478	0.86	0.389	-.595273	1.529123
sigma	.684297	.1257101			.4773897	.980881

The Wald test of the hypothesis that $\kappa = 0$ (test for the appropriateness of the lognormal) is reported in the output above. The p-value is 0.389, suggesting that lognormal might be an adequate model for these data.

The Wald test for $\kappa = 1$ is

```
. test [kappa] = 1
 ( 1)  [/]kappa = 1
          chi2(  1) =    0.97
        Prob > chi2 =    0.3253
```

providing some support against rejecting the Weibull model.

We now fit the exponential, Weibull, loglogistic, and lognormal models separately. To directly compare coefficients, we will ask Stata to report the exponential and Weibull models in AFT form by specifying the time option. The output from fitting these models and the results from the generalized gamma model are summarized in table 2.

Table 2. Results obtained from streg, using cancer.dta with drug as an indicator variable

Parameter	Exponential	Weibull	Lognormal	Loglogistic	Generalized gamma
Age	−0.0886715	−0.0714323	−0.0833996	−0.0803289	−0.078042
Drug	1.682625	1.305563	1.445838	1.420237	1.394658
Constant	7.146218	6.289679	6.580887	6.446711	6.456091
Ancillary		1.682751	0.751136	0.429276	0.684297
Kappa					0.466925
Log likelihood	−48.397094	−42.931335	−42.800864	−43.21698	−42.452006
AIC	102.7942	93.86267	93.60173	94.43396	94.90401

The largest log likelihood was obtained for the generalized gamma model; however, the lognormal model is preferred by the AIC.

◁

Parameterization of ancillary parameters

By default, all ancillary parameters are estimated as being constant. For example, the ancillary parameter, p, of the Weibull distribution is assumed to be a constant that is not dependent on any covariates. streg's ancillary() and anc2() options allow for complete parameterization of parametric survival models. By specifying, for example,

```
. streg x1 x2, distribution(weibull) ancillary(x2 z1 z2)
```

both λ and the ancillary parameter, p, are parameterized in terms of covariates.

Ancillary parameters are usually restricted to be strictly positive, in which case the logarithm of the ancillary parameter is modeled using a linear predictor, which can assume any value on the real line.

▷ Example 7

Consider a dataset in which we model the time until hip fracture as Weibull for patients on the basis of age, sex, and whether the patient wears a hip-protective device (variable protect). We believe that the hazard is scaled according to sex and the presence of the device but believe the hazards for both sexes to be of different *shapes*.

```
. use https://www.stata-press.com/data/r18/hip3, clear
(Hip-fracture study)

. streg protect age, distribution(weibull) ancillary(male) nolog

        Failure _d: fracture
  Analysis time _t: time1
        ID variable: id

Weibull PH regression

No. of subjects =    148                        Number of obs =      206
No. of failures =     37
Time at risk    = 1,703

                                                LR chi2(2)    =   39.80
Log likelihood = -69.323532                     Prob > chi2   =  0.0000
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]
_t					
protect	-2.130058	.3567005	-5.97	0.000	-2.829178 -1.430938
age	.0939131	.0341107	2.75	0.006	.0270573 .1607689
_cons	-10.17575	2.551821	-3.99	0.000	-15.17722 -5.174269
ln_p					
male	-.4887189	.185608	-2.63	0.008	-.8525039 -.1249339
_cons	.4540139	.1157915	3.92	0.000	.2270667 .6809611

From our estimation results, we see that $\widehat{\ln(p)} = 0.454$ for females and $\widehat{\ln(p)} = 0.454 - 0.489 = -0.035$ for males. Thus $\widehat{p} = 1.57$ for females and $\widehat{p} = 0.97$ for males. When we combine this with the main equation in the model, the estimated hazards are then

$$\widehat{h}(t_j|\mathbf{x}_j) = \begin{cases} \exp\left(-10.18 - 2.13\texttt{protect}_j + 0.09\texttt{age}_j\right) 1.57t_j^{0.57} & \text{if female} \\ \exp\left(-10.18 - 2.13\texttt{protect}_j + 0.09\texttt{age}_j\right) 0.97t_j^{-0.03} & \text{if male} \end{cases}$$

If we believe this model, we would say that the hazard for males given `age` and `protect` is almost constant over time.

Contrast this with what we obtain if we type

```
. streg protect age if male, distribution(weibull)
. streg protect age if !male, distribution(weibull)
```

which is completely general, because not only the shape parameter, p, will differ over both sexes but also the regression coefficients.

◁

The `anc2()` option is for use only with the gamma regression model, because it contains two ancillary parameters—`anc2()` is used to parameterize κ.

Stratified estimation

When we type

```
. streg xvars, distribution(distname) strata(varname)
```

we are asking that a completely stratified model be fit. By *completely stratified*, we mean that both the model's intercept and any ancillary parameters are allowed to vary for each level of the strata variable. That is, we are constraining the coefficients on the covariates to be the same across strata but allowing the intercept and ancillary parameters to vary.

▷ Example 8

We demonstrate this by fitting a stratified Weibull model to the cancer data, with the drug variable left in its original state: drug==1 refers to the placebo, and drug==2 and drug==3 correspond to two alternative treatments.

```
. use https://www.stata-press.com/data/r18/cancer
(Patient survival in drug trial)
. stset studytime, failure(died)
 (output omitted)
. streg age, distribution(weibull) strata(drug) nolog
```

```
        Failure _d: died
  Analysis time _t: studytime
```

Weibull PH regression
Strata variable: drug

No. of subjects = 48	Number of obs = 48
No. of failures = 31	
Time at risk = 744	
	LR chi2(3) = 16.58
Log likelihood = −41.113074	Prob > chi2 = 0.0009

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]
_t						
age	.1212332	.0367538	3.30	0.001	.049197	.1932694
drug						
Other	−4.561178	2.339448	−1.95	0.051	−9.146411	.0240556
NA	−3.715737	2.595986	−1.43	0.152	−8.803776	1.372302
_cons	−10.36921	2.341022	−4.43	0.000	−14.95753	−5.780896
ln_p						
drug						
Other	.4872195	.332019	1.47	0.142	−.1635257	1.137965
NA	.2194213	.4079989	0.54	0.591	−.5802418	1.019084
_cons	.4541282	.1715663	2.65	0.008	.1178645	.7903919

◁

Completely stratified models are fit by including a stratum variable as a factor variable in the main equation and in any of the ancillary equations. The strata() option is thus merely a shorthand method for including i.drug in both the main equation and the ancillary equation(s).

We associate the term "stratification" with this process by noting that the intercept term of the main equation is a component of the baseline hazard (or baseline survivor) function. By allowing this intercept, as well as the ancillary shape parameter, to vary with respect to the strata, we allow the baseline functions to completely vary over the strata, analogous to a stratified Cox model.

▷ Example 9

We can produce a less-stratified model by specifying a factor variable in the `ancillary()` option.

```
. streg age, distribution(weibull) ancillary(i.drug) nolog

        Failure _d: died
  Analysis time _t: studytime

Weibull PH regression

No. of subjects =  48                              Number of obs =       48
No. of failures =  31
Time at risk    = 744
                                                   LR chi2(1)     =   9.61
Log likelihood = -44.596379                        Prob > chi2    = 0.0019
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
_t						
age	.1126419	.0362786	3.10	0.002	.0415373	.1837466
_cons	-10.95772	2.308489	-4.75	0.000	-15.48227	-6.433162
ln_p						
drug						
Other	-.3279568	.11238	-2.92	0.004	-.5482176	-.107696
NA	-.4775351	.1091141	-4.38	0.000	-.6913948	-.2636755
_cons	.6684086	.1327284	5.04	0.000	.4082657	.9285514

By doing this, we are restricting not only the coefficients on the covariates to be the same across "strata" but also the intercept, while allowing only the ancillary parameter to differ.

◁

By using `ancillary()` or `strata()`, we may thus consider a wide variety of models, depending on what we believe about the effect of the covariate(s) in question. For example, when fitting a Weibull PH model to the cancer data, we may choose from many models, depending on what we want to assume is the effect of the categorical variable `drug`. For all models considered below, we assume implicitly that the effect of `age` is proportional on the hazard function.

1. `drug` has no effect:

    `. streg age, distribution(weibull)`

2. The effect of `drug` is proportional on the hazard (scale), and the effect of `age` is the same for each level of `drug`:

    `. streg age i.drug, distribution(weibull)`

3. `drug` affects the shape of the hazard, and the effect of `age` is the same for each level of `drug`:

    `. streg age, distribution(weibull) ancillary(i.drug)`

4. `drug` affects both the scale and shape of the hazard, and the effect of `age` is the same for each level of `drug`:

    `. streg age, distribution(weibull) strata(drug)`

5. `drug` affects both the scale and shape of the hazard, and the effect of `age` is different for each level of `drug`:

> `. streg drug##c.age, distribution(weibull) strata(drug)`

These models may be compared using Wald or likelihood-ratio tests when the models in question are nested (such as 3 nested within 4) or by using the AIC for nonnested models.

Everything we said regarding the modeling of ancillary parameters and stratification applies to AFT models as well, for which interpretations may be stated in terms of the baseline survivor function, that is, the unaccelerated probability of survival past time t.

❑ Technical note

When fitting PH models, `streg` will, by default, display the exponentiated regression coefficients, labeled as hazard ratios. However, in our previous examples using `ancillary()` and `strata()`, the regression outputs displayed the untransformed coefficients instead. This change in behavior has to do with the modeling of the ancillary parameter. When we use one or more covariates from the main equation to model an ancillary parameter, hazard ratios (and time ratios for AFT models) lose their interpretation. `streg`, as a precaution, disallows the display of hazard/time ratios when `ancillary()`, `anc2()`, or `strata()` is specified.

Keep this in mind when comparing results across various model specifications. For example, when comparing a stratified Weibull PH model to a standard Weibull PH model, be sure that the latter is displayed using the `nohr` option.

❑

(Unshared-) frailty models

A frailty model is a survival model with unobservable heterogeneity, or *frailty*. At the observation level, frailty is introduced as an unobservable multiplicative effect, α, on the hazard function, such that

$$h(t|\alpha) = \alpha h(t)$$

where $h(t)$ is a nonfrailty hazard function, say, the hazard function of any of the six parametric models supported by `streg` described earlier in this entry. The frailty, α, is a random positive quantity and, for model identifiability, is assumed to have mean 1 and variance θ.

Exploiting the relationship between the cumulative hazard function and survivor function yields the expression for the survivor function, given the frailty

$$S(t|\alpha) = \exp\left\{-\int_0^t h(u|\alpha)du\right\} = \exp\left\{-\alpha\int_0^t \frac{f(u)}{S(u)}du\right\} = \{S(t)\}^\alpha$$

where $S(t)$ is the survivor function that corresponds to $h(t)$.

Because α is unobservable, it must be integrated out of $S(t|\alpha)$ to obtain the unconditional survivor function. Let $g(\alpha)$ be the probability density function of α, in which case an estimable form of our frailty model is achieved as

$$S_\theta(t) = \int_0^\infty S(t|\alpha)g(\alpha)d\alpha = \int_0^\infty \{S(t)\}^\alpha g(\alpha)d\alpha$$

Given the unconditional survivor function, we can obtain the unconditional hazard and density in the usual way:

$$f_\theta(t) = -\frac{d}{dt} S_\theta(t) \qquad h_\theta(t) = \frac{f_\theta(t)}{S_\theta(t)}$$

Hence, an unshared-frailty model is merely a typical parametric survival model, with the additional estimation of an overdispersion parameter, θ. In a standard survival regression, the likelihood calculations are based on $S(t)$, $h(t)$, and $f(t)$. In an unshared-frailty model, the likelihood is based analogously on $S_\theta(t)$, $h_\theta(t)$, and $f_\theta(t)$.

At this stage, the only missing piece is the choice of frailty distribution, $g(\alpha)$. In theory, any continuous distribution supported on the positive numbers that has expectation 1 and finite variance θ is allowed here. For mathematical tractability, however, we limit the choice to either the gamma$(1/\theta, \theta)$ distribution or the inverse-Gaussian distribution with parameters 1 and $1/\theta$, denoted as IG$(1, 1/\theta)$. The gamma(a, b) distribution has probability density function

$$g(x) = \frac{x^{a-1} e^{-x/b}}{\Gamma(a) b^a}$$

and the IG(a, b) distribution has density

$$g(x) = \left(\frac{b}{2\pi x^3} \right)^{1/2} \exp \left\{ -\frac{b}{2a} \left(\frac{x}{a} - 2 + \frac{a}{x} \right) \right\}$$

Therefore, performing the integrations described above will show that specifying `frailty(gamma)` will result in the frailty survival model (in terms of the nonfrailty survivor function, $S(t)$)

$$S_\theta(t) = [1 - \theta \log \{S(t)\}]^{-1/\theta}$$

Specifying `frailty(invgaussian)` will give

$$S_\theta(t) = \exp \left\{ \frac{1}{\theta} \left(1 - [1 - 2\theta \log \{S(t)\}]^{1/2} \right) \right\}$$

Regardless of the choice of frailty distribution, $\lim_{\theta \to 0} S_\theta(t) = S(t)$, and thus the frailty model reduces to $S(t)$ when there is no heterogeneity present.

When using frailty models, distinguish between the hazard faced by the individual (subject), $\alpha h(t)$, and the "average" hazard for the population, $h_\theta(t)$. Similarly, an individual will have probability of survival past time t equal to $\{S(t)\}^\alpha$, whereas $S_\theta(t)$ will measure the proportion of the population surviving past time t. You specify $S(t)$ as before with `distribution(`*distname*`)`, and the list of possible parametric forms for $S(t)$ is given in table 1. Thus when you specify `distribution()` you are specifying a model for an individual with frailty equal to one. Specifying `frailty(`*distname*`)` determines which of the two above forms for $S_\theta(t)$ is used.

The output of the estimation remains unchanged from the nonfrailty version, except for the additional estimation of θ and a likelihood-ratio test of $H_0\colon \theta = 0$. For more information on frailty models, Hougaard (1986) offers an excellent introduction. For a Stata-specific overview, see Gutierrez (2002).

▷ Example 10

Consider as an example a survival analysis of data on women with breast cancer. Our hypothetical dataset consists of analysis times on 80 women with covariates age, smoking, and dietfat, which measures the average weekly calories from fat ($\times 10^3$) in the patient's diet over the course of the study.

```
. use https://www.stata-press.com/data/r18/bc
. list in 1/12
```

	age	smoking	dietfat	t	dead
1.	30	1	4.919	14.2	0
2.	50	0	4.437	8.21	1
3.	47	0	5.85	5.64	1
4.	49	1	5.149	4.42	1
5.	52	1	4.363	2.81	1
6.	29	0	6.153	35	0
7.	49	1	3.82	4.57	1
8.	27	1	5.294	35	0
9.	47	0	6.102	3.74	1
10.	59	0	4.446	2.29	1
11.	35	0	6.203	15.3	0
12.	26	0	4.515	35	0

The data are well fit by a Weibull model for the distribution of survival time conditional on age, smoking, and dietary fat. By omitting the dietfat variable from the model, we hope to introduce unobserved heterogeneity.

```
. stset t, fail(dead)
(output omitted )
. streg age smoking, distribution(weibull) frailty(gamma)

        Failure _d: dead
  Analysis time _t: t

Fitting Weibull model ...

Fitting constant-only model:
Iteration 0:  Log likelihood = -137.15363
Iteration 1:  Log likelihood =  -136.3927
Iteration 2:  Log likelihood = -136.01557
Iteration 3:  Log likelihood = -136.01202
Iteration 4:  Log likelihood = -136.01201

Fitting full model:
Iteration 0:  Log likelihood = -85.933969
Iteration 1:  Log likelihood =  -73.61173
Iteration 2:  Log likelihood = -68.999447
Iteration 3:  Log likelihood = -68.340858
Iteration 4:  Log likelihood = -68.136187
Iteration 5:  Log likelihood = -68.135804
Iteration 6:  Log likelihood = -68.135804
```

```
Weibull PH regression
Gamma frailty
No. of subjects =        80                            Number of obs =        80
No. of failures =        58
Time at risk     = 1,257.07
                                                       LR chi2(2)     = 135.75
Log likelihood = -68.135804                            Prob > chi2    = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.475948	.1379987	4.16	0.000	1.228811	1.772788
smoking	2.788548	1.457031	1.96	0.050	1.00143	7.764894
_cons	4.57e-11	2.38e-10	-4.57	0.000	1.70e-15	1.23e-06
/ln_p	1.087761	.222261	4.89	0.000	.6521376	1.523385
/lntheta	.3307466	.5250758	0.63	0.529	-.698383	1.359876
p	2.967622	.6595867			1.91964	4.587727
1/p	.3369701	.0748953			.2179729	.520931
theta	1.392007	.7309092			.4973889	3.895711

```
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard.
LR test of theta=0: chibar2(01) = 22.57          Prob >= chibar2 = 0.000
```

We could also use an inverse-Gaussian distribution to model the heterogeneity.

```
. streg age smoking, distribution(weibull) frailty(invgauss) nolog
        Failure _d: dead
  Analysis time _t: t
Weibull PH regression
Inverse-Gaussian frailty
No. of subjects =        80                            Number of obs =        80
No. of failures =        58
Time at risk     = 1,257.07
                                                       LR chi2(2)     = 125.44
Log likelihood = -73.838578                            Prob > chi2    = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.284133	.0463256	6.93	0.000	1.196473	1.378217
smoking	2.905409	1.252785	2.47	0.013	1.247892	6.764528
_cons	1.11e-07	2.34e-07	-7.63	0.000	1.83e-09	6.79e-06
/ln_p	.7173904	.1434382	5.00	0.000	.4362567	.9985241
/lntheta	.2374778	.8568064	0.28	0.782	-1.441832	1.916788
p	2.049079	.2939162			1.546906	2.714273
1/p	.4880241	.0700013			.3684228	.6464518
theta	1.268047	1.086471			.2364941	6.799082

```
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard.
LR test of theta=0: chibar2(01) = 11.16          Prob >= chibar2 = 0.000
```

The results are similar with respect to the choice of frailty distribution, with the gamma frailty model producing a slightly higher likelihood. Both models show a statistically significant level of unobservable heterogeneity because the p-value for the likelihood-ratio (LR) test of $H_0: \theta = 0$ is virtually zero in both cases.

❑ Technical note

With gamma-distributed or inverse-Gaussian–distributed frailty, hazard ratios decay over time in favor of the *frailty effect*, and thus the displayed "Haz. ratio" in the above output is actually the hazard ratio only for $t = 0$. The degree of decay depends on θ. Should the estimated θ be close to zero, the hazard ratios regain their usual interpretation. The rate of decay and the limiting hazard ratio differ between the gamma and inverse-Gaussian models; see Gutierrez (2002) for details.

For this reason, many researchers prefer fitting frailty models in the AFT metric because the interpretation of regression coefficients is unchanged by the frailty—the factors in question serve to either accelerate or decelerate the survival experience. The only difference is that with frailty models, the unconditional probability of survival is described by $S_\theta(t)$ rather than $S(t)$.

❑

❑ Technical note

The LR test of $\theta = 0$ is a boundary test and thus requires careful consideration concerning the calculation of its p-value. In particular, the null distribution of the LR test statistic is not the usual χ^2_1 but rather is a 50:50 mixture of a χ^2_0 (point mass at zero) and a χ^2_1, denoted as $\overline{\chi}^2_{01}$. See Gutierrez, Carter, and Drukker (2001) for more details.

❑

To verify that the significant heterogeneity is caused by the omission of dietfat, we now refit the Weibull/inverse-Gaussian frailty model with dietfat included.

```
. streg age smoking dietfat, distribution(weibull) frailty(invgauss) nolog
        Failure _d: dead
  Analysis time _t: t

Weibull PH regression
Inverse-Gaussian frailty

No. of subjects =        80                          Number of obs =        80
No. of failures =        58
Time at risk    = 1,257.07
                                                     LR chi2(3)     = 246.41
Log likelihood = -13.352142                          Prob > chi2    = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.74928	.0985246	9.93	0.000	1.566453	1.953447
smoking	5.203552	1.704943	5.03	0.000	2.737814	9.889992
dietfat	9.229842	2.219331	9.24	0.000	5.761312	14.78656
_cons	1.07e-20	4.98e-20	-9.92	0.000	1.22e-24	9.45e-17
/ln_p	1.431742	.0978847	14.63	0.000	1.239892	1.623593
/lntheta	-14.29793	2673.364	-0.01	0.996	-5253.995	5225.399
p	4.185987	.4097439			3.45524	5.071278
1/p	.2388923	.0233839			.197189	.2894155
theta	6.17e-07	.0016502			0	.

```
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard.
LR test of theta=0: chibar2(01) = 0.00               Prob >= chibar2 = 1.000
```

The estimate of the frailty variance component θ is near zero, and the p-value of the test of H_0: $\theta = 0$ equals one, indicating negligible heterogeneity. A regular Weibull model could be fit to these data (with dietfat included), producing almost identical estimates of the hazard ratios and ancillary parameter, p, so such an analysis is omitted here.

Also hazard ratios now regain their original interpretation. Thus an increase in weekly calories from fat of 1,000 would increase the risk of death by more than ninefold.

◁

Shared-frailty models

A generalization of the frailty models considered in the previous section is the *shared-frailty* model, where the frailty is assumed to be group specific; this is analogous to a panel-data regression model. For observation j from the ith group, the hazard is

$$h_{ij}(t|\alpha_i) = \alpha_i h_{ij}(t)$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, n_i$, where by $h_{ij}(t)$ we mean $h(t|\mathbf{x}_{ij})$, which is the individual hazard given covariates $\mathbf{x}_{ij}$.

Shared-frailty models are appropriate when you wish to model the frailties as being specific to groups of subjects, such as subjects within families. Here a shared-frailty model may be used to model the degree of correlation within groups; that is, the subjects within a group are correlated because they share the same common frailty.

▷ Example 11

Consider the data from a study of 38 kidney dialysis patients, as described in McGilchrist and Aisbett (1991). The study is concerned with the prevalence of infection at the catheter-insertion point. Two recurrence times (in days) are measured for each patient, and each recorded time is the time from initial insertion (onset of risk) to infection or censoring.

```
. use https://www.stata-press.com/data/r18/catheter
(Kidney data, McGilchrist and Aisbett, Biometrics, 1991)

. list patient time infect age female in 1/10
```

	patient	time	infect	age	female
1.	1	16	1	28	0
2.	1	8	1	28	0
3.	2	13	0	48	1
4.	2	23	1	48	1
5.	3	22	1	32	0
6.	3	28	1	32	0
7.	4	318	1	31.5	1
8.	4	447	1	31.5	1
9.	5	30	1	10	0
10.	5	12	1	10	0

Each patient (`patient`) has two recurrence times (`time`) recorded, with each catheter insertion resulting in either infection (`infect==1`) or right-censoring (`infect==0`). Among the covariates measured are `age` and sex (`female==1` if female, `female==0` if male).

One subtlety to note concerns the use of the generic term *subjects*. In this example, the subjects are the individual catheter insertions, not the patients themselves. This is a function of how the data were recorded—the onset of risk occurs at catheter insertion (of which there are two for each patient) not, say, at the time of admission of the patient into the study. Thus we have two subjects (insertions) within each group (patient).

It is reasonable to assume independence of patients but unreasonable to assume that recurrence times within each patient are independent. One solution would be to fit a standard survival model, adjusting the standard errors of the parameter estimates to account for the possible correlation by specifying vce(cluster patient).

We could also model the correlation by assuming that the correlation is the result of a latent patient-level effect, or frailty. That is, rather than fitting a standard model and specifying vce(cluster patient), we fit a frailty model and specify shared(patient). Assuming that the time to infection, given age and female, follows a Weibull distribution, and inverse-Gaussian distributed frailties, we get

```
. stset time, fail(infect)
 (output omitted )

. streg age female, distribution(weibull) frailty(invgauss) shared(patient) nolog

       Failure _d: infect
  Analysis time _t: time

Weibull PH regression
Inverse-Gaussian shared frailty                    Number of obs     =       76
Group variable: patient                            Number of groups  =       38
                                                   Obs per group:
No. of subjects =     76                                        min =        2
No. of failures =     58                                        avg =        2
Time at risk    = 7,424                                         max =        2
                                                   LR chi2(2)        =     9.84
Log likelihood = -99.093527                        Prob > chi2       =   0.0073
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.006918	.013574	0.51	0.609	.9806623	1.033878
female	.2331376	.1046382	-3.24	0.001	.0967322	.5618928
_cons	.0110089	.0099266	-5.00	0.000	.0018803	.0644557
/ln_p	.1900625	.1315342	1.44	0.148	-.0677398	.4478649
/lntheta	.0357272	.7745362	0.05	0.963	-1.482336	1.55379
p	1.209325	.1590676			.9345036	1.564967
1/p	.8269074	.1087666			.638991	1.070087
theta	1.036373	.8027085			.2271066	4.729362

```
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard.
LR test of theta=0: chibar2(01) = 8.70              Prob >= chibar2 = 0.002
```

Contrast this with what we obtain by assuming a subject-level lognormal model:

```
. streg age female, distribution(lnormal) frailty(invgauss) shared(patient) nolog

        Failure _d: infect
  Analysis time _t: time

Lognormal AFT regression                         Number of obs     =       76
Inverse-Gaussian shared frailty                  Number of groups  =       38
Group variable: patient
                                                 Obs per group:
No. of subjects =     76                                     min =        2
No. of failures =     58                                     avg =        2
Time at risk    = 7,424                                      max =        2
                                                 LR chi2(2)        =    16.34
Log likelihood = -97.614583                      Prob > chi2       =   0.0003
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
age	-.0066762	.0099457	-0.67	0.502	-.0261694	.0128171
female	1.401719	.3334931	4.20	0.000	.7480844	2.055354
_cons	3.336709	.4972641	6.71	0.000	2.362089	4.311329
/lnsigma	.0625872	.1256185	0.50	0.618	-.1836205	.3087949
/lntheta	-1.606248	1.190775	-1.35	0.177	-3.940125	.7276282
sigma	1.064587	.1337318			.8322516	1.361783
theta	.2006389	.2389159			.0194458	2.070165

```
LR test of theta=0: chibar2(01) = 1.53            Prob >= chibar2 = 0.108
```

The frailty effect is insignificant at the 10% level in the latter model yet highly significant in the former. We thus have two possible stories to tell concerning these data: If we believe the first model, we believe that the individual hazard of infection continually rises over time (Weibull), but there is a significant frailty effect causing the population hazard to begin falling after some time. If we believe the second model, we believe that the individual hazard first rises and then declines (lognormal), meaning that if a given insertion does not become infected initially, the chances that it will become infected begin to decrease after a certain point. Because the frailty effect is insignificant, the population hazard mirrors the individual hazard in the second model.

As a result, both models view the population hazard as rising initially and then falling past a certain point. The second version of our story corresponds to higher log likelihood, yet perhaps not significantly higher given the limited data. More investigation is required. One idea is to fit a more distribution-agnostic form of a frailty model, such as a piecewise exponential (Cleves, Gould, and Marchenko 2016, 345–348) or a Cox model with frailty; see [ST] **stcox**.

◁

Shared-frailty models are also appropriate when the frailties are subject specific yet there exist multiple records per subject. Here you would share frailties across the same id() variable previously stset. When you have subject-specific frailties and uninformative episode splitting, it makes no difference whether you fit a shared or an unshared frailty model. The estimation results will be the same.

Stored results

streg stores the following in e():

Scalars

e(N)	number of observations
e(N_sub)	number of subjects
e(N_fail)	number of failures
e(N_g)	number of groups
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_aux)	number of auxiliary parameters
e(k_dv)	number of dependent variables
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(ll_0)	log likelihood, constant-only model
e(ll_c)	log likelihood, comparison model
e(N_clust)	number of clusters
e(chi2)	χ^2
e(chi2_c)	χ^2, comparison model
e(risk)	total time at risk
e(g_min)	smallest group size
e(g_avg)	average group size
e(g_max)	largest group size
e(theta)	frailty parameter
e(aux_p)	ancillary parameter (weibull)
e(gamma)	ancillary parameter (gompertz, loglogistic)
e(sigma)	ancillary parameter (ggamma, lnormal)
e(kappa)	ancillary parameter (ggamma)
e(p)	p-value for model test
e(p_c)	p-value for comparison test
e(rank)	rank of e(V)
e(rank0)	rank of e(V), constant-only model
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	model or regression name
e(cmd2)	streg
e(cmdline)	command as typed
e(dead)	_d
e(depvar)	_t
e(strata)	stratum variable
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(shared)	frailty grouping variable
e(fr_title)	title in output identifying frailty
e(wtype)	weight type
e(wexp)	weight expression
e(t0)	_t0
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(frm2)	hazard or time
e(chi2type)	Wald or LR; type of model χ^2 test
e(offset1)	offset for main equation
e(stcurve)	stcurve
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(properties)	b V

e(predict)	program used to implement predict
e(predict_sub)	predict subprogram
e(footnote)	program used to implement the footnote display
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

For an introduction to survival models, see Cleves, Gould, and Marchenko (2016). For an introduction to survival analysis directed at social scientists, see Box-Steffensmeier and Jones (2004).

Consider for $j = 1, \ldots, n$ observations the trivariate response, (t_{0j}, t_j, d_j), representing a period of observation, $(t_{0j}, t_j]$, ending in either failure ($d_j = 1$) or right-censoring ($d_j = 0$). This structure allows analysis of a wide variety of models and may be used to account for delayed entry, gaps, time-varying covariates, and multiple failures per subject. Regardless of the structure of the data, once they are stset, the data may be treated in a common manner by streg: the stset-created variable _t0 holds the t_{0j}, _t holds the t_j, and _d holds the d_j.

For a given survivor function, $S(t)$, the density function is obtained as

$$f(t) = -\frac{d}{dt}S(t)$$

and the hazard function (the instantaneous rate of failure) is obtained as $h(t) = f(t)/S(t)$. Available forms for $S(t)$ are listed in table 1. For a set of covariates from the jth observation, $\mathbf{x}_j$, define $S_j(t) = S(t|\mathbf{x} = \mathbf{x}_j)$, and similarly define $h_j(t)$ and $f_j(t)$. For example, in a Weibull PH model, $S_j(t) = \exp\{-\exp(\mathbf{x}_j\boldsymbol{\beta})t^p\}$.

Parameter estimation

In this command, $\boldsymbol{\beta}$ and the ancillary parameters are estimated via maximum likelihood. A subject known to fail at time t_j contributes to the likelihood function the value of the density at time t_j conditional on the entry time t_{0j}, $f_j(t_j)/S_j(t_{0j})$. A censored observation, known to survive only up to time t_j, contributes $S_j(t_j)/S_j(t_{0j})$, which is the probability of surviving beyond time t_j conditional on the entry time, t_{0j}. The log likelihood is thus given by

$$\log L = \sum_{j=1}^{n} \{d_j \log f_j(t_j) + (1 - d_j)\log S_j(t_j) - \log S_j(t_{0j})\}$$

Implicit in the above log-likelihood expression are the regression parameters, $\boldsymbol{\beta}$, and the ancillary parameters because both are components of the chosen $S_j(t)$ and its corresponding $f_j(t)$; see table 1. streg reports maximum likelihood estimates of $\boldsymbol{\beta}$ and of the ancillary parameters (if any for the chosen model). The reported log-likelihood value is $\log L_r = \log L + T$, where $T = \sum \log(t_j)$ is summed over uncensored observations. The adjustment removes the time units from $\log L$. Whether the adjustment is made makes no difference to any test or result since such tests and results depend on differences in log-likelihood functions or their second derivatives, or both.

Specifying `ancillary()`, `anc2()`, or `strata()` will parameterize the ancillary parameter(s) by using the linear predictor, $\mathbf{z}_j \boldsymbol{\alpha}_z$, where the covariates, $\mathbf{z}_j$, need not be distinct from $\mathbf{x}_j$. Here streg will report estimates of $\boldsymbol{\alpha}_z$ in addition to estimates of $\boldsymbol{\beta}$. The log likelihood here is simply the log likelihood given above, with $\mathbf{z}_j \boldsymbol{\alpha}_z$ substituted for the ancillary parameter. If the ancillary parameter is constrained to be strictly positive, its logarithm is parameterized instead; that is, we substitute the linear predictor for the logarithm of the ancillary parameter in the above log likelihood. The gamma model has two ancillary parameters, σ and κ; we parameterize σ by using `ancillary()` and κ by using `anc2()`, and the linear predictors used for each may be distinct. Specifying `strata()` includes factor levels for the strata in the main equation and uses the factor levels to parameterize any ancillary parameters that exist for the chosen model.

Unshared-frailty models have a log likelihood of the above form, with $S_\theta(t)$ and $f_\theta(t)$ substituted for $S(t)$ and $f(t)$, respectively. Equivalently, for gamma-distributed frailties,

$$\log L = \sum_{j=1}^{n} \left[\theta^{-1} \log \left\{ 1 - \theta \log S_j(t_{0j}) \right\} - \left(\theta^{-1} + d_j \right) \log \left\{ 1 - \theta \log S_j(t_j) \right\} + d_j \log h_j(t_j) \right]$$

and for inverse-Gaussian–distributed frailties,

$$\log L = \sum_{j=1}^{n} \left[\theta^{-1} \left\{ 1 - 2\theta \log S_j(t_{0j}) \right\}^{1/2} - \theta^{-1} \left\{ 1 - 2\theta \log S_j(t_j) \right\}^{1/2} + \right.$$
$$\left. d_j \log h_j(t_j) - \frac{1}{2} d_j \log \left\{ 1 - 2\theta \log S_j(t_j) \right\} \right]$$

In a shared-frailty model, the frailty is common to a group of observations. Thus, to form an unconditional likelihood, the frailties must be integrated out at the group level. The data are organized as $i = 1, \ldots, n$ groups with the ith group comprising $j = 1, \ldots, n_i$ observations. The log likelihood is the sum of the log-likelihood contributions for each group. Define $D_i = \sum_j d_{ij}$ as the number of failures in the ith group. For gamma frailties, the log-likelihood contribution for the ith group is

$$\log L_i = \sum_{j=1}^{n_i} d_{ij} \log h_{ij}(t_{ij}) - (1/\theta + D_i) \log \left\{ 1 - \theta \sum_{j=1}^{n_i} \log \frac{S_{ij}(t_{ij})}{S_{ij}(t_{0ij})} \right\} +$$
$$D_i \log \theta + \log \Gamma(1/\theta + D_i) - \log \Gamma(1/\theta)$$

This formula corresponds to the log-likelihood contribution for multiple-record data. For single-record data, the denominator $S_{ij}(t_{0ij})$ is equal to 1. This formula is not applicable to data with delayed entries or gaps.

For inverse-Gaussian frailties, define

$$C_i = \left\{ 1 - 2\theta \sum_{j=1}^{n_i} \log \frac{S_{ij}(t_{ij})}{S_{ij}(t_{0ij})} \right\}^{-1/2}$$

The log-likelihood contribution for the ith group then becomes

$$\log L_i = \theta^{-1}(1 - C_i^{-1}) + B(\theta C_i, D_i) + \sum_{j=1}^{n_i} d_{ij} \{ \log h_{ij}(t_{ij}) + \log C_i \}$$

The function $B(a, b)$ is related to the modified Bessel function of the third kind, commonly known as the BesselK function; see Wolfram (2003, 775–776). In particular,

$$B(a, b) = a^{-1} + \frac{1}{2} \left\{ \log \left(\frac{2}{\pi} \right) - \log a \right\} + \log \text{BesselK} \left(\frac{1}{2} - b, a^{-1} \right)$$

For both unshared- and shared-frailty models, estimation of θ takes place jointly with the estimation of $\boldsymbol{\beta}$ and the ancillary parameters.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using vce(robust) and vce(cluster *clustvar*), respectively. See [P] _robust, particularly *Maximum likelihood estimators* and *Methods and formulas*. If observations in the dataset represent repeated observations on the same subjects (that is, there are time-varying covariates), the assumption of independence of the observations is highly questionable, meaning that the conventional estimate of variance is not appropriate. We strongly advise that you use the vce(robust) and vce(cluster *clustvar*) options here. (streg knows to specify vce(cluster *clustvar*) if you specify vce(robust).) vce(robust) and vce(cluster *clustvar*) do not apply in shared-frailty models, where the correlation within groups is instead modeled directly.

streg also supports estimation with survey data. For details on VCEs with survey data, see [SVY] **Variance estimation**.

Benjamin Gompertz (1779–1865) came from a Jewish family who left Holland and settled in England. Excluded from a university education, he was self-educated in mathematics. In 1819, his publications in mathematics earned him an invitation to join the Royal Society. In 1824, he was appointed as actuary and head clerk of the Alliance Assurance Company.

Gompertz carried out pioneering work on the application of differential calculus to actuarial questions, particularly the dependence of mortality on age. He is credited with introducing, in 1825, the concept that mortality is a continuous function over time. From this idea came the notion of a survival function, and ultimately, parametric survival-time analysis. Gompertz's work also had a strong influence on the practice of demography, where it is used in the study of parity and fertility.

Aside from his work in actuarial sciences, Gompertz contributed to astronomy and the study of astronomical instruments. He was a member of the Astronomical Society nearly from its founding in 1820. The society's memoirs recognize him as an important contributor to the study of the aberration of light. He also helped to develop the society's catalog of the stars and make improvements to its instruments, including the convertible pendulum, transit instruments for studying the position of stars, and the differential sextant, his own invention.

Ernst Hjalmar Waloddi Weibull (1887–1979) was a Swedish applied physicist most famous for his work on the statistics of material properties. He worked in Germany and Sweden as an inventor and a consulting engineer, publishing his first paper on the propagation of explosive waves in 1914, thereafter becoming a full professor at the Royal Institute of Technology in 1924. Weibull wrote two important papers, "Investigations into strength properties of brittle materials" and "The phenomenon of rupture in solids", which discussed his ideas about the statistical distributions of material strength. These articles came to the attention of engineers in the late 1930s.

References

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19: 716–723. https://doi.org/10.1109/TAC.1974.1100705.

Bottai, M., and N. Orsini. 2013. A command for Laplace regression. *Stata Journal* 13: 302–314.

Bower, H., M. J. Crowther, and P. C. Lambert. 2016. strcs: A command for fitting flexible parametric survival models on the log-hazard scale. *Stata Journal* 16: 989–1012.

Box-Steffensmeier, J. M., and B. S. Jones. 2004. *Event History Modeling: A Guide for Social Scientists*. Cambridge: Cambridge University Press.

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata Press.

Cox, D. R., and D. Oakes. 1984. *Analysis of Survival Data*. London: Chapman and Hall/CRC.

Crowder, M. J., A. C. Kimber, R. L. Smith, and T. J. Sweeting. 1991. *Statistical Analysis of Reliability Data*. London: Chapman and Hall/CRC.

Crowther, M. J., K. R. Abrams, and P. C. Lambert. 2013. Joint modeling of longitudinal and survival data. *Stata Journal* 13: 165–184.

Cui, J. 2005. Buckley–James method for analyzing censored data, with an application to a cardiovascular disease and an HIV/AIDS study. *Stata Journal* 5: 517–526.

Fisher, R. A., and L. H. C. Tippett. 1928. Limiting forms of the frequency distribution of the largest or smallest member of a sample. *Mathematical Proceedings of the Cambridge Philosophical Society* 24: 180–190. https://doi.org/10.1017/S0305004100015681.

Gutierrez, R. G. 2002. Parametric frailty and shared frailty survival models. *Stata Journal* 2: 22–44.

Gutierrez, R. G., S. L. Carter, and D. M. Drukker. 2001. sg160: On boundary-value likelihood-ratio tests. *Stata Technical Bulletin* 60: 15–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 269–273. College Station, TX: Stata Press.

Hooker, P. F. 1965. Benjamin Gompertz. *Journal of the Institute of Actuaries* 91: 203–212. https://doi.org/10.1017/S0020268100038786.

Hosmer, D. W., Jr., S. A. Lemeshow, and S. May. 2008. *Applied Survival Analysis: Regression Modeling of Time to Event Data*. 2nd ed. New York: Wiley.

Hougaard, P. 1986. Survival models for heterogeneous populations derived from stable distributions. *Biometrika* 73: 387–396. https://doi.org/10.2307/2336215.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd ed. New York: Springer.

Lambert, P. C., and P. Royston. 2009. Further development of flexible parametric models for survival analysis. *Stata Journal* 9: 265–290.

Lee, E. T., and J. W. Wang. 2013. *Statistical Methods for Survival Data Analysis*. 4th ed. New York: Wiley.

McGilchrist, C. A., and C. W. Aisbett. 1991. Regression with frailty in survival analysis. *Biometrics* 47: 461–466. https://doi.org/10.2307/2532138.

Olshansky, S. J., and B. A. Carnes. 1997. Ever since Gompertz. *Demography* 34: 1–15. https://doi.org/10.2307/2061656.

Peto, R., and P. Lee. 1973. Weibull distributions for continuous-carcinogenesis experiments. *Biometrics* 29: 457–470. https://doi.org/10.2307/2529169.

Pike, M. C. 1966. A method of analysis of a certain class of experiments in carcinogenesis. *Biometrics* 22: 142–161. https://doi.org/10.2307/2528221.

Royston, P. 2006. Explained variation for survival models. *Stata Journal* 6: 83–96.

Royston, P., and P. C. Lambert. 2011. *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model*. College Station, TX: Stata Press.

Royston, P., and M. K. B. Parmar. 2002. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in Medicine* 21: 2175–2197. https://doi.org/10.1002/sim.1203.

Schoenfeld, D. A. 1982. Partial residuals for the proportional hazards regression model. *Biometrika* 69: 239–241. https://doi.org/10.2307/2335876.

Weibull, W. 1939. A statistical theory of the strength of materials. In *Ingeniörs Vetenskaps Akademien Handlingar*, vol. 151. Stockholm: Generalstabens Litografiska Anstalts Förlag.

Wolfram, S. 2003. *The Mathematica Book*. 5th ed. Champaign, IL: Wolfram Media.

Also see

[ST] **streg postestimation** — Postestimation tools for streg

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[ST] **stcox** — Cox proportional hazards model

[ST] **stcrreg** — Competing-risks regression

[ST] **stintreg** — Parametric models for interval-censored survival-time data

[ST] **stintcox** — Cox proportional hazards model for interval-censored survival-time data

[ST] **sts** — Generate, graph, list, and test the survivor and related functions

[ST] **stset** — Declare data to be survival-time data

[BAYES] **bayes: streg** — Bayesian parametric survival models

[CAUSAL] **stteffects** — Treatment-effects estimation for observational survival-time data

[FMM] **fmm: streg** — Finite mixtures of parametric survival models

[ME] **mestreg** — Multilevel mixed-effects parametric survival models

[MI] **Estimation** — Estimation commands for use with mi estimate

[PSS-2] **power exponential** — Power analysis for a two-sample exponential test

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtstreg** — Random-effects parametric survival models

[U] **20 Estimation and postestimation commands**

Title

> **streg postestimation** — Postestimation tools for streg

Postestimation commands

The following postestimation command is of special interest after `streg`:

Command	Description
*`estat gofplot`	produce goodness-of-fit plot
`stcurve`	plot the survivor, failure, hazard, or cumulative hazard function

*`estat gofplot` is not appropriate with `svy` estimation results.

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of estimates
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
*`hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of coefficients
`linktest`	link test for model specification
*`lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
`predict`	hazard ratios, survivor functions, influence statistics, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of estimates
`suest`	seemingly unrelated estimation
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

predict creates a new variable containing predictions such as median and mean survival times; hazards; hazard ratios; linear predictions; standard errors; probabilities; Cox–Snell, martingale-like, and deviance residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

> predict [*type*] *newvar* [*if*] [*in*] [, *statistic options*]

> predict [*type*] *stub** [*if*] [*in*], <u>sc</u>ores

statistic	Description
Main	
<u>median</u> time	median survival time; the default
<u>median</u> lntime	median ln(survival time)
mean time	mean survival time
mean <u>ln</u>time	mean ln(survival time)
hazard	hazard
hr	hazard ratio, also known as the relative hazard
xb	linear prediction $\mathbf{x}_j\widehat{\boldsymbol{\beta}}$
stdp	standard error of the linear prediction; $\text{SE}(\mathbf{x}_j\widehat{\boldsymbol{\beta}})$
<u>s</u>urv	$S(t|t_0)$
* <u>csu</u>rv	$S(t \mid \text{earliest } t_0 \text{ for subject})$
* <u>csn</u>ell	Cox–Snell residuals
* <u>mg</u>ale	martingale-like residuals
* <u>dev</u>iance	deviance residuals

options	Description
oos	make *statistic* available in and out of sample
nooffset	ignore the offset() variable specified in streg
alpha1	predict *statistic* conditional on frailty value equal to one
<u>uncond</u>itional	predict *statistic* unconditionally on the frailty
<u>marg</u>inal	synonym for unconditional
<u>part</u>ial	produce observation-level results

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ... if` wanted only for the estimation sample. Starred statistics are calculated for the estimation sample by default, but the `oos` option makes them available both in and out of sample.

When no option is specified, the predicted median survival time is calculated for all models. The predicted hazard ratio, option `hr`, is available only for the exponential, Weibull, and Gompertz models. The `mean time` and `mean lntime` options are not available for the Gompertz model. Unconditional estimates of `mean time` and `mean lntime` are not available if `frailty()` was specified with `streg`; see [ST] **streg**.

`csnell`, `mgale`, and `deviance` are not allowed with `svy` estimation results.

Options for predict

⌐ Main ⌐

`median time` calculates the predicted median survival time in analysis-time units. This is the prediction from time 0 conditional on constant covariates. When no options are specified with `predict`, the predicted median survival time is calculated for all models.

`median lntime` calculates the natural logarithm of what `median time` produces.

`mean time` calculates the predicted mean survival time in analysis-time units. This is the prediction from time 0 conditional on constant covariates. This option is not available for Gompertz regressions and is available for frailty models only if `alpha1` is specified, in which case what you obtain is an estimate of the mean survival time conditional on a frailty effect of one.

`mean lntime` predicts the mean of the natural logarithm of `time`. This option is not available for Gompertz regression and is available for frailty models only if `alpha1` is specified, in which case what you obtain is an estimate of the mean log survival-time conditional on a frailty effect of one.

`hazard` calculates the predicted hazard.

`hr` calculates the hazard ratio. This option is valid only for models having a proportional-hazards parameterization.

`xb` calculates the linear prediction from the fitted model. That is, you fit the model by estimating a set of parameters, $\beta_0, \beta_1, \beta_2, \ldots, \beta_k$, and the linear prediction is $\widehat{y}_j = \widehat{\beta}_0 + \widehat{\beta}_1 x_{1j} + \widehat{\beta}_2 x_{2j} + \cdots + \widehat{\beta}_k x_{kj}$, often written in matrix notation as $\widehat{y}_j = \mathbf{x}_j \widehat{\boldsymbol{\beta}}$.

The x_{1j}, x_{2j}, ..., x_{kj} used in the calculation are obtained from the data currently in memory and need not correspond to the data on the independent variables used in estimating $\boldsymbol{\beta}$.

`stdp` calculates the standard error of the prediction, that is, the standard error of $\widehat{y}_j$.

`surv` calculates each observation's predicted survivor probability, $S(t|t_0)$, where t_0 is `_t0`, the analysis time at which each record became at risk. For multiple-record data, see the `csurv` option below.

`csurv` calculates the predicted $S(t|\text{earliest } t_0)$ for each subject in multiple-record data by calculating the conditional survivor values, $S(t|t_0)$ (see the `surv` option above), and then multiplying them.

What you obtain from `surv` will differ from what you obtain from `csurv` only if you have multiple records for that subject.

In the presence of gaps or delayed entry, the estimates obtained from `csurv` can be different for subjects with gaps from those without gaps, having the same covariate values, because the probability of survival over gaps is assumed to be 1. Thus the predicted cumulative conditional survivor function is not a smooth function of time `_t` for constant values of the covariates. Use `stcurve, survival` to obtain a smooth estimate of the cumulative survivor function $S(t|x)$.

`csnell` calculates the Cox–Snell generalized residuals. For multiple-record-per-subject data, by default only one value per subject is calculated and it is placed on the last record for the subject.

Adding the `partial` option will produce partial Cox–Snell residuals, one for each record within subject; see `partial` below. Partial Cox–Snell residuals are the additive contributions to a subject's overall Cox–Snell residual. In single-record-per-subject data, the partial Cox–Snell residuals are the Cox–Snell residuals.

`mgale` calculates the martingale-like residuals. For multiple-record data, by default only one value per subject is calculated and it is placed on the last record for the subject.

Adding the `partial` option will produce partial martingale residuals, one for each record within subject; see `partial` below. Partial martingale residuals are the additive contributions to a subject's overall martingale residual. In single-record data, the partial martingale residuals are the martingale residuals.

`deviance` calculates the deviance residuals. Deviance residuals are martingale residuals that have been transformed to be more symmetric about zero. For multiple-record data, by default only one value per subject is calculated and it is placed on the last record for the subject.

Adding the `partial` option will produce partial deviance residuals, one for each record within subject; see `partial` below. Partial deviance residuals are transformed partial martingale residuals. In single-record data, the partial deviance residuals are the deviance residuals.

`oos` makes `csurv`, `csnell`, `mgale`, and `deviance` available both in and out of sample. `oos` also dictates that summations and other accumulations take place over the sample as defined by `if` and `in`. By default, the summations are taken over the estimation sample, with `if` and `in` merely determining which values of *newvar* are to be filled in once the calculation is finished.

`nooffset` is relevant only if you specified `offset(`*varname*`)` with `streg`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as $\mathbf{x}\boldsymbol{\beta}$ rather than $\mathbf{x}\boldsymbol{\beta} + \text{offset}$.

`alpha1`, when used after fitting a frailty model, specifies that *statistic* be predicted conditional on a frailty value equal to one. This is the default for shared-frailty models.

`unconditional` and `marginal`, when used after fitting a frailty model, specify that *statistic* be predicted unconditional on the frailty. That is, the prediction is averaged over the frailty distribution. This is the default for unshared-frailty models.

`partial` is relevant only for multiple-record data and is valid with `csnell`, `mgale`, and `deviance`. Specifying `partial` will produce "partial" versions of these statistics, where one value is calculated for each record instead of one for each subject. The subjects are determined by the `id()` option of `stset`.

Specify `partial` if you wish to perform diagnostics on individual records rather than on individual subjects. For example, a partial deviance can be used to diagnose the fitted characteristics of an individual record rather than those of the set of records for a given subject.

`scores` calculates equation-level score variables. The number of score variables created depends upon the chosen distribution.

The first new variable will always contain $\partial \ln L / \partial(\mathbf{x}_j \boldsymbol{\beta})$.

The subsequent new variables will contain the partial derivative of the log likelihood with respect to the ancillary parameters.

margins

Description for margins

margins estimates margins of response for median and mean survival times, hazard ratios, and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

margins [*marginlist*] [, *options*]

margins [*marginlist*] , <u>predict</u>(*statistic ...*) [<u>predict</u>(*statistic ...*) ...] [*options*]

statistic	Description
<u>median</u> time	median survival time; the default
<u>median</u> <u>ln</u>time	median ln(survival time)
mean time	mean survival time
mean <u>ln</u>time	mean ln(survival time)
hr	hazard ratio, also known as the relative hazard
xb	linear prediction $\mathbf{x}_j \widehat{\boldsymbol{\beta}}$
<u>hazard</u>	not allowed with margins
stdp	not allowed with margins
<u>surv</u>	not allowed with margins
<u>csurv</u>	not allowed with margins
<u>csnell</u>	not allowed with margins
mgale	not allowed with margins
deviance	not allowed with margins

Statistics not allowed with margins are functions of stochastic quantities other than e(b).

For the full syntax, see [R] **margins**.

Remarks and examples

predict after streg is used to generate a variable containing predicted values or residuals.

For a more detailed discussion on residuals, read *Residuals and diagnostic measures* in the [ST] **stcox postestimation** entry. Many of the concepts and ideas presented there also apply to streg models.

Regardless of the metric used, predict can generate predicted median survival times and median log survival-times for all models, and predicted mean times and mean log survival-times where available. Predicted survival, hazard, and residuals are also available for all models. The predicted hazard ratio can be calculated only for models with a proportional-hazards parameterization, that is, the Weibull, exponential, and Gompertz models. However, the estimation need not take place in the log-hazard metric. You can perform, for example, a Weibull regression specifying the time option and then ask that hazard ratios be predicted.

After fitting a frailty model, you can use `predict` with the `alpha1` option to generate predicted values based on $S(t)$ or use the `unconditional` option to generate predictions based on $S_\theta(t)$; see [ST] **streg**.

▷ Example 1

Let's return to example 1 of [ST] **streg** concerning the ability of emergency generators with new-style bearings to withstand overloads. Assume that, as before, we fit a proportional hazards Weibull model:

```
. use https://www.stata-press.com/data/r18/kva
(Generator experiment)

. stset failtime
 (output omitted )

. streg load bearings, distribution(weibull) nolog

        Failure _d: 1 (meaning all fail)
  Analysis time _t: failtime

Weibull PH regression

No. of subjects =  12                                   Number of obs =      12
No. of failures =  12
Time at risk    = 896
                                                        LR chi2(2)    =   30.27
Log likelihood = 5.6934189                              Prob > chi2   = 0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
load	1.599315	.1883807	3.99	0.000	1.269616	2.014631
bearings	.1887995	.1312109	-2.40	0.016	.0483546	.7371644
_cons	2.51e-20	2.66e-19	-4.26	0.000	2.35e-29	2.68e-11
/ln_p	2.051552	.2317074	8.85	0.000	1.597414	2.505691
p	7.779969	1.802677			4.940241	12.25202
1/p	.1285352	.0297826			.0816192	.2024193

Note: **_cons** estimates baseline hazard.

Now we can predict both the median survival time and the log-median survival time for each observation:

```
. predict time, time
(option median time assumed; predicted median time)

. predict lntime, lntime
(option median lntime assumed; predicted median log time)

. format time lntime %9.4f
```

```
. list failtime load bearings time lntime
```

	failtime	load	bearings	time	lntime
1.	100	15	0	127.5586	4.8486
2.	140	15	1	158.0407	5.0629
3.	97	20	0	94.3292	4.5468
4.	122	20	1	116.8707	4.7611
5.	84	25	0	69.7562	4.2450
6.	100	25	1	86.4255	4.4593
7.	54	30	0	51.5845	3.9432
8.	52	30	1	63.9114	4.1575
9.	40	35	0	38.1466	3.6414
10.	55	35	1	47.2623	3.8557
11.	22	40	0	28.2093	3.3397
12.	30	40	1	34.9504	3.5539

◁

▷ Example 2

Using the cancer data of example 6 in [ST] **streg**, again with drug remapped into a drug-treatment indicator, we can examine the various residuals that Stata produces. For a more detailed discussion on residuals, read *Residuals and diagnostic measures* in [ST] **stcox postestimation**. Many of the concepts and ideas presented there also apply to streg models. For a more technical presentation of these residuals, see *Methods and formulas*.

We will begin by evaluating goodness of fit based on the generalized Cox–Snell residuals. If the model fits the data, these residuals should have a standard exponential distribution with $\lambda = 1$. One way to verify the fit is to calculate an empirical estimate of the cumulative hazard function—based, for example, on the Kaplan–Meier survival estimates or the Aalen–Nelson estimator, taking the Cox–Snell residuals as the time variable and the censoring variable as before—and plot it against the Cox–Snell residuals. If the model fits the data, the plot should be a straight line with a slope of 1.

After fitting the model, we can use the estat gofplot command to create this graph. Here the km option specifies that the cumulative hazard function be based on the minus log of the Kaplan–Meier estimator. By default, the cumulative hazard function is based on the Nelson–Aalen estimator.

```
. use https://www.stata-press.com/data/r18/cancer, clear
(Patient survival in drug trial)
. replace drug = drug==2 | drug==3                    // 0, placebo : 1, nonplacebo
(48 real changes made)
. quietly stset studytime, failure(died)
. quietly streg age drug, distribution(exp)
. estat gofplot, km
```

We did this separately for each of four distributions. Results are plotted in figure 1:
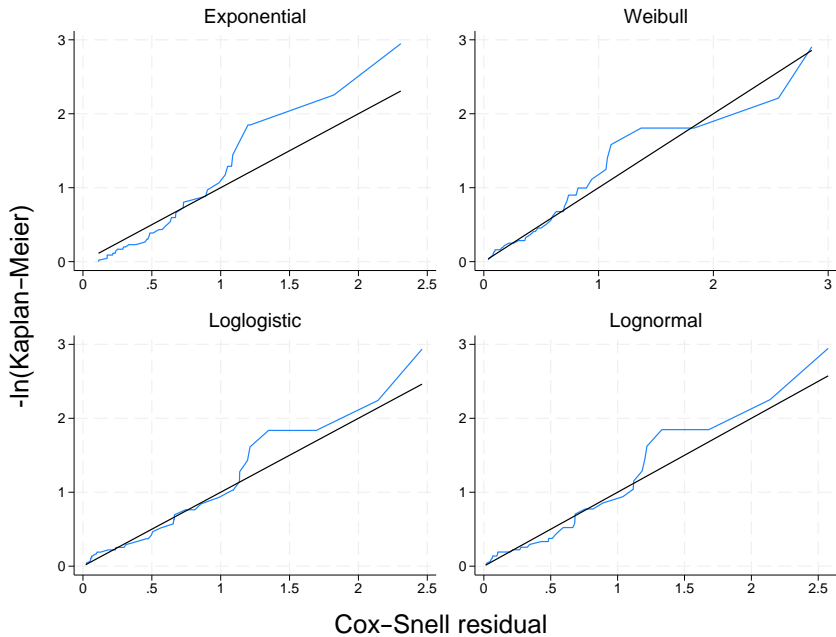
Figure 1. Cox–Snell residuals to evaluate model fit of four regression models

The plots indicate that the Weibull and lognormal models fit the data best and that the exponential model fits poorly. These results are consistent with our previous results (in [ST] **streg**) based on Akaike's information criterion.

◁

▷ Example 3

Let's now look at the martingale-like and deviance residuals. We use the term "martingale-like" because, although these residuals do not arise naturally from martingale theory for parametric survival models as they do for the Cox proportional hazards model, they do share similar form. We can generate these residuals by using predict's mgale option. Martingale residuals take values between $-\infty$ and 1 and therefore are difficult to interpret. The deviance residuals are a rescaling of the martingale-like residuals so that they are symmetric about zero and thus more like residuals obtained from linear regression. Plots of either deviance residuals against the linear predictor (that is, the log relative hazard in PH models) or of deviance residuals against individual predictors can be useful in identifying aberrant observations and in assessing model fit. Continuing with our modified cancer data, we plot the deviance residual obtained after fitting a lognormal model:

```
. quietly streg age drug, distribution(lnormal)
. predict dev, deviance
. scatter dev studytime, yline(0) m(o)
```
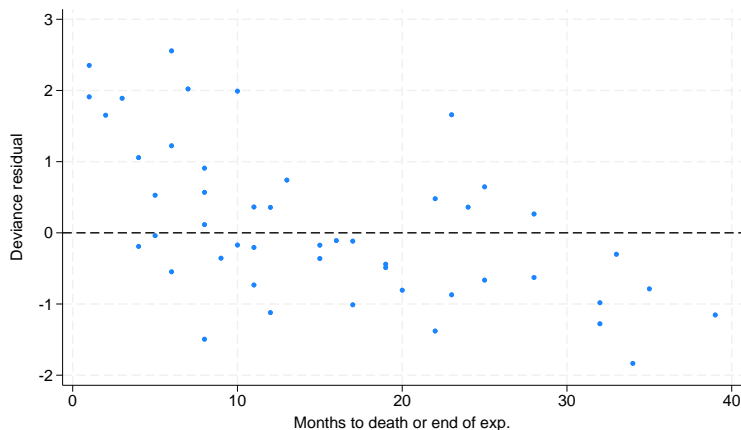

Figure 2. Deviance residuals to evaluate model fit of lognormal model

Figure 2 shows the deviance residuals to be relatively well behaved, with a few minor early exceptions.

◁

Methods and formulas

predict *newvar*, *options* may be used after streg to predict various quantities, according to the following *options*:

median time:
$$newvar_j = \{t : \widehat{S}_j(t) = 1/2\}$$

where $\widehat{S}_j(t)$ is $S_j(t)$ with the parameter estimates "plugged in".

median lntime:
$$newvar_j = \left\{y : \widehat{S}_j(e^y) = 1/2\right\}$$

mean time:
$$newvar_j = \int_0^\infty \widehat{S}_j(t)dt$$

mean lntime:

$$newvar_j = \int_{-\infty}^{\infty} y e^y \widehat{f}_j(e^y) dy$$

where $\widehat{f}_j(t)$ is $f_j(t)$ with the parameter estimates plugged in.

hazard:

$$newvar_j = \widehat{f}_j(t_j)/\widehat{S}_j(t_j)$$

hr (PH models only):

$$newvar_j = \exp(\mathbf{x}_j^* \widehat{\boldsymbol{\beta}^*})$$

where $\widehat{\boldsymbol{\beta}^*}$ does not contain the constant and $\mathbf{x}_j^*$ does not contain the coefficient of 1 corresponding to the constant.

xb:

$$newvar_j = \mathbf{x}_j \widehat{\boldsymbol{\beta}}$$

stdp:

$$newvar_j = \widehat{\mathrm{se}}(\mathbf{x}_j \widehat{\boldsymbol{\beta}})$$

surv and csurv:

$$newvar_j = \widehat{S}_j(t_j)/\widehat{S}_j(t_{0j})$$

The above represents the probability of survival past time t_j given survival up until t_{0j} and represents what you obtain when you specify surv. If csurv is specified, these probabilities are multiplied (in time order) over a subject's multiple observations. What is obtained is then equal to the probability of survival past time t_j, given survival through the earliest observed t_{0j}, and given the subject's (possibly time-varying) covariate history. In single-record-per-subject data, surv and csurv are identical.

csnell:

$$newvar_j = -\log \widehat{S}_j(t_j)$$

The Cox–Snell (1968) residual, CS_j, for observation j at time t_j is defined as $\widehat{H}_j(t_j) = -\log \widehat{S}_j(t_j)$, which is the estimated cumulative hazard function obtained from the fitted model (Collett 2003, 111–112). Cox and Snell argued that if the correct model has been fit to the data, these residuals are n observations from an exponential distribution with unit mean. Thus a plot of the cumulative hazard rate of the residuals against the residuals themselves should result in a straight line of slope 1. Cox–Snell residuals can never be negative and therefore are not symmetric about zero. The options csnell and partial store in each observation that observation's contribution to the subject's Cox–Snell residual, which we refer to as a partial Cox–Snell residual. If only csnell is specified, partial residuals are summed within each subject to obtain one overall Cox–Snell residual for that subject. If there is only 1 observation per subject, partial has no effect.

`mgale`:
$$newvar_j = d_j - \text{CS}_j$$

Martingale residuals follow naturally from martingale theory for Cox proportional hazards, but their development does not carry over for parametric survival models. However, martingale-like residuals similar to those obtained for Cox can be derived from the Cox–Snell residuals: $M_j = d_j - \text{CS}_j$, where CS_j are the Cox–Snell residuals, as previously described.

Because martingale-like residuals are calculated from the Cox–Snell residuals, they also could be partial or not. Partial martingale residuals are generated with the `mgale` and `partial` options, and overall martingale residuals are generated with the `mgale` option.

Martingale residuals can be interpreted as the difference over time between the number of deaths in the data and the expected number from the fitted model. These residuals take values between $-\infty$ and 1 and have an expected value of zero, although, like the Cox–Snell residuals, they are not symmetric about zero, making them difficult to interpret.

`deviance`:
$$newvar_j = \text{sign}(M_j)\left[-2\left\{M_j + d_j \log(d_j - M_j)\right\}\right]^{1/2}$$

Deviance residuals are a scaling of the martingale-like residuals in an attempt to make them symmetric about zero. When the model fits the data, these residuals are symmetric about zero and thus can be more readily used to examine the data for outliers. If you also specify the `partial` option, you obtain partial deviance residuals, one for each observation.

`predict` also allows two options for use after fitting frailty models: `alpha1` and `unconditional`. If `unconditional` is specified, the above predictions are modified to be based on $S_\theta(t)$ and $f_\theta(t)$, rather than $S(t)$ and $f(t)$; see [ST] **streg**. If `alpha1` is specified, the predictions are as described above.

References

Boswell, T. M., and R. G. Gutierrez. 2011. Stata tip 94: Manipulation of prediction parameters for parametric survival regression models. *Stata Journal* 11: 143–144.

Collett, D. 2003. *Modelling Binary Data*. 2nd ed. London: Chapman and Hall/CRC.

Cox, D. R., and E. J. Snell. 1968. A general definition of residuals (with discussion). *Journal of the Royal Statistical Society, Series B* 30: 248–275. https://doi.org/10.1111/j.2517-6161.1968.tb00724.x.

Also see

[ST] **streg** — Parametric survival models

[ST] **estat gofplot** — Goodness-of-fit plots after streg, stcox, stintreg, or stintcox

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[U] **20 Estimation and postestimation commands**

Title

sts — Generate, graph, list, and test the survivor and related functions

Description

sts graphs, reports, and creates variables containing the estimated survivor and related functions, such as the Nelson–Aalen cumulative hazard function. For the survivor function, sts tests and produces Kaplan–Meier estimates or, via Cox regression, covariate-adjusted estimates.

sts graph is equivalent to typing sts by itself—it graphs the survivor function. You can also graph the failure, hazard, or cumulative hazard function.

sts list lists the estimated survivor, failure, or cumulative hazard function.

sts test tests the equality of the survivor function across groups.

sts generate creates new variables containing the estimated survivor function, the Nelson–Aalen cumulative hazard function, and other related functions.

sts can be used with single- or multiple-record or single- or multiple-failure st data.

Syntax

sts $\big[\,\underline{\text{graph}}\,\big]$ $\big[\,if\,\big]$ $\big[\,in\,\big]$ $\big[\,,\;\ldots\big]$

sts $\underline{\text{list}}$ $\big[\,if\,\big]$ $\big[\,in\,\big]$ $\big[\,,\;\ldots\big]$

sts $\underline{\text{test}}$ *varlist* $\big[\,if\,\big]$ $\big[\,in\,\big]$ $\big[\,,\;\ldots\big]$

sts $\underline{\text{generate}}$ *newvar* = ... $\big[\,if\,\big]$ $\big[\,in\,\big]$ $\big[\,,\;\ldots\big]$

You must stset your data before using sts; see [ST] **stset**.
fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

See [ST] **sts graph**, [ST] **sts list**, [ST] **sts test**, and [ST] **sts generate** for details of syntax.

Remarks and examples

Remarks are presented under the following headings:

sts concerns the survivor function, $S(t)$; the probability of surviving to t or beyond; the cumulative hazard function, $H(t)$; and the hazard function, $h(t)$. Its subcommands can list and generate variables containing $\widehat{S}(t)$ and $\widehat{H}(t)$ and test the equality of $S(t)$ over groups. Also:

- All subcommands share a common syntax.

- All subcommands deal with either the Kaplan–Meier product-limit or the Nelson–Aalen estimates unless you request covariate-adjusted survival estimates.

- If you request to adjust for covariates, all subcommands perform the adjustment in the same way, which is described below.

The full details of each subcommand are found in the entries following this one, but each subcommand provides so many options to control exactly how the listing looks, how the graph appears, the form of the test to be performed, or what exactly is to be generated that the simplicity of sts can be easily overlooked.

So, without getting burdened by the details of syntax, let us demonstrate several simple usages of sts commands first.

▷ Example 1

```
. use https://www.stata-press.com/data/r18/drugtr
```

Graph the Kaplan–Meier survivor function	`. sts graph` `. sts graph, by(drug)`
Graph the Nelson–Aalen cumulative hazard function	`. sts graph, cumhaz` `. sts graph, cumhaz by(drug)`
Graph the estimated hazard function	`. sts graph, hazard` `. sts graph, hazard by(drug)`
List the Kaplan–Meier survivor function	`. sts list` `. sts list, by(drug) compare`
List the Nelson–Aalen cumulative hazard function	`. sts list, cumhaz` `. sts list, cumhaz by(drug) compare`
Generate variable containing the Kaplan–Meier survivor function	`. sts gen surv = s` `. sts gen surv_by_drug = s, by(drug)`
Generate variable containing the Nelson–Aalen cumulative hazard function	`. sts gen haz = na` `. sts gen haz_by_drug = na, by(drug)`

Test equality of survivor functions	`. sts test drug`
	`. gen agecat = autocode(age,4,47,67)`
	`. sts test drug, strata(agecat)`

◁

Listing, graphing, and generating variables

You can list the overall survivor function by typing sts list, and you can graph it by typing sts graph or sts. sts assumes that you mean graph when you do not type a subcommand.

Or, you can list the Nelson–Aalen cumulative hazard function by typing sts list, cumhaz, and you can graph it by typing sts graph, cumhaz.

Consider the Stanford heart transplant data introduced in [ST] **stset**. When you type sts list, you are shown all the details:

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)
. stset, noshow
. sts list
Kaplan–Meier survivor function
```

Time	At risk	Fail	Net lost	Survivor function	Std. error	[95% conf. int.]	
1	103	1	0	0.9903	0.0097	0.9331	0.9986
2	102	3	0	0.9612	0.0190	0.8998	0.9852
3	99	3	0	0.9320	0.0248	0.8627	0.9670
5	96	1	0	0.9223	0.0264	0.8507	0.9604
(output omitted)							
1586	2	0	1	0.1519	0.0493	0.0713	0.2606
1799	1	0	1	0.1519	0.0493	0.0713	0.2606

Note: Net lost equals the number lost minus the number who entered.

When you type sts graph or just sts, you are shown a graph of the same result detailed by list:

```
. sts graph
```


Kaplan–Meier survival estimate
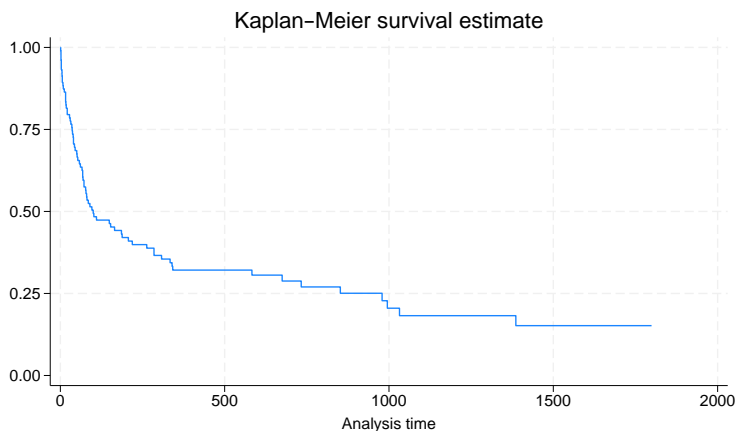
sts generate is a rarely used command. Typing sts generate survf = s creates a new variable, survf, containing the same survivor function that list just listed and graph just graphed:

```
. sts gen survf = s
. sort t1
. list t1 survf in 1/10
```

	t1	survf
1.	1	.99029126
2.	1	.99029126
3.	1	.99029126
4.	1	.99029126
5.	2	.96116505
6.	2	.96116505
7.	2	.96116505
8.	2	.96116505
9.	2	.96116505
10.	2	.96116505

sts generate is provided if you want to make a calculation, listing, or graph that sts cannot already do for you.

Comparing survivor or cumulative hazard functions

sts allows you to compare survivor or cumulative hazard functions. sts graph and sts graph, cumhaz are probably most successful at this. For example, survivor functions can be plotted using

```
. sts graph, by(posttran)
```
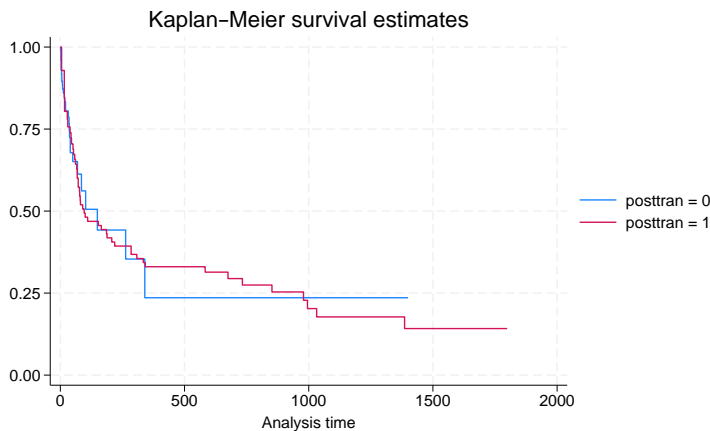
and Nelson–Aalen cumulative hazard functions can be plotted using

. sts graph, cumhaz by(posttran)

Nelson–Aalen cumulative hazard estimates
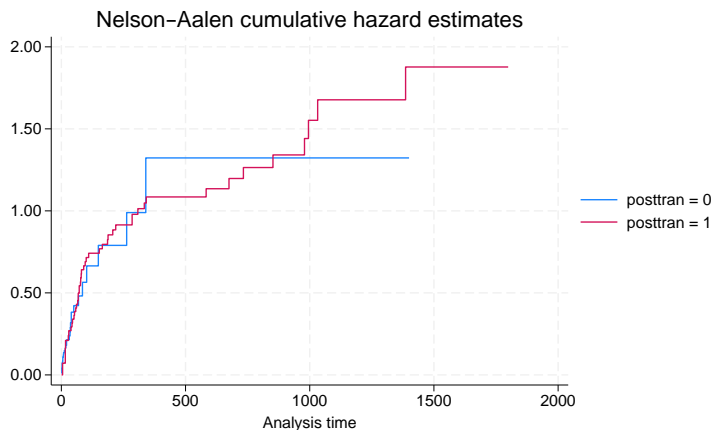
To compare survivor functions, we typed `sts graph`, just as before, and then we added `by(posttran)` to see the survivor functions for the groups designated by `posttran`. Here there are two groups, but as far as the `sts` command is concerned, there could have been more. `cumhaz` was also added to compare cumulative hazard functions.

You can also plot and compare estimated hazard functions by using `sts graph, hazard`. The hazard is estimated as a kernel smooth of the increments that sum to form the estimated cumulative hazard. The increments themselves do not estimate the hazard, but the smooth is weighted so that it estimates the hazard; see [ST] **sts graph**.

Just as you can compare survivor functions graphically by typing `sts graph, by(posttran)` and cumulative hazard functions by typing `sts graph, cumhaz by(posttran)`, you can obtain detailed listings by typing `sts list, by(posttran)` and `sts list, cumhaz by(posttran)`, respectively. Below we list the survivor function and specify `enter`, which adds a number-who-enter column:

```
. sts list, by(posttran) enter
Kaplan-Meier survivor function
By variable: posttran
```

Time	At risk	Fail	Lost	Enter	Survivor function	Std. error	[95% conf. int.]	
posttran=0								
0	0	0	0	103	1.0000	.	.	.
1	103	1	3	0	0.9903	0.0097	0.9331	0.9986
2	99	3	3	0	0.9603	0.0195	0.8976	0.9849
(output omitted)								
427	2	0	1	0	0.2359	0.1217	0.0545	0.4882
1400	1	0	1	0	0.2359	0.1217	0.0545	0.4882
posttran=1								
1	0	0	0	3	1.0000	.	.	.
2	3	0	0	3	1.0000	.	.	.
3	6	0	0	3	1.0000	.	.	.
4	9	0	0	2	1.0000	.	.	.
5	11	0	0	3	1.0000	.	.	.
5.1	14	1	0	0	0.9286	0.0688	0.5908	0.9896
6	13	0	0	1	0.9286	0.0688	0.5908	0.9896
8	14	0	0	2	0.9286	0.0688	0.5908	0.9896
10	16	0	0	2	0.9286	0.0688	0.5908	0.9896
(output omitted)								
1586	2	0	1	0	0.1420	0.0546	0.0566	0.2653
1799	1	0	1	0	0.1420	0.0546	0.0566	0.2653

sts list's compare option allows you to compare survivor or cumulative hazard functions by listing the groups side by side.

```
. sts list, by(posttran) compare
Kaplan-Meier survivor function
By variable: posttran
```

posttran		0	1
Time	1	0.9903	1.0000
	225	0.4422	0.3934
	449	0.2359	0.3304
	673	0.2359	0.3139
	897	0.2359	0.2535
	1121	0.2359	0.1774
	1345	0.2359	0.1774
	1569	.	0.1420
	1793	.	0.1420
	2017	.	.

If we include the `cumhaz` option, the cumulative hazard functions are listed:

```
. sts list, cumhaz by(posttran) compare
Nelson-Aalen cumulative hazard function
By variable: posttran
posttran                  0            1

Time        1        0.0097       0.0000
          225        0.7896       0.9145
          449        1.3229       1.0850
          673        1.3229       1.1350
          897        1.3229       1.3411
         1121        1.3229       1.6772
         1345        1.3229       1.6772
         1569            .        1.8772
         1793            .        1.8772
         2017            .            .
```

When you specify `compare`, the same detailed survivor or cumulative hazard function is calculated, but it is then evaluated at 10 or so given times, and those evaluations are listed.

Also see the `risktable(`*numlist*`)` option for displaying an at-risk table with `sts list` (example 1 in [ST] **sts list**) and `sts graph` (*Adding an at-risk table* in [ST] **sts graph**).

Testing equality of survivor functions

`sts test` tests equality of survivor functions:

```
. sts test posttran
Equality of survivor functions
Log-rank test
                  Observed        Expected
posttran          events          events

       0            30             31.20
       1            45             43.80

   Total            75             75.00
                        chi2(1) =    0.13
                        Pr>chi2 = 0.7225
```

When you do not specify otherwise, `sts test` performs the log-rank test, but it can also perform the Wilcoxon test:

```
. sts test posttran, wilcoxon
Equality of survivor functions
Wilcoxon-Breslow-Gehan test
                  Observed        Expected        Sum of
posttran          events          events          ranks

       0            30             31.20            -85
       1            45             43.80             85

   Total            75             75.00              0
                        chi2(1) =    0.14
                        Pr>chi2 = 0.7083
```

`sts test` also performs stratified tests; see [ST] **sts test**.

Covariate-adjusted estimates

All the estimates of the survivor function we have seen so far are the Kaplan–Meier product-limit estimates. sts can make covariate-adjusted estimates of the survivor and related functions. We want to illustrate this and explain how it is done.

The heart transplant dataset is not the best for demonstrating this feature because we are starting with survivor functions that are similar already, so let's switch to data on a fictional drug trial:

```
. use https://www.stata-press.com/data/r18/drug2, clear
(Patient survival in drug trial)

. st
-> stset studytime, failure(died)

Survival-time data settings

         Failure event: died!=0 & died<.
Observed time interval: (0, studytime]
     Exit on or before: failure
```

The st command showed us how the dataset is currently declared.

This dataset contains 48 subjects, all observed from time 0:

```
. stdescribe

      Failure _d: died
  Analysis time _t: studytime
```

Category	Total	Mean	Min	Median	Max
			— Per subject —		
Number of subjects	48				
Number of records	48	1	1	1	1
Entry time (first)		0	0	0	0
Exit time (final)		15.5	1	12.5	39
Subjects with gap	0				
Time on gap	0				
Time at risk	744	15.5	1	12.5	39
Failures	31	.6458333	0	1	1

The dataset contains variables age and drug:

```
. summarize age drug
```

Variable	Obs	Mean	Std. dev.	Min	Max
age	48	47.125	9.492718	32	67
drug	48	.5833333	.4982238	0	1

We are comparing the outcomes of `drug = 1` with that of the placebo, `drug = 0`. Here are the survivor curves for the two groups:

. sts graph, by(drug)
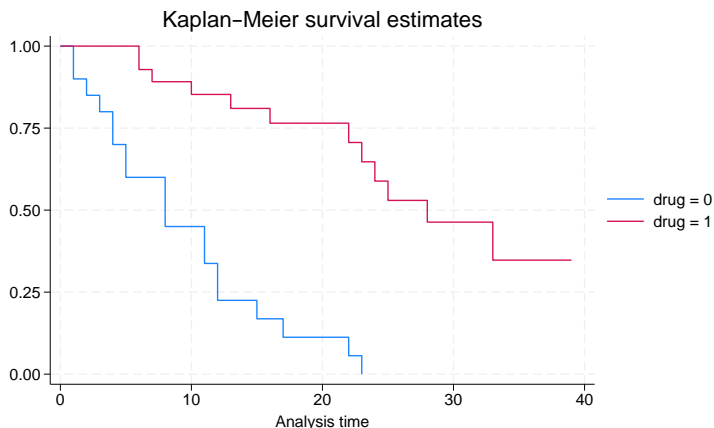
Here are the survivor curves adjusted for age (and scaled to age 50):

. sts graph, by(drug) adjustfor(age, at(age=50))
          Failure _d: died
    Analysis time _t: studytime
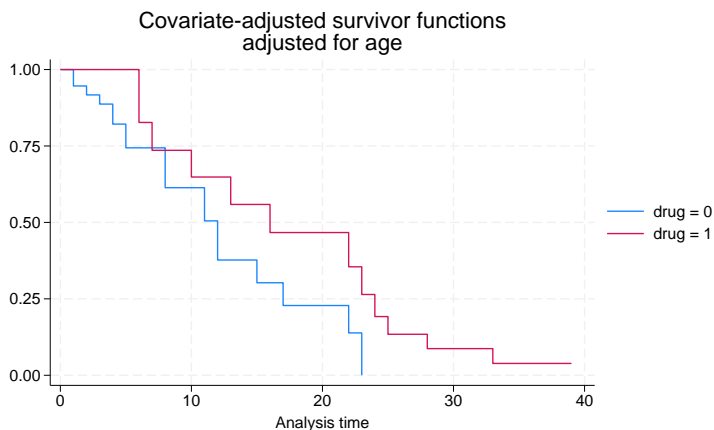
The age difference between the two samples accounts for much of the difference between the survivor functions.

By default, `adjustfor()` adjusts the survivor functions to 0 values of the covariates. Adding the suboption `at(age=50)` adjusts the estimates to age 50 instead of age 0.

The above is equivalent to the following:

```
. generate age50 = age-50
. sts graph, by(drug) adjustfor(age50)
```

When you type by(*group*) adjustfor(*vars*), sts fits a separate Cox proportional hazards model on *vars* (estimation via stcox) and retrieves the separately estimated baseline survivor functions. sts graph graphs the baseline survivor functions, sts list lists them, and sts generate saves them.

Thus sts list can list what sts graph plots:

```
. sts list, by(drug) adjustfor(age, at(age=50)) compare

        Failure _d: died
  Analysis time _t: studytime

Covariate-adjusted survivor function
By variable: drug
Adjusted for: age at at() values

drug                    0          1

Time      1        0.9463     1.0000
          5        0.7439     1.0000
          9        0.6135     0.7358
         13        0.3770     0.5588
         17        0.2282     0.4668
         21        0.2282     0.4668
         25             .     0.1342
         29             .     0.0872
         33             .     0.0388
         37             .     0.0388
         41             .          .
```

In both the graph and the listing, we must adjust for variable age at age $= 50$ by specifying the suboption at() in adjustfor(). Otherwise, the survivor functions are adjusted to age $= 0$, which is at birth:

```
. sts list, by(drug) adjustfor(age) compare

        Failure _d: died
  Analysis time _t: studytime

Covariate-adjusted survivor function
By variable: drug
Adjusted for: age = 0

drug                    0          1

Time      1        0.9994     1.0000
          5        0.9970     1.0000
          9        0.9951     0.9995
         13        0.9903     0.9990
         17        0.9853     0.9987
         21        0.9853     0.9987
         25             .     0.9965
         29             .     0.9958
         33             .     0.9944
         37             .     0.9944
         41             .          .
```

These are equivalent to what we obtained previously but not nearly so informative because of the scaling of the survivor function. The adjustfor(age) option scales the survivor function to correspond to age $= 0$. age is calendar age, so the survivor function is scaled to correspond to a newborn.

There is another way that `sts` will adjust the survivor function. Rather than specifying `by(`*group*`)` `adjustfor(`*vars*`)`, we can specify `strata(`*group*`)` `adjustfor(`*vars*`)`:

```
. sts list, strata(drug) adjustfor(age, at(age=50)) compare

          Failure _d: died
    Analysis time _t: studytime

Stratified covariate-adjusted survivor function
Strata variable: drug
Adjusted for: age at at() values

drug                    0            1
───────────────────────────────────────
Time      1         0.9526       1.0000
          5         0.7668       1.0000
          9         0.6417       0.7626
         13         0.4080       0.5995
         17         0.2541       0.5139
         21         0.2541       0.5139
         25            .         0.1800
         29            .         0.1247
         33            .         0.0614
         37            .         0.0614
         41            .            .
───────────────────────────────────────
```

When we specify `strata()` instead of `by()`, instead of fitting separate Cox models for each stratum, `sts list` fits one stratified Cox model and retrieves the stratified baseline survivor function. That is, `strata()` rather than `by()` constrains the effect of the `adjustfor()` variables to be the same across strata. In our example, the survivor function is also adjusted to age 50.

Counting the number lost to censoring

sts list shows the number lost in the fourth column of its output:

```
. sts list

       Failure _d: died
  Analysis time _t: studytime

Kaplan-Meier survivor function
```

Time	At risk	Fail	Lost	Survivor function	Std. error	[95% conf. int.]	
1	48	2	0	0.9583	0.0288	0.8435	0.9894
2	46	1	0	0.9375	0.0349	0.8186	0.9794
3	45	1	0	0.9167	0.0399	0.7930	0.9679
4	44	2	0	0.8750	0.0477	0.7427	0.9418
5	42	2	0	0.8333	0.0538	0.6943	0.9129
6	40	2	1	0.7917	0.0586	0.6474	0.8820
7	37	1	0	0.7703	0.0608	0.6236	0.8656
8	36	3	1	0.7061	0.0661	0.5546	0.8143
9	32	0	1	0.7061	0.0661	0.5546	0.8143
10	31	1	1	0.6833	0.0678	0.5302	0.7957
11	29	2	1	0.6362	0.0708	0.4807	0.7564
12	26	2	0	0.5872	0.0733	0.4304	0.7145
13	24	1	0	0.5628	0.0742	0.4060	0.6931
15	23	1	1	0.5383	0.0749	0.3821	0.6712
16	21	1	0	0.5127	0.0756	0.3570	0.6483
17	20	1	1	0.4870	0.0761	0.3326	0.6249
19	18	0	2	0.4870	0.0761	0.3326	0.6249
20	16	0	1	0.4870	0.0761	0.3326	0.6249
22	15	2	0	0.4221	0.0786	0.2680	0.5684
23	13	2	0	0.3572	0.0788	0.2087	0.5083
24	11	1	0	0.3247	0.0780	0.1809	0.4771
25	10	1	1	0.2922	0.0767	0.1543	0.4449
28	8	1	1	0.2557	0.0753	0.1247	0.4093
32	6	0	2	0.2557	0.0753	0.1247	0.4093
33	4	1	0	0.1918	0.0791	0.0676	0.3634
34	3	0	1	0.1918	0.0791	0.0676	0.3634
35	2	0	1	0.1918	0.0791	0.0676	0.3634
39	1	0	1	0.1918	0.0791	0.0676	0.3634

sts graph, if you specify the **lost** option, will show that number, too, except it will report a total number lost for the time interval where the survivor function is constant:

. sts graph, lost
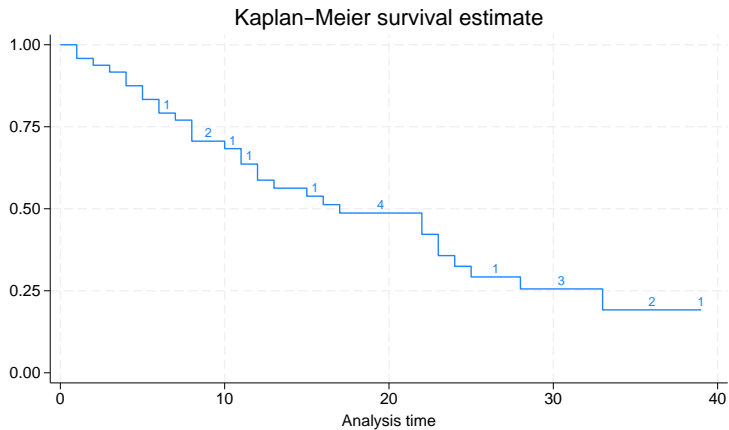
With complicated survival data—meaning delayed entry or multiple records per subject—**sts list** reports the number of net lost, defined as the number of censored minus the number who enter, instead of the number of lost. With simple survival data—with 1 observation per subject—net lost corresponds to lost.

With more complicated survival data, the number of net lost may surprise you. With complicated data, the vague term *lost* can mean many things. Sometimes subjects are lost, but mostly there are many censorings followed by reentries—a subject is censored at time 5 immediately to reenter the data with different covariates. This is called thrashing.

There are other possibilities: a subject can be lost, but only for a while, and so reenter the data with a gap; a subject can be censored out of one stratum to enter another. There are too many possibilities to dedicate a column in a table or a plotting symbol in a graph to each one. **sts**'s solution is to define *lost* as *net lost*, meaning censored minus entered, and show that number for complicated survival data. How we define *lost* does not affect the calculation of the survivor function; it merely affects a number that researchers often report.

Defining lost as censored minus entered results in exactly what is desired for simple survival data. Because everybody enters at time 0, calculating censored minus entered amounts to calculating censored $- 0$. The number of net lost is the number of censored.

In more complicated data, calculating censored minus entered results in the number really lost if there are no gaps and no delayed entry. Then the subtraction smooths the thrashing. In an interval, five might be censored and three reenter, so $5 - 3 = 2$ were lost.

In even more complicated data, calculating censored minus entered results in something reasonable once you understand how to interpret negative numbers and are cautious in interpreting positive ones. Five might be censored and three might enter (from the five? who can say?), resulting in two net lost; or three might be censored and five enter, resulting in -2 net lost.

sts with more complicated survival data, by default, reports the net lost but will, if you specify the **enter** option, report the pure number censored and the pure number who enter. Sometimes you will want to do that. Earlier in this entry, we used **sts list** to display the survivor functions in the Stanford heart transplant data for subjects pre- and posttransplantation, and we slipped in an **enter** option:

```
. use https://www.stata-press.com/data/r18/stan3, clear
(Heart transplant data)

. stset, noshow

. sts list, by(posttran) enter
```

Kaplan–Meier survivor function
By variable: posttran

Time	At risk	Fail	Lost	Enter	Survivor function	Std. error	[95% conf. int.]	
posttran=0								
0	0	0	0	103	1.0000	.	.	.
1	103	1	3	0	0.9903	0.0097	0.9331	0.9986
2	99	3	3	0	0.9603	0.0195	0.8976	0.9849
3	93	3	3	0	0.9293	0.0258	0.8574	0.9657
(output omitted)								
427	2	0	1	0	0.2359	0.1217	0.0545	0.4882
1400	1	0	1	0	0.2359	0.1217	0.0545	0.4882
posttran=1								
1	0	0	0	3	1.0000	.	.	.
2	3	0	0	3	1.0000	.	.	.
3	6	0	0	3	1.0000	.	.	.
4	9	0	0	2	1.0000	.	.	.
5	11	0	0	3	1.0000	.	.	.
5.1	14	1	0	0	0.9286	0.0688	0.5908	0.9896
6	13	0	0	1	0.9286	0.0688	0.5908	0.9896
8	14	0	0	2	0.9286	0.0688	0.5908	0.9896
(output omitted)								
1586	2	0	1	0	0.1420	0.0546	0.0566	0.2653
1799	1	0	1	0	0.1420	0.0546	0.0566	0.2653

We did that to keep you from being shocked at negative numbers for the net lost. In this complicated dataset, the value of posttran changes over time. All patients start with posttran = 0, and later some change to posttran = 1.

Thus, at time 1 in the `posttran = 0` group, three are lost—to the group but not to the experiment. Simultaneously, in the `posttran = 1` group, we see that three enter. Had we not specified the `enter` option, you would not have seen that three enter, and you would have seen that −3 were, in net, lost:

```
. sts list, by(posttran)
Kaplan-Meier survivor function
By variable: posttran
```

| | At | | Net | Survivor | Std. | | |
Time	risk	Fail	lost	function	error	[95% conf. int.]	
posttran=0							
1	103	1	3	0.9903	0.0097	0.9331	0.9986
2	99	3	3	0.9603	0.0195	0.8976	0.9849
3	93	3	3	0.9293	0.0258	0.8574	0.9657
(output omitted)							
427	2	0	1	0.2359	0.1217	0.0545	0.4882
1400	1	0	1	0.2359	0.1217	0.0545	0.4882
posttran=1							
1	0	0	-3	1.0000	.	.	.
2	3	0	-3	1.0000	.	.	.
3	6	0	-3	1.0000	.	.	.
4	9	0	-2	1.0000	.	.	.
5	11	0	-3	1.0000	.	.	.
5.1	14	1	0	0.9286	0.0688	0.5908	0.9896
6	13	0	-1	0.9286	0.0688	0.5908	0.9896
8	14	0	-2	0.9286	0.0688	0.5908	0.9896
(output omitted)							
1586	2	0	1	0.1420	0.0546	0.0566	0.2653
1799	1	0	1	0.1420	0.0546	0.0566	0.2653

```
Note: Net lost equals the number lost minus the number who entered.
```

Here specifying `enter` makes the table easier to explain, but do not jump to the conclusion that specifying `enter` is always a good idea. In this same dataset, let's look at the overall survivor function, first with the `enter` option:

```
. sts list, enter
Kaplan-Meier survivor function
```

| | At | | | | Survivor | Std. | | |
Time	risk	Fail	Lost	Enter	function	error	[95% conf. int.]	
0	0	0	0	103	1.0000	.	.	.
1	103	1	3	3	0.9903	0.0097	0.9331	0.9986
2	102	3	3	3	0.9612	0.0190	0.8998	0.9852
3	99	3	3	3	0.9320	0.0248	0.8627	0.9670
(output omitted)								
1571	3	0	1	0	0.1519	0.0493	0.0713	0.2606
1586	2	0	1	0	0.1519	0.0493	0.0713	0.2606
1799	1	0	1	0	0.1519	0.0493	0.0713	0.2606

At time 1, three are lost and three enter. There is no delayed entry in this dataset, and there are no gaps; so, it is the same three that were lost and reentered, and no one was really lost. At time 1571, on the other hand, a patient really was lost. This is all more clearly revealed when we do not specify the enter option:

```
. sts list

Kaplan–Meier survivor function

                  At            Net    Survivor      Std.
    Time        risk   Fail    lost    function     error     [95% conf. int.]

       1         103      1       0      0.9903    0.0097     0.9331     0.9986
       2         102      3       0      0.9612    0.0190     0.8998     0.9852
       3          99      3       0      0.9320    0.0248     0.8627     0.9670
(output omitted )
    1571           3      0       1      0.1519    0.0493     0.0713     0.2606
    1586           2      0       1      0.1519    0.0493     0.0713     0.2606
    1799           1      0       1      0.1519    0.0493     0.0713     0.2606

Note: Net lost equals the number lost minus the number who entered.
```

Thus, to summarize:

- The sts list and graph commands will show the number lost or censored. sts list, by default, shows this number on the detailed output. sts graph shows the number when you specify the lost option.

- By default, the number lost with simple survival data is the pure number lost or censored and with complicated survival data is the net number lost, defined as censored minus entered.

- Both commands allow you to specify the enter option and then show the number who actually entered, and the number lost becomes the actual number censored, not censored minus entered.

❑ Technical note

There is one other issue about the Kaplan–Meier estimator regarding delayed entry. When the earliest entry into the study occurs after $t = 0$, one may still calculate the Kaplan–Meier estimation, but the interpretation changes. Rather than estimating $S(t)$, you are now estimating $S(t|t_{\min})$, the probability of surviving past time t given survival to time $t_{\min}$, where $t_{\min}$ is the earliest entry time.

❑

Video examples

How to graph survival curves

How to calculate the Kaplan-Meier survivor and Nelson-Aalen cumulative hazard functions

How to test the equality of survivor functions using nonparametric tests

Stored results

See *Stored results* in [ST] **sts test**.

Methods and formulas

Unless covariate-adjusted estimates are requested, `sts` estimates the survivor function by using the Kaplan–Meier product-limit method.

When the `cumhaz` option is specified, `sts` estimates the cumulative hazard function by using the Nelson–Aalen estimator.

For an introduction to the Kaplan–Meier product-limit method and the log-rank test, see Pagano and Gauvreau (2022, 487–491) and Oliveira (2013); for a detailed discussion, see Cox and Oakes (1984), Kalbfleisch and Prentice (2002), or Klein and Moeschberger (2003). For an introduction to survival analysis with examples using the `sts` commands, see Dupont (2009).

Let t_j, $j = 1, \ldots,$ denote the times at which failure occurs. Let n_j be the number at risk of failure just before time t_j and d_j be the number of failures at time t_j. Let $S(t) = \Pr(T > t)$ define the survivor function and $F(t) = 1 - S(t) = \Pr(T \leq t)$ define the failure function.

The nonparametric maximum-likelihood estimate of the survivor function (Kaplan and Meier 1958) is

$$\widehat{S}(t) = \prod_{j \mid t_j \leq t} \left(\frac{n_j - d_j}{n_j} \right)$$

(Kalbfleisch and Prentice 2002, 15).

The estimate of the failure function, $\widehat{F}(t)$, is computed as $1 - \widehat{S}(t)$.

The standard error reported is given by Greenwood's formula (Greenwood 1926):

$$\widehat{\text{Var}}\{\widehat{S}(t)\} = \widehat{S}^2(t) \sum_{j \mid t_j \leq t} \frac{d_j}{n_j(n_j - d_j)}$$

(Kalbfleisch and Prentice 2002, 17–18). These standard errors, however, are not used for confidence intervals. Instead, the asymptotic variance of $\ln[-\ln \widehat{S}(t)]$,

$$\widehat{\sigma}^2(t) = \frac{\sum \frac{d_j}{n_j(n_j - d_j)}}{\left\{ \sum \ln\left(\frac{n_j - d_j}{n_j} \right) \right\}^2}$$

is used, where sums are calculated over $j \mid t_j \leq t$ (Kalbfleisch and Prentice 2002, 18). The confidence bounds are then $\widehat{S}(t)^{\exp(\pm z_{\alpha/2} \widehat{\sigma}(t))}$, where $z_{\alpha/2}$ is the $(1 - \alpha/2)$ quantile of the normal distribution. `sts` suppresses reporting the standard error and confidence bounds if the data are `pweighted` because these formulas are no longer appropriate.

When the `adjustfor()` option is specified, the survivor function estimate, $\widehat{S}(t)$, is the baseline survivor function estimate $\widehat{S}_0(t)$ of `stcox`; see [ST] **stcox**. If, `by()`, is specified, $\widehat{S}(t)$ is obtained from fitting separate Cox models on `adjustfor()` for each of the `by()` groups. If instead `strata()` is specified, one Cox model on `adjustfor()`, stratified by `strata()`, is fit. If an adjustment to covariate values other than 0 is requested, $\widehat{S}(t)$ is the survivor function adjusted for the specified values.

The Nelson–Aalen estimator of the cumulative hazard rate function is derived from Nelson (1972) and Aalen (1978) and is defined up to the largest observed time as

$$\widehat{H}(t) = \sum_{j \mid t_j \leq t} \frac{d_j}{n_j}$$

Its variance (Aalen 1978) may be estimated by

$$\widehat{\mathrm{Var}}\{\widehat{H}(t)\} = \sum_{j|t_j \leq t} \frac{d_j}{{n_j}^2}$$

Pointwise confidence intervals are calculated using the asymptotic variance of $\ln \widehat{H}(t)$,

$$\widehat{\phi}^2(t) = \frac{\widehat{\mathrm{Var}}\{\widehat{H}(t)\}}{\{\widehat{H}(t)\}^2}$$

The confidence bounds are then $\widehat{H}(t) \exp\{\pm z_{\alpha/2}\widehat{\phi}(t)\}$. If the data are `pweight`ed, these formulas are not appropriate, and then confidence intervals are not reported.

Major Greenwood (1880–1949) was born in London to a medical family. His given name, "Major", was also that of his father and grandfather. Greenwood trained as a doctor but followed a career in medical research, learning statistics from Karl Pearson. He worked as a medical statistician and epidemiologist at the Lister Institute, the Ministry of Health, and the London School of Hygiene and Tropical Medicine. With interests ranging from clinical trials to historical subjects, Greenwood played a major role in developing biostatistics in the first half of the twentieth century.

Edward Lynn Kaplan (1920–2006) was born in Philadelphia, Pennsylvania. He obtained a bachelors degree in mathematics from Carnegie Mellon University in 1941 and in 1951 obtained a PhD in mathematics from Princeton University. His graduate adviser was John W. Tukey, who was also Paul Meier's advisor. Both Kaplan and Meier were working on the issue of censored observations around the same time, but Edward Kaplan submitted his paper to the *Journal of the American Statistical Association* first. When Meier submitted his paper, it was turned down, and the two were prompted to instead collaborate on a single publication. This gave rise to the 1958 paper introducing the Kaplan–Meier estimator, a method of estimating survival functions. The estimator made a lasting contribution to the medical field but has also been applied to several other fields when studying the time until a certain event occurs. In 1961, Kaplan joined the faculty at Oregon State University as a mathematics professor and in 1982 published a book on mathematical programming. He passed away in 2006 and is remembered with warm regard.

Paul Meier (1924–2011) was born in Newark, New Jersey. Meier obtained a PhD in statistics from Princeton University and later taught at the University of Chicago, where he spent 35 years of his career. In 1958, he published a paper with Edward L. Kaplan introducing the Kaplan–Meier estimator, a method of estimating survival functions, which has been applied to the research of many diseases. The article has over 53,000 citations.

Meier revolutionized medical research by strongly advocating for randomized clinical trials. He was one of the founding members of the Society for Clinical Trials, serving as president in 1986 and 1987. In 1986, the American Statistical Association named him Statistician of the Year. He was also elected as fellow of the John Guggenheim Memorial Foundation, the Royal Statistical Society, and the American Academy of Arts and Sciences. The extent of his contributions is not limited to his research. He also served on advisory committees for the Food and Drug Administration, the National Institutes of Health, and some environmental committees.

Wayne B. Nelson (1936–) was born in Chicago and received degrees in physics and statistics from Caltech and the University of Illinois. A longtime employee of General Electric, he now works as a consultant, specializing in reliability analysis and accelerated testing.

Nelson and Odd Aalen met for the first time at a conference at the University of South Carolina in 2003.

Odd Olai Aalen (1947–) was born in Oslo, Norway, and studied there and at Berkeley, where he was awarded a PhD in 1975 for a thesis on counting processes. He is a professor of statistics at the University of Oslo and works on survival and event history analysis. Aalen was one of the prime movers in introducing martingale ideas to this branch of statistics.

Aalen and Wayne Nelson met for the first time at a conference at the University of South Carolina in 2003.

References

Aalen, O. O. 1978. Nonparametric inference for a family of counting processes. *Annals of Statistics* 6: 701–726. https://doi.org/10.1214/aos/1176344247.

Breslow, N. E. 1992. Kaplan and Meier (1958) "Nonparametric estimation from incomplete observations". In *Breakthroughs in Statistics, Vol. II: Methodology and Distribution*, ed. S. Kotz and N. L. Johnson, 311–338. New York: Springer.

Clerc-Urmès, I., M. Grzebyk, and G. Hédelin. 2014. Net survival estimation with stns. *Stata Journal* 14: 87–102.

Coviello, V., P. W. Dickman, K. Seppä, and A. Pokhrel. 2015. Estimating net survival using a life-table approach. *Stata Journal* 15: 173–185.

Cox, D. R., and D. Oakes. 1984. *Analysis of Survival Data*. London: Chapman and Hall/CRC.

Dickman, P. W., and V. Coviello. 2015. Estimating and modeling relative survival. *Stata Journal* 15: 186–215.

Dupont, W. D. 2009. *Statistical Modeling for Biomedical Researchers: A Simple Introduction to the Analysis of Complex Data*. 2nd ed. Cambridge: Cambridge University Press.

Greenwood, M. 1926. The natural duration of cancer. *Reports on Public Health and Medical Subjects* 33: 1–26.

Hogben, L. T. 1950. Major Greenwood, 1880–1949. *Obituary Notices of Fellows of the Royal Society* 7: 139–154. https://doi.org/10.1098/rsbm.1950.0010.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Kaplan, E. L., and P. Meier. 1958. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association* 53: 457–481. https://doi.org/10.2307/2281868.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data*. 2nd ed. New York: Springer.

Linhart, J. M., J. S. Pitblado, and J. F. Hassell. 2004. From the help desk: Kaplan–Meier plots with stsatrisk. *Stata Journal* 4: 56–65.

Lo, S.-H., and D. Madigan. 2012. Obituary: Paul Meier 1924–2011. *IMS Bulletin* 41(1): 6.

Marks, H. M. 2004. A conversation with Paul Meier. *Clinical Trials* 1: 131–138. https://doi.org/10.1191/1740774504cn011xx.

Nelson, W. 1972. Theory and applications of hazard plotting for censored failure data. *Technometrics* 14: 945–966. https://doi.org/10.2307/1267144.

Newman, S. C. 2001. *Biostatistical Methods in Epidemiology*. New York: Wiley.

Oliveira, A. G. 2013. *Biostatistics Decoded*. Chichester, UK: Wiley.

Pagano, M., and K. Gauvreau. 2022. *Principles of Biostatistics*. 3rd ed. Boca Raton, FL: CRC Press.

Smythe, B. 2006. Obituary: Edward Kaplan 1920–2006. *IMS Bulletin* 35(10): 7.

Stalpers, L. J. A., and E. L. Kaplan. 2018. Edward L. Kaplan and the Kaplan–Meier survival curve. *BSHM Bulletin* 33: 109–135. https://doi.org/10.1080/17498430.2018.1450055.

Wilkinson, L. 1998. Greenwood, Major. In Vol. 2 of *Encyclopedia of Biostatistics*, ed. P. Armitage and T. Colton, 1778–1780. Chichester, UK: Wiley.

Also see

[ST] **stci** — Confidence intervals for means and percentiles of survival time

[ST] **stcox** — Cox proportional hazards model

[ST] **sts generate** — Create variables containing survivor and related functions

[ST] **sts graph** — Graph the survivor or related function

[ST] **sts list** — List the survivor or related function

[ST] **sts test** — Test equality of survivor functions

[ST] **stset** — Declare data to be survival-time data

[ST] *adjustfor_option* — Adjust survivor and related functions for covariates at specific values

[ST] **Survival analysis** — Introduction to survival analysis commands

Title

---

sts generate — Create variables containing survivor and related functions

---

Description

sts generate creates new variables containing the estimated survivor and failure functions, the Nelson–Aalen cumulative hazard (integrated hazard) function, and other related functions. See [ST] **sts** for an introduction to this command.

sts generate can be used with single- or multiple-record or single- or multiple-failure st data.

Quick start

Create new variable surv containing the Kaplan–Meier survivor function using stset data

    sts generate surv = s

Create sesurv containing the pointwise standard error for the survivor function

    sts generate sesurv = se(s)

Create surv2 with separate survivor functions for each level of v1

    sts generate surv2 = s, by(v1)

Create surv3 with survivor function adjusted for $v2 = 0$

    sts generate surv3 = s, adjustfor(v2)

Same as above, but create surv4 with stratification by levels of svar

    sts generate surv3 = s, adjustfor(v2) strata(svar)

Create cumhaz containing the Nelson–Aalen estimate of the cumulative hazard function, and create lbchaz and ubchaz containing 95% lower and upper confidence interval bounds for the estimated function

    sts generate cumhaz = na lbchaz = lb(na) ubchaz = ub(na)

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Generate survivor and related functions

Syntax

> sts <u>generate</u> *newvar* =
>
> $\Big\{$ s $\mid$ se(s) $\mid$ h $\mid$ se(lls) $\mid$ lb(s) $\mid$ ub(s) $\mid$ f $\mid$ se(f) $\mid$ lb(f) $\mid$ ub(f) $\mid$ na $\mid$ se(na) $\mid$ lb(na) $\mid$
>
> ub(na) $\mid$ n $\mid$ d $\Big\}$
>
> $\big[$ *newvar* = $\{\ldots\}$ $\ldots\big]$ $\big[$ *if* $\big]$ $\big[$ *in* $\big]$ $\big[$, *options* $\big]$

options	Description
Options	
by(*varlist*)	calculate separately for each group formed by *varlist*
strata(*varlist*)	stratify on different groups of *varlist*
<u>adjust</u>for(*varlist* $\big[$, *suboptions* $\big]$)	adjust the estimates to specific values of *varlist*; default is zero values
<u>l</u>evel(#)	set confidence level; default is level(95)

You must stset your data before using sts generate; see [ST] **stset**.

Functions

> **Main**

s produces the Kaplan–Meier product-limit estimate of the survivor function, $\widehat{S}(t)$, or, if adjustfor() is specified, the baseline survivor function from a Cox regression model on the adjustfor() variables. If adjustment to covariate values other than 0 is requested, the baseline survivor function is adjusted to the specified values.

se(s) produces the Greenwood, pointwise standard error, $\widehat{se}\{\widehat{S}(t)\}$. This option may not be used with adjustfor().

h produces the estimated hazard component, $\Delta H_j = H(t_j) - H(t_{j-1})$, where t_j is the current failure time and t_{j-1} is the previous one. This is mainly a utility function used to calculate the estimated cumulative hazard, $H(t_j)$, yet you can estimate the hazard via a kernel smooth of the ΔH_j; see [ST] **sts graph**. It is recorded at all the points at which a failure occurs and is computed as d_j/n_j, where d_j is the number of failures occurring at time t_j and n_j is the number at risk at t_j before the occurrence of the failures.

se(lls) produces $\widehat{\sigma}(t)$, the standard error of $\ln\{-\ln\widehat{S}(t)\}$. This option may not be used with adjustfor().

lb(s) produces the lower bound of the confidence interval for $\widehat{S}(t)$ based on $\ln\{-\ln\widehat{S}(t)\}$: $\widehat{S}(t)^{\exp(-z_{\alpha/2}\widehat{\sigma}(t))}$, where $z_{\alpha/2}$ is the $(1 - \alpha/2)$ quantile of the standard normal distribution. This option may not be used with adjustfor().

ub(s) produces the upper bound of the confidence interval for $\widehat{S}(t)$ based on $\ln\{-\ln\widehat{S}(t)\}$: $\widehat{S}(t)^{\exp(z_{\alpha/2}\widehat{\sigma}(t))}$, where $z_{\alpha/2}$ is the $(1 - \alpha/2)$ quantile of the standard normal distribution. This option may not be used with adjustfor().

f produces the Kaplan–Meier product-limit estimate of the failure function, $1 - \widehat{S}(t)$, or, if adjustfor() is specified, produces the baseline failure function from a Cox regression model on the adjustfor() variables. If adjustment to covariate values other than 0 is requested, the baseline failure function is adjusted to the specified values.

`se(f)` produces the Greenwood, pointwise standard error, $\widehat{\mathrm{se}}\{1 - \widehat{S}(t)\} = \widehat{\mathrm{se}}\{\widehat{S}(t)\}$. This option may not be used with `adjustfor()` and is a synonym for `se(s)`, except the variable labeling.

`lb(f)` produces the lower bound of the confidence interval for $1 - \widehat{S}(t)$ based on $\ln\{-\ln\widehat{S}(t)\}$: $\widehat{S}(t)^{\exp(-z_{\alpha/2}\widehat{\sigma}(t))}$, where $z_{\alpha/2}$ is the $(1 - \alpha/2)$ quantile of the standard normal distribution. This option may not be used with `adjustfor()`.

`ub(f)` produces the upper bound of the confidence interval for $1 - \widehat{S}(t)$ based on $\ln\{-\ln\widehat{S}(t)\}$: $\widehat{S}(t)^{\exp(z_{\alpha/2}\widehat{\sigma}(t))}$, where $z_{\alpha/2}$ is the $(1 - \alpha/2)$ quantile of the standard normal distribution. This option may not be used with `adjustfor()`.

`na` produces the Nelson–Aalen estimate of the cumulative hazard function. This option may not be used with `adjustfor()`.

`se(na)` produces pointwise standard error for the Nelson–Aalen estimate of the cumulative hazard function, $\widehat{H}(t)$. This option may not be used with `adjustfor()`.

`lb(na)` produces the lower bound of the confidence interval for $\widehat{H}(t)$ based on the log-transformed cumulative hazard function. This option may not be used with `adjustfor()`.

`ub(na)` produces the corresponding upper bound. This option may not be used with `adjustfor()`.

`n` produces n_j, the number at risk just before time t_j. This option may not be used with `adjustfor()`.

`d` produces d_j, the number failing at time t_j. This option may not be used with `adjustfor()`.

Options

> Options

`by(`*varlist*`)` performs a separate calculation for each by-group. By-groups are identified by equal values of the variables in *varlist*. `by()` may not be combined with `strata()`.

`strata(`*varlist*`)` requests estimates of the survivor, failure, and hazard functions stratified on variables in *varlist*. It requires specifying `adjustfor()` and may not be combined with `by()`.

`adjustfor(`*varlist*`[`, *suboptions*`])` adjusts the estimates of the survivor, failure, and hazard functions to specific values of *varlist*. The default is to adjust to 0 values, that is, to produce a baseline function. If you want to adjust the function to values different from 0, you can use `adjustfor()`'s `at()` suboption. `adjustfor()` is available only with functions s, f, and h.

suboptions are `atzeros` (the default), `atmeans`, `atomeans`, `atbase`, and `at()`; see [ST] *adjustfor_option*.

If you specify `adjustfor()` with `by()`, `sts generate` fits separate Cox regression models for each group, using the `adjustfor()` variables as covariates. The separately calculated baseline functions are then retrieved.

If you specify `adjustfor()` with `strata()`, `sts generate` fits a stratified-on-group Cox regression model, using the `adjustfor()` variables as covariates. The stratified baseline function is then retrieved.

If adjustment to covariate values other than 0 is requested, the function is estimated at the specified covariate values.

`level(#)` specifies the confidence level, as a percentage, for the `lb(s)`, `ub(s)`, `lb(f)`, `ub(f)`, `lb(na)`, and `ub(na)` functions. The default is `level(95)` or as set by `set level`; see [U] **20.8 Specifying the width of confidence intervals**.

Remarks and examples

sts generate is a seldom-used command that gives you access to the calculations listed by sts list and graphed by sts graph.

Use of this command is demonstrated in [ST] **sts**.

Methods and formulas

See [ST] **sts**.

References

See [ST] **sts** for references.

Also see

[ST] **sts** — Generate, graph, list, and test the survivor and related functions

[ST] **sts graph** — Graph the survivor or related function

[ST] **sts list** — List the survivor or related function

[ST] **sts test** — Test equality of survivor functions

[ST] **stset** — Declare data to be survival-time data

[ST] *adjustfor_option* — Adjust survivor and related functions for covariates at specific values

Title

> **sts graph —** Graph the survivor or related function

Description

sts graph graphs the estimated survivor or failure function, the Nelson–Aalen estimated cumulative (integrated) hazard function, or the estimated hazard function. See [ST] **sts** for an introduction to this command.

sts graph can be used with single- or multiple-record or single- or multiple-failure st data.

Quick start

Graph the Kaplan–Meier survivor function using stset data

    sts graph

Estimate and graph separate survivor functions for each level of v1

    sts graph, by(v1)

Same as above, and show number lost because of censoring at each time point on the plot

    sts graph, by(v1) lost

Add a table below the graph with number at risk in each group at times 0, 10, 20, and 30

    sts graph, by(v1) risktable(0(10)30)

Specify the color for each line

    sts graph, by(v1) plot1opts(lcolor(green)) plot2opts(lcolor(blue))

Graph the Nelson–Aalen cumulative hazard functions for each level of v1

    sts graph, by(v1) cumhaz

Same as above, and save the graph as mygraph.gph

    sts graph, by(v1) cumhaz saving(mygraph)

Graph the estimated hazard function

    sts graph, hazard

Same as above, but use the Gaussian kernel function for the kernel density estimate

    sts graph, hazard kernel(gaussian)

Menu

Statistics > Survival analysis > Graphs > Survivor or related function

Syntax

> sts graph [*if*] [*in*] [, *options*]

options	Description
Main	
survival	graph Kaplan–Meier survivor function; the default
failure	graph Kaplan–Meier failure function
cumhaz	graph Nelson–Aalen cumulative hazard function
hazard	graph smoothed hazard estimate
by(*varlist*)	estimate and graph separate functions for each group formed by *varlist*
strata(*varlist*)	stratify on different groups of *varlist*
separate	show curves on separate graphs; default is to show curves one on top of another
adjustfor(*varlist* [, *suboptions*])	adjust the estimates to specific values of *varlist*; default is zero values
ci	show pointwise confidence bands
At-risk table	
risktable	show table of number at risk beneath graph
risktable(*risk_spec*)	show customized table of number at risk beneath graph
Options	
level(*#*)	set confidence level; default is level(95)
per(*#*)	units to be used in reported rates
noshow	do not show st setting information
tmax(*#*)	show graph for $t \leq \#$
tmin(*#*)	show graph for $t \geq \#$
noorigin	begin survival or failure curve at first exit time; default is to begin at $t = 0$
width(*#* [*#...*])	override default bandwidth(s)
kernel(*kernel*)	kernel function
noboundary	no boundary correction
lost	show number lost
enter	show number entered and number lost
atrisk	show numbers at risk at beginning of each interval
censored(single)	show one hash mark at each censoring time, no matter what number is censored
censored(number)	show one hash mark at each censoring time and number censored above hash mark
censored(multiple)	show multiple hash marks for multiple censoring at the same time
censopts(*hash_options*)	affect rendition of hash marks
lostopts(*marker_label_options*)	affect rendition of numbers lost
atriskopts(*marker_label_options*)	affect rendition of numbers at risk

Plot

<u>plot</u>opts(*cline_options*)	affect rendition of plotted lines
<u>plot</u>#opts(*cline_options*)	affect rendition of the #th plotted line; may not be combined with separate

CI plot

<u>ci</u>opts(*area_options*)	affect rendition of confidence bands
<u>ci</u>#opts(*area_options*)	affect rendition of the #th confidence band; may not be combined with separate

Add plots

<u>addplot</u>(*plot*)	add other plots to the generated graph

Y axis, X axis, Titles, Legend, Overall

twoway_options	any options documented in [G-3] *twoway_options*
<u>byopts</u>(*byopts*)	how subgraphs are combined, labeled, etc.

where *risk_spec* is

  [*numlist*] [, *table_options* group(*group*)]

  numlist specifies the points at which the number at risk is to be evaluated, *table_options* customizes the table of number at risk, and group(*group*) specifies a specific group/row for *table_options* to be applied.

table_options	Description
Main	
axis_label_options	control table by using axis labeling options; seldom used
order(*order_spec*)	select which rows appear and their order
<u>righttitles</u>	place titles on right side of the table
<u>fail</u>events	show number failed in the at-risk table
<u>lastfail</u>ure	show number failed in the last time interval; requires failevents
text_options	affect rendition of table elements and titles
Row titles	
<u>rowtitle</u>([*text*] [, *rtext_options*])	change title for a row
Title	
title([*text*] [, *ttext_options*])	change overall table title

where *order_spec* is

  # ["*text*" ["*text*" ...]] [...]

text_options	Description
size(*textsizestyle*)	size of text
color(*colorstyle*)	color of text
justification(*justificationstyle*)	text left-justified, centered, or right-justified
format(%*fmt*)	format values per %*fmt*
topgap(*size*)	margin above rows
bottomgap(*size*)	margin beneath rows
style(*textstyle*)	overall style of text

style() does not appear in the dialog box.

rtext_options	Description
size(*textsizestyle*)	size of text
color(*colorstyle*)	color of text
justification(*justificationstyle*)	text left-justified, centered, or right-justified
at(*#*)	override x position of titles
topgap(*size*)	margin above rows
style(*textstyle*)	overall style of text

style() does not appear in the dialog box.

ttext_options	Description
size(*textsizestyle*)	size of text
color(*colorstyle*)	color of text
justification(*justificationstyle*)	text left-justified, centered, or right-justified
at(*#*)	override x position of titles
topgap(*size*)	margin above rows
bottomgap(*size*)	margin beneath rows
style(*textstyle*)	overall style of text

style() does not appear in the dialog box.

group	Description
#*rownum*	specify group by row number in table
value	specify group by value of group
label	specify group by text of value label associated with group

hash_options	Description
line_options	change look of dropped lines
marker_label_options	add marker labels; any options documented in [G-3] *marker_label_options*, except mlabel()

risktable() may be repeated and is *merged-explicit*; see [G-4] **Concept: repeated options**.
You must stset your data before using sts graph; see [ST] **stset**.
fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

Options

    ⌐ Main └───

`survival`, `failure`, `cumhaz`, and `hazard` specify the function to graph.

> `survival` specifies that the Kaplan–Meier survivor function be plotted. This option is the default
> if a function is not specified.
>
> `failure` specifies that the Kaplan–Meier failure function, $1 - S(t + 0)$, be plotted.
>
> `cumhaz` specifies that the Nelson–Aalen estimate of the cumulative hazard function be plotted.
>
> `hazard` specifies that an estimate of the hazard function be plotted. This estimate is calculated
> as a weighted kernel-density estimate using the estimated hazard contributions, $\Delta \widehat{H}(t_j) = \widehat{H}(t_j) - \widehat{H}(t_{j-1})$. These hazard contributions are the same as those obtained by `sts generate`
> *newvar* = h.

`by(`*varlist*`)` estimates a separate function for each by-group and plots all the functions on one graph.
By-groups are identified by equal values of the variables in *varlist*. `by()` may not be combined
with `strata()`.

`strata(`*varlist*`)` produces estimates of the survivor, failure, or hazard functions stratified on variables
in *varlist* and plots all the groups on one graph. It requires specifying `adjustfor()` and may not
be combined with `by()`.

> If you have more than one `strata()` variable but need only one, use `egen` to create it; see
> [D] **egen**.

`separate` is meaningful only with `by()` or `strata()`; it requests that each group be placed on its
own graph rather than one on top of the other. Sometimes curves have to be placed on separate
graphs—such as when you specify `ci`—because otherwise it would be too confusing.

`adjustfor(`*varlist*`[ ,` *suboptions* `])` adjusts the estimates of the survivor, failure, or hazard function
to specific values of *varlist*. The default is to adjust to 0 values, that is, to produce a baseline
function. If you want to adjust the function to values different from 0, you can use `adjustfor()`'s
`at()` suboption.

> *suboptions* are `atzeros` (the default), `atmeans`, `atomeans`, `atbase`, and `at()`; see [ST] *adjust-
> for_option*.
>
> Say you want to produce the survivor function adjusted to age of patients and the ages in your
> sample are 40–60 years. If you want to adjust the function to age 40, type
>
> > `. sts graph, adjustfor(age, at(age=40))`
>
> Or if you want to adjust the function to the mean age, type
>
> > `. sts graph, adjustfor(age, atmeans)`
>
> `adjustfor()` is not available with `cumhaz` or `ci`.
>
> If you specify `adjustfor()` with `by()`, `sts graph` fits separate Cox regression models for each
> group, using the `adjustfor()` variables as covariates. The separately calculated baseline functions
> are then retrieved.
>
> If you specify `adjustfor()` with `strata()`, `sts graph` fits a stratified-on-group Cox regression
> model, using the `adjustfor()` variables as covariates. The stratified baseline function is then
> retrieved.
>
> If adjustment to covariate values other than 0 is requested, the function is estimated at the specified
> covariate values.

ci includes pointwise confidence bands. The default is not to produce these bands. ci is not allowed with adjustfor() or pweights.

> [At-risk table]

risktable or risktable([*numlist*][, *table_options*]) displays a table showing the number at risk beneath the plot. Neither option may be used with separate or adjustfor().

risktable displays the table in the default format with number at risk shown for each time reported on the x axis.

risktable([*numlist*][, *table_options*]) specifies that the number at risk be evaluated at the points specified in *numlist* or that the rendition of the table be changed by *table_options*.

At time 0, the number at risk is zero by definition. But for graphical purposes, sts graph reports the number of subjects in a study (the number who entered at time 0) at time 0 instead of a zero.

There are two ways to change the points at which the numbers at risk are evaluated.

1. The x axis of the graph may be altered. For example:

    . sts graph, xlabel(0(5)40) risktable

2. A *numlist* can be specified directly in the risktable() option. For example:

    . sts graph, risktable(0(5)40)

The above two examples produce the same at-risk table. In general, using xlabel() provides the x-axis labels only for the specified values, whereas using risktable() provides the graph's default labels as well as the labels for the specified values.

risktable(numlist) displays labels on the x axis for the specified *numlist* in addition to the graph's default x-axis labels. This may lead to options xmtick(###) and xmlabel(###), specified using the ### rule with sts graph, being ignored. In this case, you can manually specify equally spaced labels by using option xlabel(), or you can specify the positions for minor ticks or minor labels as a numlist in xmtick() or xmlabel().

table_options affect the rendition of the at-risk table and may be any of the following:

group(#*rownum* | *value* | *label*) specifies that all the suboptions specified in the risktable() apply only to the specified group. Because the risktable() option may be repeated, this option allows different rows of the at-risk table to be displayed with different colors, font sizes, etc.

When both a value and a value label are matched, the value label takes precedence.

risktable() may be specified with or without the group() suboption. When specified without group(), each suboption is applied to all available groups or rows. risktable() specified without group() is considered to be global and is itself merged-explicit. See [G-4] **Concept: repeated options** for more information on how repeated options are merged.

Consider the following example:

    . sts graph, by(drug) risktable(, color(red) size(small))
    > risktable(, color(navy))

The example above would produce a table where all rows are colored navy with small text.

Combining global risktable() options with group-specific risktable() options can be useful. When global options are combined with group-specific options, group-specific options always take precedence.

Consider the following example:

```
. sts graph, by(drug) risktable(, color(navy))
> risktable(, color(red) group(#1))
```

The example above would produce a table with the first row colored red and all remaining rows colored navy.

> Main

axis_label_options control the table by using axis labeling options. These options are seldom used. See [G-3] *axis_label_options*.

order() specifies which and in what order rows are to appear in the at-risk table. Optionally, order() can be used to override the default text.

order(# # # ...) is the syntax used for identifying which rows to display and their order. order(1 2 3) would specify that row 1 is to appear first in the table, followed by row 2, followed by row 3. order(1 2 3) is the default if there are three groups. If there were four groups, order(1 2 3 4) would be the default, and so on. If there were four groups and you specified order(1 2 3), the fourth row would not appear in the at-risk table. If you specified order(2 1 3), row 2 would appear first, followed by row 1, followed by row 3.

order(# "*text*" # "*text*" ...) is the syntax used for specifying the row order and alternate row titles.

Consider the following at-risk table:

```
drug = 1    20    8     2
drug = 2    14    10    4     1
drug = 3    14    13    10    5
```

Specifying order(1 "Placebo" 3 2) would produce

```
Placebo    20    8     2
drug = 3   14    13    10    5
drug = 2   14    10    4     1
```

and specifying order(1 "Placebo" 3 "Drug 2" 2 "Drug 1") would produce

```
Placebo    20    8     2
Drug 2     14    13    10    5
Drug 1     14    10    4     1
```

righttitles specifies that row titles be placed to the right of the at-risk values. The default is to place row titles to the left of the at-risk values.

failevents specifies that the number of failure events be shown in parentheses, after the time in which the risk values were calculated.

lastfailure specifies that the number of failure events be shown in parentheses, after the time in which the risk values were calculated, for the last time interval. By default, the number of failures is not displayed for the last time interval. lastfailure requires specifying failevents.

text_options affect the rendition of both row titles and number at risk and may be any of the following:

size(*textsizestyle*) specifies the size of text.

color(*colorstyle*) specifies the color of text.

justification(*justificationstyle*) specifies how text elements are to be justified.

format(%*fmt*) specifies how numeric values are to be formatted.

topgap(*size*) specifies how much space is to be placed above each row.

bottomgap(*size*) specifies how much space is to be placed beneath each row.

style(*textstyle*) specifies the style of text. This option does not appear in the dialog box.

Row titles

rowtitle([*text*][, *rtext_options*]) changes the default text or rendition of row titles. Specifying rowtitle(, color(navy)) would change the color of all row titles to navy.

rowtitle() is often combined with group() to change the text or rendition of a title. Specifying rowtitle(Placebo) group(#2) would change the title of the second row to Placebo. Specifying rowtitle(, color(red)) group(#3) would change the color of the row title for the third row to red.

Row titles may include more than one line. Lines are specified one after the other, each enclosed in double quotes. Specifying rowtitle("Experimental drug") group(#1) would produce a one-line row title, and specifying rowtitle("Experimental" "Drug") group(#1) would produce a multiple-line row title.

rtext_options affect the rendition of both row titles and number at risk and may be any of the following:

    size(*textsizestyle*) specifies the size of text.

    color(*colorstyle*) specifies the color of text.

    justification(*justificationstyle*) specifies how text elements are to be justified.

    at(#) allows you to reposition row titles or the overall table title to align with a specific location on the x axis.

    topgap(*size*) specifies how much space is to be placed above each row.

    style(*textstyle*) specifies the style of text. This option does not appear on the dialog box.

Title

title([*title*][, *ttext_options*]) may be used to override the default title for the at-risk table and affect the rendition of its text.

Titles may include one line of text or multiple lines. title("At-risk table") will produce a one-line title, and title("At-risk" "table") will produce a multiple-line title.

ttext_options affect the rendition of both row titles and number at risk and may be any of the following:

    size(*textsizestyle*) specifies the size of text.

    color(*colorstyle*) specifies the color of text.

    justification(*justificationstyle*) specifies how text elements are to be justified.

at(*#*) allows you to reposition row titles or the overall table title to align with a specific location on the x axis.

at(rowtitles) places the overall table title at the default position calculated for the row titles. This option is sometimes useful for alignment when the default justification has not been used.

topgap(*size*) specifies how much space is to be placed above each row.

bottomgap(*size*) specifies how much space is to be placed beneath each row.

style(*textstyle*) specifies the style of text. This option does not appear on the dialog box.

Options

level(*#*) specifies the confidence level, as a percentage, for the pointwise confidence interval around the survivor, failure, or cumulative hazard function; see [U] **20.8 Specifying the width of confidence intervals**.

per(*#*) specifies the units used to report the survival or failure rates. For example, if the analysis time is in years, specifying per(100) results in rates per 100 person-years.

noshow prevents sts graph from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

tmax(*#*) specifies that the plotted curve be graphed only for $t \leq \#$. This option does not affect the calculation of the function, rather the portion that is displayed. This option affects the x-axis labels the same way as option risktable() does, by adding the specified value to the graph's default x-axis labels.

tmin(*#*) specifies that the plotted curve be graphed only for $t \geq \#$. This option does not affect the calculation of the function, rather the portion that is displayed. This option affects the x-axis labels the same way as option risktable() does, by adding the specified value to the graph's default x-axis labels.

noorigin requests that the plot of the survival or failure curve begin at the first exit time instead of beginning at $t = 0$ (the default). This option is ignored when cumhaz or hazard is specified.

width(*# [#...]*) is for use with hazard and specifies the bandwidth to be used in the kernel smooth used to plot the estimated hazard function. If width() is not specified, a default bandwidth is used as described in [R] **kdensity**. If it is used with by(), multiple bandwidths may be specified, one for each group. If there are more groups than the k bandwidths specified, the default bandwidth is used for the $k + 1, \ldots$ remaining groups. If any bandwidth is specified as . (dot), the default bandwidth is used for that group.

kernel(*kernel*) is for use with hazard and specifies the kernel function to be used in calculating the weighted kernel-density estimate required to produce a smoothed hazard-function estimator. The default kernel is Epanechnikov, yet *kernel* may be any of the kernels supported by kdensity; see [R] **kdensity**.

noboundary is for use with hazard. It specifies that no boundary-bias adjustments are to be made when calculating the smoothed hazard-function estimator. By default, the smoothed hazards are adjusted near the boundaries. If the epan2, biweight, or rectangular kernel is used, the bias correction near the boundary is performed using boundary kernels. For other kernels, the plotted range of the smoothed hazard function is restricted to be within one bandwidth of each endpoint. For these other kernels, specifying noboundary merely removes this range restriction.

lost specifies that the number lost be shown on the plot. This number is shown in a small size above each flat part of the plotted function.

If enter is not specified, then the number displayed is the number censored minus the number who enter. If enter is specified, then the number displayed is the pure number censored. The underlying logic is described in [ST] **sts**.

lost may not be used with hazard.

enter specifies that the number who enter be shown on the plot, as well as the number lost. The number who enter is shown in a small size beneath each flat part of the plotted function.

enter may not be used with hazard.

atrisk specifies that the number at risk at the beginning of each interval be shown on the plot. This number is shown in a small size beneath each flat part of the plotted function.

atrisk may not be used with hazard.

censored(single | number | multiple) specifies that hash marks be placed on the graph to indicate censored observations.

censored(single) places one hash mark at each censoring time, regardless of the number of censorings at that time.

censored(number) places one hash mark at each censoring time and displays the number of censorings about the hash mark.

censored(multiple) places multiple hash marks for multiple censorings at the same time. For instance, if 3 observations are censored at time 5, three hash marks are placed at time 5. censored(multiple) is intended for use when there are few censored observations; if there are too many censored observations, the graph can look bad. In such cases, we recommend that censored(number) be used.

censored() may not be used with hazard.

censopts(*hash_options*) specifies options that affect how the hash marks for censored observations are rendered; see [G-3] *line_options*. When combined with censored(number), censopts() also specifies how the count of censoring is rendered; see [G-3] *marker_label_options*, except mlabel() is not allowed.

lostopts(*marker_label_options*) specifies options that affect how the numbers lost are rendered; see [G-3] *marker_label_options*. This option implies the lost option.

atriskopts(*marker_label_options*) specifies options that affect how the numbers at risk are rendered; see [G-3] *marker_label_options*. This option implies the atrisk option.

___Plot_____

plotopts(*cline_options*) affects the rendition of the plotted lines; see [G-3] *cline_options*. This option may not be combined with by(*varlist*) or strata(*varlist*), unless separate is also specified.

plot#opts(*cline_options*) affects the rendition of the #th plotted line; see [G-3] *cline_options*. This option may not be combined with separate.

___CI plot_____

ciopts(*area_options*) affects the rendition of the confidence bands; see [G-3] *area_options*. This option may not be combined with by(*varlist*) or strata(*varlist*), unless separate is also specified.

ci#opts(*area_options*) affects the rendition of the #th confidence band; see [G-3] ***area_options***. This option may not be combined with separate.

Add plots
---

addplot(*plot*) provides a way to add other plots to the generated graph; see [G-3] ***addplot_option***.

Y axis, X axis, Titles, Legend, Overall
---

twoway_options are any of the options documented in [G-3] ***twoway_options***. These include options for titling the graph (see [G-3] ***title_options***) and for saving the graph to disk (see [G-3] ***saving_option***).

byopts(*byopts*) affects the appearance of the combined graph when by() or adjustfor() is specified, including the overall graph title and the organization of subgraphs. byopts() may not be specified with separate. See [G-3] ***by_option***.

Remarks and examples

Remarks are presented under the following headings:

> *Including the number lost on the graph*
> *Graphing the Nelson–Aalen cumulative hazard function*
> *Graphing the hazard function*
> *Adding an at-risk table*
> *On boundary bias for smoothed hazards*
> *Video example*

If you have not read [ST] **sts**, please do so.

By default, sts graph displays the Kaplan–Meier product-limit estimate of the survivor or failure function. Only one of sts graph's options, adjustfor(), modifies the calculation. All the other options merely determine how the results of the calculation are graphed.

We demonstrate many of sts graph's features in [ST] **sts**. This discussion picks up where that entry leaves off.

See Cefalu (2011) for covariate-adjusted estimates and confidence intervals.

Including the number lost on the graph

In *Covariate-adjusted estimates* in [ST] **sts**, we introduced a simple drug-trial dataset with 1 observation per subject. Here is a graph of the survivor functions, by drug, including the number lost because of censoring:

```
. use https://www.stata-press.com/data/r18/drug2
(Patient survival in drug trial)

. sts graph, by(drug) lost
        Failure _d: died
  Analysis time _t: studytime
```
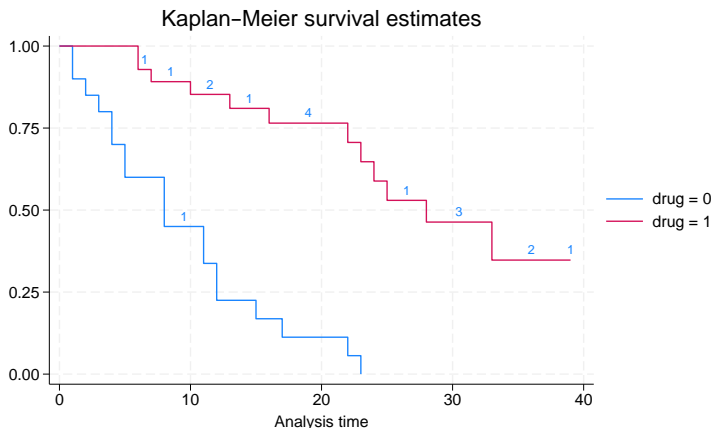

There is no late entry in these data, so we modify the data so that a few subjects entered late. Here is the same graph on the modified data:

```
. use https://www.stata-press.com/data/r18/drug2b
(Patient survival in drug trial)

. sts graph, by(drug) lost
        Failure _d: died
  Analysis time _t: studytime
```
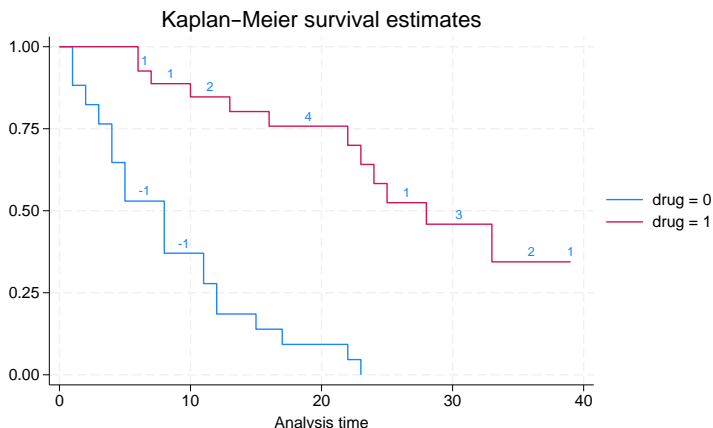
Note the negative numbers. These occur because, by default, `lost` means censored minus entered. Here -1 means that 1 entered, or 2 entered and 1 was lost, etc. If we specify the `enter` option, we will see the censored and entered separately:

```
. sts graph, by(drug) lost enter
         Failure _d: died
   Analysis time _t: studytime
```
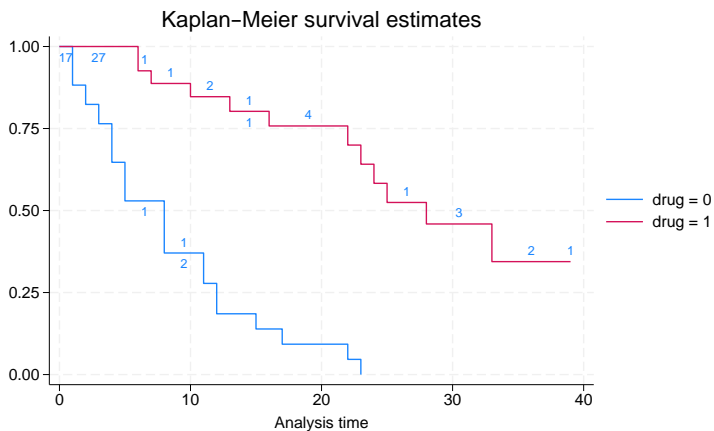

Although it might appear that specifying `enter` with `lost` is a good idea, that is not always true.

We have yet another version of the data—the correct data not adjusted to have late entry—but in this version we have multiple records per subject. The data are the same, but where there was one record in the first dataset, sometimes there are now two because we have a covariate that is changing over time. From this dataset, here is the graph with the number lost shown:

```
. use https://www.stata-press.com/data/r18/drug2c
(Patient survival in drug trial)
. sts graph, by(drug) lost
        Failure _d: died
  Analysis time _t: studytime
        ID variable: id
```
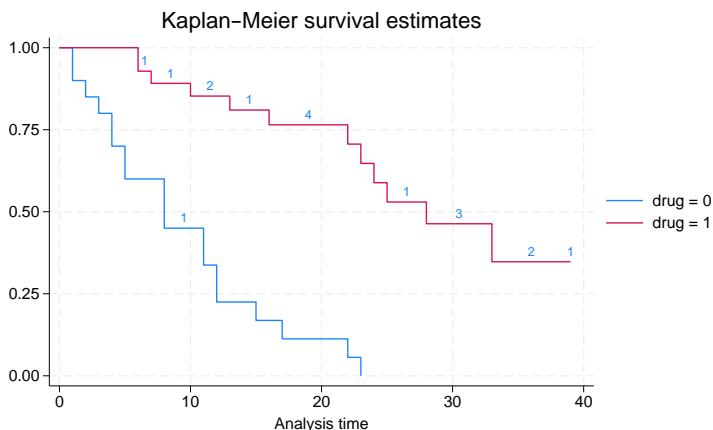

This looks just like the first graph we presented, as indeed it should. Again we emphasize that the data are logically, if not physically, equivalent. If, however, we graph the number lost and entered, we get a graph showing a lot of activity:

```
. sts graph, by(drug) lost enter
        Failure _d: died
  Analysis time _t: studytime
        ID variable: id
```
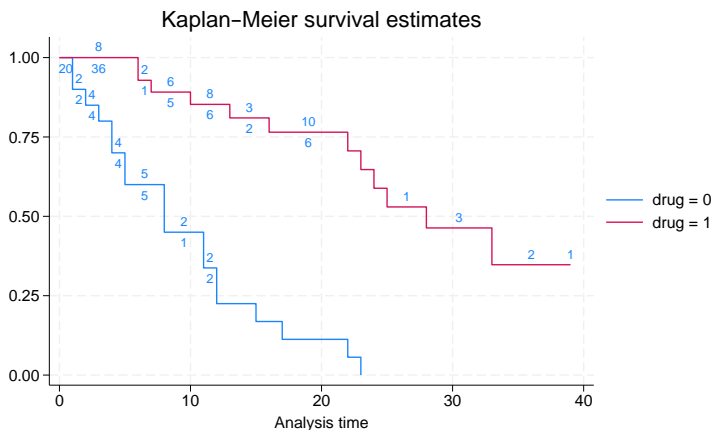
All of that activity goes by the name *thrashing*—subjects are being censored to enter the data again, but with different covariates. This graph was better when we did not specify `enter` because the censored-minus-entered calculation smoothed out the thrashing.

Graphing the Nelson–Aalen cumulative hazard function

We can plot the Nelson–Aalen estimate of the cumulative (integrated) hazard function by specifying the `cumhaz` option. For example, from the 1-observation-per-subject drug-trial dataset, here is a graph of the cumulative hazard functions by drug:

```
. use https://www.stata-press.com/data/r18/drug2
(Patient survival in drug trial)

. stset, noshow

. sts graph, cumhaz by(drug)
```
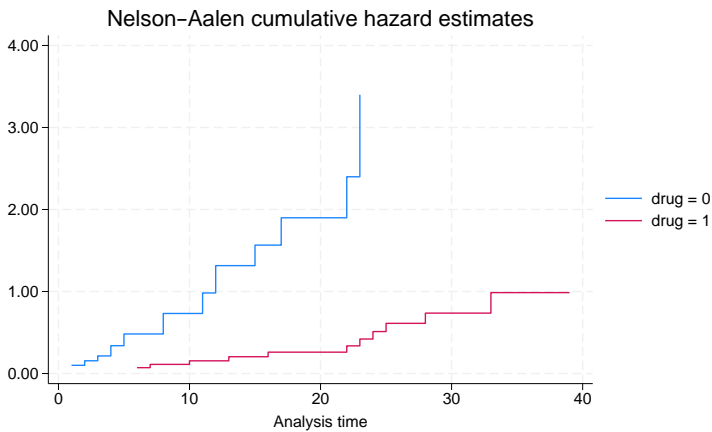

And here is a plot including the number lost because of censoring:

```
. sts graph, cumhaz by(drug) lost
```
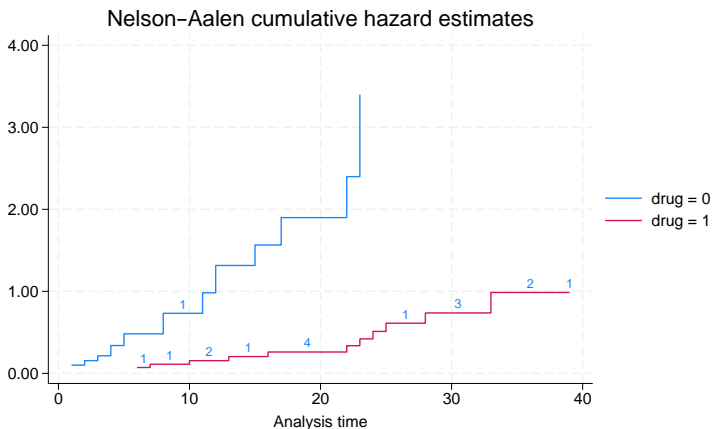
Graphing the hazard function

sts graph may also be used to plot an estimate of the hazard function. This graph is based on a weighted kernel smooth of the estimated hazard contributions, $\Delta\widehat{H}(t_j) = \widehat{H}(t_j) - \widehat{H}(t_{j-1})$, obtained by sts generate *newvar* = h. There are thus issues associated with selecting a kernel function and a bandwidth, although sts graph will use defaults if we do not want to worry about this.

```
. sts graph, hazard by(drug)
```


Smoothed hazard estimates
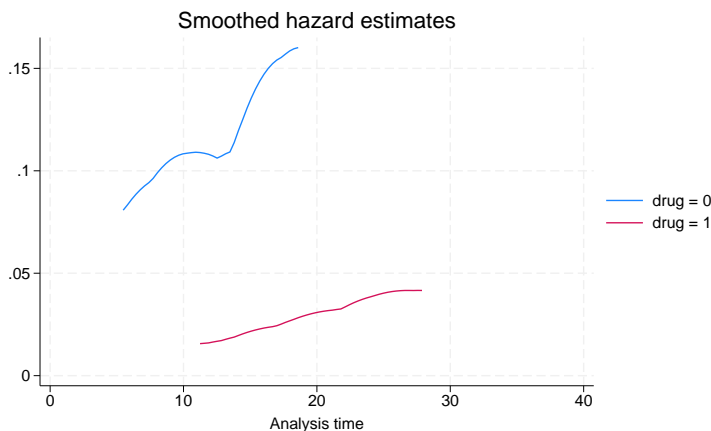
We can also adjust and customize the kernel smooth.

```
. sts graph, hazard by(drug) kernel(gauss) width(5 7)
> title(Comparison of hazard functions)
```


Comparison of hazard functions
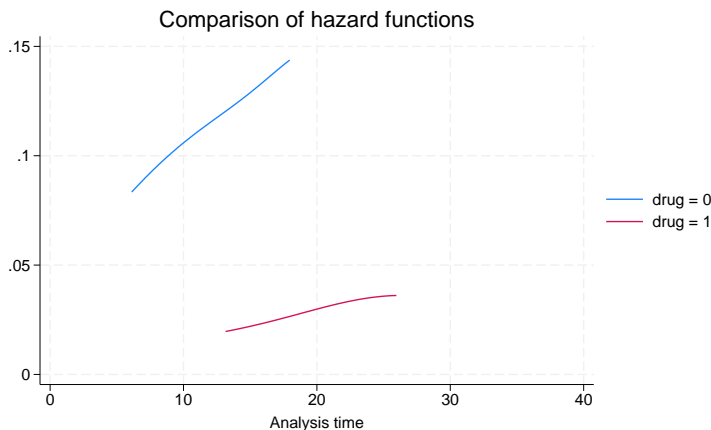
Adding an at-risk table

A table showing the number at risk may be added beneath a survivor, failure, or Nelson–Aalen cumulative hazard plot.

```
. sts graph, by(drug) risktable
```
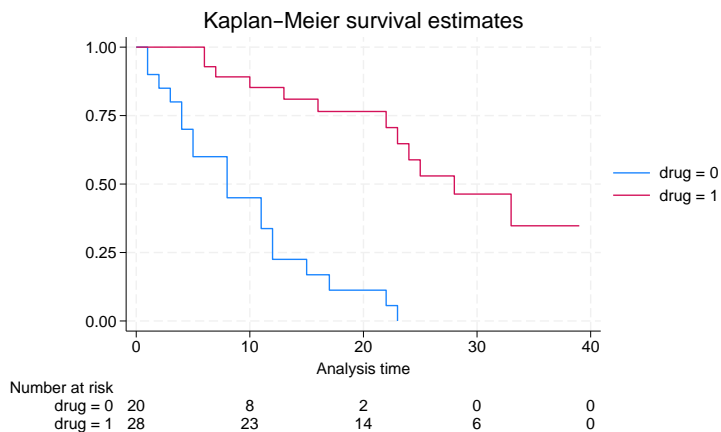

For graphical purposes, `sts graph` reports the number at risk at time 0 as the number of subjects who entered at time 0 instead of an actual zero.

By default, both the legend and the at-risk table share space at the bottom of the graph. Placing the legend in an empty area inside the plot may often be desirable.

```
. sts graph, by(drug) risktable
> legend(ring(0) position(2) rows(2) region(lcolor(black)))
```
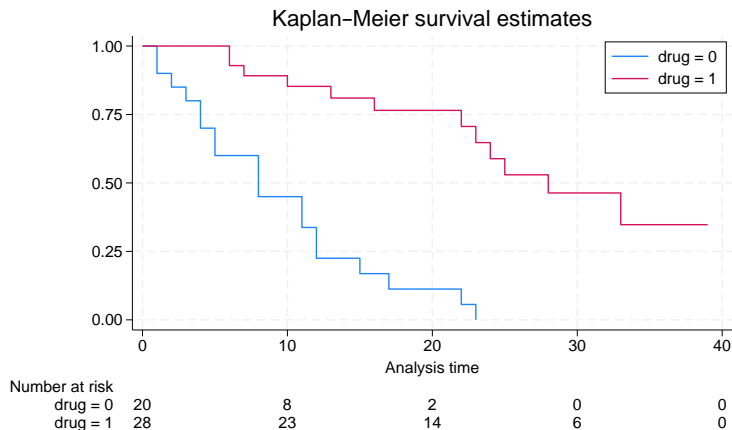

By default, row titles are placed on the left of the at-risk table and are right-justified. We can illustrate this by changing the text of the row titles to have an unequal length.

```
. sts graph, by(drug) risktable(, order(1 "Placebo" 2 "Test drug"))
```
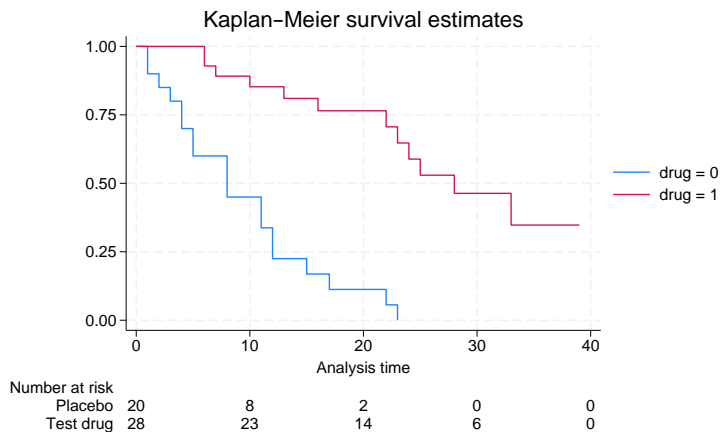

If desired, the text of row titles can be left-justified.

```
. sts graph, by(drug) risktable(, order(1 "Placebo" 2 "Test drug")
> rowtitle(, justification(left)))
```
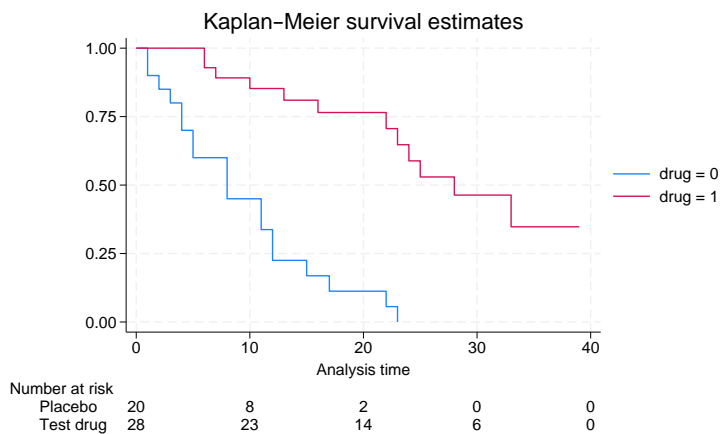

In addition to left justification, the table title can be aligned with the row titles.

```
. sts graph, by(drug) risktable(, order(1 "Placebo" 2 "Test drug")
> rowtitle(, justification(left)) title(, at(rowtitle)))
```
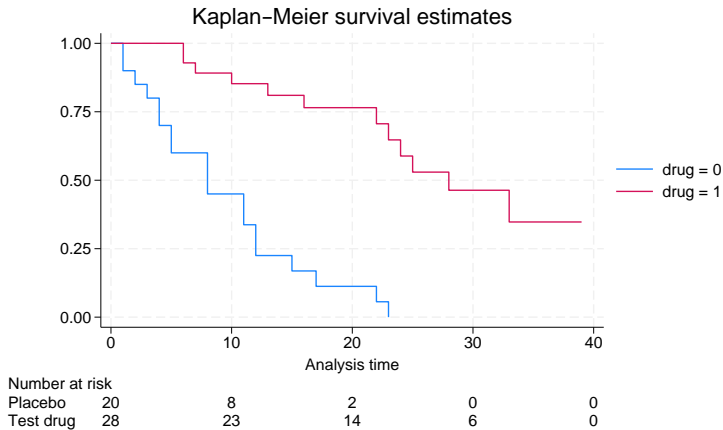

On boundary bias for smoothed hazards

sts graph uses the usual smoothing kernel technique to estimate the hazard function. Kernel estimators commonly encounter bias when estimating near the boundaries of the data range, and therefore estimates of the hazard function in the boundary regions are generally less reliable. To alleviate this problem, estimates that use the epan2, biweight, and rectangular kernels are adjusted at the boundaries with what are known as *boundary kernels* (for example, Müller and Wang [1994]; Hess, Serachitopol, and Brown [1999]). For estimates using other kernels, no boundary adjustment is made. Instead, the default graphing range is constrained to be the range $[L + b, R - b]$, where L and R are the respective minimum and maximum analysis times at which failure occurred and b is the bandwidth.

Video example

How to graph survival curves

Methods and formulas

See *Methods and formulas* in [ST] **sts** for detailed information.

Smoothed hazard estimate

The estimated hazard is calculated as a kernel smooth of the estimated hazard contributions, $\Delta\widehat{H}(t_j) = \widehat{H}(t_j) - \widehat{H}(t_{j-1})$, using

$$\widehat{h}(t) = b^{-1} \sum_{j=1}^{D} K_t \left(\frac{t - t_j}{b} \right) \Delta\widehat{H}(t_j)$$

where $K_t(\ )$ is the kernel (Müller and Wang 1994) function, b is the bandwidth, and the summation is over the D times at which failure occurs (Klein and Moeschberger 2003, 167). If `adjustfor()` is specified, the $\Delta\widehat{H}(t_j)$ are instead obtained from `stcox` as the estimated baseline contributions from a Cox model; see [ST] **stcox** for details on how the $\Delta\widehat{H}(t_j)$ are calculated in this case.

Pointwise confidence bands for smoothed hazard functions are calculated using the method based on a log transformation,

$$\widehat{h}(t)\exp\left[\pm\frac{Z_{1-\alpha/2}\sigma\{\widehat{h}(t)\}}{\widehat{h}(t)}\right]$$

See Klein and Moeschberger (2003, 168) for details.

References

Cefalu, M. S. 2011. Pointwise confidence intervals for the covariate-adjusted survivor function in the Cox model. *Stata Journal* 11: 64–81.

Hess, K. R., D. M. Serachitopol, and B. W. Brown. 1999. Hazard function estimators: A simulation study. *Statistics in Medicine* 18: 3075–3088. https://doi.org/10.1002/(SICI)1097-0258(19991130)18:22⟨3075::AID-SIM244⟩3.0.CO;2-6.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data.* 2nd ed. New York: Springer.

Müller, H.-G., and J.-L. Wang. 1994. Hazard rate estimation under random censoring with varying kernels and bandwidths. *Biometrics* 50: 61–76. https://doi.org/10.2307/2533197.

Also see [ST] **sts** for more references.

Also see

Title

> **sts list —** List the survivor or related function

Description   Quick start   Menu   Syntax
Options   Remarks and examples   Methods and formulas   References
Also see

Description

sts list lists the estimated survivor, failure, or the Nelson–Aalen estimated cumulative (integrated) hazard function. See [ST] sts for an introduction to this command.

sts list can be used with single- or multiple-record or single- or multiple-failure st data.

Quick start

Kaplan–Meier survivor function using stset data
    sts list

Show the number at risk and the survivor function only at specified times
    sts list, risktable(10 20 30 40 50)

Survivor function for each level of v1
    sts list, by(v1)

Results of above saved in mydata.dta
    sts list, by(v1) saving(mydata)

Show numbers at risk and survivor functions for groups of v1 only at specified times
    sts list, by(v1) risktable(10(10)50)

Same as above, but report survivor functions for each group side by side
    sts list, by(v1) risktable(10(10)50) compare

Failure function
    sts list, failure

Failure function adjusted for $v2 = 0$
    sts list, failure adjustfor(v2)

Same as above, but adjusted for mean value of v2
    sts list, failure adjustfor(v2, atmeans)

Failure function adjusted for $v2 = 10$ and stratified on levels of svar
    sts list, failure adjustfor(v2, at(v2=10)) strata(svar)

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > List the survivor or related function

Syntax

> sts <u>l</u>ist $[\textit{if}]$ $[\textit{in}]$ [, $\textit{options}$]

options	Description
[Main]	
<u>surv</u>ival	report Kaplan–Meier survivor function; the default
<u>f</u>ailure	report Kaplan–Meier failure function
<u>cumh</u>az	report Nelson–Aalen cumulative hazard function
by(*varlist*)	estimate separate functions for each group formed by *varlist*
strata(*varlist*)	stratify on different groups of *varlist*; requires adjustfor()
<u>adj</u>ustfor(*varlist* [, *suboptions*])	adjust the estimates to specific values of *varlist*; default is zero values
[Options]	
<u>l</u>evel(*#*)	set confidence level; default is level(95)
<u>risk</u>table	report estimated survivor or related function at all distinct observed times; the default
<u>risk</u>table(*numlist*)	report estimated survivor or related function at specified times
tmax(*#*)	display estimated function for $t \leq \#$
tmin(*#*)	display estimated function for $t \geq \#$
<u>ent</u>er	report number who entered and number lost as pure censored instead of censored minus lost (net lost)
<u>nosh</u>ow	do not show st setting information
<u>compare</u>	report groups of estimated functions side by side; requires by() or strata()
<u>savi</u>ng(*filename* [, replace])	save results to *filename*; use replace to overwrite existing *filename*
atrisk0	report number who entered as number at risk at time 0 instead of a zero

You must stset your data before using sts list; see [ST] **stset**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

atrisk0 does not appear in the dialog box.

Options

> [Main]

survival, failure, and cumhaz specify the function to report.

> survival specifies that the Kaplan–Meier survivor function be listed. This option is the default if a function is not specified.

> failure specifies that the Kaplan–Meier failure function $1 - S(t + 0)$ be listed.

> cumhaz specifies that the Nelson–Aalen estimate of the cumulative hazard function be listed.

by(*varlist*) estimates a separate function for each by-group. By-groups are identified by equal values of the variables in *varlist*. by() may not be combined with strata().

strata(*varlist*) requests estimates of the survivor or failure function stratified on variables in *varlist*. It requires specifying adjustfor() and may not be combined with by() or cumhaz.

adjustfor(*varlist*[, *suboptions*]) adjusts the estimate of the survivor or failure function to specific values of *varlist*. The default is to adjust to 0 values, that is, to produce a baseline function. If you want to adjust the function to values different from 0, you can use adjustfor()'s at() suboption.

suboptions are atzeros (the default), atmeans, atomeans, atbase, and at(); see [ST] *adjustfor_option*.

Say you want to produce the survivor function adjusted to age of patients and the ages in your sample are 40–60 years. If you want to adjust the function to age 40, type

    . sts list, adjustfor(age, at(age=40))

Or if you want to adjust the function to the mean age, type

    . sts list, adjustfor(age, atmeans)

adjustfor() is not available with cumhaz.

If you specify adjustfor() with by(), sts list fits separate Cox regression models for each group, using the adjustfor() variables as covariates. The separately calculated baseline functions are then retrieved.

If you specify adjustfor() with strata(), sts list fits a stratified-on-group Cox regression model, using the adjustfor() variables as covariates. The stratified baseline function is then retrieved.

If adjustment to covariate values other than 0 is requested, the function is estimated at the specified covariate values.

───┐ Options ┌──

level(*#*) specifies the confidence level, as a percentage, for the Greenwood pointwise confidence interval of the survivor or failure function or for the pointwise confidence interval of the Nelson–Aalen cumulative hazard function; see [U] **20.8 Specifying the width of confidence intervals**.

risktable or risktable(*numlist*) reports the estimated survivor, failure, or cumulative hazard function.

risktable, the default, reports the results at all the distinct observed times or, if functions are being compared, at about 10 times chosen over the observed interval. At time 0, the number at risk is zero. At other times, the number at risk records the number of subjects who are under observation and did not fail immediately prior to the reported time.

risktable(*numlist*) reports the results at time points specified in *numlist*. You can think of risktable(*numlist*) as a table representation of sts graph, risktable(*numlist*), except the number of failures is always reported with sts list, and the number at risk at time 0 is not the number of subjects in a study but is a zero. To match sts graph, you can use the atrisk0 option to report the number at risk at time 0 as the number who entered at that time instead of a zero.

risktable(5 10 20 30 50 90) would display the function at the designated times.

risktable(10 20 to 100) would display the function at times 10, 20, 30, 40, . . . , 100.

risktable(0 5 10 to 100 200) would display the function at times 0, 5, 10, 15, . . . , 100, and 200.

risktable(20) would display the function at time 20.

See example 1 under *Remarks and examples*.

tmax(#) specifies that the estimated function and other summaries be reported only for $t \leq \#$. This option does not affect the calculation of the function, rather the portion that is displayed, but it does affect the calculation of certain summaries such as the number of failures and the number lost. tmax(#) is ignored when # is smaller than the maximum time specified in risktable(*numlist*).

tmin(#) specifies that the estimated function and other summaries be reported only for $t \geq \#$. This option does not affect the calculation of the function, rather the portion that is displayed, but it does affect the calculation of certain summaries such as the number who entered. tmin(#) is ignored when # is larger than the minimum time specified in risktable(*numlist*).

enter specifies that the table contain the number who entered and, correspondingly, that the number lost be displayed as the pure number censored rather than censored minus entered (net lost). The logic underlying this is explained in *Counting the number lost to censoring* in [ST] **sts**.

noshow prevents sts list from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

compare is specified only with by() or strata(). It compares the survivor, failure, or cumulative hazard functions and lists them side by side rather than first one and then the next.

saving(*filename* [, replace]) saves the results in a Stata data file (.dta file).

replace specifies that *filename* be overwritten if it exists.

The following option is available with sts list but is not shown in the dialog box:

atrisk0 specifies to report the number who entered as the number at risk at time 0 instead of the default zero value. By definition, the number at risk at time 0 is a zero. atrisk0 is useful with risktable(*numlist*) to match the at-risk table as reported by the corresponding sts graph, risktable(*numlist*) specification.

Remarks and examples

sts list reports the estimated survivor, failure, or cumulative hazard function and, in the absence of probability weights, its standard errors and confidence intervals.

The Kaplan–Meier product-limit estimate of the survivor function is displayed by default or if you specify the survival option. Use the failure option to display the Kaplan–Meier product-limit estimate of the failure function. Or use the cumhaz option to display the Nelson–Aalen estimate of the cumulative hazard function.

Only three of sts list's options—adjustfor(), by(), and strata()—modify the calculation of a function. All the other options merely determine how the results of the calculation are displayed.

Specify the by() option to perform the calculation separately on the different groups and the compare option to see the results side by side.

Specify the adjustfor() option to calculate an adjusted survivor or failure function. If you also specify by() or strata(), this further modifies how the adjustment is made. With by(), the function is estimated after fitting separate Cox regression models for each group using the adjustfor() variables as covariates. With strata(), which requires adjustfor(), the function is estimated after fitting a stratified-on-group Cox regression model using the adjustfor() variables as covariates. Only the estimate of the function is reported with adjustfor().

By default, `sts list` displays the estimated function and other summaries such as the number at risk, the number of failures, and the number lost (censored) at all distinct observed times. You can specify the `risktable(`*numlist*`)` option to see the results at specific times. This is recommended for large datasets. In the presence of delayed entry or gaps, the command reports the number lost as net lost—the number of censored minus the number who entered. You can specify the `enter` option to see the number lost and the number who entered separately; see *Counting the number lost to censoring* in [ST] **sts**.

You can also restrict the maximum and minimum ranges of the observed times using the `tmax(`*#*`)` and `tmin(`*#*`)` options. These options do not affect the calculation of the estimated function but will affect the computation of certain summaries such as the number of failures and the number lost and will affect the display of the results; see example 1.

We demonstrate many of `sts list`'s features in [ST] **sts**. This discussion picks up where that entry leaves off.

▷ Example 1: Using option risktable()

By default, `sts list` will bury you in output. With the Stanford heart transplant data introduced in [ST] **stset**, the following commands produce 154 lines of output.

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)

. stset, noshow

. sts list, by(posttran)

Kaplan-Meier survivor function
By variable: posttran
```

Time	At risk	Fail	Net lost	Survivor function	Std. error	[95% conf. int.]	
posttran=0							
1	103	1	3	0.9903	0.0097	0.9331	0.9986
2	99	3	3	0.9603	0.0195	0.8976	0.9849
3	93	3	3	0.9293	0.0258	0.8574	0.9657
(output omitted)							
1400	1	0	1	0.2359	0.1217	0.0545	0.4882
posttran=1							
1	0	0	-3	1.0000	.	.	.
2	3	0	-3	1.0000	.	.	.
(output omitted)							
5.1	14	1	0	0.9286	0.0688	0.5908	0.9896
6	13	0	-1	0.9286	0.0688	0.5908	0.9896
(output omitted)							
1799	1	0	1	0.1420	0.0546	0.0566	0.2653

Note: Net lost equals the number lost minus the number who entered.

`risktable()` is the solution. Here is another detailed, but more useful, view of the heart transplant data:

```
. sts list, risktable(10 40 to 170) by(posttran)
Kaplan–Meier survivor function
By variable: posttran
                At            Net    Survivor       Std.
    Time      risk   Fail    lost    function      error       [95% conf. int.]

posttran=0
      10        74      9      34      0.8724     0.0346     0.7858     0.9256
      40        31      4      11      0.6781     0.0601     0.5446     0.7801
      70        16      1       5      0.6126     0.0704     0.4603     0.7339
     100        10      1       0      0.5616     0.0810     0.3900     0.7022
     130         9      1       1      0.5054     0.0903     0.3199     0.6646
     160         7      2       5      0.4422     0.0986     0.2480     0.6204
posttran=1
      10        16      6     -31      0.9286     0.0688     0.5908     0.9896
      40        41      9     -11      0.7391     0.0900     0.5140     0.8716
      70        43      8      -5      0.6002     0.0841     0.4172     0.7423
     100        40      2       1      0.4814     0.0762     0.3271     0.6198
     130        37      1       0      0.4687     0.0752     0.3174     0.6063
     160        36     18      18      0.4561     0.0742     0.3076     0.5928

Notes: Survival-time summaries shown for specified times in (0, 1799].
       Net lost equals the number lost minus the number who entered.
```

We specified `risktable(10 40 to 170)`, but the time 170 was not displayed because the numlist 10 40 to 170 ends at 160. Specifying the numlist as 10 40 to 160 would have been clearer.

We obtain the same numbers at risk as produced by `sts graph`:

```
. sts graph, risktable(10 40 to 170, failevents lastfailure) by(posttran)
> xlabel(10 40 to 160 170) tmax(170)
```


Kaplan–Meier survival estimates
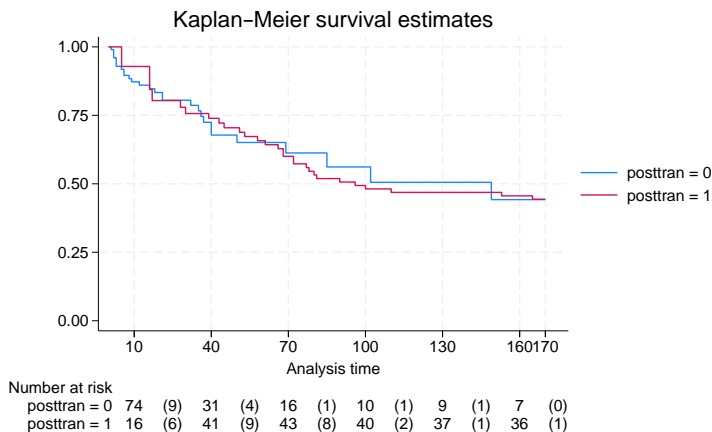
The number of failures for the last interval reported by `sts graph` is 0 in the first group and 1 in the second group, whereas `sts list` reported 2 and 18, respectively. We specified `tmax(170)` with `sts graph` to restrict the range of the observed times to the maximum of 170, whereas `sts list` used the observed maximum times for each group (1,400 and 1,799). If we want to match the numbers of failures, we could specify `tmax(170)` with `sts list`:

```
. sts list, risktable(10 40 to 170) by(posttran) tmax(170)
```

Kaplan–Meier survivor function
By variable: posttran

Time	At risk	Fail	Net lost	Survivor function	Std. error	[95% conf. int.]	
posttran=0							
10	74	9	34	0.8724	0.0346	0.7858	0.9256
40	31	4	11	0.6781	0.0601	0.5446	0.7801
70	16	1	5	0.6126	0.0704	0.4603	0.7339
100	10	1	0	0.5616	0.0810	0.3900	0.7022
130	9	1	1	0.5054	0.0903	0.3199	0.6646
160	7	0	1	0.4422	0.0986	0.2480	0.6204
posttran=1							
10	16	6	-31	0.9286	0.0688	0.5908	0.9896
40	41	9	-11	0.7391	0.0900	0.5140	0.8716
70	43	8	-5	0.6002	0.0841	0.4172	0.7423
100	40	2	1	0.4814	0.0762	0.3271	0.6198
130	37	1	0	0.4687	0.0752	0.3174	0.6063
160	36	1	-1	0.4561	0.0742	0.3076	0.5928

Notes: Survival-time summaries shown for specified times in (0, 170].
       Net lost equals the number lost minus the number who entered.

◁

▷ Example 2: Nelson–Aalen estimated cumulative hazard

Continuing with example 1, we can produce similar output for the Nelson–Aalen estimated cumulative hazard by specifying the cumhaz option:

```
. sts list, cumhaz risktable(10 40 to 170) by(posttran)
```

Nelson–Aalen cumulative hazard function
By variable: posttran

Time	At risk	Fail	Net lost	Cumulative hazard	Std. error	[95% conf. int.]	
posttran=0							
10	74	9	34	0.1349	0.0391	0.0764	0.2382
40	31	4	11	0.3824	0.0871	0.2448	0.5976
70	16	1	5	0.4813	0.1124	0.3044	0.7608
100	10	1	0	0.5646	0.1400	0.3473	0.9178
130	9	1	1	0.6646	0.1720	0.4002	1.1037
160	7	2	5	0.7896	0.2126	0.4658	1.3385
posttran=1							
10	16	6	-31	0.0714	0.0714	0.0101	0.5071
40	41	9	-11	0.2929	0.1176	0.1334	0.6433
70	43	8	-5	0.4981	0.1360	0.2916	0.8507
100	40	2	1	0.7155	0.1542	0.4691	1.0915
130	37	1	0	0.7418	0.1564	0.4908	1.1214
160	36	18	18	0.7689	0.1587	0.5130	1.1523

Notes: Survival-time summaries shown for specified times in (0, 1799].
       Net lost equals the number lost minus the number who entered.

We can use the `compare` option to compare the cumulative hazard functions of the two groups side by side:

```
. sts list, cumhaz risktable(10 40 to 170) by(posttran) compare

Nelson-Aalen cumulative hazard function
By variable: posttran

posttran                0           1

Time       10      0.1349      0.0714
           40      0.3824      0.2929
           70      0.4813      0.4981
          100      0.5646      0.7155
          130      0.6646      0.7418
          160      0.7896      0.7689
```

◁

Video example

How to calculate the Kaplan-Meier survivor and Nelson-Aalen cumulative hazard functions

Methods and formulas

See [ST] **sts**.

References

See [ST] **sts** for references.

Also see

[ST] **sts** — Generate, graph, list, and test the survivor and related functions

[ST] **sts generate** — Create variables containing survivor and related functions

[ST] **sts graph** — Graph the survivor or related function

[ST] **sts test** — Test equality of survivor functions

[ST] **stset** — Declare data to be survival-time data

[ST] *adjustfor_option* — Adjust survivor and related functions for covariates at specific values

Title

sts test — Test equality of survivor functions

Description

sts test tests the equality of survivor functions across two or more groups. The log-rank, Cox, Wilcoxon–Breslow–Gehan, Tarone–Ware, Peto–Peto–Prentice, and Fleming–Harrington tests are provided, in both unstratified and stratified forms.

sts test also provides a test for trend.

sts test can be used with single- or multiple-record or single- or multiple-failure st data.

Quick start

Log-rank test for the equality of survivor functions across levels of v1 using [stset](#) data

    `sts test v1`

Stratified log-rank test for equality of survivor functions across v1 with strata svar

    `sts test v1, strata(svar)`

Same as above, and show tests for each stratum

    `sts test v1, strata(svar) detail`

Log-rank test for equality, and test for a trend in survivor functions for v1

    `sts test v1, trend`

Test equality of survivor functions using the Wilcoxon–Breslow–Gehan test

    `sts test v1, wilcoxon`

Likelihood-ratio test for the equality of survivor functions based on the Cox model

    `sts test v1, cox`

Stratified Cox test of equality of survivor functions with strata svar

    `sts test v1, cox strata(svar)`

Test equality of survivor functions using the Tarone–Ware test

    `sts test v1, tware`

Same as above, and test for a trend using the same weights as used in the Tarone–Ware test

    `sts test v1, tware trend`

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Test equality of survivor functions

Syntax

> sts <u>test</u> *varlist* [*if*] [*in*] [, *options*]

options	Description
Main	
<u>logrank</u>	perform log-rank test of equality; the default
<u>cox</u>	perform Cox test of equality
<u>wilcoxon</u>	perform Wilcoxon–Breslow–Gehan test of equality
<u>tware</u>	perform Tarone–Ware test of equality
<u>peto</u>	perform Peto–Peto–Prentice test of equality
<u>fh</u>(*p q*)	perform generalized Fleming–Harrington test of equality
<u>trend</u>	test trend of the survivor function across three or more ordered groups
<u>strata</u>(*varlist*)	perform stratified test on *varlist*, displaying overall test results
<u>detail</u>	display individual test results; modifies strata()
Options	
<u>mat</u>(*mname₁ mname₂*)	store vector **u** in *mname₁* and matrix **V** in *mname₂*
<u>noshow</u>	do not show st setting information
<u>notitle</u>	suppress title

You must stset your data before using sts test; see [ST] **stset**.

collect is allowed; see [U] **11.1.10 Prefix commands**.

Note that fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

Options

> [Main]

logrank, cox, wilcoxon, tware, peto, and fh(*p q*) specify the test of equality desired. logrank is the default, unless the data are pweighted, in which case cox is the default and is the only possibility.

> wilcoxon specifies the Wilcoxon–Breslow–Gehan test; tware, the Tarone–Ware test; peto, the Peto–Peto–Prentice test; and fh(), the generalized Fleming–Harrington test. The Fleming–Harrington test requires two arguments, p and q. When $p = 0$ and $q = 0$, the Fleming–Harrington test reduces to the log-rank test; when $p = 1$ and $q = 0$, the test reduces to the Mann–Whitney–Wilcoxon test.

trend specifies that a test for trend of the survivor function across three or more ordered groups be performed.

strata(*varlist*) requests that a stratified test be performed.

detail modifies strata(); it requests that, in addition to the overall stratified test, the tests for the individual strata be reported. detail is not allowed with cox.

> [Options]

mat(*mname₁ mname₂*) requests that the vector **u** be stored in *mname₁* and that matrix **V** be stored in *mname₂*. The other tests are rank tests of the form $\mathbf{u}'\mathbf{V}^{-1}\mathbf{u}$. This option may not be used with cox.

noshow prevents sts test from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

notitle requests that the title printed above the test be suppressed.

Remarks and examples

Remarks are presented under the following headings:

sts test tests the equality of the survivor function across groups. With the exception of the Cox test, these tests are members of a family of statistical tests that are extensions to censored data of traditional nonparametric rank tests for comparing two or more distributions. A technical description of these tests can be found in the *Methods and formulas* section of this entry. Simply, at each distinct failure time in the data, the contribution to the test statistic is obtained as a weighted standardized sum of the difference between the observed and expected number of deaths in each of the k groups. The expected number of deaths is obtained under the null hypothesis of no differences between the survival experience of the k groups.

The weights or weight function used determines the test statistic. For example, when the weight is 1 at all failure times, the log-rank test is computed, and when the weight is the number of subjects at risk of failure at each distinct failure time, the Wilcoxon–Breslow–Gehan test is computed.

The following table summarizes the weights used for each statistical test.

Test	Weight at each distinct failure time (t_i)
Log-rank	1
Wilcoxon–Breslow–Gehan	n_i
Tarone–Ware	$\sqrt{n_i}$
Peto–Peto–Prentice	$\widetilde{S}(t_i)$
Fleming–Harrington	$\widehat{S}(t_{i-1})^p\{1 - \widehat{S}(t_{i-1})\}^q$

where $\widehat{S}(t_i)$ is the estimated Kaplan–Meier survivor-function value for the combined sample at failure time t_i, $\widetilde{S}(t_i)$ is a modified estimate of the overall survivor function described in *Methods and formulas*, and n_i is the number of subjects in the risk pool at failure time t_i.

These tests are appropriate for testing the equality of survivor functions across two or more groups. Up to 800 groups are allowed.

The "Cox" test is related to the log-rank test but is performed as a likelihood-ratio test (or, alternatively, as a Wald test) on the results from a Cox proportional hazards regression. The log-rank test should be preferable to what we have labeled the Cox test, but with pweighted data the log-rank test is not appropriate. Whether you perform the log-rank or Cox test makes little substantive difference with most datasets.

`sts test, trend` can be used to test against the alternative hypothesis that the failure rate increases or decreases as the level of the *k* groups increases or decreases. This test is appropriate only when there is a natural ordering of the comparison groups, for example, when each group represents an increasing or decreasing level of a therapeutic agent.

`trend` is not valid when `cox` is specified.

The log-rank test

`sts test`, by default, performs the log-rank test, which is, to be clear, the exponential scores test (Savage 1956; Mantel and Haenszel 1959; Mantel 1963, 1966). This test is most appropriate when the hazard functions are thought to be proportional across the groups if they are not equal.

This test statistic is constructed by giving equal weights to the contribution of each failure time to the overall test statistic.

In *Testing equality of survivor functions* in [ST] **sts**, we demonstrated the use of this command with the heart transplant data, a multiple-record, single-failure st dataset.

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)

. sts test posttran

        Failure _d: died
  Analysis time _t: t1
        ID variable: id

Equality of survivor functions
Log-rank test
```

posttran	Observed events	Expected events
0	30	31.20
1	45	43.80
Total	75	75.00

```
                    chi2(1) =   0.13
                    Pr>chi2 = 0.7225
```

We cannot reject the hypothesis that the survivor functions are the same.

`sts test, logrank` can also perform the stratified log-rank test. Say that it is suggested that calendar year of acceptance also affects survival and that there are three important periods: 1967–1969, 1970–1972, and 1973–1974. Therefore, a stratified test should be performed:

```
. stset, noshow

. generate group = 1 if year <= 69
(117 missing values generated)

. replace group=2 if year>=70 & year<=72
(78 real changes made)

. replace group=3 if year>=73
(39 real changes made)
```

```
. sts test posttran, strata(group)
Equality of survivor functions
Stratified log-rank test
Strata: group
```

posttran	Observed events	Expected events*
0	30	31.51
1	45	43.49
Total	75	75.00

```
* aggregated over stratum-specific results
                            chi2(1) =    0.20
                            Pr>chi2 = 0.6547
```

When a stratified test is requested, Stata computes the expected events under the null hypothesis separately for each stratum, and those results are aggregated over strata.

We now want to show the within-stratum tests, so we type

```
. sts test posttran, strata(group) detail
Equality of survivor functions
Stratified log-rank test
Strata: group
-> group = 1
```

posttran	Observed events	Expected events
0	14	13.59
1	17	17.41
Total	31	31.00

```
                            chi2(1) =    0.03
                            Pr>chi2 = 0.8558
-> group = 2
```

posttran	Observed events	Expected events
0	13	13.63
1	20	19.37
Total	33	33.00

```
                            chi2(1) =    0.09
                            Pr>chi2 = 0.7663
-> group = 3
```

posttran	Observed events	Expected events
0	3	4.29
1	8	6.71
Total	11	11.00

```
                            chi2(1) =    0.91
                            Pr>chi2 = 0.3410
```

```
-> Total
```

posttran	Observed events	Expected events*
0	30	31.51
1	45	43.49
Total	75	75.00

* aggregated over stratum-specific results

```
              chi2(1) =   0.20
              Pr>chi2 = 0.6547
```

The Wilcoxon (Breslow–Gehan) test

`sts test, wilcoxon` performs the generalized Wilcoxon test of Breslow (1970) and Gehan (1965). This test is appropriate when hazard functions are thought to vary in ways other than proportionally and when censoring patterns are similar across groups.

The Wilcoxon test statistic is constructed by weighting the contribution of each failure time to the overall test statistic by the number of subjects at risk. Thus it gives heavier weights to earlier failure times when the number at risk is higher. As a result, this test is susceptible to differences in the censoring pattern of the groups.

`sts test, wilcoxon` works the same way as `sts test, logrank`:

```
. sts test posttran, wilcoxon
Equality of survivor functions
Wilcoxon-Breslow-Gehan test
```

posttran	Observed events	Expected events	Sum of ranks
0	30	31.20	-85
1	45	43.80	85
Total	75	75.00	0

```
              chi2(1) =   0.14
              Pr>chi2 = 0.7083
```

With the `strata()` option, `sts test, wilcoxon` performs the stratified test:

```
. sts test posttran, wilcoxon strata(group)
Equality of survivor functions
Stratified Wilcoxon-Breslow-Gehan test
Strata: group
```

posttran	Observed events	Expected events*	Sum of ranks*
0	30	31.51	-40
1	45	43.49	40
Total	75	75.00	0

* aggregated over stratum-specific results

```
              chi2(1) =   0.22
              Pr>chi2 = 0.6385
```

As with `sts test, logrank`, you can also specify the `detail` option to see the within-stratum tests.

The Tarone–Ware test

sts test, tware performs a test suggested by Tarone and Ware (1977), with weights equal to the square root of the number of subjects in the risk pool at time t_i.

Like Wilcoxon's test, this test is appropriate when hazard functions are thought to vary in ways other than proportionally and when censoring patterns are similar across groups. The test statistic is constructed by weighting the contribution of each failure time to the overall test statistic by the square root of the number of subjects at risk. Thus, like the Wilcoxon test, it gives heavier weights, although not as large, to earlier failure times. Although less susceptible to the failure and censoring pattern in the data than Wilcoxon's test, this could remain a problem if large differences in these patterns exist between groups.

sts test, tware works the same way as sts test, logrank:

```
. sts test posttran, tware
Equality of survivor functions
Tarone-Ware test
```

	Observed	Expected	Sum of
posttran	events	events	ranks
0	30	31.20	-9.3375685
1	45	43.80	9.3375685
Total	75	75.00	0

```
                            chi2(1) =   0.12
                            Pr>chi2 = 0.7293
```

With the strata() option, sts test, tware performs the stratified test:

```
. sts test posttran, tware strata(group)
Equality of survivor functions
Stratified Tarone-Ware test
Strata: group
```

	Observed	Expected	Sum of
posttran	events	events*	ranks*
0	30	31.51	-7.4679345
1	45	43.49	7.4679345
Total	75	75.00	0

```
* aggregated over stratum-specific results
                            chi2(1) =   0.21
                            Pr>chi2 = 0.6464
```

As with sts test, logrank, you can also specify the detail option to see the within-stratum tests.

The Peto–Peto–Prentice test

sts test, peto performs an alternative to the Wilcoxon test proposed by Peto and Peto (1972) and Prentice (1978). The test uses as the weight function an estimate of the overall survivor function, which is similar to that obtained using the Kaplan–Meier estimator. See *Methods and formulas* for details.

This test is appropriate when hazard functions are thought to vary in ways other than proportionally, but unlike the Wilcoxon–Breslow–Gehan test, it is not affected by differences in censoring patterns across groups.

`sts test, peto` works the same way as `sts test, logrank`:

```
. sts test posttran, peto
Equality of survivor functions
Peto-Peto-Prentice test
```

	Observed	Expected	Sum of
posttran	events	events	ranks
0	30	31.20	-.86708453
1	45	43.80	.86708453
Total	75	75.00	0

```
                            chi2(1) =    0.15
                            Pr>chi2 = 0.6979
```

With the `strata()` option, `sts test, peto` performs the stratified test:

```
. sts test posttran, peto strata(group)
Equality of survivor functions
Stratified Peto-Peto-Prentice test
Strata: group
```

	Observed	Expected	Sum of
posttran	events	events*	ranks*
0	30	31.51	-1.4212233
1	45	43.49	1.4212233
Total	75	75.00	0

```
* aggregated over stratum-specific results
                            chi2(1) =    0.43
                            Pr>chi2 = 0.5129
```

As with the previous tests, you can also specify the `detail` option to see the within-stratum tests.

The generalized Fleming–Harrington tests

`sts test, fh(p q)` performs the Harrington and Fleming (1982) class of test statistics. The weight function at each distinct failure time, t, is the product of the Kaplan–Meier survivor estimate at time $t-1$ raised to the p power and $1-$ the Kaplan–Meier survivor estimate at time $t-1$ raised to the q power. Thus, when specifying the Fleming and Harrington option, you must specify two nonnegative arguments, p and q.

When $p > q$, the test gives more weights to earlier failures than to later ones. When $p < q$, the opposite is true, and more weight is given to later failures than to earlier ones. When p and q are both zero, the weight is 1 at all failure times and the test reduces to the log-rank test.

sts test, fh(p q) works the same way as sts test, logrank. If we specify $p = 0$ and $q = 0$ we will get the same results as the log-rank test:

```
. sts test posttran, fh(0 0)
Equality of survivor functions
Fleming–Harrington test
Powers: p = 0, q = 0
```

posttran	Observed events	Expected events	Sum of ranks
0	30	31.20	-1.1995511
1	45	43.80	1.1995511
Total	75	75.00	0

```
                         chi2(1) =    0.13
                         Pr>chi2 = 0.7225
```

We could, for example, give more weight to later failures than to earlier ones.

```
. sts test posttran, fh(0 3)
Equality of survivor functions
Fleming–Harrington test
Powers: p = 0, q = 3
```

posttran	Observed events	Expected events	Sum of ranks
0	30	31.20	-.10288411
1	45	43.80	.10288411
Total	75	75.00	0

```
                         chi2(1) =    0.01
                         Pr>chi2 = 0.9065
```

Similarly to the previous tests, with the strata() option, sts test, fh() performs the stratified test:

```
. sts test posttran, fh(0 3) strata(group)
Equality of survivor functions
Stratified Fleming–Harrington test
Powers: p = 0, q = 3
Strata: group
```

posttran	Observed events	Expected events*	Sum of ranks*
0	30	31.51	-.05315105
1	45	43.49	.05315105
Total	75	75.00	0

* aggregated over stratum-specific results

```
                         chi2(1) =    0.00
                         Pr>chi2 = 0.9494
```

As with the other tests, you can also specify the detail option to see the within-stratum tests.

The "Cox" test

The term *Cox test* is our own, and this test is a variation on the log-rank test using Cox regression.

One way of thinking about the log-rank test is as a Cox proportional hazards model on indicator variables for each of the groups. The log-rank test is a test that the coefficients are zero or, if you prefer, that the hazard ratios are one. The log-rank test is, in fact, a score test of that hypothesis performed on a slightly different (partial) likelihood function that handles ties more accurately.

Many researchers think that a (less precise) score test on the precise likelihood function is preferable to a (more precise) likelihood-ratio test on the approximate likelihood function used in Cox regression estimation. In our experience, it makes little difference:

```
. sts test posttran, cox
Equality of survivor functions
Cox regression-based test

             Observed      Expected     Relative
posttran       events        events       hazard

       0           30         31.20       0.9401
       1           45         43.80       1.0450

   Total           75         75.00       1.0000
                              LR chi2(1) =    0.13
                              Pr>chi2    = 0.7222
Note: Reporting LR model test from Cox regression
      on posttran.
```

By comparison, `sts test, logrank` also reported $\chi^2 = 0.13$, although the significance level was 0.7225, meaning that the χ^2 values differed in the fourth digit. As mentioned by Kalbfleisch and Prentice (2002, 20), a primary advantage of the log-rank test is the ease with which it can be explained to nonstatisticians, because the test statistic is the difference between the observed and expected number of failures within groups.

Our purpose in offering `sts test, cox` is not to promote its use instead of the log-rank test but to provide a test for researchers with sample-weighted data.

If you have sample weights (if you specified `pweight`s when you `stset` the data), you cannot run the log-rank or Wilcoxon tests. The Cox regression model, however, has been generalized to sample-weighted data, and Stata's `stcox` can fit models with such data. In sample-weighted data, the likelihood-ratio statistic is no longer appropriate, but the Wald test based on the robust estimator of variance is.

Thus if we treated these data as sample-weighted data, we would obtain

```
. generate one = 1
. stset t1 [pw=one], id(id) time0(_t0) failure(died) noshow
Survival-time data settings

           ID variable: id
        Failure event: died!=0 & died<.
Observed time interval: (_t0, t1]
    Exit on or before: failure
               Weight: [pweight=one]
_____

          172  total observations
            0  exclusions
_____

          172  observations remaining, representing
          103  subjects
           75  failures in single-failure-per-subject data
     31,938.1  total analysis time at risk and under observation
                                          At risk from t =         0
                                 Earliest observed entry t =         0
                                    Last observed exit t =     1,799

. sts test posttran, cox
Equality of survivor functions
Cox regression-based test
```

posttran	Observed events	Expected events	Relative hazard
0	30.00	31.20	0.9401
1	45.00	43.80	1.0450
Total	75.00	75.00	1.0000

```
                              Wald chi2(1) =   0.13
                              Pr>chi2      = 0.7181
Note: Reporting Wald model test from Cox
      regression on posttran.
```

`sts test, cox` now reports the Wald statistic, which is, to two digits, 0.13, just like all the others.

The trend test

When the groups to be compared have a natural order, such as increasing or decreasing age groups or drug dosage, you may want to test the null hypothesis that there is no difference in failure rate among the groups versus the alternative hypothesis that the failure rate increases or decreases as you move from one group to the next.

We illustrate this test with a dataset from a carcinogenesis experiment reprinted in Marubini and Valsecchi (1995, 126). Twenty-nine experimental animals were exposed to three levels (0, 1.5, 2.0) of a carcinogenic agent. The time in days to tumor formation was recorded. Here are a few of the observations:

```
. use https://www.stata-press.com/data/r18/marubini, clear
. list time event group dose in 1/9
```

	time	event	group	dose
1.	67	1	2	1.5
2.	150	1	2	1.5
3.	47	1	3	2
4.	75	0	1	0
5.	58	1	3	2
6.	136	1	2	1.5
7.	58	1	3	2
8.	150	1	2	1.5
9.	43	0	2	1.5

In these data, there are two variables that indicate exposure level. The `group` variable is coded 1, 2, and 3, indicating a one-unit separation between exposures. The `dose` variable records the actual exposure dosage. To test the null hypothesis of no difference among the survival experience of the three groups versus the alternative hypothesis that the survival experience of at least one of the groups is different, it does not matter if we use `group` or `dose`.

```
. stset time, fail(event) noshow
Survival-time data settings

         Failure event: event!=0 & event<.
Observed time interval: (0, time]
     Exit on or before: failure

─────────────────────────────────────────────────────────────────────
         29  total observations
          0  exclusions
─────────────────────────────────────────────────────────────────────
         29  observations remaining, representing
         15  failures in single-record/single-failure data
      2,564  total analysis time at risk and under observation
                                          At risk from t =         0
                                Earliest observed entry t =         0
                                   Last observed exit t =       246
. sts test group
Equality of survivor functions
Log-rank test
```

	Observed	Expected
group	events	events
1	4	6.41
2	6	6.80
3	5	1.79
Total	15	15.00

```
            chi2(2) =    8.05
            Pr>chi2 = 0.0179
```

```
. sts test dose
Equality of survivor functions
Log-rank test
```

dose	Observed events	Expected events
0	4	6.41
1.5	6	6.80
2	5	1.79
Total	15	15.00

```
                   chi2(2) =   8.05
                 Pr>chi2 = 0.0179
```

For the trend test, however, the distance between the values is important, so using group or dose will produce different results.

```
. sts test group, trend
Equality of survivor functions
Log-rank test
```

group	Observed events	Expected events
1	4	6.41
2	6	6.80
3	5	1.79
Total	15	15.00

```
                   chi2(2) =   8.05
                 Pr>chi2 = 0.0179
```

Test for trend in survivor functions

```
                   chi2(1) =   5.87
                 Pr>chi2 = 0.0154
```

```
. sts test dose, trend
Equality of survivor functions
Log-rank test
```

dose	Observed events	Expected events
0	4	6.41
1.5	6	6.80
2	5	1.79
Total	15	15.00

```
                   chi2(2) =   8.05
                 Pr>chi2 = 0.0179
```

Test for trend in survivor functions

```
                   chi2(1) =   3.66
                 Pr>chi2 = 0.0557
```

Although the above trend test was constructed using the log-rank test, any of the previously mentioned weight functions can be used. For example, a trend test on the data can be performed using the same weights as the Peto–Peto–Prentice test by specifying the peto option.

```
. sts test dose, trend peto

Equality of survivor functions
Peto-Peto-Prentice test
```

dose	Observed events	Expected events	Sum of ranks
0	4	6.41	-1.2792221
1.5	6	6.80	-1.3150418
2	5	1.79	2.5942639
Total	15	15.00	0

```
                          chi2(2) =    8.39
                          Pr>chi2 = 0.0150

Test for trend in survivor functions
                          chi2(1) =    2.85
                          Pr>chi2 = 0.0914
```

Video example

How to test the equality of survivor functions using nonparametric tests

Stored results

sts test stores the following in r():

Scalars

r(df)	degrees of freedom	r(chi2)	χ^2
r(df_tr)	degrees of freedom, trend test	r(chi2_tr)	χ^2, trend test

Methods and formulas

Let $t_1 < t_2 < \cdots < t_k$ denote the ordered failure times; let d_j be the number of failures at t_j and n_j be the population at risk just before t_j; and let d_{ij} and n_{ij} denote the same things for group i, $i = 1, \ldots, r$.

We are interested in testing the null hypothesis

$$H_0 \colon \lambda_1(t) = \lambda_2(t) = \cdots = \lambda_r(t)$$

where $\lambda(t)$ is the hazard function at time t, against the alternative hypothesis that at least one of the $\lambda_i(t)$ is different for some t_j.

As described in Klein and Moeschberger (2003, 205–216), Kalbfleisch and Prentice (2002, 20–22), and Collett (2015, 50–51), if the null hypothesis is true, the expected number of failures in group i at time t_j is $e_{ij} = n_{ij}d_j/n_j$, and the test statistic

$$\mathbf{u}' = \sum_{j=1}^{k} W(t_j)(d_{1j} - e_{1j}, \ldots, d_{rj} - e_{rj})$$

is formed. $W(t_j)$ is a positive weight function defined as zero when n_{ij} is zero. The various test statistics are obtained by selecting different weight functions, $W(t_j)$. See the table in the *Remarks and examples* section of this entry for a list of these weight functions. For the Peto–Peto–Prentice test,

$$W(t_j) = \widetilde{S}(t_j) = \prod_{\ell : t_\ell \leq t_j} \left(1 - \frac{d_\ell}{n_\ell + 1}\right)$$

The variance matrix $\mathbf{V}$ for $\mathbf{u}$ has elements

$$V_{il} = \sum_{j=1}^{k} \frac{W(t_j)^2 n_{ij} d_j (n_j - d_j)}{n_j (n_j - 1)} \left(\delta_{il} - \frac{n_{ij}}{n_j}\right)$$

where $\delta_{il} = 1$ if $i = l$ and $\delta_{il} = 0$, otherwise.

For the unstratified test, statistic $\mathbf{u}'\mathbf{V}^{-1}\mathbf{u}$ is distributed as χ^2 with $r - 1$ degrees of freedom.

For the stratified test, let $\mathbf{u}_s$ and $\mathbf{V}_s$ be the results of performing the above calculation separately within stratum, and define $\mathbf{u} = \sum_s \mathbf{u}_s$ and $\mathbf{V} = \sum_s \mathbf{V}_s$. The χ^2 test is given by $\mathbf{u}'\mathbf{V}^{-1}\mathbf{u}$ redefined in this way.

The "Cox" test is performed by fitting a (possibly stratified) Cox regression using stcox on $r - 1$ indicator variables, one for each group with one of the indicators omitted. The χ^2 test reported is then the likelihood-ratio test (no pweights) or the Wald test (based on the robust estimate of variance); see [ST] **stcox**.

The reported relative hazards are the exponentiated coefficients from the Cox regression renormalized, and the renormalization plays no role in calculating the test statistic. The renormalization is chosen so that the expected-number-of-failures-within-group weighted average of the regression coefficients is 0 (meaning that the hazard is 1). Let b_i, $i = 1, \ldots, r - 1$, be the estimated coefficients, and define $b_r = 0$. The constant K is then calculated with

$$K = \sum_{i=1}^{r} e_i b_i / d$$

where $e_i = \sum_j e_{ij}$ is the expected number of failures for group i, d is the total number of failures across all groups, and r is the number of groups. The reported relative hazards are $\exp(b_i - K)$.

The trend test assumes that there is natural ordering of the r groups, $r > 2$. Here we are interested in testing the null hypothesis

$$H_0 : \lambda_1(t) = \lambda_2(t) = \cdots = \lambda_r(t)$$

against the alternative hypothesis

$$H_a : \lambda_1(t) \leq \lambda_2(t) \leq \cdots \leq \lambda_r(t)$$

The test uses **u** as previously defined with any of the available weight functions. The test statistic is given by

$$
\frac{\left(\sum_{i=1}^{r} a_i u_i\right)^2}{\mathbf{a}'\mathbf{V}\mathbf{a}}
$$

where $a_1 \leq a_2 \leq \cdots \leq a_r$ are scores defining the relationship of interest. A score is assigned to each comparison group, equal to the value of the grouping variable for that group. **a** is the vector of these scores.

References

Breslow, N. E. 1970. A generalized Kruskal–Wallis test for comparing *K* samples subject to unequal patterns of censorship. *Biometrika* 57: 579–594. https://doi.org/10.1093/biomet/57.3.579.

Collett, D. 2015. *Modelling Survival Data in Medical Research.* 3rd ed. Boca Raton, FL: Chapman and Hall/CRC.

Gehan, E. A. 1965. A generalized Wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika* 52: 203–223. https://doi.org/10.2307/2333825.

Harrington, D. P., and T. R. Fleming. 1982. A class of rank test procedures for censored survival data. *Biometrika* 69: 553–566. https://doi.org/10.2307/2335991.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data.* 2nd ed. New York: Wiley.

Karrison, T. G. 2016. Versatile tests for comparing survival curves based on weighted log-rank statistics. *Stata Journal* 16: 678–690.

Klein, J. P., and M. L. Moeschberger. 2003. *Survival Analysis: Techniques for Censored and Truncated Data.* 2nd ed. New York: Springer.

Mantel, N. 1963. Chi-square tests with one degree of freedom; extensions of the Mantel–Haenszel procedure. *Journal of the American Statistical Association* 58: 690–700. https://doi.org/10.1080/01621459.1963.10500879.

———. 1966. Evaluation of survival data and two new rank-order statistics arising in its consideration. *Cancer Chemotherapy Reports* 50: 163–170.

Mantel, N., and W. Haenszel. 1959. Statistical aspects of the analysis of data from retrospective studies of disease. *Journal of the National Cancer Institute* 22: 719–748. Reprinted in *Evolution of Epidemiologic Ideas: Annotated Readings on Concepts and Methods*, ed. S. Greenland, pp. 112–141. Newton Lower Falls, MA: Epidemiology Resources.

Marubini, E., and M. G. Valsecchi. 1995. *Analysing Survival Data from Clinical Trials and Observational Studies.* New York: Wiley.

Peto, R., and J. Peto. 1972. Asymptotically efficient rank invariant test procedures. *Journal of the Royal Statistical Society, Series A* 135: 185–207. https://doi.org/10.2307/2344317.

Prentice, R. L. 1978. Linear rank tests with right censored data. *Biometrika* 65: 167–179. https://doi.org/10.2307/2335292.

Savage, I. R. 1956. Contributions to the theory of rank-order statistics—the two-sample case. *Annals of Mathematical Statistics* 27: 590–615. https://doi.org/10.1214/aoms/1177728170.

Tarone, R. E., and J. H. Ware. 1977. On distribution-free tests for equality of survival distributions. *Biometrika* 64: 156–160. https://doi.org/10.2307/2335790.

White, I. R., S. Walker, and A. G. Babiker. 2002. strbee: Randomization-based efficacy estimator. *Stata Journal* 2: 140–150.

Wilcoxon, F. 1945. Individual comparisons by ranking methods. *Biometrics* 1: 80–83. https://doi.org/10.2307/3001968.

Also see

Title

stset — Declare data to be survival-time data

[Description](#)   [Quick start](#)   [Menu](#)   [Syntax](#)
[Options](#)   [Remarks and examples](#)   [References](#)   [Also see](#)

Description

st refers to survival-time data, which are fully described below.

stset declares the data in memory to be st data, informing Stata of key variables and their roles in a survival-time analysis. When you stset your data, stset runs various data consistency checks to ensure that what you have declared makes sense. If the data are weighted, you specify the weights when you stset the data, not when you issue the individual st commands.

streset changes how the st dataset is declared. In multiple-record data, streset can also temporarily set the sample to include records from before the time at risk (called the past) and records after failure (called the future). Then typing streset without arguments resets the sample back to the analysis sample.

st displays how the dataset is currently declared.

Whenever you type stset or streset, Stata runs or reruns data consistency checks to ensure that what you are now declaring (or declared in the past) makes sense. Thus if you have made any changes to your data or simply wish to verify how things are, you can type streset with no options.

stset, clear is for use by programmers. It causes Stata to forget the st markers, making the data no longer st data to Stata. The data remain unchanged. It is not necessary to stset, clear before doing another stset.

Quick start

Single-record-per-subject survival data

Specify time of failure, recorded in tvar, for data without censoring
        stset tvar

Specify time of censoring or failure, tvar, and specify that event $= 2$ represents a failure
        stset tvar, failure(event==2)

Specify that event $= 2$ and event $= 3$ represent failures
        stset tvar, failure(event==2 3)

Specify failure using indicator variable fail
        stset tvar, failure(fail)

Same as above, and specify that subjects become at risk at time torig
        stset tvar, failure(fail) origin(time torig)

Specify that subjects become at risk at time 0, but enter the study at time tenter
        stset tvar, fail(failure) enter(time tenter)

Specify subjects become at risk at time `torig`, and enter the study at time `tenter`
```
stset tvar, fail(failure) origin(time torig) enter(time tenter)
```

Same as above, but specify analysis time in years when time variables are measured in days
```
stset tvar, fail(failure) origin(time torig) enter(time tenter)  ///
    scale(365.25)
```

Convert analysis time units back to days
```
streset, scale(1)
```

Display previous `st` settings and verify that any changes to data correspond to settings
```
stset
```

Multiple-record-per-subject survival data

Specify analysis-time variable `tvar` with failure indicator `fail` and subject identifier `idvar`
```
stset tvar, failure(fail) id(idvar)
```

Same as above, and specify that subjects become at risk at time `torig`
```
stset tvar, failure(fail) id(idvar) origin(time torig)
```

Same as above, and specify that subjects enter the study at time `tenter`
```
stset tvar, failure(fail) id(idvar) origin(time torig)  ///
    enter(time tenter)
```

Same as above, and specify that subjects exit the study at time `texit`
```
stset tvar, failure(fail) id(idvar) origin(time torig)  ///
    enter(time tenter) exit(time texit)
```

Menu

Statistics > Survival analysis > Setup and utilities > Declare data to be survival-time data

Syntax

Single-record-per-subject survival data

    stset *timevar* $\big[$*if*$\big]$ $\big[$*weight*$\big]$ $\big[$, *single_options*$\big]$

    streset $\big[$*if*$\big]$ $\big[$*weight*$\big]$ $\big[$, *single_options*$\big]$

    st $\big[$, <u>noc</u>md <u>not</u>able$\big]$

    stset, clear

Multiple-record-per-subject survival data

    stset *timevar* $\big[$*if*$\big]$ $\big[$*weight*$\big]$, id(*idvar*) <u>f</u>ailure(*failvar*$\big[$==*numlist*$\big]$)
        $\big[$*multiple_options*$\big]$

    streset $\big[$*if*$\big]$ $\big[$*weight*$\big]$ $\big[$, *multiple_options*$\big]$

    streset, { <u>p</u>ast | <u>f</u>uture | <u>p</u>ast <u>f</u>uture }

    st $\big[$, <u>noc</u>md <u>not</u>able$\big]$

    stset, clear

single_options	Description
Main	
<u>f</u>ailure(*failvar*$\big[$==*numlist*$\big]$)	failure event
<u>nos</u>how	prevent other st commands from showing st setting information
Options	
<u>o</u>rigin(<u>t</u>ime *exp*)	define when a subject becomes at risk
<u>sc</u>ale(*#*)	rescale time value
<u>e</u>nter(<u>t</u>ime *exp*)	specify when subject first enters study
<u>ex</u>it(<u>t</u>ime *exp*)	specify when subject exits study
Advanced	
if(*exp*)	select records for which *exp* is true; recommended rather than if *exp*
time0(*varname*)	mechanical aspect of interpretation about records in dataset; seldom used

multiple_options	Description
┌─ **Main** ─┐	
* id(*idvar*)	multiple-record ID variable
* <u>f</u>ailure(*failvar* [==*numlist*])	failure event
<u>nosh</u>ow	prevent other st commands from showing st setting information
┌─ **Options** ─┐	
<u>o</u>rigin([*varname*==*numlist*] <u>t</u>ime *exp* | min)	define when a subject becomes at risk
<u>sc</u>ale(*#*)	rescale time value
<u>e</u>nter([*varname*==*numlist*] <u>t</u>ime *exp*)	specify when subject first enters study
<u>e</u>xit(failure | [*varname*==*numlist*] <u>t</u>ime *exp*)	specify when subject exits study
┌─ **Advanced** ─┐	
if(*exp*)	select records for which *exp* is true; recommended rather than if *exp*
ever(*exp*)	select subjects for which *exp* is ever true
never(*exp*)	select subjects for which *exp* is never true
after(*exp*)	select records within subject on or after the first time *exp* is true
<u>befo</u>re(*exp*)	select records within subject before the first time *exp* is true
time0(*varname*)	mechanical aspect of interpretation about records in dataset; seldom used

* id() and failure() are required with stset multiple-record-per-subject survival data.

fweights, iweights, and pweights are allowed; see [U] **11.1.6 weight**.

Examples

. stset ftime	(time measured from 0, all failed)
. stset ftime, failure(died)	(time measured from 0, censoring)
. stset ftime, failure(died) id(id)	(time measured from 0, censoring & ID)
. stset ftime, failure(died==2,3)	(time measured from 0, failure codes)
. stset ftime, failure(died) origin(time dob)	(time measured from dob, censoring)

You cannot harm your data by using stset, so feel free to experiment.

Options

Options are presented under the following headings:

> *Options for use with stset and streset*
> *Options unique to streset*
> *Options for st*

Options for use with stset and streset

Main

id(*idvar*) specifies the subject-ID variable; observations with equal, nonmissing values of *idvar* are
assumed to be the same subject. *idvar* may be string or numeric. Observations for which *idvar* is
missing (. or "") are ignored.

When id() is not specified, each observation is assumed to represent a different subject and thus
constitutes a one-record-per-subject survival dataset.

When you specify id(), the data are said to be multiple-record data, even if it turns out that there
is only one record per subject. Perhaps they would better be called potentially multiple-record
data.

If you specify id(), stset requires that you specify failure().

Specifying id() never hurts; we recommend it because a few st commands, such as stsplit,
require an ID variable to have been specified when the dataset was stset.

failure(*failvar*[==*numlist*]) specifies the failure event.

If failure() is not specified, all records are assumed to end in failure. This is allowed with
single-record data only.

If failure(*failvar*) is specified, *failvar* is interpreted as an indicator variable; 0 and missing
mean censored, and all other values are interpreted as representing failure.

If failure(*failvar*==*numlist*) is specified, records with *failvar* taking on any of the values in
numlist are assumed to end in failure, and all other records are assumed to be censored.

noshow prevents other st commands from showing the key st variables at the top of their output.

Options

origin([*varname*==*numlist*] time *exp* | min) and scale(#) define analysis time; that is, origin()
defines when a subject becomes at risk. Subjects become at risk when time = origin(). All
analyses are performed in terms of time since becoming at risk, called analysis time.

Let us use the terms *time* for how time is recorded in the data and t for analysis time. Analysis
time t is defined

$$t = \frac{time - \texttt{origin()}}{\texttt{scale()}}$$

t is time from origin in units of scale.

By default, origin(time 0) and scale(1) are assumed, meaning that $t = time$. Then you must
ensure that *time* in your data is measured as time since becoming at risk. Subjects are exposed
at $t = time = 0$ and later fail. Observations with $t = time \leq 0$ are ignored because information
before becoming at risk is irrelevant.

origin() determines when the clock starts ticking. scale() plays no substantive role, but it can
be handy for making t units more readable (such as converting days to years).

origin(time *exp*) sets the origin to *exp*. For instance, if *time* were recorded as dates, such as
05jun1998, in your data and variable expdate recorded the date when subjects were exposed, you
could specify origin(time expdate). If instead all subjects were exposed on 12nov1997, you
could specify origin(time mdy(11,12,1997)).

origin(time *exp*) may be used with single- or multiple-record data.

origin(*varname*==*numlist*) is for use with multiple-record data; it specifies the origin indirectly. If *time* were recorded as dates in your data, variable `obsdate` recorded the (ending) date associated with each record, and subjects became at risk upon, say, having a certain operation— and that operation were indicated by `code==217`—then you could specify `origin(code==217)`. `origin(code==217)` would mean, for each subject, that the origin time is the earliest time at which `code==217` is observed. Records before that would be ignored (because $t < 0$). Subjects who never had `code==217` would be ignored entirely.

origin(*varname*==*numlist* time *exp*) sets the origin to the later of the two times determined by *varname*==*numlist* and *exp*.

origin(min) sets origin to the earliest time observed, minus 1. This is an odd thing to do and is described in *example 10*.

origin() is an important concept; see *Key concepts*, *Two concepts of time*, and *The substantive meaning of analysis time*.

scale() makes results more readable. If you have *time* recorded in days (such as Stata dates, which are really days since 01jan1960), specifying `scale(365.25)` will cause results to be reported in years.

enter([*varname*==*numlist*] time *exp*) specifies when a subject first comes under observation, meaning that any failures, were they to occur, would be recorded in the data.

Do not confuse enter() and origin(). origin() specifies when a subject first becomes at risk. In many datasets, becoming at risk and coming under observation are coincident. Then it is sufficient to specify origin().

enter(time *exp*), enter(*varname*==*numlist*), and enter(*varname*==*numlist* time *exp*) follow the same syntax as origin(). In multiple-record data, both *varname*==*numlist* and time *exp* are interpreted as the earliest time implied, and if both are specified, the later of the two times is used.

exit(failure | [*varname*==*numlist*] time *exp*) specifies the latest time under which the subject is both under observation and at risk. The emphasis is on latest; obviously, subjects also exit the risk pool when their data run out.

exit(failure) is the default. When the first failure event occurs, the subject is removed from the analysis risk pool, even if the subject has subsequent records in the data and even if some of those subsequent records document other failure events. Specify exit(time .) if you wish to keep all records for a subject after failure. You want to do this if you have multiple-failure data.

exit(*varname*==*numlist*), exit(time *exp*), and exit(*varname*==*numlist* time *exp*) follow the same syntax as origin() and enter(). In multiple-record data, both *varname*==*numlist* and time *exp* are interpreted as the earliest time implied. exit differs from origin() and enter() in that if both are specified, the earlier of the two times is used.

___

> Advanced

if(*exp*), ever(*exp*), never(*exp*), after(*exp*), and before(*exp*) select relevant records.

if(*exp*) selects records for which *exp* is true. We strongly recommend specifying this if() option rather than if *exp* following stset or streset. They differ in that if *exp* removes the data from consideration before calculating beginning and ending times and other quantities. The if() option, on the other hand, sets the restriction after all derived variables are calculated. See *if( ) versus if exp*.

if() may be specified with single- or multiple-record data. The remaining selection options are for use with multiple-record data only.

ever(*exp*) selects only subjects for which *exp* is ever true.

never(*exp*) selects only subjects for which *exp* is never true.

after(*exp*) selects records within subject on or after the first time *exp* is true.

before(*exp*) selects records within subject before the first time *exp* is true.

time0(*varname*) is seldom specified because most datasets do not contain this information. time0() should be used exclusively with multiple-record data, and even then you should consider whether origin() or enter() would be more appropriate.

time0() specifies a mechanical aspect of interpretation about the records in the dataset, namely, the beginning of the period spanned by each record. See *Intermediate exit and reentry times (gaps)*.

Options unique to streset

past expands the stset sample to include the entire recorded past of the relevant subjects, meaning that it includes observations before becoming at risk and those excluded because of after(), etc.

future expands the stset sample to include the records on the relevant subjects after the last record that previously was included, if any, which typically means to include all observations after failure or censoring.

past future expands the stset sample to include all records on the relevant subjects.

Typing streset without arguments resets the sample to the analysis sample. See *Past and future records* for more information.

Options for st

nocmd suppresses displaying the last stset command.

notable suppresses displaying the table summarizing what has been stset.

Remarks and examples

Remarks are presented under the following headings:

What are survival-time data?

Survival-time data—what we call st data—document spans of time ending in an event. For instance,

```
                                              died
                      x1=17                    |
                      x2=22                    |
         |<─────────────────────────────────>|
         |─────────────────────────────────────────────> t
         0                                    9
```
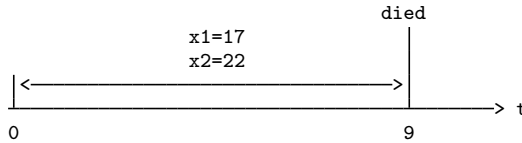
which indicates $x1 = 17$ and $x2 = 22$ over the time span 0 to 9, and $\mathtt{died} = 1$. More formally, it means $x1 = 17$ and $x2 = 22$ for $0 < t \leq 9$, which we often write as $(0, 9]$. However you wish to say it, this information might be recorded by the observation

```
id              end     x1      x2      died
101               9      17      22        1
```

and we call this single-record survival data.

The data can be more complicated. For instance, we might have

```
                                              died
              x1=17              x1=12         |
              x2=22              x2=22         |
         |<──────────────>|<──────────────>|
         |──────────────────────────────────────> t
         0                4                9
```
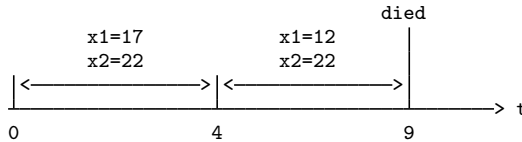
meaning

$$x1 = 17 \text{ and } x2 = 22 \text{ during } (0, 4]$$
$$x1 = 12 \text{ and } x2 = 22 \text{ during } (4, 9], \text{ and then } \mathtt{died} = 1.$$

and this would be recorded by the data

```
id    begin    end     x1      x2      died
101       0      4      17      22        0
101       4      9      12      22        1
```

We call this multiple-record survival data.

These two formats allow you to record many different possibilities. The last observation on a person need not be failure,

```
                                        lost because of censoring
                      x1=17                    |
                      x2=22                    |
         |<─────────────────────────────────>|
         |─────────────────────────────────────────────> t
         0                                    9

id              end     x1      x2      died
101               9      17      22        0
```

or
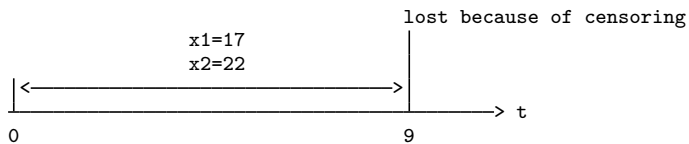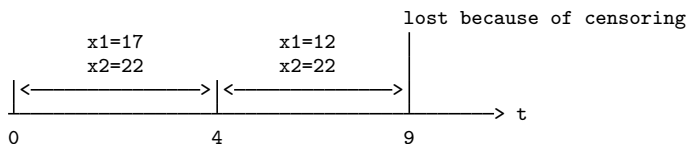
```
                                        lost because of censoring
              x1=17              x1=12         |
              x2=22              x2=22         |
         |<──────────────>|<──────────────>|
         |──────────────────────────────────────> t
         0                4                9

id    begin    end     x1      x2      died
101       0      4      17      22        0
101       4      9      12      22        0
```

Multiple-record data might have gaps,

```
                                                            died
                  x1=17                           x1=12
                  x2=22                           x2=22
          |<──────────────>|  (not observed)  |<──────────────>|
          |                |                   |                |──> t
          0                4                   9                14
     id   begin     end     x1      x2      died
     101       0       4     17      22        0
     101       9      14     12      22        1
```
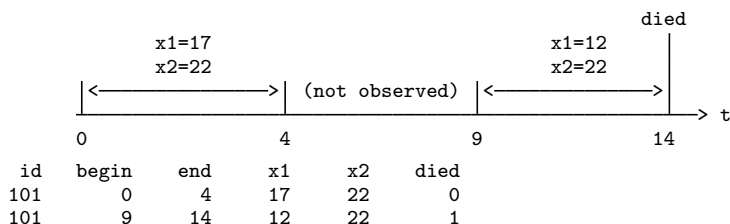
or subjects might not be observed from the onset of risk,

```
       exposure                              died
          |                  x1=17             |
          |                  x2=22             |
          |         |<────────────────────────>|
          |         |                          |──> t
          0         2                          9
          begin     end     x1      x2      died
              2       9      17      22        1
```

and

```
       exposure                              died
          |           x1=17        x1=12       |
          |           x2=22        x2=22       |
          |       |<─────────>|<─────────>|
          |       |           |           |──> t
          0       1           4           9
     id   begin     end     x1      x2      died
     101       1       4      17      22        0
     101       4       9      12      22        1
```
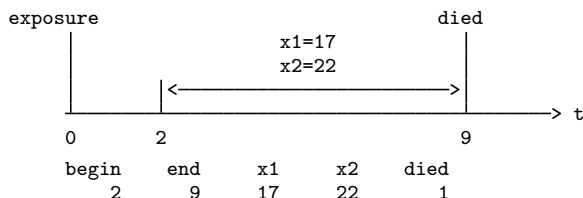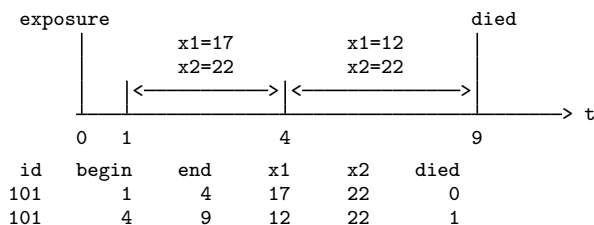
The failure event might not be death but instead something that can be repeated:

```
                       1st                         2nd
                    infarction                  infarction
             x1=17        |    x1=12        |  x1=10        |
             x2=22        |    x2=22        |  x2=22        |
        |<───────────>|<───────────>|<─────────>|
        |             |             |           |──> t
        0             4             9           13
     id   begin     end     x1      x2     infarc
     101       0       4      17      22        1
     101       4       9      12      22        0
     101       9      13      10      22        1
```
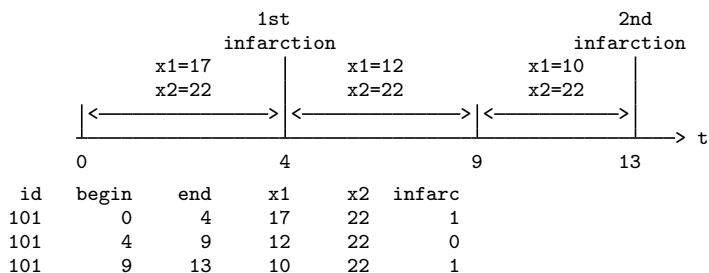
Our data may be in different time units; rather than t where $t = 0$ corresponds to the onset of risk, we might have time recorded as age,

```
                                       died
                  x1=17                  |
                  x2=22                  |
          |<─────────────────────────────>|
          |                               |──> age
          20                              29
     id   age0     age1     x1      x2      died
     101     20       29     17      22        1
```
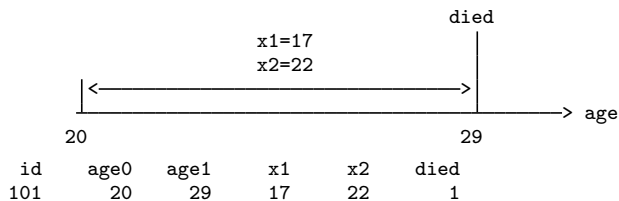
or time recorded as calendar dates:

```
                                          died
               x1=17              x1=12
               x2=22              x2=22
          |<---------------->|<--------------->|
          |                  |                 |          ---> date
      01jan1998          02may1998         15oct1998
```

id	bdate	edate	x1	x2	died
101	01jan1998	02may1998	17	22	0
101	02may1998	15oct1998	12	22	1
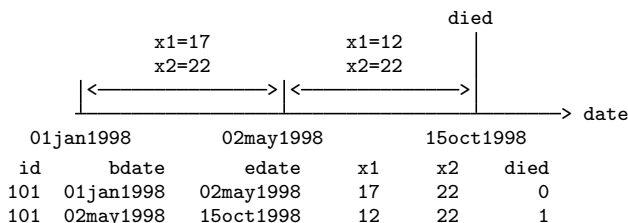
Finally, we can mix these diagrams however we wish, so we might have time recorded according to the calendar, unobserved periods after the onset of risk, subsequent gaps, and multiple failure events.

The st commands analyze data like these, and the first step is to tell st about your data by using stset. You do not change your data to fit some predefined mold; you describe your data with stset, and the rest of the st commands just do the right thing.

Before we discuss using stset, let's describe one more style of recording time-to-event data because it is common and is inappropriate for use with st. It is inappropriate, but it is easy to convert to the survival-time form. It is called snapshot data, which are data for which you do not know spans of time but you have information recorded at various points in time:

```
         x1=17            x1=12
         x2=22            x2=22              died
         |(unobserved)|  |(unobserved)|      |
         |            |  |            |       |      ---> t
         0            4  4            9       9
```

id	t	x1	x2	died
101	0	17	22	0
101	4	12	22	0
101	9	.	.	1
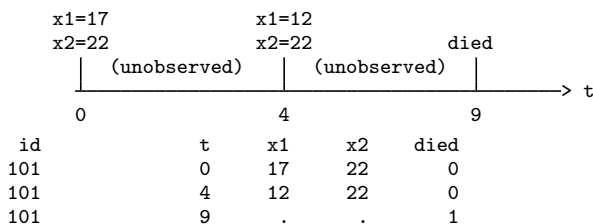
In this snapshot dataset, all we know are the values of x1 and x2 at $t = 0$ and $t = 4$, and we know that the subject died at $t = 9$. Snapshot data can be converted to survival-time data if we are willing to assume that x1 and x2 remained constant between times:

```
                                       died
            x1=17              x1=12
            x2=22              x2=22
          |<-------------->|<-------------->|
          |                |                |    --->
          0                4                9
```

id	begin	end	x1	x2	died
101	0	4	17	22	0
101	4	9	12	22	1
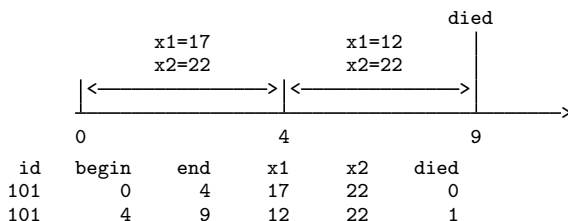
The snapspan command makes this conversion. If you have snapshot data, first see [ST] **snapspan** to convert it to survival-time data and then use stset to tell st about the converted data; see example 10 first.

Key concepts

time, or, better, *time units*, is how time is recorded in your data. It might be numbers (such as 0, 1, 2, ..., with time = 0 corresponding to some exposure event), a subject's age, or calendar time.

events are things that happen at an instant in time, such as being exposed to an environmental hazard, being diagnosed as myopic, becoming employed, being promoted, becoming unemployed, having a heart attack, and dying.

failure event is the event indicating failure as it is defined for analysis. This can be a single or compound event. The failure event might be when variable dead is 1, or it might be when variable diag is any of 115, 121, or 133.

at risk means the subject is at risk of the failure event occurring. For instance, if the failure event is becoming unemployed, a person must be employed. The subject is not at risk before being employed. Once employed, the subject becomes at risk; once again, the subject is no longer at risk once the failure event occurs. If subjects become at risk upon the occurrence of some event, it is called the exposure event. Gaining employment is the exposure event in our example.

origin is the time when the subject became at risk. If time is recorded as numbers such as 0, 1, 2, ..., with time = 0 corresponding to the exposure event, then origin = 0. Alternatively, origin might be the age of the subject when diagnosed or the date when the subject was exposed. Regardless, origin is expressed in time units.

scale is just a fixed number, typically 1, used in mapping time to analysis time t.

t, or *analysis time*, is $(\text{time} - \text{origin})/\text{scale}$, meaning the time since onset of being at risk measured in scale units.

$t = 0$ corresponds to the onset of risk, and scale just provides a way to make the units of t more readable. You might have time recorded in days from 01jan1960 and want t recorded in years, in which case scale would be 365.25.

Time is how time is recorded in your data, and t is how time is reported in the analysis.

under observation means that, should the failure event occur, it would be observed and recorded in the data. Sometimes subjects are under observation only after they are at risk. This would be the case, for instance, if subjects enrolled in a study after being diagnosed with cancer and if, to enroll in the study, subjects were required to be diagnosed with cancer.

Being under observation does not mean that the subject is necessarily at risk. A subject may come under observation before being at risk, and in fact, a subject under observation may never become at risk.

entry time and *exit time* mark when a subject is first and last under observation. The emphasis here is on the words *first* and *last*; entry time and exit time do not record observational gaps; there is only one entry time and one exit time per subject.

Entry time and exit time might be expressed as times (recorded in time units), or they might correspond to the occurrence of some event (such as enrolling in the study).

Often the entry time corresponds to $t = 0$; that is, because $t = (\text{time} - \text{origin})/\text{scale}$, time = origin, meaning that time equals when the subject became at risk.

Often the exit time corresponds to when the failure event occurs or, failing that, the end of data for the subject.

delayed entry means that entry time corresponds to $t > 0$; the subject became at risk before coming under observation.

ID refers to a subject identification variable; equal values of ID indicate that the records are on the same subject. An ID variable is required for multiple-record data and is optional, but recommended, with single-record data.

time0 refers to the beginning time (recorded in time units) of a record. Some datasets have this variable, but most do not. If the dataset does not contain the beginning time for each record, subsequent records are assumed to begin where previous records ended. A time0 variable may be created for these datasets by using the snapspan command; see [ST] **snapspan**. Do not confuse time0—a mechanical aspect of datasets—with entry time—a substantive aspect of analysis.

gaps refer to gaps in observation between entry time and exit time. During a gap, a subject is not under observation. Gaps can arise only if the data contain a time0 variable, because otherwise subsequent records beginning when previous records end would preclude there being gaps in the data. Gaps are distinct from delayed entry.

past history is a term we use to mean information recorded in the data before the subject was both at risk and under observation. Complex datasets can contain such observations. Say that the dataset contains histories on subjects from birth to death. You might tell st that a subject becomes at risk once diagnosed with a particular kind of cancer. The past history on the subject would then refer to records before the subject was diagnosed.

The word *history* is often dropped, and the term simply becomes *past*. For instance, we might want to know whether the subject smoked in the past.

future history is a term meaning information recorded in the data after the subject is no longer at risk. Perhaps the failure event is not so serious as to preclude the possibility of data after failure.

The word *history* is often dropped, and the term simply becomes *future*. Perhaps the failure event is cardiac infarction, and you want to know whether the subject died soon in the future so that you can exclude the subject.

Survival-time datasets

In survival-time datasets, observations (records) document a span of time. The span might be explicitly indicated, such as

```
          begin   end     x1      x2
              3     9     17      22              <- spans (3,9]
```

or it might be implied that the record begins at 0,

```
                  end     x1      x2
                    9     17      22              <- spans (0,9]
```

or it might be implied because there are multiple records per subject:

```
          id      end     x1      x2
           1        4     17      22              <- spans (0,4]
           1        9     12      22              <- spans (4,9]
```

Records may have an event indicator:

```
          begin   end     x1      x2    died
              3     9     17      22       1      <- spans (3,9], died at t=9

                  end     x1      x2    died
                    9     17      22       1      <- spans (0,9], died at t=9

          id      end     x1      x2    died
           1        4     17      22       0      <- spans (0,4],
           1        9     12      22       1      <- spans (4,9], died at t=9
```

The first two examples are called single-record survival-time data because there is one record per subject.

The final example is called multiple-record survival-time data. There are two records for the subject with id $= 1$.

Either way, survival-time data document time spans. Characteristics are assumed to remain constant over the span, and the event is assumed to occur at the end of the span.

Using stset

Once you have `stset` your data, you can use the other st commands.

If you `save` your data after `stset`ting, you will not have to re-`stset` in the future; Stata will remember.

`stset` declares your data to be survival-time data. It does not change the data, although it does add a few variables to your dataset.

This means that you can re-`stset` your data as often as you wish. In fact, the `streset` command encourages this. Using complicated datasets often requires typing long `stset` commands, such as

```
. stset date, fail(event==27 28) origin(event==15) enter(event==22)
```

Later, you might want to try `fail(event==27)`. You could retype the `stset` command, making the substitution, or you could type

```
. streset, fail(event==27)
```

`streset` takes what you type, merges it with what you have previously declared with `stset`, and performs the combined `stset` command.

▷ Example 1: Single-record data

Generators are run until they fail. Here is some of our dataset:

```
. use https://www.stata-press.com/data/r18/kva
(Generator experiment)
. list in 1/3
```

	failtime	load	bearings
1.	100	15	0
2.	140	15	1
3.	97	20	0

The `stset` command for this dataset is

```
. stset failtime

Survival-time data settings

          Failure event: (assumed to fail at time=failtime)
Observed time interval: (0, failtime]
    Exit on or before: failure

────────────────────────────────────────────────────────────────────
           12  total observations
            0  exclusions
────────────────────────────────────────────────────────────────────
           12  observations remaining, representing
           12  failures in single-record/single-failure data
          896  total analysis time at risk and under observation
                                          At risk from t =          0
                                 Earliest observed entry t =          0
                                    Last observed exit t =        140
```

When you type `stset` *timevar*, *timevar* is assumed to be the time of failure. More generally, you will learn, *timevar* is the time of failure or censoring. Here *timevar* is `failtime`.

```
. describe
Contains data from https://www.stata-press.com/data/r18/kva.dta
 Observations:              12                   Generator experiment
    Variables:               7                   8 Jan 2022 15:59
```

Variable name	Storage type	Display format	Value label	Variable label
failtime	int	%9.0g		Time until failure (hrs.)
load	byte	%9.0g		Overload (kVA)
bearings	byte	%9.0g		Has new bearings
_st	byte	%8.0g		1 if record is to be used; 0 otherwise
_d	byte	%8.0g		1 if failure; 0 if censored
_t	int	%10.0g		Analysis time when record ends
_t0	byte	%10.0g		Analysis time when record begins

```
Sorted by:
```

When you stset this dataset, Stata added the system variables _st, _d, _t, and _t0 to your data.

◁

▷ Example 2: Single-record data with censoring

Generators are run until they fail, but during the experiment, the room flooded, so some generators were run only until the flood. Here are some of our data:

```
. use https://www.stata-press.com/data/r18/kva2
(Generator experiment)
. list in 1/4
```

	failtime	load	bearings	failed
1.	100	15	0	1
2.	140	15	1	0
3.	97	20	0	1
4.	122	20	1	1

Here the second generator did not fail at time 140; the experiment was merely discontinued then. The stset command for this dataset is

```
. stset failtime, failure(failed)

Survival-time data settings

         Failure event: failed!=0 & failed<.
Observed time interval: (0, failtime]
     Exit on or before: failure

─────────────────────────────────────────────────────────────────────
         12  total observations
          0  exclusions
─────────────────────────────────────────────────────────────────────
         12  observations remaining, representing
         11  failures in single-record/single-failure data
        896  total analysis time at risk and under observation
                                           At risk from t =         0
                                 Earliest observed entry t =         0
                                    Last observed exit t =       140
```

When you type stset *timevar*, failure(*failvar*), *timevar* is interpreted as the time of failure or censoring, which is determined by the value of *failvar*. *failvar* = 0 and *failvar* = . (missing) indicate censorings, and all other values indicate failure.

◁

▷ Example 3: Multiple-record data

Assume that we are analyzing survival time of patients with a particular kind of cancer. In this dataset, the characteristics of patients vary over time, perhaps because new readings were taken or because the drug therapy was changed. Some of the data are

```
. list, separator(0)
```

	patid	t	died	x1	x2
1.	90	100	0	1	0
2.	90	150	1	0	0
3.	91	50	1	1	1
4.	92	100	0	0	0
5.	92	150	0	0	1
6.	92	190	0	0	0
7.	93	100	0	0	0

(*output omitted*)

There are two records for patient 90, and died is 0 in the first record but 1 in the second. The interpretation of these two records is

Interval (0,100]:	x1 = 1 and x2 = 0
Interval (100,150]:	x1 = 0 and x2 = 0
at t = 150:	the patient died

Similarly, here is how you interpret the other records:

Patient 91:

Interval (0, 50]:	x1 = 1 and x2 = 1
at t = 50:	the patient died

Patient 92:

Interval (0,100]:	x1 = 0 and x2 = 0
Interval (100,150]:	x1 = 0 and x2 = 1
Interval (150,190]:	x1 = 0 and x2 = 0
at t = 190:	the patient was lost because of censoring

Look again at patient 92's data:

patid	t	died	x1	x2
92	100	0	0	0
92	150	0	0	1
92	190	0	0	0

died = 0 for the first event. Mechanically, this removes the subject from the data at t = 100—the patient is treated as censored. The next record, however, adds the patient back into the data (at t = 100) with new characteristics.

The `stset` command for this dataset is

```
. stset t, id(patid) failure(died)

Survival-time data settings

           ID variable: patid
        Failure event: died!=0 & died<.
Observed time interval: (t[_n-1], t]
    Exit on or before: failure

─────────────────────────────────────────────────────────────────
         126  total observations
           0  exclusions
─────────────────────────────────────────────────────────────────
         126  observations remaining, representing
          40  subjects
          26  failures in single-failure-per-subject data
       2,989  total analysis time at risk and under observation
                                        At risk from t =         0
                                 Earliest observed entry t =      0
                                  Last observed exit t =        139
```

When you have multiple-record data, you specify `stset`'s `id(idvar)` option. When you type `stset` *timevar*, `id(idvar)` `failure(failvar)`, *timevar* denotes the end of the period (just as it does in single-record data). The first record within *idvar* is assumed to begin at time 0, and later records are assumed to begin where the previous record left off. *failvar* should contain 0 on all but, possibly, the last record within *idvar*, unless your data contain multiple failures (in which case you must specify the `exit()` option; see *Setting multiple failures* below).

◁

▷ Example 4: Multiple-record data with multiple events

We have the following data on hospital patients admitted to a particular ward:

```
patid     day     sex     x1      x2     code
  101       5       1      10      10      177
  101      13       1      20       8      286
  101      21       1      16      11      208
  101      24       1      11      17      401
  102       8       0      20      19      204
  102      18       0      19       1      401
  103     etc.
```

Variable `code` records various actions; code 401 indicates being discharged alive, and 402 indicates death. We `stset` this dataset by typing

```
. stset day, id(patid) fail(code==402)

Survival-time data settings

           ID variable: patid
         Failure event: code==402
Observed time interval: (day[_n-1], day]
     Exit on or before: failure
────────────────────────────────────────────────────────────────────
        243  total observations
          0  exclusions
────────────────────────────────────────────────────────────────────
        243  observations remaining, representing
         40  subjects
         15  failures in single-failure-per-subject data
      1,486  total analysis time at risk and under observation
                                        At risk from t =          0
                             Earliest observed entry t =          0
                                 Last observed exit t =         62
```

When you specify `failure(`*eventvar* `==`*#*`)`, the failure event is as specified. You may include a list of numbers following the equal signs. If failure were codes 402 and 403, you could specify `failure(code == 402 403)`. If failure were codes 402, 403, 404, 405, 406, 407, and 409, you could specify `failure(code == 402/407 409)`.

◁

▷ Example 5: Multiple-record data recording time rather than t

More reasonably, the hospital data in the above example would not contain days since admission but would contain admission and current dates. In the dataset below, `adday` contains the day of admission, and `curday` contains the ending date of the record, both recorded as number of days since the ward opened:

```
        patid     adday     curday     sex      x1      x2     code
          101       287        292       1      10      10      177
          101         .        300       1      20       8      286
          101         .        308       1      16      11      208
          101         .        311       1      11      17      401
          102       289        297       0      20      19      204
          102         .        307       0      19       1      401
          103      etc.
```

This is the same dataset as shown in example 4. Previously, the first record on patient 101 was recorded 5 days after admission. In this dataset, $292 - 287 = 5$. We would stset this dataset by typing

```
. stset curday, id(patid) fail(code==402) origin(time adday)

Survival-time data settings

         ID variable: patid
      Failure event: code==402
Observed time interval: (curday[_n-1], curday]
  Exit on or before: failure
  Time for analysis: (time-origin)
             Origin: time adday
_____

      243  total observations
        0  ignored because never entered
_____

      243  observations remaining, representing
       40  subjects
       15  failures in single-failure-per-subject data
    1,486  total analysis time at risk and under observation
                                    At risk from t =           0
                           Earliest observed entry t =         0
                               Last observed exit t =         62
```

origin() sets when a subject becomes at risk. It does this by defining analysis time.

When you specify stset *timevar*, ... origin(time *originvar*), analysis time is defined as $t = (timevar - originvar)/\texttt{scale()}$. In analysis-time units, subjects become at risk at $t = 0$. See *Two concepts of time* and *The substantive meaning of analysis time* below.   ◁

▷ Example 6: Multiple-record data with time recorded as a date

Even more reasonably, dates would not be recorded as integers 428, 433, and 453, meaning the number of days since the ward opened. The dates would be recorded as dates:

patid	addate	curdate	sex	x1	x2	code
101	18aug1998	23aug1998	1	10	10	177
101	.	31aug1998	1	20	8	286
101	.	08sep1998	1	16	11	208
101	.	11sep1998	1	11	17	401
102	20aug1998	28aug1998	0	20	19	204
102	.	07sep1998	0	19	1	401
103	*etc.*					

That, in fact, changes nothing. We still type what we previously typed:

```
. stset curdate, id(patid) fail(code==402) origin(time addate)
```

Stata dates are, in fact, integers—they are the number of days since 01jan1960—and it is merely Stata's %td display format that makes them display as dates.   ◁

▷ Example 7: Multiple-record data with extraneous information

Perhaps we wish to study the outcome after a certain operation, said operation being indicated by code 286. Subjects become at risk when the operation is performed. Here we do not type

```
. stset curdate, id(patid) fail(code==402) origin(time addate)
```

We instead type

```
. stset curdate, id(patid) fail(code==402) origin(code==286)
```

The result of typing this would be to set analysis time t to

$$t = \texttt{curdate} - (\text{the value of } \texttt{curdate} \text{ when } \texttt{code==286})$$

Let's work through this for the first patient:

patid	addate	curdate	sex	x1	x2	code
101	18aug1998	23aug1998	1	10	10	177
101	.	31aug1998	1	20	8	286
101	.	08sep1998	1	16	11	208
101	.	11sep1998	1	11	17	401

The event 286 occurred on 31aug1998, and thus the values of t for the four records are

$$t_1 = curdate_1 - 31\text{aug}1998 = 23\text{aug}1998 - 31\text{aug}1998 = -8$$
$$t_2 = curdate_2 - 31\text{aug}1998 = 31\text{aug}1998 - 31\text{aug}1998 = 0$$
$$t_3 = curdate_3 - 31\text{aug}1998 = 08\text{sep}1998 - 31\text{aug}1998 = 8$$
$$t_4 = curdate_4 - 31\text{aug}1998 = 11\text{sep}1998 - 31\text{aug}1998 = 11$$

Information prior to $t = 0$ is not relevant because the subject is not yet at risk. Thus the relevant data on this subject are

t in $(0, 8]$ $\texttt{sex} = 1$, $\texttt{x1} = 16$, $\texttt{x2} = 11$
t in $(8, 11]$ $\texttt{sex} = 1$, $\texttt{x1} = 11$, $\texttt{x2} = 17$, and the subject is censored ($\texttt{code} \neq 402$)

That is precisely the logic that stset went through. For your information, stset quietly creates the variables

_st	1 if the record is to be used, 0 if ignored
_t0	Analysis time when record begins
_t	Analysis time when record ends
_d	1 if failure, 0 if censored

You can examine these variables after issuing the stset command:

```
. list _st _t0 _t _d
```

	_st	_t0	_t	_d
1.	0	.	.	.
2.	0	.	.	.
203.	1	0	8	0
204.	1	8	11	0

Results are just as we anticipated. Do not let the observation numbers bother you; stset sorts the data in a way it finds convenient. Feel free to re-sort the data; if any of the st commands need the data in a different order, they will sort it themselves.

There are two ways of specifying origin():

origin(time *timevar*)   or   origin(time *exp*)
origin(*eventvar* == *numlist*)

In the first syntax—which is denoted by typing the word `time`—you directly specify when a subject becomes at risk. In the second syntax—which is denoted by typing a variable name and equal signs—you specify the same thing indirectly. The subject becomes at risk when the specified event occurs (which may be never).

Information prior to `origin()` is ignored. That information composes what we call the past history.

◁

▷ Example 8: Multiple-record data with delayed entry

In another analysis, we want to use the above data to analyze all patients, not just those undergoing a particular operation. In this analysis, subjects become at risk when they enter the ward. For this analysis, however, we need information from a particular test, and that information is available only if the test is administered to the patient. Even if the test is administered, some amount of time passes before that. Assume that when the test is administered, `code==152` is inserted into the patient's hospital record.

To summarize, we want `origin(time addate)`, but patients do not enter our sample until `code==152`. The way to `stset` these data is

```
. stset curdate, id(patid) fail(code==402) origin(time addate) enter(code==152)
```

Patient 107 has code 152:

patid	addate	curdate	sex	x1	x2	code
107	22aug1998	25aug1998	1	9	13	274
107	.	28aug1998	1	19	19	152
107	.	30aug1998	1	18	12	239
107	.	07sep1998	1	12	11	401

In analysis time, $t = 0$ corresponds to 22aug1998. The test was not administered, however, until 6 days later. The analysis times for these records are

$$t_1 = curdate_1 - 22aug1998 = 25aug1998 - 22aug1998 = 3$$

$$t_2 = curdate_2 - 22aug1998 = 28aug1998 - 22aug1998 = 6$$

$$t_3 = curdate_3 - 22aug1998 = 30aug1998 - 22aug1998 = 8$$

$$t_4 = curdate_4 - 22aug1998 = 07sep1998 - 22aug1998 = 16$$

and the data we want in our sample are

t in $(6, 8]$ sex $= 1$, x1 $= 18$, x2 $= 12$
t in $(8, 16]$ sex $= 1$, x1 $= 12$, x2 $= 11$, and patient was censored (code $\neq 402$)

The above `stset` command produced this:

```
. list _st _t0 _t _d
```

	_st	_t0	_t	_d
1.	0	.	.	.
2.	0	.	.	.
39.	1	6	8	0
40.	1	8	16	0

◁

▷ Example 9: Multiple-record data with extraneous information and delayed entry

The `origin()` and `enter()` options can be combined. For instance, we want to analyze patients receiving a particular operation (time at risk begins upon `code == 286`, but patients may not enter the sample before a test is administered, denoted by `code == 152`). We type

```
. stset curdate, id(patid) fail(code==402) origin(code==286) enter(code==152)
```

If we typed the above commands, it would not matter whether the test was performed before or after the operation.

A patient who had the test and then the operation would enter at analysis time $t = 0$.

A patient who had the operation and then the test would enter at analysis time $t > 0$, the analysis time being the time the test was performed.

If we wanted to require that the operation be performed after the test, we could type

```
. stset curdate, id(patid) fail(code==402) origin(code==286) after(code==152)
```

Admittedly, this can be confusing. The way to proceed is to find a complicated case in your data and then list `_st _t0 _t _d` for that case after you `stset` the data.

◁

▷ Example 10: Real data

All of our hospital ward examples are artificial in one sense: it is unlikely the data would have come to us in survival-time form:

```
patid     addate     curdate    sex     x1      x2     code
  101  18aug1998  23aug1998      1      10      10      177
  101          .  31aug1998      1      20       8      286
  101          .  08sep1998      1      16      11      208
  101          .  11sep1998      1      11      17      401
  102  20aug1998  28aug1998      0      20      19      204
  102          .  07sep1998      0      19       1      401
  103      etc.
```

Rather, we would have received a snapshot dataset:

```
patid       date   sex     x1      x2     code
  101  18aug1998     1      10      10       22
  101  23aug1998     .      20       8      177
  101  31aug1998     .      16      11      286
  101  08sep1998     .      11      17      208
  101  11sep1998     .       .       .      401
  102  20aug1998     0      20      19       22
  102  28aug1998     .      19       1      204
  102  07sep1998     .       .       .      401
  103  etc.
```

In a snapshot dataset, we have a time (here a date) and values of the variables as of that instant.

This dataset can be converted to the appropriate form by typing

```
. snapspan patid date code
```

The result would be as follows:

```
patid        date     sex      x1      x2     code
  101   18aug1998       .       .       .       22
  101   23aug1998       1      10      10      177
  101   31aug1998       .      20       8      286
  101   08sep1998       .      16      11      208
  101   11sep1998       .      11      17      401
  102   20aug1998       .       .       .       22
  102   28aug1998       0      20      19      204
  102   07sep1998       .      19       1      401
```

This is virtually the same dataset with which we have been working, but it differs in two ways:

1. The variable sex is not filled in for all the observations because it was not filled in on the original form. The hospital wrote down the sex on admission and then never bothered to document it again.

2. We have no admission date (addate) variable. Instead, we have an extra first record for each patient with code $= 22$ (22 is the code the hospital uses for admissions).

The first problem is easily fixed, and the second, it turns out, is not a problem because we can vary what we type when we stset the data.

First, let's fix the problem with variable sex. There are two ways to proceed. One would be simply to fill in the variable ourselves:

```
. by patid (date), sort: replace sex = sex[_n-1] if sex>=.
```

We could also perform a phony stset that is good enough to set all the data and then use stfill to fill in the variable for us. Let's begin with the phony stset:

```
. stset date, id(patid) origin(min) fail(code==-1)

Survival-time data settings

           ID variable:  patid
         Failure event:  code==-1
Observed time interval:  (date[_n-1], date]
     Exit on or before:  failure
     Time for analysis:  (time-origin)
                Origin:  min

_____

      283  total observations
        0  exclusions
_____

      283  observations remaining, representing
       40  subjects
        0  failures in single-failure-per-subject data
    2,224  total analysis time at risk and under observation
                                     At risk from t =          0
                              Earliest observed entry t =       0
                                 Last observed exit t =         89
```

Typing stset date, id(patid) origin(min) fail(code == -1) does not produce anything we would want to use for analysis. This is a trick to get the dataset temporarily stset so that we can use some st data management commands on it.

The first part of the trick is to specify origin(min). This defines the analysis time as $t = 0$, corresponding to the minimum observed value of the time variable minus 1. The time variable is date here. Why the minimum minus 1? Because st ignores observations for which analysis time $t < 0$. origin(min) provides a phony definition of t that ensures $t > 0$ for all observations.

The second part of the trick is to specify `fail(code == -1)`, and you might have to vary what you type. We just wanted to choose an event that we know never happens, thus ensuring that no observations are ignored after failure.

Now that we have the dataset `stset`, we can use the other st commands. Do not use the st analysis commands unless you want ridiculous results, but one of the st data management commands is just what we want:

```
. stfill sex, forward

        Failure _d: code==-1
   Analysis time _t: (date-origin)
           Origin: min
      ID variable: patid

Replacing missing values with previously observed values:
            sex: 203 real changes made
```

Problem one solved.

The second problem concerns the lack of an admission-date variable. That is not really a problem because we have a new first observation with code $= 22$ recording the date of admission, so every place we previously coded `origin(time addate)`, we substitute `origin(code == 22)`.

Problem two solved.

We also solved the big problem—converting a snapshot dataset into a survival-time dataset; see [ST] **snapspan**.

◁

Two concepts of time

The st system has two concepts of time. The first is *time* in italics, which corresponds to how time is recorded in your data. The second is analysis time, which we write as t. Substantively, analysis time is time at risk. `stset` defines analysis time in terms of *time* via

$$t = \frac{time - \texttt{origin()}}{\texttt{scale()}}$$

t and *time* can be the same thing, and by default they are because, by default, `origin()` is 0 and `scale()` is 1.

All the st analysis commands work with analysis time, t.

By default, if you do not specify the `origin()` and `scale()` options, your time variables are expected to be the analysis-time variables. This means that *time* $= 0$ corresponds to when subjects became at risk, and that means, among other things, that observations for which *time* < 0 are ignored because survival analysis concerns persons who are at risk, and no one is at risk before $t = 0$.

`origin` determines when the clock starts ticking. If you do not specify `origin()`, `origin(time 0)` is assumed, meaning that $t = time$ and that persons are at risk from $t = time = 0$.

time and t will often differ. *time* might be calendar time and t the length of time since some event, such as being born or being exposed to some risk factor. `origin()` sets when $t = 0$. `scale()` merely sets a constant that makes t more readable.

The syntax for the origin() option makes it look more complicated than it really is

$$\texttt{origin(} \big[\, \textit{varname} \mathbin{==} \textit{numlist} \,\big]\ \texttt{time}\ \textit{exp}\ |\ \texttt{min)}$$

This says that there are four different ways to specify origin():

> origin(time *exp*)
> origin(*varname* == *numlist*)
> origin(*varname* == *numlist* time *exp*)
> origin(min)

The first syntax can be used with single- or multiple-record data. It states that the origin is given by *exp*, which can be a constant for all observations, a variable (and hence may vary subject by subject), or even an expression composed of variables and constants. Perhaps the origin is a fixed date or a date recorded in the data when the subject was exposed or when the subject turned 18.

The second and third syntaxes are for use with multiple-record data. The second states that the origin corresponds to the (earliest) time when the designated event occurred. Perhaps the origin is when an operation was performed. The third syntax calculates the origin both ways and then selects the later one.

The fourth syntax does something odd; it sets the origin to the minimum time observed minus 1. This is not useful for analysis but is sometimes useful for playing data management tricks; see example 10 above.

Let's start with the first syntax. Say that you had the data

```
    faildate     x1     x2
    28dec1997    12     22
    12nov1997    15     22
    03feb1998    55     22
```

and that all the observations came at risk on the same date, 01nov1997. You could type

```
    . stset faildate, origin(time mdy(11,1,1997))
```

Remember that stset adds _t0 and _t to your dataset and that they contain the time span for each record, documented in analysis-time units. After typing stset, you can list the results:

```
    . list faildate x1 x2 _t0 _t
```

	faildate	x1	x2	_t0	_t
1.	28dec1997	12	22	0	57
2.	12nov1997	15	22	0	11
3.	03feb1998	55	22	0	94

Record 1 reflects the period $(0, 57\,]$ in analysis-time units, which are days here. stset calculated the 57 from $28\text{dec}1997 - 01\text{nov}1997 = 13{,}876 - 13{,}819 = 57$. (Dates such as 28dec1997 are really just integers containing the number of days from 01jan1960, and Stata's %td display format makes them display nicely. 28dec1997 is really the number 13,876.)

As another example, we might have data recording exposure and failure dates:

```
      expdate    faildate     x1     x2
    07may1998   22jun1998     12     22
    02feb1998   11may1998     11     17
```

The way to `stset` this dataset is

```
. stset faildate, origin(time expdate)
```

and the result, in analysis units, is

```
. list expdate faildate x1 x2 _t0 _t
```

	expdate	faildate	x1	x2	_t0	_t
1.	07may1998	22jun1998	12	22	0	46
2.	02feb1998	11may1998	11	17	0	98

There is nothing magical about dates. Our original data could just as well have been

expdate	faildate	x1	x2
32	78	12	22
12	110	11	17

and the result would still be the same because $78 - 32 = 46$ and $110 - 12 = 98$.

Specifying an expression can sometimes be useful. Suppose that your dataset has the variable `date` recording the date of event and variable `age` recording the subject's age as of `date`. You want to make $t = 0$ correspond to when the subject turned 18. You could type `origin(time date-int((age-18)*365.25))`.

`origin(varname == numlist)` is for use with multiple-record data. It states when each subject became at risk indirectly; the subject became at risk at the earliest time that *varname* takes on any of the enumerated values. Say that you had

patid	date	x1	x2	event
101	12nov1997	15	22	127
101	28dec1997	12	22	155
101	03feb1998	55	22	133
101	05mar1998	14	22	127
101	09apr1998	12	22	133
101	03jun1998	13	22	101
102	22nov1997	.	.	.

and assume event $= 155$ represents the onset of exposure. You might `stset` this dataset by typing

```
. stset date, id(patid) origin(event==155) ...
```

If you did that, the information for patient 101 before 28dec1997 would be ignored in subsequent analysis. The prior information would not be removed from the dataset; it would just be ignored. Probably something similar would happen for patient 102, or if patient 102 has no record with event $= 155$, all the records on the patient would be ignored.

For analysis time, $t = 0$ would correspond to when event 155 occurred. Here are the results in analysis-time units:

patid	date	x1	x2	event	_t0	_t
101	12nov1997	15	22	127	.	.
101	28dec1997	12	22	155	.	.
101	03feb1998	55	22	133	0	37
101	05mar1998	14	22	127	37	67
101	09apr1998	12	22	133	67	102
101	03jun1998	13	22	101	102	157
102	22nov1997	.	.	.	.	.

Patient 101's second record is excluded from the analysis. That is not a mistake. Records document durations, date reflects the end of the period, and events occur at the end of periods. Thus event 155 occurred at the instant date = 28dec1997, and the relevant first record for the patient is $(28\text{dec}1997, 03\text{feb}1998]$ in time units, which is $(0, 37]$ in t units.

The substantive meaning of analysis time

In specifying origin(), you must ask yourself whether two subjects with identical characteristics face the same risk of failure. The answer is that they face the same risk when they have the same value of $t = (time - \text{origin}())/\text{scale}()$ or, equivalently, when the same amount of time has elapsed from origin().

Say that we have the following data on smokers who have died:

ddate	x1	x2	reason
11mar1984	23	11	2
15may1994	21	9	1
22nov1993	22	13	2
etc.			

We wish to analyze death due to reason==2. However, typing

```
. stset ddate, fail(reason==2)
```

would probably not be adequate. We would be saying that smokers were at risk of death from 01jan1960. Would it matter? It would if we planned on doing anything parametric because parametric hazard functions, except for the exponential, are functions of analysis time, and the location of 0 makes a difference.

Even if we were thinking of performing nonparametric analysis, there would probably be difficulties. We would be asserting that two "identical" persons (in terms of x1 and x2) face the same risk on the same calendar date. Does the risk of death due to smoking really change as the calendar changes?

It would be more reasonable to assume that the risk changes with how long a subject has been smoking and that our data would probably include that date. We would type

```
. stset ddate, fail(reason==2) origin(time smdate)
```

if smdate were the name of the date-started-smoking variable. We would now be saying that the risk is equal when the number of days smoked is the same. We might prefer to see t in years,

```
. stset ddate, fail(reason==2) origin(time smdate) scale(365.25)
```

but that would make no substantive difference.

Consider single-record data on firms that went bankrupt:

incorp	bankrupt	x1	x2	btype
22jan1983	11mar1984	23	11	2
17may1992	15may1994	21	9	1
03nov1991	22nov1993	22	13	2
etc.				

Say that we wish to examine the risk of a particular kind of bankruptcy, btype == 2, among firms that become bankrupt. Typing

```
. stset bankrupt, fail(btype==2)
```

would be more reasonable than it was in the smoking example. It would not be reasonable if we were thinking of performing any sort of parametric analysis, of course, because then location of $t = 0$ would matter, but it might be reasonable for semiparametric analysis. We would be asserting that two "identical" firms (with respect to the characteristics we model) have the same risk of bankruptcy when the calendar dates are the same. We would be asserting that the overall state of the economy matters.

It might be reasonable to instead measure time from the date of incorporation:

```
. stset bankrupt, fail(btype==2) origin(time incorp)
```

Understand that the choice of `origin()` is a substantive decision.

Setting the failure event

You set the failure event by using the `failure()` option.

In single-record data, if `failure()` is not specified, every record is assumed to end in a failure. For instance, with

	failtime	load	bearings
1.	100	15	0
2.	140	15	1
etc.			

you would type `stset failtime`, and the first observation would be assumed to fail at time $= 100$; the second, at time $= 140$; and so on.

`failure(`*varname*`)` specifies that a failure occurs whenever *varname* is not zero and is not missing. For instance, with

	failtime	load	bearings	burnout
1.	100	15	0	1
2.	140	15	1	0
3.	97	20	0	1
4.	122	20	1	0
5.	84	25	0	1
6.	100	25	1	1
etc.				

you might type `stset failtime, failure(burnout)`. Observations 1, 3, 5, and 6 would be assumed to fail at times 100, 97, 84, and 100, respectively; observations 2 and 4 would be assumed to be censored at times 140 and 122.

Similarly, if the data were

	failtime	load	bearings	burnout
1.	100	15	0	1
2.	140	15	1	0
3.	97	20	0	2
4.	122	20	1	.
5.	84	25	0	2
6.	100	25	1	3
etc.				

the result would be the same. Nonzero, nonmissing values of the failure variable are assumed to represent failures. (Perhaps `burnout` contains a code on how the burnout occurred.)

`failure(`*varname* `== `*numlist*`)` specifies that a failure occurs whenever *varname* takes on any of the values of *numlist*. In the above example, specifying

```
. stset failtime, failure(burnout==1 2)
```

would treat observation 6 as censored.

```
. stset failtime, failure(burnout==1 2 .)
```

would also treat observation 4 as a failure.

```
. stset failtime, failure(burnout==1/3 6 .)
```

would treat `burnout==1`, `burnout==2`, `burnout==3`, `burnout==6`, and `burnout==.` as representing failures and all other values as representing censorings. (Perhaps we want to examine "failure due to meltdown", and these are the codes that represent the various kinds of meltdown.)

`failure()` is treated the same way in both single- and multiple-record data. Consider

```
         patno      t      x1      x2     died
  1.        1       4      23      11       1
  2.        2       5      21       9       0
  3.        2       8      22      13       1
  4.        3       7      20       5       0
  5.        3       9      22       5       0
  6.        3      11      21       5       0
  7.        4     ...
```

Typing

```
. stset t, id(patno) failure(died)
```

would treat

```
         patno==1   as dying            at t==4
         patno==2   as dying            at t==8
         patno==3   as being censored   at t==11
```

Intervening records on the same subject are marked as "censored". Technically, they are not really censored if you think about it carefully; they are simply marked as not failing. Look at the data for subject 3:

```
         patno      t      x1      x2     died
           3        9      22       5       0
           3       11      21       5       0
```

The subject is not censored at $t = 9$ because there are more data on the subject; it is merely the case that the subject did not die at that time. At $t = 9$, x1 changed from 22 to 21. The subject is really censored at $t = 11$ because the subject did not die and there are no more records on the subject.

Typing `stset t, id(patno) failure(died)` would mark the same persons as dying and the same persons as censored, as in the previous case. If `died` contained not 0 and 1, but 0 and nonzero, nonmissing codes for the reason for death would be

```
         patno      t      x1      x2     died
  1.        1       4      23      11      103
  2.        2       5      21       9       0
  3.        2       8      22      13      207
  4.        3       7      20       5       0
  5.        3       9      22       5       0
  6.        3      11      21       5       0
  7.        4     ...
```

Typing

```
. stset t, id(patno) failure(died)
```

or

```
. stset t, id(patno) failure(died==103 207)
```

would yield the same results; subjects 1 and 2 would be treated as dying and subject 3 as censored.

Typing

```
. stset t, id(patno) failure(died==207)
```

would treat subject 2 as dying and subjects 1 and 3 as censored. Thus when you specify the values for the code, the code variable need not ever contain 0. In

	patno	t	x1	x2	died
1.	1	4	23	11	103
2.	2	5	21	9	13
3.	2	8	22	13	207
4.	3	7	20	5	11
5.	3	9	22	5	12
6.	3	11	21	5	12
7.	4	...			

typing

```
. stset t, id(patno) failure(died==207)
```

treats patient 2 as dying and 1 and 3 as censored. Typing

```
. stset t, id(patno) failure(died==103 207)
```

treats patients 1 and 2 as dying and 3 as censored.

Setting multiple failures

In multiple-record data, records after the first failure event are ignored unless you specify the `exit()` option. Consider the following data:

	patno	t	x1	x2	code
1.	1	4	21	7	14
2.	1	5	21	7	11
3.	1	7	20	7	17
4.	1	8	22	7	22
5.	1	9	22	7	22
6.	1	11	21	7	29
7.	2	...			

Perhaps code 22 represents the event of interest—say, the event "visited the doctor". Were you to type `stset t, id(patno) failure(code == 22)`, the result would be as if the data contained

	patno	t	x1	x2	code
1.	1	4	21	7	14
2.	1	5	21	7	11
3.	1	7	20	7	17
4.	1	8	22	7	22

Records after the first occurrence of the failure event are ignored. If you do not want this, you must specify the `exit()` option. Probably you would want to specify `exit(time .)`, here meaning that subjects are not to exit the risk group until their data run out. Alternatively, perhaps code 142 means "entered the nursing home" and, once that event happens, you no longer want them in the risk group. Then you would code `exit(code == 142)`; see *Final exit times* below.

First entry times

Do not confuse `enter()` with `origin()`. `origin()` specifies when a subject first becomes at risk. `enter()` specifies when a subject first comes under observation. In most datasets, becoming at risk and coming under observation are coincident. Then it is sufficient to specify `origin()` alone, although you could specify both options.

Some persons enter the data after they have been at risk of failure. Say that we are studying deaths due to exposure to substance X and we know the date at which a person was first exposed to the substance. We are willing to assume that persons are at risk from the date of exposure forward. A person arrives at our door who was exposed 15 years ago. Can we add this person to our data? The statistical issue is labeled *left-truncation*, and the problem is that had the person died before arriving at our door, we would never have known about her. We can add her to our data, but we must be careful to treat her subsequent survival time as conditional on having already survived 15 years.

Say that we are examining visits to the widget repair facility, "failure" being defined as a visit (so failures can be repeated). The risk begins once a person buys a widget. We have a woman who bought a widget 3 years ago, and she has no records on when she has visited the facility in the last 3 years. Can we add her to our data? Yes, as long as we are careful to treat her subsequent behavior as already being 3 years after she first became at risk.

The jargon for this is "under observation". All this means is that any failures would be observed. Before being under observation, failure would not be observed.

If `enter()` is not specified, we assume that subjects are under observation at the time they enter the risk group as specified by `origin()`, 0 if `origin()` is not specified, or possibly `time0()`. To be precise, subject i is assumed to first enter the analysis risk pool at

$$time_i = \max\big(\text{earliest time0() for } i, \text{enter(), origin()}\big)$$

Say that we have multiple-record data recording "came at risk" (`mycode == 1`), "enrolled in our study" (`mycode == 2`), and "failed because of risk" (`mycode == 3`). We `stset` this dataset by typing

```
. stset time, id(id) origin(mycode==1) enter(mycode==2) failure(mycode==3)
```

The above `stset` correctly handles the came at risk/came under observation problem regardless of the order of events 1, 2, and 3. For instance, if the subject comes under observation before he or she becomes at risk, the subject will be treated as entering the analysis risk pool at the time he or she came at risk.

Say that we have the same data in single-record format: variable `riskdate` documents becoming at risk and variable `enr_date` the date of enrollment in our study. We would `stset` this dataset by typing

```
. stset time, origin(time riskdate) enter(time enr_date) failure(mycode==3)
```

For a final example, let's return to the multiple-record way of recording our data and say that we started enrolling people in our study on 12jan1998 but that, up until 16feb1998, we do not trust that our records are complete (we had start-up problems). We would `stset` that dataset by typing

```
. stset time, origin(mycode==1) enter(mycode==2 time mdy(2,16,1998))
> fail(mycode==3)
```

`enter(`*varname*`==`*numlist* `time` *exp*`)` is interpreted as

$$\max(\text{time of earliest event in } \textit{numlist}, \textit{exp})$$

Thus persons having `mycode == 2` occurring before 16feb1998 are assumed to be under observation from 16feb1998, and those having `mycode == 2` thereafter are assumed to be under observation from the time of `mycode == 2`.

Final exit times

exit() specifies the latest time under which the subject is both under observation and at risk of the failure event. The emphasis is on latest; obviously subjects also exit the data when their data run out.

When you type

```
. stset ..., ... failure(outcome==1/3 5) ...
```

the result is as if you had typed

```
. stset ..., ... failure(outcome==1/3 5) exit(failure) ...
```

which, in turn, is the same as

```
. stset ..., ... failure(outcome==1/3 5) exit(outcome==1/3 5) ...
```

When are people to be removed from the analysis risk pool? When their data end, of course, and when the event 1, 2, 3, or 5 first occurs. How are they to be removed? According to their status at that time. If the event is 1, 2, 3, or 5 at that instant, they exit as a failure. If the event is something else, they exit as censored.

Perhaps events 1, 2, 3, and 5 represent death due to heart disease, and that is what we are studying. Say that outcome == 99 represents death for some other reason. Obviously, once the person dies, she is no longer at risk of dying from heart disease, so we would want to specify

```
. stset ..., ... failure(outcome==1/3 5) exit(outcome==1/3 5 99) ...
```

When we explicitly specify exit(), we must list all the reasons for which a person is to be removed other than simply running out of data. Here it would have been a mistake to specify just exit(99) because that would have left persons in the analysis risk pool who died for reasons 1, 2, 3, and 5. We would have treated those people as if they were still at risk of dying.

In fact, it probably would not have mattered had we specified exit(99) because, once a person is dead, he or she is unlikely to have any subsequent records anyway. By that logic, we did not even have to specify exit(99) because death is death and there should be no records following it.

For other kinds of events, however, exit() becomes important. Let's assume that the failure event is to be diagnosed with heart disease. A person may surely have records following diagnosis, but even so,

```
. stset ..., ... failure(outcome==22) ...
```

would be adequate because, by not specifying exit(), we are accepting the default that exit() is equivalent to failure(). Once outcome 22 occurs, subsequent records on the subject will be ignored—they constitute the future history of the subject.

Say, however, that we wish to treat as censored persons diagnosed with kidney disease. We would type

```
. stset ..., ... failure(outcome==22) exit(outcome==22 29) ...
```

assuming that outcome = 29 is "diagnosed with kidney disease". It is now of great importance that we specified exit(outcome==22 29) and not just exit(outcome==29) because, had we omitted code 22, persons would have remained in the analysis risk pool even after the failure event, that is, being diagnosed with heart disease.

If, in addition, our data were untrustworthy after 22nov1998 (perhaps not all the data have been entered yet), we would type

```
. stset ..., ... failure(outcome==22) exit(outcome==22 29 time
> mdy(11,22,1998)) ...
```

If we type exit(*varname*==*numlist* time *exp*), the exit time is taken to be

$$\min(\text{time of earliest event in } \textit{numlist}, \textit{exp})$$

For some analyses, repeated failures are possible. If you have repeated failure data, you specify the exit() option and include whatever reasons, if any, that would cause the person to be removed. If there are no such reasons and you wish to retain all observations for the person, you type

```
. stset ..., ... exit(time .) ...
```

exit(time .) specifies that the maximum time a person can be in the risk pool is infinite; thus subjects will not be removed until their data run out.

Intermediate exit and reentry times (gaps)

Gaps arise when a subject is temporarily not under observation. The statistical importance of gaps is that, if failure is death and if the person died during such a gap, he would not have been around to be found again. The solution to this is to remove the person from the risk pool during the observational gap.

To determine that you have gaps, your data must provide starting and ending times for each record. Most datasets provide only ending times, making it impossible to know that you have gaps.

You use time0() to specify the beginning times of records. time0() specifies a mechanical aspect of interpreting the records in the dataset, namely, the beginning of the period spanned by each record. Do not confuse time0() with origin(), which specifies when a subject became at risk, or with enter(), which specifies when a subject first comes under observation.

time0() merely identifies the beginning of the time span covered by each record. Say that we had two records on a subject, the first covering the span (40,49] and the second, (49,57]:

```
        |<── record 1 ──>|<- record 2 ->|
   ─────────────────────────────────────────> time
        40               49             57
```
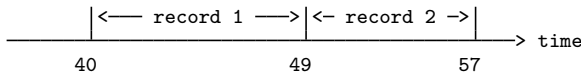
A time0() variable would contain

$$\text{40 in record 1}$$
$$\text{49 in record 2}$$

and not, for instance, 40 and 40. A time0() variable varies record by record for a subject.

Most datasets merely provide an end-of-record time value, *timevar*, which you specify by typing stset *timevar*, ... When you have multiple records per subject and you do not specify a time0() variable, stset assumes that each record begins where the previous one left off.

if( ) versus if exp

Both the if *exp* qualifier and the if (*exp*) option select records for which *exp* is true. We strongly recommend specifying the if() option in preference to the if *exp* qualifier. They differ in that the if *exp* qualifier removes data from consideration before calculating beginning and ending times and other quantities as well. The if() option, on the other hand, sets the restriction after all derived variables are calculated. To appreciate this difference, consider the following multiple-record data:

```
patno       t      x1      x2     code
    3       7      20       5       14
    3       9      22       5       23
    3      11      21       5       29
```

Consider the difference in results between typing

```
. stset t if x1!=22, failure(code==14)
```

and

```
. stset t,  if(x1!=22) failure(code==14)
```

The first would remove record 2 from consideration at the outset. In constructing beginning and ending times, stset and streset would see

```
      patno       t       x1       x2      code
          3        7       20        5        14
          3       11       21        5        29
```

and would construct the result:
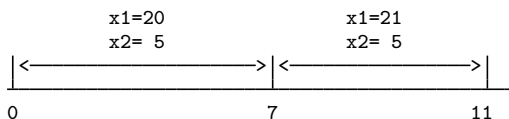
In the second case, the result would be
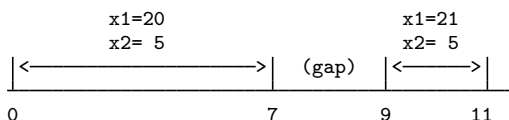
The latter result is correct and the former incorrect because $x1 = 21$ is not true in the interval $(7, 9)$.

The only reason to specify the if *exp* qualifier is to ignore errors in the data—observations that would confuse stset and streset—without actually dropping the offending observations from the dataset.

You specify the if() option to exclude observations that are not themselves errors. Specifying if() yields the same result as specifying if *exp* on the subsequent st commands after the dataset has been stset.

Past and future records

Consider the hospital ward data that we have seen before:

```
    patid     addate     curdate     sex     x1     x2     code
      101  18aug1998  23aug1998       1      10     10      177
      101          .  31aug1998       1      20      8      286
      101          .  08sep1998       1      16     11      208
      101          .  11sep1998       1      11     17      401
      etc.
```

Say that you stset this dataset such that you selected the middle two records. Perhaps you typed

```
. stset curdate, id(patid) origin(time addate) enter(code==286) failure(code==208)
```

The first record for the subject, because it was not selected, is called a *past history record*. Any earlier records that were not selected would also be called past history records.

The last record for the subject, because it was not selected, is called a *future history record*. Any later records that were not selected would also be called future history records.

If you typed

```
. streset, past
```

the first three records for this subject would be selected.

Typing

```
. streset, future
```

would select the last three records for this subject.

If you typed

```
. streset, past future
```

all four records for this subject would be selected.

If you then typed

```
. streset
```

the original two records would be selected, and things would be back just as they were before.

After typing `streset, past`; `streset, future`; or `streset, past future`, you would not want to use any analysis commands. `streset` did some strange things, especially with the analysis-time variable, to include the extra records. It would be the wrong sample, anyway.

You might, however, want to use certain data management commands on the data, especially those for creating new variables.

Typically, `streset, past` is of greater interest than the other commands. Past records—records prior to being at risk or excluded for other reasons—are not supposed to play a role in survival analysis. `stset` makes sure they do not. But it is sometimes reasonable to ask questions about them such as, was the subject ever on the drug cisplatin? Has the subject ever been married? Did the subject ever have a heart attack?

To answer questions like that, you sometimes want to dig into the past. Typing `streset, past` makes that easy, and once the past is set, the data can be used with `stgen` and `st_is 2`. You might well type the following:

```
. stset curdate, id(patid) origin(addate) enter(code==286) failure(code==208)
. streset, past
. stgen attack = ever(code==177)
. streset
. stcox attack ...
```

Do not be concerned about doing something inappropriate while having the past or future set; st will not let you:

```
. stset curdate, id(patid) origin(time addate) enter(code==286) failure(code==208)
  (output omitted)
. streset, past
  (output omitted)
. stcox x1
you last streset, past
you must type streset to restore the analysis sample
r(119);
```

Using streset

`streset` is a useful tool for gently modifying what you have previously `stset`. Rather than typing the whole `stset` command, you can type `streset` followed just by what has changed.

For instance, you might type

```
. stset curdate, id(patid) origin(time addate) enter(code==286) failure(code==208)
```

and then later want to restrict the analysis to subjects who ever have x1>20. You could retype the whole `stset` command and add `ever(x1>20)`, but it would be easier to type

```
. streset, ever(x1>20)
```

If later you decide you want to remove the restriction, type

```
. streset, ever(.)
```

That is the general rule for resetting options to the default: type '.' as the option's argument.

Be careful using `streset` because you can make subtle mistakes. In another analysis with another dataset, consider the following:

```
. stset date, fail(code==2) origin(code==1107)
. ...
. streset date, fail(code==9) origin(code==1422) after(code==1423)
. ...
. streset, fail(code==2) origin(code==1107)
```

If, in the last step, you are trying to get back to the results of the first `stset`, you will fail. The last `streset` is equivalent to

```
. stset date, fail(code==9) origin(code==1107) after(code==1423)
```

`streset()` remembers the previously specified options and uses them if you do not override them. Both `stset` and `streset` display the current command line. Make sure that you verify that the command is what you intended.

Performance and multiple-record-per-subject datasets

`stset` and `streset` do not drop data; they simply mark data to be excluded from consideration. Some survival-time datasets can be large, although the relevant subsamples are small. In such cases, you can reduce memory requirements and speed execution by dropping the irrelevant observations.

`stset` and `streset` mark the relevant observations by creating a variable named $\_st$ (it is always named this) containing 1 and 0; $\_st = 1$ marks the relevant observations and $\_st = 0$ marks the irrelevant ones. If you type

```
. drop if _st==0
```

or equivalently

```
. keep if _st==1
```

or equivalently

```
. keep if _st
```

you will drop the irrelevant observations. All st commands produce the same results whether you do this or not. Be careful, however, if you are planning future `stset`s or `streset`s. Observations that are irrelevant right now might be relevant later.

One solution to this conundrum is to keep only those observations that are relevant after setting the entire history:

```
. stset date, fail(code==9) origin(code==1422) after(code==1423)
. streset, past future
. keep if _st
. streset
```

As a final note, you may drop the irrelevant observations as marked by $\_st = 0$, but do not drop the $\_st$ variable itself. The other st commands expect to find variable $\_st$.

Sequencing of events within t

Consider the following bit of data:

etime	failtime	fail
0	5	1
0	5	0
5	7	1

Note all the different events happening at time 5: the first observation fails, the second is censored, and the third enters.

What does it mean for something to happen at time 5? In particular, is it at least potentially possible for the second observation to have failed at time 5; that is, was it in the risk group when the first observation failed? How about the third observation? Was it in the risk group, and could it have potentially failed at time 5?

Stata sequences events within a time as follows:

first,   at time t the failures occur
then,   at time $t + 0$ the censorings are removed from the risk group
finally, at time $t + 0 + 0$ the new entries are added to the risk group

Thus, to answer the questions:

Could the second observation have potentially failed at time 5? Yes.

Could the third observation have potentially failed at time 5? No, because it was not yet in the risk group.

By this logic, the following makes no sense:

etime	failtime	fail
5	5	1

This would mark a subject as failing before being at risk. It would make no difference if `fail` were 0—the subject would then be marked as being censored too soon. Either way, `stset` would flag this as an error. If you had a subject who entered and immediately exited, you would code this as

etime	failtime	fail
4.99	5	1

Weights

`stset` allows you to specify `fweight`s, `pweight`s, and `iweight`s.

`fweight`s are Stata's frequency or replication weights. Consider the data

failtime	load	bearings	count
100	15	0	3
140	15	1	2
97	20	0	1

and the `stset` command

```
. stset failtime [fw=count]

Survival-time data settings

         Failure event: (assumed to fail at time=failtime)
Observed time interval: (0, failtime]
    Exit on or before: failure
               Weight: [fweight=count]
_____

          3  total observations
          0  exclusions
_____

          3  physical observations remaining, equal to
          6  weighted observations, representing
          6  failures in single-record/single-failure data
        677  total analysis time at risk and under observation
                                          At risk from t =          0
                                Earliest observed entry t =          0
                                      Last observed exit t =        140
```

This combination is equivalent to the expanded data

failtime	load	bearings
100	15	0
100	15	0
100	15	0
140	15	1
140	15	1
97	20	0

and the command

```
. stset failtime
```

pweights are Stata's sampling weights—the inverse of the probability that the subject was chosen from the population. pweights are typically integers, but they do not have to be. For instance, you might have

time0	time	died	sex	reps
0	300	1	0	1.50
0	250	0	1	4.50
30	147	1	0	2.25

Here `reps` is how many patients each observation represents in the underlying population—perhaps when multiplied by 10. The `stset` command for these data is

```
. stset time [pw=reps], origin(time time0) failure(died)
```

For variance calculations, the scale of the pweights does not matter. `reps` in the 3 observations shown could just as well be 3, 9, and 4.5. Nevertheless, the scale of the pweights is used when you ask for counts. For instance, `stsum` would report the person-time at risk as

$$(300 - 0)\, 1.5 + (250 - 0)\, 4.5 + (147 - 30)\, 2.25 = 1{,}838.25$$

for the 3 observations shown. `stsum` would count that $1.5 + 2.25 = 3.75$ persons died, and so the incidence rate for these 3 observations would be $3.75/1{,}838.25 = 0.0020$. The incidence rate is thus unaffected by the scale of the weights. Similarly, the coefficients and confidence intervals reported by, for instance, `streg, dist(exponential)` would be unaffected. The 95% confidence interval for the incidence rate would be $[0.0003, 0.0132]$, regardless of the scale of the weights.

If these 3 observations were examined unweighted, the incidence rate would be 0.0030 and the 95% confidence interval would be $[0.0007, 0.0120]$.

Finally, stset allows you to set iweights, which are Stata's "importance" weights, but we recommend that you do not. iweights are provided for those who wish to create special effects by manipulating standard formulas. The st commands treat iweights just as they would fweights, although they do not require that the weights be integers, and push their way through conventional variance calculations. Thus results—counts, rates, and variances—depend on the scale of these weights.

Data warnings and errors flagged by stset

When you stset your data, stset runs various checks to verify that what you are setting makes sense. stset refuses to set the data only if, in multiple-record, weighted data, weights are not constant within ID. Otherwise, stset merely warns you about any inconsistencies that it identifies.

Although stset will set the data, it will mark out records that it cannot understand; for instance,

```
. stset curdate, origin(time addate) failure(code==402) id(patid)

          ID variable: patid
       Failure event: code==402
Observed time interval: (curdate[_n-1], curdate]
    Exit on or before: failure
    Time for analysis: (time-origin)
               Origin: time addate
_____

     243  total observations
       1  event time missing (curdate>=.)               PROBABLE ERROR
       4  multiple records at same instant              PROBABLE ERROR
          (curdate[_n-1]==curdate)
_____

     238  observations remaining, representing
      40  subjects
      15  failures in single failure-per-subject data
   1,478  total analysis time at risk and under observation
                                        At risk from t =          0
                                Earliest observed entry t =       0
                                   Last observed exit t =        62
```

You must ensure that the result, after exclusions, is correct.

The warnings stset might issue include

```
          ignored because patid missing
          event time missing                          PROBABLE ERROR
          entry time missing                          PROBABLE ERROR
          entry on or after exit (etime>t)            PROBABLE ERROR
          observations end on or before enter()
          observations end on or before origin()
          multiple records at same instant (t[_n-1]==t)   PROBABLE ERROR
          overlapping records (t[_n-1]>entry time)    PROBABLE ERROR
          weights invalid                             PROBABLE ERROR
```

stset sets $\_st = 0$ when observations are excluded for whatever reason. Thus observations with any of the above problems can be found among the $\_st = 0$ observations.

Using survival-time data in Stata

In the examples above, we have shown you how Stata expects survival-time data to be recorded. To summarize:

- Each subject's history is represented by 1 or more observations in the dataset.

- Each observation documents a span of time. The observation must contain when the span ends (exit time) and may optionally contain when the span begins (entry time). If the entry time is not recorded, it is assumed to be 0 or, in multiple-record data, the exit time of the subject's previous observation, if there is one. By *previous*, we mean that the data have already been temporally ordered on exit times within subject. The physical order of the observations in your dataset does not matter.

- Each observation documents an outcome associated with the exit time. Unless otherwise specified with failure(), 0 and missing mean censored, and nonzero means failed.

- Each observation contains other variables (called covariates) that are assumed to be constant over the span of time recorded by the observation.

Data rarely arrive in this neatly organized form. For instance, Kalbfleisch and Prentice (2002, 4–5) present heart transplant survival data from Stanford (Crowley and Hu 1977). These data can be converted into the correct st format in at least two ways. The first method is shown in example 11. A second, shorter, method using the st commands is described in example 3 of [ST] **stsplit**.

▷ Example 11

Here we will describe the process that uses the standard Stata commands.

```
. use https://www.stata-press.com/data/r18/stan2, clear
(Heart transplant data)

. describe

Contains data from https://www.stata-press.com/data/r18/stan2.dta
 Observations:           103                    Heart transplant data
    Variables:             5                    30 Nov 2022 11:14

Variable        Storage   Display    Value
    name           type    format    label      Variable label

id               int      %8.0g                 Patient identifier
died             byte     %8.0g                 Survival status (1=dead)
stime            float    %8.0g                 Survival time (days)
transplant       byte     %8.0g                 Heart transplant
wait             int      %8.0g                 Waiting time

Sorted by:
```

The data are from 103 patients selected as transplantation candidates. There is one record on each patient, and the important variables, from an st-command perspective, are

```
id             the patient's ID number
transplant     whether the patient received a transplant
wait           when (after acceptance) the patient received the transplant
stime          when (after acceptance) the patient died or was censored
died           the patient's status at stime
```

To better understand, let's examine two records from this dataset:

```
. list id transplant wait stime died if id==44 | id==16
```

	id	transp~t	wait	stime	died
33.	44	0	0	40	1
71.	16	1	28	308	1

Patient 44 never did receive a new heart; he or she died 40 days after acceptance while still on the waiting list. Patient 16 did receive a new heart—28 days after acceptance—yet died 308 days after acceptance.

Our goal is to turn this into st data that contain the histories of each of these patients. That is, we want records that appear as

id	t1	died	posttran
16	28	0	0
16	308	1	1
44	40	1	0

or, even more explicitly, as

id	t0	t1	died	posttran
16	0	28	0	0
16	28	308	1	1
44	0	40	1	0

The new variable, posttran, would be 0 before transplantation and 1 afterward.

Patient 44 would have one record in this new dataset recording that he or she died at time 40 and that posttran was 0 over the entire interval.

Patient 16, however, would have two records: one documenting the duration $(0, 28]$, during which posttran was 0, and one documenting the duration $(28, 308]$, during which posttran was 1.

Our goal is to take the first dataset and convert it into the second, which we can then stset. We make the transformation by using Stata's other data management commands. One way we could do this is by typing

```
. expand 2 if transplant
(69 observations created)
. by id, sort: gen byte posttran = (_n==2)
. by id: gen t1 = stime if _n==_N
(69 missing values generated)
. by id: replace t1 = wait if _n==1 & transplant
(69 real changes made)
. by id: replace died=0 if _n==1 & transplant
(45 real changes made)
```

expand 2 if transplant duplicated the observations for patients who had transplant $\neq$ 0. Considering our two sample patients, we would now have the following data:

id	transp~t	wait	stime	died
44	0	0	40	1
16	1	28	308	1
16	1	28	308	1

We would have 1 observation for patient 44 and 2 identical observations for patient 16.

We then by id, sort: gen posttran = (_n==2), resulting in

```
id  transp~t    wait    stime    died  posttran
16         1      28      308       1         0
16         1      28      308       1         1
44         0       0       40       1         0
```

This type of trickiness is discussed in [U] **13.7 Explicit subscripting**. Statements such as _n==2 produce values 1 (meaning true) and 0 (meaning false), so new variable posttran will contain 1 or 0 depending on whether _n is or is not 2. _n is the observation counter and, combined with by ID variable:, becomes the observation-within-ID counter. Thus we set posttran to 1 on second records and to 0 on all first records.

Finally, we produce the exit-time variable. Final exit time is just stime, and that is handled by the command by id: gen t1 = stime if _n==_N. _n is the observation-within-ID counter and _N is the total number of observations within id, so we just set the last observation on each patient to stime. Now we have

```
id  transp~t    wait    stime    died  posttran        t1
16         1      28      308       1         0         .
16         1      28      308       1         1       308
44         0       0       40       1         0        40
```

All that is left to do is to fill in t1 with the value from wait on the interim records, meaning replace t1=wait if it is an interim record.

There are many ways we could identify the interim records. In the output above, we did it by

```
. by id: replace t1 = wait if _n==1 & transplant
```

meaning that an interim record is a first record of a person who did receive a transplant. More easily, but with more trickery, we could have just said

```
. replace t1=wait if t1>=.
```

because the only values of t1 left to be filled in are the missing ones. Another alternative would be

```
. by id: replace t1 = wait if _n==1 & _N==2
```

which would identify the first record of two-record pairs. There are many alternatives, but they would all produce the same thing:

```
id  transp~t    wait    stime    died  posttran        t1
16         1      28      308       1         0        28
16         1      28      308       1         1       308
44         0       0       40       1         0        40
```

There is one more thing we must do, which is to reset died to contain 0 on the interim records:

```
. by id: replace died=0 if _n==1 & transplant
```

The result is

```
id  transp~t    wait    stime    died  posttran        t1
16         1      28      308       0         0        28
16         1      28      308       1         1       308
44         0       0       40       1         0        40
```

We now have the desired result and are ready to `stset` our data:

```
. stset t1, failure(died) id(id)

Survival-time data settings

            ID variable: id
         Failure event: died!=0 & died<.
Observed time interval: (t1[_n-1], t1]
     Exit on or before: failure
──────────────────────────────────────────────────────────────────────────────
        172  total observations
          2  multiple records at same instant                  PROBABLE ERROR
             (t1[_n-1]==t1)
──────────────────────────────────────────────────────────────────────────────
        170  observations remaining, representing
        102  subjects
         74  failures in single-failure-per-subject data
     31,933  total analysis time at risk and under observation
                                          At risk from t =          0
                                 Earliest observed entry t =          0
                                   Last observed exit t =      1,799
```

Well, something went wrong. Two records were excluded. There are few enough data here that we could just list the dataset and look for the problem, but let's pretend otherwise. We want to find the records that, within patient, are marked as exiting at the same time:

```
. by id: gen problem = t1==t1[_n-1]
. sort id died
. list id if problem

         ┌────┐
         │ id │
         ├────┤
  61.    │ 38 │
         └────┘

. list id transplant wait stime died posttran t1 if id==38

       ┌────────────────────────────────────────────────────┐
       │ id   transp~t   wait   stime   died   posttran   t1 │
       ├────────────────────────────────────────────────────┤
  60.  │ 38         1      5       5      0          0    5 │
  61.  │ 38         1      5       5      1          1    5 │
       └────────────────────────────────────────────────────┘
```

There is no typographical error in these data—we checked that variables `transplant`, `wait`, and `stime` contain what the original source published. Those variables indicate that patient 38 waited 5 days for a heart transplant, received one on the fifth day, and then died on the fifth day, too.

That makes perfect sense, but not to Stata, which orders events within t as failures, followed by censorings, followed by entries. Reading `t1`, Stata went for this literal interpretation: patient 38 was censored at time 5 with `posttran = 0`; then, at time 5, patient 38 died; and then, at time 5, patient 38 reentered the data, but this time with `posttran = 1`. That made no sense to Stata.

Stata's sequencing of events may surprise you, but trust us, there are good reasons for it, and really, the ordering convention does not matter. To fix this problem, we just have to put a little time between the implied entry at time 5 and the subsequent death:

```
. replace t1 = 5.1 in 61
(1 real change made)
. list id transplant wait stime died posttran t1 if id==38
```

	id	transp~t	wait	stime	died	posttran	t1
60.	38	1	5	5	0	0	5
61.	38	1	5	5	1	1	5.1

Now the data make sense both to us and to Stata: until time 5, the patient had posttran $= 0$; then, at time 5, the value of posttran changed to 1; and then, at time 5.1, the patient died.

```
. stset t1, id(id) failure(died)

Survival-time data settings

         ID variable: id
      Failure event: died!=0 & died<.
Observed time interval: (t1[_n-1], t1]
   Exit on or before: failure
─────────────────────────────────────────────────────────────────────
        172  total observations
          0  exclusions
─────────────────────────────────────────────────────────────────────
        172  observations remaining, representing
        103  subjects
         75  failures in single-failure-per-subject data
   31,938.1  total analysis time at risk and under observation
                                       At risk from t =          0
                              Earliest observed entry t =          0
                                 Last observed exit t =      1,799
```

This dataset is now ready for use with all the other st commands. Here is an illustration:

```
. use https://www.stata-press.com/data/r18/stan3, clear
(Heart transplant data)
. stset, noshow
. stsum, by(posttran)
```

		Incidence	Number of	Survival time		
posttran	Time at risk	rate	subjects	25%	50%	75%
0	5,936	.0050539	103	36	149	340
1	26,002.1	.0017306	69	39	96	979
Total	31,938.1	.0023483	103	36	100	979

```
. stcox age posttran surgery year
Iteration 0:  Log likelihood = -298.31514
Iteration 1:  Log likelihood =  -289.7344
Iteration 2:  Log likelihood = -289.53498
Iteration 3:  Log likelihood = -289.53378
Iteration 4:  Log likelihood = -289.53378
Refining estimates:
Iteration 0:  Log likelihood = -289.53378

Cox regression with Breslow method for ties

No. of subjects =      103                        Number of obs =     172
No. of failures =       75
Time at risk    = 31,938.1
                                                  LR chi2(4)    =   17.56
Log likelihood = -289.53378                       Prob > chi2   = 0.0015
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf. interval]	
age	1.030224	.0143201	2.14	0.032	1.002536	1.058677
posttran	.9787243	.3032597	-0.07	0.945	.5332291	1.796416
surgery	.3738278	.163204	-2.25	0.024	.1588759	.8796
year	.8873107	.059808	-1.77	0.076	.7775022	1.012628

◁

Video example

Learn how to set up your data for survival analysis

References

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata Press.

Crowley, J., and M. Hu. 1977. Covariance analysis of heart transplant survival data. *Journal of the American Statistical Association* 72: 27–36. https://doi.org/10.2307/2286902.

Kalbfleisch, J. D., and R. L. Prentice. 2002. *The Statistical Analysis of Failure Time Data*. 2nd ed. New York: Wiley.

Also see

[ST] **snapspan** — Convert snapshot data to time-span data

[ST] **stdescribe** — Describe survival-time data

Title

stsplit — Split and join time-span records

Description

stsplit with the at(*numlist*) or every(#) option splits episodes into two or more episodes at the implied time points since being at risk or after a time point specified by after(). Each resulting record contains the follow-up on one subject through one time band. Expansion on multiple time scales may be obtained by repeatedly using stsplit. *newvar* specifies the name of the variable to be created containing the observation's category. The new variable records the interval to which each new observation belongs and is bottom coded.

stsplit, at(failures) performs episode splitting at the failure times (per stratum).

stjoin performs the reverse operation, namely, joining episodes back together when such can be done without losing information.

Quick start

Split episodes in [stset] data at analysis times 5, 10, and 15, and create new time category identifier timecat

```
stsplit timecat, at(5 10 15)
```

Join data that has been split after dropping variable created by stsplit

```
drop timecat
stjoin
```

Split episodes at the value in startvar

```
stsplit timecat, at(0) after(time=startvar)
```

Split data at 30, 40, and 50 time units after the value in startvar

```
stsplit timecat, at(30 40 50) after(time=startvar)
```

Split data every 10 time units after startvar

```
stsplit timecat, every(10) after(time=startvar)
```

Split data at failure times

```
stsplit, at(failures)
```

Same as above, and create risk-set identifier variable riskvar

```
stsplit, at(failures) riskset(riskvar)
```

Menu

stsplit

Statistics > Survival analysis > Setup and utilities > Split time-span records

stjoin

Statistics > Survival analysis > Setup and utilities > Join time-span records

Syntax

Split at designated times

> stsplit *newvar* [*if*], { at(*numlist*) | <u>eve</u>ry(*#*) } [*stsplitDT_options*]

Split at failure times

> stsplit [*if*], at(<u>f</u>ailures) [*stsplitFT_options*]

Join episodes

> stjoin [, <u>c</u>ensored(*numlist*)]

stsplitDT_options	Description
Main	
* at(*numlist*)	split records at specified analysis times
* <u>eve</u>ry(*#*)	split records when analysis time is a multiple of *#*
<u>af</u>ter(*spec*)	use time since *spec* for at() or every() rather than time since onset of risk; see *Options*
trim	exclude observations outside of range
<u>nop</u>reserve	do not save original data; programmer's option

*Either at(*numlist*) or every(*#*) is required with stsplit at designated times.

stsplitFT_options	Description
Main	
* at(<u>f</u>ailures)	split at observed failure times
strata(*varlist*)	restrict splitting to failures within stratum defined by *varlist*
<u>ris</u>kset(*newvar*)	create a risk-set ID variable named *newvar*
<u>nop</u>reserve	do not save original data; programmer's option

*at(failures) is required with stsplit at failure times.

You must stset your dataset by using the id() option before using stsplit or stjoin; see [ST] **stset**. nopreserve does not appear in the dialog box.

Options for stsplit

Main

at(*numlist*) or every(*#*) is required in syntax one; at(failures) is required for syntax two.
These options specify the analysis times at which the records are to be split.

at(5(5)20) splits records at $t = 5$, $t = 10$, $t = 15$, and $t = 20$.

If at($[...]$ max) is specified, max is replaced by a suitably large value. For instance, to split
records every five analysis-time units from time zero to the largest follow-up time in our data, we
could find out what the largest time value is by typing summarize _t and then explicitly typing
it into the at() option, or we could just specify at(0(5)max).

every(*#*) is a shorthand for at(*#*(*#*)max); that is, episodes are split at each positive multiple of
#.

after(*spec*) specifies the reference time for at() or every(). Syntax one can be thought of as
corresponding to after(*time of onset of risk*), although you cannot really type this. You could
type, however, after(time=birthdate) or after(time=marrydate) or after(marrydate).

spec has syntax

$$\big[\{\texttt{time} \,|\, \texttt{t} \,|\, \texttt{\_t}\} =\big]\, \{exp \,|\, \texttt{min}(exp) \,|\, \texttt{asis}(exp)\}$$

where

time specifies that the expression be evaluated in the same time units as *timevar* in stset *timevar*,
.... This is the default.

t and _t specify that the expression be evaluated in units of analysis time. t and _t are synonyms;
it makes no difference whether you specify one or the other.

exp specifies the reference time. For multiepisode data, *exp* should be constant within subject ID.

min(*exp*) specifies that for multiepisode data, the minimum of *exp* be taken within ID.

asis(*exp*) specifies that for multiepisode data, *exp* be allowed to vary within ID.

trim specifies that observations with values less than the minimum or greater than the maximum
value listed in at() be excluded from subsequent analysis. Such observations are not dropped
from the data; trim merely sets their value of variable _st to 0 so that they will not be used, yet
they can still be retrieved the next time the dataset is stset.

strata(*varlist*) specifies up to five strata variables. Observations with equal values of the variables
are assumed to be in the same stratum. strata() restricts episode splitting to failures that occur
within the stratum, and memory requirements are reduced when strata are specified.

riskset(*newvar*) specifies the name for a new variable recording the distinct risk set in which an
episode occurs, and missing otherwise.

The following option is available with stsplit but is not shown in the dialog box:

nopreserve is intended for use by programmers. It speeds the transformation by not saving the
original data, which can be restored should things go wrong or if you press *Break*. Programmers
often specify this option when they have already preserved the original data. nopreserve does
not affect the transformation.

Option for stjoin

censored(*numlist*) specifies values of the failure variable, *failvar*, from
stset ..., failure(*failvar*=...) that indicate "no event" (censoring).

If you are using stjoin to rejoin records after stsplit, you do not need to specify censored(). Just do not forget to drop the variable created by stsplit before typing stjoin. See example 4 below.

Neither do you need to specify censored() if, when you stset your dataset, you specified failure(*failvar*) and not failure(*failvar*=...). Then stjoin knows that *failvar* = 0 and *failvar* = . (missing) correspond to no event. Two records can be joined if they are contiguous and record the same data and the first record has *failvar* = 0 or *failvar* = ., meaning no event at that time.

You may need to specify censored(), and you probably do if, when you stset the dataset, you specified failure(*failvar*=...). If stjoin is to join records, it needs to know what events do not count and can be discarded. If the only such event is *failvar* = ., you do not need to specify censored().

Remarks and examples

Remarks are presented under the following headings:

> What stsplit does and why
> Using stsplit to split at designated times
> Time versus analysis time
> Splitting data on recorded ages
> Using stsplit to split at failure times

What stsplit does and why

stsplit splits records into two or more records on the basis of analysis time or on a variable that depends on analysis time, such as age. stsplit takes data like

id	_t0	_t	x1	x2	_d
1	0	18	12	11	1

and produces

id	_t0	_t	x1	x2	_d	tcat
1	0	5	12	11	0	0
1	5	10	12	11	0	5
1	10	18	12	11	1	10

or

id	_t0	_t	x1	x2	_d	agecat
1	0	7	12	11	0	30
1	7	17	12	11	0	40
1	17	18	12	11	1	50

The above alternatives record the same underlying data: subject 1 had x1 = 12 and x2 = 11 during $0 < t \leq 18$, and at $t = 18$, the subject failed.

The difference between the two alternatives is that the first breaks out the analysis times 0–5, 5–10, and 10–20 (although subject 1 failed before $t = 20$). The second breaks out age 30–40, 40–50, and 50–60. You cannot tell from what is presented above, but at $t = 0$, subject 1 was 33 years old.

In our example, that the subject started with one record is not important. The original data on the subject might have been

```
id    _t0    _t    x1    x2    _d
 1      0    14    12    11     0
 1     14    18    12     9     1
```

and then we would have obtained

```
id    _t0    _t    x1    x2    _d    tcat
 1      0     5    12    11     0       0
 1      5    10    12    11     0       5
 1     10    14    12    11     0      10
 1     14    18    12     9     1      10
```

or

```
id    _t0    _t    x1    x2    _d    agecat
 1      0     7    12    11     0       30
 1      7    14    12    11     0       40
 1     14    17    12     9     0       40
 1     17    18    12     9     1       50
```

Also we could just as easily have produced records with analysis time or age recorded in single-year categories. That is, we could start with

```
id    _t0    _t    x1    x2    _d
 1      0    14    12    11     0
 1     14    18    12     9     1
```

and produce

```
id    _t0    _t    x1    x2    _d    tcat
 1      0     1    12    11     0       0
 1      1     2    12    11     0       1
 1      2     3    12    11     0       2
 ...
```

or

```
id    _t0    _t    x1    x2    _d    agecat
 1      0     1    12    11     0       30
 1      1     2    12    11     0       31
 1      2     3    12    11     0       32
 ...
```

Moreover, we can even do this splitting on more than one variable. Let's go back and start with

```
id    _t0    _t    x1    x2    _d
 1      0    18    12    11     1
```

Let's split it into the analysis-time intervals 0–5, 5–10, and 10–20, *and* let's split it into 10-year age intervals 30–40, 40–50, and 50–60. The result would be

```
id    _t0    _t    x1    x2    _d    tcat    agecat
 1      0     5    12    11     0       0       30
 1      5     7    12    11     0       5       30
 1      7    10    12    11     0       5       40
 1     10    17    12    11     0      10       40
 1     17    18    12    11     1      10       50
```

Why would we want to do any of this?

We might want to split on a time-dependent variable, such as age, if we want to estimate a Cox proportional hazards model and include current age among the regressors (although we could instead use stcox's tvc() option) or if we want to make tables by age groups (see [ST] **strate**).

Using stsplit to split at designated times

stsplit's syntax to split at designated times is, ignoring other options,

> stsplit *newvar* [*if*], at(*numlist*)
>
> stsplit *newvar* [*if*], at(*numlist*) after(*spec*)

at() specifies the analysis times at which records are to be split. Typing at(5 10 15) splits records at the indicated analysis times and separates records into the four intervals 0–5, 5–10, 10–15, and 15+.

In the first syntax, the splitting is done on analysis time, t. In the second syntax, the splitting is done on 5, 10, and 15 analysis-time units after the time given by after(*spec*).

In either case, stsplit also creates *newvar* containing the interval to which each observation belongs. Here *newvar* would contain 0, 5, 10, and 15; it would contain 0 if the observation occurred in the interval 0–5, 5 if the observation occurred in the interval 5–10, and so on. To be precise,

Category	Precise meaning	*newvar* value
0–5	$(-\infty, 5]$	0
5–10	$(5, 10]$	5
10–15	$(10, 15]$	10
15+	$(15, \infty)$	15

If any of the at() numbers are negative (which would be allowed only by specifying the after() option and would be unusual), the first category is labeled one less than the minimum value specified by at().

Consider the data

```
id     yr0     yr1  yrborn    x1  event
 1    1990    1995    1960     5     52
 2    1993    1997    1964     3     47
```

In these data, subjects became at risk in yr0. The failure event of interest is event $= 47$, so we stset our dataset by typing

```
. stset yr1, id(id) origin(time yr0) failure(event==47)
(output omitted )
```

and that results in

```
id   _t0    _t     yr0     yr1  yrborn    x1  event    _d
 1     0     5    1990    1995    1960     5     52     0
 2     0     4    1993    1997    1964     3     47     1
```

In the jargon of st, variables _t0 and _t record the span of each record in analysis-time (t) units. Variables yr0 and yr1 also record the time span, but in time units. Variable _d records 1 for failure and 0 otherwise.

Typing `stsplit cat, at(2 4 6 8)` would split the records on the basis of analysis time:

```
. stsplit cat, at(2 4 6 8)
(3 observations (episodes) created)
. order id _t0 _t yr0 yr1 yrborn x1 event _d cat
. list id-cat
```

	id	_t0	_t	yr0	yr1	yrborn	x1	event	_d	cat
1.	1	0	2	1990	1992	1960	5	.	0	0
2.	1	2	4	1990	1994	1960	5	.	0	2
3.	1	4	5	1990	1995	1960	5	52	0	4
4.	2	0	2	1993	1995	1964	3	.	0	0
5.	2	2	4	1993	1997	1964	3	47	1	2

The first record, which represented the analysis-time span $(0, 5]$, was split into three records: $(0, 2]$, $(2, 4]$, and $(4, 5]$. The yrborn and x1 values from the single record were duplicated in $(0, 2]$, $(2, 4]$, and $(4, 5]$. The original event variable was changed to missing at $t = 2$ and $t = 4$ because we do not know the value of event; all we know is that event is 52 at $t = 5$. The _d variable was correspondingly set to 0 for $t = 2$ and $t = 4$ because we do know, at least, that the subject did not fail.

stsplit also keeps your original time variables up to date in case you want to streset or re-stset your dataset. yr1 was updated, too.

Now let's go back to our original dataset after we stset it but before we split it,

```
id   _t0   _t    yr0    yr1  yrborn   x1  event   _d
 1     0    5   1990   1995    1960    5     52    0
 2     0    4   1993   1997    1964    3     47    1
```

and consider splitting on age. In 1990, subject 1 is age $1990 - \text{yrborn} = 1990 - 1960 = 30$, and subject 2 is 29. If we type

```
. stsplit acat, at(30 32 34) after(time=yrborn)
```

we will split the data according to

```
        age <= 30   (called acat=0)
30 < age <= 32   (called acat=30)
32 < age <= 34   (called acat=32)
34 < age         (called acat=34)
```

The result will be

```
id   _t0   _t    yr0    yr1  yrborn   x1  event   _d  acat
 1     0    2   1990   1992    1960    5      .    0    30
 1     2    4   1990   1994    1960    5      .    0    32
 1     4    5   1990   1995    1960    5     52    0    34
 2     0    1   1993   1994    1964    3      .    0     0
 2     1    3   1993   1996    1964    3      .    0    30
 2     3    4   1993   1997    1964    3     47    1    32
```

The original record on subject 1 corresponding to $(0, 5]$ was split into $(0, 2]$, $(2, 4]$, and $(4, 5]$ because those are the t-values at which age becomes 32 and 34.

You can stsplit the data more than once. Now having these data, if we typed

```
. stsplit cat, at(2 4 6 8)
```

the result would be

id	_t0	_t	yr0	yr1	yrborn	x1	event	_d	acat	cat
1	0	2	1990	1992	1960	5	.	0	30	0
1	2	4	1990	1994	1960	5	.	0	32	2
1	4	5	1990	1995	1960	5	52	0	34	4
2	0	1	1993	1994	1964	3	.	0	0	0
2	1	2	1993	1995	1964	3	.	0	30	0
2	2	3	1993	1996	1964	3	.	0	30	2
2	3	4	1993	1997	1964	3	47	1	32	2

Whether we typed

```
. stsplit acat, at(30 32 34) after(time=yrborn)
. stsplit cat, at(2 4 6 8)
```

or

```
. stsplit cat, at(2 4 6 8)
. stsplit acat, at(30 32 34) after(time=yrborn)
```

would make no difference.

Time versus analysis time

Be careful using the after() option if, when you stset your dataset, you specified stset's scale() option. We say be careful, but actually we mean be appreciative, because stsplit will do just what you would expect if you did not think too hard.

When you split a record on a time-dependent variable, at() is still specified in analysis-time units, meaning units of time/scale().

For instance, if your original data recorded time as Stata dates, that is, number of days since 1960,

id	date0	date1	birthdate	x1	event
1	14apr1993	27mar1995	12jul1959	5	52

...

and you previously stset your dataset by typing

```
. stset date1, id(id) origin(time date0) scale(365.25) ...
```

and you now wanted to split on the age implied by birthdate, you would specify the split points in *years* since birth:

```
. stsplit agecat, at(20(5)60) after(time=birth)
```

at(20(5)60) means to split the records at the ages, measured in years, of 20, 25, ..., 60.

When you stset your dataset, you basically told st how you recorded times (you recorded them as dates) and how to map such times (dates) into analysis time. That was implied by what you typed, and all of st remembers that. Typing

```
. stsplit agecat, at(20(5)60) after(time=birth)
```

tells stsplit to split the data on 20, 25, ..., 60 analysis-time units after birthdate for each subject.

Splitting data on recorded ages

Recorded ages can sometimes be tricky. Consider the data

```
id    yr0     yr1    age    x1  event
 1    1980   1996    30      5    52
...
```

When was age $= 30$ recorded—1980 or 1996? Put aside that question because things are about to get worse. Say that you stset this dataset so that yr0 is the origin(),

```
id   _t0   _t     yr0     yr1   age   x1  event
 1     0   16    1980    1996    30    5    52
...
```

and then split on analysis time by typing stsplit cat, at(5(5)20). The result would be

```
id   _t0   _t     yr0     yr1   age   x1  event
 1     0    5    1980    1985    30    5     .
 1     5   10    1980    1990    30    5     .
 1    10   15    1980    1995    30    5     .
 1    15   16    1980    1996    30    5    52
```

Regardless of the answer to the question on when age was measured, age is most certainly not 30 in the newly created records, although you might argue that age at baseline was 30 and that is what you wanted, anyway.

The only truly safe way to deal with ages is to convert them back to birthdates at the outset. Here we would, early on, type

```
. generate bdate = yr1 - age            (if age was measured at yr1)
```

or

```
. generate bdate = yr0 - age            (if age was measured at yr0)
```

In fact, stsplit tries to protect you from making age errors. Suppose that you did not do as we just recommended. Say that age was measured at yr1, and you typed, knowing that stsplit wants a date,

```
. stsplit acat, at(20(5)50) after(time= yr1-age)
```

on these already stsplit data. stsplit will issue the error message "after() should be constant within id". To use the earliest date, you need to type

```
. stsplit acat, at(20(5)50) after(time= min(yr1-age))
```

Nevertheless, be aware that when you stsplit data, if you have recorded ages in your data, and if the records were not already split to control for the range of those ages, then age values, just like all the other variables, are carried forward and no longer reflect the age of the newly created record.

▷ Example 1: Splitting on age

Consider the data from a heart disease and diet survey. The data arose from a study described more fully in Morris, Marr, and Clayton (1977) and analyzed in Clayton and Hills (1993). (Their results differ slightly from ours because the dataset has been updated.)

```
. use https://www.stata-press.com/data/r18/diet
(Diet data with dates)

. describe

Contains data from https://www.stata-press.com/data/r18/diet.dta
 Observations:               337                  Diet data with dates
    Variables:                11                  1 May 2022 19:01
─────────────────────────────────────────────────────────────────────────────
Variable        Storage   Display    Value
    name           type    format    label      Variable label
─────────────────────────────────────────────────────────────────────────────
id               int      %9.0g                 Subject identity number
fail             byte     %8.0g                 Outcome (CHD = 1 3 13)
job              byte     %8.0g                 Occupation
month            byte     %8.0g                 Month of survey
energy           float    %9.0g                 Total energy (1000kcals/day)
height           float    %9.0g                 Height (cm)
weight           float    %9.0g                 Weight (kg)
hienergy         byte     %9.0g                 Indicator for high energy
doe              int      %td                   Date of entry
dox              int      %td                   Date of exit
dob              int      %td                   Date of birth
─────────────────────────────────────────────────────────────────────────────

Sorted by: id
```

In this dataset, the outcome variable, `fail`, has been coded as 0, 1, 3, 5, 12, 13, 14, and 15. Codes 1, 3, and 13 indicated coronary heart disease (CHD), other nonzero values code other events such as cancer, and 0 is used to mean "no event" at the end of the study.

The variable `hienergy` is coded 1 if the total energy consumption is more than 2.75 Mcal and 0 otherwise.

We would like to expand the data, using age as the time scale with 10-year age bands. We do this by first `stset`ting the dataset, specifying the date of birth as the origin.

```
. stset dox, failure(fail) origin(time dob) enter(time doe) scale(365.25) id(id)

Survival-time data settings

           ID variable: id
        Failure event: fail!=0 & fail<.
Observed time interval: (dox[_n-1], dox]
    Enter on or after: time doe
    Exit on or before: failure
    Time for analysis: (time-origin)/365.25
               Origin: time dob
─────────────────────────────────────────────────────────────────────────────
        337  total observations
          0  exclusions
─────────────────────────────────────────────────────────────────────────────
        337  observations remaining, representing
        337  subjects
         80  failures in single-failure-per-subject data
  4,603.669  total analysis time at risk and under observation
                                            At risk from t =          0
                                 Earliest observed entry t =   30.07529
                                   Last observed exit t =   69.99863
```

The origin is set to date of birth, making time-since-birth analysis time, and the scale is set to 365.25, so that time since birth is measured in years.

Let's list a few records and verify that the analysis-time variables $\_t0$ and $\_t$ are indeed recorded as we expect:

```
. list id dob doe dox fail _t0 _t if id==1 | id==34
```

	id	dob	doe	dox	fail	_t0	_t
1.	1	04jan1915	16aug1964	01dec1976	0	49.615332	61.908282
34.	34	12jun1899	16apr1959	31dec1966	3	59.841205	67.550992

We see that patient 1 was 49.6 years old at time of entry into our study and left at age 61.9. Patient 34 entered the study at age 59.8 and exited the study with CHD at age 67.6.

Now we can split the data by age:

```
. stsplit ageband, at(40(10)70)
(418 observations (episodes) created)
```

stsplit added 418 observations to the dataset in memory and generated a new variable, ageband, which identifies each observation's age group.

```
. list id _t0 _t ageband fail height if id==1 | id==34
```

	id	_t0	_t	ageband	fail	height
1.	1	49.615332	50	40	.	175.387
2.	1	50	60	50	.	175.387
3.	1	60	61.908282	60	0	175.387
61.	34	59.841205	60	50	.	177.8
62.	34	60	67.550992	60	3	177.8

The single record for the subject with $id = 1$ has expanded to three records. The first refers to the age band 40–49, coded 40, and the subject spends $\_t - \_t0 = 0.384668$ years in this band. The second refers to the age band 50–59, coded 50, and the subject spends 10 years in this band, and so on. The follow-up in each of the three bands is censored ($fail = .$). The single record for the subject with $id = 34$ is expanded to two age bands; the follow-up for the first band was censored ($fail = .$), and the follow-up for the second band ended in CHD ($fail = 3$).

The values for variables that do not change with time, such as height, are simply repeated in the new records. This can lead to much larger datasets after expansion. Dropping unneeded variables before using stsplit may be necessary.

◁

▷ Example 2: Splitting on age and time in study

To use stsplit to expand the records on two time scales simultaneously, such as age and time in study, we can first expand on the age scale as described in example 1, and then on the time-in-study scale with the command

```
. stsplit timeband, at(0(5)25) after(time=doe)
(767 observations (episodes) created)

. list id _t0 _t ageband fail if id==1 | id==34
```

	id	_t0	_t	ageband	fail
1.	1	49.615332	50	40	.
2.	1	50	54.615332	50	.
3.	1	54.615332	59.615332	50	.
4.	1	59.615332	60	50	.
5.	1	60	61.908282	60	0
111.	34	59.841205	60	50	.
112.	34	60	64.841205	60	.
113.	34	64.841205	67.550992	60	3

By splitting the data by using two time scales, we partitioned the data into time cells corresponding to a *Lexis diagram* as described, for example, in Clayton and Hills (1993). Also see Keiding (1998) for an overview of Lexis diagrams. Each new observation created by splitting the data records the time that the individual spent in a Lexis cell. We can obtain the time spent in the cell by calculating the difference $\_t - \_t0$. For example, the subject with $id = 1$ spent 0.384668 years $(50 - 49.615332)$ in the cell corresponding to age 40–49 and study time 0–5, and 4.615332 years $(54.615332 - 50)$ in the cell for age 50–59 and study time 0–5.

We can also do these expansions in reverse order, that is, split first on study time and then on age.

◁

▷ Example 3: Explanatory variables that change with time

In the previous examples, time, in the form of age or time in study, is the explanatory variable to be studied or controlled for, but in some studies other explanatory variables also vary with time. The stsplit command can sometimes be used to expand the records so that in each new record such an explanatory variable is constant over time. For example, in the Stanford heart data (see [ST] **stset**), we would like to split the data and generate the explanatory variable posttran, which takes the value 0 before transplantation and 1 thereafter. The follow-up must therefore be divided into time before transplantation and time after.

We first generate for each observation an entry time and an exit time that preserve the correct follow-up time in such a way that the time of transplants is the same for all individuals. By summarizing wait, the time to transplant, we obtain its maximum value of 310. By selecting a value greater than this maximum, say, 320, we now generate two new variables:

```
. use https://www.stata-press.com/data/r18/stanford, clear
(Heart transplant data)

. generate enter = 320 - wait

. generate exit = 320 + stime
```

We have created a new artificial time scale where all transplants are coded as being performed at time 320. By defining enter and exit in this manner, we maintain the correct total follow-up time for each patient. We now stset and stsplit the data:

```
. stset exit, enter(time enter) failure(died) id(id)
Survival-time data settings

            ID variable: id
         Failure event: died!=0 & died<.
Observed time interval: (exit[_n-1], exit]
     Enter on or after: time enter
     Exit on or before: failure
_____
         103  total observations
           0  exclusions
_____
         103  observations remaining, representing
         103  subjects
          75  failures in single-failure-per-subject data
    34,589.1  total analysis time at risk and under observation
                                         At risk from t =             0
                                Earliest observed entry t =            10
                                   Last observed exit t =         2,119
. stsplit posttran, at(0,320)
(69 observations (episodes) created)
. replace posttran=0 if transplant==0
(34 real changes made)
. replace posttran=1 if posttran==320
(69 real changes made)
```

We replaced posttran in the last command so that it is now a 0/1 indicator variable. We can now generate our follow-up time, t1, as the difference between our analysis-time variables, list the data, and stset the dataset.

```
. generate t1 =_t - _t0
. list id enter exit _t0 _t posttran if id==16 | id==44
```

	id	enter	exit	_t0	_t	posttran
41.	44	320	360	320	360	0
110.	16	292	320	292	320	0
111.	16	292	628	320	628	1

```
. stset t1, failure(died) id(id)
Survival-time data settings

            ID variable: id
         Failure event: died!=0 & died<.
Observed time interval: (t1[_n-1], t1]
     Exit on or before: failure
_____
         172  total observations
           0  exclusions
_____
         172  observations remaining, representing
         103  subjects
          75  failures in single-failure-per-subject data
    31,938.1  total analysis time at risk and under observation
                                         At risk from t =             0
                                Earliest observed entry t =             0
                                   Last observed exit t =         1,799
```

Using stsplit to split at failure times

To split data at failure times, you would use `stsplit` with the following syntax, ignoring other options:

$$\texttt{stsplit} \; [\, \textit{if} \,] \; \texttt{, at(failures)}$$

This form of episode splitting is useful for Cox regression with time-varying covariates. Splitting at the failure times is useful because of a property of the maximum partial-likelihood estimator for a Cox regression model: the likelihood is evaluated only at the times at which failures occur in the data, and the computation depends only on the risk pools at those failure times. Changes in covariates between failure times do not affect estimates for a Cox regression model. Thus, to fit a model with time-varying covariates, all you have to do is define the values of these time-varying covariates at all failure times at which a subject was at risk (Collett 2015, chap. 8). After splitting at failure times, you define time-varying covariates by referring to the system variable `_t` (analysis time) or the *timevar* variable used to `stset` the data.

After splitting at failure times, all st commands still work fine and produce the same results as before splitting. To fit parametric models with time-varying covariates, it does not suffice to specify covariates at failure times. Stata can fit "piecewise constant" models by manipulating data using `stsplit, {at()|every()}`.

◁

▷ Example 4: Splitting on failure times to test the proportional-hazards assumption

Collett (2015, 187–190) presents data on 26 ovarian cancer patients who underwent two different chemotherapy protocols after a surgical intervention. Here are a few of the observations:

```
. use https://www.stata-press.com/data/r18/ocancer, clear
. list patient time cens treat age rdisea in 1/6, separator(0)
```

	patient	time	cens	treat	age	rdisea
1.	1	156	1	1	66	2
2.	2	1040	0	1	38	2
3.	3	59	1	1	72	2
4.	4	421	0	2	53	2
5.	5	329	1	1	43	2
6.	6	769	0	2	59	2

The `treat` variable indicates the chemotherapy protocol administered, `age` records the age of the patient at the beginning of the treatment, and `rdisea` records each patient's residual disease after surgery. After `stset`ting this dataset, we fit a Cox proportional-hazards regression model on `age` and `treat` to ascertain the effect of treatment, controlling for age.

```
. stset time, failure(cens) id(patient)

Survival-time data settings

           ID variable: patient
        Failure event: cens!=0 & cens<.
Observed time interval: (time[_n-1], time]
    Exit on or before: failure
─────────────────────────────────────────────────────────────────────────
          26  total observations
           0  exclusions
─────────────────────────────────────────────────────────────────────────
          26  observations remaining, representing
          26  subjects
          12  failures in single-failure-per-subject data
      15,588  total analysis time at risk and under observation
                                            At risk from t =          0
                                 Earliest observed entry t =          0
                                    Last observed exit t =      1,227
. stcox age treat, nolog nohr

        Failure _d: cens
  Analysis time _t: time
       ID variable: patient

Cox regression with no ties

No. of subjects =      26                        Number of obs =        26
No. of failures =      12
Time at risk    = 15,588
                                                 LR chi2(2)     =   15.82
Log likelihood = -27.073767                      Prob > chi2    =  0.0004
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]
age	.1465698	.0458537	3.20	0.001	.0566982	.2364415
treat	-.7959324	.6329411	-1.26	0.209	-2.036474	.4446094

One way to test the proportional-hazards assumption is to include in the model a term for the interaction between age and time at risk, which is a continuously varying covariate. This can be easily done by first splitting the data at the failure times and then generating the interaction term.

```
. stsplit, at(failures)
(12 failure times)
(218 observations (episodes) created)

. generate tage = age * _t

. stcox age treat tage, nolog nohr

        Failure _d: cens
  Analysis time _t: time
       ID variable: patient

Cox regression with no ties

No. of subjects =      26                        Number of obs =       244
No. of failures =      12
Time at risk    = 15,588
                                                 LR chi2(3)     =   16.36
Log likelihood = -26.806607                      Prob > chi2    =  0.0010
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]
age	.2156499	.1126093	1.92	0.055	-.0050602	.43636
treat	-.6635945	.6695492	-0.99	0.322	-1.975887	.6486978
tage	-.0002031	.0002832	-0.72	0.473	-.0007582	.000352

Other time-varying interactions of age and time at risk could be generated. For instance,

```
. generate lntage = age * ln(_t)
. generate dage = age * (_t >= 500)
```

Although in most analyses in which we include interactions we also include main effects, if we include in a Cox regression a multiplicative interaction between analysis time (or any transformation) and some covariate, we should not include the analysis time as a covariate in stcox. The analysis time is constant within each risk set, and hence, its effect is not identified.

❑ Technical note

If our interest really were just in performing this test of the proportional-hazards assumption, we would not have had to use stsplit at all. We could have just typed

```
. stcox age treat, tvc(age)
```

to have fit a model including $t*age$, and if we wanted instead to include $\ln(t)*age$ or $age*t \geq 500$, we could have typed

```
. stcox age treat, tvc(age) texp(ln(_t))
. cstoc age treat, tvc(age) texp(_t >= 500)
```

Still, it is worth understanding how stsplit could be used to obtain the same results for instances when stcox's tvc() and texp() options are not rich enough to handle the desired specification.

◻

Assume that we want to control for rdisea as a stratification variable. If the data are already split at all failure times, we can proceed with

```
. stcox age treat tage, strata(rdisea)
```

If the data are not yet split, and memory is scarce, then we could just split the data at the failure times within the respective stratum. That is, with the original data in memory, we could type

```
. stset time, failure(cens) id(patient)
. stsplit, at(failures) strata(rdisea)
. generate tage = age * _t
. stcox treat age tage, strata(rdisea)
```

This would save memory by reducing the size of the split dataset.

◁

❑ Technical note

Of course, the above model could also be obtained by typing

```
. stcox age treat, tvc(age) strata(rdisea)
```

without splitting the data.

◻

▷ Example 5: Cox regression versus conditional logistic regression

Cox regression with the "exact partial" method of handling ties is tightly related to conditional logistic regression. In fact, we can perform Cox regression via clogit, as illustrated in the following example using Stata's cancer data. First, let's fit the Cox model.

```
. use https://www.stata-press.com/data/r18/cancer, clear
(Patient survival in drug trial)

. generate id =_n

. stset studytim, failure(died) id(id)

Survival-time data settings

           ID variable: id
         Failure event: died!=0 & died<.
Observed time interval: (studytime[_n-1], studytime]
     Exit on or before: failure

_____

          48  total observations
           0  exclusions

_____

          48  observations remaining, representing
          48  subjects
          31  failures in single-failure-per-subject data
         744  total analysis time at risk and under observation
                                            At risk from t =          0
                                  Earliest observed entry t =          0
                                      Last observed exit t =         39

. stcox age drug, nolog nohr exactp

         Failure _d: died
  Analysis time _t: studytime
        ID variable: id

Cox regression with exact partial likelihood method for ties

No. of subjects =  48                                Number of obs =       48
No. of failures =  31
Time at risk    = 744
                                                     LR chi2(2)     =   38.13
Log likelihood = -73.10556                           Prob > chi2    =  0.0000

_____

          _t │ Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
─────────────┼─────────────────────────────────────────────────────────────
         age │   .1169906   .0374955     3.12   0.002     .0435008    .1904805
        drug │  -1.664873   .3437487    -4.84   0.000    -2.338608   -.9911376
_____
```

We will now perform the same analysis by using clogit. To do this, we first split the data at failure times, specifying the riskset() option so that a risk set identifier is added to each observation. We then fit the conditional logistic regression, using _d as the outcome variable and the risk set identifier as the grouping variable.

```
. stsplit, at(failures) riskset(RS)
(21 failure times)
(534 observations (episodes) created)

. clogit _d age drug, group(RS) nolog
note: multiple positive outcomes within groups encountered.
```

Conditional (fixed-effects) logistic regression				Number of obs =	573
				LR chi2(2) =	38.13
				Prob > chi2 =	0.0000
Log likelihood = −73.10556				Pseudo R2 =	0.2069

_d	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
age	.1169906	.0374955	3.12	0.002	.0435008	.1904805
drug	−1.664873	.3437487	−4.84	0.000	−2.338608	−.9911376

◁

▷ Example 6: Joining data that have been split with stsplit

Let's return to the first example. We split the diet data into age bands, using the following commands:

```
. use https://www.stata-press.com/data/r18/diet, clear
(Diet data with dates)
. stset dox, failure(fail) origin(time dob) enter(time doe) scale(365.25) id(id)
 (output omitted )
. stsplit ageband, at(40(10)70)
(418 observations (episodes) created)
```

We can rejoin the data by typing stjoin:

```
. stjoin
(option censored(0) assumed)
(no observations eliminated)
```

Nothing happened! stjoin will combine records that are contiguous and record the same data. Here, when we split the data, stsplit created the new variable ageband, and that variable takes on different values across the split observations. Remember to drop the variable that stsplit creates:

```
. drop ageband
. stjoin
(option censored(0) assumed)
(418 observations eliminated)
```

◁

Wilhelm Lexis (1837–1914) was born near Aachen in Germany. He studied law, mathematics, and science at the University of Bonn and developed interests in the social sciences during a period in Paris. Lexis held posts at universities in Strasburg (now Strasbourg, in France), Dorpat (now Tartu, in Estonia), Freiburg, Breslau (now Wroclaw, in Poland), and Göttingen. During this peripatetic career, he carried out original work in statistics on the analysis of dispersion, foreshadowing the later development of χ^2 and analysis of variance.

Acknowledgments

References

Clayton, D. G., and M. Hills. 1993. *Statistical Models in Epidemiology*. Oxford: Oxford University Press.

——. 1995. ssa7: Analysis of follow-up studies. *Stata Technical Bulletin* 27: 19–26. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 219–227. College Station, TX: Stata Press.

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata.* Rev. 3rd ed. College Station, TX: Stata Press.

Collett, D. 2015. *Modelling Survival Data in Medical Research*. 3rd ed. Boca Raton, FL: Chapman and Hall/CRC.

Hertz, S. 2001. Wilhelm Lexis. In *Statisticians of the Centuries*, ed. C. C. Heyde and E. Seneta, 204–207. New York: Springer.

Keiding, N. 1998. Lexis diagrams. In *Encyclopedia of Biostatistics*, ed. P. Armitage and T. Colton, 2844–2850. New York: Wiley.

Lexis, W. H. 1875. *Einleitung in die Theorie der Bevölkerungsstatistik*. Strassburg: Trübner.

Morris, J. N., J. W. Marr, and D. G. Clayton. 1977. Diet and heart: A postscript. *BMJ* 19: 1307–1314. https://doi.org/10.1136/bmj.2.6098.1307.

Weesie, J. 1998a. ssa11: Survival analysis with time-varying covariates. *Stata Technical Bulletin* 41: 25–43. Reprinted in *Stata Technical Bulletin Reprints*, vol. 7, pp. 268–292. College Station, TX: Stata Press.

——. 1998b. dm62: Joining episodes in multi-record survival time data. *Stata Technical Bulletin* 45: 5–6. Reprinted in *Stata Technical Bulletin Reprints*, vol. 8, pp. 27–28. College Station, TX: Stata Press.

Also see

[ST] **stset** — Declare data to be survival-time data

Title

stsum — Summarize survival-time data

Description

stsum presents summary statistics: time at risk; incidence rate; number of subjects; and the 25th, 50th, and 75th percentiles of survival time.

stsum can be used with single- or multiple-record or single- or multiple-failure st data.

Quick start

Time at risk, incidence rate, number of subjects, and quartiles of survival time for stset data

    stsum

Same as above, but only report statistics for observations with $v1 = 1$

    stsum if v1==1

Report separate summary statistics for each level of v1

    stsum, by(v1)

Menu

Statistics > Survival analysis > Summary statistics, tests, and tables > Summarize survival-time data

Syntax

> stsum [*if*] [*in*] [, by(*varlist*) <u>nosho</u>w]

You must stset your data before using stsum; see [ST] **stset**.

by and collect are allowed; see [U] **11.1.10 Prefix commands**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

Options

> [Main]

by(*varlist*) requests separate summaries for each group along with an overall total. Observations are in the same group if they have equal values of the variables in *varlist*. *varlist* may contain any number of string or numeric variables.

noshow prevents stsum from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

Remarks and examples

Remarks are presented under the following headings:

> *Single-failure data*
> *Multiple-failure data*
> *Video example*

Single-failure data

Here is an example of stsum with single-record survival data:

```
. use https://www.stata-press.com/data/r18/page2
. stset, noshow
. stsum
```

	Time at risk	Incidence rate	Number of subjects	Survival time 25%	50%	75%
Total	9,118	.0039482	40	198	232	261

```
. stsum, by(group)
```

group	Time at risk	Incidence rate	Number of subjects	Survival time 25%	50%	75%
1	4,095	.0041514	19	190	216	234
2	5,023	.0037826	21	232	233	280
Total	9,118	.0039482	40	198	232	261

stsum works equally well with multiple-record survival data. Here is a summary of the multiple-record Stanford heart transplant data introduced in [ST] **stset**:

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)

. stsum

        Failure _d: died
  Analysis time _t: t1
      ID variable: id
```

	Time at risk	Incidence rate	Number of subjects	25%	Survival time 50%	75%
Total	31,938.1	.0023483	103	36	100	979

stsum with the by() option may produce results with multiple-record data that, at first, you may think are in error.

```
. stsum, by(posttran) noshow
```

posttran	Time at risk	Incidence rate	Number of subjects	25%	Survival time 50%	75%
0	5,936	.0050539	103	36	149	340
1	26,002.1	.0017306	69	39	96	979
Total	31,938.1	.0023483	103	36	100	979

For the time at risk, $5{,}936 + 26{,}002.1 = 31{,}938.1$, but, for the number of subjects, $103 + 69 \neq 103$. The posttran variable is not constant for the subjects in this dataset:

```
. stset, noshow

. stvary posttran
```

	Subjects for whom the variable is				
			never	always	sometimes
Variable	constant	varying	missing	missing	missing
posttran	34	69	103	0	0

In this dataset, subjects have one or two records. All subjects were eligible for heart transplantation. They have one record if they die or are lost because of censoring before transplantation, and they have two records if the operation was performed. Then the first record records their survival up to transplantation and the second records their subsequent survival. posttran is 0 in the first record and 1 in the second.

Thus all 103 subjects have records with posttran = 0, and when stsum reported results for this group, it summarized the pretransplantation survival. The incidence of death was 0.005, and median survival time was 149 days.

The posttran = 1 line of stsum's output summarizes the posttransplantation survival: 69 patients underwent transplantation, incidence of death was 0.002, and median survival time was 96 days. For these data, this is not 96 more days, but 96 days in total. That is, the clock was not reset at transplantation. Thus, without attributing cause, we can describe the differences between the groups as an increased hazard of death at early times followed by a decreased hazard later.

Multiple-failure data

If you simply type stsum with multiple-failure data, the reported survival time is the survival time to the first failure, assuming that the hazard function is not indexed by number of failures.

Here we have some multiple-failure data:

```
. use https://www.stata-press.com/data/r18/mfail2
. st
-> stset t, id(id) failure(d) time0(t0) exit(time .) noshow
Survival-time data settings
           ID variable: id
        Failure event: d!=0 & d<.
Observed time interval: (t0, t]
    Exit on or before: time .
. stsum
```

	Time at risk	Incidence rate	Number of subjects	Survival time 25%	50%	75%
Total	435,444	.0018556	926	201	420	703

To understand this output, let's also obtain output for each failure separately:

```
. stgen nf = nfailures()
. stsum, by(nf)
```

nf	Time at risk	Incidence rate	Number of subjects	Survival time 25%	50%	75%
0	263,746	.0020057	926	196	399	604
1	121,890	.0018131	529	252	503	816
2	38,807	.0014946	221	415	687	.
3	11,001	0	58	.	.	.
Total	435,444	.0018556	926	201	420	703

The stgen command added, for each subject, a variable containing the number of previous failures. For a subject, up to and including the first failure, nf is 0. Then nf is 1 up to and including the second failure, and then it is 2, and so on; see [ST] stgen.

The first line of the output, corresponding to nf = 0, states that among those who had experienced no failures yet, the incidence rate for (first) failure is 0.0020. The distribution of the time to the first failure is as shown.

Similarly, the second line, corresponding to nf = 1, is for those who have already experienced one failure. The incidence rate for (second) failures is 0.0018, and the distribution of time of (second) failures is as shown.

When we simply typed stsum, we obtained the same information shown as the total line of the more detailed output. The total incidence rate is easy to interpret, but what is the "total" survival-time distribution? It is an estimate of the distribution of the time to the first failure assuming that the hazard function $h(t)$ is the same across failures—that the second failure is no different from the first failure. This is an odd definition of "same" because the clock, t, is not reset in $h(t)$. What is the hazard of a failure—any failure—at time t? The answer is $h(t)$.

Another definition of "same" would have it that the hazard of a failure is given by $h(\tau)$, where τ is the time since last failure—that the process repeats. These definitions are different unless $h()$ is a constant function of t (τ).

So let's examine these multiple-failure data under the process-replication idea. The key variables in these st data are `id`, `t0`, `t`, and `d`:

```
. st
-> stset t, id(id) failure(d) time0(t0) exit(time .) noshow

Survival-time data settings

           ID variable: id
         Failure event: d!=0 & d<.
Observed time interval: (t0, t]
     Exit on or before: time .
```

Our goal is, for each subject, to reset `t0` and `t` to 0 after every failure event. We are going to have to trick Stata, or at least trick `stset`, because it will not let us set data where the same subject has multiple records summarizing the overlapping periods. So, the trick is to create a new `id` variable that is different for every ID−nf combination (remember, `nf` is the variable we previously created that records the number of prior failures). Then all the "new" subjects can have their clocks start at time 0:

```
. egen newid = group(id nf)

. sort newid t

. by newid: replace t = t - t0[1]
(808 real changes made)

. by newid: generate newt0 = t0 - t0[1]

. stset t, failure(d) id(newid) time0(newt0)

Survival-time data settings

           ID variable: newid
         Failure event: d!=0 & d<.
Observed time interval: (newt0, t]
     Exit on or before: failure

────────────────────────────────────────────────────────────────
      1,734  total observations
          0  exclusions
────────────────────────────────────────────────────────────────
      1,734  observations remaining, representing
      1,734  subjects
        808  failures in single-failure-per-subject data
    435,444  total analysis time at risk and under observation
                                       At risk from t =          0
                             Earliest observed entry t =          0
                                  Last observed exit t =        797
```

`stset` no longer thinks that we have multiple-failure data. Whereas with `id`, subjects had multiple failures, `newid` gives a unique identity to each ID−nf combination. Each "new" subject has at most one failure.

```
. stsum, by(nf)

        Failure _d: d
  Analysis time _t: t
       ID variable: newid
```

nf	Time at risk	Incidence rate	Number of subjects	Survival time 25%	50%	75%
0	263,746	.0020057	926	196	399	604
1	121,890	.0018131	529	194	384	580
2	38,807	.0014946	221	210	444	562
3	11,001	0	58	.	.	.
Total	435,444	.0018556	1734	201	404	602

Compare this table with the one we previously obtained. The incidence rates are the same, but the survival times differ because now we measure the times from one failure to the next. Previously, we measured the time from a fixed point. The time between events in these data appears to be independent of event number.

❑ Technical note

The method shown for converting multiple-failure data to replicated-process single-event failure data is completely general. The generic outline of the conversion process is

```
. stgen nf = nfailures()
. egen newid = group(id nf)
. sort newid t
. by newid: replace t = t - t0[1]
. by newid: generate newt0 = t0 - t0[1]
. stset t, failure(d) id(newid) t0(newt0)
```

where *id*, *t*, *t0*, and *d* are the names of your key survival-time variables.

Once you have done this to your data, you need exercise only one caution. If, in fitting models with stcox, streg, etc., you wish to obtain robust estimates of variance, you should include the vce(cluster *id*) option.

When you specify the vce(robust) option, stcox, streg, etc., assume that you mean vce(cluster *stset_id_variable*), which, here, will be vce(cluster newid). The data, however, are really more clustered than that. Two "subjects" with different newid values may, in fact, be the same real subject. vce(cluster *id*) is what is appropriate.

❑

Video example

[How to describe and summarize survival data](#)

Stored results

stsum stores the following in r():

Scalars

r(p25)	25th percentile	r(risk)	time at risk
r(p50)	50th percentile	r(ir)	incidence rate
r(p75)	75th percentile	r(N_sub)	number of subjects

Methods and formulas

The 25th, 50th, and 75th percentiles of survival times are obtained from $S(t)$, the Kaplan–Meier product-limit estimate of the survivor function. The 25th percentile, for instance, is obtained as the minimum value of t such that $S(t) \leq 0.75$.

Also see

Title

> **sttocc** — Convert survival-time data to case–control data

Description

sttocc generates a nested case–control study dataset from a cohort-study dataset by sampling controls from the risk sets. For each case, the controls are chosen randomly from those members of the cohort who are at risk at the failure time of the case. That is, the resulting case–control sample is matched with respect to analysis time—the time scale used to compute risk sets. The following variables are added to the dataset:

_case	Coded 0 for controls, 1 for cases
_set	Case–control ID; matches cases and controls that belong together
_time	Analysis time of the case's failure

The names of these three variables can be changed by specifying the generate() option. *varlist* defines variables that, in addition to those used in the creation of the case–control study, will be retained in the final dataset. If *varlist* is not specified, all variables are carried over into the resulting dataset.

When the resulting dataset is analyzed as a matched case–control study, odds ratios will estimate corresponding rate-ratio parameters in the proportional hazards model for the cohort study.

Randomness in the matching is obtained using Stata's runiform() function. To ensure that the sample truly is random, you should set the random-number seed; see [R] **set seed**.

Quick start

Create a nested case–control dataset from a cohort dataset that has been stset, matching cases to controls based on analysis time

    sttocc

Same as above, but match on analysis time and categorical variable catvar

    sttocc, match(catvar)

Same as above, but match 3 controls for each case

    sttocc, match(catvar) number(3)

Same as above, and name the case indicator case, the matching identifier mid, and the case's failure time ftime

    sttocc, match(catvar) number(3) generate(case mid ftime)

Menu

Statistics > Survival analysis > Setup and utilities > Convert survival-time data to case–control data

Syntax

> sttocc [*varlist*] [, *options*]

options	Description
Main	
<u>match</u>(*matchvarlist*)	match cases and controls on analysis time and specified categorical variables; default is to match on analysis time only
<u>number</u>(*#*)	use *#* controls for each case; default is number(1)
<u>nodots</u>	suppress displaying dots during calculation
Options	
<u>generate</u>(*case set time*)	new variable names; default is _case, _set, and _time

You must stset your data before using sttocc; see [ST] **stset**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

Options

[**Main**]

match(*matchvarlist*) specifies more categorical variables for matching controls to cases. When match() is not specified, cases and controls are matched with respect to time only. If match(*matchvarlist*) is specified, the cases will also be matched by *matchvarlist*.

number(*#*) specifies the number of controls to draw for each case. The default is 1, even though this is not a sensible choice.

nodots requests that dots not be placed on the screen at the beginning of each case–control group selection. By default, dots are displayed to show progress.

[**Options**]

generate(*case set time*) specifies variable names for the three new variables; the default is _case, _set, and _time.

Remarks and examples

Nested case–control studies are an attractive alternative to full Cox regression analysis, particularly when time-varying explanatory variables are involved. They are also attractive when some explanatory variables involve laborious coding. For example, you can create a file with a subset of variables for all subjects in the cohort, generate a nested case–control study, and go on to code the remaining data only for those subjects selected.

In the same way as with Cox regression, the results of the analysis are critically dependent on the choice of analysis time (time scale). The choice of analysis time may be calendar time—so that controls would be chosen from subjects still being monitored on the date that the case fails—but other time scales, such as age or time in study, may be more appropriate in some studies. Remember that the analysis time set in selecting controls is implicitly included in the model in subsequent analysis.

match() requires that controls also be matched to the case with respect to other categorical variables, such as sex. This produces an analysis closely mirroring stratified Cox regression. If we wanted to match on calendar time and 5-year age bands, we could first type stsplit ageband ...

to create the age bands and then specify match(ageband) on the sttocc command. Analyzing the resulting data as a matched case–control study would estimate rate ratios in the underlying cohort that are controlled for calendar time (very finely) and age (less finely). Such analysis could be carried out by Mantel–Haenszel (odds ratio) calculations, for example, using mhodds, or by conditional logistic regression using clogit.

When ties occur between entry times, censoring times, and failure times, the following convention is adopted:

$$\text{Entry time} < \text{Failure time} < \text{Censoring time}$$

Thus censored subjects and subjects entering at the failure time of the case are included in the risk set and are available for selection as controls. Tied failure times are broken at random.

▷ Example 1: Creating a nested case–control study

Using the diet data introduced in example 1 of [ST] stsplit, we will illustrate the use of sttocc, letting age be analysis time. Controls are chosen from subjects still being monitored at the age at which the case fails.

```
. use https://www.stata-press.com/data/r18/diet
(Diet data with dates)
. stset dox, failure(fail) enter(time doe) id(id) origin(time dob) scale(365.25)
Survival-time data settings
            ID variable: id
         Failure event: fail!=0 & fail<.
Observed time interval: (dox[_n-1], dox]
     Enter on or after: time doe
     Exit on or before: failure
     Time for analysis: (time-origin)/365.25
                Origin: time dob
_____
        337  total observations
          0  exclusions
_____
        337  observations remaining, representing
        337  subjects
         80  failures in single-failure-per-subject data
  4,603.669  total analysis time at risk and under observation
                                            At risk from t =         0
                                 Earliest observed entry t =  30.07529
                                      Last observed exit t =  69.99863
. set seed 9123456
. sttocc, match(job) n(5) nodots
note: 2 sets of tied failure times detected; splitting at random.
      Failures are assumed to precede censorings.
Survival-time data settings
            Failure _d: fail
       Analysis time _t: (dox-origin)/365.25
                Origin: time dob
     Enter on or after: time doe
           ID variable: id
           Matching for: job
Converting survival-time data to case-control data:
  Sampling 5 controls for each of 80 cases ...
Data are now case-control data with new variables:
    _case     Case-control indicator
    _set      Case-control ID
    _time     Analysis time of the case's failure
```

The above two commands create a new dataset in which there are five controls per case, matched on job, with the age of the subjects when the case failed recorded in the variable _time. The case indicator is given in _case and the matched set number, in _set. Because we did not specify the optional *varlist*, all variables are carried over into the new dataset.

```
. describe
Contains data from https://www.stata-press.com/data/r18/diet.dta
 Observations:           480                  Diet data with dates
    Variables:            14                  1 May 2022 19:01
```

Variable name	Storage type	Display format	Value label	Variable label
id	int	%9.0g		Subject identity number
fail	byte	%8.0g		Outcome (CHD = 1 3 13)
job	byte	%8.0g		Occupation
month	byte	%8.0g		Month of survey
energy	float	%9.0g		Total energy (1000kcals/day)
height	float	%9.0g		Height (cm)
weight	float	%9.0g		Weight (kg)
hienergy	byte	%9.0g		Indicator for high energy
doe	int	%td		Date of entry
dox	int	%td		Date of exit
dob	int	%td		Date of birth
_case	byte	%8.0g		0 for controls; 1 for cases
_set	long	%12.0g		Case–control ID
_time	double	%10.0g		Analysis time of the case's failure

```
Sorted by: _set  _case
     Note: Dataset has changed since last saved.
```

We can verify that the controls were correctly selected:

```
. gen ageentry=(doe-dob)/365.25
. gen ageexit=(dox-dob)/365.25
. sort _set _case id
. list _set id _case _time ageentry ageexit job, sepby(_set)
```

	_set	id	_case	_time	ageentry	ageexit	job
1.	1	65	0	42.57358	40.11225	56.82409	0
2.	1	73	0	42.57358	36.58043	52.70636	0
3.	1	74	0	42.57358	37.09788	53.39083	0
4.	1	75	0	42.57358	31.13484	47.26078	0
5.	1	86	0	42.57358	38.14921	54.10815	0
6.	1	90	1	42.57358	31.4141	42.57358	0
7.	2	203	0	47.8987	41.26215	61.22108	2
8.	2	207	0	47.8987	43.6386	63.51266	2
9.	2	236	0	47.8987	45.30048	57.42368	2
10.	2	281	0	47.8987	44.34223	61.54963	2
11.	2	333	0	47.8987	46.37645	61.8371	2
12.	2	196	1	47.8987	45.46475	47.8987	2
13.	3	37	0	47.964408	35.2115	52.67351	0
14.	3	66	0	47.964408	40.09309	56.9692	0
				(output omitted)			
479.	80	180	0	68.596851	61.55784	69.99863	1
480.	80	108	1	68.596851	55.72074	68.59686	1

The controls do indeed belong to the appropriate risk set. The controls in each set enter at an age that is less than the age of the case at failure, and they exit at an age that is greater than the age of the case at failure. To estimate the effect of high energy, use [clogit](#), just as you would for any matched case–control study:

```
. clogit _case hienergy, group(_set) or
Iteration 0:  Log likelihood = -143.22071
Iteration 1:  Log likelihood = -143.22071
Conditional (fixed-effects) logistic regression        Number of obs =     480
                                                       LR chi2(1)    =    0.24
                                                       Prob > chi2   =  0.6241
Log likelihood = -143.22071                            Pseudo R2     =  0.0008
```

_case	Odds ratio	Std. err.	z	P>|z|	[95% conf. interval]	
hienergy	.88683	.217505	-0.49	0.624	.54837	1.434191

◁

Acknowledgments

The original version of sttocc was written by David Clayton (retired) of the Cambridge Institute for Medical Research and Michael Hills (1934–2021) of the London School of Hygiene and Tropical Medicine.

References

Clayton, D. G., and M. Hills. 1993. *Statistical Models in Epidemiology*. Oxford: Oxford University Press.

Langholz, B., and D. C. Thomas. 1990. Nested case-control and case-cohort methods of sampling from a cohort: A critical comparison. *American Journal of Epidemiology* 131: 169–176. https://doi.org/10.1093/oxfordjournals.aje.a115471.

Also see

Title

> **sttoct** — Convert survival-time data to count-time data

Description

sttoct converts survival-time (st) data to count-time (ct) data; see [ST] **ct**.

At present, there is absolutely no reason that you would want to do this.

Quick start

Convert survival-time data to count-time data using stset data

```
sttoct
```

Same as above, and specify that counts are recorded for groups identified by v1

```
sttoct, by(v1)
```

Syntax

> sttoct *newfailvar newcensvar* [*newentvar*] [, *options*]

options	Description
by(*varlist*)	reflect counts by group, where groups are defined by observations with equal values of *varlist*
replace	proceed with transformation, even if current data are not saved
noshow	do not show st setting information

You must stset your data before using sttoct; see [ST] **stset**.

fweights, iweights, and pweights may be specified using stset; see [ST] **stset**.

There is no dialog-box interface for sttoct.

Options

by(*varlist*) specifies that counts reflect counts by group where the groups are defined by observations with equal values of *varlist*.

replace specifies that it is okay to proceed with the transformation, even though the current dataset has not been saved on disk.

noshow prevents sttoct from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of every st command; see [ST] **stset**.

Remarks and examples

sttoct is a never-used command that is included only for completeness. The definition of ct data is found in [ST] ct. In the current version of Stata, all you can do with ct data is convert the data to st data (which thus provides access to Stata's survival analysis capabilities to those with ct data), so there is little point in converting st data to ct data.

The converted dataset will contain

varlist	from by(*varlist*), if specified
t	the exit-time variable previously stset
newfailvar	number of failures at *t*
newcensvar	number of censored at *t* (after failures)
newentvar	if specified, number of entries at *t* (after censorings)

The resulting dataset will be ctset automatically.

There are two forms of the sttoct command:

1. sttoct *failvar censvar*, ...

2. sttoct *failvar censvar entvar*, ...

That is, specifying *entvar* makes a difference.

Case 1: entvar not specified

This is possible only if

    a. the risk is not recurring;

    b. the original st data are single-record data, or if the data are multiple-record data, all subjects enter at time 0 and have no gaps thereafter; and

    c. if by(*varlist*) is specified, subjects do not have changing values of the variables in *varlist* over their histories.

If you do not specify *entvar*, sttoct verifies that (a), (b), and (c) are true. If the assumptions are true, sttoct converts your data and counts each subject only once. That is, in multiple-record data, all thrashing (censoring followed by immediate reenter with different covariates) is removed.

Case 2: entvar specified

Any kind of survival-time data can be converted to count-time data with an entry variable. You can convert your data in this way whether assumptions (a), (b), and (c) are true or not.

When you specify a third variable, thrashing is not removed, even if it could be—even if assumptions (a), (b), and (c) are true.

Also see

[ST] **ct** — Count-time data

[ST] **st_is** — Survival analysis subroutines for programmers

[ST] **stset** — Declare data to be survival-time data

[ST] **sttocc** — Convert survival-time data to case–control data

Title

> **stvary** — Report variables that vary over time

Description

stvary is for use with multiple-record datasets, for which id() has been stset. It reports whether values of variables within subject vary over time and reports their pattern of missing values. Although stvary is intended for use with multiple-record st data, it may be used with single-record data as well, but this produces little useful information.

stvary ignores weights, even if you have set them. stvary summarizes the variables in the computer or data sense of the word.

Quick start

Report whether variables vary over time and whether they have missing values using multiple-record
stset data

    stvary

Same as above, but only show report for x1 and x2

    stvary x1 x2

Same as above, but with separate reports for each level of v1

    by v1, sort: stvary x1 x2

Menu

Statistics > Survival analysis > Setup and utilities > Report variables that vary over time

Syntax

> stvary [*varlist*] [*if*] [*in*] [, <u>nosh</u>ow]

You must stset your data before using stvary; see [ST] stset.

by and collect are allowed; see [U] **11.1.10 Prefix commands**.

fweights, iweights, and pweights may be specified using stset; see [ST] stset.

Option

> **Main**

noshow prevents stvary from showing the key st variables. This option is seldom used because most people type stset, show or stset, noshow to set whether they want to see these variables mentioned at the top of the output of every st command; see [ST] stset.

Remarks and examples

Consider a multiple-record dataset. A subject's sex, presumably, does not change, but his or her age might. stvary allows you to verify that values vary in the way that you expect:

```
. use https://www.stata-press.com/data/r18/stan3
(Heart transplant data)
. stvary
        Failure _d: died
  Analysis time _t: t1
      ID variable: id

              Subjects for whom the variable is
                                          never    always sometimes
   Variable |   constant      varying    missing   missing   missing
  _____|_____
       year |        103            0        103         0         0
        age |        103            0        103         0         0
      stime |        103            0        103         0         0
    surgery |        103            0        103         0         0
  transplant |        103            0        103         0         0
       wait |        103            0        103         0         0
    posttran |         34           69        103         0         0
```

That 103 values for year are "constant" does not mean that year itself is a constant—it means merely that, for each subject, the value of year does not change across the records. Whether the values of year vary across subjects is still an open question.

Now look at the bottom of the table: posttran is constant over time for 34 subjects and varies for the remaining 69.

Below we have another dataset, and we will examine just two of the variables:

```
. use https://www.stata-press.com/data/r18/stvaryex
. stvary sex drug
        Failure _d: fail
  Analysis time _t: t
  Exit on or before: time .
      ID variable: id

              Subjects for whom the variable is
                                          never    always sometimes
   Variable |   constant      varying    missing   missing   missing
  _____|_____
        sex |        119            1        119         3         1
       drug |        121            2        123         0         0
```

Clearly, there are errors in the sex variable; for 119 of the subjects, sex does not change over time, but for one, it does. Also we see that we do not know the sex of three of the patients, but for another, we sometimes know it and sometimes do not. The latter must be a simple data-construction error. As for drug, we see that for two of our patients, the drug administered varied over time. Perhaps this is an error, or perhaps those two patients were treated differently from all the rest.

Video example

How to describe and summarize survival data

Stored results

stvary stores the following in `r()`:

Scalars
r(cons)	number of subjects for whom variable is constant when not missing
r(varies)	number of subjects for whom nonmissing values vary
r(never)	number of subjects for whom variable is never missing
r(always)	number of subjects for whom variable is always missing
r(miss)	number of subjects for whom variable is sometimes missing

Reference

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata Press.

Also see

[ST] **stdescribe** — Describe survival-time data

[ST] **stfill** — Fill in by carrying forward values of covariates

[ST] **stset** — Declare data to be survival-time data

Title

---

adjustfor_option — Adjust survivor and related functions for covariates at specific values

---

Description

The `sts`, `stphplot`, and `stintphplot` commands support the `adjustfor(`*varlist*`)` option to adjust the estimates of the survivor and related functions for covariates in *varlist* at specific values. You can adjust the functions to overall means, group-specific means, and zero covariate values. And you can use the `at()` option, specified within `adjustfor()` with `sts`, `stphplot`, or `stintphplot` and directly with `stcurve`, to adjust for specific covariate values and other summary statistics.

Quick start

Adjust for average values of covariates `x1`, `x2`, and factor variable `x3`
>   *st_command* ... , ... `adjustfor(x1 x2 i.x3, atmeans)`

Adjust for `x1` $= 0$, `x2` $= 0$, and the base level of `x3`
>   *st_command* ... , ... `adjustfor(x1 x2 i.x3, atzeros)`

Adjust for `x1` $= 3$, `x2` $= 4$, and `x3` $= 1$
>   *st_command* ... , ... `adjustfor(x1 x2 i.x3, at(x1=3 x2=4 x3=1))`

Adjust for the means of `x1`, `x2`, and the base level of factor variable `x3`
>   *st_command* ... , ... `adjustfor(x1 x2 i.x3, atmeans atbase)`

The same as above, but using the `at()` specification
>   *st_command* ... , ... `adjustfor(x1 x2 i.x3, at((mean) x1 x2 (base) x3))`

The same as above, but for `stcurve`
>   `stcurve, survival at((mean) x1 x2 (base) x3)`

Adjust for group-specific means of covariates `x1` and `x2`
>   *st_command* ... , ... `adjustfor(x1 x2, atmeans) by(group)`

Adjust for overall means of covariates `x1` and `x2`
>   *st_command* ... , ... `adjustfor(x1 x2, atomeans) by(group)`

Syntax

Adjust for covariates in sts commands, stphplot, and stintphplot

    st_command ... $\Big[$, ... <u>adjustfor</u>(*varlist* $\big[$, *suboptions* $\big]$) ... $\Big]$

where *st_command* is one of sts graph, sts list, sts generate, stphplot, or stintphplot.

Adjust for specific covariate values in stcurve

    stcurve ... $\Big[$, ... at(*atspec*) ... $\Big]$

suboptions	Description
atmeans	adjust for *varlist* at group-specific means
atomeans	adjust for *varlist* at overall means
atzeros	adjust for *varlist* at zeros
atbase	adjust for factor variables in *varlist* at base levels
at(*atspec*)	adjust for *varlist* at specified values

at() overrides atmeans, atomeans, atzeros, and atbase; see *Syntax of at()*.

atomeans is the default for stphplot and stintphplot, and atzeros is the default for sts graph, sts list, and sts generate.

For stcurve, at() is a standalone option and can be repeated.

Options

atmeans adjusts the estimates of the survivor and related functions to group-specific means of *varlist*. It may not be combined with atomeans or atzeros. atmeans is equivalent to atomeans when option by() is not specified with *st_command*.

atomeans adjusts the estimates of the survivor and related functions to the overall means of *varlist*. It may not be combined with atmeans or atzeros.

atzeros adjusts the estimates of the survivor and related functions to zero values of continuous variables in *varlist* and base levels of factor variables. It may not be combined with atomeans or atmeans.

atbase adjusts the estimates of the survivor and related functions to the base levels of the factor variables in *varlist*. Without this option, the factor variables are set to their mean values, unless atzeros is specified or assumed by the command.

at(*atspec*) adjusts the estimates of the survivor and related functions to the specified values of covariates in *varlist*. at() may be used for continuous or factor covariates. With stcurve, multiple at() options can be specified, and each will produce a different set of estimates.

    at(x1=20) uses the specified value of covariate x1.

    at(x1=20 x2=1) uses the specified values of covariates x1 and x2.

    at(x1=(20 30 40 50)) (stcurve only) first uses the value of 20 for x1, then the value of 30, and so on. stcurve produces separate results for each specified value.

    at(x1=(20(10)50)) (stcurve only) does the same as at(x1=(20 30 40 50)); that is, you may specify a numlist.

at((mean) x1 (median) x2) uses the specified summary statistics as the values for x1 and x2. at((p25) _all) uses the respective 25th percentile values for all covariates. See *Syntax of at()* for the full list of summary-statistic modifiers.

at((mean) _all (median) x x2=1.2 z=(1 2 3)) is processed from general to specific, with settings for named covariates overriding general settings specified via _all. Thus, all covariates are set to their means except for x (set to its median), x2 (set to 1.2), and z (set to 1, then to 2, and finally to 3).

See *Syntax of at()* for more information.

Remarks and examples

Remarks are presented under the following headings:

> *Using adjustfor() with sts, stphplot, and stintphplot*
> *Syntax of at()*

For examples of using the adjustfor() option, see *Covariate-adjusted estimates* in [ST] **sts**, example 1 in [ST] **stcox PH-assumption tests**, and example 3 in [ST] **stintcox PH-assumption plots**. For examples of specifying at(), see *Using at() with stcurve* in [ST] **stcurve**.

Using adjustfor() with sts, stphplot, and stintphplot

When you specify the adjustfor(*varlist*) option with sts, stphplot, or stintphplot, the command fits a Cox model with *varlist* as predictors and computes the survivor (or related) function for the specified values of *varlist*. By default, sts adjusts to zero covariate values (computes the baseline functions), and stphplot and stintphplot adjust to overall means. That is, when adjustfor(*varlist*) is specified, sts assumes adjustfor(*varlist*, atzeros), and stphplot and stintphplot assume adjustfor(*varlist*, atomeans).

For instance, the following two commands are producing the same "log–log" plots.

```
. stphplot, by(group) adjustfor(x1 x2)
. stphplot, by(group) adjustfor(x1 x2, atomeans)
```

To use group-specific means instead of the overall means, we could specify

```
. stphplot, by(group) adjustfor(x1 x2, atmeans)
```

To use zero values, we could specify

```
. stphplot, by(group) adjustfor(x1 x2, atzeros)
```

As we mentioned earlier, sts or, more specifically, sts graph, sts list, and sts generate adjust the function estimates to zero covariate values by default. For instance, the following two commands plot the same baseline survivor function.

```
. sts graph, by(drug) adjustfor(age)
. sts graph, by(drug) adjustfor(age, atzeros)
```

If we wanted the survivor curves adjusted for age and scaled to age 50, we could generate a new variable centered at 50 and adjust the baseline survivor curve to that new variable:

```
. generate double age50 = age-50
. sts graph, by(drug) adjustfor(age50)
```

More conveniently, we could simply use the at() suboption of the adjustfor() option to adjust for age 50:

```
. sts graph, by(drug) adjustfor(age, at(age=50))
```

Syntax of at()

The `at()` option can be specified within the `adjustfor()` option with the `sts`, `stphplot`, and `stintphplot` commands. It can also be used directly with [stcurve](#).

In at(*atspec*), *atspec* may contain one or more of the following specifications,

> *varname* = #
>
> *varname* = (*numlist*) (stcurve only)
>
> _frailty = (*numlist*) (stcurve only)
>
> (*stat*) *varlist*

where

1. *varname* and *varlist* must contain covariates from *varlist* specified in `adjustfor()` or, for `stcurve`, *varlist* specified with the survival regression model.

2. variables (whether in *varname* or *varlist*) may be continuous variables, factor variables, or specific-level variables, such as `age`, `group`, or `3.group`.

3. *varlist* may also be one of three standard lists:

   a. _all (all covariates);

   b. _factor (all factor-variable covariates); or

   c. _continuous (all continuous covariates).

4. *stat* can be any of the following:

stat	Description	Variables allowed
mean	means	all
median	medians	continuous
p1	1st percentile	continuous
p2	2nd percentile	continuous
...	3rd–49th percentiles	continuous
p50	50th percentile (same as median)	continuous
...	51st–97th percentiles	continuous
p98	98th percentile	continuous
p99	99th percentile	continuous
min	minimums	continuous
max	maximums	continuous
zero	zeros	all
base	base level	factors

Any *stat* except zero and base may be prefixed with an o to get the overall statistic such as omean, omedian, and op25. Overall statistics differ from their correspondingly named statistics only when the by() option is specified. For factor variables, zero is a synonym for base. If *stat* is not followed by a varlist, *stat* is ignored.

5. _frailty is allowed only in the at() option with stcurve after fitting a shared-frailty model with stcox. When _frailty is specified, frailties are set to the values in the corresponding numlist. When _frailty is not specified, frailties are set to 1.

atspec can involve settings for multiple covariates and, with stcurve, multiple settings for one covariate. The following rules are applied when more than one covariate or value is included:

1. When more than one covariate is referenced in *atspec* but each covariate is set to only one value, all settings are applied in combination. For example, at(x1=5 x2=0) results in one scenario, with x1 set to 5, x2 set to 0, and all other covariates set to their defaults.

2. When multiple values are specified for a covariate, the covariate will be set to each of the values in turn. For example, at(x1=5 x1=10) or, equivalently, at(x1=(5 10)) specifies that x1 be set first to 5 and then to 10. This is allowed only with the stcurve command.

3. When multiple values are specified for more than one covariate, all possible combinations of settings are applied in turn. For example, at(x1=(5 10) x2=(0 1)) results in four scenarios: x1 = 5 and x2 = 0; x1 = 5 and x2 = 1; x1 = 10 and x2 = 0; and x1 = 10 and x2 = 1. This is allowed only with the stcurve command.

4. Settings may be specified for groups of covariates using three general varlists—_all, _factor, and _continuous. When *atspec* includes both specifications with general varlists and specifications with named covariates, the specifications for named covariates take precedence over general ones. For example, at(x1=10 (means) _all) sets x1 to 10 while setting all other covariates to their means.

5. Only one (*stat*) *varlist* specification can be applied to a covariate. If more than one is specified, the rightmost specification is respected. For example, at((means) x1 x2 (medians) x1 x2) sets both x1 and x2 to their medians.

6. When both a (*stat*) specification and another specification are included for a named covariate, the other specification takes precedence. For example, at(x1=5 (means) x1) sets x1 to 5.

In addition, with the stcurve command, at() can be repeated. When multiple at() options are specified, *atspec*s are processed sequentially. For instance, at(x1=5) at(x2=0) results in stcurve producing two curves. The first sets x1 to 5 and all other covariates, including x2, to their means. The second sets x2 to 0 and all other covariates to their means. Note that this is different from the single at(x1=5 x2=0) specification, which sets x1 and x2 to the specified values simultaneously.

Also see

[ST] **stcox PH-assumption tests** — Tests of proportional-hazards assumption after stcox

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[ST] **sts** — Generate, graph, list, and test the survivor and related functions

[ST] **sts generate** — Create variables containing survivor and related functions

[ST] **sts graph** — Graph the survivor or related function

[ST] **sts list** — List the survivor or related function

Glossary

accelerated failure-time model. A model in which everyone has, in a sense, the same survivor function, $S(\tau)$, and an individual's τ_j is a function of his or her characteristics and of time, such as $\tau_j = t * \exp(\beta_0 + \beta_1 x_{1j} + \beta_2 x_{2j})$.

AFT, accelerated failure time. See *accelerated failure-time model*.

analysis time. Analysis time is like time, except that 0 has a special meaning: $t = 0$ is the time of onset of risk, the time when failure first became possible.

Analysis time is usually not what is recorded in a dataset. A dataset of patients might record calendar time. Calendar time must then be mapped to analysis time.

The letter t is reserved for time in analysis-time units. The term *time* is used for time measured in other units.

The *origin* is the *time* corresponding to $t = 0$, which can vary subject to subject. Thus $t = time - origin$.

at risk. A subject is at risk from the instant the first failure event becomes possible and usually stays that way until failure, but a subject can have periods of being at risk and not at risk.

attributable fraction. An attributable fraction is the reduction in the risk of a disease or other condition of interest when a particular risk factor is removed.

baseline. In survival analysis, baseline is the state at which the covariates, usually denoted by the row vector $\mathbf{x}$, are zero. For example, if the only measured covariate is systolic blood pressure, the baseline survivor function would be the survivor function for someone with zero systolic blood pressure. This may seem ridiculous, but covariates are usually centered so that the mathematical definition of baseline (covariate is zero) translates into something meaningful (mean systolic blood pressure).

baseline covariates, **time-independent covariates**, **time-invariant covariates**, and **constant covariates**. Covariates whose values are constant over time, such as those recorded at baseline (at time 0).

boundary kernel. A boundary kernel is a special kernel used to smooth hazard functions in the boundaries of the data range. Boundary kernels are applied when the `epan2`, `biweight`, or `rectangle kernel()` is specified with `stcurve, hazard` or `sts graph, hazard`.

case I interval-censored data and **current status data**. Case I interval-censored data occur when the only survival information available is whether the event of interest occurred before or after the observed time, leading to data in which an observation is either left-censored or right-censored. Case I interval-censored data can be viewed as a special case of case II interval-censored data without uncensored and interval-censored on $(t_l, t_u]$ observations.

case II interval-censored data and **general interval-censored data**. Case II interval censored data occur when, for some observations, we do not know the exact failure time t, but only know that the failure happened within a random time interval $(t_l, t_u]$, or before the left endpoint of the time interval t_l, or after the right endpoint of the time interval t_u.

cause-specific hazard. In a competing-risks analysis, the cause-specific hazard is the hazard function that generates the events of a given type. For example, if heart attack and stroke are competing events, then the cause-specific hazard for heart attacks describes the biological mechanism behind heart attacks independently of that for strokes. Cause-specific hazards can be modeled using Cox regression, treating the other events as censored.

censored, **uncensored**, **left-censored**, **right-censored**, and **interval-censored**. An observation is cen-
sored when the exact time of failure is not known, and it is uncensored when the exact time of
failure is known.

An observation is left-censored when the exact time of failure is not known; it is merely known
that the failure occurred before t_l. Suppose that the event of interest is becoming employed. If a
subject is already employed when first interviewed, his outcome is left-censored.

An observation is right-censored when the time of failure is not known; it is merely known that
the failure occurred after t_r. If a patient survives until the end of a study, the patient's time of
death is right-censored.

An observation is interval-censored when the time of failure is not known; it is merely known
that the failure occurred after t_l but before t_r. Suppose that the event of interest is an onset of
breast cancer. Patients are assessed periodically during their yearly checkups. The actual time of
the onset of the disease, if present, is rarely known. Often, it is only known that the disease
happened between the last and the current checkups. The time to the onset of breast cancer is then
interval-censored.

In common usage, *censored* without a modifier means right-censored.

Also see *truncation, left-truncation, and right-truncation*.

CIF. See *cumulative incidence function*.

competing risks. Competing risks models are survival-data models in which the failures are generated
by more than one underlying process. For example, death may be caused by either heart attack or
stroke. There are various methods for dealing with competing risks. One direct way is to duplicate
failures for one competing risk as censored observations for the other risk and stratify on the risk
type. Another is to directly model the cumulative incidence of the event of interest in the presence
of competing risks. The former method uses `stcox` and the latter, `stcrreg`.

confounding. In the analysis of contingency tables, factor or interaction effects are said to be
confounded when the effect of one factor is combined with that of another. For example, the
effect of alcohol consumption on esophageal cancer may be confounded with the effects of age,
smoking, or both. In the presence of confounding, it is often useful to stratify on the confounded
factors that are not of primary interest, in the above example, age and smoking.

constant covariates. See *baseline covariates, time-independent covariates, time-invariant covariates,
and constant covariates*.

count-time data. See *ct data*.

covariates. Covariates are the explanatory variables that appear in a model. For instance, if survival
time were to be explained by age, sex, and treatment, then those variables would be the covariates.
Also see *time-varying covariates*.

crude estimate. A crude estimate has not been adjusted for the effects of other variables. Disregarding
a stratification variable, for example, yields a crude estimate.

ct data. ct stands for count time. ct data are an aggregate organized like a life table. Each observation
records a time, the number known to fail at that time, the number censored, and the number of
new entries. See [ST] **ctset**.

cumulative hazard. See *hazard, cumulative hazard, and hazard ratio*.

cumulative incidence estimator. In a competing-risks analysis, the cumulative incidence estimator estimates the cumulative incidence function (CIF). Assume for now that you have one event of interest (type 1) and one competing event (type 2). The cumulative incidence estimator for type 1 failures is then obtained by

$$\widehat{\text{CIF}}_1(t) = \sum_{j:t_j \leq t} \widehat{h}_1(t_j)\widehat{S}(t_{j-1})$$

with

$$\widehat{S}(t) = \prod_{j:t_j \leq t} \left\{1 - \widehat{h}_1(t_j) - \widehat{h}_2(t_j)\right\}$$

The t_j index the times at which events (of any type) occur, and $\widehat{h}_1(t_j)$ and $\widehat{h}_2(t_j)$ are the cause-specific hazard contributions for type 1 and type 2, respectively. $\widehat{S}(t)$ estimates the probability that you are event free at time t.

The above generalizes to multiple competing events in the obvious way.

cumulative incidence function. In a competing-risks analysis, the cumulative incidence function, or CIF, is the probability that you will observe the event of primary interest before a given time. Formally,

$$\text{CIF}(t) = P(T \leq t \text{ and event type of interest})$$

for time-to-failure, T.

cumulative subhazard. See *subhazard, cumulative subhazard, and subhazard ratio*.

current status data. See *case I interval-censored data*.

DFBETA. A DFBETA measures the change in the regressor's coefficient because of deletion of that subject. Also see *partial DFBETA*.

effect size. The effect size is the size of the clinically significant difference between the treatments being compared, often expressed as the hazard ratio (or the log of the hazard ratio) in survival analysis.

event. An event is something that happens at an instant in time, such as being exposed to an environmental hazard, being diagnosed as myopic, or becoming employed.

The failure event is of special interest in survival analysis, but there are other equally important events, such as the exposure event, from which analysis time is defined.

In st data, events occur at the end of the recorded time span.

event of interest. In a competing-risks analysis, the event of interest is the event that is the focus of the analysis, that for which the cumulative incidence in the presence of competing risks is estimated.

event time. The time of the occurrence of an event of interest. With right-censored data, it is synonymous with survival time. With interval-censored data, it is more proper to use the term "event time" rather than "survival time".

event-time interval. In the context of interval-censored data, the event-time interval is the interval that brackets the event time.

event-time variables. In the context of interval-censored data, event-time variables correspond to the variables that contain the lower and upper time endpoints of the event-time interval.

examination time. In the context of interval-censored data, the examination time is the time when the data are collected on a subject. It is not the time of the occurrence of an event of interest. Multiple-record interval-censored data record examination times.

external covariates. Time-varying covariates that are external to an individual under study, such as air temperature and various experimental settings. The marginal distribution of external covariates does not depend on any failure-time model parameters. Baseline covariates are external covariates because they are constant over time. Any time-varying covariates whose values are predetermined or can be computed deterministically are external covariates. For instance, any interaction of a baseline covariate with a deterministic function of time is an external covariate. Unlike with internal covariates, the survivor function is well defined in the presence of external covariates.

failure event. Survival analysis is really time-to-failure analysis, and the failure event is the event under analysis. The failure event can be death, heart attack, myopia, or finding employment. Many authors—including Stata—write as if the failure event can occur only once per subject, but when we do, we are being sloppy. Survival analysis encompasses repeated failures, and all of Stata's survival analysis features can be used with repeated-failure data.

failure function. The failure function, $F(t)$, is the probability of failing (experiencing a failure event) before or at time t. It is a cumulative distribution function of T and the reverse of the survivor function: $F(t) = \Pr(T \le t) = 1 - S(t)$. Also see survivor function.

frailty. In survival analysis, it is often assumed that subjects are alike—homogeneous—except for their observed differences. The probability that subject j fails at time t may be a function of j's covariates and random chance. Subjects j and k, if they have equal covariate values, are equally likely to fail.

Frailty relaxes that assumption. The probability that subject j fails at time t becomes a function of j's covariates and j's unobserved frailty value, ν_j. Frailty ν is assumed to be a random variable. Parametric survival models can be fit even in the presence of such heterogeneity.

Shared frailty refers to the case in which groups of subjects share the same frailty value. For instance, subjects 1 and 2 may share frailty value ν because they are genetically related. Both parametric and semiparametric models can be fit under the shared-frailty assumption.

future history. Future history is information recorded after a subject is no longer at risk. The word *history* is often dropped, and the term becomes simply *future*. Perhaps the failure event is cardiac infarction, and you want to know whether the subject died soon in the *future*, in which case you might exclude the subject from analysis.

Also see past history.

gaps. Gaps refers to gaps in observation between entry time and exit time; see under observation.

general interval-censored data. See case II interval-censored data.

hazard, **cumulative hazard**, and **hazard ratio**. The hazard or hazard rate at time t, $h(t)$, is the instantaneous rate of failure at time t conditional on survival until time t. Hazard rates can exceed 1. Say that the hazard rate were 3. If an individual faced a constant hazard of 3 over a unit interval and if the failure event could be repeated, the individual would be expected to experience three failures during the time span.

The cumulative hazard, $H(t)$, is the integral of the hazard function $h(t)$, from 0 (the onset of risk) to t. It is the total number of failures that would be expected to occur up until time t, if the failure event could be repeated. The relationship between the cumulative hazard function, $H(t)$, and the survivor function, $S(t)$, is

$$S(t) = \exp\{-H(t)\}$$
$$H(t) = -\ln\{S(t)\}$$

The hazard ratio is the ratio of the hazard function evaluated at two different values of the covariates: $h(t \mid \mathbf{x})/h(t \mid \mathbf{x}_0)$. The hazard ratio is often called the relative hazard, especially when $h(t \mid \mathbf{x}_0)$ is the baseline hazard function.

hazard contributions. Hazard contributions are the increments of the estimated cumulative hazard function obtained through either a nonparametric or semiparametric analysis. For these analysis types, the estimated cumulative hazard is a step function that increases every time a failure occurs. The hazard contribution for that time is the magnitude of that increase.

Because the time between failures usually varies from failure to failure, hazard contributions do not directly estimate the hazard. However, one can use the hazard contributions to formulate an estimate of the hazard function based on the method of smoothing.

ID variable. An ID variable identifies groups; equal values of an ID variable indicate that the observations are for the same group. For instance, a stratification ID variable would indicate the strata to which each observation belongs.

When an ID variable is referred to without modification, it means subjects, and usually this occurs in multiple-record st data. In multiple-record data, each physical observation in the dataset represents a time span, and the ID variable ties the separate observations together:

idvar	t0	t
1	0	5
1	5	7

ID variables are usually numbered 1, 2, ..., but that is not required. An ID variable might be numbered 1, 3, 7, 22, ..., or -5, -4, ..., or even 1, 1.1, 1.2, ...

incidence and **incidence rate**. Incidence is the number of new failures (for example, number of new cases of a disease) that occur during a specified period in a population at risk (for example, of the disease).

Incidence rate is incidence divided by the sum of the length of time each individual was exposed to the risk.

Do not confuse incidence with prevalence. Prevalence is the fraction of a population that has the disease. Incidence refers to the rate at which people contract a disease, whereas prevalence is the total number actually sick at a given time.

internal covariates. Time-varying covariates that are generated by an individual under study, such as blood measurements and wellness status. The internal covariate cannot be determined without an individual in a study and thus naturally contains information about the failure-time process. The mere existence of a measurement for an individual implies his or her survival. The survivor function and thus various predictions are not well defined in the presence of internal covariates. Also see external covariates.

interval-censored data. Interval-censored data may contain different types of censoring: right-censoring, left-censoring, or interval-censoring. Interval-censoring occurs when the failure time or event time is not exactly observed but is known only to lie within some interval. See case I interval-censored data or current status data and case II interval-censored data of general interval-censored data.

Interval-censored data can be stored in two different formats. In a single-record-per-subject format, the event-time information is recorded as interval data containing the lower and upper endpoints of the event-time interval (see *event-time variables*) for each subject. In a multiple-record-per-subject format, the event-time information is recorded as potentially multiple *examination times* per subject with the corresponding event status indicators for each examination time. The latter format is useful for storing *time-varying covariates*.

Kaplan–Meier product-limit estimate. This is an estimate of the survivor function, which is the product of conditional survival to each time at which an event occurs. The simple form of the calculation, which requires tallying the number at risk and the number who die and at each time, makes accounting for censoring easy. The resulting estimate is a step function with jumps at the event times.

left-censored. See *censored, uncensored, left-censored, right-censored, and interval-censored*.

left-truncation. See *truncation, left-truncation, and right-truncation*.

life table. Also known as a mortality table or actuarial table, a life table is a table that shows for each analysis time the fraction that survive to that time. In mortality tables, analysis time is often age.

likelihood displacement value. A likelihood displacement value is an influence measure of the effect of deleting a subject on the overall coefficient vector. Also see *partial likelihood displacement value*.

LMAX value. An LMAX value is an influence measure of the effect of deleting a subject on the overall coefficient vector and is based on an eigensystem analysis of efficient score residuals. Also see *partial LMAX value*.

multiarm trial. A multiarm trial is a trial comparing survivor functions of more than two groups.

multiple-record interval-censored data and **multiple-record-per-subject interval-censored data**. See *interval-censored data*.

multiple-record st data. See *st data*.

nonparametric maximum-likelihood estimation. Nonparametric maximum-likelihood estimation is maximum likelihood estimation of a nonparametric density function. In the context of NPMLE of interval-censored data, the likelihood function is viewed as a function of both regression coefficients and the baseline cumulative hazard function, which is infinite dimensional. Under NPMLE, the baseline cumulative hazard is approximated by a step function with nonnegative jumps at unique values of the observed *event-time intervals*. See [ST] **stintcox**.

odds and **odds ratio**. The odds in favor of an event are $o = p/(1 - p)$, where p is the probability of the event. Thus if $p = 0.2$, the odds are 0.25, and if $p = 0.8$, the odds are 4.

The log of the odds is $\ln(o) = \text{logit}(p) = \ln\{p/(1 - p)\}$, and logistic-regression models, for instance, fit $\ln(o)$ as a linear function of the covariates.

The odds ratio is a ratio of two odds: o_1/o_0. The individual odds that appear in the ratio are usually for an experimental group and a control group, or two different demographic groups.

offset variable and **exposure variable**. An offset variable is a variable that is to appear on the right-hand side of a model with coefficient 1:

$$y_j = \text{offset}_j + b_0 + b_1 x_j + \cdots$$

In the above, b_0 and b_1 are to be estimated. The offset is not constant. Offset variables are often included to account for the amount of exposure. Consider a model where the number of events

observed over a period is the length of the period multiplied by the number of events expected in a unit of time:

$$n_j = T_j e(X_j)$$

When we take logs, this becomes

$$\log(n_j) = \log(T_j) + \log\{e(X_j)\}$$

$\ln(T_j)$ is an offset variable in this model.

When the log of a variable is an offset variable, the variable is said to be an exposure variable. In the above, T_j is an exposure variable.

partial DFBETA. A partial DFBETA measures the change in the regressor's coefficient because of deletion of that individual record. In single-record data, the partial DFBETA is equal to the DFBETA. Also see *DFBETA*.

partial likelihood displacement value. A partial likelihood displacement value is an influence measure of the effect of deleting an individual record on the coefficient vector. For single-record data, the partial likelihood displacement value is equal to the likelihood displacement value. Also see *likelihood displacement value*.

partial LMAX value. A partial LMAX value is an influence measure of the effect of deleting an individual record on the overall coefficient vector and is based on an eigensystem analysis of efficient score residuals. In single-record data, the partial LMAX value is equal to the LMAX value. Also see *LMAX value*.

past history. Past history is information recorded about a subject before the subject was both *at risk* and *under observation*. Consider a dataset that contains information on subjects from birth to death and an analysis in which subjects became at risk once diagnosed with a particular kind of cancer. The past history on the subject would then refer to records before the subjects were diagnosed.

The word *history* is often dropped, and the term becomes simply *past*. For instance, we might want to know whether a subject smoked in the past.

Also see *future history*.

penalized log-likelihood function. This is a log-likelihood function that contains an added term, usually referred to as a roughness penalty, that reduces its value when the model overfits the data. In Cox models with frailty, such functions are used to prevent the variance of the frailty from growing too large, which would allow the individual frailty values to perfectly fit the data.

power. The power of a test is the probability of correctly rejecting the null hypothesis when it is false. It is often denoted as $1 - \beta$ in statistical literature, where β is the type II error probability. Commonly used values for power are 80% and 90%. Also see *type I error* and *type II error*.

profile likelihood function. A profile likelihood function is a likelihood function in which the parameter is partitioned into two parts, (θ, η), where θ is a low-dimensional parameter of interest and η is a higher-dimensional nuisance parameter. Let $I_n(\theta, \eta)$ be the full likelihood. Then, the profile likelihood for θ is defined as

$$pl_n(\theta) = \sup_\eta I_n(\theta, \eta)$$

For instance, `stintcox` estimates the standard errors of regression coefficients $\boldsymbol{\beta}$ using the profile likelihood function of $\boldsymbol{\beta}$ by treating baseline hazard contributions h_k's as nuisance parameters.

proportional hazards model. This is a model in which, between individuals, the ratio of the instantaneous failure rates (the hazards) is constant over time.

right-censored. See *censored, uncensored, left-censored, right-censored, and interval-censored*.

right-truncation. See *truncation, left-truncation, and right-truncation*.

risk factor. This is a variable associated with an increased or decreased risk of failure.

risk pool. At a particular point in time, this is the subjects at risk of failure.

semiparametric model. This is a model that is not fully parameterized. The Cox proportional hazards model is such a model:

$$h(t) = h_0(t) \exp(\beta_1 x_1 + \cdots + \beta_k x_k)$$

In the Cox model, $h_o(t)$ is left unparameterized and not even estimated. Meanwhile, the relative effects of covariates are parameterized as $\exp(\beta_1 x_1 + \cdots + \beta_k x_k)$.

shape parameter. A shape parameter governs the shape of a probability distribution. One example is the parameter p of the Weibull model.

single-record interval-censored data and **single-record-per-subject interval-censored data**. See *interval-censored data*.

single-record st data. See *st data*.

singleton-group data. A singleton is a frailty group that contains only 1 observation. A dataset containing only singletons is known as singleton-group data.

SMR. See *standardized mortality (morbidity) ratio*.

snapshot data. Snapshot data are those in which each record contains the values of a set of variables for a subject at an instant in time. The name arises because each observation is like a snapshot of the subject.

In snapshot datasets, one usually has a group of observations (snapshots) for each subject.

Snapshot data must be converted to st data before they can be analyzed. This requires making assumptions about what happened between the snapshots. See [ST] **snapspan**.

spell data. Spell data are survival data in which each record represents a fixed period, consisting of a begin time, an end time, possibly a censoring/failure indicator, and other measurements (covariates) taken during that specific period.

st data. st stands for survival time. In survival-time data, each observation represents a span of survival, recorded in variables $t0$ and t. For instance, if in an observation $t0$ were 3 and t were 5, the span would be $(t0, t]$, meaning from just after $t0$ up to and including t.

Sometimes variable $t0$ is not recorded; $t0$ is then assumed to be 0. In such a dataset, an observation that had $t = 5$ would record the span $(0, 5]$.

Each observation also includes a variable d, called the failure variable, which contains 0 or nonzero (typically, 1). The failure variable records what happened at the end of the span: 0, the subject was still alive (had not yet failed) or 1, the subject died (failed).

Sometimes variable d is not recorded; d is then assumed to be 1. In such a dataset, all time-span observations would be assumed to end in failure.

Finally, each observation in an st dataset can record the entire history of a subject or each can record a part of the history. In the latter case, groups of observations record the full history. One observation might record the period $(0, 5]$ and the next, $(5, 8]$. In such cases, there is a variable ID that records the subject for which the observation records a time span. Such data are called multiple-record st data. When each observation records the entire history of a subject, the data are called single-record st data. In the single-record case, the ID variable is optional.

See [ST] **stset**.

standardized mortality (morbidity) ratio. Standardized mortality (morbidity) ratio (SMR) is the observed number of deaths divided by the expected number of deaths. It is calculated using indirect standardization: you take the population of the group of interest—say, by age, sex, and other factors—and calculate the expected number of deaths in each cell (expected being defined as the number of deaths that would have been observed if those in the cell had the same mortality as some other population). You then take the ratio to compare the observed with the expected number of deaths. For instance,

Age	(1) Population of group	(2) Deaths per 100,000 in general pop.	(1)×(2) Expected # of deaths	(4) Observed deaths
25–34	95,965	105.2	100.9	92
34–44	78,280	203.6	159.4	180
44–54	52,393	428.9	224.7	242
55–64	28,914	964.6	278.9	312
Total			763.9	826

$$\text{SMR} = 826/763.9 = 1.08$$

stratified model. A stratified survival model constrains regression coefficients to be equal across levels of the stratification variable, while allowing other features of the model to vary across strata.

stratified test. A stratified test is performed separately for each stratum. The stratum-specific results are then combined into an overall test statistic.

subhazard, **cumulative subhazard**, and **subhazard ratio**. In a competing-risks analysis, the hazard of the subdistribution (or subhazard for short) for the event of interest (type 1) is defined formally as

$$\overline{h}_1(t) = \lim_{\delta \to 0} \left\{ \frac{P(t < T \leq t + \delta \text{ and event type 1}) | \; T > t \text{ or } (T \leq t \text{ and not event type 1})}{\delta} \right\}$$

Less formally, think of this hazard as that which generates failure events of interest while keeping subjects who experience competing events "at risk" so that they can be adequately counted as not having any chance of failing.

The cumulative subhazard $\overline{H}_1(t)$ is the integral of the subhazard function $\overline{h}_1(t)$, from 0 (the onset of risk) to t. The cumulative subhazard plays a very important role in competing-risks analysis. The cumulative incidence function (CIF) is a direct function of the cumulative subhazard:

$$\text{CIF}_1(t) = 1 - \exp\{-\overline{H}_1(t)\}$$

The subhazard ratio is the ratio of the subhazard function evaluated at two different values of the covariates: $\overline{h}_1(t|\mathbf{x})/\overline{h}_1(t|\mathbf{x}_0)$. The subhazard ratio is often called the relative subhazard, especially when $\overline{h}_1(t|\mathbf{x}_0)$ is the baseline subhazard function.

survival-time data. See *st data*.

survivor function. Also known as the survivorship function and the survival function, the survivor function, $S(t)$, is 1) the probability of surviving beyond time t, or equivalently, 2) the probability that there is no failure event prior to t, 3) the proportion of the population surviving to time t, or equivalently, 4) the reverse cumulative distribution function of T, the time to the failure event: $S(t) = \Pr(T > t)$. Also see *hazard*.

thrashing. Subjects are said to thrash when they are censored and immediately reenter with different covariates.

time-dependent covariates. See *time-varying covariates.*

time-independent covariates. See *baseline covariates, time-independent covariates, time-invariant covariates, and constant covariates.*

time-invariant covariates. See *baseline covariates, time-independent covariates, time-invariant covariates, and constant covariates.*

time-varying covariates. Time-varying covariates appear in a survival model whose values vary over time. The values of the covariates vary, not the effect. For instance, in a proportional hazards model, the log hazard at time t might be $b \times \text{age}_t + c \times \text{treatment}_t$. Variable age might be time varying, meaning that as the subject ages, the value of age changes, which correspondingly causes the hazard to change. The effect b, however, remains constant.

Time-varying variables are either continuously varying or discretely varying.

In the continuously varying case, the value of the variable x at time t is $x_t = x_0 + f(t)$ or $x_t = x_0 \times f(t)$, where $f()$ is some function and often is the identity function, so that $x_t = x_0 + t$ or $x_t = x_0 \times t$.

In the discretely varying case, the value of x changes at certain times and often in no particular pattern:

idvar	*t0*	*t*	*bp*
1	0	5	150
1	5	7	130
1	7	9	135

In the above data, the value of *bp* is 150 over the period $(0, 5]$, then 130 over $(5, 7]$, and 135 over $(7, 9]$.

Also see *external covariates* and *internal covariates.*

truncation, **left-truncation**, and **right-truncation**. In survival analysis, truncation occurs when subjects are observed only if their failure times fall within a certain observational period of a study. Censoring, on the other hand, occurs when subjects are observed for the whole duration of a study, but the exact times of their failures are not known; it is known only that their failures occurred within a certain time span.

Left-truncation occurs when subjects come under observation only if their failure times exceed some time t_l. It is only because they did not fail before t_l that we even knew about their existence. Left-truncation differs from left-censoring in that, in the censored case, we know that the subject failed before time t_l, but we just do not know exactly when.

Imagine a study of patient survival after surgery, where patients cannot enter the sample until they have had a post-surgical test. The patients' survival times will be left-truncated. This is a "delayed entry" problem, one common type of left-truncation.

Right-truncation occurs when subjects come under observation only if their failure times do not exceed some time t_r. Right-truncated data typically occur in registries. For example, a cancer registry includes only subjects who developed a cancer by a certain time, and thus survival data from this registry will be right-truncated.

type I error and **false-positive result**. The type I error of a test is the error of rejecting the null hypothesis when it is true. The probability of committing a type I error, significance level of a

test, is often denoted as α in statistical literature. One traditionally used value for α is 5%. Also see *type II error* and *power*.

type II error and **false-negative result**. The type II error of a test is the error of not rejecting the null hypothesis when it is false. The probability of committing a type II error is often denoted as β in statistical literature. Commonly used values for β are 20% or 10%. Also see *type I error* and *power*.

under observation. A subject is under observation when failure events, should they occur, would be observed (and so recorded in the dataset). Being under observation does not mean that a subject is necessarily at risk. Subjects usually come under observation before they are at risk. The statistical concern is with periods when subjects are at risk but not under observation, even when the subject is (later) known not to have failed during the hiatus.

In such cases, since failure events would not have been observed, the subject necessarily had to survive the observational hiatus, and that leads to bias in statistical results unless the hiatus is accounted for properly.

Entry time and exit time record when a subject first and last comes under observation, between which there may be observational gaps, but usually there are not. There is only one entry time and one exit time for each subject. Often, entry time corresponds to analysis time $t = 0$, or before, and exit time corresponds to the time of failure.

Delayed entry means that the entry time occurred after $t = 0$.

Subject and author index

See the combined subject index and the combined author index in the *Stata Index*.