Title

> **codebook —** Describe data contents

Description

codebook examines the variable names, labels, and data to produce a codebook describing the dataset.

Quick start

Codebook of all variables in the dataset
    codebook

Codebook of variables v1, v2, and v3
    codebook v1 v2 v3

Codebook of all variables starting with code
    codebook code*

Include dataset name, last saved date, and variable notes in the codebook
    codebook, header notes

Report problems with labels, constant-valued variables, embedded spaces and binary 0 in string variables, and noninteger date variables
    codebook, problems

Codebook for dataset with English and Spanish variable and value labels using label languages en and es
    codebook, languages(en es)

Menu

Data > Describe data > Describe data contents (codebook)

Syntax

> codebook [*varlist*] [*if*] [*in*] [, *options*]

options	Description
Options	
<u>all</u>	print complete report without missing values
<u>head</u>er	print dataset name and last saved date
<u>note</u>s	print any notes attached to variables
mv	report pattern of missing values
<u>t</u>abulate(*#*)	set tables/summary statistics threshold; default is tabulate(9)
<u>prob</u>lems	report potential problems in dataset
<u>d</u>etail	display detailed report on the variables; only with problems
<u>c</u>ompact	display compact report on the variables
dots	display a dot for each variable processed; only with compact
Languages	
<u>l</u>anguages[(*namelist*)]	use with multilingual datasets; see [D] **label language** for details

collect is allowed; see [U] **11.1.10 Prefix commands**.

Options

> Options

all is equivalent to specifying the header and notes options. It provides a complete report, which excludes only performing mv.

header adds to the top of the output a header that lists the dataset name, the date that the dataset was last saved, etc.

notes lists any notes attached to the variables; see [D] **notes**.

mv specifies that codebook search the data to determine the pattern of missing values. This is a CPU-intensive task.

tabulate(*#*) specifies the number of unique values of the variables to use to determine whether a variable is categorical or continuous. Missing values are not included in this count. The default is 9; when there are more than nine unique values, the variable is classified as continuous. Extended missing values will be included in the tabulation.

problems specifies that a summary report is produced describing potential problems that have been diagnosed:

- Variables that are labeled with an undefined value label
- Incompletely value-labeled variables
- Variables that are constant, including always missing
- Leading, trailing, and embedded spaces in string variables
- Embedded binary 0 (\0) in string variables
- Noninteger-valued date variables

See the discussion of these problems and advice on overcoming them following example 5.

detail may be specified only with the problems option. It specifies that the detailed report on the variables not be suppressed.

compact specifies that a compact report on the variables be displayed. compact may not be specified with any options other than dots.

dots specifies that a dot be displayed for every variable processed. dots may be specified only with compact.

***

Languages

languages [(*namelist*)] is for use with multilingual datasets; see [D] **label language**. It indicates that the codebook pertains to the languages in *namelist* or to all defined languages if no such list is specified as an argument to languages(). The output of codebook lists the data label and variable labels in these languages and which value labels are attached to variables in these languages.

Problems are diagnosed in all of these languages, as well. The problem report does not provide details in which language problems occur. We advise you to rerun codebook for problematic variables; specify detail to produce the problem report again.

If you have a multilingual dataset but do not specify languages(), all output, including the problem report, is shown in the "active" language.

Remarks and examples   [stata.com]

codebook, without arguments, is most usefully combined with log to produce a printed listing for enclosure in a notebook documenting the data; see [U] **15 Saving and printing output—log files**. codebook is, however, also useful interactively, because you can specify one or a few variables.

▷ Example 1

codebook examines the data in producing its results. For variables that codebook thinks are continuous, it presents the mean; the standard deviation; and the 10th, 25th, 50th, 75th, and 90th percentiles. For variables that it thinks are categorical, it presents a tabulation. In part, codebook makes this determination by counting the number of unique values of the variable. If the number is nine or fewer, codebook reports a tabulation; otherwise, it reports summary statistics.

codebook distinguishes the standard missing values (.) and the extended missing values (.a through .z, denoted by .*). If extended missing values are found, codebook reports the number of distinct missing value codes that occurred in that variable. Missing values are ignored with the tabulate option when determining whether a variable is treated as continuous or categorical.

```
. use https://www.stata-press.com/data/r18/educ3
(ccdb46, 52-54)

. codebook fips division, all
               Dataset:  https://www.stata-press.com/data/r18/educ3.dta
            Last saved:  6 Mar 2022 22:20

                 Label:  ccdb46, 52-54
   Number of variables:  42
Number of observations:  956
                  Size:  145,312 bytes ignoring labels, etc.

_dta:
  1.  confirmed data with steve on 7/22
```

---

fips				state/place code

```
                Type: Numeric (long)

               Range: [10060,560050]         Units: 1
       Unique values: 956                    Missing .: 0/956

                Mean: 256495
           Std. dev.: 156998

         Percentiles:    10%      25%      50%      75%      90%
                       61462   120426   252848   391360   482530
```

---

division				Census Division

```
                Type: Numeric (int)
               Label: division

               Range: [1,9]                  Units: 1
       Unique values: 9                      Missing .: 4/956
      Unique mv codes: 2                      Missing .*: 2/956

         Tabulation: Freq.   Numeric  Label
                        69         1  N. Eng.
                        97         2  Mid Atl
                       202         3  E.N.C.
                        78         4  W.N.C.
                       115         5  S. Atl.
                        46         6  E.S.C.
                        89         7  W.S.C.
                        59         8  Mountain
                       195         9  Pacific
                         4         .
                         2        .a
```

Because division has nine unique nonmissing values, codebook reported a tabulation. If division had contained one more unique nonmissing value, codebook would have switched to reporting summary statistics, unless we had included the tabulate(#) option.

◁

▷ Example 2

The `mv` option is useful. It instructs `codebook` to search the data to determine patterns of missing values. Different kinds of missing values are not distinguished in the patterns.

```
. use https://www.stata-press.com/data/r18/citytemp
(City temperature data)
. codebook cooldd heatdd tempjan tempjuly, mv
```

cooldd					Cooling degree days

```
            Type: Numeric (int)
           Range: [0,4389]                   Units: 1
   Unique values: 438                  Missing .: 3/956
            Mean: 1240.41
       Std. dev.: 937.668
     Percentiles:        10%       25%       50%       75%       90%
                         411       615       940      1566      2761
  Missing values:         heatdd==mv <-> cooldd==mv
                         tempjan==mv --> cooldd==mv
                        tempjuly==mv --> cooldd==mv
```

heatdd					Heating degree days

```
            Type: Numeric (int)
           Range: [0,10816]                  Units: 1
   Unique values: 471                  Missing .: 3/956
            Mean: 4425.53
       Std. dev.: 2199.6
     Percentiles:        10%       25%       50%       75%       90%
                        1510      2460      4950      6232      6919
  Missing values:         cooldd==mv <-> heatdd==mv
                         tempjan==mv --> heatdd==mv
                        tempjuly==mv --> heatdd==mv
```

tempjan					Average January temperature

```
            Type: Numeric (float)
           Range: [2.2,72.6]                 Units: .1
   Unique values: 310                  Missing .: 2/956
            Mean:  35.749
       Std. dev.: 14.1881
     Percentiles:        10%       25%       50%       75%       90%
                        20.2      25.1      31.3      47.8      55.1
  Missing values:       tempjuly==mv <-> tempjan==mv
```

```
─────────────────────────────────────────────────────────────────────
tempjuly                                              Average July temperature
─────────────────────────────────────────────────────────────────────

                  Type: Numeric (float)

                 Range: [58.1,93.6]                   Units: .1
       Unique values: 196                      Missing .: 2/956

                  Mean: 75.0538
            Std. dev.: 5.49504

           Percentiles:      10%       25%       50%       75%       90%
                            68.8      71.8     74.25      78.7      82.3

       Missing values:          tempjan==mv <-> tempjuly==mv
```

codebook reports that if tempjan is missing, tempjuly is also missing, and vice versa. In the output
for the cooldd variable, codebook also reports that the pattern of missing values is the same for
cooldd and heatdd. In both cases, the correspondence is indicated with "<->".

For cooldd, codebook also states that "tempjan==mv --> cooldd==mv". The one-way arrow
means that a missing tempjan value implies a missing cooldd value but that a missing cooldd
value does not necessarily imply a missing tempjan value.   ◁

Another feature of codebook—this one for numeric variables—is that it can determine the units
of the variable. For instance, in the example above, tempjan and tempjuly both have units of 0.1,
meaning that temperature is recorded to tenths of a degree. codebook handles precision considerations
in making this determination (tempjan and tempjuly are floats; see [U] **13.12 Precision and
problems therein**). If we had a variable in our dataset recorded in 100s (for example, 21,500 or
36,800), codebook would have reported the units as 100. If we had a variable that took on only
values divisible by 5 (5, 10, 15, etc.), codebook would have reported the units as 5.

▷ Example 3

We can use the label language command (see [D] **label language**) and the label command (see
[D] **label**) to create German value labels for our auto dataset. These labels are reported by codebook:

```
. use https://www.stata-press.com/data/r18/auto
(1978 automobile data)
. label language en, rename
(language default renamed en)
. label language de, new
(language de now current language)
. label data "1978 Automobile Daten"
. label variable foreign "Art Auto"
. label values foreign origin_de
. label define origin_de 0 "Innen" 1 "Auslndisch"
```

```
. codebook foreign
```

foreign	Art Auto

```
                Type: Numeric (byte)
               Label: origin_de
               Range: [0,1]                          Units: 1
       Unique values: 2                          Missing .: 0/74
           Tabulation: Freq.   Numeric  Label
                          52         0  Innen
                          22         1  Auslndisch
```

```
. codebook foreign, languages(en de)
```

foreign	in en: Car origin
	in de: Art Auto

```
                Type: Numeric (byte)
         Label in en: origin
         Label in de: origin_de
               Range: [0,1]                          Units: 1
       Unique values: 2                          Missing .: 0/74
           Tabulation: Freq. Numeric   origin        origin_de
                          52        0  Domestic      Innen
                          22        1  Foreign       Auslndisch
```

With the languages() option, the value labels are shown in the specified active and available
languages.

◁

▷ Example 4

codebook, compact summarizes the variables in your dataset, including variable labels. It is an
alternative to the summarize command.

```
. use https://www.stata-press.com/data/r18/auto, clear
(1978 automobile data)
```

```
. codebook, compact
```

Variable	Obs	Unique	Mean	Min	Max	Label
make	74	74	.	.	.	Make and model
price	74	74	6165.257	3291	15906	Price
mpg	74	21	21.2973	12	41	Mileage (mpg)
rep78	69	5	3.405797	1	5	Repair record 1978
headroom	74	8	2.993243	1.5	5	Headroom (in.)
trunk	74	18	13.75676	5	23	Trunk space (cu. ft.)
weight	74	64	3019.459	1760	4840	Weight (lbs.)
length	74	47	187.9324	142	233	Length (in.)
turn	74	18	39.64865	31	51	Turn circle (ft.)
displacement	74	31	197.2973	79	425	Displacement (cu. in.)
gear_ratio	74	36	3.014865	2.19	3.89	Gear ratio
foreign	74	2	.2972973	0	1	Car origin

```
. summarize
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
        make |          0
       price |         74    6165.257    2949.496       3291      15906
         mpg |         74     21.2973    5.785503         12         41
       rep78 |         69    3.405797    .9899323          1          5
    headroom |         74    2.993243    .8459948        1.5          5
-------------+---------------------------------------------------------
       trunk |         74    13.75676    4.277404          5         23
      weight |         74    3019.459    777.1936       1760       4840
      length |         74    187.9324    22.26634        142        233
        turn |         74    39.64865    4.399354         31         51
displacement |         74    197.2973    91.83722         79        425
-------------+---------------------------------------------------------
   gear_ratio |         74    3.014865    .4562871       2.19       3.89
      foreign |         74    .2972973    .4601885          0          1
```

◁

▷ Example 5

When codebook determines that neither a tabulation nor a listing of summary statistics is appropriate, for instance, for a string variable or for a numeric variable taking on many labeled values, it reports a few examples instead.

```
. use https://www.stata-press.com/data/r18/funnyvar
. codebook name
```

----
name   (unlabeled)

                  Type: String (str5), but longest is str3
         Unique values: 10   Missing "": 0/10
              Examples: "1 0"
                        "3"
                        "5"
                        "7"
             Warning: Variable has embedded blanks.

codebook is also on the lookout for common problems that might cause you to make errors when dealing with the data. For string variables, this includes leading, embedded, and trailing blanks and embedded binary 0 (\0). In the output above, codebook informed us that name includes embedded blanks. If name had leading or trailing blanks, it would have mentioned that, too.

When variables are value labeled, codebook performs two checks. First, if a value label *labname* is associated with a variable, codebook checks whether *labname* is actually defined. Second, it checks whether all values are value labeled. Partial labeling of a variable may mean that the label was defined incorrectly (for instance, the variable has values 0 and 1, but the value label maps 1 to "male" and 2 to "female") or that the variable was defined incorrectly (for example, a variable gender with three values). codebook checks whether date variables are integer valued.

If the problems option is specified, codebook does not provide detailed descriptions of each variable but reports only the potential problems in the data.

```
. codebook, problems
Potential problems in dataset https://www.stata-press.com/data/r18/
> funnyvar.dta
               Potential problem   Variables

  constant (or all missing) vars   human planet
     vars with nonexisting label   educ
        incompletely labeled vars  gender
str# vars that may be compressed   name address city country planet
 string vars with leading blanks   city country
string vars with trailing blanks   planet
string vars with embedded blanks   name address
    string vars with embedded \0   mugshot
    noninteger-valued date vars    birthdate
```

◁

In the example above, `codebook, problems` reported various potential problems with the dataset. These problems include

- Constant variables, including variables that are always missing

   Variables that are constant, taking the same value in all observations, or that are always missing, are often superfluous. Such variables, however, may also indicate problems. For instance, variables that are always missing may occur when importing data with an incorrect input specification. Such variables may also occur if you generate a new variable for a subset of the data, selected with an expression that is false for all observations.

   Advice: Carefully check the origin of constant variables. If you are saving a constant variable, be sure to compress the variable to use minimal storage.

- Variables with nonexisting value labels

   Stata treats value labels as separate objects that can be attached to one or more variables. A problem may arise if variables are linked to value labels that are not yet defined or if an incorrect value label name was used.

   Advice: Attach the correct value label, or label define the value label. See [D] **label**.

- Incompletely labeled variables

   A variable is called "incompletely value labeled" if the variable is value labeled but no mapping is provided for some values of the variable. An example is a variable with values 0, 1, and 2 and value labels for 1, 2, and 3. This situation usually indicates an error, either in the data or in the value label.

   Advice: Change either the data or the value label.

- String variables that may be compressed

   The storage space used by a string variable is determined by its data type; see [D] **Data types**. For instance, the storage type `str20` indicates that 20 bytes are used per observation. If the declared storage type exceeds your requirements, memory and disk space is wasted.

   Advice: Use compress to store the data as compactly as possible.

- String variables with leading or trailing blanks

    In most applications, leading and trailing spaces do not affect the meaning of variables but are probably side effects from importing the data or from data manipulation. Spurious leading and trailing spaces force Stata to use more memory than required. In addition, manipulating strings with leading and trailing spaces is harder.

    Advice: Remove leading and trailing blanks from a string variable s by typing

        replace s = strtrim(s)

    See [FN] **String functions**.

- String variables with embedded blanks

    String variables with embedded blanks are often appropriate; however, sometimes they indicate problems importing the data.

    Advice: Verify that blanks are meaningful in the variables.

- String variables with embedded binary 0 (\0)

    String variables with embedded binary 0 (\0) are allowed; however, caution should be used when working with them as some commands and functions may only work with the plain-text portion of a binary string, ignoring anything after the first binary 0.

    Advice: Be aware of binary strings in your data and whether you are manipulating them in a way that is only appropriate with plain-text values.

- Noninteger-valued date variables

    Stata's date and time formats were designed for use with integer values but will work with noninteger values.

    Advice: Carefully inspect the nature of the noninteger values. If noninteger values in a variable are the consequence of roundoff error, you may want to round the variable to the nearest integer.

        replace time = round(time)

Of course, more problems not reported by codebook are possible. These might include

- Numerical data stored as strings

    After importing data into Stata, you may discover that some string variables can actually be interpreted as numbers. Stata can do much more with numerical data than with string data. Moreover, string representation usually makes less efficient use of computer resources. destring will convert string variables to numeric.

    A string variable may contain a "field" with numeric information. An example is an address variable that contains the street name followed by the house number. The Stata string functions can extract the relevant substring.

- Categorical variables stored as strings

    Most statistical commands do not allow string variables. Moreover, string variables that take only a limited number of distinct values are an inefficient storage method. Use value-labeled numeric values instead. These are easily created with encode.

- Duplicate observations

    See [D] **duplicates**.

- Observations that are always missing

  Drop observations that are missing for all variables in *varlist* using the rownonmiss()
  egen function:

  egen nobs = rownonmiss(*varlist*)

  drop if nobs==0

  Specify _all for *varlist* if only observations that are always missing should be dropped.

Stored results

codebook stores the following lists of variables with potential problems in r():

Macros
r(cons)	constant (or missing)
r(labelnotfound)	undefined value labeled
r(notlabeled)	value labeled but with unlabeled categories
r(str_type)	compressible
r(str_leading)	leading blanks
r(str_trailing)	trailing blanks
r(str_embedded)	embedded blanks
r(str_embedded0)	embedded binary 0 (\0)
r(realdate)	noninteger dates

References

Bjärkefur, K., L. Cardoso de Andrade, and B. Daniels. 2020. iefieldkit: Commands for primary data collection and cleaning. *Stata Journal* 20: 892–915.

———. 2023. iefieldkit: Commands for primary data collection and cleaning (update). *Stata Journal* 23: 875–883.

Cox, N. J. 2012. Software Updates: Speaking Stata: Distinct observations. *Stata Journal* 12: 352.

Cox, N. J., and G. M. Longton. 2008. Speaking Stata: Distinct observations. *Stata Journal* 8: 557–568.

Long, J. S. 2009. *The Workflow of Data Analysis Using Stata*. College Station, TX: Stata Press.

Also see

[D] describe — Describe data in memory or in a file

[D] ds — Compactly list variables with specified properties

[D] inspect — Display simple summary of data's attributes

[D] labelbook — Label utilities

[D] notes — Place notes in data

[D] split — Split string variables into parts

[U] 15 Saving and printing output—log files

For suggested citations, see the FAQ on citing Stata documentation.