STATA CHOICE MODELS REFERENCE MANUAL

RELEASE 18

A Stata Press Publication
StataCorp LLC
College Station, Texas

Version 18

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845

ISBN-10: 1-59718-375-X
ISBN-13: 978-1-59718-375-8

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 Choice Models Reference Manual*. College Station, TX: Stata Press.

Contents

Intro . . . . . Introduction to choice models manual 1
Intro 1 . . . . . Interpretation of choice models 4
Intro 2 . . . . . Data layout 17
Intro 3 . . . . . Descriptive statistics 21
Intro 4 . . . . . Estimation commands 27
Intro 5 . . . . . Models for discrete choices 32
Intro 6 . . . . . Models for rank-ordered alternatives 56
Intro 7 . . . . . Models for panel data 65
Intro 8 . . . . . Random utility models, assumptions, and estimation 73

cmchoiceset . . . . . Tabulate choice sets 78

cmclogit . . . . . Conditional logit (McFadden's) choice model 87
cmclogit postestimation . . . . . Postestimation tools for cmclogit 103
cmmixlogit . . . . . Mixed logit choice model 109
cmmixlogit postestimation . . . . . Postestimation tools for cmmixlogit 128
cmmprobit . . . . . Multinomial probit choice model 132
cmmprobit postestimation . . . . . Postestimation tools for cmmprobit 160
cmrologit . . . . . Rank-ordered logit choice model 166
cmrologit postestimation . . . . . Postestimation tools for cmrologit 184
cmroprobit . . . . . Rank-ordered probit choice model 187
cmroprobit postestimation . . . . . Postestimation tools for cmroprobit 201

cmsample . . . . . Display reasons for sample exclusion 208
cmset . . . . . Declare data to be choice model data 217
cmsummarize . . . . . Summarize variables by chosen alternatives 226
cmtab . . . . . Tabulate chosen alternatives 232

cmxtmixlogit . . . . . Panel-data mixed logit choice model 242
cmxtmixlogit postestimation . . . . . Postestimation tools for cmxtmixlogit 260

margins . . . . . Adjusted predictions, predictive margins, and marginal effects 264

nlogit . . . . . Nested logit regression 289
nlogit postestimation . . . . . Postestimation tools for nlogit 313

Glossary 318

Subject and author index 319

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 choice models manual

Description Remarks and examples

Description

Choice models (CM) are models for data with outcomes that are choices. The choices are selected by a decision maker, such as a person or a business, from a set of possible alternatives. For instance, we could model choices made by consumers who select a breakfast cereal from several different brands. Or we could model choices made by businesses who chose whether to buy TV, radio, Internet, or newspaper advertising.

Models for choice data come in two varieties—models for discrete choices and models for rank-ordered alternatives. When each individual selects a single alternative, say, he or she purchases one box of cereal, the data are discrete choice data. When each individual ranks the choices, say, he or she orders cereals from most favorite to least favorite, the data are rank-ordered data. Stata has commands for fitting both discrete choice models and rank-ordered models.

This manual documents commands for working with and summarizing choice data, for fitting models, and for interpreting the results of those models.

Remarks and examples

The entries in this manual are organized as follows:

Introductions
Declaring and summarizing data
Fitting choice models
Postestimation
Glossary

Introductions

We recommend that you read the introductions first. In them, you will learn the language of choice models. We will show you how data for choice models are organized and how to explore these data using special summary statistic commands. You will learn about the models available for choice data and what makes each one unique. You will also learn about how to interpret results of these models. Choice models are known for being difficult to interpret, but Stata makes interpretation easy. So we start by telling you all about interpretation in the first introduction.

[CM] **Intro 1**	Interpretation of choice models
[CM] **Intro 2**	Data layout
[CM] **Intro 3**	Descriptive statistics
[CM] **Intro 4**	Estimation commands
[CM] **Intro 5**	Models for discrete choices
[CM] **Intro 6**	Models for rank-ordered alternatives
[CM] **Intro 7**	Models for panel data
[CM] **Intro 8**	Random utility models, assumptions, and estimation

Declaring and summarizing data

Before you fit a model with one of the cm commands, you will need to cmset your data.

[CM] **cmset**	Declare data to be choice model data

You can explore your data using the specialized commands for computing summary statistics for choice model data.

[CM] **cmchoiceset**	Tabulate choice sets
[CM] **cmsample**	Display reasons for sample exclusion
[CM] **cmsummarize**	Summarize variables by chosen alternatives
[CM] **cmtab**	Tabulate chosen alternatives

Fitting choice models

When you are ready to fit one of the choice models to your data, you can find information on syntax, additional examples, and methods and formulas in the entry for the command.

The following commands fit models for discrete choices:

[CM] **cmclogit**	Conditional logit (McFadden's) choice model
[CM] **cmmixlogit**	Mixed logit choice model
[CM] **cmmprobit**	Multinomial probit choice model
[CM] **nlogit**	Nested logit regression

The following commands fit models for rank-ordered alternatives:

[CM] **cmrologit**	Rank-ordered logit choice model
[CM] **cmroprobit**	Rank-ordered probit choice model

The following command fits models for discrete choices in panel data:

[CM] **cmxtmixlogit**	Panel-data mixed logit choice model

Postestimation

After you fit a model, postestimation commands allow you to estimate effects, obtain predictions, perform tests, and more. In particular, the margins command is invaluable in interpreting the results of choice models. The postestimation commands are documented in the following entries:

[CM] **margins**	Adjusted predictions, predictive margins, and marginal effects
[CM] **cmclogit postestimation**	Postestimation tools for cmclogit
[CM] **cmmixlogit postestimation**	Postestimation tools for cmmixlogit
[CM] **cmmprobit postestimation**	Postestimation tools for cmmprobit
[CM] **cmrologit postestimation**	Postestimation tools for cmrologit
[CM] **cmroprobit postestimation**	Postestimation tools for cmroprobit
[CM] **cmxtmixlogit postestimation**	Postestimation tools for cmxtmixlogit
[CM] **nlogit postestimation**	Postestimation tools for nlogit

Glossary

Finally, we provide a glossary that can be referred to as needed.

[CM] **Glossary** Glossary

Title

Intro 1 — Interpretation of choice models

Description Remarks and examples Also see

Description

Choice models have a reputation for being difficult to interpret. The coefficients that we estimate when we fit a choice model rarely allow a researcher to directly test hypotheses of interest. From the sign of the coefficient, we can determine the direction of an effect, and in the cases of conditional logit and mixed logit models, we can compute odds ratios and relative-risk ratios. Beyond this, the coefficients are almost uninterpretable.

There is no need for concern, however. Stata has a unique command, `margins`, that can be used after any of the `cm` choice model commands to easily answer questions of real scientific interest. In this entry, we introduce you to `margins`, and we give you a preview of the types of inferences that you can make when you use `margins` with the results of your choice models. To stay focused, we use `cmclogit` to fit all models in this introduction. That is not a limitation. The inferences we make and the associated `margins` commands would be the same for any of the `cm` estimation commands.

Remarks and examples

Remarks are presented under the following headings:

Interpretation of coefficients
Inferences from margins
    Expected choice probabilities
    Effects of a continuous covariate
    Effects of a categorical covariate
    Effects of an alternative-specific covariate
More inferences using margins

Interpretation of coefficients

We demonstrate with a transportation example. We have 210 individuals who choose a method of travel between two cities—by airplane, train, bus, or car. We also have information on the length of time each method will take, each individual's income, and the number of people traveling together.

```
. use https://www.stata-press.com/data/r18/travel
(Modes of travel)
. generate time = traveltime+termtime
```

Above, we generated a `time` variable representing total travel time. We use `cmset` to specify that the `id` variable identifies the travelers and the `mode` variable records the possible methods of travel, called alternatives. Then, we fit a conditional logistic regression model where `choice` indicates the selected travel method. Of our covariates, `time` is the only one that varies across the four travel alternatives. `income` and `partysize` are listed in the `casevars()` option because they have only one value per individual (or case) rather than a value for each alternative.

```
. cmset id mode
     Case ID variable: id
Alternatives variable: mode
```

```
. cmclogit choice time, casevars(income partysize)
Iteration 0:  Log likelihood = -249.36629
Iteration 1:  Log likelihood = -236.01608
Iteration 2:  Log likelihood = -235.65162
Iteration 3:  Log likelihood = -235.65065
Iteration 4:  Log likelihood = -235.65065
Conditional logit choice model                  Number of obs      =        840
Case ID variable: id                            Number of cases    =        210
Alternatives variable: mode                     Alts per case: min =          4
                                                               avg =        4.0
                                                               max =          4
                                                  Wald chi2(7)     =      71.14
Log likelihood = -235.65065                       Prob > chi2      =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
mode						
time	-.0041641	.0007588	-5.49	0.000	-.0056512	-.002677
Air	(base alternative)					
Train						
income	-.0613414	.0122637	-5.00	0.000	-.0853778	-.0373051
partysize	.4123606	.2406358	1.71	0.087	-.0592769	.883998
_cons	3.39349	.6579166	5.16	0.000	2.103997	4.682982
Bus						
income	-.0363345	.0134318	-2.71	0.007	-.0626605	-.0100086
partysize	-.1370778	.3437092	-0.40	0.690	-.8107354	.5365798
_cons	2.919314	.7658496	3.81	0.000	1.418276	4.420351
Car						
income	-.0096347	.0111377	-0.87	0.387	-.0314641	.0121947
partysize	.7350802	.2184636	3.36	0.001	.3068993	1.163261
_cons	.7471042	.6732971	1.11	0.267	-.5725338	2.066742

What can we determine from these results? The coefficient on `time` is negative, so the probability of choosing a method of travel decreases as the travel time increases. For the train alternative, the coefficient on `income` is negative. Because air travel is the base alternative, this negative coefficient tells us that as income increases, people are less likely to choose a train over an airplane. For the car alternative, the coefficient on `partysize` is positive. As party size increases, people are more likely to choose a car over an airplane.

Inferences from margins

The output from `cmclogit`, or from any other `cm` command, gives us a little information. However, it does not typically answer questions a researcher is truly interested in. In our example, for instance, we might want to know the following:

- What percentage of individuals do we expect to choose air travel?
- What is the effect of income? How does the probability of selecting car travel change as income increases from $30,000 to $40,000 or from $40,000 to $50,000? How does the probability of selecting train travel change?
- What if new security measures are implemented and wait times at the airport increase by 60 minutes? How does that affect the probability of choosing each method of travel?

We can answer each of these questions and many others using `margins`.

Expected choice probabilities

Let's start with the first question. We simply type `margins` without any options to obtain the average predicted probability of choosing each method of travel.

```
. margins
Predictive margins                                      Number of obs = 840
Model VCE: OIM
Expression: Pr(mode|1 selected), predict()
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome						
Air	.2761905	.0275268	10.03	0.000	.2222389	.330142
Train	.3	.0284836	10.53	0.000	.2441731	.3558269
Bus	.1428571	.0234186	6.10	0.000	.0969576	.1887567
Car	.2809524	.028043	10.02	0.000	.2259891	.3359156

Based on this model and our random sample of travelers between the two cities, we expect 28% of individuals to travel by air. We also expect 30% to travel by train, 14% to travel by bus, and 28% to travel by car.

Effects of a continuous covariate

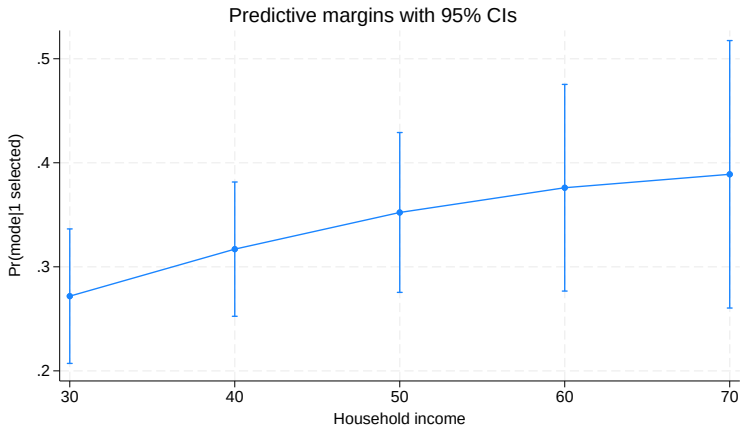
What does our model say would happen if values of a covariates change? We will first explore the effect of income. What would the expected probability of choosing car travel be if everyone made \$30,000? What if everyone made \$40,000? We can estimate expected probabilities of car travel for counterfactual income levels from \$30,000 to \$70,000 in \$10,000 increments by using the `at()` option with `margins`. We type

```
. margins, at(income=(30(10)70)) outcome(Car)
Predictive margins                                      Number of obs = 840
Model VCE: OIM
Expression: Pr(mode|1 selected), predict()
Outcome:    Car
1._at: income = 30
2._at: income = 40
3._at: income = 50
4._at: income = 60
5._at: income = 70
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_at						
1	.2717914	.0329811	8.24	0.000	.2071497	.3364331
2	.3169817	.0329227	9.63	0.000	.2524544	.3815091
3	.3522391	.0391994	8.99	0.000	.2754097	.4290684
4	.3760093	.050679	7.42	0.000	.2766802	.4753383
5	.3889296	.0655865	5.93	0.000	.2603825	.5174768

We can plot these probabilities to visualize the effect of income.

```
. marginsplot
Variables that uniquely identify margins: income
```



The expected probability of choosing car transportation increases as income increases. But are these differences statistically significant? We can test for differences in the expected probabilities for each \$10,000 increase in income. For this, we use the `contrast()` option and request reverse adjacent (`ar`) contrasts. We also simplify the output that reports a test of the differences by including the `nowald` and `effects` options.

```
. margins, at(income=(30(10)70)) outcome(Car)
> contrast(atcontrast(ar) nowald effects)
Contrasts of predictive margins                          Number of obs = 840
Model VCE: OIM

Expression: Pr(mode|1 selected), predict()
Outcome:    Car

1._at: income = 30

2._at: income = 40

3._at: income = 50

4._at: income = 60

5._at: income = 70
```

	Contrast	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_at						
(2 vs 1)	.0451903	.016664	2.71	0.007	.0125296	.0778511
(3 vs 2)	.0352574	.017903	1.97	0.049	.0001681	.0703466
(4 vs 3)	.0237702	.0190387	1.25	0.212	-.013545	.0610854
(5 vs 4)	.0129204	.0200549	0.64	0.519	-.0263866	.0522273

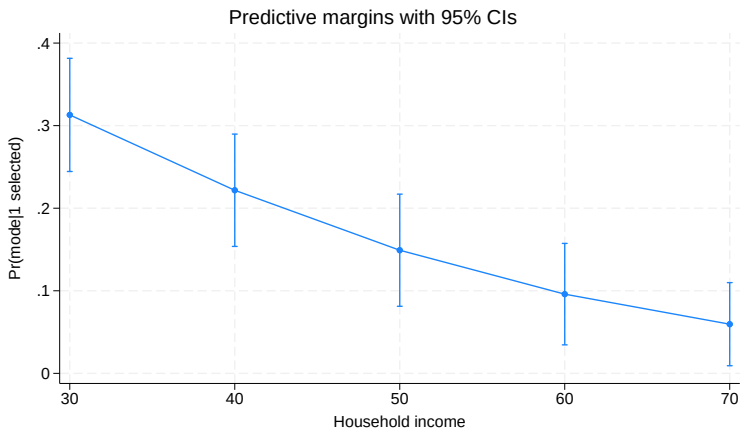
From the first line in this table, we see that the effect of having \$40,000 instead of \$30,000 is a 0.045 increase in the expected probability of selecting car travel. Having \$50,000 instead of \$40,000 increases the expected probability of car travel by 0.035. Both of these effects are significant at a 5% significance level, but increases in income from \$50,000 to \$60,000 and from \$60,000 to \$70,000 do not lead to significantly different probabilities of car travel.

We can also evaluate the effect of income on the probability of taking a train. We include the `outcome(Train)` option instead of `outcome(Car)` in our `margins` command.

```
. margins, at(income=(30(10)70)) outcome(Train)
Predictive margins                                      Number of obs = 840
Model VCE: OIM
Expression: Pr(mode|1 selected), predict()
Outcome:    Train
1._at: income = 30
2._at: income = 40
3._at: income = 50
4._at: income = 60
5._at: income = 70
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_at						
1	.3129898	.034968	8.95	0.000	.2444538	.3815257
2	.2217291	.0346969	6.39	0.000	.1537244	.2897338
3	.1491281	.0346442	4.30	0.000	.0812268	.2170294
4	.0959391	.0313489	3.06	0.002	.0344965	.1573818
5	.0595511	.0256786	2.32	0.020	.009222	.1098802

```
. marginsplot
Variables that uniquely identify margins: income
```



As income levels increase, the expected probability of choosing train transportation decreases.

We can again test for differences in the probabilities using reverse adjacent contrasts.

```
. margins, at(income=(30(10)70)) outcome(Train)
> contrast(atcontrast(ar) nowald effects)
Contrasts of predictive margins                          Number of obs = 840
Model VCE: OIM

Expression: Pr(mode|1 selected), predict()
Outcome:    Train

1._at: income = 30

2._at: income = 40

3._at: income = 50

4._at: income = 60

5._at: income = 70
```

	Contrast	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_at						
(2 vs 1)	-.0912606	.0174009	-5.24	0.000	-.1253659	-.0571554
(3 vs 2)	-.072601	.0104846	-6.92	0.000	-.0931505	-.0520516
(4 vs 3)	-.053189	.0066782	-7.96	0.000	-.0662779	-.0401
(5 vs 4)	-.036388	.0064958	-5.60	0.000	-.0491195	-.0236566

This time, we find a significant decrease in the expected probability of choosing train transportation for each $10,000 increase in income.

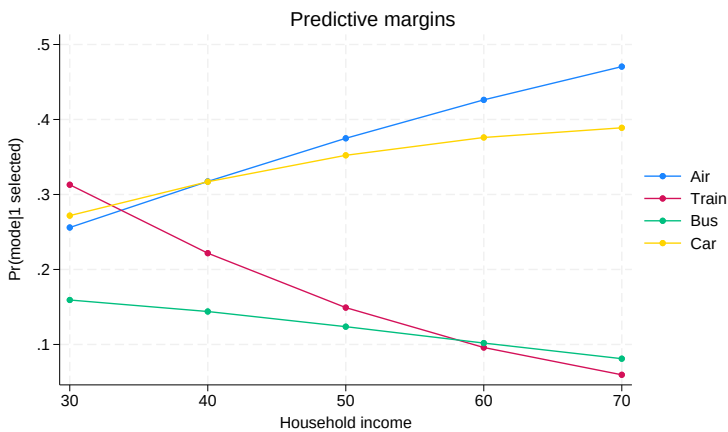
We do not need to limit our analysis to just car or train travel. How does the probability of each method of travel change with income?

We can type

```
. margins, at(income=(30(10)70))
```

This produces lots of output, so we do not show it here. We instead show you the graph of all the results. We omit the confidence intervals so that it is easy to see the probabilities for all four methods of travel.

```
. marginsplot, noci
Variables that uniquely identify margins: income _outcome
```



As income goes up, the expected probabilities of selecting bus and train transportation decrease, and the expected probabilities of choosing air and car transportation increase.

This graph allows us to visually compare the travel methods at each income level. We can formally test for differences in the expected probabilities of the travel methods. For instance, at the $30,000 income level, are all four methods of travel equally likely to be selected? We can use the `contrast(outcomejoint)` option to request this test.

```
. margins, at(income=30) contrast(outcomejoint)
Contrasts of predictive margins                               Number of obs = 840
Model VCE: OIM
Expression: Pr(mode|1 selected), predict()
At: income = 30
```

	df	chi2	P>chi2
_outcome	3	12.80	0.0051

We find that at least one of these expected probabilities is significantly different from the others.

We might want to ask a more specific question. Is there a difference in the expected probabilities of selecting train and bus travel when income is $50,000? We use the `outcome()` option to specify these two methods of travel and the `contrast(outcomecontrast(r))` option to request that `margins` estimate the difference between the two probabilities.

```
. margins, at(income=50) outcome(Bus Train)
> contrast(outcomecontrast(r) nowald effects)
Contrasts of predictive margins                               Number of obs = 840
Model VCE: OIM
Expression: Pr(mode|1 selected), predict()
At: income = 50
```

	Contrast	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome (Bus vs Train)	-.0254125	.0504996	-0.50	0.615	-.1243899	.0735648

We do not find a significant difference in the probabilities of selecting bus and train travel at this income level.

Effects of a categorical covariate

The questions we answered above were about a continuous variable, `income`. What if we want to evaluate the effect of a categorical variable instead? We include the variable as a factor variable in our model and again use `margins`. To demonstrate, we create a variable representing income quartiles and change our model to use it rather than the continuous income covariate.

```
. xtile income_cat = income, nquantiles(4)
. label define quartiles 1 "Quartile 1" 2 "Quartile 2"
>                        3 "Quartile 3" 4 "Quartile 4"
. label values income_cat quartiles
. cmclogit choice time, casevars(i.income_cat partysize)
  (output omitted)
```

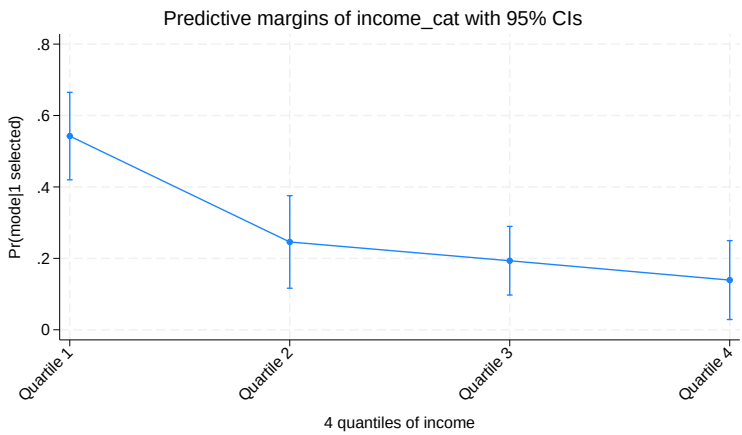
We estimate the expected probability of selecting train travel for each income quartile by typing

```
. margins income_cat, outcome(Train)
Predictive margins                                      Number of obs = 840
Model VCE: OIM

Expression: Pr(mode|1 selected), predict()
Outcome:    Train
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
income_cat						
Quartile 1	.5424554	.0624929	8.68	0.000	.4199715	.6649392
Quartile 2	.2459475	.0661098	3.72	0.000	.1163746	.3755203
Quartile 3	.1933788	.0490343	3.94	0.000	.0972733	.2894843
Quartile 4	.1391895	.0563599	2.47	0.014	.0287262	.2496529

We can again plot the results using `marginsplot`.

```
. marginsplot
Variables that uniquely identify margins: income_cat
```



We use the `ar.` operator to test for differences in the expected probabilities across adjacent income quartiles. This time, we attach the contrast operator to the name of the variable.

```
. margins ar.income_cat, outcome(Train) contrast(nowald effects)
Contrasts of predictive margins                               Number of obs = 840
Model VCE: OIM

Expression: Pr(mode|1 selected), predict()
Outcome:    Train
```

	Contrast	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
income_cat (Quartile 2 vs Quartile 1)	-.2965079	.0905087	-3.28	0.001	-.4739018	-.1191141
(Quartile 3 vs Quartile 2)	-.0525687	.082642	-0.64	0.525	-.214544	.1094067
(Quartile 4 vs Quartile 3)	-.0541893	.0744351	-0.73	0.467	-.2000794	.0917009

We find that the expected probability of choosing train travel is significantly different when moving from the first to the second income quartile.

Effects of an alternative-specific covariate

So far, we have explored the effects of the case-specific income variable. We can also ask questions about an alternative-specific variable such as travel time.

Perhaps new security measures are added. What would we expect if wait times at the airport increase by 60 minutes for all flights? We can use `margins`' `at()` option to evaluate this scenario.

```
. margins, at(time=generate(time+60)) alternative(Air)
Predictive margins                                            Number of obs = 840
Model VCE: OIM

Expression:  Pr(mode|1 selected), predict()
Alternative: Air
At: time = time+60
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome						
Air	.238248	.0261139	9.12	0.000	.1870657	.2894303
Train	.3138431	.0293829	10.68	0.000	.2562537	.3714325
Bus	.1513295	.0243434	6.22	0.000	.1036173	.1990417
Car	.2965794	.0289688	10.24	0.000	.2398016	.3533572

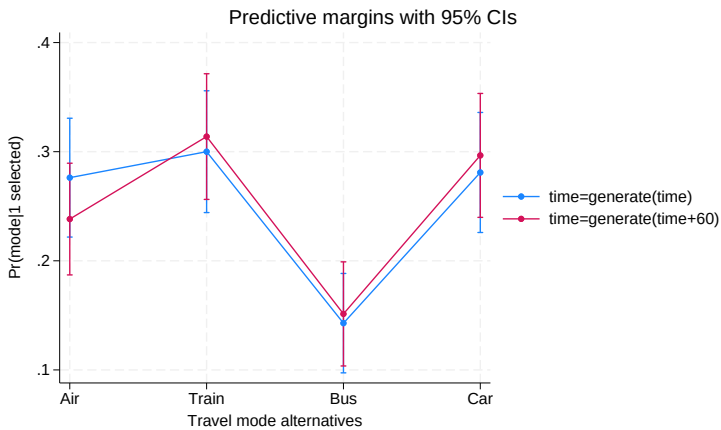
As we would anticipate, the expected probability of selecting air travel decreases when travel time increases. The probability of choosing air travel is now 0.24. If we look back at our first `margins` command, we can see that with the original travel times, the expected probability of choosing air travel was 0.28. Rather than looking at results from multiple commands, we can estimate probabilities with the original travel times and with the increased travel times all with a single `margins` command.

```
. margins, at(time=generate(time)) at(time=generate(time+60))
> alternative(Air)
Predictive margins                                      Number of obs = 840
Model VCE: OIM
Expression:  Pr(mode|1 selected), predict()
Alternative: Air
1._at: time =    time
2._at: time = time+60
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome#_at						
Air#1	.2761905	.0277787	9.94	0.000	.2217453	.3306357
Air#2	.238248	.0261139	9.12	0.000	.1870657	.2894303
Train#1	.3	.0284715	10.54	0.000	.2441968	.3558032
Train#2	.3138431	.0293829	10.68	0.000	.2562537	.3714325
Bus#1	.1428571	.0232375	6.15	0.000	.0973125	.1884018
Bus#2	.1513295	.0243434	6.22	0.000	.1036173	.1990417
Car#1	.2809524	.0280657	10.01	0.000	.2259446	.3359602
Car#2	.2965794	.0289688	10.24	0.000	.2398016	.3533572

Now it is easy to plot both scenarios together using `marginsplot`. This time we include the `xdimension()` option to place the four travel choices along the x axis.

```
. marginsplot, xdimension(_outcome)
Variables that uniquely identify margins: _atopt _outcome
Multiple at() options specified:
      _atoption=1: time=generate(time)
      _atoption=2: time=generate(time+60)
```



We see that the probability of air travel decreases, while the probability of choosing each of the other methods of travel increases a little.

We can again use contrasts to test for a difference. We use the `atcontrast(r)` to request comparisons to a reference level (the original travel times).

```
. margins, at(time=generate(time)) at(time=generate(time+60))
> alternative(Air) contrast(atcontrast(r) nowald effects)
Contrasts of predictive margins                               Number of obs = 840
Model VCE: OIM
Expression:  Pr(mode|1 selected), predict()
Alternative: Air
1._at: time =    time
2._at: time = time+60
```

	Contrast	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_at@_outcome						
(2 vs 1) Air	-.0379425	.0056615	-6.70	0.000	-.0490387	-.0268462
(2 vs 1) Train	.0138431	.0027216	5.09	0.000	.0085089	.0191773
(2 vs 1) Bus	.0084724	.001936	4.38	0.000	.004678	.0122668
(2 vs 1) Car	.015627	.0026002	6.01	0.000	.0105308	.0207232

Each of these differences is statistically significant. Of course, the researcher would need to decide whether an expected 3.8 percentage point decrease in air travel is meaningful.

We might take this even one step further. What if the speed of train transportation improves at the same time that speed of air travel declines? What if train travel is now 60 minutes faster while air travel is 60 minutes slower? We cannot specify all of these changes directly in the `at()` option, but we can create a new variable that represents this scenario.

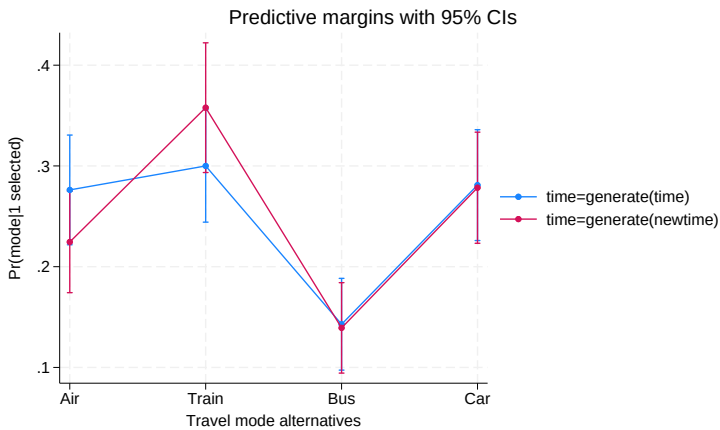
```
. generate newtime = time
. replace newtime = time+60 if mode==1
(210 real changes made)
. replace newtime = time-60 if mode==2
(210 real changes made)
```

Now we can specify our `newtime` variable in the `at()` option. We also include the `alternative(simultaneous)` option to specify that the changes to air and train travel be made simultaneously.

```
. margins, at(time=generate(time)) at(time=generate(newtime))
> alternative(simultaneous)
Predictive margins                                      Number of obs = 840
Model VCE: OIM
Expression: Pr(mode|1 selected), predict()
1._at: time =    time
2._at: time = newtime
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome#_at						
Air#1	.2761905	.0277787	9.94	0.000	.2217453	.3306357
Air#2	.224539	.0256826	8.74	0.000	.1742021	.274876
Train#1	.3	.0284715	10.54	0.000	.2441968	.3558032
Train#2	.3578287	.0328487	10.89	0.000	.2934464	.4222111
Bus#1	.1428571	.0232375	6.15	0.000	.0973125	.1884018
Bus#2	.1392549	.0228972	6.08	0.000	.0943773	.1841325
Car#1	.2809524	.0280657	10.01	0.000	.2259446	.3359602
Car#2	.2783773	.0281348	9.89	0.000	.2232342	.3335205

```
. marginsplot, xdimension(_outcome)
Variables that uniquely identify margins: _atopt _outcome
Multiple at() options specified:
      _atoption=1: time=generate(time)
      _atoption=2: time=generate(newtime)
```



Now it appears that the expected probability of selecting air travel decreases, the expected probability of selecting train travel increases, and the expected probabilities of selecting the other methods of transportation do not change much. Let's test for differences.

```
. margins, at(time=generate(time)) at(time=generate(newtime))
> alternative(simultaneous) contrast(atcontrast(r) nowald effects)
Contrasts of predictive margins                          Number of obs = 840
Model VCE: OIM
Expression: Pr(mode|1 selected), predict()
1._at: time =    time
2._at: time = newtime
```

	Contrast	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_at@_outcome						
(2 vs 1) Air	-.0516514	.0080654	-6.40	0.000	-.0674594	-.0358435
(2 vs 1) Train	.0578287	.0109404	5.29	0.000	.0363859	.0792716
(2 vs 1) Bus	-.0036023	.0022959	-1.57	0.117	-.0081021	.0008976
(2 vs 1) Car	-.002575	.0036587	-0.70	0.482	-.009746	.0045959

Now the expected probability of selecting air travel is 5.2 percentage points lower than it was with the original travel times, and the expected probability of selecting train travel is 5.7 percentage points higher.

More inferences using margins

Here we have demonstrated how you can use the results of your choice model to answer some interesting questions. But this is just a small sample of the types of inference that you can do using `margins` after a choice model estimator. For more examples, see [CM] **margins**, [CM] **Intro 5**, [CM] **Intro 6**, [CM] **cmclogit**, [CM] **cmmixlogit**, and [CM] **cmxtmixlogit**. Regardless of the `cm` command you use to fit your model, you may be interested in all of these examples because the same `margins` commands can be used after any of the models to estimate the same types of margins and test the same types of hypotheses.

Also see

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

Title

Intro 2 — Data layout

Description Remarks and examples Also see

Description

This introduction describes the data layout required by all cm commands and describes how to cmset your data.

Remarks and examples

Remarks are presented under the following headings:

Data layout for choice models
cmset: Cross-sectional data
cmset: Panel data

Data layout for choice models

cm commands require data in a different form than the usual Stata data format. Typically in Stata, a single Stata observation corresponds to a single statistical observation—that is why Stata calls rows in a Stata dataset "observations". But cm commands need multiple Stata observations to hold the data for a single statistical observation. So as not to confuse statistical observations with Stata observations, we call a single statistical observation a "case" and use this terminology throughout the CM manual.

Here is an example of choice data. We show data for the first three individuals.

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. list consumerid car purchase gender income dealers if consumerid <= 3,
> sepby(consumerid) abbrev(10)
```

	consumerid	car	purchase	gender	income	dealers
1.	1	American	1	Male	46.7	9
2.	1	Japanese	0	Male	46.7	11
3.	1	European	0	Male	46.7	5
4.	1	Korean	0	Male	46.7	1
5.	2	American	1	Male	26.1	10
6.	2	Japanese	0	Male	26.1	7
7.	2	European	0	Male	26.1	2
8.	2	Korean	0	Male	26.1	1
9.	3	American	0	Male	32.7	8
10.	3	Japanese	1	Male	32.7	6
11.	3	European	0	Male	32.7	2

These fictitious data represent persons who purchased a car with their choices categorized by the nationality of the manufacturer, American, Japanese, European, or Korean. The first variable is `consumerid`, a variable identifying individual consumers; it is called the case ID variable.

The second variable shown is `car`, which holds the possible choices available to the consumer. The possible choices are called "alternatives", and this variable is referred to as the alternatives variable. We see that the first two consumers had all four nationalities of cars as alternatives. The third had only American, Japanese, and European as alternatives because there were no Korean dealerships in his or her community.

The third variable `purchase` is a 0/1 variable indicating which car the person purchased. For discrete choice models (`cmclogit`, `cmmprobit`, `cmmixlogit`, and `cmxtmixlogit`), this variable is the dependent variable in the estimation.

The variables `gender` and `income` are case-specific variables; they are constant within case. The variable `dealers` contains the number of dealerships of each nationality that are located in the consumer's community. It varies both by alternative and by individual. It is an alternative-specific variable. The case-specific variables and the alternative-specific variables will be used as independent variables in the estimation. It is important to distinguish between them because they are handled differently by the estimation commands and grouped separately when you run the command. See [CM] **Intro 5** for examples.

If you are familiar with Stata, you know that this data arrangement is called "long data". There are multiple Stata observations for each distinct value of the case ID variable. Wide data would be just one Stata observation for each case. All `cm` commands require data be in the long form.

The long-data format has implications for how missing values are handled by the `cm` commands. By default, any missing value within any of the observations for a case causes the entire case to be dropped from the analysis. The option `altwise` (meaning alternativewise), which all `cm` commands allow, causes only the observations with missing values to be dropped. See [CM] **cmsample** for a longer discussion about missing values. See example 3 in [CM] **cmclogit** for an estimation example.

cmset: Cross-sectional data

If a command begins with `cm`, you must `cmset` your data before you can run the command.

For cross-sectional data with identified alternatives, we pass the case ID variable and alternatives variable as arguments:

```
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of different
      sizes found.

     Case ID variable: consumerid
Alternatives variable: car
```

The command echoed back the variable names that we set and, in this instance, also displayed a message, "alternatives are unbalanced across choice sets; choice sets of different sizes found." A "choice set" is the set of available alternatives for a case. This message is merely saying that the number of alternatives per case differs across the cases.

To see a tabulation of the choice sets, we type `cmchoiceset`:

```
. cmchoiceset
Tabulation of choice-set possibilities
 Choice set |      Freq.     Percent        Cum.
------------+-----------------------------------
      1 2 3 |        380       42.94       42.94
    1 2 3 4 |        505       57.06      100.00
------------+-----------------------------------
      Total |        885      100.00
Note: Total is number of cases.
. label list nation
nation:
           1 American
           2 Japanese
           3 European
           4 Korean
```

The output shows there are two choice sets, $\{1,2,3\}$ and $\{1,2,3,4\}$. We also listed the value label `nation`, which is the value label for the alternatives variable `car`, to see the correspondence between the numerical values and the nationalities. The two choice sets are all four nationalities and all nationalities except Korean. See [CM] **cmchoiceset** for more ways to use this command.

For some CM estimators, such as `cmmixlogit`, having explicitly identified alternatives is optional. For the model fit by the `cmrologit` estimator, the alternatives are not identified, so there is no alternatives variable. When there is no alternatives variable, we `cmset` our data using the option `noalternatives` and pass the case ID variable as an argument:

```
. cmset consumerid, noalternatives
note: alternatives are unbalanced across choice sets; choice sets of different
      sizes found.
     Case ID variable: consumerid
Alternatives variable: <none>
```

cmset: Panel data

Here is an example of panel choice data:

```
. use https://www.stata-press.com/data/r18/transport, clear
(Transportation choice data)
. list id t alt if id == 1, sepby(t)
```

	id	t	alt
1.	1	1	Car
2.	1	1	Public
3.	1	1	Bicycle
4.	1	1	Walk
5.	1	2	Car
6.	1	2	Public
7.	1	2	Bicycle
8.	1	2	Walk
9.	1	3	Car
10.	1	3	Public
11.	1	3	Bicycle
12.	1	3	Walk

The first variable, id, is an ID for individuals, and the second variable, t, is the time. The set of data for an individual makes up a "panel", so the individual ID is the panel ID.

For panel choice data, cmset takes three variables—when there is an alternatives variable, as there is in this example. The first variable identifies the panels, the second gives the time, and the third is the alternatives variable.

```
. cmset id t alt
note: case identifier _caseid generated from id and t.
note: panel by alternatives identifier _panelaltid generated from id and alt.

                     Panel data: Panels id and time t
               Case ID variable: _caseid
          Alternatives variable: alt
Panel by alternatives variable: _panelaltid (strongly balanced)
                  Time variable: t, 1 to 3
                          Delta: 1 unit

Note: Data have been xtset.
```

cmset has created two new variables: _caseid and _panelaltid. See example 2 in [CM] **cmset** for details about these variables. You do not need to concern yourself with them, however. Just leave them in your dataset, and the cm commands will use them automatically to make things work.

Also see

[CM] **Intro 3** — Descriptive statistics

[CM] **cmchoiceset** — Tabulate choice sets

[CM] **cmsample** — Display reasons for sample exclusion

[CM] **cmset** — Declare data to be choice model data

Title

Intro 3 — Descriptive statistics

Description Remarks and examples Also see

Description

In this entry, we introduce you to four helper commands that let you quickly see some basic attributes of your CM data: `cmchoiceset`, `cmsample`, `cmtab`, and `cmsummarize`.

Remarks and examples

Remarks are presented under the following headings:

cmchoiceset: Tabulating choice sets
cmsample: Looking at problem observations
cmtab: Tabulating chosen alternatives versus other variables
cmsummarize: Descriptive statistics for CM variables

cmchoiceset: Tabulating choice sets

Let's again use the data that we used in [CM] **Intro 2**.

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. list consumerid car purchase gender income if consumerid <= 3,
> sepby(consumerid) abbrev(10)
```

	consumerid	car	purchase	gender	income
1.	1	American	1	Male	46.7
2.	1	Japanese	0	Male	46.7
3.	1	European	0	Male	46.7
4.	1	Korean	0	Male	46.7
5.	2	American	1	Male	26.1
6.	2	Japanese	0	Male	26.1
7.	2	European	0	Male	26.1
8.	2	Korean	0	Male	26.1
9.	3	American	0	Male	32.7
10.	3	Japanese	1	Male	32.7
11.	3	European	0	Male	32.7

The case ID variable is `consumerid`. The alternatives variable is `car`. The 0/1 variable `purchase` indicates the nationality of car purchased. The variables `gender` and `income` are case-specific variables.

We cmset our data:

```
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of different
      sizes found.
     Case ID variable: consumerid
Alternatives variable: car
```

We use cmchoiceset to see the choice sets:

```
. cmchoiceset
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2 3	380	42.94	42.94
1 2 3 4	505	57.06	100.00
Total	885	100.00	

```
Note: Total is number of cases.
. label list nation
nation:
           1 American
           2 Japanese
           3 European
           4 Korean
```

There are two choice sets, $\{1,2,3\}$ and $\{1,2,3,4\}$. The value label `nation`, which labels the alternatives variable `car`, shows the correspondence between the numerical values and the nationalities. One choice set includes all four nationalities, and the other includes all nationalities except Korean.

`cmchoiceset` can be used after a `cm` estimation command to see the choice sets in the estimation sample. Here we fit a model using cmclogit and then run `cmchoiceset` restricted to the estimation sample.

```
. cmclogit purchase dealers, casevars(i.gender income)
  (output omitted)
. cmchoiceset if e(sample)
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2 3	373	43.27	43.27
1 2 3 4	489	56.73	100.00
Total	862	100.00	

```
Note: Total is number of cases.
```

We see that the estimation sample had 862 cases, whereas the earlier `cmchoiceset` output showed that the full sample had 885 cases.

By default, missing values are handled casewise, meaning that any missing value in any observation composing the case causes the entire case to be omitted from the estimation sample. In this example, $885 - 862 = 23$ cases contained missing values.

If you want to omit only observations with missing values and not the entire case, specify the option `altwise`. We refit the model using the `altwise` option and look at the choice sets.

```
. cmclogit purchase dealers, casevars(i.gender income) altwise
  (output omitted)
. cmchoiceset if e(sample)
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2	2	0.23	0.23
1 2 3	378	42.71	42.94
1 2 3 4	489	55.25	98.19
1 2 4	4	0.45	98.64
1 3	2	0.23	98.87
1 3 4	2	0.23	99.10
2 3	3	0.34	99.44
2 3 4	5	0.56	100.00
Total	885	100.00	

```
Note: Total is number of cases.
```

Handling the missing values alternativewise gives six new choice sets, albeit each with low frequency.

Handling missing values casewise never creates new choice sets. Handling missing values with `altwise` almost always changes the choice sets used in the estimation. You should be aware of the consequences. For instance, a dataset with balanced choice sets will typically become unbalanced when missing values are handled alternativewise. See example 3 in [CM] **cmclogit** for more details.

`cmchoiceset` also creates two-way (and three-way) tabulations. You can tabulate a variable, typically a case-specific one, against choice sets to see whether there is any association between the variable and choice sets. If you have panel data, you can tabulate the choice sets versus time to see whether choice sets change over time. See [CM] **cmchoiceset**.

`cmchoiceset` has a `generate(`*newvar*`)` option, which creates a variable with categories of the choice sets. This variable can be used in the `over()` option of `margins` to compute predicted probabilities and marginal effects separately for each choice set. See example 3 in [CM] **cmchoiceset** for an example.

cmsample: Looking at problem observations

Let's load and try to `cmset` a dataset to which we added some errors.

```
. use https://www.stata-press.com/data/r18/carchoice_errors, clear
(Car choice data with errors)
. cmset consumerid car
at least one choice set has more than one instance of the same alternative
r(459);
```

We get an error and our data are not `cmset`. We need to fix the repeated alternatives in `car`, the alternatives variable. The `cmsample` command can locate these problem observations. But to run `cmsample`, the data must be `cmset`. To do this, we use `cmset` with the `force` option. (Note: `cmsample` is the only command that works after suppressing an error using `cmset, force`. All other `cm` commands will give the same error about repeated alternatives unless the problematic observations are dropped or excluded using an `if` restriction.)

```
. cmset consumerid car, force
note: at least one choice set has more than one instance of the same
      alternative.
     Case ID variable: consumerid
Alternatives variable: car
```

Now we can run `cmsample`. We specify the option `generate(flag)` to create a variable named `flag` that identifies the problem observations.

```
. cmsample, generate(flag)
```

Reason for exclusion	Freq.	Percent	Cum.
observations included	3,153	99.78	99.78
repeated alternatives within case*	7	0.22	100.00
Total	3,160	100.00	

```
* indicates an error
```

`cmsample` produced a table that showed there are seven observations that contain the cases with the repeated alternatives. We can see the problems by listing the observations with `flag != 0`:

```
. list consumerid car flag if flag != 0, sepby(consumerid) abbr(10)
```

	consumerid	car	flag
397.	111	American	repeated alternatives within case*
398.	111	Japanese	repeated alternatives within case*
399.	111	Japanese	repeated alternatives within case*
1035.	290	American	repeated alternatives within case*
1036.	290	Japanese	repeated alternatives within case*
1037.	290	Japanese	repeated alternatives within case*
1038.	290	Korean	repeated alternatives within case*

We will need to fix or drop these cases before we can run other CM commands.

`cmsample` can identify many different problems in your choice data—16 different problems in all! To see its full capabilities, see [CM] **cmsample**.

cmtab: Tabulating chosen alternatives versus other variables

Let's reload our earlier dataset so we are not dealing with a dataset with `cm` errors.

```
. use https://www.stata-press.com/data/r18/carchoice, clear
(Car choice data)
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of different
      sizes found.
     Case ID variable: consumerid
Alternatives variable: car
```

The `cmtab` command requires the `choice(`*varname*`)` option, where *varname* is a 0/1 variable indicating which alternative was chosen. Typically, it is the dependent variable used in a discrete choice model. Typing `cmtab` without any other arguments gives a tabulation of the chosen alternatives:

```
. cmtab, choice(purchase)
Tabulation of chosen alternatives (purchase = 1)
```

Nationality of car	Freq.	Percent	Cum.
American	384	43.39	43.39
Japanese	326	36.84	80.23
European	135	15.25	95.48
Korean	40	4.52	100.00
Total	885	100.00	

Typing `cmtab` with a variable gives a tabulation of that variable versus the chosen alternatives.

```
. cmtab gender, choice(purchase) column
Tabulation for chosen alternatives (purchase = 1)
    gender is constant within case
```

Key
frequency *column percentage*

Nationality of car	Gender: 0 = Female, 1 = Male Female	Male	Total
American	96 40.68	280 44.73	376 43.62
Japanese	110 46.61	206 32.91	316 36.66
European	22 9.32	108 17.25	130 15.08
Korean	8 3.39	32 5.11	40 4.64
Total	236 100.00	626 100.00	862 100.00

We see that in these data, the most popular nationality of car among females was Japanese, with 47% of them purchasing a Japanese car. Among males, American cars were the most popular, with 45% of them buying an American car.

See [CM] **cmtab** for the full capabilities of the command.

cmsummarize: Descriptive statistics for CM variables

The `cmsummarize` command produces descriptive statistics for CM variables. For each variable in the command's *varlist*, it selects observations that correspond to chosen alternatives and displays statistics categorized by the chosen alternatives. The chosen alternatives are specified by the `choice(`*varname*`)` option, which is required, just as it is with `cmtab`.

Here is an example where we display the quartiles of the case-specific variable `income`:

```
. cmsummarize income, choice(purchase) stats(p25 p50 p75) format(%5.1f)
Statistics by chosen alternatives (purchase = 1)
    income is constant within case
Summary for variables: income
Group variable: _chosen_alternative (purchase = 1)
```

_chosen_alternative	p25	p50	p75
American	30.6	42.0	46.6
Japanese	39.0	44.4	48.8
European	40.5	44.6	49.2
Korean	25.4	35.5	44.2
Total	33.0	43.3	46.7

We see that buyers of European cars have the greatest median income and buyers of Korean cars the least compared with buyers of cars of other nationalities.

See [CM] **cmsummarize** for the full capabilities of the command.

Also see

[CM] **Intro 2** — Data layout

[CM] **cmchoiceset** — Tabulate choice sets

[CM] **cmsample** — Display reasons for sample exclusion

[CM] **cmset** — Declare data to be choice model data

[CM] **cmsummarize** — Summarize variables by chosen alternatives

[CM] **cmtab** — Tabulate chosen alternatives

Title

Intro 4 — Estimation commands

Description Remarks and examples

Description

Seven commands for fitting choice models (CM) are documented in this manual. These commands are used almost exclusively with choice data. Many other commands can also be useful for modeling choice data but are often used with other types of data as well. In this entry, we give you an overview of estimation commands available in Stata for modeling choice data.

Remarks and examples

Remarks are presented under the following headings:

Specialized choice model commands
Other commands for choice models
Models for cross-sectional data
Models for panel data
Multilevel models for clustered data

Specialized choice model commands

The following commands are documented in this manual and are designed specifically for fitting choice models:

Estimators for discrete choices

`cmclogit`	Conditional logit (McFadden's) choice model
`cmmixlogit`	Mixed logit choice model
`cmmprobit`	Multinomial probit choice model
`nlogit`	Nested logit regression

See [CM] **Intro 5** for details on these estimators.

Estimators for rank-ordered choices

`cmrologit`	Rank-ordered logit choice model
`cmroprobit`	Rank-ordered probit choice model

See [CM] **Intro 6** for details on these estimators.

Estimator for discrete choices with panel data

`cmxtmixlogit`	Panel-data mixed logit choice model

See [CM] **Intro 7** for details on this estimator.

What is special about the seven estimators listed here is that they all require your data to be in long form. That is, each case consists of multiple Stata observations, one for each of its available alternatives. All of these estimators allow you to include alternative-specific variables as covariates in your model. In addition, each of these estimators handles unbalanced choice sets. In [CM] **Intro 5**, [CM] **Intro 6**, and [CM] **Intro 7**, we provide more in-depth introductions to these estimators for discrete choices, rank-ordered alternatives, and discrete choices in panel data, respectively.

Other commands for choice models

Many other commands in Stata can be used for choice modeling. When you use these general commands with choice data, it is important to consider the restrictions or limitations of the model to make sure that it is the best command for modeling your choice data. For instance, the mlogit command fits multinomial logit models. When you use multinomial logit to fit a choice model, you are required to have only case-specific variables as predictors. Multinomial logit also requires balanced choice sets (that is, every decision maker must have the same available alternatives). Another example is the clogit command. You can use it to fit the same McFadden's choice model fit by cmclogit. In fact, cmclogit calls clogit to produce its estimates. However, because cmclogit is specifically designed for choice models, it gives you appropriate handling of missing values for choice data, and the postestimation command margins gives you options for unbalanced data after cmclogit. Nonetheless, when your choice data meet the requirements for one of Stata's commands for binary or categorical outcomes, they are useful for choice modeling.

Models for cross-sectional data

Stata has many commands for fitting binary and categorical outcome models that can be applied to some types of choice data. When a decision maker chooses from only two possible alternatives, the commands for binary outcomes may be useful. When a decision maker chooses from more than two outcomes, the commands for categorical outcomes may be appropriate. In addition to commands for common models such as logistic and probit, you can select from commands that address problems including heteroskedasticity, endogenous covariates, and sample selection. You can use the fmm: prefix to fit mixtures of choice models. If you want to simultaneously model more than one outcome variable or if you want to include latent variables, you can use gsem to fit a generalized structural equation model that allows binary and categorical outcomes. You can also use Bayesian estimation. The bayes: prefix allows you to fit Bayesian regression models. bayesmh is a flexible command that allows you to specify your own Bayesian model.

The following commands can fit choice models for cross-sectional data:

Estimators for binary choices

cloglog	Complementary log–log regression
logistic	Logistic regression, reporting odds ratios
logit	Logistic regression, reporting coefficients
probit	Probit regression

Exact statistics

exlogistic	Exact logistic regression

With endogenous sample selection

heckprobit	Probit model with sample selection

With heteroskedasticity

hetprobit	Heteroskedastic probit model

With endogenous covariates

ivprobit	Probit model with continuous endogenous covariates

With endogenous covariates and sample selection

eprobit	Extended probit regression

Finite mixture models

`fmm: cloglog`	Finite mixtures of complementary log–log regression models
`fmm: logit`	Finite mixtures of logistic regression models
`fmm: probit`	Finite mixtures of probit regression models

Multiple outcome variables and latent variables

`gsem`	Generalized structural equation model estimation command

Bayesian estimation

`bayes: cloglog`	Bayesian complementary log–log regression
`bayes: logistic`	Bayesian logistic regression, reporting odds ratios
`bayes: logit`	Bayesian logistic regression, reporting coefficients
`bayes: probit`	Bayesian probit regression
`bayes: hetprobit`	Bayesian heteroskedastic probit regression
`bayes: heckprobit`	Bayesian probit model with sample selection
`bayesmh`	Bayesian models using Metropolis–Hastings algorithm+

Estimators for categorical outcomes

`clogit`	Conditional (fixed-effects) logistic regression
`mlogit`	Multinomial (polytomous) logistic regression
`mprobit`	Multinomial probit regression

Finite mixture models

`fmm: mlogit`	Finite mixtures of multinomial (polytomous) logistic regression models

Systems of equations and latent variables

`gsem`	Generalized structural equation model estimation command

Bayesian estimation

`bayes: clogit`	Bayesian conditional logistic regression
`bayes: mlogit`	Bayesian multinomial logistic regression
`bayes: mprobit`	Bayesian multinomial probit regression
`bayesmh`	Bayesian models using Metropolis–Hastings algorithm+

Models for panel data

If you are working with panel data, you may be interested in standard panel-data commands for binary outcomes. For categorical outcomes, the `gsem` command can fit a multinomial logit model with random effects in addition to accommodating multiple outcome variables and latent variables for both binary and categorical outcomes. Bayesian estimation is also available. The `bayes:` prefix provides support for some of the `xt` commands, and it can be used with `me` commands to fit random-effects models. The following commands can be useful for choice models with panel data:

Estimators for binary choices

`xtcloglog`	Random-effects and population-averaged cloglog models
`xtlogit`	Fixed-effects, random-effects, and population-averaged logit models
`xtprobit`	Random-effects and population-averaged probit models

With endogenous covariates and sample selection

`xteprobit`	Extended random-effects probit regression

Multiple outcome variables and latent variables

`gsem`	Generalized structural equation models

Bayesian estimation

`bayes: mecloglog`	Bayesian multilevel complementary log–log regression
`bayes: xtlogit`	Bayesian random-effects logit model
`bayes: xtprobit`	Bayesian random-effects probit model
`bayesmh`	Bayesian models using Metropolis–Hastings algorithm+

Estimators for categorical choices

Multinomial logistic regression

`xtmlogit`	Fixed-effects and random-effects multinomial logit models

Multiple outcome variables and latent variables

`gsem`	Generalized structural equation model estimation command

Bayesian estimation

`bayes: xtmlogit`	Bayesian random-effects multinomial logit model
`bayesmh`	Bayesian models using Metropolis–Hastings algorithm+

In addition to the commands listed here, commands listed in the previous section that fit models for cross-sectional data can be used with panel data provided that they allow the `vce(cluster)` option. The point estimates from these commands have a population-averaged interpretation and are consistent but less efficient than the estimates from an appropriate panel-data estimator. The default standard errors reported by commands for cross-sectional data are inappropriate for panel or otherwise clustered data because they assume that observations are independent. However, by including the `vce(cluster)` option, you will get standard errors that relax this assumption and provide valid inference for this type of data.

Multilevel models for clustered data

You can also use multilevel modeling commands for choice models when your data are clustered or grouped and observations within the clusters are not independent. Perhaps your observations are students, and those students come from multiple classrooms. You might even have students within classrooms and classrooms within schools. The `me` commands fit multilevel models that account for the correlation within clusters. For categorical outcomes, the `gsem` command can fit a multilevel multinomial logit model in addition to accommodating multiple outcome variables and latent variables for both binary and categorical outcomes. Bayesian estimation is also available.

The following commands can be useful for multilevel choice models for clustered data:

Estimators for binary choices

`mecloglog`	Multilevel mixed-effects complementary log–log regression
`melogit`	Multilevel mixed-effects logistic regression
`meprobit`	Multilevel mixed-effects probit regression

Multiple outcome variables and latent variables

`gsem`	Generalized structural equation model estimation command

Bayesian estimation

`bayes: mecloglog`	Bayesian multilevel complementary log–log regression
`bayes: melogit`	Bayesian multilevel logistic regression
`bayes: meprobit`	Bayesian multilevel probit regression
`bayesmh`	Bayesian models using Metropolis–Hastings algorithm[+]

Estimators for categorical choices

Multilevel multinomial logistic regression

`gsem`	Generalized structural equation model estimation command

Multiple outcome variables and latent variables

`gsem`	Generalized structural equation model estimation command

Bayesian estimation

`bayesmh`	Bayesian models using Metropolis–Hastings algorithm[+]

Title

Intro 5 — Models for discrete choices

Description Remarks and examples References Also see

Description

This introduction covers the commands `cmclogit`, `cmmixlogit`, `cmmprobit`, and `nlogit`. These estimation commands fit discrete choice models, that is, models in which each decision maker chooses a single alternative from a finite set of available alternatives.

Remarks and examples

Remarks are presented under the following headings:

Overview of CM commands for discrete choices
cmclogit: McFadden's choice model
Looking at cases with missing values using cmsample
margins after CM estimation
cmmixlogit: Mixed logit choice models
cmmprobit: Multinomial probit choice models
nlogit: Nested logit choice models
Relationships with other estimation commands
    Duplicating cmclogit using clogit
    Multinomial logistic regression and McFadden's choice model
Estimation considerations
    Setting the number of integration points
    Convergence
    More than one chosen alternative

Overview of CM commands for discrete choices

Stata has four commands designed for fitting discrete choice models. Here we give you a brief overview of the similarities and differences in the models fit by these commands.

Each of these commands allows both alternative-specific and case-specific predictors, and each one handles unbalanced choice sets properly. Each of these models can be derived as a random utility model in which each decision maker selects the alternative that provides the highest utility. See [CM] **Intro 8** for more information on the random utility model formulation of these discrete choice models.

The difference in these models largely hinges on an assumption known as independence of irrelevant alternatives (IIA). Briefly, the IIA assumption means that relative probability of selecting alternatives should not change if we introduce or eliminate another alternative. As an example, suppose that a restaurant has one chicken entree and one steak entree on the menu and that these are equally likely to be selected. If a vegetarian option is introduced, the probabilities of selecting chicken and steak will both decrease, but they should still be equal to each other if the IIA assumption holds. If the probability of selecting steak now is greater than the probability of selecting chicken, or vice versa, the IIA assumption does not hold. More technically, the IIA assumption means that the error terms cannot be correlated across alternatives. See [CM] **Intro 8** for more information on this assumption and how it applies to each choice model.

cmclogit fits McFadden's choice model using conditional logistic regression. Of the four models discussed in this entry, McFadden's choice model has the most straightforward formulation. However, it does require that you make the IIA assumption.

cmmixlogit fits a mixed logit regression for choice models. This model allows random coefficients on one or more of the alternative-specific predictors in the model. This means that the coefficients on these variables are allowed to vary across individuals. We do not estimate the coefficients for each individual. Instead, we assume that the coefficients follow a distribution such as normal distribution, and we estimate the parameters of that distribution. Through these random coefficients, the model allows correlation across alternatives. In this way, the mixed logit model relaxes the IIA assumption.

cmmprobit fits a multinomial probit choice model. Like cmclogit, this command estimates fixed coefficients for all predictors, but it relaxes the IIA assumption in another way. It directly models the correlation between the error terms for the different alternatives.

nlogit fits a nested logit choice model. With this model, similar alternatives—alternatives whose errors are likely to be correlated—can be grouped into nests. Extending our restaurant example, suppose there are now seven entrees. Three include chicken, two include steak, and two are vegetarian. The researcher could specify a nesting structure where entrees are grouped by type. The nested logit model then accounts for correlation of alternatives within the same nest and thus relaxes the IIA assumption.

Below, we provide further introductions to these models, demonstrate how to fit and interpret them using Stata, and tell you more about their relationships with each other and with other Stata estimation commands.

cmclogit: McFadden's choice model

McFadden's choice model is fit using conditional logistic regression. In Stata, this model can also be fit by the command clogit. In fact, cmclogit calls clogit to fit McFadden's choice model. However, cmclogit is designed for choice data and has features that clogit does not. cmclogit properly handles missing values for choice models, checks for errors in the alternatives variable and case-specific variables, and has appropriate postestimation commands such as the special version of margins designed for use after CM estimation.

To demonstrate cmclogit, we use the same dataset we used in [CM] **Intro 2**. We load the data, list the first three cases, and use cmset.

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. list consumerid car purchase gender income dealers if consumerid <= 3,
> sepby(consumerid) abbrev(10)
```

	consumerid	car	purchase	gender	income	dealers
1.	1	American	1	Male	46.7	9
2.	1	Japanese	0	Male	46.7	11
3.	1	European	0	Male	46.7	5
4.	1	Korean	0	Male	46.7	1
5.	2	American	1	Male	26.1	10
6.	2	Japanese	0	Male	26.1	7
7.	2	European	0	Male	26.1	2
8.	2	Korean	0	Male	26.1	1
9.	3	American	0	Male	32.7	8
10.	3	Japanese	1	Male	32.7	6
11.	3	European	0	Male	32.7	2

```
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.
     Case ID variable: consumerid
Alternatives variable: car
```

We passed `cmset` the case ID variable `consumerid` and the alternatives variable `car`, which contains possible choices of the nationality of car purchased, American, Japanese, European, or Korean.

The 0/1 variable `purchase` indicates which nationality of car was purchased. It is our dependent variable for `cmclogit`. Before we fit our model, let's run `cmtab` to see the observed choices in the data.

```
. cmtab, choice(purchase)
Tabulation of chosen alternatives (purchase = 1)
```

Nationality of car	Freq.	Percent	Cum.
American	384	43.39	43.39
Japanese	326	36.84	80.23
European	135	15.25	95.48
Korean	40	4.52	100.00
Total	885	100.00	

Most of the people in these data purchased American cars (43%), followed by Japanese cars (37%) and European cars (15%). Korean cars were purchased the least (5%).

For predictors, we have the case-specific variables `gender` and `income`, and the alternative-specific variable `dealers`, which contains the number of dealerships of each nationality in the consumer's community. We fit the model:

```
. cmclogit purchase dealers, casevars(i.gender income)

Iteration 0:  Log likelihood = -959.21405
Iteration 1:  Log likelihood = -948.48587
Iteration 2:  Log likelihood =  -948.1217
Iteration 3:  Log likelihood = -948.12096
Iteration 4:  Log likelihood = -948.12096

Conditional logit choice model                  Number of obs      =      3,075
Case ID variable: consumerid                    Number of cases    =        862

Alternatives variable: car                      Alts per case: min =          3
                                                               avg =        3.6
                                                               max =          4

                                                  Wald chi2(7)     =      51.03
Log likelihood = -948.12096                       Prob > chi2      =     0.0000
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	.0448082	.0262818	1.70	0.088	-.0067032	.0963196
American	(base alternative)					
Japanese						
gender						
Male	-.379326	.1712399	-2.22	0.027	-.71495	-.0437021
income	.0154978	.0065145	2.38	0.017	.0027296	.0282659
_cons	-.4787261	.331378	-1.44	0.149	-1.128215	.1707628
European						
gender						
Male	.653345	.2647694	2.47	0.014	.1344065	1.172283
income	.0343647	.0080286	4.28	0.000	.0186289	.0501006
_cons	-2.839606	.461613	-6.15	0.000	-3.744351	-1.934861
Korean						
gender						
Male	.0679233	.4464535	0.15	0.879	-.8071094	.942956
income	-.0377716	.0158434	-2.38	0.017	-.068824	-.0067191
_cons	.0511728	.8033048	0.06	0.949	-1.523276	1.625621

Note that alternative-specific variables (if any) follow the dependent variable. Case-specific variables (if any) are placed in the option `casevars()`. Because `cmclogit` requires us to specify which variables are alternative specific and which are case specific, it can verify that our data are coded as we expect. It checks whether the specified case-specific variables are truly case specific. If they are not, we get an error.

You may also see messages from `cmclogit` about the alternative-specific variables. For example,

```
note: variable dealers has 2 cases that are not alternative-specific; there is
      no within-case variability.
```

Alternative-specific variables can vary by alternative and by case, but they do not have to vary by alternative for every case. This message tells us that there are two cases for which the alternative-specific variable is constant within case. If an alternative-specific variable is constant within case for a large proportion of the cases, we might question how alternative specific that variable really is and

be concerned about its predictive value. If a variable that is supposed to be alternative specific is in fact case specific, we will get an error.

Looking at the results from `cmclogit`, we first see that the coefficient on `dealers` is positive; based on this model, we expect the probability of purchasing a vehicle of any nationality to increase as the number of dealerships increases. However, notice that this coefficient is different from 0 at the 10% level but not at the 5% level.

American cars are chosen as the base alternative, so coefficients on the alternative-specific variables are interpreted relative to them. For instance, for the Japanese alternative, the coefficient on `Male` is negative, which indicates that males are less likely to select a Japanese car than an American car.

Looking at cases with missing values using cmsample

From the header of the `cmclogit` output, we see that our model was fit using 862 cases in our model. However, we see from the previous `cmtab` output that there are a total of 885 cases in the data. There must be missing values in one or more of the variables. Let's track down the variables and the cases with missing values using `cmsample`. First, we run `cmsample` specifying all the variables we used with `cmclogit`. The only difference is that the dependent variable goes in the `choice()` option.

```
. cmsample dealers, choice(purchase) casevars(i.gender income)
```

Reason for exclusion	Freq.	Percent	Cum.
observations included	3,075	97.31	97.31
casevars missing	85	2.69	100.00
Total	3,160	100.00	

The results tell us that the missing values are in the *casevars*, either `gender` or `income` or both. Note that the tabulation produced by `cmsample` shows counts of observations not cases.

Second, we look at `gender` alone with `cmsample`:

```
. cmsample, casevars(i.gender) generate(flag)
```

Reason for exclusion	Freq.	Percent	Cum.
observations included	3,075	97.31	97.31
casevar missing	85	2.69	100.00
Total	3,160	100.00	

These are the cases with missing values. We also specified the `generate()` option to create a variable whose nonzero values indicate cases with missing values or other problems. We list these cases:

```
. sort consumerid car
. list consumerid car gender flag if flag != 0, sepby(consumerid) abbr(10)
```

	consumerid	car	gender	flag
509.	142	American	.	casevar missing
510.	142	Japanese	Male	casevar missing
511.	142	European	Male	casevar missing
512.	142	Korean	Male	casevar missing
516.	144	American	.	casevar missing
517.	144	Japanese	Male	casevar missing
518.	144	European	Male	casevar missing

(*output omitted*)

We could have listed the observations with missing values of `gender` by typing `list if missing(gender)`. But using `cmsample` in this way allows us to list entire cases, potentially giving us a way to fix the problem. In this example, we could decide all the nonmissing values of `gender` are valid and fill in the missing values with the nonmissing ones for that case. However, we will not do this for the purpose of our example.

See [CM] **cmsample** and example 3 in [CM] **cmclogit** for more on missing values in choice data.

margins after CM estimation

Above, we interpreted a few of the coefficients from the `clogit` results. In [CM] **Intro 1**, we showed you that you can use `margins` to further interpret the results of your choice model. Here we demonstrate how we can apply some of `margins` special choice model features to interpret the results of this model.

First, we type `margins` without any arguments to get the average predicted probabilities for the different alternatives.

```
. margins
Predictive margins                                      Number of obs = 3,075
Model VCE: OIM
Expression: Pr(car|1 selected), predict()
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome						
American	.4361949	.016755	26.03	0.000	.4033556	.4690342
Japanese	.3665893	.0162405	22.57	0.000	.3347585	.3984202
European	.1508121	.0119794	12.59	0.000	.1273328	.1742913
Korean	.0464037	.0069301	6.70	0.000	.032821	.0599865

Based on this model and assuming we have a random or otherwise representative sample, these are the expected proportions in the population.

`margins` can produce many types of estimates. Suppose we want to know how the probability of a person selecting a European car changes when the number of European dealerships increases. If this probability increases (as we expect it to), the increase must come at the expense of American, Japanese, or Korean cars. Which one of these is affected the most?

First, let's estimate the expected probability of purchasing each nationality of car if each community adds a new European dealership. We can use the `at(dealers=(dealers+1)` option to request this computation.

```
. margins, at(dealers=generate(dealers+1)) alternative(European)
Predictive margins                                      Number of obs = 3,075
Model VCE: OIM

Expression:  Pr(car|1 selected), predict()
Alternative: European
At: dealers = dealers+1
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome						
American	.4333003	.0168164	25.77	0.000	.4003407	.4662598
Japanese	.3641274	.0162604	22.39	0.000	.3322577	.3959971
European	.1564365	.0127751	12.25	0.000	.1313978	.1814752
Korean	.0461358	.0068959	6.69	0.000	.0326201	.0596516

These look similar to the expected probabilities we estimated using the original number of dealerships in each community. By using the `contrast()` option, we can estimate the differences between these probabilities and the original ones. We include the `nowald` option to simplify the output.

```
. margins, at(dealers=generate(dealers)) at(dealers=generate(dealers+1))
> alternative(European) contrast(atcontrast(r) nowald)
Contrasts of predictive margins                         Number of obs = 3,075
Model VCE: OIM

Expression:  Pr(car|1 selected), predict()
Alternative: European

1._at: dealers =   dealers

2._at: dealers = dealers+1
```

	Contrast	Delta-method std. err.	[95% conf.	interval]
_at@_outcome				
(2 vs 1) American	-.0028946	.0017268	-.0062791	.0004899
(2 vs 1) Japanese	-.0024619	.0014701	-.0053434	.0004195
(2 vs 1) European	.0056244	.0033521	-.0009456	.0121944
(2 vs 1) Korean	-.0002679	.0001686	-.0005983	.0000625

Increasing the number of European dealerships by one increases the expected probability of selecting a European car by 0.0056. This increase comes at the expense of American cars slightly more than Japanese cars. The probability of someone purchasing an American car decreases by 0.0029, and the probability of someone purchasing a Japanese car decreases by 0.0025. The probability of buying a Korean car is barely changed, only a tiny decrease of 0.0003 in the probability. All of these changes are very small. We can look at the 95% confidence intervals to see that none of these changes in probabilities is significantly different from 0 at the 5% level.

We will ignore the lack of significance for now and explore one of `margins`'s features specific to choice models. As we mentioned before, the choice sets are unbalanced. Some consumers do not have the choice of a Korean car (corresponding to `car == 4`) as one of their available alternatives.

```
. cmchoiceset
Tabulation of choice-set possibilities
 Choice set |      Freq.     Percent        Cum.
------------+-----------------------------------
      1 2 3 |        380       42.94       42.94
    1 2 3 4 |        505       57.06      100.00
------------+-----------------------------------
      Total |        885      100.00
Note: Total is number of cases.
```

How does `margins` handle the fact that some persons do not have the choice of Korean cars among their alternatives? By default, `margins` sets the probability of buying a Korean car for these consumers to zero and keeps it fixed at zero.

If we want to look at only those consumers who have Korean in their choice set, we can use the `outcome(... , altsubpop)` option.

```
. margins, at(dealers=generate(dealers)) at(dealers=generate(dealers+1))
> alternative(European) contrast(atcontrast(r) nowald) outcome(Korean, altsubpop)
Contrasts of predictive margins                         Number of obs = 3,075
Model VCE: OIM
Expression:  Pr(car|1 selected), predict()
Alternative: European
Outcome:     Korean
1._at: dealers =   dealers
2._at: dealers = dealers+1
------------------------------------------------------------------------------
             |            Delta-method
             |   Contrast   std. err.     [95% conf. interval]
-------------+----------------------------------------------------------------
         _at |
   (2 vs 1)  |  -.0004722   .0002972     -.0010547    .0001103
------------------------------------------------------------------------------
```

The probability of buying a Korean car among those who have the choice of buying a Korean decreases by 0.0005 when a European dealership is added. This change is bigger than what we estimated earlier, as we expect, because we omitted all those persons whose change was fixed at zero.

When we model these data, it seems reasonable to keep the probability of buying a Korean car fixed at zero for those consumers who do not have Korean in their choice set. The result gives a picture of the total population represented by the sample; to omit them gives a picture of only those communities with Korean dealerships. See [CM] **margins** for more examples and another discussion of this issue.

If you have not already read [CM] **Intro 1**, we recommend that you also read the examples of interpreting results of `cm` commands using `margins` that are provided in that entry. For more information on `margins`, see its main entry in the Stata manuals, [R] **margins**. You will also want to see the separate entry for it in this manual, [CM] **margins**, which describes the special features of this command when used after `cm` commands and includes lots of choice model examples.

cmmixlogit: Mixed logit choice models

cmmixlogit fits a mixed logit regression for choice data. Like cmclogit, cmmixlogit is used to model the probability that a decision maker chooses one alternative from a set of available alternatives.

In the mixed logit model, the coefficients on alternative-specific variables can be treated as fixed or random. Specifying random coefficients can model correlation of choices across alternatives, thereby relaxing the IIA property that is imposed by McFadden's choice model. In this sense, the mixed logit model fit by cmmixlogit is more general than models fit by cmclogit. McFadden and Train (2000) show that the mixed logit model can approximate a wide class of choice representations. See [CM] **Intro 8** for a description of the IIA property and how mixed logit models can fit deviations from it.

We continue with the same dataset we have been using in this introduction: consumer data on choices of nationalities of cars. The data arrangement required by cmmixlogit is exactly the same as that for cmclogit.

Mixed logit choice models can fit random coefficients for alternative-specific variables. We take dealers, the number of dealers of each nationality in each consumer's community, which is an alternative-specific variable, and fit random coefficients for it.

```
. cmmixlogit purchase, random(dealers) casevars(i.gender income)
note: alternatives are unbalanced.

Fitting fixed parameter model:

Fitting full model:

Iteration 0:  Log simulated-likelihood = -966.81349
Iteration 1:  Log simulated-likelihood = -949.54388
Iteration 2:  Log simulated-likelihood = -948.15757
Iteration 3:  Log simulated-likelihood = -948.12546
Iteration 4:  Log simulated-likelihood = -948.12106
Iteration 5:  Log simulated-likelihood = -948.12099
Iteration 6:  Log simulated-likelihood = -948.12097
Iteration 7:  Log simulated-likelihood = -948.12096

Mixed logit choice model                    Number of obs      =      3,075
Case ID variable: consumerid                Number of cases    =        862

Alternatives variable: car                  Alts per case: min =          3
                                                           avg =        3.6
                                                           max =          4

Integration sequence:      Hammersley
Integration points:               623       Wald chi2(7)       =      51.03
Log simulated-likelihood = -948.12096       Prob > chi2        =     0.0000
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	.0448203	.0262821	1.71	0.088	-.0066917	.0963323
/Normal						
sd(dealers)	.0001994	.1981032			.	.
American	(base alternative)					
Japanese						
gender						
Male	-.3793276	.1712401	-2.22	0.027	-.714952	-.0437032
income	.015498	.0065144	2.38	0.017	.00273	.0282661
_cons	-.4786729	.3313762	-1.44	0.149	-1.128158	.1708125
European						
gender						
Male	.6533193	.2647746	2.47	0.014	.1343706	1.172268
income	.0343656	.0080288	4.28	0.000	.0186295	.0501017
_cons	-2.839604	.4616206	-6.15	0.000	-3.744363	-1.934844
Korean						
gender						
Male	.0676844	.4464111	0.15	0.879	-.8072653	.9426341
income	-.0377614	.0158428	-2.38	0.017	-.0688128	-.00671
_cons	.0511088	.8032683	0.06	0.949	-1.523268	1.625486

```
LR test vs. fixed parameters: chibar2(01) =     0.00  Prob >= chibar2 = 0.5000
```

The estimated standard deviation for the random coefficient is small, and the likelihood-ratio test shown at the bottom of the table that compares this random-coefficients model with a fixed-coefficient model is not significant. A model with random coefficients for `dealers` is no better than one with a fixed coefficient. Note that this fixed-coefficient model is precisely the model fit earlier by `cmclogit`.

We used the default distribution for the random coefficients: a normal (Gaussian) distribution. Let's fit the model again using a lognormal distribution for the coefficient of `dealers`.

Because the lognormal distribution is only defined over positive real values, the coefficient values coming from this distribution will only be positive. This constrains the coefficient to be positive. Is this constraint okay? We believe that increasing the number of dealerships in a community of a given nationality should always increase the probability that someone in the community buys that type of car and never decrease the probability. So constraining the coefficient to be positive is what we want. (If we want to constrain the coefficient to be negative, we could create a variable equal to `-dealers` and fit a random lognormal coefficient for it.)

```
. cmmixlogit purchase, random(dealers, lognormal) casevars(i.gender income)
note: alternatives are unbalanced.

Fitting fixed parameter model:

Fitting full model:

Iteration 0:  Log simulated-likelihood = -948.13062
Iteration 1:  Log simulated-likelihood =  -948.1226
Iteration 2:  Log simulated-likelihood = -948.12155
Iteration 3:  Log simulated-likelihood = -948.12106
Iteration 4:  Log simulated-likelihood = -948.12096
Iteration 5:  Log simulated-likelihood = -948.12096

Mixed logit choice model                    Number of obs      =      3,075
Case ID variable: consumerid                Number of cases    =        862
Alternatives variable: car                  Alts per case: min =          3
                                                           avg =        3.6
                                                           max =          4
Integration sequence:      Hammersley
Integration points:               623       Wald chi2(7)       =      79.14
Log simulated-likelihood = -948.12096       Prob > chi2        =     0.0000
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	-3.105499	.5869861	-5.29	0.000	-4.255971	-1.955028
/Lognormal						
sd(dealers)	.0036636	4.480108			.	.
American	(base alternative)					
Japanese						
gender						
Male	-.3793272	.1712406	-2.22	0.027	-.7149526	-.0437018
income	.0154978	.0065145	2.38	0.017	.0027296	.0282661
_cons	-.4787181	.3313811	-1.44	0.149	-1.128213	.170777
European						
gender						
Male	.6533465	.2647669	2.47	0.014	.1344129	1.17228
income	.0343648	.0080286	4.28	0.000	.0186291	.0501005
_cons	-2.83959	.4616219	-6.15	0.000	-3.744353	-1.934828
Korean						
gender						
Male	.0679287	.4464459	0.15	0.879	-.8070892	.9429466
income	-.0377715	.0158431	-2.38	0.017	-.0688234	-.0067196
_cons	.0511891	.8033007	0.06	0.949	-1.523251	1.62563

```
LR test vs. fixed parameters: chibar2(01) =     0.00  Prob >= chibar2 = 0.5000
```

The random-coefficients model is still not significantly different from a fixed coefficient model.

At first glance, the requirement of including random coefficients on alternative-specific variables in this model may seem limiting. What if we do not have alternative-specific variables for which random coefficients are appropriate? Note that the constants in the model are alternative specific. They are automatically included in the model for us, but we could have equivalently typed `i.car` in the list of alternative-specific variables to include indicators for the alternatives. We can turn any of or all the constants into random intercepts. Let's do this with the constant for the European alternative. Now we need to use the factor-variable specification for the alternative-specific constants. Because we want fixed coefficients on Japanese and Korean indicators, we type `i(2 4).car` in the fixed portion of the model. To get random coefficients for the European constant, we type `random(i3.car)`. We also specify the options `noconstant` and `collinear` (or else `cmmixlogit` would drop the constants).

```
. cmmixlogit purchase dealers i(2 4).car, random(i3.car)
> casevars(i.gender income) noconstant collinear
note: alternatives are unbalanced.

Fitting fixed parameter model:

Fitting full model:

Iteration 0:  Log simulated-likelihood = -1717.8292  (not concave)
Iteration 1:  Log simulated-likelihood = -1471.6665  (not concave)
Iteration 2:  Log simulated-likelihood = -1456.0693  (not concave)
Iteration 3:  Log simulated-likelihood = -1431.4506  (not concave)
Iteration 4:  Log simulated-likelihood = -1412.2678  (not concave)
Iteration 5:  Log simulated-likelihood = -1382.4808  (not concave)
Iteration 6:  Log simulated-likelihood = -1359.4781  (not concave)
Iteration 7:  Log simulated-likelihood = -1341.5917  (not concave)
Iteration 8:  Log simulated-likelihood = -1327.6059  (not concave)
Iteration 9:  Log simulated-likelihood = -1316.6209  (not concave)
Iteration 10: Log simulated-likelihood = -1307.9616  (not concave)
Iteration 11: Log simulated-likelihood = -1294.3419  (not concave)
Iteration 12: Log simulated-likelihood =  -1155.848  (not concave)
Iteration 13: Log simulated-likelihood = -998.89495
Iteration 14: Log simulated-likelihood = -950.28922
Iteration 15: Log simulated-likelihood = -949.17489
Iteration 16: Log simulated-likelihood = -949.17151
Iteration 17: Log simulated-likelihood = -949.16844
Iteration 18: Log simulated-likelihood = -949.16776
Iteration 19: Log simulated-likelihood = -949.16759
Iteration 20: Log simulated-likelihood = -949.16755
Iteration 21: Log simulated-likelihood = -949.16754
```

```
Mixed logit choice model                        Number of obs      =      3,075
Case ID variable: consumerid                    Number of cases    =        862
Alternatives variable: car                      Alts per case: min =          3
                                                               avg =        3.6
                                                               max =          4
Integration sequence:      Hammersley
Integration points:               623           Wald chi2(9)       =     200.60
Log simulated-likelihood = -949.16754           Prob > chi2        =     0.0000
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	.0502172	.0259823	1.93	0.053	-.0007071	.1011415
car						
Korean	.2926751	.7850917	0.37	0.709	-1.246076	1.831427
European	-2.591301	.4270812	-6.07	0.000	-3.428365	-1.754237
/Normal						
sd(3.car)	.0001367	1.640459			.	.
American	(base alternative)					
Japanese						
gender						
Male	-.5113203	.1448827	-3.53	0.000	-.7952852	-.2273554
income	.007125	.0029512	2.41	0.016	.0013408	.0129093
European						
gender						
Male	.5843488	.2610755	2.24	0.025	.0726502	1.096047
income	.0304615	.0075051	4.06	0.000	.0157518	.0451712
Korean						
gender						
Male	.0012654	.444595	0.00	0.998	-.8701249	.8726556
income	-.0413199	.0156261	-2.64	0.008	-.0719464	-.0106934

```
LR test vs. fixed parameters: chibar2(01) =     0.00  Prob >= chibar2 = 0.5000
```

This model with a random intercept for the European alternative is not significantly different from a fixed-coefficient model. But this illustrates one of the features of `cmmixlogit`. Making the alternative-specific constants random allows us to fit models that do not satisfy IIA and test them against a fixed-coefficient model that does satisfy IIA.

See [CM] **cmmixlogit** for examples where the random-coefficients model fits better than the one with fixed coefficients. There we demonstrate how to further interpret results of these models. In addition, you can use `margins` in the same ways shown in [CM] **Intro 1** and as we did after `cmclogit` above to interpret mixed logit models.

cmmprobit: Multinomial probit choice models

`cmmprobit` fits a multinomial probit (MNP) choice model. The formulation of the utility for MNP is described in [CM] **Intro 8**. The model is similar to McFadden's choice model (`cmclogit`), except that the random-error term is modeled using a multivariate normal distribution, and you can explicitly model the covariance.

When there are no alternative-specific variables in your model, covariance parameters are not identifiable. In this case, better alternatives are `mprobit`, which is geared specifically toward models with only case-specific variables, or a random-intercept model fit by `cmmixlogit`.

The covariance parameters are set using the `correlation()` and `stddev()` options of `cmmprobit`. In general, there are $J(J+1)/2$ possible covariance parameters, where J is the number of possible alternatives. One of the alternatives is set as the base category, and only the relative differences among the utilities matter. This reduces the possible number of covariance parameters by J.

The scale of the utilities does not matter. Multiply the utilities for all alternatives by the same constant, and the relative differences are unchanged. This further reduces the number of covariance parameters by one. So there are a total of $J(J-1)/2-1$ covariance parameters you can fit. But you do not have to fit all of them. You can set some of them to fixed values, either zero or nonzero. Or you can constrain some of them to be equal.

When J is large, it is a good idea to initially fit just a few parameters and then gradually increase the number. If you try to fit a lot of parameters, your model may have a hard time converging because some of the parameters may not be identified. For example, the true variance for one of the alternatives may be zero, and if you try to estimate the standard deviation for the alternative, the model may not converge because zero is not part of the estimable parameter space.

See *Covariance structures* in [CM] **cmmprobit** for full details on all the choices for specifying the covariance parameters.

`cmmprobit` has some options for reducing the number of covariance parameters. In particular, `correlation(exchangeable)` fits a model in which correlations between the alternatives are all the same. Another way to reduce the number of parameters estimated is the `factor(#)` option. `cmmprobit` with `factor(#)` fits a covariance matrix of the form $I+\mathbf{C}'\mathbf{C}$, where the row dimension of the matrix $\mathbf{C}$ is #.

Let's fit a model using `factor(1)` with the data from the previous examples.

```
. cmmprobit purchase dealers, casevars(i.gender income) factor(1)

Iteration 0:  Log simulated-likelihood = -949.38598
Iteration 1:  Log simulated-likelihood = -949.08161  (backed up)
Iteration 2:  Log simulated-likelihood = -948.87143  (backed up)
Iteration 3:  Log simulated-likelihood = -948.84362  (backed up)
Iteration 4:  Log simulated-likelihood = -948.83433  (backed up)
Iteration 5:  Log simulated-likelihood = -948.53624  (backed up)
Iteration 6:  Log simulated-likelihood = -948.52521
Iteration 7:  Log simulated-likelihood = -948.42813
Iteration 8:  Log simulated-likelihood = -948.14286
Iteration 9:  Log simulated-likelihood = -948.03466
Iteration 10: Log simulated-likelihood = -948.01302
Iteration 11: Log simulated-likelihood = -947.83629
Iteration 12: Log simulated-likelihood = -947.78297
Iteration 13: Log simulated-likelihood =  -947.6765
Iteration 14: Log simulated-likelihood = -947.60503
Iteration 15: Log simulated-likelihood =  -947.5831
Iteration 16: Log simulated-likelihood = -947.55131
Iteration 17: Log simulated-likelihood = -947.50624
Iteration 18: Log simulated-likelihood = -947.46284
Iteration 19: Log simulated-likelihood = -947.44467
Iteration 20: Log simulated-likelihood = -947.40163
Iteration 21: Log simulated-likelihood = -947.32181
Iteration 22: Log simulated-likelihood = -947.29791
Iteration 23: Log simulated-likelihood = -947.23404
Iteration 24: Log simulated-likelihood = -947.17847
Iteration 25: Log simulated-likelihood = -947.13231
Iteration 26: Log simulated-likelihood = -947.08427
```

```
Iteration 27: Log simulated-likelihood = -946.83137
Iteration 28: Log simulated-likelihood = -946.73195
Iteration 29: Log simulated-likelihood = -946.44451
Iteration 30: Log simulated-likelihood = -946.37077
Iteration 31: Log simulated-likelihood = -946.34252
Iteration 32: Log simulated-likelihood = -946.32218
Iteration 33: Log simulated-likelihood = -946.31672
Iteration 34: Log simulated-likelihood = -946.31499
Iteration 35: Log simulated-likelihood = -946.31489
Iteration 36: Log simulated-likelihood = -946.31487
Iteration 37: Log simulated-likelihood = -946.31486
Iteration 38: Log simulated-likelihood =  -946.3148
Iteration 39: Log simulated-likelihood =  -946.3141
Iteration 40: Log simulated-likelihood = -946.31203
Iteration 41: Log simulated-likelihood =  -946.3114
Iteration 42: Log simulated-likelihood = -946.31114
Iteration 43: Log simulated-likelihood = -946.31109
Iteration 44: Log simulated-likelihood = -946.31109

Multinomial probit choice model                 Number of obs      =      3,075
Case ID variable: consumerid                    Number of cases    =        862

Alternatives variable: car                      Alts per case: min =          3
                                                               avg =        3.6
                                                               max =          4

Integration sequence:      Hammersley
Integration points:               704           Wald chi2(7)       =      33.07
Log simulated-likelihood = -946.31109           Prob > chi2        =     0.0000
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	.043345	.027397	1.58	0.114	-.0103522	.0970422
American	(base alternative)					
Japanese						
gender						
Male	-.4549281	.1454853	-3.13	0.002	-.7400741	-.1697821
income	.0092406	.0054458	1.70	0.090	-.0014331	.0199142
_cons	-.419605	.2779042	-1.51	0.131	-.9642872	.1250773
European						
gender						
Male	.5630869	.4209101	1.34	0.181	-.2618817	1.388056
income	.0201237	.0102355	1.97	0.049	.0000625	.0401849
_cons	-2.273778	1.499661	-1.52	0.129	-5.21306	.6655044
Korean						
gender						
Male	.3081901	.4970798	0.62	0.535	-.6660685	1.282449
income	-.035191	.0346554	-1.02	0.310	-.1031144	.0327323
_cons	-.9509444	1.056018	-0.90	0.368	-3.020701	1.118812
/c1_2	-.8477297	1.362819	-0.62	0.534	-3.518807	1.823347
/c1_3	-1.675403	1.433511	-1.17	0.243	-4.485033	1.134226

```
(car=American is the alternative normalizing location)
(car=Japanese is the alternative normalizing scale)

. matrix b704 = e(b)
```

The estimated covariance parameters are shown in the output, but more useful is to see the estimated covariance matrix or correlation matrix. The postestimation command `estat` will display them. `estat covariance` shows the covariance matrix, and `estat correlation` shows the correlations.

```
. estat covariance
```

	Japanese	European	Korean
Japanese	2		
European	-.8477297	1.718646	
Korean	-1.675403	1.420289	3.806976

```
Note: Covariances are for alternatives differenced with American.
. estat correlation
```

	Japanese	European	Korean
Japanese	1.0000		
European	-0.4572	1.0000	
Korean	-0.6072	0.5553	1.0000

```
Note: Correlations are for alternatives differenced with American.
```

There are four alternatives in these data. But the matrices shown here are only 3×3. This is because the parameterization for the covariance matrix is, by default, differed by the base category, which in this case is the alternative American.

To see an undifferenced parameterization, we specify the `structural` option:

```
. cmmprobit purchase dealers, casevars(i.gender income) factor(1) structural
Iteration 0:  Log simulated-likelihood = -949.81324
Iteration 1:  Log simulated-likelihood = -948.95649  (backed up)
Iteration 2:  Log simulated-likelihood = -948.71164  (backed up)
Iteration 3:  Log simulated-likelihood = -948.70869  (backed up)
Iteration 4:  Log simulated-likelihood = -948.65719  (backed up)
Iteration 5:  Log simulated-likelihood = -948.52707
Iteration 6:  Log simulated-likelihood = -948.52682
Iteration 7:  Log simulated-likelihood = -948.44886
Iteration 8:  Log simulated-likelihood = -948.29451
Iteration 9:  Log simulated-likelihood = -948.22865
Iteration 10: Log simulated-likelihood = -948.14213
Iteration 11: Log simulated-likelihood = -947.96801
Iteration 12: Log simulated-likelihood = -947.95862
Iteration 13: Log simulated-likelihood = -947.85813
Iteration 14: Log simulated-likelihood = -947.84956
Iteration 15: Log simulated-likelihood =  -947.7153
Iteration 16: Log simulated-likelihood = -947.67296
Iteration 17: Log simulated-likelihood = -947.57769
Iteration 18: Log simulated-likelihood = -947.42721
Iteration 19: Log simulated-likelihood = -947.19551
Iteration 20: Log simulated-likelihood = -947.11421
Iteration 21: Log simulated-likelihood = -946.90873
Iteration 22: Log simulated-likelihood = -946.75482
Iteration 23: Log simulated-likelihood = -946.64695
Iteration 24: Log simulated-likelihood = -946.56345
Iteration 25: Log simulated-likelihood = -946.44076
Iteration 26: Log simulated-likelihood =  -946.3817
Iteration 27: Log simulated-likelihood = -946.35537
Iteration 28: Log simulated-likelihood = -946.34227
Iteration 29: Log simulated-likelihood = -946.33841
Iteration 30: Log simulated-likelihood = -946.33808
```

```
Iteration 31: Log simulated-likelihood = -946.33792
Iteration 32: Log simulated-likelihood = -946.33659
Iteration 33: Log simulated-likelihood = -946.33345
Iteration 34: Log simulated-likelihood = -946.32049
Iteration 35: Log simulated-likelihood = -946.30741
Iteration 36: Log simulated-likelihood = -946.29907
Iteration 37: Log simulated-likelihood = -946.29486
Iteration 38: Log simulated-likelihood = -946.29353
Iteration 39: Log simulated-likelihood = -946.29243
Iteration 40: Log simulated-likelihood = -946.29179
Iteration 41: Log simulated-likelihood = -946.29159
Iteration 42: Log simulated-likelihood = -946.29158
```

```
Multinomial probit choice model              Number of obs      =      3,075
Case ID variable: consumerid                 Number of cases    =        862

Alternatives variable: car                   Alts per case: min =          3
                                                            avg =        3.6
                                                            max =          4

Integration sequence:      Hammersley
Integration points:               704        Wald chi2(7)       =      29.28
Log simulated-likelihood = -946.29158        Prob > chi2        =     0.0001
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	.0702705	.0443614	1.58	0.113	-.0166763	.1572172
American	(base alternative)					
Japanese						
gender						
Male	-.5091748	.1855697	-2.74	0.006	-.8728848	-.1454648
income	.013291	.0067122	1.98	0.048	.0001354	.0264467
_cons	-.4826255	.3426393	-1.41	0.159	-1.154186	.1889352
European						
gender						
Male	.784163	.7110074	1.10	0.270	-.6093859	2.177712
income	.0295921	.0163378	1.81	0.070	-.0024295	.0616136
_cons	-3.121519	2.487812	-1.25	0.210	-7.997541	1.754503
Korean						
gender						
Male	.5169586	.6933728	0.75	0.456	-.8420271	1.875944
income	-.0271951	.021043	-1.29	0.196	-.0684387	.0140485
_cons	-1.150509	1.559621	-0.74	0.461	-4.20731	1.906292
/c1_3	-1.407566	2.060875	-0.68	0.495	-5.446806	2.631674
/c1_4	-1.709069	1.381237	-1.24	0.216	-4.416244	.9981062

```
(car=American is the alternative normalizing location)
(car=Japanese is the alternative normalizing scale)
```

```
. estat covariance
```

	American	Japanese	European	Korean
American	1			
Japanese	0	2		
European	0	-1.407566	2.981243	
Korean	0	-1.709069	2.405628	3.920917

```
. estat correlation
```

	American	Japanese	European	Korean
American	1.0000			
Japanese	0.0000	1.0000		
European	0.0000	-0.5764	1.0000	
Korean	0.0000	-0.6103	0.7036	1.0000

When using the `structural` option, you must carefully specify the covariance parameterization because, as we described earlier, not all of $J(J+1)/2$ elements of the covariance matrix are identifiable. There are at most $J(J-1)/2-1$ estimable parameters, so either elements have to be set to fixed values or constraints need to be imposed. Specifying any desired parameterization is straightforward. It merely requires learning how to use the `correlation()` and `stddev()` options. See *Covariance structures* in [CM] **cmmprobit**.

nlogit: Nested logit choice models

`nlogit` fits nested logit choice models. Alternatives can be nested within alternatives. For example, the data could represent first-level choices of what restaurant to dine at and second-level choices of what is ordered at the restaurant. Clearly, the menu choices will depend upon the type of restaurant. The second-level alternatives are conditional on the first-level alternatives.

Although `nlogit` fits choice models, it is not a `cm` command, and you do not have to `cmset` your data. Because of the nested alternatives, `nlogit` has its own unique data requirements.

See [CM] **nlogit** for full details on nested logit choice models.

Relationships with other estimation commands

If you are familiar with conditional logistic regression or with multinomial logistic regression, you may find it helpful to see how the `cm` commands, and in particular `cmclogit`, compare with Stata's `clogit` and `mlogit` commands.

Duplicating cmclogit using clogit

Both `cmclogit` and `clogit` fit conditional logistic regression models. `cmclogit` has special handling of errors, alternative-specific and case-specific variables, and special postestimation commands that are appropriate for choice data. However, you can fit the same model with `cmclogit` and `clogit`.

Before we try to duplicate our `cmclogit` results with `clogit`, we will drop the cases with missing values using the `flag` variable that we created with our earlier `cmsample` command. We do this because `clogit` does not handle missing values the same way `cmclogit` does. By default, `cmclogit` drops the entire case when any observation in the case has a missing value. `clogit` drops only the observations that contain missing values.

```
. drop if flag != 0
(85 observations deleted)
```

To duplicate our `cmclogit` results with `clogit`, we merely have to create interactions of the case-specific variables (`gender` and `income`) with the alternatives variable `car`. To do this, we include the factor-variable terms `car##gender` and `car##c.income` in our `clogit` specification. (We use `c.income` because `income` is continuous; see [U] **11.4.3 Factor variables** for more on factor variables.) The alternative-specific variable `dealers` is included in the estimation as is.

```
. clogit purchase dealers car##gender car##c.income, group(consumerid)
note: 1.gender omitted because of no within-group variance.
note: income omitted because of no within-group variance.

Iteration 0:  Log likelihood = -959.21405
Iteration 1:  Log likelihood = -948.48587
Iteration 2:  Log likelihood =  -948.1217
Iteration 3:  Log likelihood = -948.12096
Iteration 4:  Log likelihood = -948.12096

Conditional (fixed-effects) logistic regression         Number of obs =  3,075
                                                        LR chi2(10)   = 279.12
                                                        Prob > chi2   = 0.0000
Log likelihood = -948.12096                             Pseudo R2     = 0.1283
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
dealers	.0448082	.0262818	1.70	0.088	-.0067032	.0963196
car						
Japanese	-.4787261	.331378	-1.44	0.149	-1.128215	.1707628
European	-2.839606	.461613	-6.15	0.000	-3.744351	-1.934861
Korean	.0511728	.8033048	0.06	0.949	-1.523276	1.625621
gender						
Male	0	(omitted)				
car#gender						
Japanese # Male	-.379326	.1712399	-2.22	0.027	-.71495	-.0437021
European # Male	.653345	.2647694	2.47	0.014	.1344065	1.172283
Korean#Male	.0679233	.4464535	0.15	0.879	-.8071094	.942956
income	0	(omitted)				
car#c.income						
Japanese	.0154978	.0065145	2.38	0.017	.0027296	.0282659
European	.0343647	.0080286	4.28	0.000	.0186289	.0501006
Korean	-.0377716	.0158434	-2.38	0.017	-.068824	-.0067191

The output is in a different order, but all the coefficient estimates and their standard errors are exactly the same as the earlier results from `cmclogit`.

And they should be—because `cmclogit` calls `clogit` to do the estimation.

Multinomial logistic regression and McFadden's choice model

Multinomial logistic regression (`mlogit`) is a special case of McFadden's choice model (`cmclogit`). When there are only case-specific variables in the model and when the choice sets are balanced (that is, every case has the same alternatives), then `mlogit` gives the same results as `cmclogit`.

We can illustrate this, but the choice data we are working with are not balanced. So let's just use a subset of the dataset that is balanced. We can see the distinct choice sets using `cmchoiceset`.

```
. cmchoiceset, generate(choiceset)
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2 3	373	43.27	43.27
1 2 3 4	489	56.73	100.00
Total	862	100.00	

```
Note: Total is number of cases.
```

We included the `generate()` option to create an indicator variable `choiceset` with categories of the choice sets. We use this variable to keep only those cases that have the alternatives $\{1, 2, 3, 4\}$.

```
. keep if choiceset == "1 2 3 4":choiceset
(1,119 observations deleted)
```

(If you are not familiar with the `"1 2 3 4":choiceset` syntax, see [U] **13.11 Label values**.)

Now we run `cmclogit` on this balanced sample:

```
. cmclogit purchase, casevars(i.gender income)
Iteration 0:  Log likelihood = -580.83991
Iteration 1:  Log likelihood = -575.60247
Iteration 2:  Log likelihood = -575.21416
Iteration 3:  Log likelihood = -575.21287
Iteration 4:  Log likelihood = -575.21287
Conditional logit choice model                  Number of obs      =      1,956
Case ID variable: consumerid                    Number of cases    =        489
Alternatives variable: car                      Alts per case: min =          4
                                                               avg =        4.0
                                                               max =          4
                                                  Wald chi2(6)     =      41.24
Log likelihood = -575.21287                       Prob > chi2      =     0.0000
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
American	(base alternative)					
Japanese						
gender						
Male	-.7164669	.2351233	-3.05	0.002	-1.1773	-.2556338
income	.0174375	.0087817	1.99	0.047	.0002257	.0346493
_cons	-.2370371	.4413551	-0.54	0.591	-1.102077	.6280029
European						
gender						
Male	.2128877	.3494225	0.61	0.542	-.4719679	.8977433
income	.0409691	.0110817	3.70	0.000	.0192494	.0626888
_cons	-2.940079	.5956109	-4.94	0.000	-4.107455	-1.772703
Korean						
gender						
Male	-.1892108	.4595242	-0.41	0.681	-1.089862	.71144
income	-.0361748	.016143	-2.24	0.025	-.0678145	-.004535
_cons	-.0367581	.8051745	-0.05	0.964	-1.614871	1.541355

To run `mlogit`, we must create a categorical dependent variable containing the chosen alternative, American, Japanese, European, or Korean. The values of the alternatives variable `car` at the observations representing the chosen alternative (`purchase` equal to one) yield a dependent variable appropriate for `mlogit`.

```
. keep if purchase == 1
(1,467 observations deleted)

. mlogit car i.gender income

Iteration 0:  Log likelihood = -596.47415
Iteration 1:  Log likelihood = -575.81328
Iteration 2:  Log likelihood = -575.21417
Iteration 3:  Log likelihood = -575.21287
Iteration 4:  Log likelihood = -575.21287

Multinomial logistic regression                 Number of obs =    489
                                                LR chi2(6)    =  42.52
                                                Prob > chi2   = 0.0000
Log likelihood = -575.21287                     Pseudo R2     = 0.0356
```

car	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
American	(base outcome)					
Japanese						
gender						
Male	-.7164669	.2351233	-3.05	0.002	-1.1773	-.2556338
income	.0174375	.0087817	1.99	0.047	.0002257	.0346493
_cons	-.2370371	.4413551	-0.54	0.591	-1.102077	.6280029
European						
gender						
Male	.2128877	.3494225	0.61	0.542	-.4719679	.8977433
income	.0409691	.0110817	3.70	0.000	.0192494	.0626888
_cons	-2.940079	.5956109	-4.94	0.000	-4.107455	-1.772703
Korean						
gender						
Male	-.1892108	.4595242	-0.41	0.681	-1.089862	.71144
income	-.0361748	.016143	-2.24	0.025	-.0678145	-.004535
_cons	-.0367581	.8051745	-0.05	0.964	-1.614871	1.541355

The estimates are identical.

Estimation considerations

When fitting choice models, you may need to address issues such as setting the number of integration points, lack of convergence, or data with multiple outcomes selected. Below, we provide advice on these topics.

Setting the number of integration points

In *Maximum simulated likelihood* of [CM] **Intro 8**, we describe how the estimators for `cmmixlogit`, `cmxtmixlogit`, `cmmprobit`, and `cmroprobit` all approximate integrals using Monte-Carlo simulation to compute their likelihoods. Monte-Carlo simulation creates additional variance in the estimated results, and the variance is dependent on the number of points used in the integration. More points give smaller Monte-Carlo variance. Hence, when fitting final models, it is a good idea to use the option `intpoints(#)` to increase the number of integration points and check that the coefficient and parameter estimates and their standard estimates are stable. That is, check that they do not change appreciably as the number of integration points is increased.

In the first cmmprobit example in this introduction, the default number of integration points was 704. We stored the coefficient vector from that estimation in the vector b704. Let's open a fresh copy of our data and refit the same model, specifying intpoints(2000).

```
. use https://www.stata-press.com/data/r18/carchoice, clear
(Car choice data)

. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.

     Case ID variable: consumerid
Alternatives variable: car

. cmmprobit purchase dealers, casevars(i.gender income) factor(1)
> intpoints(2000)
```

(*iteration log omitted*)

```
Multinomial probit choice model                 Number of obs      =      3,075
Case ID variable: consumerid                    Number of cases    =        862

Alternatives variable: car                      Alts per case: min =          3
                                                               avg =        3.6
                                                               max =          4

Integration sequence:      Hammersley
Integration points:              2000             Wald chi2(7)     =      32.62
Log simulated-likelihood = -946.31243             Prob > chi2      =     0.0000
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	.0440584	.0279595	1.58	0.115	-.0107413	.0988581
American	(base alternative)					
Japanese						
gender						
Male	-.4558936	.1451544	-3.14	0.002	-.740391	-.1713961
income	.0091975	.0054259	1.70	0.090	-.0014371	.019832
_cons	-.4174475	.2776516	-1.50	0.133	-.9616346	.1267396
European						
gender						
Male	.57696	.4404936	1.31	0.190	-.2863916	1.440312
income	.0204216	.0108951	1.87	0.061	-.0009325	.0417757
_cons	-2.326064	1.586286	-1.47	0.143	-5.435127	.7829993
Korean						
gender						
Male	.3182168	.5023253	0.63	0.526	-.6663227	1.302756
income	-.0345119	.033025	-1.05	0.296	-.0992397	.0302159
_cons	-.9586931	1.055673	-0.91	0.364	-3.027775	1.110389
/c1_2	-.896706	1.400021	-0.64	0.522	-3.640697	1.847285
/c1_3	-1.667291	1.366339	-1.22	0.222	-4.345266	1.010685

```
(car=American is the alternative normalizing location)
(car=Japanese is the alternative normalizing scale)

. matrix b2000 = e(b)

. display mreldif(b704, b2000)
.02582179
```

We put the coefficient vector in b2000 and compared it with the earlier results using the mreldif() function, which computes relative differences between vectors (or between matrices). We see that

there is a maximum relative difference between the coefficients from the two estimations of about 3%.

We now double the number of integration points to 4000 and store the coefficient vector in `b4000`. We omit showing the `cmmprobit` results and show only the comparison of the coefficient vectors:

```
. display mreldif(b2000, b4000)
.00178383
```

The relative difference declined as `intpoints()` is increased. The maximum relative difference between the estimation with 2000 points and the one with 4000 points is only 0.2%.

When we look at the differences between coefficients from different runs, it is important to note the values of the coefficients relative to their standard errors. For example, we may have a variance parameter that is near zero with a big standard error (relative to the parameter estimate). The relative difference of the parameter estimate between runs with different `intpoints()` may not decline rapidly with increasing numbers of points because we are essentially just fitting random noise.

Convergence

Sometimes, you will try to fit a model with one of the CM commands, and the model will not converge. You might see an iteration log that goes on and on with `(backed up)` or `(not concave)` at the end of each line.

In the previous section, we showed you how increasing the number of integration points using the option `intpoints(#)` improves precision of the estimates by reducing the random variance of the Monte-Carlo integration. The randomness of the Monte-Carlo integration can affect convergence in a random way. It is possible that rerunning the command with a different random-number seed (using `set seed #` or the option `intseed(#)`) may cause a model to converge that previously did not. Increasing the number of integration points might cause a model to converge that did not when fewer points were used. It is also possible that a model may converge using the default number of integration points, but no longer converge when more integration points are used.

Our advice is when your model is not converging, first try increasing the number of integration points. If this does not help, try thinking about your model. Perhaps, this should have been the first thing to try. But this might be more painful than setting `intpoints()` to a big number.

Lack of convergence may be trying to tell you something about your model. Perhaps, the model is misspecified. That is, your model is not close to the true data-generating process. Or, perhaps, you simply need to collect more data.

You may want to try simplifying your model. It is best to start with a covariance parameterization with just a few parameters and then gradually increase them. For `cmmprobit`, using `correlation(independent)` and `stddev(heteroskedastic)` is a good model to start with. Look at the variances before trying to parameterize any correlations. Using `correlation(fixed` *matname*`)` lets you specify which elements are fixed and which are estimated. You can also fit models with just one free correlation parameter. `cmroprobit`, which we describe in [CM] **Intro 6**, has the same options and the same advice can be followed.

For the mixed logit models fit by `cmmixlogit` and `cmxtmixlogit`, the covariance parameterization is specified by different options, but the same general advice applies. If you are having convergence problems, start with a simple model and gradually increase the number of covariance parameters estimated.

More than one chosen alternative

What if we have data in which more than one alternative is chosen for some of or all the cases?

Well, first, we need to assess whether the data are in fact rank-ordered alternatives. If so, see [CM] **Intro 6**. There are two CM estimators for rank-ordered alternatives: `cmrologit` and `cmroprobit`.

Second, we need to assess whether the data are perhaps actually panel data and whether the choices were made at different times. For example, we might have data on how people commuted to work on a given week. Some people may have driven a car every day, but some may have driven a car some days and taken the bus on other days. Data such as these are panel data. If we have data by day of the week, we can analyze them as panel data. See [CM] **Intro 7** and example 4 in [CM] **cmclogit**.

But what if the data arose from a design in which multiple choices were allowed and not ranked? For example, suppose consumers were given four breakfast cereals and asked to pick their two favorites, without picking a single most favorite. These data are not rank-ordered data, nor are they panel data.

We note that the random utility model (see [CM] **Intro 8**) for discrete choices yields only one chosen alternative per case: that with the greatest utility. In rank-ordered models, it yields a set of ranked alternatives without any ties. Because the utility function is continuous, ties are theoretically impossible.

Train (2009, sec. 2.2) notes that the set of alternatives can always be made mutually exclusive by considering the choice of two alternatives as a separate alternative. For example, with one or two choices allowed from alternatives A, B, and C, the set of alternatives is A only, B only, C only, A and B, A and C, and B and C, a total of six alternatives. When there are only a few alternatives, this may be an appropriate way to model your data.

References

McFadden, D. L., and K. E. Train. 2000. Mixed MNL models for discrete response. *Journal of Applied Econometrics* 15: 447–470. https://doi.org/10.1002/1099-1255(200009/10)15:5⟨447::AID-JAE570⟩3.0.CO;2-1.

Train, K. E. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. New York: Cambridge University Press.

Also see

[CM] **Intro 1** — Interpretation of choice models

[CM] **Intro 2** — Data layout

[CM] **Intro 3** — Descriptive statistics

[CM] **Intro 4** — Estimation commands

[CM] **cmclogit** — Conditional logit (McFadden's) choice model

[CM] **cmmixlogit** — Mixed logit choice model

[CM] **cmmprobit** — Multinomial probit choice model

[CM] **nlogit** — Nested logit regression

Title

Intro 6 — Models for rank-ordered alternatives

Description Remarks and examples References Also see

Description

This introduction covers the commands `cmroprobit` and `cmrologit`. These estimation commands each fit choice models for rank-ordered alternatives. That is, models in which each decision maker ranks alternatives from a finite set of available alternatives.

Remarks and examples

Remarks are presented under the following headings:

Overview of CM commands for rank-ordered alternatives
cmroprobit: Probit regression for rank-ordered alternatives
Expected choice probabilities (the margins command) after cmroprobit
cmrologit: Logistic regression for rank-ordered alternatives

Overview of CM commands for rank-ordered alternatives

Stata has two commands designed for fitting choice models for rank-ordered alternatives. Below, we give you a brief overview of the models fit by these commands.

`cmroprobit` fits an extension of the multinomial probit choice model for rank-ordered alternatives. It allows both alternative-specific and case-specific predictors. It does not assume IIA; instead, it models the correlation of errors across alternatives. If you are not familiar with IIA, see *Overview of CM commands for discrete choices* in [CM] **Intro 5** and see [CM] **Intro 8**. `cmroprobit` allows tied ranks, but computation time increases with the number of ties, so in practice, it works best when there are only a small number of ties.

`cmrologit` fits a choice model for rank-ordered alternatives for the case in which alternatives are not explicitly identified. That is, there is no variable specifying the alternatives. Alternatives are known only by their characteristics as given by a set of alternative-specific variables. All predictors must be alternative-specific variables. This model assumes IIA is true. It allows tied ranks.

cmroprobit: Probit regression for rank-ordered alternatives

`cmroprobit` is similar to `cmmprobit`. Both are probit regression models, and both likelihoods are computed using simulated integration. Covariance structures are specified in exactly the same manner in each of the commands. Indeed, every option for `cmmprobit` works with `cmroprobit` and does the same thing. The difference is, of course, that `cmroprobit` has ranked alternatives as outcomes and `cmmprobit` requires a single choice of one alternative.

`cmmprobit` is described in the introduction [CM] **Intro 5** and in its manual entry [CM] **cmmprobit**. You should read the discussion of `cmmprobit` in the earlier introduction if you have not already done so.

Just like cmmprobit, cmroprobit models the random-error term of the utility using a multivariate normal distribution and allows the user to specify many different parameterizations for the covariance matrix.

Here is an example using cmroprobit. We use data from the Wisconsin Longitudinal Study cited by Long and Freese (2014, 477). This is a study involving high school graduates who were asked to rank their preferences of four job characteristics: esteem, variety, autonomy, and security.

The case-specific covariates are female (1 if female and 0 if male) and score, a score on a general mental ability test measured in standard deviations. The dataset also includes variables high and low, which indicate whether the respondent's current job is high or low in esteem, variety, autonomy, and security. These two variables together yield three possible ratings for the characteristics of his or her current job—high, low, or neither. From the high and low variables, we create a new variable currentjob that we include as an alternative-specific variable in our model.

The alternatives were ranked (variable rank) such that 1 is the most preferred alternative and 4 is the least, and respondents were allowed to have ties in their rankings. A variable noties indicates those persons who did not have any ties in their rankings.

Here we prepare our data and list them for three respondents:

```
. use https://www.stata-press.com/data/r18/wlsrank
(1992 Wisconsin Longitudinal Study data on job values)
. keep if noties
(11,244 observations deleted)
. generate currentjob = 1 if low==1
(1,304 missing values generated)
. replace currentjob = 2 if low==0 & high==0
(805 real changes made)
. replace currentjob = 3 if high==1
(499 real changes made)
. label define current 1 "Low" 2 "Neither" 3 "High"
. label values currentjob current
. list id jobchar rank female score currentjob in 1/12, sepby(id)
```

	id	jobchar	rank	female	score	curren~b
1.	13	Esteem	4	Male	.3246512	Low
2.	13	Variety	2	Male	.3246512	High
3.	13	Autonomy	1	Male	.3246512	Neither
4.	13	Security	3	Male	.3246512	Low
5.	19	Esteem	3	Female	.0492111	Neither
6.	19	Variety	2	Female	.0492111	Neither
7.	19	Autonomy	4	Female	.0492111	Neither
8.	19	Security	1	Female	.0492111	High
9.	22	Esteem	4	Female	1.426412	High
10.	22	Variety	1	Female	1.426412	Neither
11.	22	Autonomy	2	Female	1.426412	High
12.	22	Security	3	Female	1.426412	Neither

Note that we kept only the data without ties for our analysis. cmroprobit handles ties by evaluating the likelihoods of all possible ways of breaking a tie. For example, suppose someone reported ranks $(1, 1, 3, 4)$, where the two 1s indicate a tie. For this person, cmroprobit computes likelihoods for ranks $(1, 2, 3, 4)$ and $(2, 1, 3, 4)$. If there is a 3-way tie among the ranks, 6 different likelihoods are computed. If there is a 4-way tie among the ranks, 24 different likelihoods are computed. If there

are ties in your data, cmroprobit will be slower than if there were no ties. For this example, we drop the cases with ties in these data, just to make it run faster. Running the example with ties takes about 7 times longer, adjusting for the number of cases omitted (which is substantial because 87% of respondents had ties among their rankings).

Before we can run cmroprobit, we must cmset our data:

```
. cmset id jobchar
     Case ID variable: id
Alternatives variable: jobchar
```

We can now fit our model. We specify the option reverse because by default a bigger rank indicates a preferred alternative, whereas in these data, it is the opposite—a smaller rank indicates a preferred alternative.

```
. cmroprobit rank i.currentjob, casevars(i.female score) reverse structural
note: variable 2.currentjob has 69 cases that are not alternative-specific;
      there is no within-case variability.
note: variable 3.currentjob has 107 cases that are not alternative-specific;
      there is no within-case variability.

Iteration 0:  Log simulated-likelihood = -1102.9667
Iteration 1:  Log simulated-likelihood = -1095.0846  (backed up)
Iteration 2:  Log simulated-likelihood = -1091.2528  (backed up)
Iteration 3:  Log simulated-likelihood = -1086.9763  (backed up)
Iteration 4:  Log simulated-likelihood = -1086.8123  (backed up)
Iteration 5:  Log simulated-likelihood = -1086.3018  (backed up)
Iteration 6:  Log simulated-likelihood = -1086.0779
Iteration 7:  Log simulated-likelihood = -1085.7839
Iteration 8:  Log simulated-likelihood = -1085.4865  (backed up)
Iteration 9:  Log simulated-likelihood = -1084.9575
Iteration 10: Log simulated-likelihood = -1084.0493
Iteration 11: Log simulated-likelihood = -1083.9447
Iteration 12: Log simulated-likelihood = -1083.1608
Iteration 13: Log simulated-likelihood =  -1082.959
Iteration 14: Log simulated-likelihood = -1082.7606
Iteration 15: Log simulated-likelihood = -1081.3702
Iteration 16: Log simulated-likelihood = -1080.8372
Iteration 17: Log simulated-likelihood = -1080.6969
Iteration 18: Log simulated-likelihood = -1080.5508
Iteration 19: Log simulated-likelihood = -1079.9994
Iteration 20: Log simulated-likelihood = -1079.9774
Iteration 21: Log simulated-likelihood = -1079.9741
Iteration 22: Log simulated-likelihood = -1079.9734
Iteration 23: Log simulated-likelihood = -1079.9733
Iteration 24: Log simulated-likelihood = -1079.9733

Reparameterizing to correlation metric and refining estimates

Iteration 0:  Log simulated-likelihood = -1079.9733
Iteration 1:  Log simulated-likelihood = -1079.9733
```

```
Rank-ordered probit choice model              Number of obs      =      1,660
Case ID variable: id                          Number of cases    =        415
Alternatives variable: jobchar                Alts per case: min =          4
                                                             avg =        4.0
                                                             max =          4
Integration sequence:      Hammersley
Integration points:               642         Wald chi2(8)       =      33.92
Log simulated-likelihood = -1079.9733         Prob > chi2        =     0.0000
```

rank	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
jobchar						
currentjob						
Neither	.0694754	.1092521	0.64	0.525	-.1446549	.2836056
High	.44359	.1216771	3.65	0.000	.2051073	.6820727
Esteem	(base alternative)					
Variety						
female						
Female	.1354483	.1843801	0.73	0.463	-.2259301	.4968266
score	.1407149	.0977656	1.44	0.150	-.0509022	.3323319
_cons	1.734451	.1449841	11.96	0.000	1.450288	2.018615
Autonomy						
female						
Female	.2562946	.1645936	1.56	0.119	-.0663029	.5788921
score	.189966	.0873585	2.17	0.030	.0187466	.3611854
_cons	.7007339	.1203077	5.82	0.000	.4649352	.9365326
Security						
female						
Female	.2326753	.2055824	1.13	0.258	-.1702589	.6356095
score	-.1779948	.1101965	-1.62	0.106	-.393976	.0379865
_cons	1.343435	.1598743	8.40	0.000	1.030088	1.656783
/lnsigma3	-.1088538	.1629293	-0.67	0.504	-.4281894	.2104817
/lnsigma4	.3181601	.115562	2.75	0.006	.0916627	.5446575
/atanhr3_2	-.1607581	.2035115	-0.79	0.430	-.5596332	.2381171
/atanhr4_2	-.2718915	.1700903	-1.60	0.110	-.6052624	.0614793
/atanhr4_3	-.3839589	.2473491	-1.55	0.121	-.8687541	.1008364
sigma1	1	(base alternative)				
sigma2	1	(scale alternative)				
sigma3	.8968615	.146125			.651688	1.234273
sigma4	1.374596	.1588511			1.095995	1.724018
rho3_2	-.1593874	.1983414			-.5077053	.2337165
rho4_2	-.265384	.158111			-.5407836	.0614019
rho4_3	-.3661406	.2141897			-.7007406	.100496

```
(jobchar=Esteem is the alternative normalizing location)
(jobchar=Variety is the alternative normalizing scale)
```

We also specified the option `structural` to fit a variance–covariance parameterization based on the full 4×4 variance–covariance matrix. The postestimation commands, `estat covariance` and `estat correlation`, show the estimated covariance and correlations matrices, respectively.

```
. estat covariance
```

	Esteem	Variety	Autonomy	Security
Esteem	1			
Variety	0	1		
Autonomy	0	-.1429484	.8043605	
Security	0	-.3647959	-.4513864	1.889515

```
. estat correlation
```

	Esteem	Variety	Autonomy	Security
Esteem	1.0000			
Variety	0.0000	1.0000		
Autonomy	0.0000	-0.1594	1.0000	
Security	0.0000	-0.2654	-0.3661	1.0000

By default, `cmroprobit` fits a covariance matrix parameterized by differences between alternatives, which yields a 3×3 matrix. See *Covariance structures* in [CM] **cmmprobit** for details.

Expected choice probabilities (the margins command) after cmroprobit

As with the other `cm` estimators, running `margins` afterward can help us understand the model results.

```
. margins
Predictive margins                                      Number of obs = 1,660
Model VCE: OIM
Expression: Pr(jobchar), predict(pr1)
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome						
Esteem	.0228205	.0059232	3.85	0.000	.0112112	.0344297
Variety	.4503611	.0238804	18.86	0.000	.4035564	.4971658
Autonomy	.1461409	.0165112	8.85	0.000	.1137795	.1785022
Security	.3806896	.0232362	16.38	0.000	.3351474	.4262318

Typing `margins` without any options after `cmroprobit` gives the average predicted probability that a particular outcome is the highest-ranked choice. This is indicated on the output by `predict(pr1)`. Also note that the probabilities sum to one. Here variety has the greatest probability (45%) of being the characteristic ranked first. Security has the second greatest probability (38%) of being ranked first.

The above probabilities from `margins` are the expected probabilities of ranking these characteristics as most important based on our model and the characteristics of the individuals in this sample. We can also ask questions about what our model tells would happen if covariates change. For instance, what would we expect these probabilities to be if everyone ranked his or her current job as high in security? To answer this, we use the option `alternative(security)` with `margins` to select the security alternative, and we set it to the `High` ranking by specifying `3.currentjob`.

```
. margins 3.currentjob, alternative(Security)
Predictive margins                                      Number of obs = 1,660
Model VCE: OIM

Expression:  Pr(jobchar), predict(pr1)
Alternative: Security
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome# currentjob						
Esteem#High	.0196645	.0053179	3.70	0.000	.0092416	.0300874
Variety#High	.4184274	.0252003	16.60	0.000	.3690358	.467819
Autonomy # High	.1342234	.0160335	8.37	0.000	.1027982	.1656485
Security # High	.4276949	.02543	16.82	0.000	.3778529	.4775369

Based on this model, we expect that 43% of individuals whose current job is high in security would rank security as being the most important job characteristic.

See [CM] **Intro 1** and [CM] **margins** for details and more examples of running `margins` after `cm` estimators.

cmrologit: Logistic regression for rank-ordered alternatives

`cmrologit` fits a rank-ordered logistic regression model for choice data (Beggs, Cardell, and Hausman 1981). This model is also known as the Plackett–Luce model (Marden 1995), as the exploded logit model (Punj and Staelin 1978), and as the choice-based method of conjoint analysis (Hair et al. 2010).

The model fit by `cmrologit` is unique among the `cm` estimators in that the model does not have explicitly identified alternatives. (`cmmixlogit` and `cmxtmixlogit` can fit models without identified alternatives as well, but for these commands having no alternatives is optional.) Independent variables for `cmrologit` must all be alternative specific; that is, they must vary within case. Any purely case-specific variables will be dropped from the estimation.

Like `cmroprobit`, `cmrologit` allows the ranks to be tied. However, `cmrologit` uses a different computational method, and computation with ties is speedy.

Here is an example using `cmrologit`. We have data on human resource managers ranking their perceived desirability of fictitious applicants. The dataset has 29 cases, each consisting of 10 applicants that are ranked. The variable `caseid` identifies the cases, and we list the observations for `caseid == 7`.

```
. use https://www.stata-press.com/data/r18/evignet, clear
(Vignet study employer prefs (Inge de Wolf 2000))
. list pref female age grades edufit workexp if caseid == 7, noobs
```

pref	female	age	grades	edufit	workexp
0	yes	28	A/B	no	none
0	no	25	C/D	yes	one year
0	no	25	C/D	yes	none
0	yes	25	C/D	no	internship
1	no	25	C/D	yes	one year
2	no	25	A/B	yes	none
3	yes	25	A/B	yes	one year
4	yes	25	A/B	yes	none
5	no	25	A/B	yes	internship
6	yes	28	A/B	yes	one year

The variable `pref` contains the managers' ranks, with 6 indicating the applicant they rank the highest and 0 indicating the four least desirable applicants (tied for last). Predictors for the outcome include `female`, `age`, `grades`, `edufit` (whether the applicant's education fits the job requirements), and `workexp` (work experience). Note that all of these variables represent characteristics of the applicants. There are no variables representing any of the traits of the managers doing the ranking; such variables would be case specific and would be dropped from the model.

As with all `cm` commands, the data must be `cmset`. But here there is no alternatives variable, so we use the `noalternatives` option and only have to specify the case ID variable.

```
. cmset caseid, noalternatives
     Case ID variable: caseid
Alternatives variable: <none>
```

We now fit our model. We include the `baselevels` option to show the base levels of the factor variables to make the output more understandable. (The option `baselevels` can be used with all estimation commands that allow factor variables; see **[U] 11.4.3 Factor variables**.)

```
. cmrologit pref i.female i.age i.grades i.edufit i.workexp, baselevels
Iteration 0:  Log likelihood = -342.28088
Iteration 1:  Log likelihood = -300.81224
Iteration 2:  Log likelihood =  -300.2559
Iteration 3:  Log likelihood =  -300.2549
Iteration 4:  Log likelihood =  -300.2549
Refining estimates:
Iteration 0:  Log likelihood =  -300.2549

Rank-ordered logit choice model                 Number of obs     =        290
Case ID variable: caseid                        Number of cases   =         29

Ties adjustment: exactm                         Obs per case:
                                                              min =         10
                                                              avg =      10.00
                                                              max =         10

                                                LR chi2(7)        =      84.05
Log likelihood = -300.2549                      Prob > chi2       =     0.0000
```

pref	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
female						
no	0	(base)				
yes	-.053564	.1676711	-0.32	0.749	-.3821933	.2750654
age						
22	0	(base)				
25	.2227	.2529694	0.88	0.379	-.2731109	.7185109
28	-.0374488	.2692834	-0.14	0.889	-.5652347	.4903371
grades						
C/D	0	(base)				
A/B	1.07066	.1877038	5.70	0.000	.7027673	1.438553
edufit						
no	0	(base)				
yes	.4591287	.174471	2.63	0.008	.1171718	.8010855
workexp						
none	0	(base)				
internship	.6644342	.2613156	2.54	0.011	.152265	1.176603
one year	1.463883	.245367	5.97	0.000	.9829721	1.944793

`grades`, `edufit`, and `workexp` are all significant predictors of the ranked outcomes.

The maximum likelihood estimates from `cmrologit` are obtained as the maximum partial-likelihood estimates of an appropriately specified Cox regression model for waiting time; see [ST] **stcox**. A higher ranking of an alternative is formally equivalent to a higher hazard rate of failure. A higher stated preference has a shorter waiting time until failure. `cmrologit` uses `stcox` to fit the rank-ordered logit model based on such a specification of the data in Cox terms. See *Methods and formulas* in [CM] **cmrologit**.

References

Beggs, S., S. Cardell, and J. A. Hausman. 1981. Assessing the potential demand for electric cars. *Journal of Econometrics* 17: 1–19. https://doi.org/10.1016/0304-4076(81)90056-7.

Gutiérrez-Vargas, A. A., M. Meulders, and M. Vandebroek. 2021. randregret: A command for fitting random regret minimization models using Stata. *Stata Journal* 21: 626–658.

Hair, J. F., Jr., W. C. Black, B. J. Babin, and R. E. Anderson. 2010. *Multivariate Data Analysis*. 7th ed. Upper Saddle River, NJ: Pearson.

Long, J. S., and J. Freese. 2014. *Regression Models for Categorical Dependent Variables Using Stata*. 3rd ed. College Station, TX: Stata Press.

Marden, J. I. 1995. *Analyzing and Modeling Rank Data*. London: Chapman and Hall.

Punj, G. N., and R. Staelin. 1978. The choice process for graduate business schools. *Journal of Marketing Research* 15: 588–598. https://doi.org/10.1177/002224377801500408.

Also see

[CM] **Intro 1** — Interpretation of choice models

[CM] **Intro 2** — Data layout

[CM] **Intro 3** — Descriptive statistics

[CM] **Intro 4** — Estimation commands

[CM] **cmrologit** — Rank-ordered logit choice model

[CM] **cmroprobit** — Rank-ordered probit choice model

Title

Intro 7 — Models for panel data

Description Remarks and examples Also see

Description

This introduction covers the command `cmxtmixlogit`. This is the only one of the `cm` estimation commands that explicitly models panel data. Other `cm` estimation commands, however, can be used with panel data when run with an appropriate variance estimator, that is, `vce(cluster` *idvar*`)`, `vce(bootstrap, cluster(`*idvar*`))`, or `vce(jackknife, cluster(`*idvar*`))`.

`cmxtmixlogit` fits a mixed logit model to panel choice data. `cmxtmixlogit` models a sequence of choices rather than a single choice, as commands for cross-sectional data do. As with `cmmixlogit`, random coefficients can be fit to model the correlation of choices across alternatives, and the property of independence of irrelevant alternatives (IIA) is not assumed. See *Overview of CM commands for discrete choices* in [CM] **Intro 5**, and see [CM] **Intro 8** if you are not familiar with this assumption.

Remarks and examples

Remarks are presented under the following headings:

Data layout for panel choice data
A cmxtmixlogit model
Time-series operators
Using other cm estimation commands with panel data

Data layout for panel choice data

In panel choice data, decision makers make multiple choices at different times. The data layout is similar to that for cross-sectional data, the difference being that there are repeated cases for each decision maker.

Here is an example of panel choice data. These fictitious data represent individuals' choices of transportation mode at multiple times. We list the data for the first two people:

```
. use https://www.stata-press.com/data/r18/transport
(Transportation choice data)
. list if id <= 2, sepby(t)
```

	id	t	alt	choice	trcost	trtime	age	income	parttime
1.	1	1	Car	1	4.14	0.13	3.0	3	Full-time
2.	1	1	Public	0	4.74	0.42	3.0	3	Full-time
3.	1	1	Bicycle	0	2.76	0.36	3.0	3	Full-time
4.	1	1	Walk	0	0.92	0.13	3.0	3	Full-time
5.	1	2	Car	1	8.00	0.14	3.2	5	Full-time
6.	1	2	Public	0	3.14	0.12	3.2	5	Full-time
7.	1	2	Bicycle	0	2.56	0.18	3.2	5	Full-time
8.	1	2	Walk	0	0.64	0.39	3.2	5	Full-time
9.	1	3	Car	1	1.76	0.18	3.4	5	Part-time
10.	1	3	Public	0	2.25	0.50	3.4	5	Part-time
11.	1	3	Bicycle	0	0.92	1.05	3.4	5	Part-time
12.	1	3	Walk	0	0.58	0.59	3.4	5	Part-time
13.	2	1	Car	0	4.36	0.23	3.0	2	Full-time
14.	2	1	Public	0	4.43	0.43	3.0	2	Full-time
15.	2	1	Bicycle	0	1.25	1.23	3.0	2	Full-time
16.	2	1	Walk	1	0.89	0.12	3.0	2	Full-time
17.	2	2	Car	0	7.14	0.23	3.2	3	Part-time
18.	2	2	Public	1	1.54	0.12	3.2	3	Part-time
19.	2	2	Bicycle	0	2.75	0.95	3.2	3	Part-time
20.	2	2	Walk	0	0.53	1.64	3.2	3	Part-time
21.	2	3	Car	0	6.69	0.17	3.4	2	Full-time
22.	2	3	Public	1	1.32	0.34	3.4	2	Full-time
23.	2	3	Bicycle	0	0.60	0.49	3.4	2	Full-time
24.	2	3	Walk	0	0.68	0.63	3.4	2	Full-time

Individuals (identified by the variable `id`) at each of three time points (time variable `t`) could choose between four modes of transportation (alternatives variable `alt`) with the one chosen alternative indicated by the binary variable `choice`. The first person chose to use a car at all three time points. The second person walked at time = 1 and took public transportation at the other two times.

Cost of travel (`trcost`, measured in $) and travel time (`trtime`, measured in hours) are alternative-specific variables. Variables `age` (measured in decades), `income` (annual income measured in $10,000), and `parttime` (indicating a part-time or full-time job) are case specific.

Before we can fit the model, we must `cmset` the data. For panel data, `cmset` requires three variables: first, the variable identifying individuals (`id`), second, the time variable (`t`), and third, the alternatives variable (`alt`). (`cmxtmixlogit`, like `cmmixlogit`, can fit models without explicitly identified alternatives. In this case, there is no alternatives variable, and the option `noalternatives` is specified.)

```
. cmset id t alt
note: case identifier _caseid generated from id and t.
note: panel by alternatives identifier _panelaltid generated from id and alt.

                    Panel data: Panels id and time t
              Case ID variable: _caseid
         Alternatives variable: alt
Panel by alternatives variable: _panelaltid (strongly balanced)
                 Time variable: t, 1 to 3
                         Delta: 1 unit

Note: Data have been xtset.
```

The notes displayed by `cmset` say it has created two new variables: `_caseid` and `_panelaltid`. Let's list their values for the first two individuals.

```
. list id t alt _caseid _panelaltid if id <= 2, sepby(alt) abbr(11)
```

	id	t	alt	_caseid	_panelaltid
1.	1	1	Car	1	1
2.	1	2	Car	2	1
3.	1	3	Car	3	1
4.	1	1	Public	1	2
5.	1	2	Public	2	2
6.	1	3	Public	3	2
7.	1	1	Bicycle	1	3
8.	1	2	Bicycle	2	3
9.	1	3	Bicycle	3	3
10.	1	1	Walk	1	4
11.	1	2	Walk	2	4
12.	1	3	Walk	3	4
13.	2	1	Car	4	5
14.	2	2	Car	5	5
15.	2	3	Car	6	5
16.	2	1	Public	4	6
17.	2	2	Public	5	6
18.	2	3	Public	6	6
19.	2	1	Bicycle	4	7
20.	2	2	Bicycle	5	7
21.	2	3	Bicycle	6	7
22.	2	1	Walk	4	8
23.	2	2	Walk	5	8
24.	2	3	Walk	6	8

`_caseid` is a variable that identifies cases. For choice model data, remember that a case is a single statistical observation but consists of multiple Stata observations. Each distinct value of panel ID × time represents a single statistical observation, that is, a case. The values of `_caseid` correspond to the distinct values of panel ID × time, in this example the values of `id` × `t`.

`_panelaltid` is a variable that uniquely identifies the distinct values of panel ID × alternative. We will explain why this variable is needed when we show an example with time-series operators. But you can skip over the explanation. These new variables make `cmxtmixlogit` work as you would expect. You need not be concerned about them, just leave them in your dataset.

A cmxtmixlogit model

Continuing with the previous example, we wish to model the effect of travel cost (`trcost`), travel time (`trtime`), income, and age on the choice of transportation mode.

We assume that all individuals have the same preferences with respect to travel cost but that preferences with respect to travel time are heterogeneous, and we model these heterogeneous preferences with a random coefficient for `trtime` by specifying the option `random(trtime)`.

The dependent variable is `choice`, the binary variable indicating which alternative was chosen. The variable `trcost` is included following the dependent variable; placing it in this position means that it should have a fixed coefficient. Specifying `casevars(age income)` includes the case-specific variables `age` and `income` in the model with fixed coefficients.

```
. cmxtmixlogit choice trcost, random(trtime) casevars(age income)
Fitting fixed parameter model:
Fitting full model:
Iteration 0:  Log simulated-likelihood =  -1025.707  (not concave)
Iteration 1:  Log simulated-likelihood = -1014.2513
Iteration 2:  Log simulated-likelihood = -1006.2212
Iteration 3:  Log simulated-likelihood = -1005.9904
Iteration 4:  Log simulated-likelihood = -1005.9899
Iteration 5:  Log simulated-likelihood = -1005.9899
```

```
Mixed logit choice model                        Number of obs        =      6,000
                                                Number of cases      =      1,500
Panel variable: id                              Number of panels     =        500
Time variable: t                                Cases per panel: min =          3
                                                                 avg =        3.0
                                                                 max =          3
Alternatives variable: alt                      Alts per case:   min =          4
                                                                 avg =        4.0
                                                                 max =          4
Integration sequence:      Hammersley
Integration points:               594           Wald chi2(8)         =     432.68
Log simulated-likelihood = -1005.9899           Prob > chi2          =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
alt						
trcost	-.8388216	.0438587	-19.13	0.000	-.9247829	-.7528602
trtime	-1.508756	.2641554	-5.71	0.000	-2.026492	-.9910212
/Normal						
sd(trtime)	1.945596	.2594145			1.498161	2.526661
Car	(base alternative)					
Public						
age	.1538915	.0672638	2.29	0.022	.0220569	.2857261
income	-.3815444	.0347459	-10.98	0.000	-.4496451	-.3134437
_cons	-.5756547	.3515763	-1.64	0.102	-1.264732	.1134222
Bicycle						
age	.20638	.0847655	2.43	0.015	.0402426	.3725174
income	-.5225054	.0463235	-11.28	0.000	-.6132978	-.4317131
_cons	-1.137393	.4461318	-2.55	0.011	-2.011795	-.2629909
Walk						
age	.3097417	.1069941	2.89	0.004	.1000372	.5194463
income	-.9016697	.0686042	-13.14	0.000	-1.036132	-.7672078
_cons	-.4183279	.5607111	-0.75	0.456	-1.517302	.6806458

The coefficients for `trcost` and `trtime` are negative, indicating that as cost and travel time increase, the probability of selecting a method of travel decreases. In the `Public`, `Bicycle`, and `Walk` sections of the output, we see coefficients for the case-specific variables. These are each interpreted relative to the base alternative `Car`. We can use `margins` to more easily interpret the results of this model; see [CM] **Intro 1** and [CM] **margins**.

Because we did not specify a distribution in the `random()` option, we got the default distribution for the random coefficient, which is the normal distribution. Other options for the distribution are available. If we had multiple variables in the `random()` option, we could optionally specify `corrmetric()` to pick the form of the correlation modeled. See [CM] **cmxtmixlogit** for more information on options for random coefficients.

Time-series operators

When you `cmset` panel data with specified alternatives, your data are automatically `xtset`. You can type `xtset` to see the settings:

```
. xtset
Panel variable: _panelaltid (strongly balanced)
 Time variable: t, 1 to 3
         Delta: 1 unit
```

`_panelaltid` becomes the "panel" identifier for viewing the data as `xt` data. This is why `cmxtmixlogit` needs this variable. It is created so you can use Stata's time-series operators (see **[U] 11.4.3.6 Using factor variables with time-series operators**) with `cmxtmixlogit`. For instance, if you want to include lags of alternative-specific variables in your model, the lags must be specific to the alternative, and Stata's time-series lag operator needs to know how to do this.

To illustrate, we add a lag `trtime` to our earlier model. We also specify `correlated` for the random coefficients of `trtime` and its lag so that the distributions of the random coefficients are correlated. Note that because of the additional complexity of this model, it is computationally intensive and may take a few minutes to fit.

```
. cmxtmixlogit choice, random(trtime L.trtime, correlated) casevars(age income)
Fitting fixed parameter model:
Fitting full model:
Iteration 0:  Log simulated-likelihood = -726.49438  (not concave)
Iteration 1:  Log simulated-likelihood = -725.73356
Iteration 2:  Log simulated-likelihood = -724.30029
Iteration 3:  Log simulated-likelihood = -720.40177
Iteration 4:  Log simulated-likelihood = -720.28639
Iteration 5:  Log simulated-likelihood = -720.07741
Iteration 6:  Log simulated-likelihood = -720.07434
Iteration 7:  Log simulated-likelihood = -720.07411
Iteration 8:  Log simulated-likelihood = -720.07411
Refining estimates:
Iteration 0:  Log simulated-likelihood = -720.07411
Iteration 1:  Log simulated-likelihood = -720.07411
```

```
Mixed logit choice model                        Number of obs        =      4,000
                                                Number of cases      =      1,000
Panel variable: id                              Number of panels     =        500
Time variable: t                                Cases per panel: min =          2
                                                                 avg =        2.0
                                                                 max =          2
Alternatives variable: alt                      Alts per case:   min =          4
                                                                 avg =        4.0
                                                                 max =          4
Integration sequence:      Hammersley
Integration points:               625           Wald chi2(8)         =      82.87
Log simulated-likelihood = -720.07411           Prob > chi2          =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
alt						
trtime						
--.	-1.02391	.3884411	-2.64	0.008	-1.785241	-.2625792
L1.	-.7797073	.3843897	-2.03	0.043	-1.533097	-.0263174
/Normal						
sd(trtime)	.8594882	.6233604			.2074386	3.56115
corr(trtime, L.trtime)	.4457922	.6071271	0.73	0.463	-.7639532	.9614328
sd(L.trtime)	1.576005	.4241405			.9299912	2.670768
Car	(base alternative)					
Public						
age	.0848749	.0715193	1.19	0.235	-.0553005	.2250502
income	-.208774	.0336985	-6.20	0.000	-.2748219	-.1427261
_cons	.1079519	.3923718	0.28	0.783	-.6610826	.8769865
Bicycle						
age	.2542854	.1066569	2.38	0.017	.0452418	.4633291
income	-.3155109	.0531635	-5.93	0.000	-.4197094	-.2113123
_cons	-.462521	.5845974	-0.79	0.429	-1.608311	.6832688
Walk						
age	.5830396	.1878859	3.10	0.002	.21479	.9512892
income	-.8183397	.1207108	-6.78	0.000	-1.054929	-.5817508
_cons	.0269189	.9301377	0.03	0.977	-1.796118	1.849955

Including the lag of `trtime` in this model may not have made much conceptual sense, but we did so for the purpose of showing how to use time-series operators with `cmxtmixlogit`.

Using other cm estimation commands with panel data

`cm` estimation commands for cross-sectional data can also be used with panel data. The estimates from these commands have a population-averaged interpretation when used with panel data. The `cmsettings` tell these cross-sectional `cm` commands that the data are panel data. In this case, by default, the `cm` commands report cluster–robust standard errors that account for the within-panel correlation.

Here is what we get if we run a cmclogit model on our previous panel choice data.

```
. cmclogit choice trcost trtime, casevars(age income)
note: data were cmset as panel data, and the default vcetype for panel data is
      vce(cluster id); see cmclogit.

Iteration 0:  Log pseudolikelihood = -1197.9902
Iteration 1:  Log pseudolikelihood = -1035.4817
Iteration 2:  Log pseudolikelihood = -1027.6346
Iteration 3:  Log pseudolikelihood = -1027.6227
Iteration 4:  Log pseudolikelihood = -1027.6227

Conditional logit choice model                  Number of obs      =      6,000
Case ID variable: _caseid                       Number of cases    =       1500

Alternatives variable: alt                      Alts per case: min =          4
                                                               avg =        4.0
                                                               max =          4

                                                   Wald chi2(8)    =     335.13
Log pseudolikelihood = -1027.6227                  Prob > chi2     =     0.0000

                                  (Std. err. adjusted for 500 clusters in id)
```

choice	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
alt						
trcost	-.7667673	.0464592	-16.50	0.000	-.8578258	-.6757089
trtime	-.6572159	.1700226	-3.87	0.000	-.990454	-.3239778
Car	(base alternative)					
Public						
age	.1588594	.0784292	2.03	0.043	.0051409	.3125779
income	-.3479798	.0405743	-8.58	0.000	-.4275039	-.2684557
_cons	-.8253419	.3651235	-2.26	0.024	-1.540971	-.109713
Bicycle						
age	.2025874	.0867382	2.34	0.020	.0325835	.3725912
income	-.4538989	.0436598	-10.40	0.000	-.5394705	-.3683273
_cons	-1.505446	.4571108	-3.29	0.001	-2.401367	-.6095252
Walk						
age	.307546	.1077107	2.86	0.004	.0964369	.518655
income	-.7648748	.0616934	-12.40	0.000	-.8857917	-.6439579
_cons	-.959179	.5054328	-1.90	0.058	-1.949809	.0314511

By default, cmclogit used the variance estimator given by vce(cluster id). If you wish to change the variance estimator, simply set the vce() option to what you want.

Also see

[CM] **Intro 1** — Interpretation of choice models

[CM] **Intro 2** — Data layout

[CM] **Intro 3** — Descriptive statistics

[CM] **Intro 4** — Estimation commands

[CM] **cmclogit** — Conditional logit (McFadden's) choice model

[CM] **cmxtmixlogit** — Panel-data mixed logit choice model

Title

Intro 8 — Random utility models, assumptions, and estimation

Description Remarks and examples References Also see

Description

In this introduction, we introduce you to the random utility model (RUM) formulation under which choice models are typically derived. We also discuss the independence of irrelevant alternatives (IIA) assumption. We tell you which models rely on this assumption and how the other models relax the assumption. Finally, we introduce you to maximum simulated likelihood (MSL) estimation, which is used by many of the choice model commands.

Remarks and examples

Remarks are presented under the following headings:

Random utility models
Alternative-specific variables and case-specific variables
Independence of irrelevant alternatives
Estimators that do not assume IIA
Maximum simulated likelihood

Random utility models

Choice models are typically derived under an assumption of utility-maximizing behavior by the decision maker (Train 2009). Say that the decision makers are enumerated as $i = 1, 2, \ldots, N$, each facing a choice among $a = 1, 2, \ldots, A$ alternatives. The decision makers derive a certain utility (such as a profit or other benefit) from each possible choice. The utility can be expressed as

$$U_{ia} = V_{ia} + \epsilon_{ia} \tag{1}$$

where U_{ia} is the utility of alternative a for the ith decision maker and V_{ia} is the observed component of the utility, typically modeled as a linear function of observed data vectors. The term ϵ_{ia} represents the unobserved components of the utility. The ϵ_{ia} are assumed to have a random distribution, the precise formulation of which depends on the choice model. This general model is called a random utility model.

When there is a discrete choice, the largest U_{ia} among the $a = 1, 2, \ldots, A$ alternatives gives the alternative chosen by the ith decision maker. When there are rank-ordered choices, the order of the U_{ia} corresponds to the ranks assigned by the decision maker.

For a discrete choice model, the probability that the ith decision maker picks alternative a is

$$\begin{aligned} P_{ia} &= \Pr(U_{ia} > U_{ib} \quad \text{for all } b \neq a) \\ &= \Pr(V_{ia} + \epsilon_{ia} > V_{ib} + \epsilon_{ib} \quad \text{for all } b \neq a) \\ &= \Pr(\epsilon_{ia} - \epsilon_{ib} > V_{ib} - V_{ia} \quad \text{for all } b \neq a) \end{aligned}$$

With the distribution of the $\boldsymbol{\epsilon}_i$ given by $f(\boldsymbol{\epsilon}_i)$, this probability can be written as

$$P_{ia} = \int I(\epsilon_{ia} - \epsilon_{ib} > V_{ib} - V_{ia} \quad \text{for all } a \neq b)\, f(\boldsymbol{\epsilon}_i)\, d\boldsymbol{\epsilon}_i \tag{2}$$

where $I(\cdot)$ is the indicator function equal to 1 when the expression inside the parentheses is true and 0 otherwise.

Alternative-specific variables and case-specific variables

Some observed measures are characteristics related to the alternative. For example, if alternatives are different modes of public transportation—bus, subway, or commuter train—one measure might be the cost of a ticket for each alternative. We call these measures alternative specific.

Other observed measures are characteristics of the decision maker alone, for example, his or her age or income. We call these measures case specific. Because case-specific measures may affect different alternatives in different ways, there is a not a single coefficient estimated for each case-specific measure, but rather $A-1$ of them, one for each alternative less one. The coefficients estimate relative differences among alternatives due to the case-specific variable.

In McFadden's choice model, the observed part of the utility is modeled as

$$V_{ia} = \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a + c_a$$

where $\boldsymbol{\alpha}$ are coefficients for $\mathbf{w}_{ia}$, a vector of k alternative-specific variables; $\boldsymbol{\delta}_a$ are coefficients (which vary by alternative) for $\mathbf{z}_i$, a vector of m case-specific variables; and c_a are alternative-specific intercepts.

Only $A-1$ alternative-specific intercepts c_a and $A-1$ coefficients $\boldsymbol{\delta}_a$ for the case-specific variables need to be estimated because only relative differences among the utilities U_{ia} matter. Hence, for a model with k alternative-specific variables and m case-specific variables, a total of $k+m(A-1)+A-1$ coefficients will be estimated: k for the alternative-specific variables, $m(A-1)$ for the case-specific variables, and $A-1$ alternative-specific intercepts.

Independence of irrelevant alternatives

When the relative probabilities of two alternatives in the model do not depend on the characteristics of other alternatives, the model has the IIA property. In terms of the utilities of (1), only models in which errors are independent across alternatives have the IIA property. Thus, whether IIA is plausible hinges on whether the errors are independent over the alternatives or whether they might be correlated.

Stata has estimators for models that have IIA and for models that do not have IIA. For discrete choice models, multinomial logit (`mlogit`) and McFadden's choice model (`cmclogit`) have the IIA property. The mixed logit model (`cmmixlogit` and `cmxtmixlogit`) and the multinomial probit model (`cmmprobit`) allow you to explicitly model the correlations of the errors to fit models that do not have the IIA property. The nested logit model (`nlogit`) models nested alternatives and also does not impose IIA. For rank-ordered outcomes, the rank-ordered logit model (`cmrologit`) imposes IIA, and the rank-ordered probit model (`cmroprobit`) can fit models that do not impose IIA.

Let's illustrate IIA with an example. Consider a commuter who has the choice of either using a car or walking to get to work. A choice model gives the probabilities P_{car} and P_{walk}, the probabilities for the choices car and walking. Now suppose that a new bus line opens, and the commuter can now take a bus to work. Now the choice model has P_{bus} as well, the probability of taking a bus. Because the probabilities must sum to one, $P_{\text{car}} + P_{\text{walk}}$ must be smaller now. But what about their ratio $P_{\text{car}}/P_{\text{walk}}$? Should that change?

The property of IIA implies that the ratio $P_{\text{car}}/P_{\text{walk}}$ does not change when the new alternative of taking a bus becomes available. Say $P_{\text{car}}/P_{\text{walk}} = 2$ before the availability of taking a bus. So the model says a car is taken with probability $2/3$ and walking with probability $1/3$.

Now a bus is available, say, with $P_{\text{bus}} = 0.25$. IIA assumes that the existence of the choice of a bus does not change the relative appeal of taking a car over walking. So $P_{\text{car}}/P_{\text{walk}} = 2$ is still true, and now $P_{\text{car}} = 0.5$ and $P_{\text{walk}} = 0.25$. Discussions of whether these new predicted probabilities are valid hinge on whether the assumed independence is realistic.

Models that have the IIA property have the property because of the functional formulation of the model. Consider McFadden's choice model (McFadden 1974), which has IIA and is fit by the command `cmclogit` using conditional logistic regression. The probabilities in McFadden's choice model are given by

$$P_{ia} = \frac{e^{V_{ia}}}{\sum_{j=1}^{A} e^{V_{ij}}} \tag{3}$$

See *Methods and formulas* in [CM] **cmclogit** and *Methods and formulas* in [R] **clogit**.

The ratio for the probability of alternative a to the probability for alternative b is

$$\frac{P_{ia}}{P_{ib}} = \frac{e^{V_{ia}}}{e^{V_{ib}}} \tag{4}$$

The ratio is independent of the probabilities of any of the other alternatives, and hence, the McFadden's choice model satisfies IIA.

Regardless of whether the true model for our data has IIA, if we fit a McFadden's choice model to our data, the probabilities from the model will satisfy IIA. IIA is a mathematical consequence of the formulation of McFadden's choice model.

If the errors in (1) are correlated, the choice probabilities do not have the form of (3), and the ratio in (4) depends on the characteristics of other alternatives. If you do not want to assume independence of errors across alternatives at the outset, you can fit a model that estimates correlation parameters and test these parameters.

Estimators that do not assume IIA

The CM estimators for discrete choice models that do not assume IIA are `cmmprobit`, `cmmixlogit`, and its extension to panel data, `cmxtmixlogit`. For rank-ordered alternatives, `cmroprobit` fits models that do not assume IIA.

As described earlier, for McFadden's choice model, the utility is modeled as

$$\begin{aligned} U_{ia} &= V_{ia} + \epsilon_{ia} \\ &= \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a + c_a + \epsilon_{ia} \end{aligned} \tag{5}$$

where $\boldsymbol{\alpha}$ are coefficients for the alternative-specific variables $\mathbf{w}_{ia}$; $\boldsymbol{\delta}_a$ are coefficients for the case-specific variables $\mathbf{z}_i$; c_a are intercepts; and ϵ_{ia} are unobserved random variables, modeled as independent type I (Gumbel-type) extreme-value random variables.

In the mixed logit model fit by `cmmixlogit`, the utility is

$$\begin{aligned} U_{ia} &= V_{ia} + \epsilon_{ia} \\ &= \mathbf{x}_{ia}\boldsymbol{\beta}_i + \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a + c_a + \epsilon_{ia} \end{aligned} \tag{6}$$

where $\boldsymbol{\beta}_i$ are random coefficients that vary over individuals in the population and $\mathbf{x}_{ia}$ is a vector of alternative-specific variables. The other terms are as in McFadden's choice model.

The $\boldsymbol{\beta}_i$ are not directly estimated. They are assumed to have a particular distribution, and the parameters of the distribution are estimated. For example, if the $\boldsymbol{\beta}_i$ are assumed to have a multivariate normal distribution, $\boldsymbol{\beta}_i \sim N(\mu, \boldsymbol{\Sigma})$, then the mixed logit model estimates μ and $\boldsymbol{\Sigma}$.

The probability of the ith individual picking alternative a is derived from an integral over the distribution of $\boldsymbol{\beta}$, $f(\boldsymbol{\beta})$,

$$P_{ia} = \int P_{ia}(\boldsymbol{\beta}) f(\boldsymbol{\beta}) d\boldsymbol{\beta} \tag{7}$$

where

$$P_{ia}(\boldsymbol{\beta}) = \frac{e^{V_{ia}}}{\sum_{b=1}^{A} e^{V_{ib}}} \tag{8}$$

Although (8) looks like (3), it is (7) that gives the probabilities P_{ia}, and IIA is not satisfied because the random parameters cause the errors in (1) to be correlated. This correlation means that nothing similar to (4) is true.

In a mixed logit model, one or more alternative-specific variables are selected to have random coefficients. When the random coefficients are modeled with a nonzero variance, IIA is not true. When the variance is zero, IIA is true. Testing whether variance parameters are zero gives a model-based test of IIA.

The multinomial probit model fit by `cmmprobit` also does not assume IIA. Its utility is formulated as

$$U_{ia} = \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a + c_a + \xi_{ia} \tag{9}$$

where the random-error term $\boldsymbol{\xi}_i = (\xi_{i1}, \xi_{i2}, \ldots, \xi_{iA})$ is distributed multivariate normal with mean zero and covariance matrix $\boldsymbol{\Sigma}$. The other terms in the model are the same as they are in McFadden's choice model. Probabilities P_{ia} are computed using (2).

The ratio P_{ia}/P_{ib} is not independent of the other P_{ic} when the covariance matrix $\boldsymbol{\Sigma}$ is specified with nonzero correlation parameters for the alternatives. If, say, the errors for alternative b and c are correlated, a change in P_{ic} will cause a change in P_{ia}/P_{ib}. So IIA is not satisfied in this case.

`cmmixlogit` and `cmmprobit` are described in the introduction [CM] **Intro 5**. `cmroprobit` is covered in [CM] **Intro 6**, and `cmxtmixlogit` in [CM] **Intro 7**.

Maximum simulated likelihood

The integral in (2), for `cmmixlogit`, `cmxtmixlogit`, `cmmprobit`, and `cmroprobit` models, must be approximated because it has no closed-form solution. The integral is computed by simulation, and the estimation is said to be done using MSL.

Consistency of the MSL estimator requires that the number of points in the simulation be sufficiently large. More points will produce more precise estimates by reducing approximation error, at the cost of increased computation time. This should be kept in mind when fitting these models. Practically speaking, it means that when you have found a model that you consider final, you should increase the number of integration points. It also means that if your model is having a hard time converging, the first thing you should try to get the model to converge is increasing the number of integration points.

See *Setting the number of integration points* in [CM] **Intro 5** for examples and advice about setting the number of integration points.

See *Methods and formulas* in `cmmixlogit` and *Methods and formulas* in `cmmprobit` for further statistical details, and see Cameron and Trivedi (2005) for an introduction to MSL estimation.

References

Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.

McFadden, D. L. 1974. Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics*, ed. P. Zarembka, 105–142. New York: Academic Press.

Train, K. E. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. New York: Cambridge University Press.

Also see

[CM] **Intro 5** — Models for discrete choices

Title

cmchoiceset — Tabulate choice sets

Description Quick start Menu Syntax
Options Remarks and examples Stored results Also see

Description

`cmchoiceset` tabulates choice sets for choice data. It is useful when choice sets are unbalanced, that is, when alternatives are not the same for every case.

Quick start

One-way tabulation of choice sets for `cmset` data

```
cmchoiceset
```

Same as above, but omit missing values of the alternatives variable observation by observation rather than casewise (the default)

```
cmchoiceset, altwise
```

One-way tabulation of the size of the choice sets

```
cmchoiceset, size
```

Two-way tabulation of choice sets versus the case-specific variable `x`

```
cmchoiceset x
```

Same as above, but `x` is not a case-specific variable, and tabulation will be by observations, not cases

```
cmchoiceset x, observations
```

Generate a variable with categories for the choice-set patterns

```
cmchoiceset, generate(cvar)
```

For panel choice data, display a two-way tabulation of choice sets versus the time variable

```
cmchoiceset, time
```

Menu

Statistics > Choice models > Setup and utilities > Tabulate choice sets

Syntax

```
cmchoiceset [varname] [if] [in] [, options]
```

options	Description
Main	
`size`	tabulate size of choice sets
`observations`	tabulate by observations, not cases; the default
`altwise`	use alternativewise deletion instead of casewise deletion
`transpose`	transpose rows and columns in two-way tables
`missing`	include missing values of *varname* in tabulation
`time`	tabulate choice sets versus time variable (only for panel CM data)
`generate(`*newvar*`, ...)`	create new variable containing categories for the choice-set patterns
Options	
tab1_options	options for one-way tables
tab2_options	options for two-way tables

tab1_options	Description
`sort`	display table in descending order of frequency

tab2_options	Description
`column`	report column percentages
`row`	report row percentages
`cell`	report cell percentages
`rowsort`	list rows in order of observed frequency
`colsort`	list columns in order of observed frequency
`[no]key`	report or suppress cell contents key

You must `cmset` your data before using `cmchoiceset`; see [CM] **cmset**.
`by` is allowed; see [D] **by**.

Options

Main

`size` tabulates the size of the choice sets rather than the choice-set patterns.

`observations` specifies that the tabulation be done by observations instead of by cases, which is the default. If *varname* is specified and *varname* is a case-specific variable (values constant within case), a tabulation of choice sets versus *varname* by cases is displayed by default. If *varname* is not a case-specific variable, a tabulation by cases cannot be produced, so the option `observations` must be specified; otherwise, an error message is given.

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in the alternatives variable or *varname*. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered.

This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

`transpose` transposes rows and columns in displays of two-way tables.

`missing` specifies that the missing values of *varname* be treated like any other value of *varname*.

`time` tabulates choice sets versus the time variable when data are panel choice data. See [CM] **cmset**.

`generate(`*newvar* [`, replace label(`*lblname*`)`]`)` creates a new variable containing categories for the choice-set patterns. The variable *newvar* is numeric and valued 1, 2, ... Its value label contains the choice-set patterns as strings. If option `size` was specified, then *newvar* contains the sizes of the choice sets.

`replace` allows any existing variable named *newvar* to be replaced.

`label(`*lblname*`)` specifies the name of the value label created when `generate(`*newvar*`)` is specified. By default, the variable name *newvar* is also used for the name of the value label.

Options

`sort` puts the table in descending order of frequency in a one-way table.

`column` displays the relative frequency, as a percentage, of each cell within its column in a two-way table.

`row` displays the relative frequency, as a percentage, of each cell within its row in a two-way table.

`cell` displays the relative frequency, as a percentage, of each cell in a two-way table.

`rowsort` and `colsort` specify that the rows and columns, respectively, be presented in order of observed frequency in a two-way table.

[`no`]`key` displays or suppresses a key above two-way tables. The default is to display the key if more than one cell statistic is requested. `key` displays the key. `nokey` suppresses its display.

Remarks and examples

`cmchoiceset` is useful when choice sets are unbalanced, meaning different cases have different sets of alternatives. For balanced choice sets—when every case has the same set of alternatives—this command merely tells you every choice set is the same.

In particular, `cmchoiceset, generate(`*newvar*`)` can be useful when using the postestimation command `margins` for unbalanced designs. The variable *newvar* can be used with `margins`'s options `over()` or `subpop()`. This allows you to look at adjusted predictions, expected probabilities, and marginal effects grouped by the different choice sets. See example 3 below and [CM] **margins** for details.

Example 1: Cross-sectional choice data, one-way tables

Here is an example with cross-sectional choice data. First, we `cmset` our data. The variable `consumerid` is our case ID, and the variable `car` gives the alternatives.

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.
     Case ID variable: consumerid
Alternatives variable: car
```

`cmset` tells us the choice sets are unbalanced. To see the choice sets, we type `cmchoiceset`:

```
. cmchoiceset
Tabulation of choice-set possibilities
 Choice set |      Freq.     Percent        Cum.
------------+-----------------------------------
      1 2 3 |        380       42.94       42.94
    1 2 3 4 |        505       57.06      100.00
------------+-----------------------------------
      Total |        885      100.00
Note: Total is number of cases.
```

The majority of choice sets are 1, 2, 3, 4, and the remaining ones are 1, 2, 3—missing alternative 4.

To see the correspondence between numeric values of alternatives and their labels, we list the value label of the alternatives variable `car`.

```
. describe car
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
car             byte    %9.0g      nation     Nationality of car
. label list nation
nation:
           1 American
           2 Japanese
           3 European
           4 Korean
```

We see that alternative 4 is Korean automobiles. This is the alternative that some consumers do not have.

To get a tabulation by observations rather than by cases, we use the `observations` option.

```
. cmchoiceset, observations
Tabulation of choice-set possibilities
 Choice set |      Freq.     Percent        Cum.
------------+-----------------------------------
      1 2 3 |      1,140       36.08       36.08
    1 2 3 4 |      2,020       63.92      100.00
------------+-----------------------------------
      Total |      3,160      100.00
Note: Total is number of observations.
```

◁

Example 2: Cross-sectional choice data, two-way tables

If you suspect that there is a relationship between the choice set and some variable in your dataset, you can examine a two-way tabulation. Here we tabulate the choice sets versus `gender`, which is a case-specific variable, meaning that it is constant within each case.

```
. cmchoiceset gender
Tabulation of choice-set possibilities by gender
```

Choice set	Gender: 0 = Female, 1 = Male Female	Male	Total
1 2 3	102	271	373
1 2 3 4	134	355	489
Total	236	626	862

```
Note: Total is number of cases.
```

We notice that this tabulation has only 862 cases, whereas the earlier one had 885 cases. The variable `gender` must have missing values. Are the observations with missing values related to the choice sets? We can look at this by specifying the options `missing` and `observations`.

```
. cmchoiceset gender, missing observations
Tabulation of choice-set possibilities by gender
```

Choice set	Gender: 0 = Female, 1 = Male Female	Male	.	Total
1 2 3	306	827	7	1,140
1 2 3 4	548	1,456	16	2,020
Total	854	2,283	23	3,160

```
Note: Total is number of observations.
```

Note that we did this tabulation by observations, not cases. If we omit the option `observations`, we get an error message:

```
. cmchoiceset gender, missing
casevar not constant within case
    Casevar gender is not constant within case for 23 cases (85 obs).
    Use option observations when gender is not a casevar.
r(459);
```

By default, `cmchoiceset` considers any *varname* passed as an argument to be a case-specific variable. The variable `gender` is a case-specific variable when cases with any missing values are omitted. But if you treat missing values like any other value, then `gender` is not a case-specific variable because when there are missing values, the missing values are not found in every observation of the case.

If you want to examine missing values in choice data, you may find the `cmsample` command useful.

The `altwise` option handles missing values differently. This option omits observations with missing values for *varname* (or the alternatives variable) and then creates choice sets based on the remaining observations.

```
. cmchoiceset gender, altwise
Tabulation of choice-set possibilities by gender
```

Choice set	Gender: 0 = Female, 1 = Male Female	Male	Total
1 2	0	2	2
1 2 3	104	274	378
1 2 3 4	134	355	489
1 2 4	0	4	4
1 3	0	2	2
1 3 4	1	1	2
2 3	0	3	3
2 3 4	1	4	5
Total	240	645	885

```
Note: Total is number of cases.
```

Using `altwise` with these data creates several additional choice sets. When we use a `cm` estimator with the option `altwise` and have variables with missing values, the same thing can happen. Here is an example:

```
. cmclogit purchase, casevar(i.gender) altwise
  (output omitted)
. cmchoiceset if e(sample) == 1
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2	2	0.23	0.23
1 2 3	378	42.71	42.94
1 2 3 4	489	55.25	98.19
1 2 4	4	0.45	98.64
1 3	2	0.23	98.87
1 3 4	2	0.23	99.10
2 3	3	0.34	99.44
2 3 4	5	0.56	100.00
Total	885	100.00	

```
Note: Total is number of cases.
```

The `altwise` option with the estimator `cmclogit` and the *casevar* `gender` creates the same choice sets as the `altwise` option does with `cmchoiceset gender`. Before using the option `altwise` with a `cm` estimator, we may want to think whether it is appropriate. In this example, it means treating 2 cases as if their only available alternatives were 1 or 2, 4 cases as if their only available alternatives were 1, 2, or 4, etc.

When doing a tabulation of choice sets versus a variable with many values, the option `transpose` is helpful.

```
. cmchoiceset dealers, observations missing transpose
Tabulation of dealers by choice-set possibilities
```

No. of dealerships in community	Choice set 1 2 3	Choice set 1 2 3 4	Total
0	0	2	2
1	28	372	400
2	135	247	382
3	155	273	428
4	107	186	293
5	121	157	278
6	132	141	273
7	84	145	229
8	107	136	243
9	113	150	263
10	90	134	224
11	45	53	98
12	17	16	33
13	6	8	14
Total	1,140	2,020	3,160

```
Note: Total is number of observations.
```

It creates a long display rather than a wide display in this instance.

◁

▷ Example 3: The generate() option

The option `generate()` can be used to create a variable containing the categories of choice-set patterns. Here we use it after running `cmclogit`.

```
. cmclogit purchase dealers, casevar(i.gender income)
  (output omitted)
. cmchoiceset if e(sample) == 1, generate(choiceset)
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2 3	373	43.27	43.27
1 2 3 4	489	56.73	100.00
Total	862	100.00	

```
Note: Total is number of cases.
. describe choiceset
```

Variable name	Storage type	Display format	Value label	Variable label
choiceset	byte	%8.0g	choiceset	Choice set

```
. label list choiceset
choiceset:
           1 1 2 3
           2 1 2 3 4
```

Note that we specified `if e(sample) == 1` with `cmchoiceset` so that the sample used for `cmchoiceset` is the same as the estimation sample used by `cmclogit`.

generate() creates a variable with values 1 and 2. Its value label contains the strings "1 2 3" and "1 2 3 4", which make the output understandable.

If we want to look at average predicted probabilities for the alternatives separately for the two different choice sets, we can use the newly created variable choiceset with the over() option in margins.

```
. margins, over(choiceset)
Predictive margins                                      Number of obs = 3,075
Model VCE: OIM
Expression: Pr(car|1 selected), predict()
Over:       choiceset
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome# choiceset						
American # 1 2 3	.4610168	.0172363	26.75	0.000	.4272343	.4947992
American # 1 2 3 4	.4172612	.0169068	24.68	0.000	.3841246	.4503979
Japanese # 1 2 3	.3840219	.0168052	22.85	0.000	.3510843	.4169596
Japanese # 1 2 3 4	.3532921	.0161223	21.91	0.000	.3216929	.3848913
European # 1 2 3	.1549613	.0123628	12.53	0.000	.1307307	.1791919
European # 1 2 3 4	.1476471	.0117985	12.51	0.000	.1245225	.1707718
Korean # 1 2 3	.	(not estimable)				
Korean # 1 2 3 4	.0817996	.0122163	6.70	0.000	.0578562	.105743

In particular, looking at predicted probabilities and marginal effects by choice sets is often useful for intentionally unbalanced designs. See [CM] **margins** for a more lengthy discussion.

◁

▷ Example 4: Panel choice data

When you have panel choice data, cmchoiceset is useful to see how choice sets vary by time—if they do vary by time. Here is an example with an unbalanced dataset.

```
. use https://www.stata-press.com/data/r18/transport_unbalanced, clear
(Transportation choice data with unbalanced choice sets)
. cmset id t alt
note: case identifier _caseid generated from id and t.
note: panel by alternatives identifier _panelaltid generated from id and alt.
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.
                    Panel data: Panels id and time t
            Case ID variable: _caseid
       Alternatives variable: alt
Panel by alternatives variable: _panelaltid (unbalanced)
                Time variable: t, 1 to 3, but with gaps
                        Delta: 1 unit
Note: Data have been xtset.
```

The output from `cmset` is telling us the data are unbalanced. Do the choice sets vary by time? `cmchoiceset` with the option `time` will answer this question.

```
. cmchoiceset, time
Tabulation of choice-set possibilities by time t
```

Choice set	Time variable 1	2	3	Total
1 2	0	1	0	1
1 2 3	5	3	0	8
1 2 3 4	483	482	500	1,465
1 2 4	6	3	0	9
1 3 4	2	3	0	5
2 3 4	4	8	0	12
Total	500	500	500	1,500

```
Note: Total is number of cases.
```

The choice sets at time `t` = 3 are balanced but are unbalanced at each of the other times.

If there were many time periods and only a few choice sets, the option `transpose` would make a more readable tabulation.

◁

Stored results

`cmchoiceset` stores the following in `r()`:

Scalars

`r(N)`	number of observations
`r(r)`	number of rows
`r(c)`	number of columns

Also see

[CM] **cmsample** — Display reasons for sample exclusion

[CM] **cmset** — Declare data to be choice model data

[CM] **cmsummarize** — Summarize variables by chosen alternatives

[CM] **cmtab** — Tabulate chosen alternatives

Title

cmclogit — Conditional logit (McFadden's) choice model

Description Quick start Menu Syntax
Options Remarks and examples Stored results Methods and formulas
References Also see

Description

`cmclogit` fits McFadden's choice model, which is a specific case of the more general conditional logistic regression model fit by `clogit`.

The command requires multiple observations for each case (representing one individual or decision maker), where each observation represents an alternative that may be chosen. `cmclogit` allows two types of independent variables: alternative-specific variables, which vary across both cases and alternatives, and case-specific variables, which vary across only cases.

Quick start

McFadden's choice model of `y` on alternative-specific variable `x1` using `cmset` data

```
cmclogit y x1
```

Same as above, and add indicators for levels of `x2`, which are constant within case

```
cmclogit y x1, casevars(i.x2)
```

Same as above, but omit alternative-specific intercepts

```
cmclogit y x1, casevars(i.x2) noconstant
```

Include only case-specific covariates (equivalent to `mlogit` when data are balanced)

```
cmclogit y, casevars(i.x2 x3)
```

Menu

Statistics > Choice models > Conditional logit (McFadden's choice) model

Syntax

```
cmclogit depvar [indepvars] [if] [in] [weight] [, options]
```

depvar equal to 1 identifies the chosen alternatives, whereas a 0 indicates the alternatives that were not chosen. There can be only one chosen alternative for each case.

options	Description
Model	
`casevars(`*varlist*`)`	case-specific variables
`basealternative(`*# | lbl | str*`)`	set base alternative
`noconstant`	suppress alternative-specific constant terms
`altwise`	use alternativewise deletion instead of casewise deletion
`offset(`*varname*`)`	include *varname* in model with coefficient constrained to 1
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `robust`, `cluster` *clustvar*, `bootstrap`, or `jackknife`
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`or`	report odds ratios and relative-risk ratios
`noheader`	do not display the header on the coefficient table
`nocnsreport`	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
`coeflegend`	display legend instead of statistics

You must `cmset` your data before using `cmclogit`; see [CM] **cmset**.

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

`bootstrap`, `by`, `collect`, `fp`, `jackknife`, and `statsby` are allowed; see [U] **11.1.10 Prefix commands**.

Weights are not allowed with the `bootstrap` prefix; see [R] **bootstrap**.

`fweights`, `iweights`, and `pweights` are allowed (see [U] **11.1.6 weight**), but they are interpreted to apply to cases as a whole, not to individual observations. See *Use of weights* in [R] **clogit**.

`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

`casevars(`*varlist*`)` specifies the case-specific numeric variables. These are variables that are constant for each case. If there are a maximum of J alternatives, there will be $J - 1$ sets of coefficients associated with each *casevar*.

`basealternative(#|lbl|str)` sets the alternative that normalizes the level of utility. The base alternative may be specified as a number when the alternatives variable is numeric, as a label when it is numeric and has a value label, or as a string when it is a string variable. The default is the alternative with the highest frequency of being chosen. This option is ignored if neither alternative-specific constants nor case-specific variables are specified.

If `vce(bootstrap)` or `vce(jackknife)` is specified, you must specify the base alternative. This is to ensure that the same model is fit with each call to `cmclogit`.

`noconstant` suppresses the $J-1$ alternative-specific constant terms.

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

`offset(`*varname*`)`, `constraints(`*numlist*|*matname*`)`; 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`), 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**.

`or` reports the estimated coefficients transformed to odds ratios for alternative-specific variables and relative-risk ratios for case-specific variables. That is, e^b rather than b is reported. Standard errors and confidence intervals are transformed accordingly. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`noheader` prevents the coefficient table header from being 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.

`technique(bhhh)` is not allowed.

The initial estimates must be specified as `from(`*matname* `[, copy])`, where *matname* is the matrix containing the initial estimates and the `copy` option specifies that only the position of each element in *matname* is relevant. If `copy` is not specified, the column stripe of *matname* identifies the estimates.

The following options are available with cmclogit but are not shown in the dialog box:

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

Remarks and examples

cmclogit fits McFadden's choice model (McFadden 1974). For a brief introduction, see Greene (2018, sec. 18.2) or Cameron and Trivedi (2022, sec. 18.5).

cmclogit requires data in long form. For each individual (or decision maker), there are multiple Stata observations, one for each of the alternatives the individual could have chosen. We call the group of Stata observations for an individual a "case". Each case represents a single statistical observation although it comprises multiple Stata observations. See [CM] **Intro 2**.

Independent variables for McFadden's choice model come in two forms: alternative specific and case specific. Alternative-specific variables vary across cases and within cases by alternative. Case-specific variables are constant within cases.

We index the set of unordered alternatives by $1, 2, \ldots, J$. Let y_{ij}, $j = 1, \ldots, J$, be an indicator variable for the alternative chosen by the ith individual (case). That is, $y_{ij} = 1$ if individual i chose alternative j, and $y_{ij} = 0$ otherwise.

The sets of possible alternatives across individuals, also known as choice sets, can be unbalanced. That is, choice sets can vary by case. For example, individual 1 could have choice set $\{1, 2, 3\}$, and individual 2 could have choice set $\{1, 2, 4\}$. For individual 1, the 4th alternative was unavailable to be chosen, and for individual 2, the 3rd alternative was unavailable. We take $1, 2, \ldots, J$ to represent all possible alternatives taken across all individuals.

Assume that we have p alternative-specific variables so that for case i we have a $J \times p$ data matrix $\mathbf{X}_i$. Further, assume that we have q case-specific variables so that we also have a $1 \times q$ data vector $\mathbf{z}_i$ for case i. Our random utility model can be expressed as

$$\mathbf{u}_i = \mathbf{X}_i\boldsymbol{\beta} + (\mathbf{z}_i\mathbf{A})' + \boldsymbol{\epsilon}_i$$

where $\mathbf{u}_i$ is the utility for case i, $\boldsymbol{\beta}$ is a $p \times 1$ vector of alternative-specific regression coefficients, and $\mathbf{A} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_J)$ is a $q \times J$ matrix of case-specific regression coefficients. The elements of the $J \times 1$ vector $\boldsymbol{\epsilon}_i$ are independent type I (Gumbel-type) extreme-value random variables with mean γ (the Euler–Mascheroni constant, approximately 0.577) and variance $\pi^2/6$.

We must fix one of the $\boldsymbol{\alpha}_j$ to be the zero vector to normalize the location. We set $\boldsymbol{\alpha}_k = \mathbf{0}$, where k is specified by the basealternative() option. The vector $\mathbf{u}_i$ quantifies the utility that the individual gains from the J alternatives. If alternative j is unavailable to the ith individual, we omit the jth row in $\mathbf{u}_i$. The alternative chosen by individual i is the one that maximizes utility.

McFadden's choice model is a specific instance of conditional logistic regression. You can use clogit to obtain the same estimates as cmclogit by specifying the case ID variable used to cmset your data as the group() variable in clogit. Your case-specific variables in casevars() must be interacted with each alternative, excluding the interaction associated with the base alternative. The alternatives variable used to produce this interaction is the alternatives variable used with cmset. These interactions are included in the clogit estimation along with the alternative-specific variables. cmclogit does this for you. See *Duplicating cmclogit using clogit* in [CM] **Intro 5** for an example that uses clogit to reproduce the results from cmclogit.

Before you can fit McFadden's choice model using cmclogit, you must first cmset your data to specify which variables in your dataset identify the cases and the alternatives; see [CM] **cmset** for information on this command.

Example 1: Consumer car choice data

We have fictitious data on 885 consumers and their choice of automobile. Each consumer chose among an American, Japanese, European, or Korean car (variable `car`). We want to explore the relationship between the choice of car and the consumer's gender (variable `gender`) and income (variable `income` in thousands of dollars). We also have the number of dealerships of each nationality in the consumer's community (variable `dealers`), which we want to include as a regressor.

The variable `dealers` is an alternative-specific variable, and `gender` and `income` are case-specific variables. Each consumer's chosen car is indicated by the variable `purchase`, a 0/1 variable.

Let's list some of the data.

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. list consumerid car purchase dealers gender income
> if consumerid <= 4, sepby(consumerid) abbr(10)
```

	consumerid	car	purchase	dealers	gender	income
1.	1	American	1	9	Male	46.7
2.	1	Japanese	0	11	Male	46.7
3.	1	European	0	5	Male	46.7
4.	1	Korean	0	1	Male	46.7
5.	2	American	1	10	Male	26.1
6.	2	Japanese	0	7	Male	26.1
7.	2	European	0	2	Male	26.1
8.	2	Korean	0	1	Male	26.1
9.	3	American	0	8	Male	32.7
10.	3	Japanese	1	6	Male	32.7
11.	3	European	0	2	Male	32.7
12.	4	American	1	5	Female	49.2
13.	4	Japanese	0	4	Female	49.2
14.	4	European	0	3	Female	49.2

We see that the first consumer, a male earning $46,700 per year, chose to purchase an American car. The third consumer purchased a Japanese car. The third and fourth consumers do not have the choice of a Korean car as a possible alternative because there are no Korean automobile dealerships in their communities.

Before we can run a `cm` estimation command, we must `cmset` our data. The first argument to `cmset` is the case ID variable, which must be numeric. For these data, it is the variable `consumerid`, which identifies individual consumers. The alternatives variable identifies the alternatives that could have been chosen. In this instance, it is the variable `car`, which gives the nationality of car, American, Japanese, European, or Korean.

```
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of different
      sizes found.
     Case ID variable: consumerid
Alternatives variable: car
```

The message from `cmset` tells us that the choice sets for these data are unbalanced (which we already knew from the data listing). The `cmchoiceset` command will display the choice sets:

```
. cmchoiceset
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2 3	380	42.94	42.94
1 2 3 4	505	57.06	100.00
Total	885	100.00	

```
Note: Total is number of cases.
```

The numeric variable **car** is labeled so that 1 = American, 2 = Japanese, 3 = European, and 4 = Korean. We see that 43% of the consumers do not have a Korean car dealership in their communities, and this alternative is not considered available to them. All consumers in these data have American, Japanese, and European dealerships locally, and everyone has these alternatives in their choice sets.

We now fit our model.

```
. cmclogit purchase dealers, casevars(i.gender income)
Iteration 0:  Log likelihood = -959.21405
Iteration 1:  Log likelihood = -948.48587
Iteration 2:  Log likelihood =  -948.1217
Iteration 3:  Log likelihood = -948.12096
Iteration 4:  Log likelihood = -948.12096

Conditional logit choice model                  Number of obs      =      3,075
Case ID variable: consumerid                    Number of cases    =        862

Alternatives variable: car                      Alts per case: min =          3
                                                               avg =        3.6
                                                               max =          4

                                                  Wald chi2(7)     =      51.03
Log likelihood = -948.12096                       Prob > chi2      =     0.0000
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	.0448082	.0262818	1.70	0.088	-.0067032	.0963196
American	(base alternative)					
Japanese						
gender						
Male	-.379326	.1712399	-2.22	0.027	-.71495	-.0437021
income	.0154978	.0065145	2.38	0.017	.0027296	.0282659
_cons	-.4787261	.331378	-1.44	0.149	-1.128215	.1707628
European						
gender						
Male	.653345	.2647694	2.47	0.014	.1344065	1.172283
income	.0343647	.0080286	4.28	0.000	.0186289	.0501006
_cons	-2.839606	.461613	-6.15	0.000	-3.744351	-1.934861
Korean						
gender						
Male	.0679233	.4464535	0.15	0.879	-.8071094	.942956
income	-.0377716	.0158434	-2.38	0.017	-.068824	-.0067191
_cons	.0511728	.8033048	0.06	0.949	-1.523276	1.625621

Displaying the results as odds ratios and relative-risk ratios makes interpretation easier.

```
. cmclogit, or noheader
```

purchase	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	1.045827	.0274862	1.70	0.088	.9933192	1.101111
American	(base alternative)					
Japanese						
gender						
Male	.6843225	.1171833	-2.22	0.027	.4892166	.9572391
income	1.015618	.0066162	2.38	0.017	1.002733	1.028669
_cons	.6195721	.2053126	-1.44	0.149	.3236104	1.186209
European						
gender						
Male	1.921959	.5088759	2.47	0.014	1.143858	3.229358
income	1.034962	.0083093	4.28	0.000	1.018804	1.051377
_cons	.0584487	.0269807	-6.15	0.000	.023651	.1444443
Korean						
gender						
Male	1.070283	.4778316	0.15	0.879	.4461458	2.56756
income	.9629329	.0152561	-2.38	0.017	.9334909	.9933034
_cons	1.052505	.8454821	0.06	0.949	.2179966	5.081575

```
Note: Exponentiated coefficients represent odds ratios for
      alternative-specific variables (first equation) and relative-risk ratios
      for case-specific variables.
Note: _cons estimates baseline relative risk for each outcome.
```

These results indicate that males are less likely to pick a Japanese car over an American car than females (relative-risk ratio 0.68) but males are more likely to choose a European car over an American car than females (relative-risk ratio 1.9). Persons with higher incomes are more likely to purchase a Japanese or European car over an American car but less likely to purchase a Korean car over an American one.

How would increasing the number of dealerships for a certain nationality of car affect the likelihood of more people buying that car? Using `margins` after `cmclogit` can answer this question based on the model fit.

Let's make the question more precise: How would the probability of a person selecting a European car change if one additional European dealership was opened in each community? If this probability increases (as we expect it to), the increase must come at the expense of American, Japanese, or Korean cars. Which one of these is affected the most?

We type `margins`, specifying the options `at(dealers=generate(dealers))` and `at(dealers=generate(dealers+1))` to estimate the probabilities of selecting each nationality of car with the current number of dealerships and with one additional dealership in each community. We add the `contrast(atcontrast(r))` option to estimate the differences between these two sets of probabilities. By including the `alternative(European)` option, we request that these differences are for a change in alternative `European` (the value label for this alternative).

```
. margins, at(dealers=generate(dealers)) at(dealers=generate(dealers+1))
> contrast(atcontrast(r)) alternative(European)
Contrasts of predictive margins                          Number of obs = 3,075
Model VCE: OIM

Expression:  Pr(car|1 selected), predict()
Alternative: European

1._at: dealers =   dealers

2._at: dealers = dealers+1
```

	df	chi2	P>chi2
_at@_outcome			
(2 vs 1) American	1	2.81	0.0937
(2 vs 1) Japanese	1	2.80	0.0940
(2 vs 1) European	1	2.82	0.0934
(2 vs 1) Korean	1	2.52	0.1121
Joint	3	2.84	0.4177

	Contrast	Delta-method std. err.	[95% conf.	interval]
_at@_outcome				
(2 vs 1) American	-.0028946	.0017268	-.0062791	.0004899
(2 vs 1) Japanese	-.0024619	.0014701	-.0053434	.0004195
(2 vs 1) European	.0056244	.0033521	-.0009456	.0121944
(2 vs 1) Korean	-.0002679	.0001686	-.0005983	.0000625

We see that adding a European dealership changes the probability of someone purchasing a European car by 0.0056. This increase comes at the expense of American cars slightly more than Japanese cars. The probability of someone purchasing an American car decreases by 0.0029 per European dealership increased, and the probability of someone purchasing a Japanese car decreases by 0.0025. The probability of buying a Korean car is barely changed, only a tiny decrease of 0.0003.

See [CM] **Intro 1** and [CM] **margins** for more on using `margins` after `cmclogit`.

◁

▷ Example 2: Changing the base alternative

In the preceding example, the base alternative category was American cars, which was chosen by default because we did not specify the option `basealternative()`. The default base category is the alternative with the highest frequency of being chosen. To set the base category to Japanese cars, we specify `basealternative(Japanese)`.

```
. cmclogit purchase dealers, casevars(i.gender income)
> basealternative(Japanese) or

Iteration 0:  Log likelihood = -961.18687
Iteration 1:  Log likelihood = -948.53711
Iteration 2:  Log likelihood = -948.12131
Iteration 3:  Log likelihood = -948.12096
Iteration 4:  Log likelihood = -948.12096

Conditional logit choice model                  Number of obs      =      3,075
Case ID variable: consumerid                    Number of cases    =        862

Alternatives variable: car                      Alts per case: min =          3
                                                               avg =        3.6
                                                               max =          4

                                                   Wald chi2(7)    =      51.03
Log likelihood = -948.12096                        Prob > chi2     =     0.0000
```

purchase	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	1.045827	.0274862	1.70	0.088	.9933192	1.101111
American						
gender						
Male	1.461299	.2502327	2.22	0.027	1.044671	2.044084
income	.9846217	.0064143	-2.38	0.017	.9721298	.9972741
_cons	1.614017	.5348498	1.44	0.149	.8430215	3.090136
Japanese	(base alternative)					
European						
gender						
Male	2.808557	.7404989	3.92	0.000	1.675163	4.708792
income	1.019046	.0083438	2.30	0.021	1.002823	1.035532
_cons	.0943372	.0429738	-5.18	0.000	.0386306	.2303745
Korean						
gender						
Male	1.564004	.698962	1.00	0.317	.6513754	3.755299
income	.9481246	.0152149	-3.32	0.001	.918768	.9784192
_cons	1.698761	1.365008	0.66	0.510	.3516937	8.205401

```
Note: Exponentiated coefficients represent odds ratios for
      alternative-specific variables (first equation) and relative-risk ratios
      for case-specific variables.
Note: _cons estimates baseline relative risk for each outcome.
```

With the default base alternative of American cars, it was hard to make comparisons involving the choice of European and Korean cars relative to Japanese cars. The differences are now easy to see. ◁

▷ Example 3: altwise handling of missing values

The `altwise` option changes how `cmclogit` handles missing values. By default, missing values are handled casewise, meaning that any missing value in any observation composing the case causes the entire case to be omitted from the estimation sample. This applies to missing values in the alternative-specific and case-specific variables, the dependent variable, the alternatives variable, and in the weights if any.

If we only want to omit only observations with missing values and not the entire case, we specify the option `altwise`. We refit the model in example 1 using `altwise`. We also specify the option `basealternative(American)` so that the base alternative is the same as it was when we did casewise deletion. (When `altwise` deletion is done, the most frequent alternative is Japanese, and if we did not specify `basealternative()`, Japanese would be used by default as the base alternative.)

```
. cmclogit purchase dealers, casevars(i.gender income) altwise or
> basealternative(American)
note: variable dealers has 1 case that is not alternative-specific; there is
      no within-case variability.

Iteration 0:  Log likelihood =  -976.1027
Iteration 1:  Log likelihood = -965.37952
Iteration 2:  Log likelihood = -965.01714
Iteration 3:  Log likelihood =  -965.0164
Iteration 4:  Log likelihood =  -965.0164

Conditional logit choice model                  Number of obs      =      3,137
Case ID variable: consumerid                    Number of cases    =        885

Alternatives variable: car                      Alts per case: min =          2
                                                               avg =        3.5
                                                               max =          4

                                                   Wald chi2(7)    =      54.18
Log likelihood =  -965.0164                        Prob > chi2     =     0.0000
```

purchase	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	1.05095	.0272151	1.92	0.055	.9989405	1.105668
American	(base alternative)					
Japanese						
gender						
Male	.6866621	.1165921	-2.21	0.027	.4922809	.9577963
income	1.01551	.0065797	2.38	0.018	1.002696	1.028488
_cons	.6287248	.2070286	-1.41	0.159	.3297418	1.198801
European						
gender						
Male	1.997394	.5266466	2.62	0.009	1.191324	3.348863
income	1.035968	.0082453	4.44	0.000	1.019933	1.052255
_cons	.0558718	.0255968	-6.30	0.000	.0227629	.1371379
Korean						
gender						
Male	1.066616	.4762214	0.14	0.885	.4445952	2.55889
income	.9628352	.0153454	-2.38	0.017	.9332236	.9933864
_cons	1.054358	.8492452	0.07	0.948	.2174589	5.112091

```
Note: Exponentiated coefficients represent odds ratios for
      alternative-specific variables (first equation) and relative-risk ratios
      for case-specific variables.
Note: _cons estimates baseline relative risk for each outcome.
```

Results are similar to the model fit in example 1. That estimation sample had 862 cases; this one has 885, a difference of 23 cases.

We suspect that handling missing values alternativewise changes the choice sets. To see the choice sets used in the estimation, we type cmchoiceset with an if restriction to the estimation sample.

```
. cmchoiceset if e(sample) == 1
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2	2	0.23	0.23
1 2 3	378	42.71	42.94
1 2 3 4	489	55.25	98.19
1 2 4	4	0.45	98.64
1 3	2	0.23	98.87
1 3 4	2	0.23	99.10
2 3	3	0.34	99.44
2 3 4	5	0.56	100.00
Total	885	100.00	

```
Note: Total is number of cases.
```

When missing values were handled casewise, there were only two distinct choice sets: $\{1, 2, 3\}$ and $\{1, 2, 3, 4\}$. Handling the missing values alternativewise gives six new choice sets, albeit each with low frequency.

Handling missing values casewise never creates new choice sets. Handling missing values with altwise almost always changes the choice sets used in the estimation. You should be aware of the consequences. For instance, a dataset with balanced choice sets will typically become unbalanced when missing values are handled alternativewise.

The cmsample command can help you to see exactly what observations and cases are dropped, whether you use the casewise default or altwise.

◁

▷ Example 4: Multiple choices per case

Let us continue with our fictitious car choice dataset but expand it so that it also contains data on the purchase of a second car. Here is what the data look like now:

```
. use https://www.stata-press.com/data/r18/carchoice_panel, clear
(Car choice panel data)
. list consumerid carnumber car purchase
> if inlist(consumerid, 6, 7), sepby(consumerid carnumber) abbr(10)
```

	consumerid	carnumber	car	purchase
18.	6	1	American	1
19.	6	1	Japanese	0
20.	6	1	European	0
21.	7	1	American	0
22.	7	1	Japanese	0
23.	7	1	European	1
24.	7	1	Korean	0
25.	7	2	American	0
26.	7	2	Japanese	0
27.	7	2	European	0
28.	7	2	Korean	1

The person with consumerid = 7 has two cars, the first a European car and the second a Korean car. The person with consumerid = 6 has only one car, an American one.

How do we model these data?

The random utility model for McFadden's choice model yields only one chosen alternative per case: that with the greatest utility. Because the utility function is continuous, ties are theoretically impossible. See *Methods and formulas*. Choice models for rank-ordered data allow for multiple alternatives to be chosen and allow for tied ranks; for more information, see [CM] **Intro 6**.

Train (2009, sec. 2.2) notes that the set of alternatives can always be made mutually exclusive by considering the choice of two alternatives as a separate alternative. For example, with one or two choices allowed from alternatives A, B, and C, the set of alternatives is A only, B only, C only, A and B, A and C, and B and C, a total of six alternatives.

We could do this with our expanded car choice data. But this would mean a model with 14 alternatives. There are four nationalities of cars. So the alternatives are only one car of one of these nationalities (four possibilities), two cars of the same nationality (four possibilities), and two cars of different nationalities (six possibilities). With so many possibilities, there are concerns both about statistical power and about ease of model interpretation.

There is another way to view our expanded car choice data. The stated design of the fictitious data collection was to take the most recent car purchased, and if another car was purchased in the previous five years by anyone in the household, then to collect data on that car as well. So these data are panel data with information from two time points.

The variable `carnumber` will do as a time variable, and we can `cmset` the data as panel choice data:

```
. cmset consumerid carnumber car
note: case identifier _caseid generated from consumerid and carnumber.
note: panel by alternatives identifier _panelaltid generated from consumerid
      and car.
note: alternatives are unbalanced across choice sets; choice sets of different
      sizes found.

                Panel data: Panels consumerid and time carnumber
          Case ID variable: _caseid
     Alternatives variable: car
Panel by alternatives variable: _panelaltid (unbalanced)
             Time variable: carnumber, 1 to 2
                     Delta: 1 unit

Note: Data have been xtset.
```

See [CM] **Intro 7** and [CM] **cmset** for more information on `cmset`ting panel choice data.

Once we have cmset the data, we can run cmclogit, issuing the same command line we used for the dataset with only one car per person.

```
. cmclogit purchase dealers, casevars(i.gender income) or
> basealternative(American)
note: data were cmset as panel data, and the default vcetype for panel data is
      vce(cluster consumerid); see cmclogit.

Iteration 0:  Log pseudolikelihood = -1236.6065
Iteration 1:  Log pseudolikelihood = -1221.0412
Iteration 2:  Log pseudolikelihood = -1220.8605
Iteration 3:  Log pseudolikelihood = -1220.8604

Conditional logit choice model                  Number of obs      =      3,728
Case ID variable: _caseid                       Number of cases    =       1045

Alternatives variable: car                      Alts per case: min =          3
                                                               avg =        3.6
                                                               max =          4

                                                   Wald chi2(7)    =      42.76
Log pseudolikelihood = -1220.8604                  Prob > chi2     =     0.0000

                          (Std. err. adjusted for 862 clusters in consumerid)
```

purchase	Odds ratio	Robust std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	1.020878	.0236626	0.89	0.373	.9755374	1.068325
American	(base alternative)					
Japanese						
gender						
Male	.6751564	.1104886	-2.40	0.016	.4898986	.9304704
income	1.01464	.0060839	2.42	0.015	1.002785	1.026634
_cons	.518233	.1608041	-2.12	0.034	.2821	.9520221
European						
gender						
Male	1.590534	.3411024	2.16	0.030	1.044711	2.421529
income	1.032775	.0074815	4.45	0.000	1.018216	1.047543
_cons	.074066	.0306088	-6.30	0.000	.0329494	.1664908
Korean						
gender						
Male	1.133754	.3022134	0.47	0.638	.6723919	1.911679
income	.9938	.0101097	-0.61	0.541	.9741815	1.013814
_cons	.5676166	.3002472	-1.07	0.284	.2012808	1.600692

```
Note: Exponentiated coefficients represent odds ratios for
      alternative-specific variables (first equation) and relative-risk ratios
      for case-specific variables.
Note: _cons estimates baseline relative risk for each outcome.
```

Note that cmclogit knew the data were panel data with some individuals having more than one case, and it automatically set the variance estimator used to vce(cluster consumerid), where consumerid is the ID for individuals. If, for whatever reason, you wish to use another variance estimator, you can set vce() explicitly and cmclogit will respect your choice. See [R] *vce_option* for details on the available choices for vce().

◁

Stored results

cmclogit stores the following in e():

Scalars

e(N)	number of observations
e(N_case)	number of cases
e(N_ic)	N for Bayesian information criterion (BIC)
e(N_clust)	number of clusters
e(k)	number of parameters
e(k_alt)	number of alternatives
e(k_indvars)	number of alternative-specific variables
e(k_casevars)	number of case-specific variables
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(const)	constant indicator
e(i_base)	base alternative index
e(chi2)	χ^2
e(p)	p-value for model test
e(alt_min)	minimum number of alternatives
e(alt_avg)	average number of alternatives
e(alt_max)	maximum number of alternatives
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	cmclogit
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(caseid)	name of case ID variable
e(altvar)	name of alternatives variable
e(alteqs)	alternative equation names
e(alt#)	alternative labels
e(wtype)	weight type
e(wexp)	weight expression
e(marktype)	casewise or altwise, type of markout
e(key_N_ic)	cases, key for N for Bayesian information criterion (BIC)
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(offset)	linear offset variable
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(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(predict)	program used to implement predict
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(stats)	alternative statistics
e(altvals)	alternative values
e(altfreq)	alternative frequencies
e(alt_casevars)	indicators for estimated case-specific coefficients—e(k_alt)×e(k_casevars)
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

In this model, we have a set of unordered alternatives indexed by $1, 2, \ldots, J$. Let y_{ij}, $j = 1, \ldots, J$, be an indicator variable for the alternative chosen by the ith individual (case). That is, $y_{ij} = 1$ if individual i chose alternative j, and $y_{ij} = 0$ otherwise.

The independent variables come in two forms: alternative specific and case specific. Alternative-specific variables vary among the alternatives and the cases, and case-specific variables vary only among cases. Assume that we have p alternative-specific variables so that for case i we have a $J \times p$ matrix, $\mathbf{X}_i$. Assume that we have q case-specific variables so that we have a $1 \times q$ vector $\mathbf{z}_i$ for case i.

The deterministic component of the random utility model can then be expressed as

$$\begin{aligned}\boldsymbol{\eta}_i &= \mathbf{X}_i\boldsymbol{\beta} + (\mathbf{z}_i\mathbf{A})' \\ &= \mathbf{X}_i\boldsymbol{\beta} + (\mathbf{z}_i \otimes \mathbf{I}_J)\,\mathrm{vec}(\mathbf{A}') \\ &= (\mathbf{X}_i,\ \mathbf{z}_i \otimes \mathbf{I}_J)\left\{\begin{array}{c}\boldsymbol{\beta} \\ \mathrm{vec}(\mathbf{A}')\end{array}\right\} \\ &= \mathbf{X}_i^*\boldsymbol{\beta}^*\end{aligned}$$

As before, $\boldsymbol{\beta}$ is a $p \times 1$ vector of alternative-specific regression coefficients, and $\mathbf{A} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_J)$ is a $q \times J$ matrix of case-specific regression coefficients. We must set one of the $\boldsymbol{\alpha}_j$ to zero to normalize the location. Here $\mathbf{I}_J$ is the $J \times J$ identity matrix, vec(·) is the vector function that creates a vector from a matrix by placing each column of the matrix on top of the other (see [M-5] **vec( )**), and $\otimes$ is the Kronecker product (see [M-2] **op_kronecker**).

We have rewritten the linear equation so that it is a form that can be used by clogit, namely, $\mathbf{X}_i^*\boldsymbol{\beta}^*$, where

$$\mathbf{X}_i^* = (\mathbf{X}_i,\ \mathbf{z}_i \otimes \mathbf{I}_J)$$

$$\boldsymbol{\beta}^* = \left\{\begin{array}{c}\boldsymbol{\beta} \\ \mathrm{vec}(\mathbf{A}')\end{array}\right\}$$

With this in mind, see *Methods and formulas* in [R] **clogit** for the computational details of the conditional logit model.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster` *clustvar*`)`. See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*. Specifying `vce(robust)` is equivalent to specifying `vce(cluster` *caseid*`)`, where *caseid* is the variable that identifies the cases.

Daniel Little McFadden (1937–) was born in North Carolina. He studied physics, psychology, and economics at the University of Minnesota and has taught economics at Pittsburgh, Berkeley, MIT, and the University of Southern California. His contributions to logit models were triggered by a student's project on freeway routing decisions, and his work consistently links economic theory and applied problems. In 2000, he shared the Nobel Prize in Economics with James J. Heckman.

References

Cameron, A. C., and P. K. Trivedi. 2022. *Microeconometrics Using Stata*. 2nd ed. College Station, TX: Stata Press.

Greene, W. H. 2018. *Econometric Analysis*. 8th ed. New York: Pearson.

McFadden, D. L. 1974. Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics*, ed. P. Zarembka, 105–142. New York: Academic Press.

Train, K. E. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. New York: Cambridge University Press.

Also see

[CM] **cmclogit postestimation** — Postestimation tools for cmclogit

[CM] **cmmixlogit** — Mixed logit choice model

[CM] **cmmprobit** — Multinomial probit choice model

[CM] **cmset** — Declare data to be choice model data

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

[CM] **nlogit** — Nested logit regression

[R] **clogit** — Conditional (fixed-effects) logistic regression

[R] **mlogit** — Multinomial (polytomous) logistic regression

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

Title

cmclogit postestimation — Postestimation tools for cmclogit

Postestimation commands predict margins Remarks and examples
Also see

Postestimation commands

The following postestimation commands are available after `cmclogit`:

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`	adjusted predictions, predictive margins, and 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`	probabilities, 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 probabilities, linear predictions, and standard errors.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [, *statistic* `nooffset`]

`predict` [*type*] *stub*`*` [*if*] [*in*], `scores`

statistic	Description
Main	
`pr`	probability alternative is chosen; the default
`xb`	linear prediction
`stdp`	standard error of the linear prediction

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`predict` omits missing values casewise if `cmclogit` used casewise deletion (the default); if `cmclogit` used alternativewise deletion (option `altwise`), `predict` uses alternativewise deletion.

Options for predict

Main

`pr`, the default, calculates the probability of choosing each alternative.

`xb` calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`nooffset` is relevant only if you specified `offset(`*varname*`)` for `cmclogit`. 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 as $\mathbf{x}\boldsymbol{\beta} + \text{offset}$.

`scores` calculates the scores for each coefficient in `e(b)`. This option requires a new variable list of length equal to the number of columns in `e(b)`. Otherwise, use the *stub*`*` syntax to have `predict` generate enumerated variables with prefix *stub*.

margins

Description for margins

`margins` estimates margins of response for probabilities and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [, *options*]

`margins` [*marginlist*] , `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`pr`	probability alternative is chosen; the default
`xb`	linear prediction
`stdp`	not allowed with `margins`
`scores`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For more details, see [CM] **margins**.

Remarks and examples

Remarks are presented under the following headings:

Testing coefficient estimates
Predicted probabilities
    Casewise versus alternativewise sample selection
Obtaining estimation statistics for the alternatives

Testing coefficient estimates

The output of `cmclogit` is displayed and stored as a multiple-equation model. Let's illustrate this with example 1 of [CM] **cmclogit**.

▷ Example 1

We load the data, `cmset` the data, and run `cmclogit`.

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of different
      sizes found.
     Case ID variable: consumerid
Alternatives variable: car
. cmclogit purchase dealers, casevars(i.gender income)
  (output omitted)
```

The coefficient estimates for `i.gender` and `income` are stored under the equation names `Japanese`, `European`, and `Korean`, that is, the names of the alternatives, except for the base alternative `American`. To test whether the coefficient estimates for `i.gender` are the same for the Japanese and Korean alternatives relative to the American base alternative, we type

```
. test [Japanese]:1.gender = [Korean]:1.gender
 ( 1)  [Japanese]1.gender - [Korean]1.gender = 0
           chi2(  1) =    1.00
         Prob > chi2 =    0.3169
```

The following shorthand syntax is useful for testing across the alternatives:

```
. test [Japanese = European = Korean]:1.gender
 ( 1)  [Japanese]1.gender - [European]1.gender = 0
 ( 2)  [Japanese]1.gender - [Korean]1.gender = 0
           chi2(  2) =   15.62
         Prob > chi2 =    0.0004
```

See [R] **test** for details.

◁

Predicted probabilities

After running `cmclogit`, you can use `predict` to obtain the estimated probability that each alternative is chosen for each case conditional on its observed data.

▷ Example 2

Continuing with the previous example, we calculate predicted probabilities and list them for the first four cases:

```
. predict p
(option pr assumed; Pr(car))
. format p %6.3f
. list consumerid car purchase gender income p
> if consumerid <= 4, sepby(consumerid) abbr(10)
```

	consumerid	car	purchase	gender	income	p
1.	1	American	1	Male	46.7	0.391
2.	1	Japanese	0	Male	46.7	0.374
3.	1	European	0	Male	46.7	0.183
4.	1	Korean	0	Male	46.7	0.053
5.	2	American	1	Male	26.1	0.493
6.	2	Japanese	0	Male	26.1	0.274
7.	2	European	0	Male	26.1	0.095
8.	2	Korean	0	Male	26.1	0.138
9.	3	American	0	Male	32.7	0.524
10.	3	Japanese	1	Male	32.7	0.337
11.	3	European	0	Male	32.7	0.138
12.	4	American	1	Female	49.2	0.391
13.	4	Japanese	0	Female	49.2	0.496
14.	4	European	0	Female	49.2	0.113

To get predicted probabilities and marginal effects averaged across the sample or for hypothetical cases (that is, predictor values set to particular values), use the `margins` postestimation command; see [CM] **Intro 1** and [CM] **margins** for more information and examples.

◁

Casewise versus alternativewise sample selection

Missing values in CM data are handled in two possible ways: casewise deletion (the default) and alternativewise (`altwise`) deletion. Casewise deletion omits the whole case whenever any observation within the case has a missing value. Alternativewise deletion omits only the observations with missing values.

`predict` uses whatever was used with `cmclogit`. If `cmclogit` used casewise deletion, `predict` uses casewise deletion. If `cmclogit` was used with the option `altwise`, `predict` uses alternativewise deletion. Should you wish to select the sample for `predict` yourself, you can use an `if` restriction with `predict`.

See example 3 in [CM] **cmclogit** for more on casewise versus alternativewise deletion.

Obtaining estimation statistics for the alternatives

▷ Example 3

`cmtab` can be used to obtain a table of the alternatives for the estimation sample. If there are missing values in the data used to fit the model, you will need to restrict `cmtab` to the estimation sample by specifying `if e(sample)`.

```
. cmtab if e(sample), choice(purchase)
Tabulation of chosen alternatives (purchase = 1)
```

Nationality of car	Freq.	Percent	Cum.
American	376	43.62	43.62
Japanese	316	36.66	80.28
European	130	15.08	95.36
Korean	40	4.64	100.00
Total	862	100.00	

`cmchoiceset` is useful for obtaining a table of the choice-set patterns.

```
. cmchoiceset if e(sample)
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2 3	373	43.27	43.27
1 2 3 4	489	56.73	100.00
Total	862	100.00	

```
Note: Total is number of cases.
```

If you have missing data or see notes mentioning cases being dropped, `cmsample` can identify omitted observations and show the reason they were omitted from the estimation sample. See [CM] **cmsample**.

◁

Also see

[CM] **cmclogit** — Conditional logit (McFadden's) choice model

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

[U] 20 Estimation and postestimation commands

Title

cmmixlogit — Mixed logit choice model

Description Quick start Menu Syntax
Options Remarks and examples Stored results Methods and formulas
References Also see

Description

`cmmixlogit` fits a mixed logit choice model, also known as a mixed multinomial logit model or random-parameter logit model, which uses random coefficients to model the correlation of choices across alternatives. The random coefficients are on variables that vary across both cases and alternatives known as alternative-specific variables.

The correlation of choices across alternatives relaxes the independence of irrelevant alternatives (IIA) property imposed by the conventional multinomial logit model fit by `mlogit` and the conditional logit choice model fit by `cmclogit`.

For a mixed logit choice model for panel data, see [CM] **cmxtmixlogit**.

Quick start

Mixed logit regression of `y` on `x1`, where the coefficients on `x1` are assumed random normal, using `cmset` data

```
cmmixlogit y, random(x1)
```

Same as above, and include levels of case-specific covariate `a`

```
cmmixlogit y, random(x1) casevars(i.a)
```

Same as above, and add covariate `x2`, whose coefficients are random triangular

```
cmmixlogit y, random(x1) random(x2, triangle) casevars(i.a)
```

Mixed logit model of `y` on `x1`, `x2`, and `x3`, where the random coefficients for `x2` and `x3` are correlated

```
cmmixlogit y x1, random(x2 x3, correlated)
```

Same as above, but omit the alternative-specific constants

```
cmmixlogit y x1, random(x2 x3, correlated) noconstant
```

Menu

Statistics > Choice models > Mixed logit model

Syntax

```
cmmixlogit depvar [indepvars] [if] [in] [weight] [, options]
```

depvar equal to 1 identifies the chosen alternatives, whereas a 0 indicates the alternatives that were not chosen. There can be only one chosen alternative for each case.

indepvars specifies the alternative-specific covariates that have fixed coefficients.

options	Description
Model	
`casevars(`*varlist*`)`	case-specific variables
`random(`*varlist* [`,` *distribution*]`)`	specify variables that are to have random coefficients and the coefficients' distribution
`corrmetric(`*metric*`)`	correlation metric for correlated random coefficients
`basealternative(`*# | lbl | str*`)`	alternative used for normalizing location
`noconstant`	suppress the alternative-specific constant terms
`altwise`	use alternativewise deletion instead of casewise deletion
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `robust`, `cluster` *clustvar*, `opg`, `bootstrap`, or `jackknife`
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`or`	report odds ratios and relative-risk ratios
`nocnsreport`	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
Integration	
`intmethod(`*seqspec*`)`	specify point set for Monte Carlo integration
`intpoints(#)`	specify number of points in each sequence
`intburn(#)`	specify starting index in the Hammersley or Halton sequence
`intseed(#)`	specify random-number seed for pseudo–random sequence
`favor(speed | space)`	favor speed or space when generating integration points
Maximization	
maximize_options	control the maximization process; seldom used
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

distribution	Description
normal	Gaussian-distributed random coefficients; the default
correlated	correlated Gaussian-distributed random coefficients
lnormal	lognormal distributed random coefficients
tnormal	truncated normal distributed random coefficients
uniform	uniform distributed random coefficients
triangle	triangular distributed random coefficients

metric	Description
correlation	standard deviation and correlation; the default
covariance	variance and covariance
cholesky	Cholesky factor

seqspec is

seqtype [, antithetics | mantithetics]

seqtype	Description
hammersley	Hammersley point set; the default
halton	Halton point set
random	uniform pseudo–random point set

You must cmset your data before using cmmixlogit; see [CM] **cmset**.
indepvars and *varlist* may contain factor variables; see [U] **11.4.3 Factor variables**.
bootstrap, by, collect, jackknife, statsby, and svy are allowed; see [U] **11.1.10 Prefix commands**.
Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.
vce() 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

casevars(*varlist*) specifies the case-specific variables that are constant for each case. If there are a maximum of A alternatives, there will be $A-1$ sets of coefficients associated with casevars().

random(*varlist* [, *distribution*]) specifies the alternative-specific variables that are to have random coefficients and optionally the assumed distribution of the random coefficients. The default distribution is normal, meaning Gaussian-distributed random coefficients. *distribution* may also be correlated, lnormal, tnormal, uniform, or triangle. random() may be specified more than once to specify different sets of variables that correspond to different coefficient distributions.

`corrmetric(`*metric*`)` specifies the estimation metric for correlated random coefficients. `corrmetric(correlation)`, the default, estimates the standard deviations and correlations of the random coefficients. `corrmetric(covariance)` estimates variances and covariances, and `corrmetric(cholesky)` estimates Cholesky factors. `corrmetric()` is allowed only when `random(`*varlist*`, correlated)` is specified.

`basealternative(`*#* | *lbl* | *str*`)` sets the alternative that normalizes the level of utility. The base alternative may be specified as a number when the alternatives variable is numeric, as a label when it is numeric and has a value label, or as a string when it is a string variable. The default is the alternative with the highest frequency of being chosen. This option is ignored if neither alternative-specific constants nor case-specific variables are specified.

`noconstant` suppresses the $A-1$ alternative-specific constant terms.

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

`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***.

Specifying `vce(robust)` is equivalent to specifying `vce(cluster` *caseid*`)`.

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `basealternative()`.

Reporting

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

`or` reports the estimated coefficients transformed to odds ratios for alternative-specific variables and relative-risk ratios for case-specific variables. That is, e^b rather than b is reported. Standard errors and confidence intervals are transformed accordingly. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`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**.

Integration

`intmethod(`*seqtype*`[ , antithetics | mantithetics ])` specifies the method of generating the point sets used in the Monte Carlo integration. `intmethod(hammersley)`, the default, uses the Hammersley sequence; `intmethod(halton)` uses the Halton sequence; and `intmethod(random)` uses a sequence of uniform random numbers.

`antithetics` and `mantithetics` specify that a unidimensional antithetic sequence or a multidimensional antithetic sequence, respectively, be generated instead of the standard implementation

of the requested *seqtype*. These methods improve the accuracy of the Monte Carlo integration at the cost of additional computation time; see *Methods and formulas*.

`intpoints(#)` specifies the number of points to use in the Monte Carlo integration. The default number of points is a function of model complexity and integration method. If `intmethod(hammersley)` or `intmethod(halton)` is used, the default is $500 + \text{floor}[2.5\sqrt{N_c\{\ln(r+5)+v\}}]$, where N_c is the number of cases, r is the number of random coefficients in the model, and v is the number of variance parameters. If `intmethod(hammersley, mantithetics)` or `intmethod(halton, mantithetics)` is used, the number of integration points is $250+\text{floor}[0.5\sqrt{N_c\{\ln(r+5)+v\}}]$. If `intmethod(random)` is used, the number of points is twice the number of points used by `intmethod(hammersley)` and `intmethod(halton)`. Larger values of `intpoints()` provide better approximations of the log likelihood at the cost of additional computation time.

`intburn(#)` specifies where in the Hammersley or Halton sequence to start, which helps reduce the correlation between the sequences of each dimension. The default is to discard the first n initial elements from each sequence, where n is the largest prime used to generate the sequences. This option may not be specified with `intmethod(random)`.

`intseed(#)` specifies the seed to use for generating uniform pseudo–random sequences. This option may be specified only with `intmethod(random)`. # must be an integer between 0 and $2^{31}-1$. The default is to use the current seed value from Stata's uniform random-number generator; see [R] **set seed**.

`favor(speed|space)` instructs `cmmixlogit` to favor either speed or space when generating the integration points. `favor(speed)` is the default. When favoring speed, the integration points are generated once and stored in memory, thus increasing the speed of evaluating the likelihood. This speed increase can be seen when there are many cases or when the user specifies many integration points in `intpoints(#)`. When favoring space, the integration points are generated repeatedly with each likelihood evaluation.

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**.

Setting the optimization type to `technique(bhhh)` resets the default *vcetype* to `vce(opg)`.

The following options are available with `cmmixlogit` but are not shown in the dialog box:

`collinear`, `coeflegend`; see [R] **Estimation options**.

Remarks and examples

`cmmixlogit` fits a mixed logit choice model, in the following simply referred to as a mixed logit model. The mixed logit model is most frequently used to model the probability that an individual chooses one of several unordered alternatives. It is also known as the mixed multinomial logit model (McFadden and Train 2000), the random-parameters logit model (Cameron and Trivedi 2005), the logit kernel model (Ben-Akiva, Bolduc, and Walker 2001), or the hybrid logit model (Ben-Akiva et al. 1997).

The mixed logit model is often used in the context of discrete choice models. These models represent how decision makers choose among a finite set of alternatives. The decision maker, called a case, is often an individual. The mixed logit model can incorporate attributes that vary across individuals, known as case-specific variables. Income, educational attainment, and age are examples of case-specific variables.

The model can also incorporate observed attributes that vary by alternative or by alternative and individual, known as alternative-specific variables. The size of the lake at a fishing site is an example of an alternative-specific covariate that varies only by alternative. The travel distance to any given fishing site is an example of an alternative-specific covariate that varies by alternative and individual.

In the mixed logit model, the coefficients on alternative-specific variables can be treated as fixed or random. Specifying random coefficients can model correlation of choices across alternatives, thereby relaxing the IIA property that is imposed by the multinomial logit models described in [R] **mlogit** and the conditional logit models described in [R] **clogit** and [CM] **cmclogit**. In this sense, the mixed logit model fit by `cmmixlogit` is more flexible than the models fit by `mlogit`, `clogit`, and `cmclogit`.

McFadden and Train (2000) show that the mixed logit model can approximate a wide class of choice representations. Although the mixed logit model was derived under a utility framework and is most often used for these applications, it also can be applied in contexts that lack this individual-choice motivation, for example, classification problems. For an introduction to mixed logit models, see Cameron and Trivedi (2005) and Train (2009). See Hole (2007) for a previous implementation of mixed logit models via the community-contributed `mixlogit` command. For the panel-data mixed logit model, see [CM] **cmxtmixlogit**.

In discrete choice, an individual chooses the alternative that yields the highest value of an unobserved ranking index known as utility. Utility is a latent variable that is a function of observed attributes of the individuals, the alternatives, random coefficients, and a random component. In other contexts, such as classification analysis, utility is just an unobserved random index. We call it utility here because this model is most frequently applied to discrete choice data.

For the mixed logit model, the utility that individual i receives from alternative a, $a = 1, 2, \ldots, A$, denoted by U_{ia}, is

$$U_{ia} = \mathbf{x}_{ia}\boldsymbol{\beta}_i + \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a + \epsilon_{ia}$$

$\boldsymbol{\beta}_i$ are random coefficients that vary over individuals in the population, and $\mathbf{x}_{ia}$ is a vector of alternative-specific variables. $\boldsymbol{\alpha}$ are fixed coefficients on $\mathbf{w}_{ia}$, a vector of alternative-specific variables. $\boldsymbol{\delta}_a$ are fixed alternative-specific coefficients on $\mathbf{z}_i$, a vector of case-specific variables. ϵ_{ia} is a random term that follows a type I extreme value distribution.

`cmmixlogit` estimates the fixed coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}_a$ and the parameters of $f(\boldsymbol{\beta})$, the distribution of the random coefficients. The mixed logit model can estimate only the parameters of $f(\boldsymbol{\beta})$, not the $\boldsymbol{\beta}_i$ per se. For example, if the random coefficients $\boldsymbol{\beta}_i$ follow a normal distribution, $\boldsymbol{\beta}_i \sim N(\mu, \boldsymbol{\Sigma})$, then the mixed logit model estimates μ and $\boldsymbol{\Sigma}$.

Note that only the rank order of the utilities for each alternative matters; that is, the location and scale of utility are irrelevant. The data reveal only the chosen alternative, so we must normalize for location by taking differences with respect to a base alternative k. The assumed type I extreme value distribution implies that the difference in the errors for alternative a and base k, $\epsilon_{ia} - \epsilon_{ik}$, follows a logistic distribution. Assuming a standard logistic distribution normalizes for scale.

The choice probabilities are the standard logistic probabilities integrated over the density $f(\boldsymbol{\beta})$. That is, the probability of choosing alternative a for individual i is

$$P_{ia} = \int P_{ia}(\boldsymbol{\beta}) f(\boldsymbol{\beta}) d\boldsymbol{\beta} \tag{1}$$

where

$$P_{ia}(\boldsymbol{\beta}) = \frac{e^{\mathbf{x}_{ia}\boldsymbol{\beta}_i + \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a}}{\sum_{a=1}^{A} e^{\mathbf{x}_{ia}\boldsymbol{\beta}_i + \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a}}$$

are the logistic probabilities, evaluated at parameters $\boldsymbol{\beta}$.

The integral in (1) must be approximated because it has no closed-form solution. `cmmixlogit` approximates (1) by simulation and estimates the model parameters by maximum simulated likelihood (MSL). Consistency of the MSL estimator requires that the number of random draws in the simulation method be sufficiently large. More draws will produce more precise estimates by reducing approximation error, at the cost of increased computation time. See *Methods and formulas* for further details, and see Cameron and Trivedi (2005) for an introduction to MSL estimation.

▷ Example 1: Mixed logit model with fixed and random coefficients

`inschoice.dta` records information about available insurance plans and the selected plan for 250 individuals. Each individual selected an insurance plan from the five alternatives that are recorded in the `insurance` variable. The binary variable `choice` records the chosen alternative; `choice` is 1 for the chosen alternative and 0 otherwise. For each individual, we have one observation for each alternative. Here are the data for the first two individuals:

```
. use https://www.stata-press.com/data/r18/inschoice
(Fictional health insurance data)
. list in 1/10, sepby(id) abbreviate(10)
```

	id	premium	deductible	income	insurance	choice
1.	1	2.87	1.70	5.74	Health	1
2.	1	3.13	2.14	5.74	HCorp	0
3.	1	2.03	2.26	5.74	SickInc	0
4.	1	1.65	2.94	5.74	MGroup	0
5.	1	0.87	3.56	5.74	MoonHealth	0
6.	2	3.52	1.24	2.89	Health	0
7.	2	3.23	1.52	2.89	HCorp	0
8.	2	2.81	2.31	2.89	SickInc	0
9.	2	1.04	2.58	2.89	MGroup	1
10.	2	0.93	3.17	2.89	MoonHealth	0

Insurance premiums (`premium`) and deductibles (`deductible`) vary over alternatives and are thus alternative-specific variables. In this example, they also vary over individuals. Income (`income`) varies only over individuals and is thus a case-specific variable.

We wish to estimate the effect of health insurance premiums, insurance deductibles, and personal income on the choice of health insurance plans. We assume that preferences with respect to deductibles vary over individuals in the population but that preferences with respect to premiums are constant over individuals in the population.

Before we can fit a model, we must `cmset` our data. The first argument to `cmset` is the case ID variable. For these data, it is the variable `id`, which identifies individuals. The second argument is the alternatives variable, which identifies the alternatives that could have been chosen. In this instance, it is the variable `insurance`, which identifies the insurance plans available.

```
. cmset id insurance
     Case ID variable: id
Alternatives variable: insurance
```

We fit a model for this outcome by using `cmmixlogit`. We specify `random(deductible)` to include random coefficients on `deductible`, and we include `premium` as *indepvar* to include a fixed coefficient on `premium`.

```
. cmmixlogit choice premium, random(deductible)
Fitting fixed parameter model:
Fitting full model:
Iteration 0:  Log simulated-likelihood = -296.14208  (not concave)
Iteration 1:  Log simulated-likelihood = -295.74886
Iteration 2:  Log simulated-likelihood = -295.05013
Iteration 3:  Log simulated-likelihood = -295.04639
Iteration 4:  Log simulated-likelihood = -295.04639
Mixed logit choice model                        Number of obs      =      1,250
Case ID variable: id                            Number of cases    =        250
Alternatives variable: insurance                Alts per case: min =          5
                                                               avg =        5.0
                                                               max =          5
Integration sequence:      Hammersley
Integration points:               567           Wald chi2(2)       =      99.32
Log simulated-likelihood = -295.04639           Prob > chi2        =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
insurance						
premium	-2.672185	.269542	-9.91	0.000	-3.200478	-2.143893
deductible	-1.109975	.337406	-3.29	0.001	-1.771278	-.4486709
/Normal						
sd(deducti~e)	.8886822	.3641774			.3980416	1.984104
Health						
_cons	.5208962	.2979741	1.75	0.080	-.0631222	1.104915
HCorp	(base alternative)					
SickInc						
_cons	-.8426238	.2909169	-2.90	0.004	-1.412811	-.2724371
MGroup						
_cons	-2.107124	.4436973	-4.75	0.000	-2.976755	-1.237494
MoonHealth						
_cons	-3.36121	.6795993	-4.95	0.000	-4.6932	-2.02922

```
LR test vs. fixed parameters: chibar2(01) =     2.99  Prob >= chibar2 = 0.0420
```

The estimated fixed coefficient on `premium` is -2.67, so an increase in a plan's premium reduces the probability that it is chosen. The estimated mean of the normally distributed coefficients on `deductible` is -1.11. The estimated standard deviation of these random coefficients is 0.89, indicating heterogeneity across individuals in the population with respect to the effect of a plan's deductible.

The likelihood-ratio (LR) test in the footer shows the result of a test against a model with only fixed coefficients and indicates that we can reject the null hypothesis that the coefficients on `deductible` are fixed.

`cmmixlogit` computes the likelihood using Monte Carlo integration. Several options are available to control how the integration is done. In almost all cases, you will never have to change any of the defaults. However, there is one of them, `intpoints(#)`, the number of integration points, that you may sometimes want to set. The default value is set to a number that is typically adequate, but when you have a model you consider final, you should run the estimation again, setting `intpoints()` to a bigger number to confirm the numerical soundness of the estimates.

We see from the header on the earlier output that `cmmixlogit` used 567 integration points by default. Let's run it again using `intpoints(1000)`.

```
. cmmixlogit choice premium, random(deductible) intpoints(1000)
Fitting fixed parameter model:
Fitting full model:
Iteration 0:  Log simulated-likelihood = -296.14179  (not concave)
Iteration 1:  Log simulated-likelihood = -295.74946
Iteration 2:  Log simulated-likelihood = -295.05088
Iteration 3:  Log simulated-likelihood = -295.04713
Iteration 4:  Log simulated-likelihood = -295.04713
Mixed logit choice model                    Number of obs      =      1,250
Case ID variable: id                        Number of cases    =        250
Alternatives variable: insurance            Alts per case: min =          5
                                                           avg =        5.0
                                                           max =          5
Integration sequence:      Hammersley
Integration points:              1000       Wald chi2(2)       =      99.30
Log simulated-likelihood = -295.04713       Prob > chi2        =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
insurance						
premium	-2.672171	.2695616	-9.91	0.000	-3.200502	-2.14384
deductible	-1.109943	.337404	-3.29	0.001	-1.771243	-.4486433
/Normal						
sd(deducti~e)	.8885033	.3643278			.3977651	1.984684
Health						
_cons	.5208928	.2979772	1.75	0.080	-.0631319	1.104917
HCorp	(base alternative)					
SickInc						
_cons	-.8426217	.2909165	-2.90	0.004	-1.412808	-.2724358
MGroup						
_cons	-2.107116	.4437085	-4.75	0.000	-2.976769	-1.237463
MoonHealth						
_cons	-3.361211	.6796514	-4.95	0.000	-4.693304	-2.029119

```
LR test vs. fixed parameters: chibar2(01) =     2.99  Prob >= chibar2 = 0.0420
```

The estimates are almost exactly the same. It does not matter which output we report, but we will use the last one. See the discussion in *Setting the number of integration points* in [CM] **Intro 5** for more information.

◁

❑ Technical note

The LR `test vs. fixed parameters` is a test of `sd(deductible)` $= 0$. This 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 0) and a χ^2_1, denoted as $\overline{\chi}^2_{01}$. See Gutierrez, Carter, and Drukker (2001) for more details.

❑

Based on the model we just fit and assuming that we have a random or otherwise representative sample of individuals, we can use `margins` to estimate the proportion of people who will select each insurance plan.

```
. margins
Predictive margins                                      Number of obs = 1,250
Model VCE: OIM
Expression: Pr(insurance), predict()
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome						
Health	.2039552	.0224378	9.09	0.000	.1599779	.2479325
HCorp	.2573735	.0240836	10.69	0.000	.2101705	.3045764
SickInc	.219662	.0223481	9.83	0.000	.1758606	.2634634
MGroup	.187416	.0218151	8.59	0.000	.1446592	.2301729
MoonHealth	.1315932	.0192139	6.85	0.000	.0939346	.1692518

We expect about 26% of individuals to select the `HCorp` insurance plan. Suppose that `HCorp` is planning to increase their premiums by 10%. What effect would we expect this to have on the percentage of individuals selecting each alternative? We estimate this by using the `at(premium=generate(premium*1.10))` and `alternative(HCorp)` options with `margins`.

```
. margins, at(premium=generate(premium*1.10)) alternative(HCorp)
Predictive margins                                      Number of obs = 1,250
Model VCE: OIM
Expression:  Pr(insurance), predict()
Alternative: HCorp
At: premium = premium*1.10
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome						
Health	.2287178	.0240772	9.50	0.000	.1815274	.2759083
HCorp	.1751858	.0188653	9.29	0.000	.1382104	.2121612
SickInc	.2439653	.0238211	10.24	0.000	.1972769	.2906538
MGroup	.2073507	.0233282	8.89	0.000	.1616282	.2530732
MoonHealth	.1447803	.020706	6.99	0.000	.1041972	.1853634

If `HCorp` makes this planned increase to their premiums, we expect that only about 18% of individuals would select their plan.

We can use `margins` to answer many other interesting questions as well. See [CM] **Intro 1** and [CM] **margins** for more information and examples.

Example 2: Correlated random coefficients

Continuing with example 1, we now assume that preferences for premiums also vary, and we estimate the parameters of a model with random coefficients on both `premium` and `deductible`. We allow the random coefficients to be correlated, assuming a multivariate normal distribution, by specifying `random(deductible premium, correlated)`.

```
. cmmixlogit choice, random(deductible premium, correlated)
Fitting fixed parameter model:
Fitting full model:
Iteration 0:  Log simulated-likelihood = -295.85468  (not concave)
Iteration 1:  Log simulated-likelihood =  -295.8229
Iteration 2:  Log simulated-likelihood = -294.48628  (not concave)
Iteration 3:  Log simulated-likelihood =  -294.3334
Iteration 4:  Log simulated-likelihood = -294.20658
Iteration 5:  Log simulated-likelihood = -294.04148
Iteration 6:  Log simulated-likelihood = -294.03592
Iteration 7:  Log simulated-likelihood = -294.03592
Refining estimates:
Iteration 0:  Log simulated-likelihood = -294.03592
Iteration 1:  Log simulated-likelihood = -294.03592
Mixed logit choice model                        Number of obs      =      1,250
Case ID variable: id                            Number of cases    =        250
Alternatives variable: insurance                Alts per case: min =          5
                                                               avg =        5.0
                                                               max =          5
Integration sequence:      Hammersley
Integration points:               588           Wald chi2(2)       =      43.46
Log simulated-likelihood = -294.03592           Prob > chi2        =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
insurance						
deductible	-1.323078	.4620074	-2.86	0.004	-2.228596	-.4175604
premium	-3.043368	.4667707	-6.52	0.000	-3.958222	-2.128514
/Normal						
sd(deducti~e)	1.250886	.6835422			.4286292	3.650512
corr(deduc~e, premium)	.5492364	.5488277	1.00	0.317	-.7273401	.9736263
sd(premium)	1.066745	.6118013			.3466394	3.28279
Health						
_cons	.4995312	.3252348	1.54	0.125	-.1379174	1.13698
HCorp	(base alternative)					
SickInc						
_cons	-.8885082	.3081869	-2.88	0.004	-1.492544	-.2844729
MGroup						
_cons	-2.302517	.5068546	-4.54	0.000	-3.295934	-1.309101
MoonHealth						
_cons	-3.692866	.8062486	-4.58	0.000	-5.273085	-2.112648

```
LR test vs. fixed parameters: chi2(3) =       5.01        Prob > chi2 = 0.1713
Note: LR test is conservative and provided only for reference.
```

The estimated means of the random coefficients on `deductible` and `premium` are −1.32 and −3.04, respectively. Beneath the estimated means, we see the estimated standard deviations of the random coefficients and their estimated correlation, which are 1.25, 1.07, and 0.55, respectively. The high standard errors on these parameters indicate that they are not precisely estimated.

◁

▷ Example 3: Lognormal random coefficients and case-specific variables

In the previous example, we assumed that the coefficients on `premium` are normally distributed. Assuming a normal distribution implies that the random coefficients could be both positive and negative. However, it is typically more plausible to assume that increasing prices do not have positive effects on the probability of choosing a corresponding alternative. We can constrain the `premium` coefficients to be negative using a lognormal distribution.

Because the lognormal distribution is defined only over positive real values, the coefficient values coming from this distribution will only be positive. We want the coefficients to be negative, so we reverse the sign of `premium`. Then positive coefficients for `-premium` are the same as negative coefficients for `premium`.

```
. generate negpremium = -premium
```

We again assume a normal distribution for random coefficients on `deductible`, and we include `income` as a case-specific variable in the `casevars()` option.

```
. cmmixlogit choice, random(deductible) random(negpremium, lnormal)
> casevars(income)

Fitting fixed parameter model:

Fitting full model:

Iteration 0:  Log simulated-likelihood = -290.56142
Iteration 1:  Log simulated-likelihood = -288.84285
Iteration 2:  Log simulated-likelihood =  -288.7695
Iteration 3:  Log simulated-likelihood = -288.76877
Iteration 4:  Log simulated-likelihood = -288.76877

Mixed logit choice model                    Number of obs      =      1,250
Case ID variable: id                        Number of cases    =        250

Alternatives variable: insurance            Alts per case: min =          5
                                                           avg =        5.0
                                                           max =          5

Integration sequence:      Hammersley
Integration points:               579       Wald chi2(6)       =      84.14
Log simulated-likelihood = -288.76877       Prob > chi2        =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
insurance						
deductible	-1.141117	.3648257	-3.13	0.002	-1.856162	-.426072
negpremium	1.065557	.1175425	9.07	0.000	.8351775	1.295936
/Normal						
sd(deducti~e)	.7946425	.4502895			.2617175	2.412742
/Lognormal						
sd(negprem~m)	.3132935	.1398278			.1306312	.7513737
Health						
income	.1434246	.164893	0.87	0.384	-.1797597	.4666089
_cons	-.212565	.8944708	-0.24	0.812	-1.965695	1.540566
HCorp	(base alternative)					
SickInc						
income	-.3091895	.1557857	-1.98	0.047	-.6145239	-.003855
_cons	.6005832	.7992584	0.75	0.452	-.9659346	2.167101
MGroup						
income	-.3490833	.2014309	-1.73	0.083	-.7438807	.0457141
_cons	-.6984732	1.006771	-0.69	0.488	-2.671708	1.274762
MoonHealth						
income	-.506413	.2502777	-2.02	0.043	-.9969483	-.0158776
_cons	-1.463112	1.267173	-1.15	0.248	-3.946725	1.020502

```
LR test vs. fixed parameters: chi2(2) =       4.33        Prob > chi2 = 0.1149

Note: LR test is conservative and provided only for reference.
```

The estimated coefficient for `income` (our case-specific variable) shows that individuals are more likely to choose insurance plan `Health` over `HCorp` as income increases. The remaining alternatives are less likely to be chosen over `HCorp` as income increases. However, note that the coefficients on `income` for `Health` and for `MGroup` are not significantly different from 0 at the 5% level.

Because we assumed a lognormal distribution for the random coefficients on `negpremium`, we have to transform the estimated mean and standard deviation before we can interpret them. The lognormal distribution is parameterized in terms of the underlying normal distribution. The estimates shown

above are the mean and standard deviation of the natural logarithm of the `premium` coefficients. The mean of the coefficients is $e^{\mu+\sigma^2/2}$, and their standard deviation is $\sqrt{e^{2\mu+\sigma^2}(e^{\sigma^2}-1)}$. We can calculate point estimates and standard errors by using `nlcom`. To account for the reversed sign of `premium`, we multiply the mean by -1.

```
. local m _b[insurance:negpremium]
. local s _b[/Lognormal:sd(negpremium)]
. nlcom mean: -exp(`m'+`s'^2/2)
       mean: -exp(_b[insurance:negpremium]+_b[/Lognormal:sd(negpremium)]^2/2)
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
mean	-3.048449	.4138273	-7.37	0.000	-3.859536	-2.237362

```
. nlcom sd: sqrt(exp(2*`m'+`s'^2)*(exp(`s'^2)-1)), nopval
         sd: sqrt(exp(2*_b[insurance:negpremium]+_b[/Lognormal:sd(negpremium)]
> ^2)*(exp(_b[/Lognormal:sd(negpremium)]^2)-1))
```

choice	Coefficient	Std. err.	[95% conf.	interval]
sd	.978981	.5431803	-.0856328	2.043595

We estimate a mean of -3.05, which in this case is almost the same result as in example 2, where we assumed a normal distribution for the `premium` coefficients. ◁

▷ Example 4: Random intercepts

At first glance, the random coefficient model may seem limiting. What if we believe there is correlation of the errors of the utilities across alternatives but we are unsure which of the alternative-specific variables to model with random coefficients and correlated distributions.

The constants in the model are alternative specific, and we can turn them into random intercepts. To model random intercepts, we simply add dummies for the alternatives to `random()` using factor-variable notation with the alternatives variable `insurance`; see [U] **11.4.3 Factor variables**. We specify `correlated` as a suboption to `random()` to estimate both standard deviations and correlations. The option `noconstant` is needed because otherwise we would have collinearity between the random intercepts means and automatically created constant terms.

```
. cmmixlogit choice deductible premium, casevars(income)
> random(i.insurance, correlated) noconstant intpoints(1000)
Fitting fixed parameter model:
Fitting full model:
Iteration 0:  Log simulated-likelihood = -290.97937  (not concave)
Iteration 1:  Log simulated-likelihood = -290.97164  (not concave)
Iteration 2:  Log simulated-likelihood = -289.97239  (not concave)
Iteration 3:  Log simulated-likelihood = -289.47635  (not concave)
Iteration 4:  Log simulated-likelihood = -289.19304  (not concave)
Iteration 5:  Log simulated-likelihood = -288.61518  (not concave)
Iteration 6:  Log simulated-likelihood =  -288.3586
Iteration 7:  Log simulated-likelihood = -287.72103
Iteration 8:  Log simulated-likelihood = -287.30414
Iteration 9:  Log simulated-likelihood = -287.29017
Iteration 10: Log simulated-likelihood = -287.28864
Iteration 11: Log simulated-likelihood = -287.28864
```

```
Refining estimates:
Iteration 0:  Log simulated-likelihood = -287.28864
Iteration 1:  Log simulated-likelihood = -287.28864

Mixed logit choice model                    Number of obs      =      1,250
Case ID variable: id                        Number of cases    =        250
Alternatives variable: insurance            Alts per case: min =          5
                                                           avg =        5.0
                                                           max =          5
Integration sequence:      Hammersley
Integration points:              1000       Wald chi2(10)      =      42.24
Log simulated-likelihood = -287.28864       Prob > chi2        =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
insurance						
deductible	-1.27205	.4228305	-3.01	0.003	-2.100782	-.4433171
premium	-3.385171	.5534678	-6.12	0.000	-4.469948	-2.300394
insurance						
HCorp	.7807315	1.264681	0.62	0.537	-1.697997	3.25946
SickInc	.9118253	1.367183	0.67	0.505	-1.767805	3.591455
MGroup	-.9128333	1.584654	-0.58	0.565	-4.018698	2.193032
MoonHealth	-1.210673	1.724469	-0.70	0.483	-4.590571	2.169224
/Normal						
sd(2.insur~e)	1.732887	.914604			.615904	4.875596
corr(2.ins~e, 3.insurance)	.8538065	.2610271	3.27	0.001	-.5495268	.9963905
corr(2.ins~e, 4.insurance)	.72224	.408129	1.77	0.077	-.6409974	.9886836
corr(2.ins~e, 5.insurance)	.8850927	.2495122	3.55	0.000	-.6956714	.9986672
sd(3.insur~e)	2.082722	1.116996			.7279793	5.958593
corr(3.ins~e, 4.insurance)	.2685434	.5308222	0.51	0.613	-.6889524	.8846027
corr(3.ins~e, 5.insurance)	.5291393	.4334075	1.22	0.222	-.5305008	.9434711
sd(4.insur~e)	2.680528	1.237726			1.084378	6.626133
corr(4.ins~e, 5.insurance)	.9534027	.1765808	5.40	0.000	-.9590828	.9999762
sd(5.insur~e)	2.368159	1.293019			.8121816	6.905076
Health						
income	.2345847	.2143513	1.09	0.274	-.1855362	.6547055
HCorp	(base alternative)					
SickInc						
income	-.3319407	.1783297	-1.86	0.063	-.6814604	.0175791
MGroup						
income	-.3150546	.2292982	-1.37	0.169	-.7644708	.1343617
MoonHealth						
income	-.461337	.2296455	-2.01	0.045	-.911434	-.01124

```
LR test vs. fixed parameters: chi2(10) =       7.29       Prob > chi2 = 0.6981
Note: LR test is conservative and provided only for reference.
```

Note that we also used `intpoints(1000)` with `cmmixlogit`. When we used the default value for `intpoints()`, the model would not converge. Because we specified `correlated` and there are

5 alternatives, we are estimating 4 standard deviations (the number of alternatives minus one because the base level is not estimated) and 6 correlations, for a total of 10 variance parameters. The default `intpoints()` was not large enough for this number of variance parameters with these data. See the discussion in *Setting the number of integration points* in [CM] **Intro 5**. This introduction also has another example of a random intercepts model.

◁

Stored results

`cmmixlogit` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(N_case)`	number of cases
`e(N_ic)`	N for Bayesian information criterion (BIC)
`e(N_clust)`	number of clusters
`e(k)`	number of parameters
`e(k_alt)`	number of alternatives
`e(k_casevars)`	number of case-specific variables
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test
`e(df_m)`	model degrees of freedom
`e(df_c)`	degrees of freedom, comparison test
`e(ll)`	log simulated-likelihood
`e(ll_c)`	log likelihood, comparison model
`e(const)`	constant indicator
`e(intpoints)`	number of raw integration points
`e(lsequence)`	length of each integration sequence
`e(intburn)`	starting sequence index
`e(chi2)`	χ^2
`e(chi2_c)`	χ^2, comparison test
`e(p)`	p-value for model test
`e(p_c)`	p-value for comparison test
`e(alt_min)`	minimum number of alternatives
`e(alt_avg)`	average number of alternatives
`e(alt_max)`	maximum number of alternatives
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	`1` if converged, `0` otherwise

Macros

`e(cmd)`	`cmmixlogit`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(caseid)`	name of case ID variable
`e(altvar)`	name of alternatives variable
`e(alteqs)`	alternative equation names
`e(alt#)`	alternative labels
`e(base)`	base alternative
`e(corrmetric)`	correlation metric for correlated random coefficients
`e(wtype)`	weight type
`e(wexp)`	weight expression
`e(marktype)`	`casewise` or `altwise`, type of markout
`e(key_N_ic)`	`cases`, key for N for Bayesian information criterion (BIC)
`e(title)`	title in estimation output
`e(clustvar)`	name of cluster variable
`e(chi2type)`	type of χ^2
`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(intmethod)	technique used to generate sequences
e(sequence)	type of sequences
e(mc_rngstate)	random-number state used
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(properties)	b V
e(predict)	program used to implement predict
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(altvals)	alternative values
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

cmmixlogit estimates the parameters of the mixed logit model by MSL. The probability that case i chooses alternative a, conditional on the random parameter $\boldsymbol{\beta}_i$, is

$$P_{ia}(\boldsymbol{\beta}) = \frac{e^{\mathbf{x}_{ia}\boldsymbol{\beta}_i+\mathbf{w}_{ia}\boldsymbol{\alpha}+\mathbf{z}_i\boldsymbol{\delta}_a}}{\sum_{a=1}^{A} e^{\mathbf{x}_{ia}\boldsymbol{\beta}_i+\mathbf{w}_{ia}\boldsymbol{\alpha}+\mathbf{z}_i\boldsymbol{\delta}_a}}$$

We get the unconditional choice probability, P_{ia}, by integrating over the mixing distribution $f(\boldsymbol{\beta})$:

$$P_{ia} = \int P_{ia}(\boldsymbol{\beta}) f(\boldsymbol{\beta}) d\boldsymbol{\beta} \tag{2}$$

This integral of dimension d, where d equals the number of random parameters, is approximated by simulation because it has no closed-form solution. The simulated likelihood for the ith case is

$$L_i = \sum_{a=1}^{A} d_{ia} \widehat{P}_{ia}$$

where d_{ia} is an indicator that takes on the value 1 for the chosen alternative and 0 otherwise. The overall log simulated-likelihood is then $\sum_{i=1}^{N} \ln L_i$. $\widehat{P}_{ia}$ are the simulated probabilities,

$$\widehat{P}_{ia} = \frac{1}{M} \sum_{m=1}^{M} P_{ia}(\boldsymbol{\beta}^m) \tag{3}$$

where $\boldsymbol{\beta}^m$ are the random parameters drawn from $f(\boldsymbol{\beta})$ and M is the number of random draws. Equation (3) is the computation used to approximate the probabilities in (2).

Computation of $\widehat{P}_{ia}$ is based on integration sequences where each point of the sequence is a draw from density $f(\boldsymbol{\beta})$. The underlying uniform sequences are either pseudo–random draws from the uniform density or deterministic sequences such as a Halton sequence. Using deterministic sequences leads to better coverage of the probability space and lower variance of the simulator and thus to having a smaller approximation error than pseudo–random sequences, given the same number of draws. `cmmixlogit` supports pseudo–random, Halton, and Hammersley sequences; see Drukker and Gates (2006) for details.

Using a higher M in (3) will produce a better approximation to the probabilities in (2), at the cost of increased computation time. M is a function of the number of raw integration points q, which may be specified using the `intpoints()` option. In the default method, $M = q$ is the number of draws used in the approximation in (3). In addition to the default method, `cmmixlogit` supports methods in which the draws are symmetric around a midpoint, known as unidimensional and multidimensional antithetic draws. These antithetic methods produce a better approximation to the probabilities in (2), at the cost of additional computation time; see Train (2009, sec. 9.3.1). For unidimensional antithetics, $M = 2q$ draws are used. For multidimensional antithetics on a problem with d random coefficients, $M = 2^d q$ draws are used.

Random coefficients with mean μ and scale parameter σ are simulated as follows,

$$\begin{aligned}
\boldsymbol{\beta}^m_{\text{normal}} &= \mu + \sigma\eta_i & \eta_i &\sim N(0,1)\\
\boldsymbol{\beta}^m_{\text{lognormal}} &= \exp(\mu + \sigma\eta_i) & \eta_i &\sim N(0,1)\\
\boldsymbol{\beta}^m_{\text{truncated_normal}} &= \mu + \sigma\eta_i & \eta_i &\sim \text{TN}(0,1,-1.96,1.96)\\
\boldsymbol{\beta}^m_{\text{uniform}} &= \mu + \sigma\eta_i & \eta_i &\sim U(-1,1)\\
\boldsymbol{\beta}^m_{\text{triangular}} &= \mu + \sigma\eta_i & \eta_i &\sim \triangle(-1,1)
\end{aligned}$$

where $N(\mu,\sigma^2)$ is the normal distribution, $\text{TN}(\mu,\sigma^2,a,b)$ is the truncated normal distribution with lower truncation point a and upper truncation point b, $U(a,b)$ is uniform over $[\,a,b\,]$, and $\triangle(a,b)$ is the triangular distribution over $[\,a,b\,]$.

Correlated random parameters drawn from the multivariate normal distribution are generated as $\boldsymbol{\beta}^m_{\text{MVN}} = \mathbf{M} + \mathbf{L}\eta_i$, where $\mathbf{M}$ is a vector of means, $\eta_i \sim N(\mathbf{0},\mathbf{I})$, and $\mathbf{L}$ is the Cholesky factor such that the variance–covariance matrix $\mathbf{V} = \mathbf{L}\mathbf{L}'$.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster` *clustvar*`)`. See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*. Specifying `vce(robust)` is equivalent to specifying `vce(cluster` *caseid*`)`, where *caseid* is the variable that identifies the cases.

References

Ben-Akiva, M., D. Bolduc, and J. Walker. 2001. Specification, identification, and estimation of the logit kernel (or continuous mixed logit) model. Manuscript, University of California, Berkeley. http://eml.berkeley.edu/reprints/misc/multinomial2.pdf.

Ben-Akiva, M., D. L. McFadden, M. Abe, U. Böckenholt, D. Bolduc, D. Gopinath, T. Morikawa, V. Ramaswamy, V. Rao, D. Revelt, and D. Steinberg. 1997. Modeling methods for discrete choice analysis. *Marketing Letters* 8: 273–286. https://doi.org/10.1023/A:1007956429024.

Brownstone, D., and K. E. Train. 1998. Forecasting new product penetration with flexible substitution patterns. *Journal of Econometrics* 89: 109–129. https://doi.org/10.1016/S0304-4076(98)00057-8.

Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.

Drukker, D. M., and R. Gates. 2006. Generating Halton sequences using Mata. *Stata Journal* 6: 214–228.

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.

Hole, A. R. 2007. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal* 7: 388–401.

McFadden, D. L., and K. E. Train. 2000. Mixed MNL models for discrete response. *Journal of Applied Econometrics* 15: 447–470. https://doi.org/10.1002/1099-1255(200009/10)15:5⟨447::AID-JAE570⟩3.0.CO;2-1.

Train, K. E. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. New York: Cambridge University Press.

Also see

[CM] **cmmixlogit postestimation** — Postestimation tools for cmmixlogit

[CM] **cmclogit** — Conditional logit (McFadden's) choice model

[CM] **cmmprobit** — Multinomial probit choice model

[CM] **cmset** — Declare data to be choice model data

[CM] **cmxtmixlogit** — Panel-data mixed logit choice model

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

[CM] **nlogit** — Nested logit regression

[SVY] **svy estimation** — Estimation commands for survey data

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

Title

cmmixlogit postestimation — Postestimation tools for cmmixlogit

Postestimation commands predict margins Methods and formulas
Also see

Postestimation commands

The following standard postestimation commands are available after `cmmixlogit`:

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`	adjusted predictions, predictive margins, and 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`	probabilities, 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 probabilities or linear predictions.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [, *statistic*]

`predict` [*type*] *stub*`*` [*if*] [*in*], `scores`

statistic	Description
Main	
`pr`	probability alternative is chosen; the default
`xb`	linear prediction

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`predict` omits missing values casewise if `cmmixlogit` used casewise deletion (the default); if `cmmixlogit` used alternativewise deletion (option `altwise`), `predict` uses alternativewise deletion.

Options for predict

Main

`pr`, the default, calculates the probability of choosing each alternative.

`xb` calculates the linear prediction.

`scores` calculates the scores for each coefficient in `e(b)`. This option requires a new variable list of length equal to the number of columns in `e(b)`. Otherwise, use the *stub*`*` syntax to have `predict` generate enumerated variables with prefix *stub*.

margins

Description for margins

`margins` estimates margins of response for probabilities and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`pr`	probability alternative is chosen; the default
`xb`	linear prediction
`scores`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For more details, see [CM] **margins**.

Methods and formulas

The predicted probability of case i choosing alternative a is

$$\widehat{P}_{ia} = \frac{1}{M}\sum_{m=1}^{M} P_{ia}(\boldsymbol{\beta}^m)$$

where M is the number of random draws and $P_{ia}(\boldsymbol{\beta}^m)$ are the logistic probabilities,

$$P_{ia}(\boldsymbol{\beta}^m) = \frac{e^{\mathbf{x}_{ia}\boldsymbol{\beta}_i^m + \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a}}{\sum_{a=1}^{A} e^{\mathbf{x}_{ia}\boldsymbol{\beta}_i^m + \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a}}$$

evaluated at the simulated coefficients $\boldsymbol{\beta}^m$. The linear predictions are

$$\frac{1}{M}\sum_{m=1}^{M} \mathbf{x}_{ia}\boldsymbol{\beta}_i^m + \mathbf{w}_{ia}\boldsymbol{\alpha} + \mathbf{z}_i\boldsymbol{\delta}_a$$

See *Methods and formulas* in [CM] **cmmixlogit** for details.

Also see

[CM] **cmmixlogit** — Mixed logit choice model

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

[U] 20 Estimation and postestimation commands

Title

cmmprobit — Multinomial probit choice model

Description Quick start Menu Syntax
Options Remarks and examples Stored results Methods and formulas
References Also see

Description

`cmmprobit` fits a multinomial probit (MNP) choice model that relaxes the independence of irrelevant alternatives (IIA) property that is characteristic of the `cmclogit` choice model and that is assumed by the MNP model fit by `mprobit`.

The command requires multiple observations for each case (representing one individual or decision maker), where each observation represents an alternative that may be chosen. `cmmprobit` allows two types of independent variables: alternative-specific variables, which vary across both cases and alternatives, and case-specific variables, which vary across only cases.

Quick start

Multinomial probit choice model of `y` on `x1` using `cmset` data

```
cmmprobit y x1
```

Same as above, and include case-specific covariate `x2`

```
cmmprobit y x1, casevars(x2)
```

Same as above, but with factor covariance structure of dimension 1

```
cmmprobit y x1, casevars(x2) factor(1)
```

Common correlation parameter in utility errors for all pairs of alternatives

```
cmmprobit y x1, correlation(exchangeable)
```

With the structural covariance parameterization

```
cmmprobit y x1, structural
```

All standard deviations of the utility errors constrained to 1

```
cmmprobit y x1, stddev(homoskedastic)
```

Menu

Statistics > Choice models > Multinomial probit model

Syntax

```
cmmprobit depvar [indepvars] [if] [in] [weight] [, options]
```

depvar equal to 1 identifies the chosen alternatives, whereas a 0 indicates the alternatives that were not chosen.

options	Description
Model	
`casevars(`*varlist*`)`	case-specific variables
`basealternative(`*#* | *lbl* | *str*`)`	alternative used for normalizing location
`scalealternative(`*#* | *lbl* | *str*`)`	alternative used for normalizing scale
`noconstant`	suppress the alternative-specific constant terms
`altwise`	use alternativewise deletion instead of casewise deletion
`constraints(`*constraints*`)`	apply specified linear constraints
Model 2	
`correlation(`*correlation*`)`	correlation structure of the utility errors
`stddev(`*stddev*`)`	variance structure of the utility errors
`factor(#)`	use the factor covariance structure with dimension #
`structural`	use the structural covariance parameterization; default is the differenced covariance parameterization
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `robust`, `cluster` *clustvar*, `opg`, `bootstrap`, or `jackknife`
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`notransform`	do not transform variance–covariance estimates to the standard deviation and correlation metric
`nocnsreport`	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
Integration	
`intmethod(`*seqtype*`)`	type of quasi–uniform or pseudo–uniform sequence
`intpoints(#)`	number of points in each sequence
`intburn(#)`	starting index in the Hammersley or Halton sequence
`intseed(`*code* | *#*`)`	pseudo–uniform random-number seed
`antithetics`	use antithetic draws
`nopivot`	do not use integration interval pivoting
`initbhhh(#)`	use the BHHH optimization algorithm for the first # iterations
`favor(speed | space)`	favor speed or space when generating integration points

Maximization

maximize_options	control the maximization process; seldom used
collinear	keep collinear variables
coeflegend	display legend instead of statistics

correlation	Description
unstructured	one correlation parameter for each pair of alternatives; correlations with the basealternative() are zero; the default
exchangeable	one correlation parameter common to all pairs of alternatives; correlations with the basealternative() are zero
independent	constrain all correlation parameters to zero
pattern *matname*	user-specified matrix identifying the correlation pattern
fixed *matname*	user-specified matrix identifying the fixed and free correlation parameters

stddev	Description
heteroskedastic	estimate standard deviation for each alternative; standard deviations for basealternative() and scalealternative() set to one
homoskedastic	all standard deviations are one
pattern *matname*	user-specified matrix identifying the standard deviation pattern
fixed *matname*	user-specified matrix identifying the fixed and free standard deviations

seqtype	Description
hammersley	Hammersley point set
halton	Halton point set
random	uniform pseudo–random point set

You must cmset your data before using cmmprobit; see [CM] **cmset**.
indepvars and *varlist* may contain factor variables; see [U] **11.4.3 Factor variables**.
bootstrap, by, collect, jackknife, and statsby are allowed; see [U] **11.1.10 Prefix commands**.
Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.
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

casevars(*varlist*) specifies the case-specific variables that are constant for each case(). If there are a maximum of J alternatives, there will be $J-1$ sets of coefficients associated with casevars().

`basealternative(#|lbl|str)` specifies the alternative used to normalize the level of utility. The base alternative may be specified as a number when the alternatives variable is numeric, as a label when it is numeric and has a value label, or as a string when it is a string variable. The standard deviation for the utility error associated with the base alternative is fixed to one, and its correlations with all other utility errors are set to zero. The default is the first alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` and `correlation()` options, the `basealternative()` will be implied by the fixed standard deviations and correlations in the matrix specifications. `basealternative()` cannot be equal to `scalealternative()`.

`scalealternative(#|lbl|str)` specifies the alternative used to normalize the scale of the utility. The scale alternative may be specified as a number, label, or string. The default is to use the second alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` option, the `scalealternative()` will be implied by the fixed standard deviations in the matrix specification. `scalealternative()` cannot be equal to `basealternative()`.

If a `fixed` or `pattern` matrix is given for the `stddev()` option, the base alternative and scale alternative are implied by the standard deviations and correlations in the matrix specifications, and they need not be specified in the `basealternative()` and `scalealternative()` options.

`noconstant` suppresses the $J-1$ alternative-specific constant terms.

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

Model 2

`correlation(`*correlation*`)` specifies the correlation structure of the utility (latent-variable) errors.

`correlation(unstructured)` is the most general and has $J(J-3)/2+1$ unique correlation parameters. This is the default unless `stddev()` or `structural` is specified.

`correlation(exchangeable)` provides for one correlation coefficient common to all utilities, except the utility associated with the `basealternative()` option.

`correlation(independent)` assumes that all correlations are zero.

`correlation(pattern` *matname*`)` and `correlation(fixed` *matname*`)` give you more flexibility in defining the correlation structure. See *Covariance structures* later in this entry for more information.

`stddev(`*stddev*`)` specifies the variance structure of the utility (latent-variable) errors.

`stddev(heteroskedastic)` is the most general and has $J-2$ estimable parameters. The standard deviations of the utility errors for the alternatives specified in `basealternative()` and `scalealternative()` are fixed to one.

`stddev(homoskedastic)` constrains all the standard deviations to equal one.

`stddev(pattern` *matname*`)` and `stddev(fixed` *matname*`)` give you added flexibility in defining the standard deviation parameters. See *Covariance structures* later in this entry for more information.

`factor(#)` requests that the factor covariance structure of dimension # be used. The `factor()` option can be used with the `structural` option but cannot be used with `stddev()` or `correlation()`. A $\# \times J$ (or $\# \times J-1$) matrix, $\mathbf{C}$, is used to factor the covariance matrix as $I+\mathbf{C}'\mathbf{C}$, where

I is the identity matrix of dimension J (or $J-1$). The column dimension of $\mathbf{C}$ depends on whether the covariance is structural or differenced. The row dimension of $\mathbf{C}$, #, must be less than or equal to floor$[\{J(J-1)/2-1\}/(J-2)]$ because there are only $J(J-1)/2-1$ identifiable covariance parameters. This covariance parameterization may be useful for reducing the number of covariance parameters that need to be estimated.

If the covariance is structural, the column of $\mathbf{C}$ corresponding to the base alternative contains zeros. The column corresponding to the scale alternative has a one in the first row and zeros elsewhere. If the covariance is differenced, the column corresponding to the scale alternative (differenced with the base) has a one in the first row and zeros elsewhere.

`structural` requests the $J \times J$ structural covariance parameterization instead of the default $J-1 \times J-1$ differenced covariance parameterization (the covariance of the utility errors differenced with that of the base alternative). The differenced covariance parameterization will achieve the same MSL regardless of the choice of `basealternative()` and `scalealternative()`. On the other hand, the structural covariance parameterization imposes more normalizations that may bound the model away from its maximum likelihood and thus prevent convergence with some datasets or choices of `basealternative()` and `scalealternative()`.

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***.

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `basealternative()` and `scalealternative()`.

Reporting

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

`notransform` prevents retransforming the Cholesky-factored covariance estimates to the correlation and standard deviation metric. `notransform` may not be specified on replay.

This option has no effect if `structural` is not specified because the default differenced covariance estimates have no interesting interpretation as correlations and standard deviations. `notransform` also has no effect if the `correlation()` and `stddev()` options are specified with anything other than their default values. Here it is generally not possible to factor the covariance matrix, so optimization is already performed using the standard deviation and correlation representations.

`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**.

Integration

`intmethod(hammersley | halton | random)` specifies the method of generating the point sets used in the quasi–Monte Carlo integration of the multivariate normal density. `intmethod(hammersley)`, the default, uses the Hammersley sequence; `intmethod(halton)` uses the Halton sequence; and `intmethod(random)` uses a sequence of uniform random numbers.

`intpoints(#)` specifies the number of points to use in the Monte Carlo integration. If option `intmethod(hammersley)` or `intmethod(halton)` is used, the default is 500 +

floor$[2.5\sqrt{N_c\{\ln(k+5)+v\}}]$, where N_c is the number of cases, k is the number of coefficients in the model, and v is the number of covariance parameters. If `intmethod(random)` is used, the number of points is the above times 2. Larger values of `intpoints()` provide better approximations of the log likelihood at the cost of additional computation time.

`intburn(#)` specifies where in the Hammersley or Halton sequence to start, which helps reduce the correlation between the sequences of each dimension. The default is `intburn(0)`. This option may not be specified with `intmethod(random)`.

`intseed(`*code*`|#)` specifies the seed to use for generating the uniform pseudo–random sequence. This option may be specified only with `intmethod(random)`. *code* refers to a string that records the state of the random-number generator `runiform()`; see [R] **set seed**. An integer value # may be used also. The default is to use the current seed value from Stata's uniform random-number generator, which can be obtained from `c(rngstate)`.

`antithetics` specifies that antithetic draws be used. The antithetic draw for the $J-1$ vector uniform-random variables, $\mathbf{x}$, is $1-\mathbf{x}$.

`nopivot` turns off integration interval pivoting. By default, `cmmprobit` will pivot the wider intervals of integration to the interior of the multivariate integration. This improves the accuracy of the quadrature estimate. However, discontinuities may result in the computation of numerical second-order derivatives using finite differencing (for the Newton–Raphson optimize technique, `tech(nr)`) when few simulation points are used, resulting in a non–positive-definite Hessian. `cmmprobit` uses the Broyden–Fletcher–Goldfarb–Shanno optimization algorithm, by default, which does not require computing the Hessian numerically using finite differencing.

`initbhhh(#)` specifies that the Berndt–Hall–Hall–Hausman (BHHH) algorithm be used for the initial # optimization steps. This option is the only way to use the BHHH algorithm along with other optimization techniques. The algorithm switching feature of `ml`'s `technique()` option cannot include `bhhh`.

`favor(speed|space)` instructs `cmmprobit` to favor either speed or space when generating the integration points. `favor(speed)` is the default. When favoring speed, the integration points are generated once and stored in memory, thus increasing the speed of evaluating the likelihood. This speed increase can be seen when there are many cases or when the user specifies many integration points, `intpoints(#)`. When favoring space, the integration points are generated repeatedly with each likelihood evaluation.

For unbalanced data, where the number of alternatives varies with each case, the estimates computed using `intmethod(random)` will vary slightly between `favor(speed)` and `favor(space)`. This is because the uniform sequences will not be identical, even when initiating the sequences using the same uniform seed, `intseed(`*code*`|#)`. For `favor(speed)`, `ncase` blocks of `intpoints(#)` $\times$ $J-2$ uniform points are generated, where J is the maximum number of alternatives. For `favor(space)`, the column dimension of the matrices of points varies with the number of alternatives that each case has.

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**.

The following options may be particularly useful in obtaining convergence with `cmmprobit`: `difficult`, `technique(`*algorithm_spec*`)`, `nrtolerance(#)`, `nonrtolerance`, and `from(`*init_specs*`)`.

If technique() contains more than one algorithm specification, bhhh cannot be one of them. To use the BHHH algorithm with another algorithm, use the initbhhh() option, and specify the other algorithm in technique().

Setting the optimization type to technique(bhhh) resets the default *vcetype* to vce(opg).

The default optimization technique is technique(bfgs).

The following options are available with cmmprobit 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
The multinomial probit model
Covariance structures
    Applying constraints to correlation parameters
Convergence problems

Introduction

cmmprobit fits a multinomial probit (MNP) choice model. The dependent variable is a 0/1 variable indicating which alternative was chosen by an individual or decision maker. The data for each decision maker compose one case, consisting of multiple Stata observations, one for each available alternative.

The choice model can include alternative-specific variables, which vary across both cases and alternatives, and case-specific variables, which vary across only cases. The MNP model allows the random-error term to have a multivariate normal distribution, which can be both heteroskedastic and correlated across the alternatives.

cmmprobit with its correlation() and stddev() options gives you many choices for modeling the covariance of the error term. Also important for specifying the covariance is setting one alternative to be the basealternative() and another to be the scalealternative(). The default choices (the default base alternative is the lowest value of the alternatives variable, and the default scale alternative is the second lowest) may be fine for your model, but you should understand their importance for specifying the covariance parameterization of the model. Note that when we talk about covariance parameters for cmmprobit models, we are referring to the correlation parameters set by correlation() and the standard deviation parameters set by stddev(). See *Covariance structures* below.

If there are no alternative-specific variables in your model, covariance parameters are not identifiable. For such a model to converge, you would need to use correlation(independent) and stddev(homoskedastic). A better alternative is to use mprobit, which is geared specifically toward models with only case-specific variables. See [R] **mprobit**.

An alternative to MNP that allows a nested correlation structure for the covariance is the nested logit model. See [CM] **nlogit**.

The multinomial probit model

In the MNP model, there are a set of J unordered alternatives, one of which is chosen by each decision maker. These outcomes are modeled by a regression on alternative-specific and case-specific covariates. Underlying the model are J utilities (latent variables),

$$\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j + \xi_{ij}$$

where i denotes cases and j denotes alternatives. $\mathbf{x}_{ij}$ is a $1 \times p$ vector of alternative-specific variables, $\boldsymbol{\beta}$ is a $p \times 1$ vector of parameters, $\mathbf{z}_i$ is a $1 \times q$ vector of case-specific variables, $\boldsymbol{\alpha}_j$ is a $q \times 1$ vector of parameters for the jth alternative, and $\boldsymbol{\xi}_i = (\xi_{i1}, \ldots, \xi_{iJ})$ is distributed as multivariate normal with mean zero and covariance matrix $\boldsymbol{\Omega}$.

The ith decision maker selects the alternative whose utility η_{ij} is highest.

Because the MNP model allows the covariance of $\boldsymbol{\xi}_i$ to have a general structure, it does not impose the IIA property inherent in multinomial logistic and conditional logistic models. The MNP model permits the probability of choosing one alternative over another to depend on the remaining alternatives.

For example, consider the choice of travel mode between two cities, air, train, bus, or car, as a function of the travel mode cost, travel time (alternative-specific variables), and an individual's income (a case-specific variable). The probability of choosing air travel over a bus may not be independent of the train alternative because both bus and train travel are public ground transportation. That is, the probability of choosing air travel is $\Pr(\eta_{\text{air}} > \eta_{\text{bus}}, \eta_{\text{air}} > \eta_{\text{train}}, \eta_{\text{air}} > \eta_{\text{car}})$, and the two events $\eta_{\text{air}} > \eta_{\text{bus}}$ and $\eta_{\text{air}} > \eta_{\text{train}}$ may be correlated.

The added flexibility of the MNP model does impose a significant computational burden because of the need to evaluate probabilities from the multivariate normal distribution. These probabilities are evaluated using a simulation technique because a closed-form solution does not exist. See *Methods and formulas* for more information.

Not all the J sets of regression coefficients $\boldsymbol{\alpha}_j$ are identifiable, nor are all $J(J+1)/2$ elements of the variance–covariance matrix $\boldsymbol{\Omega}$. As described by Train (2009, sec. 2.5), the model requires normalization because both the location (level) and scale of the utilities are irrelevant. Increasing each utility by a constant does not change which η_{ij} is the maximum for decision maker i, nor does multiplying them by a constant.

To normalize location, we choose an alternative, say, k, and take the difference between the utility k and the $J-1$ others,

$$\begin{aligned} v_{ijk} &= \eta_{ij} - \eta_{ik} \\ &= (\mathbf{x}_{ij} - \mathbf{x}_{ik})\boldsymbol{\beta} + \mathbf{z}_i(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_k) + \xi_{ij} - \xi_{ik} \\ &= \boldsymbol{\delta}_{ij'}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\gamma}_{j'} + \epsilon_{ij'} \\ &= \lambda_{ij'} + \epsilon_{ij'} \end{aligned} \tag{1}$$

where $j' = j$ if $j < k$ and $j' = j-1$ if $j > k$, so that $j' = 1, \ldots, J-1$.

We can now work with the $(J-1) \times (J-1)$ covariance matrix $\boldsymbol{\Sigma}_{(k)}$ for $\boldsymbol{\epsilon}_i' = (\epsilon_{i1}, \ldots, \epsilon_{i,J-1})$. The kth alternative here is the `basealternative()` in `cmmprobit`. From (1), the probability that decision maker i chooses alternative k, for example, is

$$\begin{aligned} \Pr(i \text{ chooses } k) &= \Pr(v_{i1k} \le 0, \ldots, v_{i,J-1,k} \le 0) \\ &= \Pr(\epsilon_{i1} \le -\lambda_{i1}, \ldots, \epsilon_{i,J-1} \le -\lambda_{i,J-1}) \end{aligned}$$

To normalize for scale, one of the diagonal elements of $\mathbf{\Sigma}_{(k)}$ must be fixed to a constant. In `cmmprobit`, this is the error variance for the alternative specified by `scalealternative()`. Thus, there are a total of, at most, $J(J-1)/2-1$ identifiable variance–covariance parameters. See *Covariance structures* below for more on this issue.

In fact, the model is slightly more general in that not all decision makers need to have faced all J alternatives. The model allows for situations in which the choice sets are unbalanced. That is, some decision makers chose among all possible alternatives, whereas other decision makers were given a choice among a subset of them, and perhaps other decision makers were given a choice among a different subset. The number of observations for each case is equal to the number of alternatives the decision maker faced.

▷ Example 1: Default covariance parameterization

Application of MNP models is common in the analysis of transportation data. Greene (2018, ex. 18.3, 839) uses travel-mode choice data between Sydney and Melbourne to demonstrate estimating parameters of various discrete choice models. The data contain information on 210 individuals' choices of travel mode. The four alternatives are air, train, bus, and car, with indices 1, 2, 3, and 4, respectively.

One alternative-specific variable is `travelcost`, a measure of generalized cost of travel that is equal to the sum of in-vehicle cost and a wagelike measure times the amount of time spent traveling. A second alternative-specific variable is the terminal time, `termtime`, which is zero for car transportation. Household income, `income`, is a case-specific variable.

```
. use https://www.stata-press.com/data/r18/travel
(Modes of travel)
. list id mode choice travelcost termtime income in 1/12, sepby(id)
```

	id	mode	choice	travel~t	termtime	income
1.	1	Air	0	70	69	35
2.	1	Train	0	71	34	35
3.	1	Bus	0	70	35	35
4.	1	Car	1	30	0	35
5.	2	Air	0	68	64	30
6.	2	Train	0	84	44	30
7.	2	Bus	0	85	53	30
8.	2	Car	1	50	0	30
9.	3	Air	0	129	69	40
10.	3	Train	0	195	34	40
11.	3	Bus	0	149	35	40
12.	3	Car	1	101	0	40

Before we can fit our MNP model, we must `cmset` our data. The first argument to `cmset` is the case ID variable. For these data, it is the variable `id`, which identifies individuals. The second argument is the alternatives variable, which identifies the alternatives that could have been chosen. In this instance, it is the variable `mode`, which gives the choices of travel mode.

We cmset the data and also run cmtab to see a tabulation of the chosen alternatives.

```
. cmset id mode
     Case ID variable: id
Alternatives variable: mode
. cmtab, choice(choice)
Tabulation of chosen alternatives (choice = 1)
```

Travel mode alternatives	Freq.	Percent	Cum.
Air	58	27.62	27.62
Train	63	30.00	57.62
Bus	30	14.29	71.90
Car	59	28.10	100.00
Total	210	100.00	

The model of travel choice is

$$\eta_{ij} = \beta_1 \texttt{travelcost}_{ij} + \beta_2 \texttt{termtime}_{ij} + \alpha_{1j} \texttt{income}_i + \alpha_{0j} + \xi_{ij}$$

The alternatives can be grouped as air and ground travel. With this in mind, we want the air alternative to be the basealternative() and choose train as the scaling alternative. Because these are the first and second alternatives in the mode variable, they are the defaults for basealternative() and scalealternative(), respectively.

```
. cmmprobit choice travelcost termtime, casevars(income)
Iteration 0:  Log simulated-likelihood = -201.33776
Iteration 1:  Log simulated-likelihood = -201.00395  (backed up)
Iteration 2:  Log simulated-likelihood = -200.80016  (backed up)
Iteration 3:  Log simulated-likelihood = -200.79329  (backed up)
Iteration 4:  Log simulated-likelihood = -200.54677  (backed up)
Iteration 5:  Log simulated-likelihood = -200.52959  (backed up)
Iteration 6:  Log simulated-likelihood = -197.81655
Iteration 7:  Log simulated-likelihood = -196.60511
Iteration 8:  Log simulated-likelihood = -195.23981
Iteration 9:  Log simulated-likelihood = -194.97557
Iteration 10: Log simulated-likelihood =  -193.8367
Iteration 11: Log simulated-likelihood = -192.71514
Iteration 12: Log simulated-likelihood = -192.62109
Iteration 13: Log simulated-likelihood = -192.29533
Iteration 14: Log simulated-likelihood = -191.79816
Iteration 15: Log simulated-likelihood =  -190.6148
Iteration 16: Log simulated-likelihood = -190.47993
Iteration 17: Log simulated-likelihood = -190.22168
Iteration 18: Log simulated-likelihood = -190.17784
Iteration 19: Log simulated-likelihood = -190.11022
Iteration 20: Log simulated-likelihood = -190.10726
Iteration 21: Log simulated-likelihood =  -190.0955
Iteration 22: Log simulated-likelihood = -190.09542
Iteration 23: Log simulated-likelihood = -190.09343
Iteration 24: Log simulated-likelihood =  -190.0933
Iteration 25: Log simulated-likelihood = -190.09323
Iteration 26: Log simulated-likelihood = -190.09322
```

```
Multinomial probit choice model                 Number of obs      =       840
Case ID variable: id                            Number of cases    =       210
Alternatives variable: mode                     Alts per case: min =         4
                                                               avg =       4.0
                                                               max =         4
Integration sequence:      Hammersley
Integration points:               600           Wald chi2(5)       =     32.16
Log simulated-likelihood = -190.09322           Prob > chi2        =    0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
mode						
travelcost	-.0097691	.0027817	-3.51	0.000	-.0152211	-.0043171
termtime	-.0377086	.0093869	-4.02	0.000	-.0561066	-.0193107
Air	(base alternative)					
Train						
income	-.0292031	.0089218	-3.27	0.001	-.0466894	-.0117168
_cons	.561912	.3945781	1.42	0.154	-.2114469	1.335271
Bus						
income	-.0127548	.00793	-1.61	0.108	-.0282973	.0027876
_cons	-.0572901	.4789444	-0.12	0.905	-.9960038	.8814236
Car						
income	-.0049142	.0077449	-0.63	0.526	-.0200939	.0102656
_cons	-1.832941	.8171904	-2.24	0.025	-3.434605	-.2312773
/lnl2_2	-.5490422	.3889427	-1.41	0.158	-1.311356	.2132714
/lnl3_3	-.6018061	.3355375	-1.79	0.073	-1.259447	.0558352
/l2_1	1.132598	.2125209	5.33	0.000	.7160651	1.549132
/l3_1	.971829	.2350542	4.13	0.000	.5111312	1.432527
/l3_2	.5201047	.2851798	1.82	0.068	-.0388374	1.079047

```
(mode=Air is the alternative normalizing location)
(mode=Train is the alternative normalizing scale)
```

By default, the differenced covariance parameterization is used, so the covariance matrix for this model is 3×3. There are two free variances to estimate and three correlations. To help ensure that the covariance matrix remains positive definite, `cmmprobit` uses the square root transformation, where it optimizes on the Cholesky-factored variance–covariance. To ensure that the diagonal elements of the Cholesky estimates remain positive, we use a log transformation.

The estimates labeled `/lnl2_2` and `/lnl3_3` in the coefficient table are the log-transformed diagonal elements of the Cholesky matrix. The estimates labeled `/l2_1`, `/l3_1`, and `/l3_2` are the off-diagonal entries for elements $(2,1)$, $(3,1)$, and $(3,2)$ of the Cholesky matrix.

The transformed parameters of the differenced covariance parameterization are difficult to interpret. You can view the untransformed covariance and correlation using the `estat` command. Typing

```
. estat covariance
```

	Train	Bus	Car
Train	2		
Bus	1.601736	1.616288	
Car	1.374374	1.401054	1.515069

```
Note: Covariances are for alternatives differenced with Air.
```

gives the covariance estimates, and typing

```
. estat correlation
```

	Train	Bus	Car
Train	1.0000		
Bus	0.8909	1.0000	
Car	0.7895	0.8953	1.0000

```
Note: Correlations are for alternatives differenced with Air.
```

gives the correlation estimates.

The pairwise correlations among the choices of train, bus, or car relative to the choice of air are all large. This is telling us that after controlling for travel cost and terminal time, the utilities for train, bus, and car relative to the utility of air are similar. To look more closely at the relative differences in utilities of train, bus, and car, we might want to make each one of them in turn the base alternative and examine those models.

After you have fit what you consider your final model, you should run the same model again, but this time setting `intpoints(#)`, the number of integration points in the simulated likelihood to a larger number. In this example, we see from the header that the default number of points was 600. We would run our model again using, say, 2000 points and see by how much the coefficient or covariance parameter estimates change. If changes are small compared with standard errors, we can have confidence in the numerical soundness of the simulation used to compute the likelihood. See *Setting the number of integration points* in [CM] **Intro 5** for more information.

◁

▷ Example 2: Reducing the number of covariance parameters

We can reduce the number of covariance parameters in the model by using the factor model by Cameron and Trivedi (2005). For large models with many alternatives, the parameter reduction can be dramatic, but for our example, we will use `factor(1)`, a one-dimension factor model, to reduce by 3 the number of parameters associated with the covariance matrix.

```
. cmmprobit choice travelcost termtime, casevars(income) factor(1)

Iteration 0:  Log simulated-likelihood = -201.33776
Iteration 1:  Log simulated-likelihood = -201.00464  (backed up)
Iteration 2:  Log simulated-likelihood = -200.80379  (backed up)
Iteration 3:  Log simulated-likelihood = -200.80009  (backed up)
Iteration 4:  Log simulated-likelihood = -200.56341  (backed up)
Iteration 5:  Log simulated-likelihood = -200.55104  (backed up)
Iteration 6:  Log simulated-likelihood = -198.68574
Iteration 7:  Log simulated-likelihood = -198.26038
Iteration 8:  Log simulated-likelihood = -197.71947
Iteration 9:  Log simulated-likelihood =  -197.7005
Iteration 10: Log simulated-likelihood = -197.32998
Iteration 11: Log simulated-likelihood = -196.87087
Iteration 12: Log simulated-likelihood = -196.85618
Iteration 13: Log simulated-likelihood = -196.85478
Iteration 14: Log simulated-likelihood = -196.85472
Iteration 15: Log simulated-likelihood = -196.85472

Multinomial probit choice model                 Number of obs      =       840
Case ID variable: id                            Number of cases    =       210

Alternatives variable: mode                     Alts per case: min =         4
                                                               avg =       4.0
                                                               max =         4

Integration sequence:      Hammersley
Integration points:               600           Wald chi2(5)       =    107.88
Log simulated-likelihood = -196.85472           Prob > chi2        =    0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
mode						
travelcost	-.0093706	.0036333	-2.58	0.010	-.0164918	-.0022494
termtime	-.0593265	.0064585	-9.19	0.000	-.0719849	-.0466682
Air	(base alternative)					
Train						
income	-.0373607	.0098226	-3.80	0.000	-.0566127	-.0181087
_cons	.1094541	.3949657	0.28	0.782	-.6646645	.8835727
Bus						
income	-.015879	.0112245	-1.41	0.157	-.0378785	.0061206
_cons	-1.082191	.4678666	-2.31	0.021	-1.999193	-.1651896
Car						
income	.0042621	.0092623	0.46	0.645	-.0138917	.0224159
_cons	-3.76572	.5541552	-6.80	0.000	-4.851844	-2.679596
/c1_2	1.182696	.3061336	3.86	0.000	.5826855	1.782707
/c1_3	1.228152	.3404839	3.61	0.000	.5608163	1.895489

```
(mode=Air is the alternative normalizing location)
(mode=Train is the alternative normalizing scale)
```

The estimates labeled /c1_2 and /c1_3 in the coefficient table are the factor loadings. These factor loadings produce the following differenced covariance estimates:

```
. estat covariance
```

	Train	Bus	Car
Train	2		
Bus	1.182696	2.398771	
Car	1.228152	1.452531	2.508358

Note: Covariances are for alternatives differenced with Air.

They also produce the following correlation estimates:

```
. estat correlation
```

	Train	Bus	Car
Train	1.0000		
Bus	0.5400	1.0000	
Car	0.5483	0.5922	1.0000

Note: Correlations are for alternatives differenced with Air.

◁

Covariance structures

The matrix $\mathbf{\Omega}$ has $J(J+1)/2$ distinct elements because it is symmetric. Selecting a base alternative, normalizing its error variance to one, and constraining the correlations between its error and the other errors reduce the number of estimable parameters by J. Moreover, selecting a scale alternative and normalizing its error variance to one also reduce the number by one. Hence, there are at most $m = J(J-1)/2 - 1$ estimable parameters in $\mathbf{\Omega}$.

In practice, estimating all m parameters can be difficult, so one must often place more restrictions on the parameters. The `cmmprobit` command provides the `correlation()` option to specify restrictions on the $J(J-3)/2+1$ correlation parameters not already restricted as a result of choosing the base alternatives, and it provides `stddev()` to specify restrictions on the $J-2$ standard deviations not already restricted as a result of choosing the base and scale alternatives.

When the `structural` option is used, `cmmprobit` fits the model by assuming that all m parameters can be estimated, which is equivalent to specifying `correlation(unstructured)` and `stddev(heteroskedastic)`. The unstructured correlation structure means that all $J(J-3)/2+1$ of the remaining correlation parameters will be estimated, and the heteroskedastic specification means that all $J-2$ standard deviations will be estimated. With these default settings, the log likelihood is maximized with respect to the Cholesky decomposition of $\mathbf{\Omega}$, and then the parameters are transformed to the standard deviation and correlation form.

The `correlation(exchangeable)` option forces the $J(J-3)/2+1$ correlation parameters to be equal, and `correlation(independent)` forces all the correlations to be zero. Using the `stddev(homoskedastic)` option forces all J standard deviations to be one. These options may help in obtaining convergence for a model if the default options do not produce satisfactory results. In fact, when you fit a complex model, it may be advantageous to first fit a simple model with only a few covariance parameters (that is, constraining some elements of the covariance) and then remove the constraints one at a time.

Advanced users may wish to specify alternative covariance structures of their own choosing, and the next few paragraphs explain how to do so.

`correlation(pattern` *matname*`)` allows you to give the name of a $J \times J$ matrix that identifies a correlation structure. Sequential positive integers starting at 1 are used to identify each correlation parameter. For example, if there are three correlation parameters, they are identified by 1, 2, and 3. The integers can be repeated to indicate that correlations with the same number should be constrained to be equal. A zero or a missing value (`.`) indicates that the correlation is to be set to zero. `cmmprobit` considers only the elements of the matrix *below* the main diagonal.

Suppose that you have a model with four alternatives, numbered 1, 2, 3, and 4, and alternative 1 is the base. The unstructured and exchangeable correlation structures identified in the 4×4 lower triangular matrices are

unstructured

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \cdot & & & \\ 0 & \cdot & & \\ 0 & 1 & \cdot & \\ 0 & 2 & 3 & \cdot \end{pmatrix} \end{array}$$

exchangeable

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \cdot & & & \\ 0 & \cdot & & \\ 0 & 1 & \cdot & \\ 0 & 1 & 1 & \cdot \end{pmatrix} \end{array}$$

In `cmmprobit`, these correlation structures correspond to `correlation(unstructured)` and `correlation(exchangeable)`, respectively, even though the correlations corresponding to the base alternative are set to zero. More formally: these terms are appropriate when considering the $(J-1)\times(J-1)$ submatrix $\mathbf{\Sigma}_{(k)}$ defined in *The multinomial probit model* above.

You can also use the `correlation(fixed` *matname*`)` option to specify a matrix that specifies fixed and free parameters. Here the free parameters (those that are to be estimated) are identified by a missing value, and nonmissing values represent correlations that are to be taken as given. Below is a correlation structure that would set the correlations of alternative 1 to be 0.5:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \cdot & & & \\ 0.5 & \cdot & & \\ 0.5 & \cdot & \cdot & \\ 0.5 & \cdot & \cdot & \cdot \end{pmatrix} \end{array}$$

The row and column numbers of the elements of the `pattern` and `fixed` matrices correspond to the order of the alternative levels.

To specify the structure of the standard deviations—the diagonal elements of $\mathbf{\Omega}$—you can use the `stddev(pattern` *matname*`)` option, where *matname* is a $1 \times J$ matrix. Sequential positive integers starting at 1 are used to identify each standard deviation parameter. The integers can be repeated to indicate that standard deviations with the same number are to be constrained to be equal. A missing value indicates that the corresponding standard deviation is to be set to one. Suppose that in example 1 with four alternatives, you wish to set the first and second standard deviations to one and constrain the third and fourth standard deviations to be equal. The following `pattern` matrix will do this:

$$\begin{array}{c} \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \cdot & \cdot & 1 & 1 \end{pmatrix} \end{array}$$

Using the `stddev(fixed` *matname*`)` option allows you to identify the fixed and free standard deviations. Fixed standard deviations are entered as positive real numbers, and free parameters are identified with missing values. For example, to constrain the first and second standard deviations to equal 0.5 and to allow the third and fourth to be estimated, you would use this `fixed` matrix:

$$\begin{array}{c} \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.5 & 0.5 & \cdot & \cdot \end{pmatrix} \end{array}$$

When supplying either the `pattern` or the `fixed` matrices, you must ensure that the model is properly scaled. At least two standard deviations must be constant for the model to be scaled. A warning is issued if `cmmprobit` detects that the model is not scaled.

▷ Example 3: Optional structural covariance parameterization

In example 1, we used the differenced covariance parameterization, the default. We now use the `structural` option to view the $J-2$ standard deviation estimates and the $(J-1)(J-2)/2$ correlation estimates. By default, `air` is the base alternative, so its standard deviation will be constrained to 1, and its correlations with the other alternatives will be constrained to 0. By default, `train` is the scale alternative, and its standard deviation will be constrained to 1.

```
. cmmprobit choice travelcost termtime, casevars(income) structural
Iteration 0:  Log simulated-likelihood = -201.33776
Iteration 1:  Log simulated-likelihood = -201.00424  (backed up)
Iteration 2:  Log simulated-likelihood = -200.80141  (backed up)
Iteration 3:  Log simulated-likelihood = -200.79567  (backed up)
Iteration 4:  Log simulated-likelihood = -200.55267  (backed up)
Iteration 5:  Log simulated-likelihood = -200.53715  (backed up)
Iteration 6:  Log simulated-likelihood = -199.76054
Iteration 7:  Log simulated-likelihood = -198.60402
Iteration 8:  Log simulated-likelihood = -197.81905
Iteration 9:  Log simulated-likelihood = -195.04024
Iteration 10: Log simulated-likelihood = -193.77392  (backed up)
Iteration 11: Log simulated-likelihood = -192.64657
Iteration 12: Log simulated-likelihood = -192.42437
Iteration 13: Log simulated-likelihood = -190.84776
Iteration 14: Log simulated-likelihood = -190.34744
Iteration 15: Log simulated-likelihood = -190.25259
Iteration 16: Log simulated-likelihood = -190.14042
Iteration 17: Log simulated-likelihood =  -190.1239
Iteration 18: Log simulated-likelihood = -190.11142
Iteration 19: Log simulated-likelihood = -190.10248
Iteration 20: Log simulated-likelihood = -190.09721
Iteration 21: Log simulated-likelihood = -190.09429
Iteration 22: Log simulated-likelihood = -190.09354
Iteration 23: Log simulated-likelihood = -190.09329
Iteration 24: Log simulated-likelihood = -190.09322
Iteration 25: Log simulated-likelihood = -190.09321
Reparameterizing to correlation metric and refining estimates
Iteration 0:  Log simulated-likelihood = -190.09321
Iteration 1:  Log simulated-likelihood = -190.09321
```

```
Multinomial probit choice model                 Number of obs      =       840
Case ID variable: id                            Number of cases    =       210
Alternatives variable: mode                     Alts per case: min =         4
                                                               avg =       4.0
                                                               max =         4
Integration sequence:      Hammersley
Integration points:               600           Wald chi2(5)       =     32.15
Log simulated-likelihood = -190.09321           Prob > chi2        =    0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
mode						
travelcost	-.009769	.0027817	-3.51	0.000	-.0152211	-.0043169
termtime	-.0377166	.0093909	-4.02	0.000	-.0561223	-.0193108
Air	(base alternative)					
Train						
income	-.0292123	.0089235	-3.27	0.001	-.046702	-.0117226
_cons	.5619727	.394604	1.42	0.154	-.211437	1.335382
Bus						
income	-.0127564	.0079299	-1.61	0.108	-.0282987	.0027858
_cons	-.057269	.4789789	-0.12	0.905	-.9960504	.8815123
Car						
income	-.0049143	.0077456	-0.63	0.526	-.0200954	.0102668
_cons	-1.833353	.8173825	-2.24	0.025	-3.435394	-.2313131
/lnsigma3	-.2422848	.492656	-0.49	0.623	-1.207873	.7233032
/lnsigma4	-.3315386	.6489737	-0.51	0.609	-1.603504	.9404266
/atanhr3_2	1.011829	.3874629	2.61	0.009	.2524152	1.771242
/atanhr4_2	.578313	.3936403	1.47	0.142	-.1932078	1.349834
/atanhr4_3	.8904187	.5589308	1.59	0.111	-.2050654	1.985903
sigma1	1	(base alternative)				
sigma2	1	(scale alternative)				
sigma3	.7848326	.3866525			.2988322	2.061231
sigma4	.7178185	.4658453			.2011904	2.561074
rho3_2	.7665173	.1598096			.2471877	.9437454
rho4_2	.5214382	.2866104			-.1908391	.874014
rho4_3	.7116005	.2759021			-.2022385	.963018

```
(mode=Air is the alternative normalizing location)
(mode=Train is the alternative normalizing scale)
. estimates store full
```

When comparing this output with that of example 1, we see that we have achieved the same log likelihood. That is, the structural parameterization using `air` as the base alternative and `train` as the scale alternative applied no restrictions on the model. This will not always be the case. We leave it up to you to try different base and scale alternatives, and you will see that not all the different combinations will achieve the same log likelihood. This is not true for the differenced covariance parameterization: it will always achieve the same log likelihood (and the maximum possible likelihood) regardless of the base and scale alternatives. This is why it is the default parameterization.

For an exercise, we can compute the differenced covariance displayed in example 1 by using the following ado-code.

```
. estat covariance
```

	Air	Train	Bus	Car
Air	1			
Train	0	1		
Bus	0	.6015877	.6159622	
Car	0	.3742979	.4008925	.5152634

```
. return list
matrices:
                r(cov) :  4 x 4
. matrix cov = r(cov)
. matrix M = (1,-1,0,0 \ 1,0,-1,0 \ 1,0,0,-1)
. matrix cov1 = M*cov*M'
. matrix list cov1
symmetric cov1[3,3]
           r1         r2         r3
r1          2
r2  1.6015877  1.6159622
r3  1.3742979  1.4008925  1.5152634
```

The slight difference in the regression coefficients between the results here and example 1 reflects the accuracy of the [M-5] **ghk( )** algorithm using 600 points from the Hammersley sequence. ◁

▷ Example 4: Exchangeable correlation matrix

We now fit a model with an exchangeable correlation matrix and compare this model with the one in example 3 using a likelihood-ratio test.

```
. cmmprobit choice travelcost termtime, casevars(income) correlation(exchangeable)
  (output omitted)
Multinomial probit choice model                 Number of obs      =       840
Case ID variable: id                            Number of cases    =       210
Alternatives variable: mode                     Alts per case: min =         4
                                                               avg =       4.0
                                                               max =         4
Integration sequence:      Hammersley
Integration points:               600           Wald chi2(5)       =     53.58
Log simulated-likelihood = -190.46413           Prob > chi2        =    0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
mode						
travelcost	-.008464	.0020451	-4.14	0.000	-.0124723	-.0044557
termtime	-.034535	.0072813	-4.74	0.000	-.0488061	-.020264
Air	(base alternative)					
Train						
income	-.0290381	.008323	-3.49	0.000	-.0453508	-.0127254
_cons	.552031	.3719408	1.48	0.138	-.1769595	1.281021
Bus						
income	-.0132539	.0074135	-1.79	0.074	-.027784	.0012762
_cons	-.0051467	.4338001	-0.01	0.991	-.8553792	.8450858
Car						
income	-.0060867	.0066376	-0.92	0.359	-.0190962	.0069228
_cons	-1.565633	.6632708	-2.36	0.018	-2.86562	-.2656464
/lnsigmaP1	-.3555	.1971535	-1.80	0.071	-.7419137	.0309138
/lnsigmaP2	-1.308023	.8859477	-1.48	0.140	-3.044449	.428402
/atanhrP1	1.116886	.3764578	2.97	0.003	.3790423	1.85473
sigma1	1	(base alternative)				
sigma2	1	(scale alternative)				
sigma3	.700823	.1381697			.4762017	1.031397
sigma4	.2703539	.2395194			.0476225	1.534803
rho3_2	.8064831	.131604			.3618755	.9521894
rho4_2	.8064831	.131604			.3618755	.9521894
rho4_3	.8064831	.131604			.3618755	.9521894

```
(mode=Air is the alternative normalizing location)
(mode=Train is the alternative normalizing scale)
. estat covariance
```

	Air	Train	Bus	Car
Air	1			
Train	0	1		
Bus	0	.5652019	.4911528	
Car	0	.2180358	.1528045	.0730912

```
. estat correlation
```

	Air	Train	Bus	Car
Air	1.0000			
Train	0.0000	1.0000		
Bus	0.0000	0.8065	1.0000	
Car	0.0000	0.8065	0.8065	1.0000

```
. lrtest full .
Likelihood-ratio test
Assumption: . nested within full
 LR chi2(2) =   0.74
Prob > chi2 = 0.6901
```

The likelihood-ratio test suggests that a common correlation is a plausible hypothesis, but this could be an artifact of the small sample size.

The labeling of the standard deviation and correlation estimates has changed from `/lnsigma` and `/atanhr`, in the previous example, to `/lnsigmaP` and `/atanhrP`. The "P" identifies the parameter's index in the pattern matrices used by `cmmprobit`. The pattern matrices are stored in `e(stdpattern)` and `e(corpattern)`.

Applying constraints to correlation parameters

Another way to fit the model with the exchangeable correlation structure in example 4 is to use the `constraint` command to define the constraints on the `/atanhr` correlation parameters explicitly and then apply those to `cmmprobit` with the `structural` option.

```
. constraint 1 [atanhr3_2]_cons = [atanhr4_2]_cons
. constraint 2 [atanhr3_2]_cons = [atanhr4_3]_cons
. cmmprobit choice travelcost termtime, casevars(income) constraints(1 2)
> structural
```

With this method, however, we must keep track of what correlation parameterizations are used in estimation, and that depends on the options specified.

◁

▷ Example 5: Specifying a pattern matrix for the correlation

In the last example, we used the `correlation(exchangeable)` option, reducing the number of correlation parameters from three to one. We can explore a two-parameter correlation model by specifying a pattern matrix in the `correlation()` option. Recall the description given in *Covariance structures*: pattern matrices are specified using missing values or zeros to indicate correlations set to zero, and a pattern of sequential integers $1, 2, \ldots$ to indicate correlations constrained to be equal. All correlations in positions with a 1 are made equal, all those marked with a 2 are equal, etc.

```
. matrix corpat = J(4, 4, .)
. matrix corpat[3,2] = 1
. matrix corpat[4,3] = 1
. matrix corpat[4,2] = 2
```

```
. matrix list corpat
corpat[4,4]
    c1  c2  c3  c4
r1   .   .   .   .
r2   .   .   .   .
r3   .   1   .   .
r4   .   2   1   .
. cmmprobit choice travelcost termtime, casevars(income)
> correlation(pattern corpat)
```

(*output omitted*)

```
Multinomial probit choice model                 Number of obs      =       840
Case ID variable: id                            Number of cases    =       210
Alternatives variable: mode                     Alts per case: min =         4
                                                               avg =       4.0
                                                               max =         4
Integration sequence:      Hammersley
Integration points:               600           Wald chi2(5)       =     40.14
Log simulated-likelihood = -190.11427           Prob > chi2        =    0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
mode						
travelcost	-.0099047	.0027288	-3.63	0.000	-.0152531	-.0045563
termtime	-.0385161	.0085542	-4.50	0.000	-.055282	-.0217501
Air	(base alternative)					
Train						
income	-.0294176	.0089726	-3.28	0.001	-.0470036	-.0118317
_cons	.5623591	.3985858	1.41	0.158	-.2188547	1.343573
Bus						
income	-.0126298	.0080906	-1.56	0.119	-.0284871	.0032276
_cons	-.0790918	.4743972	-0.17	0.868	-1.008893	.8507097
Car						
income	-.0048661	.0079038	-0.62	0.538	-.0203573	.0106251
_cons	-1.880932	.7879178	-2.39	0.017	-3.425223	-.3366416
/lnsigmaP1	-.169762	.3069501	-0.55	0.580	-.7713733	.4318492
/lnsigmaP2	-.2570832	.4928909	-0.52	0.602	-1.223132	.7089652
/atanhrP1	.9761636	.3285363	2.97	0.003	.3322443	1.620083
/atanhrP2	.5716999	.3792762	1.51	0.132	-.1716678	1.315067
sigma1	1	(base alternative)				
sigma2	1	(scale alternative)				
sigma3	.8438656	.2590247			.4623777	1.540103
sigma4	.7733039	.3811544			.2943071	2.031888
rho3_2	.7514003	.1430439			.320536	.9246362
rho4_2	.5166066	.278054			-.1700011	.865552
rho4_3	.7514003	.1430439			.320536	.9246362

```
(mode=Air is the alternative normalizing location)
(mode=Train is the alternative normalizing scale)
```

We display the covariance and correlation estimates:

```
. estat covariance
```

	Air	Train	Bus	Car
Air	1			
Train	0	1		
Bus	0	.6340809	.7121092	
Car	0	.3994939	.4903372	.5979989

```
. estat correlation
```

	Air	Train	Bus	Car
Air	1.0000			
Train	0.0000	1.0000		
Bus	0.0000	0.7514	1.0000	
Car	0.0000	0.5166	0.7514	1.0000

The alternative `air` is the `basealternative()`, so its standard deviation is 1 and its correlation with other alternatives is 0. Because `train` is the `scalealternative()`, its standard deviation is 1. The missing values in our pattern matrix were specified consistently with these settings.

The correlation of `bus` and `train` and the correlation of `bus` and `car` are equal because of the specification of "`1 1`" in the matrix `corpat`. There was only one "2" in `corpat`, so the correlation of `train` and `car` was unconstrained.

◁

Convergence problems

If you experience convergence problems, first try increasing `intpoints()`, the number of points used to compute the integral in the simulated likelihood. See *Setting the number of integration points* in [CM] **Intro 5**.

Second, examine your model specification, and reduce the number of variance and correlation parameters you are estimating. Start with a simple model with only one or two parameters, and then increase the number of parameters one by one. When the true variance for one or more elements of the variance–covariance matrix is zero, the model will have problems converging, as it should because the model is misspecified.

If you are using the `structural` option, note that changing the `basealternative()` and `scalealternative()` can affect convergence because different specifications for them can possibly specify different models.

Third, try using different starting values, such as convergent results from a simpler model. See the `from()` option in [R] **Maximize**.

Finally, you may want to try specifying `nopivot`, specifying `antithetics`, specifying `technique(nr)` with `difficult`, or specifying a switching algorithm in the `technique()` option. If you wish to use the BHHH algorithm along with another maximization algorithm in `technique()`, you must specify the `initbhhh(#)` option, where # is the number of BHHH iterations to use before switching to the algorithm specified in `technique()`. See `technique(`*algorithm_spec*`)` in [R] **Maximize**.

Stored results

cmmprobit stores the following in e():

Scalars

e(N)	number of observations
e(N_case)	number of cases
e(N_ic)	N for Bayesian information criterion (BIC)
e(N_clust)	number of clusters
e(k)	number of parameters
e(k_alt)	number of alternatives
e(k_indvars)	number of alternative-specific variables
e(k_casevars)	number of case-specific variables
e(k_sigma)	number of variance estimates
e(k_rho)	number of correlation estimates
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(df_m)	model degrees of freedom
e(ll)	log simulated-likelihood
e(const)	constant indicator
e(i_base)	base alternative index
e(i_scale)	scale alternative index
e(mc_points)	number of Monte Carlo replications
e(mc_burn)	starting sequence index
e(mc_antithetics)	antithetics indicator
e(chi2)	χ^2
e(p)	p-value for model test
e(fullcov)	unstructured covariance indicator
e(structcov)	1 if structured covariance, 0 otherwise
e(cholesky)	Cholesky-factored covariance indicator
e(alt_min)	minimum number of alternatives
e(alt_avg)	average number of alternatives
e(alt_max)	maximum number of alternatives
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	cmmprobit
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(caseid)	name of case ID variable
e(altvar)	name of alternatives variable
e(alteqs)	alternative equation names
e(alt#)	alternative labels
e(wtype)	weight type
e(wexp)	weight expression
e(marktype)	casewise or altwise, type of markout
e(key_N_ic)	cases, key for N for Bayesian information criterion (BIC)
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(correlation)	correlation structure
e(stddev)	variance structure
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(mc_method)	technique used to generate sequences
e(mc_rngstate)	random-number state used
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique

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(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(stats)	alternative statistics
e(stdpattern)	variance pattern
e(stdfixed)	fixed and free standard deviations
e(altvals)	alternative values
e(altfreq)	alternative frequencies
e(alt_casevars)	indicators for estimated case-specific coefficients—e(k_alt)×e(k_casevars)
e(corpattern)	correlation structure
e(corfixed)	fixed and free correlations
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:

Overview
Simulated likelihood

Overview

The simulated maximum-likelihood estimates for the MNP are obtained using ml; see [R] **ml**. The likelihood evaluator implements the Geweke–Hajivassiliou–Keane (GHK) algorithm to approximate the multivariate distribution function (Geweke 1989; Hajivassiliou and McFadden 1998; Keane and Wolpin 1994). The technique is also described in detail by Genz (1992), but Genz describes a more general algorithm where both lower and upper bounds of integration are finite. We briefly describe the GHK simulator and refer you to Bolduc (1999) for the score computations.

As discussed earlier, the utilities (latent variables) for a J-alternative model are $\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j + \xi_{ij}$, for $j = 1, \ldots, J$, $i = 1, \ldots, n$, and $\boldsymbol{\xi}_i' = (\xi_{i,1}, \ldots, \xi_{i,\mathrm{J}}) \sim \mathrm{MVN}(\mathbf{0}, \boldsymbol{\Omega})$. The experimenter observes alternative k for the ith observation if $k = \arg\max(\eta_{ij}, j = 1, \ldots, J)$. Let

$$
\begin{aligned}
v_{ij'} &= \eta_{ij} - \eta_{ik} \\
&= (\mathbf{x}_{ij} - \mathbf{x}_{ik})\boldsymbol{\beta} + \mathbf{z}_i(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_k) + \xi_{ij} - \xi_{ik} \\
&= \boldsymbol{\delta}_{ij'}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\gamma}_{j'} + \epsilon_{ij'}
\end{aligned}
$$

where $j' = j$ if $j < k$ and $j' = j - 1$ if $j > k$, so that $j' = 1, \ldots, J-1$. Further, $\boldsymbol{\epsilon}_i = (\epsilon_{i1}, \ldots, \epsilon_{i,J-1}) \sim \text{MVN}(\mathbf{0}, \boldsymbol{\Sigma}_{(k)})$. $\boldsymbol{\Sigma}$ is indexed by k because it depends on the choice made. We denote the deterministic part of the model as $\lambda_{ij'} = \boldsymbol{\delta}_{ij'}\boldsymbol{\beta} + \mathbf{z}_j\boldsymbol{\gamma}_{j'}$, and the probability of this event is

$$
\begin{aligned}
\Pr(y_i = k) &= \Pr(v_{i1} \le 0, \ldots, v_{i,J-1} \le 0) \\
&= \Pr(\epsilon_{i1} \le -\lambda_{i1}, \ldots, \epsilon_{i,J-1} \le -\lambda_{i,J-1}) \\
&= (2\pi)^{-(J-1)/2} \, |\boldsymbol{\Sigma}_{(k)}|^{-1/2} \int_{-\infty}^{-\lambda_{i1}} \cdots \int_{-\infty}^{-\lambda_{i,J-1}} \exp\left(-\frac{1}{2}\mathbf{z}'\boldsymbol{\Sigma}_{(k)}^{-1}\mathbf{z}\right) d\mathbf{z}
\end{aligned}
\tag{2}
$$

Simulated likelihood

For clarity in the discussion that follows, we drop the index denoting case so that for an arbitrary observation $\boldsymbol{\upsilon}' = (v_1, \ldots, v_{J-1})$, $\boldsymbol{\lambda}' = (\lambda_1, \ldots, \lambda_{J-1})$, and $\boldsymbol{\epsilon}' = (\epsilon_1, \ldots, \epsilon_{J-1})$.

The Cholesky-factored variance–covariance, $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}'$, is lower triangular,

$$
\mathbf{L} = \begin{pmatrix}
l_{11} & 0 & \ldots & 0 \\
l_{21} & l_{22} & \ldots & 0 \\
\vdots & \vdots & & \vdots \\
l_{J-1,1} & l_{J-1,2} & \ldots & l_{J-1,J-1}
\end{pmatrix}
$$

and the correlated utility errors can be expressed as linear functions of uncorrelated normal variates, $\boldsymbol{\epsilon} = \mathbf{L}\boldsymbol{\zeta}$, where $\boldsymbol{\zeta}' = (\zeta_1, \ldots, \zeta_{J-1})$ and $\zeta_j \sim \text{iid } N(0,1)$. We now have $\boldsymbol{\upsilon} = \boldsymbol{\lambda} + \mathbf{L}\boldsymbol{\zeta}$, and by defining

$$
z_j = \begin{cases}
-\dfrac{\lambda_1}{l_{11}} & \text{for } j = 1 \\
-\dfrac{\lambda_j + \sum_{i=1}^{j-1} l_{ji}\zeta_i}{l_{jj}} & \text{for } j = 2, \ldots, J-1
\end{cases}
\tag{3}
$$

we can express the probability statement (2) as the product of conditional probabilities

$$
\begin{aligned}
\Pr(y_i = k) = {} & \Pr(\zeta_1 \le z_1) \, \Pr(\zeta_2 \le z_2 \mid \zeta_1 \le z_1) \cdots \\
& \Pr(\zeta_{J-1} \le z_{J-1} \mid \zeta_1 \le z_1, \ldots, \zeta_{J-2} \le z_{J-2})
\end{aligned}
$$

because

$$
\begin{aligned}
\Pr(v_1 \le 0) &= \Pr(\lambda_1 + l_{11}\zeta_1 \le 0) \\
&= \Pr\left(\zeta_1 \le -\frac{\lambda_1}{l_{11}}\right) \\
\Pr(v_2 \le 0) &= \Pr(\lambda_2 + l_{21}\zeta_1 + l_{22}\zeta_2 \le 0) \\
&= \Pr\left(\zeta_2 \le -\frac{\lambda_2 + l_{21}\zeta_1}{l_{22}} \mid \zeta_1 \le -\frac{\lambda_1}{l_{11}}\right) \\
&\ldots
\end{aligned}
$$

The Monte Carlo algorithm then must make draws from the truncated standard normal distribution. It does so by generating $J-1$ uniform variates, $\delta_j, j = 1, \ldots, J-1$, and computing

$$\widetilde{\zeta}_j = \begin{cases} \Phi^{-1}\left\{\delta_1 \Phi\left(-\dfrac{\lambda_1}{l_{11}}\right)\right\} & \text{for } j = 1 \\ \Phi^{-1}\left\{\delta_j \Phi\left(\dfrac{-\lambda_j - \sum_{i=1}^{j-1} l_{ji}\widetilde{\zeta}_i}{l_{jj}}\right)\right\} & \text{for } j = 2, \ldots, J-1 \end{cases}$$

Define $\widetilde{z}_j$ by replacing $\widetilde{\zeta}_i$ for ζ_i in (3) so that the simulated probability for the lth draw is

$$p_l = \prod_{j=1}^{J-1} \Phi(\widetilde{z}_j)$$

To increase accuracy, the bounds of integration, λ_j, are ordered so that the largest integration intervals are on the inside. The rows and columns of the variance–covariance matrix are pivoted accordingly (Genz 1992).

For a more detailed description of the GHK algorithm in Stata, see Gates (2006).

Repeated draws are made, say, N, and the simulated likelihood for the ith case, denoted $\widehat{L}_i$, is computed as

$$\widehat{L}_i = \frac{1}{N}\sum_{l=1}^{N} p_l$$

The overall log simulated-likelihood is $\sum_i \log \widehat{L}_i$.

If the true likelihood is L_i, the error bound on the approximation can be expressed as

$$|\widehat{L}_i - L_i| \leq V(L_i) D_N\{(\delta_i)\}$$

where $V(L_i)$ is the total variation of L_i and D_N is the discrepancy, or nonuniformity, of the set of abscissas. For the uniform pseudo–random sequence, δ_i, the discrepancy is of order $O\{(\log\log N/N)^{1/2}\}$. The order of discrepancy can be improved by using quasi–random sequences.

Quasi–Monte Carlo integration is carried out by `cmmprobit` by replacing the uniform deviates with either the Halton or the Hammersley sequences. These sequences spread the points more evenly than the uniform random sequence and have a smaller order of discrepancy, $O\left[\{(\log N)^{J-1}\}/N\right]$ and $O\left[\{(\log N)^{J-2}\}/N\right]$, respectively. The Halton sequence of dimension $J-1$ is generated from the first $J-1$ primes, p_k, so that on draw l we have $\mathbf{h}_l = \{r_{p_1}(l), r_{p_2}(l), \ldots, r_{p_{J-1}}(l)\}$, where

$$r_{p_k}(l) = \sum_{j=0}^{q} b_{jk}(l) p_k^{-j-1} \in (0,1)$$

is the radical inverse function of l with base p_k so that $\sum_{j=0}^{q} b_{jk}(l) p_k^j = l$, where $p_k^q \leq l < p_k^{q+1}$ (Fang and Wang 1994).

This function is demonstrated with base $p_3 = 5$ and $l = 33$, which generates $r_5(33)$. Here $q = 2$, $b_{0,3}(33) = 3$, $b_{1,5}(33) = 1$, and $b_{2,5}(33) = 1$, so that $r_5(33) = 3/5 + 1/25 + 1/625$.

The Hammersley sequence uses an evenly spaced set of points with the first $J-2$ components of the Halton sequence

$$\mathbf{h}_l = \left\{ \frac{2l-1}{2N}, r_{p_1}(l), r_{p_2}(l), \ldots, r_{p_{J-2}}(l) \right\}$$

for $l = 1, \ldots, N$.

For a more detailed description of the Halton and Hammersley sequences, see Drukker and Gates (2006).

Computations for the derivatives of the simulated likelihood are taken from Bolduc (1999). Bolduc gives the analytical first-order derivatives for the log of the simulated likelihood with respect to the regression coefficients and the parameters of the Cholesky-factored variance–covariance matrix. `cmmprobit` uses these analytical first-order derivatives and numerical second-order derivatives.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster` *clustvar*`)`. See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*. Specifying `vce(robust)` is equivalent to specifying `vce(cluster` *caseid*`)`, where *caseid* is the variable that identifies the cases.

References

Bolduc, D. 1999. A practical technique to estimate multinomial probit models in transportation. *Transportation Research Part B* 33: 63–79. https://doi.org/10.1016/S0191-2615(98)00028-9.

Bunch, D. S. 1991. Estimability in the multinomial probit model. *Transportation Research Part B* 25: 1–12. https://doi.org/10.1016/0191-2615(91)90009-8.

Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.

Cappellari, L., and S. P. Jenkins. 2003. Multivariate probit regression using simulated maximum likelihood. *Stata Journal* 3: 278–294.

Drukker, D. M., and R. Gates. 2006. Generating Halton sequences using Mata. *Stata Journal* 6: 214–228.

Fang, K.-T., and Y. Wang. 1994. *Number-theoretic Methods in Statistics*. London: Chapman and Hall.

Gates, R. 2006. A Mata Geweke–Hajivassiliou–Keane multivariate normal simulator. *Stata Journal* 6: 190–213.

Genz, A. 1992. Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics* 1: 141–149. https://doi.org/10.1080/10618600.1992.10477010.

Geweke, J. 1989. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica* 57: 1317–1339. https://doi.org/10.2307/1913710.

Geweke, J., and M. P. Keane. 2001. Computationally intensive methods for integration in econometrics. In Vol. 5 of *Handbook of Econometrics*, ed. J. J. Heckman and E. E. Leamer, 3463–3568. Amsterdam: Elsevier. https://doi.org/10.1016/S1573-4412(01)05009-7.

Greene, W. H. 2018. *Econometric Analysis*. 8th ed. New York: Pearson.

Haan, P., and A. Uhlendorff. 2006. Estimation of multinomial logit models with unobserved heterogeneity using maximum simulated likelihood. *Stata Journal* 6: 229–245.

Hajivassiliou, V. A., and D. L. McFadden. 1998. The method of simulated scores for the estimation of LDV models. *Econometrica* 66: 863–896. https://doi.org/10.2307/2999576.

Hole, A. R. 2007. Fitting mixed logit models by using maximum simulated likelihood. *Stata Journal* 7: 388–401.

Keane, M. P., and K. I. Wolpin. 1994. The solution and estimation of discrete choice dynamic programming models by simulation and interpolation: Monte Carlo evidence. *Review of Economics and Statistics* 76: 648–672. https://doi.org/10.2307/2109768.

Luedicke, J. 2016a. Flexible discrete choice modeling using a multinomial probit model, part 1. *The Stata Blog: Not Elsewhere Classified*. http://blog.stata.com/2016/06/28/flexible-discrete-choice-modeling-using-a-multinomial-probit-model-part-1/.

——. 2016b. Flexible discrete choice modeling using a multinomial probit model, part 2. *The Stata Blog: Not Elsewhere Classified*. http://blog.stata.com/2016/07/05/flexible-discrete-choice-modeling-using-a-multinomial-probit-model-part-2/.

Train, K. E. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. New York: Cambridge University Press.

Also see

[CM] **cmmprobit postestimation** — Postestimation tools for cmmprobit

[CM] **cmclogit** — Conditional logit (McFadden's) choice model

[CM] **cmmixlogit** — Mixed logit choice model

[CM] **cmroprobit** — Rank-ordered probit choice model

[CM] **cmset** — Declare data to be choice model data

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

[CM] **nlogit** — Nested logit regression

[R] **mlogit** — Multinomial (polytomous) logistic regression

[R] **mprobit** — Multinomial probit regression

[U] 20 Estimation and postestimation commands

Title

cmmprobit postestimation — Postestimation tools for cmmprobit

Postestimation commands | predict | margins
estat | Remarks and examples | Also see

Postestimation commands

The following postestimation commands are of special interest after `cmmprobit`:

Command	Description
`estat covariance`	covariance matrix of the utility errors for the alternatives
`estat correlation`	correlation matrix of the utility errors for the alternatives
`estat facweights`	covariance factor weights matrix

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`	adjusted predictions, predictive margins, and 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`	probabilities, 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 probabilities, linear predictions, and standard errors.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [, *statistic*]

`predict` [*type*] *stub*`*` [*if*] [*in*], `scores`

statistic	Description
Main	
`pr`	probability alternative is chosen; the default
`xb`	linear prediction
`stdp`	standard error of the linear prediction

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`predict` omits missing values casewise if `cmmprobit` used casewise deletion (the default); if `cmmprobit` used alternativewise deletion (option `altwise`), `predict` uses alternativewise deletion.

Options for predict

Main

`pr`, the default, calculates the probability of choosing each alternative.

`xb` calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`scores` calculates the scores for each coefficient in `e(b)`. This option requires a new variable list of length equal to the number of columns in `e(b)`. Otherwise, use the *stub*`*` syntax to have `predict` generate enumerated variables with prefix *stub*.

margins

Description for margins

`margins` estimates margins of response for probabilities and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`pr`	probability alternative is chosen; the default
`xb`	linear prediction
`stdp`	not allowed with `margins`
`scores`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For more details, see [CM] **margins**.

estat

Description for estat

`estat covariance` computes the estimated variance–covariance matrix of the utility (latent-variable) errors for the alternatives. The estimates are displayed, and the variance–covariance matrix is stored in `r(cov)`.

`estat correlation` computes the estimated correlation matrix of the utility (latent-variable) errors for the alternatives. The estimates are displayed, and the correlation matrix is stored in `r(cor)`.

`estat facweights` displays the covariance factor weights matrix and stores it in `r(C)`.

Menu for estat

Statistics > Postestimation

Syntax for estat

Covariance matrix of the utility errors for the alternatives

```
estat covariance [ , format(%fmt) border(bspec) left(#) ]
```

Correlation matrix of the utility errors for the alternatives

```
estat correlation [ , format(%fmt) border(bspec) left(#) ]
```

Covariance factor weights matrix

```
estat facweights [ , format(%fmt) border(bspec) left(#) ]
```

`collect` is allowed with `estat covariance`; see [U] **11.1.10 Prefix commands**.

Options for estat covariance, estat correlation, and estat facweights

`format(%fmt)` sets the matrix display format. The default for `estat covariance` and `estat facweights` is `format(%9.0g)`; the default for `estat correlation` is `format(%9.4f)`.

`border(bspec)` sets the matrix display border style. The default is `border(all)`. See [P] **matlist**.

`left(#)` sets the matrix display left indent. The default is `left(2)`. See [P] **matlist**.

Remarks and examples

Remarks are presented under the following headings:

Predicted probabilities
Obtaining estimation statistics

Predicted probabilities

After fitting a multinomial probit choice model, you can use `predict` to obtain the simulated probabilities that an individual will choose each of the alternatives.

When evaluating the multivariate normal probabilities via Monte Carlo simulation, `predict` uses the same method to generate the random sequence of numbers as the previous call to `cmmprobit`. For example, if you specified `intmethod(Halton)` when fitting the model, `predict` also uses the Halton sequence.

In example 1 of [CM] **cmmprobit**, we fit a model of individuals' travel-mode choices. We can obtain the simulated probabilities that an individual chooses each alternative by using `predict`:

```
. use https://www.stata-press.com/data/r18/travel
(Modes of travel)
. cmset id mode
     Case ID variable: id
Alternatives variable: mode
. quietly cmmprobit choice travelcost termtime, casevars(income)
. predict prob
(option pr assumed; Pr(mode))
. list id mode prob choice in 1/12, sepby(id)
```

	id	mode	prob	choice
1.	1	Air	.1491488	0
2.	1	Train	.3291686	0
3.	1	Bus	.1319882	0
4.	1	Car	.3899048	1
5.	2	Air	.2565295	0
6.	2	Train	.2761068	0
7.	2	Bus	.0116262	0
8.	2	Car	.4557356	1
9.	3	Air	.2098824	0
10.	3	Train	.1082094	0
11.	3	Bus	.1671392	0
12.	3	Car	.5147675	1

Obtaining estimation statistics

Once you have fit a `cmmprobit` model, you can obtain the estimated variance or correlation matrices for the model alternatives by using the `estat` command.

To display the correlations of the errors in the utility equations, we type

```
. estat correlation
```

	Train	Bus	Car
Train	1.0000		
Bus	0.8909	1.0000	
Car	0.7895	0.8953	1.0000

Note: Correlations are for alternatives differenced with Air.

The covariance matrix can be displayed by typing

```
. estat covariance
```

	Train	Bus	Car
Train	2		
Bus	1.601736	1.616288	
Car	1.374374	1.401054	1.515069

```
Note: Covariances are for alternatives differenced with Air.
```

Also see

[CM] **cmmprobit** — Multinomial probit choice model

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

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

Title

cmrologit — Rank-ordered logit choice model

Description | Quick start | Menu | Syntax
Options | Remarks and examples | Stored results | Methods and formulas
Acknowledgment | References | Also see

Description

`cmrologit` fits the rank-ordered logistic regression model by maximum likelihood (Beggs, Cardell, and Hausman 1981). This model is also known as the Plackett–Luce model (Marden 1995), as the exploded logit model (Punj and Staelin 1978), and as the choice-based method of conjoint analysis (Hair et al. 2010).

Quick start

Rank-ordered logit model of rankings `y` on `x1`, `x2`, and `x3`, using `cmset` data

```
cmrologit y x1 x2 x3
```

Same as above, but interpret the lowest value of `y` as the best

```
cmrologit y x1 x2 x3, reverse
```

Use Efron's method for handling ties in rankings

```
cmrologit y x1 x2 x3, ties(efron)
```

With cluster–robust standard errors for clustering by levels of `cvar`

```
cmrologit y x1 x2 x3, vce(cluster cvar)
```

Menu

Statistics > Choice models > Rank-ordered logit model

Syntax

```
cmrologit depvar indepvars [if] [in] [weight] [, options]
```

options	Description
Model	
incomplete(#)	use # to code unranked alternatives; default is incomplete(0)
reverse	reverse the preference order
ties(*spec*)	method to handle ties: exactm, breslow, efron, or none
altwise	use alternativewise deletion instead of casewise deletion
notestrhs	keep right-hand-side variables that do not vary within case
offset(*varname*)	include *varname* in model with coefficient constrained to 1
SE/Robust	
vce(*vcetype*)	*vcetype* may be oim, robust, cluster *clustvar*, bootstrap, or jackknife
Reporting	
level(#)	set confidence level; default is level(95)
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 cmset your data before using cmrologit; see [CM] **cmset**.

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

bootstrap, by, collect, fp, jackknife, and statsby are allowed; see [U] **11.1.10 Prefix commands**.

Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.

fweights, iweights, and pweights are allowed, except no weights are allowed with ties(efron), and pweights are not allowed with ties(exactm); see [U] **11.1.6 weight**.

coeflegend does not appear in the dialog box.

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

Options

Model

incomplete(#) specifies the numeric value used to code alternatives that are not ranked. It is assumed that unranked alternatives are less preferred than the ranked alternatives (that is, the data record the ranking of the most preferred alternatives). It is not assumed that subjects are indifferent between the unranked alternatives. The default is incomplete(0).

reverse specifies that in the preference order, a higher number means a less attractive alternative. The default is that higher values indicate more attractive alternatives. The rank-ordered logit model is not symmetric in the sense that reversing the ordering simply leads to a change in the signs of the coefficients.

`ties(`*spec*`)` specifies the method for handling ties (indifference between alternatives) (see [ST] **stcox** for details):

`exactm`	exact marginal likelihood (default)
`breslow`	Breslow's method (default if `pweights` specified)
`efron`	Efron's method (default if robust VCE)
`none`	no ties allowed

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

`notestrhs` suppresses the test that the independent variables vary within (at least some of) the cases. Effects of variables that are always constant are not identified. For instance, a rater's gender cannot directly affect his or her rankings; it could affect the rankings only via an interaction with a variable that does vary over alternatives.

`offset(`*varname*`)`; 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`), 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***.

If `ties(exactm)` is specified, *vcetype* may be only `oim`, `bootstrap`, or `jackknife`.

Reporting

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

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

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

Remarks and examples

Remarks are presented under the following headings:

Overview
Examples
Comparing respondents
Incomplete rankings and ties
Clustered choice data
Comparison of cmrologit and clogit
On reversals of rankings

Overview

The rank-ordered logit model can be applied to analyze how decision makers combine attributes of alternatives into overall evaluations of the attractiveness of these alternatives. The model generalizes a version of McFadden's choice model, one where alternatives are not explicitly identified. It uses information about the comparison of alternatives, namely, how decision makers rank the alternatives rather than just specifying the alternative that they like best.

`cmrologit` expects the data to be in long form, similar to `clogit` (see [R] **clogit**), in which each of the ranked alternatives forms an observation. The distinction from a McFadden's choice model fit with `clogit` is that *depvar* in `cmrologit` records the rankings of the alternatives, whereas for `clogit`, *depvar* indicates a single chosen alternative by a value not equal to zero. If your data record only one preferred alternative for each case, `cmrologit` fits the same model as `clogit`.

`cmrologit` interprets equal values of *depvar* as ties. The ranking information may be incomplete "at the bottom" (least preferred alternatives). That is, unranked alternatives may be coded as 0 or as a common value that may be specified with the `incomplete()` option.

All observations related to an individual are linked together by the case ID variable that you specify in `cmset`. Alternatives are not explicitly identified by an indicator variable, so `cmset` is used with the `noalternatives` option. For example, if `id` is your case ID variable, to `cmset` your data before running `cmrologit`, you type

```
. cmset id, noalternatives
```

For details on `cmset`, see [CM] **cmset**.

Examples

A popular way to study employer preferences for characteristics of employees is the quasiexperimental "vignette method". As an example, we consider the research by de Wolf (2000) on the labor market position of social science graduates. This study addresses how the educational portfolio (for example, general skills versus specific knowledge) affects short-term and long-term labor-market opportunities.

De Wolf asked 22 human resource managers (the respondents) to rank order the 6 most suitable candidates of 20 fictitious applicants and to rank order these 6 candidates for 3 jobs, namely, 1) researcher, 2) management trainee, and 3) policy adviser. Applicants were described by 10 attributes, including their age, gender, details of their portfolio, and work experience. In this example, we analyze a subset of the data.

To simplify the output, we dropped, at random, 10 nonselected applicants per case. The resulting dataset includes 29 cases, consisting of 10 applicants each. The data are in long form: observations correspond to alternatives (the applications), and alternatives that figured in one decision task are identified by the variable `caseid`. We list the observations for `caseid==7`, in which the respondent considered applicants for a social-science research position.

```
. use https://www.stata-press.com/data/r18/evignet
(Vignet study employer prefs (Inge de Wolf 2000))
. list pref female age grades edufit workexp boardexp if caseid==7, noobs
```

pref	female	age	grades	edufit	workexp	boardexp
0	yes	28	A/B	no	none	no
0	no	25	C/D	yes	one year	no
0	no	25	C/D	yes	none	yes
0	yes	25	C/D	no	internship	yes
1	no	25	C/D	yes	one year	yes
2	no	25	A/B	yes	none	no
3	yes	25	A/B	yes	one year	no
4	yes	25	A/B	yes	none	yes
5	no	25	A/B	yes	internship	no
6	yes	28	A/B	yes	one year	yes

Here 6 applicants were selected. The rankings are stored in the variable `pref`, where a value of 6 corresponds to "best among the candidates", a value of 5 corresponds to "second-best among the candidates", etc. The applicants with a ranking of 0 were not among the best 6 candidates for the job. The respondent was not asked to express his or her preferences among these four applicants, but by the elicitation procedure, it is known that he or she ranks these four applicants below the 6 selected applicants.

The best candidate was a female, 28 years old, with education fitting the job, with good grades (A/B), with 1 year of work experience, and with experience being a board member of a fraternity, a sports club, etc. The profiles of the other candidates read similarly. Here the respondent completed the task; that is, he or she selected and rank ordered the 6 most suitable applicants. Sometimes the respondent performed only part of the task.

```
. list pref female age grades edufit workexp boardexp if caseid==18, noobs
```

pref	female	age	grades	edufit	workexp	boardexp
0	no	25	C/D	yes	none	yes
0	no	25	C/D	no	internship	yes
0	no	28	C/D	no	internship	yes
0	yes	25	A/B	no	one year	no
2	yes	25	A/B	no	none	yes
2	no	25	A/B	no	none	yes
2	no	25	A/B	no	one year	yes
5	no	25	A/B	no	none	yes
5	no	25	A/B	no	none	yes
5	yes	25	A/B	no	none	no

The respondent selected the six best candidates and segmented these six candidates into two groups: one group with the three best candidates and a second group of three candidates that were "still acceptable". The numbers 2 and 5, indicating these two groups, are arbitrary apart from the implied ranking of the groups. The ties between the candidates in a group indicate that the respondent was not able to rank the candidates within the group.

The purpose of the vignette experiment was to explore and test hypotheses about which of the employees' attributes are valued by employers, how these attributes are weighted depending on the type of job (described by variable `job` in these data), etc. In the psychometric tradition of Thurstone (1927), "value" is assumed to be linear in the attributes, with the coefficients expressing

the direction and weight of the attributes. In addition, it is assumed that "valuation" is to some extent a random procedure, captured by an additive random term. For instance, if value depends only on an applicant's age and gender, we would have

$$\text{value}(\texttt{female}_i, \texttt{age}_i) = \beta_1 \texttt{female}_i + \beta_2 \texttt{age}_i + \epsilon_i$$

where the random residual, ϵ_i, captures all omitted attributes. Thus, $\beta_1 > 0$ means that the employer assigns higher value to a woman than to a man.

Given this conceptualization of value, it is straightforward to model the decision (selection) among alternatives or the ranking of alternatives: the alternative with the highest value is selected (chosen), or the alternatives are ranked according to their value. To complete the specification of a model of choice and of ranking, we assume that the random residual ϵ_i follows an "extreme value distribution of type I", introduced in this context by Luce (1959). This specific assumption is made mostly for computational convenience.

This model is known by many names. Among others, it is known as the rank-ordered logit model in economics (Beggs, Cardell, and Hausman 1981), as the exploded logit model in marketing research (Punj and Staelin 1978), as the choice-based conjoint analysis model (Hair et al. 2010), and as the Plackett–Luce model (Marden 1995).

The model coefficients are estimated using the method of maximum likelihood. The implementation in `cmrologit` uses an analogy between the rank-ordered logit model and the Cox regression model observed by Allison and Christakis (1994); see *Methods and formulas*. The `cmrologit` command implements this method for rankings, whereas `clogit` deals with the variant of choices; that is, only the most highly valued alternative is recorded. In the latter case, the model is also known as the Luce–McFadden choice model. In fact, when the data record the most preferred (unique) alternative and no additional ranking information about preferences is available, `cmrologit` and `clogit` return the same information, though formatted somewhat differently.

Before we can fit our model, we must `cmset` our data. The argument to `cmset` is the case ID variable, which must be numeric. For these data, it is the variable `caseid`, which identifies respondents. Unlike other choice models, alternatives are not specified; that is, there is no variable identifying specific alternatives across respondents. Alternatives simply have characteristics in this model.

```
. cmset caseid, noalternatives
     Case ID variable: caseid
Alternatives variable: <none>
```

We fit our model:

```
. cmrologit pref i.female age i.grades i.edufit i.workexp i.boardexp if job==1
Iteration 0:  Log likelihood =  -95.41087
Iteration 1:  Log likelihood = -71.004807
Iteration 2:  Log likelihood = -68.045946
Iteration 3:  Log likelihood = -67.906223
Iteration 4:  Log likelihood = -67.905657
Refining estimates:
Iteration 0:  Log likelihood = -67.905657
Rank-ordered logit choice model                 Number of obs     =         80
Case ID variable: caseid                        Number of cases   =          8
Ties adjustment: No ties in data                Obs per case:
                                                              min =         10
                                                              avg =      10.00
                                                              max =         10
                                                LR chi2(7)        =      55.01
Log likelihood = -67.90566                      Prob > chi2       =     0.0000
```

pref	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
female						
yes	-.4554119	.3615579	-1.26	0.208	-1.164052	.2532285
age	-.0851138	.0822504	-1.03	0.301	-.2463216	.0760939
grades						
A/B	3.144718	.6200539	5.07	0.000	1.929434	4.360001
edufit						
yes	.7638799	.3613688	2.11	0.035	.0556102	1.47215
workexp						
internship	1.89448	.6298646	3.01	0.003	.6599679	3.128992
one year	2.9124	.6203927	4.69	0.000	1.696452	4.128347
boardexp						
yes	.8102527	.3972321	2.04	0.041	.031692	1.588813

Focusing only on the variables whose coefficients are significant at the 10% level (we are analyzing 8 respondents only!), we can write the estimated value of an applicant for a job of type 1 (research positions) as

```
value = 3.14*(A/B grades) + 0.76*edufit + 1.89*internship
        + 2.91*(1-year workexp) + 0.81*boardexp
```

Thus, employers prefer applicants for a research position (`job==1`) whose educational portfolio fits the job, who have better grades, who have more relevant work experience, and who have (extracurricular) board experience. They do not seem to care much about the sex and age of applicants, which is comforting.

Given these estimates of the valuation by employers, we consider the probabilities that each of the applications is ranked first. Under the assumption that the ϵ_i are independent and follow an extreme value type I distribution, Luce (1959) showed that the probability, π_i, that alternative i is valued higher than alternatives $2, \ldots, k$ can be written in the multinomial logit form

$$\pi_i = \Pr\{\text{value}_1 > \max(\text{value}_2, \ldots, \text{value}_m)\} = \frac{\exp(\text{value}_i)}{\sum_{j=1}^{k} \exp(\text{value}_i)}$$

The probability of observing a specific ranking can be written as the product of such terms, representing a sequential decision interpretation in which the rater first chooses the most preferred alternative, and then the most preferred alternative among the rest, etc.

The probabilities for alternatives to be ranked first are conveniently computed by `predict`.

```
. predict p if e(sample)
(option pr assumed; conditional probability that alternative is ranked first)
(210 missing values generated)
. sort caseid pref p
. list pref p grades edufit workexp boardexp if caseid==7, noobs
```

pref	p	grades	edufit	workexp	boardexp
0	.0021934	C/D	yes	none	yes
0	.0043086	C/D	no	internship	yes
0	.0051824	A/B	no	none	no
0	.0179498	C/D	yes	one year	no
1	.0403597	C/D	yes	one year	yes
2	.0226441	A/B	yes	none	no
3	.2642474	A/B	yes	one year	no
4	.0322894	A/B	yes	none	yes
5	.1505625	A/B	yes	internship	no
6	.4602626	A/B	yes	one year	yes

There clearly is a positive relation between the stated ranking and the predicted probabilities for alternatives to be ranked first, but the association is not perfect. In fact, we would not have expected a perfect association, because the model specifies a (nondegenerate) probability distribution over the possible rankings of the alternatives. These predictions for sets of 10 candidates can also be used to make predictions for subsets of the alternatives. For instance, suppose that only the last three candidates listed in this table would be available. According to parameter estimates of the rank-ordered logit model, the probability that the last of these candidates is selected equals $0.460/(0.032 + 0.151 + 0.460) = 0.715$.

Comparing respondents

The `cmrologit` model assumes that all respondents, HR managers in large public-sector organizations in The Netherlands, use the *same* valuation function; that is, they apply the same decision weights. This is the substantive interpretation of the assumption that the β's are constant between the respondents. To probe this assumption, we could test whether the coefficients vary between different groups of respondents. For a metric characteristic of the HR manager, such as `firmsize`, we can consider a trend model in the valuation weights,

$$\beta_{ij} = \alpha_{i0} + \alpha_{i1}\texttt{firmsize}_j$$

and we can test that the slopes α_{i1} of `firmsize` are zero.

```
. generate firmsize = employer
. cmrologit pref i.edufit i.grades i.workexp
> c.firmsize#(1.edufit 1.grades 1.workexp 2.workexp 1.boardexp)
> if job==1, nolog
Rank-ordered logit choice model                 Number of obs     =         80
Case ID variable: caseid                        Number of cases   =          8
Ties adjustment: No ties in data                Obs per case:
                                                              min =         10
                                                              avg =      10.00
                                                              max =         10
                                                LR chi2(9)        =      57.84
Log likelihood = -66.49266                      Prob > chi2       =     0.0000
```

pref	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
edufit						
yes	1.163862	1.134384	1.03	0.305	-1.059489	3.387213
grades						
A/B	6.657894	2.401567	2.77	0.006	1.950908	11.36488
workexp						
internship	2.346752	1.908085	1.23	0.219	-1.393026	6.08653
one year	2.78297	1.794268	1.55	0.121	-.7337314	6.299672
edufit# c.firmsize						
yes	-.0118484	.0708709	-0.17	0.867	-.1507529	.127056
grades# c.firmsize						
A/B	-.217949	.1277188	-1.71	0.088	-.4682733	.0323753
workexp# c.firmsize						
internship	-.0353952	.107952	-0.33	0.743	-.2469773	.1761869
one year	.0078179	.1125715	0.07	0.945	-.2128183	.228454
boardexp# c.firmsize						
yes	.0426636	.0235837	1.81	0.070	-.0035596	.0888868

```
. testparm c.firmsize#(i.edufit i.grades i.workexp i.boardexp)
 ( 1)  1.edufit#c.firmsize = 0
 ( 2)  1.grades#c.firmsize = 0
 ( 3)  1.workexp#c.firmsize = 0
 ( 4)  2.workexp#c.firmsize = 0
 ( 5)  1.boardexp#c.firmsize = 0
           chi2(  5) =    7.48
         Prob > chi2 =    0.1871
```

The Wald test that the slopes of the interacted `firmsize` variables are jointly zero provides no evidence upon which we would reject the null hypothesis; that is, we do not find evidence against the assumption of constant valuation weights of the attributes by firms of different size. We did not enter `firmsize` as a predictor variable. Characteristics of the decision-making agent do not vary between alternatives. Thus, an additive effect of these characteristics on the valuation of alternatives does not affect the agent's ranking of alternatives and his or her choice. Consequently, the coefficient of `firmsize` is not identified. `cmrologit` would in fact have diagnosed the problem and dropped

firmsize from the analysis. Diagnosing this problem can slow the estimation considerably; the test may be suppressed by specifying the notestrhs option.

Incomplete rankings and ties

cmrologit allows incomplete rankings and ties in the rankings as proposed by Allison and Christakis (1994). cmrologit permits rankings to be incomplete only "at the bottom"; namely, that the ranking of the least attractive alternatives for subjects may not be known—do not confuse this with the situation that a subject is indifferent between these alternatives. This form of incompleteness occurred in the example discussed here because the respondents were instructed to select and rank only the top six alternatives. It may also be that respondents refused to rank the alternatives that are very unattractive.

cmrologit does not allow other forms of incompleteness, for instance, data in which respondents indicate which of four cars they like best, and which one they like least, but not how they rank the two intermediate cars. Another example of incompleteness that cannot be analyzed with cmrologit is data in which respondents select the three alternatives they like best but are not requested to express their preferences among the three selected alternatives.

cmrologit also permits ties in rankings. cmrologit assumes that if a subject expresses a tie between two or more alternatives, he or she actually holds one particular strict preference ordering, but with all possibilities of a strict ordering consistent with the expressed weak ordering being equally probable.

For instance, suppose that a respondent ranks alternative 1 highest. He or she prefers alternatives 2 and 3 over alternative 4 and is indifferent between alternatives 2 and 3. We assume that this respondent has the strict preference ordering either $1 > 2 > 3 > 4$ or $1 > 3 > 2 > 4$, with both possibilities being equally likely. From a psychometric perspective, it may actually be more appropriate to also assume that the alternatives 2 and 3 are close; for instance, the difference between the associated valuations (utilities) is less than some threshold or minimally discernible difference. Computationally, however, this is a more demanding model.

Clustered choice data

We have seen that applicants with work experience are in a relatively favorable position. To test whether the effects of work experience vary between the jobs, we can include interactions between the type of job and the attributes of applicants. Such interactions can be obtained using factor variables.

Because some HR managers contributed data for more than one job, we cannot assume that their selection decisions for different jobs are independent. We can account for this by specifying the vce(cluster *clustvar*) option. By treating choice data as incomplete ranking data with only the most preferred alternative marked, cmrologit may be used to estimate the model parameters for clustered choice data.

```
. cmrologit pref job##(i.female i.grades i.edufit i.workexp),
> vce(cluster employer) nolog
note: 2.job 3.job omitted because of no within-case variance.

Rank-ordered logit choice model                 Number of obs     =        290
Case ID variable: caseid                        Number of cases   =         29

Ties adjustment: efron                          Obs per case:
                                                              min =         10
                                                              avg =      10.00
                                                              max =         10

                                                Wald chi2(15)     =      93.03
Log pseudolikelihood =   -296.08                Prob > chi2       =     0.0000

                              (Std. err. adjusted for 22 clusters in employer)
```

pref	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
job						
managemen..	0	(omitted)				
policy ad..	0	(omitted)				
female						
yes	-.2249088	.2396552	-0.94	0.348	-.6946244	.2448068
grades						
A/B	2.847569	.8087366	3.52	0.000	1.262475	4.432664
edufit						
yes	.6759788	.2318277	2.92	0.004	.2216049	1.130353
workexp						
internship	1.450487	.5518405	2.63	0.009	.3689	2.532075
one year	2.495849	.72438	3.45	0.001	1.07609	3.915608
job#female						
managemen.. #						
yes	.0227426	.4749959	0.05	0.962	-.9082322	.9537174
policy ad.. #						
yes	.1138279	.3559085	0.32	0.749	-.58374	.8113958
job#grades						
managemen.. #						
A/B	-2.381741	.9589772	-2.48	0.013	-4.261302	-.50218
policy ad.. #						
A/B	-1.92171	.8563522	-2.24	0.025	-3.60013	-.2432907
job#edufit						
managemen.. #						
yes	-.2427363	.4041935	-0.60	0.548	-1.034941	.5494684
policy ad.. #						
yes	-.2942018	.3578419	-0.82	0.411	-.9955589	.4071553
job#workexp						
managemen.. #						
internship	-1.183344	.7275346	-1.63	0.104	-2.609286	.2425972
managemen.. #						
one year	-1.495627	.7763694	-1.93	0.054	-3.017283	.0260292
policy ad.. #						
internship	-.6353285	.8809524	-0.72	0.471	-2.361963	1.091306
policy ad.. #						
one year	-.9522599	1.02415	-0.93	0.352	-2.959556	1.055036

The parameter estimates for the first job type are very similar to those that would have been obtained from an analysis isolated to these data. Differences are due only to an implied change in the method of handling ties. With clustered observations, `cmrologit` uses Efron's method. If we had specified the `ties(efron)` option with the separate analyses, then the parameter estimates would have been identical to the simultaneous results.

Another difference is that `cmrologit` now reports robust standard errors, adjusted for clustering within respondents. These could have been obtained for the separate analyses, as well by specifying the `vce(robust)` option. In fact, this option would also have forced `cmrologit` to switch to Efron's method as well.

Given the combined results for the three types of jobs, we can test easily whether the weights for the attributes of applicants vary between the jobs, in other words, whether employers are looking for different qualifications in applicants for different jobs. A Wald test for the equality hypothesis of no difference can be obtained with the `testparm` command:

```
. testparm job#(i.female i.grades i.edufit i.workexp)
 ( 1)  2.job#1.female = 0
 ( 2)  3.job#1.female = 0
 ( 3)  2.job#1.grades = 0
 ( 4)  3.job#1.grades = 0
 ( 5)  2.job#1.edufit = 0
 ( 6)  3.job#1.edufit = 0
 ( 7)  2.job#1.workexp = 0
 ( 8)  2.job#2.workexp = 0
 ( 9)  3.job#1.workexp = 0
 (10)  3.job#2.workexp = 0
           chi2( 10) =   18.01
         Prob > chi2 =    0.0548
```

We find only mild evidence that employers look for different qualities in candidates according to the job for which they are being considered.

❑ Technical note

Allison (1999) stressed that the comparison between groups of the coefficients of logistic regression is problematic, especially in its latent-variable interpretation. In many common latent-variable models, only the regression coefficients divided by the scale of the latent variable are identified. Thus, a comparison of logit regression coefficients between, say, men and women is meaningful only if one is willing to argue that the standard deviation of the latent residual does not differ between the sexes.

The rank-ordered logit model is also affected by this problem. While we formulated the model with a scale-free residual, we can actually think of the model for the value of an alternative as being scaled by the standard deviation of the random term, representing other relevant attributes of alternatives. Again, comparing attribute weights between jobs is meaningful to the extent that we are willing to defend the proposition that "all omitted attributes" are equally important for different kinds of jobs.

❑

Comparison of cmrologit and clogit

The rank-ordered logit model also has a sequential interpretation. A subject first chooses the best among the alternatives. Next, he or she selects the best alternative among the remaining alternatives, etc. The decisions at each of the subsequent stages are described by a conditional logit model, and a subject is assumed to apply the same decision weights at each stage.

Some authors have expressed concern that later choices may well be made more randomly than the first few decisions. A formalization of this idea is a heteroskedastic version of the rank-ordered logit model in which the scale of the random term increases with the number of decisions made (for example, Hausman and Ruud [1987]). This extended model is currently not supported by `cmrologit`. However, the hypothesis that the same decision weights are applied at the first stage and at later stages can be tested by applying a Hausman test.

First, we fit the rank-ordered logit model on the full ranking data for the first type of job.

```
. cmrologit pref age i.female i.edufit i.grades i.boardexp if job==1, nolog
Rank-ordered logit choice model                 Number of obs     =         80
Case ID variable: caseid                        Number of cases   =          8
Ties adjustment: No ties in data                Obs per case:
                                                              min =         10
                                                              avg =      10.00
                                                              max =         10
                                                LR chi2(5)        =      26.15
Log likelihood = -82.33537                      Prob > chi2       =     0.0001
```

pref	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
age	-.1160121	.0767137	-1.51	0.130	-.2663682	.0343439
female						
yes	-.0791295	.3354377	-0.24	0.814	-.7365753	.5783163
edufit						
yes	.2475177	.3186609	0.78	0.437	-.3770462	.8720817
grades						
A/B	1.866381	.4542095	4.11	0.000	.9761463	2.756615
boardexp						
yes	-.0418455	.3194106	-0.13	0.896	-.6678788	.5841878

Second, we save the estimates for later use with the `estimates` command.

```
. estimates store Ranking
```

Third, to estimate the decision weights on the basis of the most preferred alternatives only, we create a variable, `best`, that is 1 for the best alternatives and 0 otherwise. The `by` prefix is useful here.

```
. by caseid (pref), sort: generate best = (pref == pref[_N]) if job==1
(210 missing values generated)
```

By specifying `(pref)` with `by caseid`, we ensured that the data were sorted in increasing order on `pref` within `caseid`. Hence, the most preferred alternatives are last in the sort order. The expression `pref == pref[_N]` is true (1) for the most preferred alternatives, even if the alternative is not unique, and false (0) otherwise.

If the most preferred alternatives were sometimes tied, we could still fit the model for the based-alternatives-only data via `cmrologit`, but `clogit` would yield different results because it deals with ties in a less appropriate way for continuous valuations. To ascertain whether there are ties in the selected data regarding applicants for research positions, we can combine `by` with `assert`:

```
. by caseid (pref), sort: assert pref[_N-1] != pref[_N] if job==1
```

There are no ties. We can now fit the model on the choice data by using either `clogit` or `cmrologit`.

```
. cmrologit best age i.edufit i.grades i.boardexp if job==1, nolog
Rank-ordered logit choice model                 Number of obs     =         80
Case ID variable: caseid                        Number of cases   =          8
Ties adjustment: No ties in data                Obs per case:
                                                              min =         10
                                                              avg =      10.00
                                                              max =         10
                                                LR chi2(4)        =       4.32
Log likelihood = -16.25952                      Prob > chi2       =     0.3641
```

best	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
age	-.0552291	.1887951	-0.29	0.770	-.4252607	.3148024
edufit						
yes	-.049965	.7530442	-0.07	0.947	-1.525905	1.425975
grades						
A/B	1.505808	1.11493	1.35	0.177	-.6794136	3.69103
boardexp						
yes	.995195	.8461853	1.18	0.240	-.6632977	2.653688

```
. estimates store Choice
```

The same results, though with a slightly different formatted header, would have been obtained by using `clogit` on these data.

```
. clogit best age i.edufit i.grades i.boardexp if job==1, group(caseid) nolog
Conditional (fixed-effects) logistic regression         Number of obs =      80
                                                        LR chi2(4)    =    4.32
                                                        Prob > chi2   = 0.3641
Log likelihood = -16.259518                             Pseudo R2     = 0.1173
```

best	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
age	-.0552291	.1887951	-0.29	0.770	-.4252607	.3148024
edufit						
yes	-.049965	.7530442	-0.07	0.947	-1.525905	1.425975
grades						
A/B	1.505808	1.11493	1.35	0.177	-.6794136	3.69103
boardexp						
yes	.995195	.8461853	1.18	0.240	-.6632977	2.653688

The parameters of the ranking and choice models look different, but the standard errors based on the choice data are much larger. Are we estimating parameters with the ranking data that are different from those with the choice data? A Hausman test compares two estimators of a parameter. One of the estimators should be efficient under the null hypothesis, namely, that choosing the second-best alternative is determined with the same decision weights as the best, etc. In our case, the efficient estimator of the decision weights uses the ranking information. The other estimator should be consistent, even if the null hypothesis is false. In our application, this is the estimator that uses the first-choice data only.

```
. hausman Choice Ranking
                 ---- Coefficients ----
              |      (b)          (B)            (b-B)     sqrt(diag(V_b-V_B))
              |   Choice       Ranking       Difference       Std. err.
--------------+----------------------------------------------------------------
          age |  -.0552291    -.1160121         .060783        .1725068
    1.edufit  |   -.049965     .2475177        -.2974828        .6822982
    1.grades  |   1.505808     1.866381        -.3605724        1.018215
  1.boardexp  |    .995195    -.0418455         1.037041        .7835857
--------------------------------------------------------------------------------
                    b = Consistent under H0 and Ha; obtained from cmrologit.
      B = Inconsistent under Ha, efficient under H0; obtained from cmrologit.

Test of H0: Difference in coefficients not systematic

    chi2(4) = (b-B)'[(V_b-V_B)^(-1)](b-B)
            =   2.37
Prob > chi2 = 0.6672
```

We do not find evidence for misspecification. We have to be cautious, though, because Hausman-type tests are often not powerful, and the number of observations in our example is very small, which makes the quality of the method of the null distribution by a χ^2 test rather uncertain.

On reversals of rankings

The rank-ordered logit model has a property that you may find unexpected and even unfortunate. Compare two analyses with the rank-ordered logit model, one in which alternatives are ranked from "most attractive" to "least attractive", the other a reversed analysis in which these alternatives are ranked from "most unattractive" to "least unattractive". By unattractiveness, you probably mean just the opposite of attractiveness, and you expect that the weights of the attributes in predicting "attractiveness" to be minus the weights in predicting "unattractiveness". This is, however, *not* true for the rank-ordered logit model.

The assumed distribution of the random residual takes the form $F(\epsilon) = 1 - \exp\{\exp(-\epsilon)\}$. This distribution is right skewed. Therefore, slightly different models result from adding and subtracting the random residual, corresponding with high-to-low and low-to-high rankings. Thus, the estimated coefficients will differ between the two specifications, though usually not in an important way.

You may observe the difference by specifying the `reverse` option of `cmrologit`. Reversing the rank order makes rankings that are incomplete at the bottom become incomplete at the top. Only the first kind of incompleteness is supported by `cmrologit`. Thus, for this comparison, we exclude the alternatives that are not ranked, omitting the information that ranked alternatives are preferred over excluded ones.

```
. cmrologit pref grades edufit workexp boardexp if job==1 & pref!=0
  (output omitted)
. estimates store Original
. cmrologit pref grades edufit workexp boardexp if job==1 & pref!=0, reverse
  (output omitted)
. estimates store Reversed
```

```
. estimates table Original Reversed, stats(aic bic)
```

Variable	Original	Reversed
grades		
A/B	1.9842801	-1.2118836
edufit		
yes	-.16397461	-.11698355
workexp		
internship	.50190232	-.70220848
one year	2.3208572	-2.1518673
boardexp		
yes	.32930412	-.19747927
aic	97.817471	101.24092
bic	107.17348	110.59692

Thus, although the weights of the attributes for reversed rankings are indeed mostly of opposite signs, the magnitudes of the weights and their standard errors differ. Which one is more appropriate? We have no advice to offer here. The specific science of the problem will determine what is appropriate, though we would be surprised indeed if this helps here. Formal testing does not help much either because the models for the original and reversed rankings are not nested. The model-selection indices, such as the AIC and BIC, however, suggest that you stick to the rank-ordered logit model applied to the original ranking rather than to the reversed ranking.

Stored results

`cmrologit` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(N_case)`	number of cases
`e(N_ic)`	N for Bayesian information criterion (BIC)
`e(N_clust)`	number of clusters
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(ll_0)`	log likelihood of the null model (“all rankings are equiprobable”)
`e(chi2)`	χ^2
`e(r2_p)`	pseudo-R^2
`e(p)`	p-value for model test
`e(code_inc)`	value for incomplete preferences
`e(alt_min)`	minimum number of alternatives
`e(alt_avg)`	average number of alternatives
`e(alt_max)`	maximum number of alternatives
`e(rank)`	rank of `e(V)`
`e(converged)`	`1` if converged, `0` otherwise

Macros

`e(cmd)`	`cmrologit`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(caseid)`	name of case ID variable
`e(wtype)`	weight type
`e(wexp)`	weight expression
`e(marktype)`	`casewise` or `altwise`, type of markout
`e(key_N_ic)`	`cases`, key for N for Bayesian information criterion (BIC)

e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(offset)	linear offset variable
e(chi2type)	Wald or LR; type of model χ^2 test
e(reverse)	reverse, if specified
e(ties)	breslow, efron, exactm
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(properties)	b V
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(marginsdefault)	default predict() specification for 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
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

Allison and Christakis (1994) demonstrate that maximum likelihood estimates for the rank-ordered logit model can be obtained as the maximum partial-likelihood estimates of an appropriately specified Cox regression model for waiting time ([ST] **stcox**). In this analogy, a higher ranking of an alternative is formally equivalent to a higher hazard rate of failure. cmrologit uses stcox to fit the rank-ordered logit model based on such a specification of the data in Cox terms. A higher stated preference is represented by a shorter waiting time until failure. Incomplete rankings are dealt with via censoring. Moreover, decision situations (subjects) are to be treated as strata.

Finally, as proposed by Allison and Christakis, ties in rankings are handled by the marginal-likelihood method, specifying that all strict preference orderings consistent with the stated weak preference ordering are equally likely. The marginal-likelihood estimator is available in stcox via the exactm option. The methods of the marginal likelihood due to Breslow and Efron are also appropriate for the analysis of rank-ordered logit models. Because in most applications the number of ranked alternatives by one subject will be fairly small (at most, say, 20), the number of ties is small as well, and so you rarely will need to turn to methods to restrict computer time. Because the marginal-likelihood estimator in stcox does not support the cluster adjustment or pweights, you should use the Efron method in such cases.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using vce(robust) and vce(cluster *clustvar*). See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*. Specifying vce(robust) is equivalent to specifying vce(cluster *caseid*), where *caseid* is the variable that identifies the cases.

Acknowledgment

The cmrologit command was written by Jeroen Weesie of the Department of Sociology at Utrecht University, The Netherlands.

References

Allison, P. D. 1999. Comparing logit and probit coefficients across groups. *Sociological Methods and Research* 28: 186–208. https://doi.org/10.1177/0049124199028002003.

Allison, P. D., and N. Christakis. 1994. Logit models for sets of ranked items. In Vol. 24 of *Sociological Methodology*, ed. P. V. Marsden, 123–126. Oxford: Blackwell.

Beggs, S., S. Cardell, and J. A. Hausman. 1981. Assessing the potential demand for electric cars. *Journal of Econometrics* 17: 1–19. https://doi.org/10.1016/0304-4076(81)90056-7.

de Wolf, I. 2000. *Opleidingsspecialisatie en arbeidsmarktsucces van sociale wetenschappers*. Amsterdam: ThelaThesis.

Hair, J. F., Jr., W. C. Black, B. J. Babin, and R. E. Anderson. 2010. *Multivariate Data Analysis*. 7th ed. Upper Saddle River, NJ: Pearson.

Hausman, J. A., and P. A. Ruud. 1987. Specifying and testing econometric models for rank-ordered data. *Journal of Econometrics* 34: 83–104. https://doi.org/10.1016/0304-4076(87)90068-6.

Luce, R. D. 1959. *Individual Choice Behavior: A Theoretical Analysis*. New York: Dover.

Marden, J. I. 1995. *Analyzing and Modeling Rank Data*. London: Chapman and Hall.

McCullagh, P. 1993. Permutations and regression models. In *Probability Models and Statistical Analysis for Ranking Data*, ed. M. A. Fligner and J. S. Verducci, 196–215. New York: Springer.

Plackett, R. L. 1975. The analysis of permutations. *Applied Statistics* 24: 193–202. https://doi.org/10.2307/2346567.

Punj, G. N., and R. Staelin. 1978. The choice process for graduate business schools. *Journal of Marketing Research* 15: 588–598. https://doi.org/10.1177/002224377801500408.

Thurstone, L. L. 1927. A law of comparative judgment. *Psychological Reviews* 34: 273–286. https://doi.org/10.1037/h0070288.

Yellott, J. I., Jr. 1977. The relationship between Luce's choice axiom, Thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology* 15: 109–144. https://doi.org/10.1016/0022-2496(77)90026-8.

Also see

[CM] **cmrologit postestimation** — Postestimation tools for cmrologit

[CM] **cmclogit** — Conditional logit (McFadden's) choice model

[CM] **cmroprobit** — Rank-ordered probit choice model

[CM] **cmset** — Declare data to be choice model data

[R] **ologit** — Ordered logistic regression

[U] 20 Estimation and postestimation commands

Title

cmrologit postestimation — Postestimation tools for cmrologit

Postestimation commands predict margins Remarks and examples
Also see

Postestimation commands

The following postestimation commands are available after `cmrologit`:

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
`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`	probabilities, 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 probabilities, linear predictions, and standard errors.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [`,` *statistic* `nooffset`]

statistic	Description
Main	
`pr`	probability that alternatives are ranked first; the default
`xb`	linear prediction
`stdp`	standard error of the linear prediction

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`predict` omits missing values casewise if `cmrologit` used casewise deletion (the default); if `cmrologit` used alternativewise deletion (option `altwise`), `predict` uses alternativewise deletion.

Options for predict

Main

`pr`, the default, calculates the probability that alternatives are ranked first.

`xb` calculates the linear prediction.

`stdp` calculates the standard error of the linear prediction.

`nooffset` is relevant only if you specified `offset(`*varname*`)` for `cmrologit`. It modifies the calculations made by `predict` so that they ignore the offset variable; the linear prediction is treated as $\mathbf{x}_j\mathbf{b}$ rather than as $\mathbf{x}_j\mathbf{b} + \text{offset}_j$.

margins

Description for margins

`margins` estimates margins of response for linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [*options*]

statistic	Description
`xb`	linear prediction; the default
`pr`	not allowed with `margins`
`stdp`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

Because `cmrologit` does not explicitly identify alternatives (that is, there is no alternatives variable), the alternative-specific features of [CM] **margins** do not apply to `cmrologit`. See [R] **margins** for the full syntax of `margins` available after `cmrologit`.

Remarks and examples

See *Comparing respondents* and *Clustered choice data* in [CM] **cmrologit** for examples of the use of `testparm`, an alternative to the `test` command.

See *Comparison of cmrologit and clogit* and *On reversals of rankings* in [CM] **cmrologit** for examples of the use of `estimates`.

See *Comparison of cmrologit and clogit* in [CM] **cmrologit** for an example of the use of `hausman`.

Also see

[CM] **cmrologit** — Rank-ordered logit choice model

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

Title

cmroprobit — Rank-ordered probit choice model

Description Quick start Menu Syntax
Options Remarks and examples Stored results Methods and formulas
Reference Also see

Description

`cmroprobit` fits rank-ordered probit (ROP) models by using maximum simulated likelihood (MSL). The model allows you to relax the independence of irrelevant alternatives (IIA) property that is characteristic of the rank-ordered logistic model by estimating covariances between the error terms for the alternatives.

`cmroprobit` allows two types of independent variables: alternative-specific variables, in which the values of each variable vary with each alternative, and case-specific variables, which vary with each case.

The estimation technique of `cmroprobit` is nearly identical to that of `cmmprobit`, and the two routines share many of the same options; see [CM] **cmmprobit**.

Quick start

Rank-ordered probit model of rankings `y` as a function of `x1` using `cmset` data

```
cmroprobit y x1
```

Same as above, but interpret the lowest value of `y` as the best

```
cmroprobit y x1, reverse
```

Same as above, and include case-specific covariate `x2`

```
cmroprobit y x1, casevars(x2) reverse
```

Same as above, but with factor covariance structure of dimension 1

```
cmroprobit y x1, casevars(x2) reverse factor(1)
```

With all correlations of utility errors constrained to 0

```
cmroprobit y x1, correlation(independent)
```

With a common correlation parameter for all pairs of alternatives

```
cmroprobit y x1, correlation(exchangeable)
```

Menu

Statistics > Choice models > Rank-ordered probit model

Syntax

```
cmroprobit depvar [indepvars] [if] [in] [weight] [, options]
```

options	Description
Model	
casevars(*varlist*)	case-specific variables
reverse	interpret the lowest rank in *depvar* as the best; the default is the highest rank is the best
basealternative(# | *lbl* | *str*)	alternative used for normalizing location
scalealternative(# | *lbl* | *str*)	alternative used for normalizing scale
noconstant	suppress the alternative-specific constant terms
altwise	use alternativewise deletion instead of casewise deletion
constraints(*constraints*)	apply specified linear constraints
Model 2	
correlation(*correlation*)	correlation structure of the utility errors
stddev(*stddev*)	variance structure of the utility errors
factor(#)	use the factor covariance structure with dimension #
structural	use the structural covariance parameterization; default is the differenced covariance parameterization
SE/Robust	
vce(*vcetype*)	*vcetype* may be oim, robust, cluster *clustvar*, opg, bootstrap, or jackknife
Reporting	
level(#)	set confidence level; default is level(95)
notransform	do not transform variance–covariance estimates to the standard deviation and correlation metric
nocnsreport	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
Integration	
intmethod(*seqtype*)	type of quasi–uniform or pseudo–uniform sequence
intpoints(#)	number of points in each sequence
intburn(#)	starting index in the Hammersley or Halton sequence
intseed(*code* | #)	pseudo–uniform random-number seed
antithetics	use antithetic draws
nopivot	do not use integration interval pivoting
initbhhh(#)	use the BHHH optimization algorithm for the first # iterations
favor(speed | space)	favor speed or space when generating integration points

Maximization

maximize_options	control the maximization process; seldom used
collinear	keep collinear variables
coeflegend	display legend instead of statistics

correlation	Description
unstructured	one correlation parameter for each pair of alternatives; correlations with the basealternative() are zero; the default
exchangeable	one correlation parameter common to all pairs of alternatives; correlations with the basealternative() are zero
independent	constrain all correlation parameters to zero
pattern *matname*	user-specified matrix identifying the correlation pattern
fixed *matname*	user-specified matrix identifying the fixed and free correlation parameters

stddev	Description
heteroskedastic	estimate standard deviation for each alternative; standard deviations for basealternative() and scalealternative() set to one
homoskedastic	all standard deviations are one
pattern *matname*	user-specified matrix identifying the standard deviation pattern
fixed *matname*	user-specified matrix identifying the fixed and free standard deviations

seqtype	Description
hammersley	Hammersley point set
halton	Halton point set
random	uniform pseudo–random point set

You must cmset your data before using cmroprobit; see [CM] **cmset**.
indepvars and *varlist* may contain factor variables; see [U] **11.4.3 Factor variables**.
bootstrap, by, collect, jackknife, and statsby are allowed; see [U] **11.1.10 Prefix commands**.
Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.
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

casevars(*varlist*) specifies the case-specific variables that are constant for each case(). If there are a maximum of J alternatives, there will be $J-1$ sets of coefficients associated with casevars().

reverse directs cmroprobit to interpret the rank in *depvar* that is smallest in value as the most preferred alternative. By default, the rank that is largest in value is the most preferred alternative.

`basealternative(`*# | lbl | str*`)` specifies the alternative used to normalize the level of utility. The base alternative may be specified as a number when the alternatives variable is numeric, as a label when it is numeric and has a value label, or as a string when it is a string variable. The standard deviation for the utility error associated with the base alternative is fixed to one, and its correlations with all other utility errors are set to zero. The default is the first alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` and `correlation()` options, the `basealternative()` will be implied by the fixed standard deviations and correlations in the matrix specifications. `basealternative()` cannot be equal to `scalealternative()`.

`scalealternative(`*# | lbl | str*`)` specifies the alternative used to normalize the scale of the utility. The scale alternative may be specified as a number, label, or string. The default is to use the second alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` option, the `scalealternative()` will be implied by the fixed standard deviations in the matrix specification. `scalealternative()` cannot be equal to `basealternative()`.

If a `fixed` or `pattern` matrix is given for the `stddev()` option, the base alternative and scale alternative are implied by the standard deviations and correlations in the matrix specifications, and they need not be specified in the `basealternative()` and `scalealternative()` options.

`noconstant` suppresses the $J-1$ alternative-specific constant terms.

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

Model 2

`correlation(`*correlation*`)` specifies the correlation structure of the utility (latent-variable) errors.

`correlation(unstructured)` is the most general and has $J(J-3)/2+1$ unique correlation parameters. This is the default unless `stddev()` or `structural` is specified.

`correlation(exchangeable)` provides for one correlation coefficient common to all utilities, except the utility associated with the `basealternative()` option.

`correlation(independent)` assumes that all correlations are zero.

`correlation(pattern` *matname*`)` and `correlation(fixed` *matname*`)` give you more flexibility in defining the correlation structure. See *Covariance structures* in [CM] **cmmprobit** for more information.

`stddev(`*stddev*`)` specifies the variance structure of the utility (latent-variable) errors.

`stddev(heteroskedastic)` is the most general and has $J-2$ estimable parameters. The standard deviations of the utility errors for the alternatives specified in `basealternative()` and `scalealternative()` are fixed to one.

`stddev(homoskedastic)` constrains all the standard deviations to equal one.

`stddev(pattern` *matname*`)` and `stddev(fixed` *matname*`)` give you added flexibility in defining the standard deviation parameters. See *Covariance structures* in [CM] **cmmprobit** for more information.

`factor(#)` requests that the factor covariance structure of dimension # be used. The `factor()` option can be used with the `structural` option but cannot be used with `stddev()` or `correlation()`. A $\# \times J$ (or $\# \times J-1$) matrix, $\mathbf{C}$, is used to factor the covariance matrix as $I+\mathbf{C}'\mathbf{C}$, where

I is the identity matrix of dimension J (or $J-1$). The column dimension of $\mathbf{C}$ depends on whether the covariance is structural or differenced. The row dimension of $\mathbf{C}$, #, must be less than or equal to floor$[\{J(J-1)/2-1\}/(J-2)]$, because there are only $J(J-1)/2-1$ identifiable covariance parameters. This covariance parameterization may be useful for reducing the number of covariance parameters that need to be estimated.

If the covariance is structural, the column of $\mathbf{C}$ corresponding to the base alternative contains zeros. The column corresponding to the scale alternative has a one in the first row and zeros elsewhere. If the covariance is differenced, the column corresponding to the scale alternative (differenced with the base) has a one in the first row and zeros elsewhere.

`structural` requests the $J \times J$ structural covariance parameterization instead of the default $J-1 \times J-1$ differenced covariance parameterization (the covariance of the utility errors differenced with that of the base alternative). The differenced covariance parameterization will achieve the same MSL regardless of the choice of `basealternative()` and `scalealternative()`. On the other hand, the structural covariance parameterization imposes more normalizations that may bound the model away from its maximum likelihood and thus prevent convergence with some datasets or choices of `basealternative()` and `scalealternative()`.

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***.

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `basealternative()` and `scalealternative()`.

Reporting

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

`notransform` prevents retransforming the Cholesky-factored covariance estimates to the correlation and standard deviation metric. `notransform` may not be specified on replay.

This option has no effect if `structural` is not specified because the default differenced covariance estimates have no interesting interpretation as correlations and standard deviations. `notransform` also has no effect if the `correlation()` and `stddev()` options are specified with anything other than their default values. Here it is generally not possible to factor the covariance matrix, so optimization is already performed using the standard deviation and correlation representations.

`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**.

Integration

`intmethod(hammersley | halton | random)` specifies the method of generating the point sets used in the quasi–Monte Carlo integration of the multivariate normal density. `intmethod(hammersley)`, the default, uses the Hammersley sequence; `intmethod(halton)` uses the Halton sequence; and `intmethod(random)` uses a sequence of uniform random numbers.

`intpoints(#)` specifies the number of points to use in the Monte Carlo integration. If option `intmethod(hammersley)` or `intmethod(halton)` is used, the default is 500 +

floor$[2.5\sqrt{N_c\{\ln(k+5)+v\}}]$, where N_c is the number of cases, k is the number of coefficients in the model, and v is the number of variance parameters. If `intmethod(random)` is used, the number of points is the above times 2. Larger values of `intpoints()` provide better approximations of the log likelihood at the cost of additional computation time.

`intburn(#)` specifies where in the Hammersley or Halton sequence to start, which helps reduce the correlation between the sequences of each dimension. The default is `intburn(0)`. This option may not be specified with `intmethod(random)`.

`intseed(`*code*`|#)` specifies the seed to use for generating the uniform pseudo–random sequence. This option may be specified only with `intmethod(random)`. *code* refers to a string that records the state of the random-number generator `runiform()`; see [R] **set seed**. An integer value # may be used also. The default is to use the current seed value from Stata's uniform random-number generator, which can be obtained from `c(rngstate)`.

`antithetics` specifies that antithetic draws be used. The antithetic draw for the $J-1$ vector uniform-random variables, $\mathbf{x}$, is $1-\mathbf{x}$.

`nopivot` turns off integration interval pivoting. By default, `cmroprobit` will pivot the wider intervals of integration to the interior of the multivariate integration. This improves the accuracy of the quadrature estimate. However, discontinuities may result in the computation of numerical second-order derivatives using finite differencing (for the Newton–Raphson optimize technique, `tech(nr)`) when few simulation points are used, resulting in a non–positive-definite Hessian. `cmroprobit` uses the Broyden–Fletcher–Goldfarb–Shanno optimization algorithm, by default, which does not require computing the Hessian numerically using finite differencing.

`initbhhh(#)` specifies that the Berndt–Hall–Hall–Hausman (BHHH) algorithm be used for the initial # optimization steps. This option is the only way to use the BHHH algorithm along with other optimization techniques. The algorithm switching feature of `ml`'s `technique()` option cannot include `bhhh`.

`favor(speed|space)` instructs `cmroprobit` to favor either speed or space when generating the integration points. `favor(speed)` is the default. When favoring speed, the integration points are generated once and stored in memory, thus increasing the speed of evaluating the likelihood. This speed increase can be seen when there are many cases or when the user specifies many integration points, `intpoints(#)`. When favoring space, the integration points are generated repeatedly with each likelihood evaluation.

For unbalanced data, where the number of alternatives varies with each case, the estimates computed using `intmethod(random)` will vary slightly between `favor(speed)` and `favor(space)`. This is because the uniform sequences will not be identical, even when initiating the sequences using the same uniform seed, `intseed(`*code*`|#)`. For `favor(speed)`, `ncase` blocks of `intpoints(#)` $\times$ $J-2$ uniform points are generated, where J is the maximum number of alternatives. For `favor(space)`, the column dimension of the matrices of points varies with the number of alternatives that each case has.

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**.

The following options may be particularly useful in obtaining convergence with `cmroprobit`: `difficult`, `technique(`*algorithm_spec*`)`, `nrtolerance(#)`, `nonrtolerance`, and `from(`*init_specs*`)`.

If `technique()` contains more than one algorithm specification, `bhhh` cannot be one of them. To use the BHHH algorithm with another algorithm, use the `initbhhh()` option, and specify the other algorithm in `technique()`.

Setting the optimization type to `technique(bhhh)` resets the default *vcetype* to `vce(opg)`.

The default optimization technique is `technique(bfgs)`.

When you specify `from(`*matname* [`, copy`]`)`, the values in *matname* associated with the utility error variances must be for the log-transformed standard deviations and inverse-hyperbolic tangent-transformed correlations. This option makes using the coefficient vector from a previously fitted `cmroprobit` model convenient as a starting point.

The following options are available with `cmroprobit` but are not shown in the dialog box:

`collinear`, `coeflegend`; see [R] **Estimation options**.

Remarks and examples

The mathematical description and numerical computations of the rank-ordered probit model are similar to that of the multinomial probit model. The only difference is that the dependent variable of the rank-ordered probit model is ordinal, showing preferences among alternatives, as opposed to the binary dependent variable of the multinomial probit model, indicating a chosen alternative.

Only the order in the ranks, not the magnitude of their differences, is assumed to be relevant. By default, the largest rank indicates the more desirable alternative. Use the `reverse` option if the lowest rank should be interpreted as the more desirable alternative.

Tied ranks are allowed, but they increase the computation time because all permutations of the tied ranks are used in computing the likelihood for each case.

We will describe how the likelihood of a ranking is computed using the utility (latent-variable) framework here, but for details of the utility parameterization of these models and the method of maximum simulated likelihood, see [CM] **cmmprobit**.

Consider the utility of a J alternative rank-ordered probit model. Using the notation from `cmmprobit`, we have utilities (latent variables) η_{ij}, $j = 1, \ldots, J$, such that

$$\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j + \xi_{ij}$$

Here the $\mathbf{x}_{ij}$ are the alternative-specific independent variables, the $\mathbf{z}_i$ are the case-specific variables, and the ξ_{ij} are multivariate normal with mean zero and covariance $\boldsymbol{\Omega}$. Without loss of generality, assume that individual i ranks the alternatives in order of the alternative indices $j = 1, 2, \ldots, J$, so the alternative J is the preferred alternative, and alternative 1 is the least preferred alternative. The probability of this ranking given $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}_j$ is the probability that $\eta_{i,J-1} - \eta_{i,J} \leq 0$ and $\eta_{i,J-2} - \eta_{i,J-1} \leq 0$, …, and $\eta_{i,1} - \eta_{i,2} \leq 0$.

▷ Example 1

Long and Freese (2014, 477) provide an example of a rank-ordered logit model with alternative-specific variables. We use this dataset to demonstrate `cmroprobit`. The data come from the Wisconsin Longitudinal Study. This is a study of 1957 Wisconsin high school graduates who were asked to

report their relative preferences of four job characteristics: esteem, a job other people regard highly; variety, a job that is not repetitive and allows you to do a variety of things; autonomy, a job where your supervisor does not check on you frequently; and security, a job with a low risk of being laid off.

The case-specific covariates are gender, `female`, an indicator variable for females, and `score`, a score on a general mental ability test measured in standard deviations. We also have alternative-specific variables `high` and `low`, which indicate whether the respondent's current job is high or low in esteem, variety, autonomy, or security. When taken together, `high` and `low` provide three states for a respondent's current job status for each alternative—high, low, or neither. From these variables, we create a new variable, `currentjob`, that we include as an alternative-specific variable in our model.

The alternatives are ranked such that 1 is the preferred alternative and 4 is the least preferred.

```
. use https://www.stata-press.com/data/r18/wlsrank
(1992 Wisconsin Longitudinal Study data on job values)
. generate currentjob = 1 if low==1
(10,264 missing values generated)
. replace currentjob = 2 if low==0 & high==0
(6,319 real changes made)
. replace currentjob = 3 if high==1
(3,945 real changes made)
. label define current 1 "Low" 2 "Neither" 3 "High"
. label values currentjob current
. list id jobchar rank female score currentjob in 1/12, sepby(id)
```

	id	jobchar	rank	female	score	curren~b
1.	1	Esteem	3	Female	.0492111	Neither
2.	1	Variety	1	Female	.0492111	Neither
3.	1	Autonomy	4	Female	.0492111	Neither
4.	1	Security	1	Female	.0492111	Neither
5.	5	Esteem	2	Female	2.115012	High
6.	5	Variety	2	Female	2.115012	High
7.	5	Autonomy	1	Female	2.115012	Neither
8.	5	Security	2	Female	2.115012	High
9.	7	Esteem	4	Male	1.701852	Neither
10.	7	Variety	1	Male	1.701852	Low
11.	7	Autonomy	1	Male	1.701852	High
12.	7	Security	1	Male	1.701852	Neither

The three cases listed have tied ranks. `cmroprobit` will allow ties but at the cost of increased computation time. To evaluate the likelihood of the first observation, `cmroprobit` must compute

$$\Pr(\texttt{Esteem}=3,\ \texttt{Variety}=1,\ \texttt{Autonomy}=4,\ \texttt{Security}=2)+$$
$$\Pr(\texttt{Esteem}=3,\ \texttt{Variety}=2,\ \texttt{Autonomy}=4,\ \texttt{Security}=1)$$

and both of these probabilities are estimated using simulation. In fact, the full dataset contains 7,237 tied ranks, and `cmroprobit` takes a long time to estimate the parameters. For exposition, we fit the rank-ordered probit model by using the cases without ties. These cases are marked in the variable `noties`.

The model of job preference is

$$\eta_{ij} = \beta_1 \texttt{Neither}_{ij} + \beta_2 \texttt{High}_{ij} \; + \; \alpha_{1j}\texttt{female}_i + \alpha_{2j}\texttt{score}_i + \alpha_{0j} + \xi_{ij}$$

for $j = 1, 2, 3, 4$. The base alternative will be `Esteem`, so $\alpha_{01} = \alpha_{11} = \alpha_{21} = 0$.

Before we can fit our model, we must `cmset` our data. The first argument to `cmset` is the case ID variable. For these data, it is the variable `id`, which identifies respondents. The second argument is the alternatives variable, which identifies the alternatives that could have been chosen. In this instance, it is the variable `jobchar`, which gives the four job characteristics: esteem, variety, autonomy, and security.

```
. cmset id jobchar
     Case ID variable: id
Alternatives variable: jobchar
```

We fit our model:

```
. cmroprobit rank i.currentjob if noties, casevars(i.female score) reverse
note: variable 2.currentjob has 69 cases that are not alternative-specific;
      there is no within-case variability.
note: variable 3.currentjob has 107 cases that are not alternative-specific;
      there is no within-case variability.
Iteration 0:  Log simulated-likelihood = -1102.9667
Iteration 1:  Log simulated-likelihood = -1089.1146  (backed up)
Iteration 2:  Log simulated-likelihood = -1085.7877  (backed up)
Iteration 3:  Log simulated-likelihood = -1083.0085  (backed up)
Iteration 4:  Log simulated-likelihood = -1082.5081  (backed up)
Iteration 5:  Log simulated-likelihood = -1082.1977  (backed up)
Iteration 6:  Log simulated-likelihood = -1082.1208  (backed up)
Iteration 7:  Log simulated-likelihood = -1082.0995
Iteration 8:  Log simulated-likelihood = -1082.0442
Iteration 9:  Log simulated-likelihood = -1081.8316  (backed up)
Iteration 10: Log simulated-likelihood = -1081.6816
Iteration 11: Log simulated-likelihood = -1081.5777
Iteration 12: Log simulated-likelihood = -1081.5137
Iteration 13: Log simulated-likelihood = -1081.1495
Iteration 14: Log simulated-likelihood = -1081.0503
Iteration 15: Log simulated-likelihood = -1080.7247
Iteration 16: Log simulated-likelihood = -1080.1485
Iteration 17: Log simulated-likelihood = -1080.0215
Iteration 18: Log simulated-likelihood = -1080.0179
Iteration 19: Log simulated-likelihood = -1079.9916
Iteration 20: Log simulated-likelihood = -1079.9733
Iteration 21: Log simulated-likelihood = -1079.9733
```

```
Rank-ordered probit choice model              Number of obs      =      1,660
Case ID variable: id                          Number of cases    =        415
Alternatives variable: jobchar                Alts per case: min =          4
                                                             avg =        4.0
                                                             max =          4
Integration sequence:      Hammersley
Integration points:               642         Wald chi2(8)       =      33.92
Log simulated-likelihood = -1079.9733         Prob > chi2        =     0.0000
```

rank	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
jobchar						
currentjob						
Neither	.0694818	.1092531	0.64	0.525	-.1446503	.2836138
High	.4435911	.121678	3.65	0.000	.2051066	.6820757
Esteem	(base alternative)					
Variety						
female						
Female	.1354259	.1843808	0.73	0.463	-.2259537	.4968055
score	.14071	.0977659	1.44	0.150	-.0509077	.3323278
_cons	1.734496	.1449852	11.96	0.000	1.45033	2.018661
Autonomy						
female						
Female	.2562822	.1645938	1.56	0.119	-.0663158	.5788801
score	.189963	.0873586	2.17	0.030	.0187433	.3611827
_cons	.7007559	.1203087	5.82	0.000	.4649553	.9365566
Security						
female						
Female	.2326342	.2055864	1.13	0.258	-.1703079	.6355762
score	-.1779831	.1101985	-1.62	0.106	-.3939682	.0380021
_cons	1.34348	.1598787	8.40	0.000	1.030123	1.656836
/lnl2_2	.1813086	.0756934	2.40	0.017	.0329522	.329665
/lnl3_3	.4843953	.0793885	6.10	0.000	.3287967	.639994
/l2_1	.6060303	.1158474	5.23	0.000	.3789735	.8330871
/l3_1	.4491585	.1435157	3.13	0.002	.167873	.7304441
/l3_2	.2305503	.121713	1.89	0.058	-.0080029	.4691034

```
(jobchar=Esteem is the alternative normalizing location)
(jobchar=Variety is the alternative normalizing scale)
```

We specified the `reverse` option because a rank of 1 is the highest preference. The variance–covariance estimates are for the Cholesky-factored variance–covariance for the utility errors differenced with that of alternative `Esteem`. We can view the estimated correlations by typing

```
. estat correlation
```

	Variety	Autonomy	Security
Variety	1.0000		
Autonomy	0.4512	1.0000	
Security	0.2642	0.2402	1.0000

```
Note: Correlations are for alternatives differenced with Esteem.
```

and typing

```
. estat covariance
```

	Variety	Autonomy	Security
Variety	2		
Autonomy	.8570563	1.804358	
Security	.6352061	.5485839	2.889653

```
Note: Covariances are for alternatives differenced with Esteem.
```

gives the variances and covariances. In [R] **mprobit**, we explain that if the utility errors are independent, then the correlations in the differenced parameterization should be ∼0.5, and the variances should be ∼2.0, which seems to be the case here.

The coefficient estimates for the probit models can be difficult to interpret because of the normalization for location and scale. The regression estimates for the case-specific variables will be relative to the base alternative, and the regression estimates for both the case-specific and alternative-specific variables are affected by the scale normalization. The more pronounced the heteroskedasticity and correlations, the more pronounced the resulting estimate differences when choosing alternatives to normalize for location and scale. However, when using the differenced covariance structure, you will obtain the same model likelihood regardless of which alternatives you choose as the base and scale alternatives.

For model interpretation, you can obtain predicted probabilities using `predict`; see [CM] **cmroprobit postestimation**. You can also use `margins` to estimate expected probabilities, marginal effects, and more. See [CM] **Intro 1** and [CM] **margins** for more information and examples using `margins` for interpretation of the results of `cm` commands. Also, see [CM] **Intro 6** for an example after `cmroprobit`.

After you have fit what you consider your final model, you should run the same model again, but this time setting `intpoints(#)`, the number of integration points in the simulated likelihood to a larger number. In this example, we see from the header that the default number of points was 642. We would run our model again using, say, 2000 points and see by how much the coefficient or covariance parameter estimates change. If changes are small compared with standard errors, we can have confidence in the numerical soundness of the simulation used to compute the likelihood. See *Setting the number of integration points* in [CM] **Intro 5** for more information.

◁

Stored results

`cmroprobit` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(N_case)`	number of cases
`e(N_ties)`	number of ties
`e(N_ic)`	N for Bayesian information criterion (BIC)
`e(N_clust)`	number of clusters
`e(k)`	number of parameters
`e(k_alt)`	number of alternatives
`e(k_indvars)`	number of alternative-specific variables
`e(k_casevars)`	number of case-specific variables
`e(k_sigma)`	number of variance estimates
`e(k_rho)`	number of correlation estimates
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test

e(df_m)	model degrees of freedom
e(ll)	log simulated-likelihood
e(const)	constant indicator
e(i_base)	base alternative index
e(i_scale)	scale alternative index
e(mc_points)	number of Monte Carlo replications
e(mc_burn)	starting sequence index
e(mc_antithetics)	antithetics indicator
e(reverse)	1 if minimum rank is best, 0 if maximum rank is best
e(chi2)	χ^2
e(p)	p-value for model test
e(fullcov)	unstructured covariance indicator
e(structcov)	1 if structured covariance, 0 otherwise
e(cholesky)	Cholesky-factored covariance indicator
e(alt_min)	minimum number of alternatives
e(alt_avg)	average number of alternatives
e(alt_max)	maximum number of alternatives
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	cmroprobit
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(caseid)	name of case ID variable
e(altvar)	name of alternatives variable
e(alteqs)	alternative equation names
e(alt#)	alternative labels
e(wtype)	weight type
e(wexp)	weight expression
e(marktype)	casewise or altwise, type of markout
e(key_N_ic)	cases, key for N for Bayesian information criterion (BIC)
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(correlation)	correlation structure
e(stddev)	variance structure
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(mc_method)	technique used to generate sequences
e(mc_rngstate)	random-number state used
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
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(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(marginsdefault)	default predict() specification for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(stats)	alternative statistics
e(stdpattern)	variance pattern
e(stdfixed)	fixed and free standard deviations

e(altvals)	alternative values
e(altfreq)	alternative frequencies
e(alt_casevars)	indicators for estimated case-specific coefficients—e(k_alt)×e(k_casevars)
e(corpattern)	correlation structure
e(corfixed)	fixed and free correlations
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

From a computational perspective, cmroprobit is similar to cmmprobit, and the two programs share many numerical tools. Therefore, we will use the notation from *Methods and formulas* in [CM] **cmmprobit** to discuss the rank-ordered probit probability model.

The utilities (latent variables) for a J-alternative model are $\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j + \xi_{ij}$, for $j = 1, \ldots, J$, $i = 1, \ldots, n$, and $\boldsymbol{\xi}_i' = (\xi_{i,1}, \ldots, \xi_{i,\mathrm{J}}) \sim \mathrm{MVN}(\mathbf{0}, \boldsymbol{\Omega})$. Without loss of generality, assume for the ith observation that an individual ranks the alternatives in the order of their numeric indices, $\mathbf{y}_i = (J, J-1, \ldots, 1)$, so the first alternative is the most preferred, and the last alternative is the least preferred. We can then difference the utilities such that

$$\begin{aligned} v_{ik} &= \eta_{i,k+1} - \eta_{i,k} \\ &= (\mathbf{x}_{i,k+1} - \mathbf{x}_{i,k})\boldsymbol{\beta} + \mathbf{z}_i(\boldsymbol{\alpha}_{k+1} - \boldsymbol{\alpha}_k) + \xi_{i,k+1} - \xi_{ik} \\ &= \boldsymbol{\delta}_{ik}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\gamma}_k + \epsilon_{ik} \end{aligned}$$

for $k = 1, \ldots, J-1$ and where $\boldsymbol{\epsilon}_i = (\epsilon_{i1}, \ldots, \epsilon_{i,\mathrm{J}-1}) \sim \mathrm{MVN}(\mathbf{0}, \boldsymbol{\Sigma}_{(i)})$. $\boldsymbol{\Sigma}$ is indexed by i because it is specific to the ranking of individual i. We denote the deterministic part of the model as $\lambda_{ik} = \boldsymbol{\delta}_{ik}\boldsymbol{\beta} + \mathbf{z}_j\boldsymbol{\gamma}_k$, and the probability of this event is

$$\begin{aligned} \Pr(\mathbf{y}_i) &= \Pr(v_{i1} \le 0, \ldots, v_{i,J-1} \le 0) \\ &= \Pr(\epsilon_{i1} \le -\lambda_{i1}, \ldots, \epsilon_{i,J-1} \le -\lambda_{i,J-1}) \\ &= (2\pi)^{-(J-1)/2} \, |\boldsymbol{\Sigma}_{(i)}|^{-1/2} \int_{-\infty}^{-\lambda_{i1}} \cdots \int_{-\infty}^{-\lambda_{i,J-1}} \exp\left(-\tfrac{1}{2}\mathbf{z}'\boldsymbol{\Sigma}_{(i)}^{-1}\mathbf{z}\right) d\mathbf{z} \end{aligned}$$

The integral has the same form as (3) in [CM] **cmmprobit**. See [CM] **cmmprobit** for details on evaluating this integral numerically by using simulation.

cmroprobit handles tied ranks by enumeration. For k tied ranks, it will generate $k!$ rankings, where ! is the factorial operator $k! = k(k-1)(k-2)\cdots(2)(1)$. For two sets of tied ranks of size k_1 and k_2, cmroprobit will generate $k_1!k_2!$ rankings. The total probability is the sum of the probability of each ranking. For example, if there are two tied ranks such that $\mathbf{y}_i = (J, J, J-2, \ldots, 1)$, then cmroprobit will evaluate $\Pr(\mathbf{y}_i) = \Pr(\mathbf{y}_i^{(1)}) + \Pr(\mathbf{y}_i^{(2)})$, where $\mathbf{y}_i^{(1)} = (J, J-1, J-2, \ldots, 1)$ and $\mathbf{y}_i^{(2)} = (J-1, J, J-2, \ldots, 1)$.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using vce(robust) and vce(cluster *clustvar*). See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*. Specifying vce(robust) is equivalent to specifying vce(cluster *caseid*), where *caseid* is the variable that identifies the cases.

Reference

Long, J. S., and J. Freese. 2014. *Regression Models for Categorical Dependent Variables Using Stata*. 3rd ed. College Station, TX: Stata Press.

Also see

[CM] **cmroprobit postestimation** — Postestimation tools for cmroprobit

[CM] **cmmprobit** — Multinomial probit choice model

[CM] **cmrologit** — Rank-ordered logit choice model

[CM] **cmset** — Declare data to be choice model data

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

[R] **ologit** — Ordered logistic regression

[R] **oprobit** — Ordered probit regression

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

Title

cmroprobit postestimation — Postestimation tools for cmroprobit

Postestimation commands predict margins estat
Remarks and examples Also see

Postestimation commands

The following postestimation commands are of special interest after `cmroprobit`:

Command	Description
`estat covariance`	covariance matrix of the utility errors for the alternatives
`estat correlation`	correlation matrix of the utility errors for the alternatives
`estat facweights`	covariance factor weights matrix

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`	adjusted predictions, predictive margins, and 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`	probabilities, 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 probabilities, linear predictions, and standard errors.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [`,` *statistic*]

`predict` [*type*] *stub*`*` [*if*] [*in*]`,` `scores`

statistic	Description
Main	
`pr`	probability of each ranking, by case; the default
`pr1`	probability alternative is preferred
`xb`	linear prediction
`stdp`	standard error of the linear prediction

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`predict` omits missing values casewise if `cmroprobit` used casewise deletion (the default); if `cmroprobit` used alternativewise deletion (option `altwise`), `predict` uses alternativewise deletion.

Options for predict

Main

`pr`, the default, calculates the probability of each ranking. For each case, one probability is computed for the ranks in `e(depvar)`.

`pr1` calculates the probability that each alternative is preferred.

`xb` calculates the linear prediction $\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j$ for alternative j and case i.

`stdp` calculates the standard error of the linear prediction.

`scores` calculates the scores for each coefficient in `e(b)`. This option requires a new variable list of length equal to the number of columns in `e(b)`. Otherwise, use the *stub*`*` syntax to have `predict` generate enumerated variables with prefix *stub*.

margins

Description for margins

`margins` estimates margins of response for probabilities and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`pr`	not allowed with `margins`
`pr1`	probability alternative is preferred; the default
`xb`	linear prediction
`stdp`	not allowed with `margins`
`scores`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For more details, see [CM] **margins**.

estat

Description for estat

`estat covariance` computes the estimated variance–covariance matrix of the utility (latent-variable) errors for the alternatives. The estimates are displayed, and the variance–covariance matrix is stored in `r(cov)`.

`estat correlation` computes the estimated correlation matrix of the utility (latent-variable) errors for the alternatives. The estimates are displayed, and the correlation matrix is stored in `r(cor)`.

`estat facweights` displays the covariance factor weights matrix and stores it in `r(C)`.

Menu for estat

Statistics > Postestimation

Syntax for estat

Covariance matrix of the utility errors for the alternatives

```
estat covariance [ , format(%fmt) border(bspec) left(#) ]
```

Correlation matrix of the utility errors for the alternatives

```
estat correlation [ , format(%fmt) border(bspec) left(#) ]
```

Covariance factor weights matrix

```
estat facweights [ , format(%fmt) border(bspec) left(#) ]
```

`collect` is allowed with all `estat` commands; see [U] **11.1.10 Prefix commands**.

Options for estat covariance, estat correlation, and estat facweights

`format(%fmt)` sets the matrix display format. The default for `estat covariance` and `estat facweights` is `format(%9.0g)`; the default for `estat correlation` is `format(%9.4f)`.

`border(bspec)` sets the matrix display border style. The default is `border(all)`. See [P] **matlist**.

`left(#)` sets the matrix display left indent. The default is `left(2)`. See [P] **matlist**.

Remarks and examples

After fitting a rank-ordered probit choice model, you can use `predict` to obtain the probabilities of the observed rankings of the alternatives or the probabilities of each alternative being preferred.

When evaluating the multivariate normal probabilities via Monte Carlo, `predict` uses the same method to generate the random sequence of numbers as the previous call to `cmroprobit`. For example, if you specified `intmethod(halton)` when fitting the model, `predict` also uses Halton sequences.

In example 1 of [CM] **cmroprobit**, we fit a model of job characteristic preferences. This is a study of Wisconsin high school graduates who were asked to rate their relative preference of four job characteristics: esteem, variety, autonomy, and security. The alternatives are ranked such that 1 is the preferred alternative and 4 is the least preferred.

The case-specific covariates are gender, `female`, an indicator variable for females, and `score`, a score on a general mental ability test measured in standard deviations. From the variables `high` and `low`, we create an alternative-specific variable, `currentjob`, that indicates whether the respondent's current job is high, low, or neither in esteem, variety, autonomy, or security.

We load the data and `cmset` them. For speed of running this example, we keep only untied rankings. Then, we fit our `cmroprobit` model.

```
. use https://www.stata-press.com/data/r18/wlsrank
(1992 Wisconsin Longitudinal Study data on job values)

. cmset id jobchar

     Case ID variable: id
Alternatives variable: jobchar

. keep if noties
(11,244 observations deleted)

. generate currentjob = 1 if low==1
(1,304 missing values generated)

. replace currentjob = 2 if low==0 & high==0
(805 real changes made)

. replace currentjob = 3 if high==1
(499 real changes made)

. label define current 1 "Low" 2 "Neither" 3 "High"

. label values currentjob current

. cmroprobit rank i.currentjob, casevars(i.female score) reverse
note: variable 2.currentjob has 69 cases that are not alternative-specific;
      there is no within-case variability.
note: variable 3.currentjob has 107 cases that are not alternative-specific;
      there is no within-case variability.

Iteration 0:  Log simulated-likelihood = -1102.9667
Iteration 1:  Log simulated-likelihood = -1089.1146  (backed up)
Iteration 2:  Log simulated-likelihood = -1085.7877  (backed up)
Iteration 3:  Log simulated-likelihood = -1083.0085  (backed up)
Iteration 4:  Log simulated-likelihood = -1082.5081  (backed up)
Iteration 5:  Log simulated-likelihood = -1082.1977  (backed up)
Iteration 6:  Log simulated-likelihood = -1082.1208  (backed up)
Iteration 7:  Log simulated-likelihood = -1082.0995
Iteration 8:  Log simulated-likelihood = -1082.0442
Iteration 9:  Log simulated-likelihood = -1081.8316  (backed up)
Iteration 10: Log simulated-likelihood = -1081.6816
Iteration 11: Log simulated-likelihood = -1081.5777
Iteration 12: Log simulated-likelihood = -1081.5137
Iteration 13: Log simulated-likelihood = -1081.1495
Iteration 14: Log simulated-likelihood = -1081.0503
Iteration 15: Log simulated-likelihood = -1080.7247
```

```
Iteration 16: Log simulated-likelihood = -1080.1485
Iteration 17: Log simulated-likelihood = -1080.0215
Iteration 18: Log simulated-likelihood = -1080.0179
Iteration 19: Log simulated-likelihood = -1079.9916
Iteration 20: Log simulated-likelihood = -1079.9733
Iteration 21: Log simulated-likelihood = -1079.9733
```

Rank-ordered probit choice model — Number of obs = 1,660
Case ID variable: id — Number of cases = 415
Alternatives variable: jobchar — Alts per case: min = 4
avg = 4.0
max = 4
Integration sequence: Hammersley
Integration points: 642 — Wald chi2(8) = 33.92
Log simulated-likelihood = -1079.9733 — Prob > chi2 = 0.0000

rank	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
jobchar						
currentjob						
Neither	.0694818	.1092531	0.64	0.525	-.1446503	.2836138
High	.4435911	.121678	3.65	0.000	.2051066	.6820757
Esteem	(base alternative)					
Variety						
female						
Female	.1354259	.1843808	0.73	0.463	-.2259537	.4968055
score	.14071	.0977659	1.44	0.150	-.0509077	.3323278
_cons	1.734496	.1449852	11.96	0.000	1.45033	2.018661
Autonomy						
female						
Female	.2562822	.1645938	1.56	0.119	-.0663158	.5788801
score	.189963	.0873586	2.17	0.030	.0187433	.3611827
_cons	.7007559	.1203087	5.82	0.000	.4649553	.9365566
Security						
female						
Female	.2326342	.2055864	1.13	0.258	-.1703079	.6355762
score	-.1779831	.1101985	-1.62	0.106	-.3939682	.0380021
_cons	1.34348	.1598787	8.40	0.000	1.030123	1.656836
/lnl2_2	.1813086	.0756934	2.40	0.017	.0329522	.329665
/lnl3_3	.4843953	.0793885	6.10	0.000	.3287967	.639994
/l2_1	.6060303	.1158474	5.23	0.000	.3789735	.8330871
/l3_1	.4491585	.1435157	3.13	0.002	.167873	.7304441
/l3_2	.2305503	.121713	1.89	0.058	-.0080029	.4691034

(jobchar=Esteem is the alternative normalizing location)
(jobchar=Variety is the alternative normalizing scale)

We obtain the probabilities of the observed alternative rankings using `predict` with the `pr` option. The probabilities of each alternative being preferred is given by the `pr1` option.

```
. predict prob, pr
. predict prob1, pr1
. list id jobchar rank prob prob1 in 1/12, sepby(id)
```

	id	jobchar	rank	prob	prob1
1.	13	Esteem	4	.0424396	.0159974
2.	13	Variety	2	.0424396	.6024934
3.	13	Autonomy	1	.0424396	.1029332
4.	13	Security	3	.0424396	.278576
5.	19	Esteem	3	.0942127	.0140184
6.	19	Variety	2	.0942127	.4026075
7.	19	Autonomy	4	.0942127	.1232482
8.	19	Security	1	.0942127	.4601093
9.	22	Esteem	4	.1416861	.0255156
10.	22	Variety	1	.1416861	.455048
11.	22	Autonomy	2	.1416861	.2565435
12.	22	Security	3	.1416861	.2629159

The `prob` variable is constant for each case because it contains the probability of the observed ranking in the `rank` variable. The `prob1` variable contains the estimated probability of each alternative being preferred. The sum of the values in `prob1` will be approximately 1 for each case. They do not add up to exactly 1 because of approximations due to the GHK algorithm.

For examples of the specialized `estat` subcommands `covariance` and `correlation`, see [CM] **Intro 6** and [CM] **cmroprobit**.

Also see

[CM] **cmroprobit** — Rank-ordered probit choice model

[CM] **cmmprobit** — Multinomial probit choice model

[CM] **cmmprobit postestimation** — Postestimation tools for cmmprobit

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

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

Title

cmsample — Display reasons for sample exclusion

Description Quick start Menu Syntax
Options Remarks and examples Stored results Also see

Description

`cmsample` displays a table with the reasons why observations in a choice model were dropped from the estimation sample. It also flags choice-model data errors, such as errors in the alternatives variable, dependent variable, or case-specific variables. With the use of its `generate()` option, observations that were dropped or led to an error message can be identified.

Quick start

Tabulate reasons `cmclogit` excluded observations from the estimation sample when fitting a model of `y` on alternative-specific variables `x1`, `x2`, and `x3` and case-specific variables `cv1` and `cv2`

```
cmclogit y x1 x2 x3, casevars(cv1 cv2)
cmsample x1 x2 x3, choice(y) casevars(cv1 cv2)
```

Create a variable named `problem` identifying categories of problems in the choice variable `y`, and list the problem observations

```
cmsample, choice(y) generate(problem)
list y if problem != 0
```

Replace `problem` with categories of problems in case-specific variables `cv1` and `cv2`, and list the problem observations

```
cmsample, casevars(cv1 cv2) generate(problem, replace)
list cv1 cv2 if problem != 0
```

Use alternativewise rather than casewise deletion of observations with missing values

```
cmsample x1 x2 x3, choice(y) casevars(cv1 cv2) altwise
```

Allow `y` to be ranks instead of a 0/1 variable (the default)

```
cmsample x1 x2 x3, choice(y) casevars(cv1 cv2) ranks
```

Menu

Statistics > Choice models > Setup and utilities > Display reasons for sample exclusion

Syntax

```
cmsample [varlist] [if] [in] [weight] [, options]
```

varlist is a list of alternative-specific numeric variables.

options	Description
Main	
`choice(`*choicevar*`)`	0/1 variable indicating the chosen alternatives
`casevars(`*varlist*$_c$`)`	case-specific variables
`altwise`	use alternativewise deletion instead of casewise deletion
`ranks`	allow *choicevar* to be ranks
`generate(`*newvar*`[ , replace])`	create new variable containing reasons for omitting observations and for error messages; optionally replace existing variable

You must `cmset` your data before using `cmsample`; see [CM] **cmset**.

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

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

`fweights`, `iweights`, and `pweights` are allowed; see [U] **11.1.6 weight**. Weights are checked for missing values and other errors but are not used for the tabulation.

Options

Main

`choice(`*choicevar*`)` specifies a 0/1 variable indicating the chosen alternatives. Typically, it is a dependent variable in the choice model.

`casevars(`*varlist*$_c$`)` specifies case-specific numeric variables. These are variables that are constant for each case.

`altwise` specifies that alternativewise deletion be used when omitting observations due to missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier; these observations are always handled alternativewise regardless of whether `altwise` is specified.

`ranks` allows *choicevar* to be ranks. Any numeric value in *choicevar* is allowed.

`generate(`*newvar*`[ , replace])` creates a new variable containing categories of the reasons for omitting observations and for error messages. The variable *newvar* is numeric and potentially valued 0, 1, 2, . . . , 16. The value 0 indicates the observation is to be included in the estimation sample. The values 1–16 indicate cases and observations that were either marked out or would generate error messages. See the table of reasons for omitting observations and error messages for the list of values and their meaning. Specifying `replace` allows any existing variable named *newvar* to be replaced.

Remarks and examples

`cm` commands have special data requirements. `cmsample` lets you see whether the requirements are met and, if not, to identify the problems. `cmsample` can also flag cases with missing values. It can be used to check your data before running `cm` estimation commands, or it can be used after estimation commands to diagnose why cases and observations were excluded from the estimation sample.

Data for all cm commands must be arranged in long form with each case consisting of multiple Stata observations, one Stata observation for each alternative. There must be a case ID variable identifying cases for cross-sectional data. For panel data, there must be a panel ID variable and a time variable that together uniquely identify cases.

Most choice models are formulated with specified alternatives, and for these models, there must be a variable identifying the alternatives available to each case. For discrete choice models, the dependent variable must be a 0/1 variable representing a single chosen alternative within a case.

cm commands check whether the specified variables meet their data requirements and, when a check fails, respond in different ways depending on the problem. Sometimes an error message is issued. For example,

```
at least one choice set has more than one instance of the same alternative
r(459);
```

Sometimes cases with problems are simply dropped from the estimation sample, and a message is displayed. For example,

```
note: 2 cases (6 obs) dropped due to no positive outcome per case.
```

A researcher may want to look at the problematic observations to see whether the problem is caused by bad values in the data that can be fixed or simply to see and understand the problem. cmsample is a tool for doing this.

All cm commands, including cmsample, require your data to be cmset, and you may get an error message when cmsetting your data. You can identify the problematic observations by using the force option with cmset and then typing cmsample. See example 1.

If your data have missing values and you run a cm estimator, you might be surprised by how many cases were omitted from the estimation sample. By default, a missing value in any observation of a case will cause the entire case to be "marked out"—Stata talk for dropping observations from the estimation sample. This is called casewise deletion of observations. Dropping the whole case makes sense because a case is a single statistical observation. Stata found a problem with the data in this statistical observation, and Stata dropped it. The statistical observation consisted of multiple Stata observations, so multiple Stata observations were dropped, even though there may have been only one missing value in one Stata observation.

cmsample and all the other cm commands have the option altwise, which deletes only missing Stata observations from the sample, rather than deleting entire cases. It is called altwise, short for alternativewise. altwise deletion causes only the alternative containing the missing value to be dropped. The case remains in the sample; only it now lacks one or more of its alternatives and any data associated with that alternative. There may be consequences from this. For example, data with balanced choice sets may become unbalanced. With the default casewise deletion, balanced data always stays balanced. For an example of altwise deletion, see example 2.

By default, cmsample displays a table, but its real utility is its generate() option, which creates a variable with categories of the reasons observations were marked out or gave an error message. This variable can then be used to produce a listing of the problematic observations.

The table below gives cmsample's categories of reasons for deleting observations and error message possibilities.

Value	Description	Error
0	observations included	
1	if or in exclusion	
2	case ID variable missing	
3	time variable missing	
4	alternatives variable missing	
5	*varlist* missing	
6	weight missing	
7	casevars(*varlist*$_c$) missing	
8	choice variable missing	
9	choice sets of size one	
10	choice variable not 0/1	error (unless ranks specified)
11	choice variable all 0	
12	choice variable all 1	
13	choice variable multiple 1s	
14	weight not constant within case	error
15	repeated alternatives within case	error
16	casevars(*varlist*$_c$) not constant within case	error

Possibilities labeled as "error" in the table will not cause cmsample to end with an error message, but any other cm command will.

The possibilities are ordered in roughly the order that the cm estimators use when omitting observations. For example, if the choice variable is not 0/1 for the same case in which the alternatives variable has repeated values, it will be categorized as the former, not the latter—and other cm commands will complain about the former, not the latter, as well. Possibilities can be isolated simply by specifying fewer variables to cmsample.

▷ Example 1: Casewise (default) deletion of missing values

We have created a dataset with lots of missing values and several errors. Let's load it into memory and try to cmset it.

```
. use https://www.stata-press.com/data/r18/carchoice_missing
(Car choice data with missing values)

. cmset consumerid car
note: 1 case ignored because it has only one alternative.
note: 11 cases have missing values of car.
at least one choice set has more than one instance of the same alternative
r(459);
```

cmset gives an error message. But as we mentioned earlier, the thing to do now is to use the force option with cmset.

```
. cmset consumerid car, force
note: 1 case ignored because it has only one alternative.
note: 11 cases have missing values of car.
note: at least one choice set has more than one instance of the same
      alternative.

     Case ID variable: consumerid
Alternatives variable: car
```

The `force` option lets you `cmset` your data, but it only puts off the point at which you get an error message. If you try running a `cm` estimator after using `force` with `cmset`, the `cm` estimator will issue an error message (unless you use an `if` restriction to avoid the problematic observations).

`cmsample` will show you reasons for any error messages plus reasons why cases were dropped. Note that the syntax of `cmsample` is almost the same as that of a `cm` estimator. You put the case-specific variables in `casevars()`. Weights and `if` or `in` restrictions follow standard Stata syntax. The only difference is that you put the dependent variable in the `choice()` option.

```
. cmsample dealers [fw=weight] if income > 20.5,
> choice(purchase) casevars(i.gender income)
```

Reason for exclusion	Freq.	Percent	Cum.
observations included	2,812	89.04	89.04
if or in exclusion	79	2.50	91.55
caseid variable missing	7	0.22	91.77
alternatives variable missing	39	1.23	93.00
varlist missing	12	0.38	93.38
weight missing	25	0.79	94.17
casevars missing	112	3.55	97.72
choice variable missing	38	1.20	98.92
choice sets of size one	1	0.03	98.96
choice variable not 0/1*	4	0.13	99.08
choice variable all 0	6	0.19	99.27
choice variable all 1	4	0.13	99.40
choice variable multiple 1s	4	0.13	99.53
weight not constant within case*	4	0.13	99.65
repeated alternatives within case*	4	0.13	99.78
casevars not constant within case*	7	0.22	100.00
Total	3,158	100.00	

* indicates an error

As we said, we created a dataset with lots of problems! However, it is easy to track them down. The table indicates that there are several problems with the choice variable, the variable `purchase` in the `choice()` option. The variable `purchase` is supposed to be a 0/1 variable, intended to be our dependent variable in our choice model. To focus on the problems with this variable, we run `cmsample` again with just `choice(purchase)` and add the option `generate(flag)` to create a variable `flag` that identifies the errors.

```
. cmsample, choice(purchase) gen(flag)
```

Reason for exclusion	Freq.	Percent	Cum.
observations included	3,045	96.42	96.42
caseid variable missing	7	0.22	96.64
alternatives variable missing	39	1.23	97.88
choice variable missing	42	1.33	99.21
choice sets of size one	1	0.03	99.24
choice variable not 0/1*	4	0.13	99.37
choice variable all 0	8	0.25	99.62
choice variable all 1	4	0.13	99.75
choice variable multiple 1s	4	0.13	99.87
repeated alternatives within case*	4	0.13	100.00
Total	3,158	100.00	

* indicates an error

`cmsample` will always show you problems with the case ID variable and the alternatives variable. Suppose we do not care about them for now. Nor do we care about the missing values in the choice variable `purchase`. But we care about improper values in `purchase`. These are `cmsample`'s categories 9–13. So we list the observations for which `flag` has these values. We also include the case ID variable `consumerid` and the alternatives variable `car` in the listing.

```
. sort flag consumerid car
. list consumerid car purchase flag if inrange(flag, 9, 13),
> sepby(consumerid) abbr(10)
```

	consumerid	car	purchase	flag
3134.	101	American	0	choice sets of size one
3135.	80	American	0	choice variable not 0/1*
3136.	80	Japanese	2	choice variable not 0/1*
3137.	80	European	0	choice variable not 0/1*
3138.	80	Korean	0	choice variable not 0/1*
3139.	250	American	0	choice variable all 0
3140.	250	Japanese	0	choice variable all 0
3141.	250	European	0	choice variable all 0
3142.	250	Korean	0	choice variable all 0
3143.	540	Japanese	0	choice variable all 0
3144.	540	European	0	choice variable all 0
3145.	639	American	0	choice variable all 0
3146.	639	European	0	choice variable all 0
3147.	201	American	1	choice variable all 1
3148.	201	Japanese	1	choice variable all 1
3149.	201	European	1	choice variable all 1
3150.	201	Korean	1	choice variable all 1
3151.	202	American	0	choice variable multiple 1s
3152.	202	Japanese	1	choice variable multiple 1s
3153.	202	European	1	choice variable multiple 1s
3154.	202	Korean	1	choice variable multiple 1s

These are the cases we may wish to examine for possible data errors. Only the value of `purchase` $= 2$ in observation number 3136 must be fixed (or omitted with an `if` restriction). A value that is not

0, 1, or missing for a choice variable gives an error message. The other problematic observations are automatically dropped from the estimation sample, so they can be left in the dataset, and the estimation command will run.

◁

▷ Example 2: altwise handling of missing values

Let's illustrate the difference between casewise deletion (the default) and alternativewise deletion, which is done when you specify the `altwise` option.

We load the choice model dataset used in example 1 of [CM] **cmclogit**. The alternative-specific variable `dealers` has no missing values. Let's randomly change 5% of its values to missing.

```
. use https://www.stata-press.com/data/r18/carchoice, clear
(Car choice data)
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.
     Case ID variable: consumerid
Alternatives variable: car
. set seed 1
. replace dealers = . if runiform() < 0.05
(153 real changes made, 153 to missing)
```

We first run `cmsample` with the default casewise deletion:

```
. cmsample dealers, choice(purchase)
```

Reason for exclusion	Freq.	Percent	Cum.
observations included	2,638	83.48	83.48
varlist missing	522	16.52	100.00
Total	3,160	100.00	

Now with `altwise` deletion:

```
. cmsample dealers, choice(purchase) altwise gen(flag)
```

Reason for exclusion	Freq.	Percent	Cum.
observations included	2,898	91.71	91.71
varlist missing	153	4.84	96.55
choice sets of size one	3	0.09	96.65
choice variable all 0	106	3.35	100.00
Total	3,160	100.00	

Note that we randomly created 153 missing values; that's $153/3160 = 4.8\%$ of the total observations. The casewise deletion excluded 16.5% of the dataset's observations (522 out of 3160), whereas the `altwise` deletion excluded only 8.3% ($153 + 3 + 106 = 262$ out of 3160).

The `altwise` deletion introduced problems into the data. Now there are choice sets of size 1 and cases in which the choice variable is all 0. The casewise marked-out sample had only two distinct choice sets: $(1, 2, 3)$ and $(1, 2, 3, 4)$. After `altwise` deletion, there are nine choice sets. We can see the choice sets by typing `cmchoiceset` with the restriction `if flag == 0`, which includes only observations that would remain in an estimation sample.

```
. cmchoiceset if flag == 0
Tabulation of choice-set possibilities
 Choice set |      Freq.     Percent        Cum.
------------+-----------------------------------
        1 2 |         17        2.02        2.02
      1 2 3 |        349       41.55       43.57
    1 2 3 4 |        412       49.05       92.62
      1 2 4 |         19        2.26       94.88
        1 3 |         10        1.19       96.07
      1 3 4 |          9        1.07       97.14
        2 3 |          6        0.71       97.86
      2 3 4 |         17        2.02       99.88
        3 4 |          1        0.12      100.00
------------+-----------------------------------
      Total |        840      100.00
Note: Total is number of cases.
```

Be aware of the consequences of `altwise` deletion of missing values. ◁

Stored results

`cmsample` stores the following in `r()`:

Scalars

`r(N)`	number of observations both included and excluded
`r(N_included)`	number of observations included
`r(nc_included)`	number of cases included
`r(N_ex_if_in)`	number of observations excluded by `if` or `in`
`r(N_ex_caseid)`	number of observations excluded: case ID variable missing
`r(N_ex_size_1)`	number of observations excluded because of choice sets of size 1
`r(nc_ex_size_1)`	number of cases excluded because of choice sets of size 1
`r(N_ex_altvar)`	number of observations excluded because alternatives variable missing
`r(nc_ex_altvar)`	number of cases excluded because alternatives variable missing
`r(N_err_altvar)`	number of observations with repeated alternatives within case (error)
`r(nc_err_altvar)`	number of cases with repeated alternatives within case (error)
`r(N_ex_varlist)`	number of observations excluded because *varlist* missing
`r(nc_ex_varlist)`	number of cases excluded because *varlist* missing
`r(N_ex_wt)`	number of observations excluded because weight missing
`r(nc_ex_wt)`	number of cases excluded because weight missing
`r(N_err_wt_nc)`	number of observations with weight not constant within case (error)
`r(nc_err_wt_nc)`	number of cases with weight not constant within case (error)
`r(N_ex_choice)`	number of observations excluded because *choicevar* missing
`r(nc_ex_choice)`	number of cases excluded because *choicevar* missing
`r(N_ex_choice_0)`	number of observations excluded because *choicevar* all 0 for case
`r(nc_ex_choice_0)`	number of cases excluded because *choicevar* all 0 for case
`r(N_ex_choice_1)`	number of observations excluded because *choicevar* all 1 for case
`r(nc_ex_choice_1)`	number of cases excluded because *choicevar* all 1 for case
`r(N_ex_choice_011)`	number of observations excluded because *choicevar* has multiple 1s for case
`r(nc_ex_choice_011)`	number of cases excluded because *choicevar* has multiple 1s for case
`r(N_err_choice)`	number of observations with *choicevar* not 0/1 (error)
`r(nc_err_choice)`	number of cases with *choicevar* not 0/1 (error)
`r(N_ex_casevar)`	number of observations excluded because *casevars* missing
`r(nc_ex_casevar)`	number of cases excluded because *casevars* missing
`r(N_err_casevar_nc)`	number of observations with *casevars* not constant within case (error)
`r(nc_err_casevar_nc)`	number of cases with *casevars* not constant within case (error)
`r(N_ex_time)`	number of observations excluded because *timevar* missing

Macros

`r(caseid)`	name of case ID variable
`r(altvar)`	name of alternatives variable (if set)
`r(timevar)`	name of time variable (if panel data)
`r(marktype)`	`casewise` or `altwise`, type of markout

Also see

[CM] **cmchoiceset** — Tabulate choice sets

[CM] **cmset** — Declare data to be choice model data

[CM] **cmsummarize** — Summarize variables by chosen alternatives

[CM] **cmtab** — Tabulate chosen alternatives

Title

cmset — Declare data to be choice model data

Description Quick start Menu Syntax
Options Remarks and examples Stored results Also see

Description

`cmset` manages the choice model settings of a dataset. You use `cmset` to declare the data in memory to be choice model data. With cross-sectional data, you designate which variables identify cases and alternatives. With panel data, you designate which variables identify panels, time periods, and alternatives. You must `cmset` your data before you can use the other `cm` commands.

`cmset` without arguments displays how the data are currently set. `cmset` also sorts the data based on the variables that identify cases, alternatives, and panels.

Quick start

Declare dataset to be choice model data with case identifier `caseid` and alternatives (choice-set) identifier `choiceset`

```
cmset caseid choiceset
```

Declare dataset to be choice model data with unspecified alternatives

```
cmset caseid, noalternatives
```

Declare dataset to be panel choice model data with panel identifier `pvar`, time identifier `tvar`, and alternatives identifier `choiceset`

```
cmset pvar tvar choiceset
```

Declare dataset to be panel choice model data with unspecified alternatives

```
cmset pvar tvar, noalternatives
```

Indicate that observations in the panel choice model data are made monthly; `tvar2` is not formatted

```
cmset pvar tvar2 choiceset, monthly
```

Same as above, and apply `%tm` format to `tvar2`

```
cmset pvar tvar2 choiceset, format(%tm)
```

View `cm` settings

```
cmset
```

Menu

Statistics > Choice models > Setup and utilities > Declare data to be choice model data

Syntax

Declare data to be cross-sectional choice model data

```
cmset caseidvar altvar [ , force ]
cmset caseidvar, noalternatives
```

Declare data to be panel choice model data

```
cmset panelvar timevar altvar [ , tsoptions force ]
cmset panelvar timevar, noalternatives
```

Display how data are currently cmset

```
cmset
```

Clear cm settings

```
cmset, clear
```

caseidvar identifies the cases in the cross-sectional data syntax.

altvar identifies the alternatives (choice sets).

panelvar identifies the panels, and *timevar* identifies the times within panels.

tsoptions	Description
unitoptions	specify units of *timevar*
deltaoption	specify period between observations in *timevar* units

unitoptions	Description
(*default*)	*timevar*'s units to be obtained from *timevar*'s display format
clocktime	*timevar* is %tc: 0 = 1jan1960 00:00:00.000, 1 = 1jan1960 00:00:00.001, ...
daily	*timevar* is %td: 0 = 1jan1960, 1 = 2jan1960, ...
weekly	*timevar* is %tw: 0 = 1960w1, 1 = 1960w2, ...
monthly	*timevar* is %tm: 0 = 1960m1, 1 = 1960m2, ...
quarterly	*timevar* is %tq: 0 = 1960q1, 1 = 1960q2,...
halfyearly	*timevar* is %th: 0 = 1960h1, 1 = 1960h2,...
yearly	*timevar* is %ty: 1960 = 1960, 1961 = 1961, ...
generic	*timevar* is %tg: 0 = ?, 1 = ?, ...
format(%*fmt*)	specify *timevar*'s format and then apply default rule

In all cases, negative *timevar* values are allowed.

deltaoption	Example
delta(#)	delta(1) or delta(2)
delta((*exp*))	delta((7*24))
delta(# *units*)	delta(7 days) or delta(15 minutes) or delta(7 days 15 minutes)
delta((*exp*) *units*)	delta((2+3) weeks)

Allowed units for %tc and %tC *timevars* are

seconds	second	secs	sec
minutes	minute	mins	min
hours	hour		
days	day		
weeks	week		

and for all other %t *timevars* are

days	day
weeks	week

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

Options

noalternatives specifies that alternatives are not explicitly identified. That is, there is no alternatives variable. The default is that you must specify an alternatives variable.

force suppresses error messages caused by the alternatives variable *altvar*. This option is rarely used. The alternatives variable must be free of errors before cm commands can run, so this option changes only the point at which error messages will be issued. One use of the force option is to specify it with cmset and then run cmsample to identify the observations with bad values for the alternatives variable. force does not suppress all error messages. Error messages in the case ID variable and error messages in the time variable for panel data are not suppressed.

unitoptions clocktime, daily, weekly, monthly, quarterly, halfyearly, yearly, generic, and format(*%fmt*) specify the units in which *timevar* is recorded when *timevar* is specified.

timevar will often simply be a variable of counts such as 1, 2, ..., or years such as 2001, 2002, ... In other cases, *timevar* will be a formatted %t variable; see [D] **Datetime**. In any of these cases, you do not need to specify a *unitoption*.

Only when *timevar* is an unformatted time variable would you use these options. When you cmset panel choice model data, it becomes xtset as well. These options are simply passed to xtset. See [XT] **xtset** for option details.

delta() specifies the period of *timevar* and is commonly used when *timevar* is %tc or %tC. delta() is rarely used with other %t formats or with unformatted time variables. If delta() is not specified, delta(1) is assumed. See [XT] **xtset** for option details.

clear—used in cmset, clear—makes Stata forget that the data were ever cmset. This option is rarely used. Note that if you cmset your data as panel choice model data with an alternatives variable, they also become xtset. Typing cmset, clear does not clear the xt settings. To do this, you must type xtset, clear as well.

Remarks and examples

cmset declares the dataset in memory to be choice model data. You need to do this before you can use the other cm commands.

cmset sets cross-sectional choice data and panel choice data. The usual syntax for cross-sectional data is to give cmset two variables:

cmset *caseidvar altvar*

The case ID variable *caseidvar* must be numeric, and its values must be integers. The variable *altvar* containing the alternatives can be either numeric or string.

The usual syntax for panel data is to give `cmset` three variables:

`cmset` *panelvar timevar altvar*

The variable *panelvar* identifies panels, which are typically IDs for individuals or decision makers. The variable *timevar* identifies times within panels, points at which choices were made. Both *panelvar* and *timevar* must be numeric, and both must contain integers only.

For some choice models, alternatives are not explicitly identified. Alternatives are known only by their characteristics as given by alternative-specific variables. In this case, the syntax for cross-sectional data is

`cmset` *caseidvar*`, noalternatives`

and the syntax for panel data is

`cmset` *panelvar timevar*`, noalternatives`

For a brief introduction to other choice models, see [CM] **Intro 4**.

▷ Example 1: Cross-sectional choice data

Here is an example of cross-sectional choice data:

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. list consumerid car purchase if consumerid <= 4, sepby(consumerid) abbr(10)
```

	consumerid	car	purchase
1.	1	American	1
2.	1	Japanese	0
3.	1	European	0
4.	1	Korean	0
5.	2	American	1
6.	2	Japanese	0
7.	2	European	0
8.	2	Korean	0
9.	3	American	0
10.	3	Japanese	1
11.	3	European	0
12.	4	American	1
13.	4	Japanese	0
14.	4	European	0

The variable `consumerid` is the case ID variable, and the variable `car` defines the alternatives. These fictitious data represent persons who purchased a car with their choices categorized by the nationality of the manufacturer, American, Japanese, European, or Korean.

To declare the data to be `cm` data, we type

```
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.

     Case ID variable: consumerid
Alternatives variable: car
```

We have to cmset our data only once if we save our data after we cmset it. Let's illustrate this. Typing cmset without arguments shows the current settings.

```
. save carchoice_cmset
file carchoice_cmset.dta saved
. use carchoice_cmset
(Car choice data)
. cmset
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.
     Case ID variable: consumerid
Alternatives variable: car
```

For these data, the choice sets are unbalanced, and cmset gave us a message telling us this. If we want to see the distinct choice-set possibilities, we can type cmchoiceset:

```
. cmchoiceset
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2 3	380	42.94	42.94
1 2 3 4	505	57.06	100.00
Total	885	100.00	

```
Note: Total is number of cases.
```

Example 2: Data errors with cmset

If there were errors in the alternatives variable, cmset would give an error message. Here is an example with a dataset where we added errors:

```
. use https://www.stata-press.com/data/r18/carchoice_errors, clear
(Car choice data with errors)
. cmset consumerid car
at least one choice set has more than one instance of the same alternative
r(459);
```

When cmset detects errors in the alternatives variable, you may want to type cmset again with the option force, and then use cmsample:

```
. cmset consumerid car, force
note: at least one choice set has more than one instance of the same
      alternative.
     Case ID variable: consumerid
Alternatives variable: car
. cmsample, generate(flag)
```

Reason for exclusion	Freq.	Percent	Cum.
observations included	3,153	99.78	99.78
repeated alternatives within case*	7	0.22	100.00
Total	3,160	100.00	

```
* indicates an error
```

```
. list consumerid car flag if flag != 0, sepby(consumerid) abbr(10)
```

	consumerid	car	flag
397.	111	American	repeated alternatives within case*
398.	111	Japanese	repeated alternatives within case*
399.	111	Japanese	repeated alternatives within case*
1035.	290	American	repeated alternatives within case*
1036.	290	Japanese	repeated alternatives within case*
1037.	290	Japanese	repeated alternatives within case*
1038.	290	Korean	repeated alternatives within case*

Some `cm` estimators such as `cmrologit` do not require an alternatives variable. In this case, you use the `noalternatives` option and just specify the case ID variable:

```
. cmset consumerid, noalternatives
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.

     Case ID variable: consumerid
Alternatives variable: <none>
```

◁

▷ Example 3: Panel choice data

When you have panel choice data, you will have a panel ID variable and a time variable. Typically, you will also have a variable specifying the alternatives.

Here is an example in which `id` is the panel ID variable, `t` is the time variable, and variable `alt` contains the alternatives. The first panel of these data looks like

```
. use https://www.stata-press.com/data/r18/transport, clear
(Transportation choice data)
. list id t alt if id == 1, sepby(t)
```

	id	t	alt
1.	1	1	Car
2.	1	1	Public
3.	1	1	Bicycle
4.	1	1	Walk
5.	1	2	Car
6.	1	2	Public
7.	1	2	Bicycle
8.	1	2	Walk
9.	1	3	Car
10.	1	3	Public
11.	1	3	Bicycle
12.	1	3	Walk

To cmset the data, we type

```
. cmset id t alt
note: case identifier _caseid generated from id and t.
note: panel by alternatives identifier _panelaltid generated from id and alt.

                     Panel data: Panels id and time t
               Case ID variable: _caseid
          Alternatives variable: alt
Panel by alternatives variable: _panelaltid (strongly balanced)
                  Time variable: t, 1 to 3
                          Delta: 1 unit

Note: Data have been xtset.
```

Look at the notes displayed by cmset. It has created two new variables: _caseid and _panelaltid. Let's list their values for the first two panels.

```
. sort id t alt
. list id t alt _caseid _panelaltid if inlist(id, 1, 2), sepby(t) abbr(11)
```

	id	t	alt	_caseid	_panelaltid
1.	1	1	Car	1	1
2.	1	1	Public	1	2
3.	1	1	Bicycle	1	3
4.	1	1	Walk	1	4
5.	1	2	Car	2	1
6.	1	2	Public	2	2
7.	1	2	Bicycle	2	3
8.	1	2	Walk	2	4
9.	1	3	Car	3	1
10.	1	3	Public	3	2
11.	1	3	Bicycle	3	3
12.	1	3	Walk	3	4
13.	2	1	Car	4	5
14.	2	1	Public	4	6
15.	2	1	Bicycle	4	7
16.	2	1	Walk	4	8
17.	2	2	Car	5	5
18.	2	2	Public	5	6
19.	2	2	Bicycle	5	7
20.	2	2	Walk	5	8
21.	2	3	Car	6	5
22.	2	3	Public	6	6
23.	2	3	Bicycle	6	7
24.	2	3	Walk	6	8

_caseid is a variable that identifies cases. For choice model data, remember that a case is a single statistical observation but consists of multiple Stata observations. Each distinct value of panel ID × time represents a single statistical observation, that is, a case. The values of _caseid correspond to the distinct values of panel ID × time, in this example, the values of id × t.

_panelaltid is a variable that uniquely identifies the distinct values of panel ID × alternative. Why do you need this variable? It is created so you can use Stata's time-series operators. Imagine that you want to include lags of alternative-specific variables in your model. The lags must be specific to the alternative, and Stata's time-series lag operator needs to know how to do this.

When you cmset panel data with specified alternatives, your data are automatically xtset. You can type xtset to see the settings:

```
. xtset
Panel variable: _panelaltid (strongly balanced)
 Time variable: t, 1 to 3
         Delta: 1 unit
```

_panelaltid becomes the "panel" identifier viewing the data as xt data. See [CM] **Intro 7** and [CM] **cmxtmixlogit** for more on using time-series operators with panel CM data.

cmxtmixlogit allows you to fit a model with unspecified alternatives. To do this, you use the option noalternatives:

```
. use https://www.stata-press.com/data/r18/transport, clear
(Transportation choice data)
. cmset id t, noalternatives
note: case identifier _caseid generated from id and t.
            Panel data: Panels id and time t
      Case ID variable: _caseid
Alternatives variable: <none>
         Time variable: t, 1 to 3
                 Delta: 1 unit
```

_caseid is again created, and its values are the same as in the previous cmset results.

There is no _panelaltid variable because there are no specified alternatives. The data are not xtset because there is no way to match up the alternatives.

```
. xtset
panel variable not set; use xtset varname ...
r(459);
```

Because the data are not xtset, you cannot use time-series operators for panel CM models with unspecified alternatives.

◁

Stored results

cmset stores the following in r():

Scalars

r(n_cases)	number of cases
r(n_alt_min)	minimum number of alternatives per case
r(n_alt_avg)	average number of alternatives per case
r(n_alt_max)	maximum number of alternatives per case
r(altvar_min)	minimum of alternatives variable (if set when numeric)
r(altvar_max)	maximum of alternatives variable (if set when numeric)

Macros

r(caseid)	name of case ID variable
r(altvar)	name of alternatives variable (if set)

For panel data, cmset also stores the following in r():

Scalars

r(imin)	minimum panel ID
r(imax)	maximum panel ID
r(tmin)	minimum time
r(tmax)	maximum time
r(tdelta)	delta
r(gaps)	1 if there are gaps, 0 otherwise

Macros

r(origpanelvar)	name of original panel variable passed to cmset
r(panelvar)	name of panel variable
r(timevar)	name of time variable
r(tdeltas)	formatted delta
r(tmins)	formatted minimum time
r(tmaxs)	formatted maximum time
r(tsfmt)	*%fmt* of time variable
r(unit)	units of time variable: Clock, clock, daily, weekly, monthly, quarterly, halfyearly, yearly, or generic
r(unit1)	units of time variable: C, c, d, w, m, q, h, y, or ""
r(balanced)	unbalanced, weakly balanced, or strongly balanced; a set of panels are strongly balanced if they all have the same time values, otherwise weakly balanced if same number of time values, otherwise unbalanced

Also see

[CM] **cmchoiceset** — Tabulate choice sets

[CM] **cmsample** — Display reasons for sample exclusion

[CM] **cmsummarize** — Summarize variables by chosen alternatives

[CM] **cmtab** — Tabulate chosen alternatives

[XT] **xtset** — Declare data to be panel data

Title

cmsummarize — Summarize variables by chosen alternatives

Description Quick start Menu Syntax
Options Remarks and examples Also see

Description

`cmsummarize` calculates summary statistics for one or more variables grouped by chosen alternatives.

For panel choice data, `cmsummarize` calculates summary statistics grouped by chosen alternatives and by time.

Quick start

Display the means of `x1` and `x2` grouped by chosen alternatives identified by `depvar` and using `cmset` data

```
cmsummarize x1 x2, choice(depvar)
```

Same as above, and display sample size, minimum, median, and maximum

```
cmsummarize x1 x2, choice(depvar) statistics(N min median max)
```

For panel choice data, display means of `xvar` grouped by chosen alternatives and time

```
cmsummarize xvar, choice(depvar) time
```

Menu

Statistics > Choice models > Setup and utilities > Summarize variables by chosen alternatives

Syntax

```
cmsummarize varlist [if] [in] [weight] , choice(choicevar) [options]
```

options	Description
Main	
* `choice(`*choicevar*`)`	specify 0/1 variable indicating the chosen alternatives
`statistics(`*statname*`[...])`	report specified statistics
`altwise`	use alternativewise deletion instead of casewise deletion
Reporting	
`format[(`*%fmt*`)]`	display format for statistics; default format is `%9.0g`
`longstub`	put key for statistics (or variable names) on left table stub
`time`	group by time variable (only for panel CM data)
`columns(variables)`	display variables in table columns; the default
`columns(statistics)`	display statistics in table columns

* `choice()` is required.

You must `cmset` your data before using `cmsummarize`; see [CM] **cmset**.

`by` is allowed; see [D] **by**.

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

Options

Main

`choice(`*choicevar*`)` specifies the variable indicating the chosen alternative. *choicevar* must be coded as 0 and 1, with 0 indicating an alternative that was not chosen and 1 indicating the chosen alternative. `choice()` is required.

`statistics(`*statname* `[...])` specifies the statistics to be displayed; the default is equivalent to specifying `statistics(mean)`. (`stats()` is a synonym for `statistics()`.) Multiple statistics may be specified and are separated by white space, such as `statistics(mean sd)`. Available statistics are

statname	Definition	*statname*	Definition
`mean`	mean	`p1`	1st percentile
`count`	count of nonmissing observations	`p5`	5th percentile
`n`	same as `count`	`p10`	10th percentile
`sum`	sum	`p25`	25th percentile
`max`	maximum	`median`	median (same as `p50`)
`min`	minimum	`p50`	50th percentile (same as `median`)
`range`	range = `max` − `min`	`p75`	75th percentile
`sd`	standard deviation	`p90`	90th percentile
`variance`	variance	`p95`	95th percentile
`cv`	coefficient of variation (`sd/mean`)	`p99`	99th percentile
`semean`	standard error of mean ($\texttt{sd}/\sqrt{\texttt{n}}$)	`iqr`	interquartile range = `p75` − `p25`
`skewness`	skewness	`q`	equivalent to specifying `p25 p50 p75`
`kurtosis`	kurtosis		

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

Reporting

`format` and `format(%`*fmt*`)` specify how the statistics are to be formatted. The default is to use a `%9.0g` format.

`format` specifies that each variable's statistics be formatted with the variable's display format; see [D] **format**.

`format(%`*fmt*`)` specifies the format to be used for all statistics. The maximum width of the specified format should not exceed nine characters.

`longstub` specifies that the left stub of the table be made wider so that it can include names of the statistics (or variable names when `columns(statistics)` is specified) in addition to the categories of the alternatives. The default is to display the names of the statistics (or variable names) in a header.

`time` groups the statistics by values of the time variable when data are panel choice data. See [CM] **cmset**.

`columns(variables | statistics)` specifies whether to display variables or statistics in the columns of the table. `columns(variables)` is the default when more than one variable is specified.

Remarks and examples

`cmsummarize` is a convenience command for displaying summary statistics of one or more variables grouped by chosen alternatives.

The option `choice(`*choicevar*`)` is required, where *choicevar* is a 0/1 variable. *choicevar* is typically the dependent variable for choice models with 0/1 dependent variables.

For rank-ordered choice models, such as `cmroprobit`, using a dependent variable of ranks with `choice()` will give an error message. To use `cmsummarize` in this instance, you would have to create a 0/1 variable, such as a variable indicating the highest ranked alternative for each case.

For an overview of other descriptive statistics available for choice model data, see [CM] **Intro 3**.

▷ Example 1: Cross-sectional choice data

Here is an example with cross-sectional choice data. First, we `cmset` our data:

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.
     Case ID variable: consumerid
Alternatives variable: car
```

These fictitious data represent persons who purchased a car with their choices categorized by the nationality of the manufacturer, American, Japanese, European, or Korean. Statistics are calculated over groups defined by the chosen alternatives, that is, the nationality of car. Second, we type `cmsummarize`, which by default calculates means. Specifying the variable `income`, we get the means of `income` by the nationality of car purchased.

```
. cmsummarize income, choice(purchase)
Statistics by chosen alternatives (purchase = 1)
    income is constant within case
Summary for variables: income
Group variable: _chosen_alternative (purchase = 1)
```

_chosen_alternative	Mean
American	40.52394
Japanese	43.15127
European	45.80462
Korean	35.585
Total	42.05429

The mean income is highest among those that selected European cars.

Third, we specify the option `statistics(N min mean max)` to display the group sample size and the minimum, mean, and maximum of the variables `gender`, `income`, and `dealers`.

```
. cmsummarize gender income dealers, choice(purchase) statistics(N min mean max)
Statistics by chosen alternatives (purchase = 1)
    variables constant within case:
        gender
        income
Summary statistics: N, Min, Mean, Max
Group variable: _chosen_alternative (purchase = 1)
```

_chosen_alternative	gender	income	dealers
American	376	376	376
	0	20.3	2
	.7446809	40.52394	8.143617
	1	69.8	13
Japanese	316	316	316
	0	20.3	1
	.6518987	43.15127	6.25
	1	69.8	12
European	130	130	130
	0	20.3	1
	.8307692	45.80462	3.630769
	1	69.8	7
Korean	40	40	40
	0	20.9	1
	.8	35.585	2.425
	1	69.8	5
Total	862	862	862
	0	20.3	1
	.7262181	42.05429	6.50348
	1	69.8	13

Example 2: Panel choice data

When you have panel choice data, `cmsummarize` is useful to see how summary statistics grouped by chosen alternatives vary by time. Here is an example. First, we `cmset` the data:

```
. use https://www.stata-press.com/data/r18/transport, clear
(Transportation choice data)
. cmset id t alt
note: case identifier _caseid generated from id and t.
note: panel by alternatives identifier _panelaltid generated from id and alt.
                    Panel data: Panels id and time t
              Case ID variable: _caseid
         Alternatives variable: alt
Panel by alternatives variable: _panelaltid (strongly balanced)
                 Time variable: t, 1 to 3
                         Delta: 1 unit
Note: Data have been xtset.
```

Second, we specify the option `time`, which produces statistics grouped by chosen alternatives at each time point. We also specify the formatting for the statistics.

```
. cmsummarize trtime, choice(choice) statistics(median) format(%6.4f) time
Statistics by chosen alternatives (choice = 1)
time t = 1
Summary for variables: trtime
Group variable: _chosen_alternative (choice = 1)
```

_chosen_alternative	p50
Car	0.1764
Public	0.4195
Bicycle	0.5884
Walk	0.8054
Total	0.2316

```
time t = 2
Summary for variables: trtime
Group variable: _chosen_alternative (choice = 1)
```

_chosen_alternative	p50
Car	0.1731
Public	0.3729
Bicycle	0.6562
Walk	0.6671
Total	0.1897

```
time t = 3
Summary for variables: trtime
Group variable: _chosen_alternative (choice = 1)
```

_chosen_alternative	p50
Car	0.1842
Public	0.4345
Bicycle	0.4593
Walk	0.9563
Total	0.2006

If we do not specify the option `time`, statistics are calculated by the groups of chosen alternatives aggregated across time.

```
. cmsummarize trtime, choice(choice) statistics(N min median max) format(%6.0g)
Statistics by chosen alternatives (choice = 1)
Summary for variables: trtime
Group variable: _chosen_alternative (choice = 1)
```

_chosen_alternative	N	Min	p50	Max
Car	981	.1	.1789	.2499
Public	256	.1016	.4171	.7024
Bicycle	145	.102	.573	1.292
Walk	118	.1019	.8126	1.993
Total	1500	.1	.2055	1.993

Also see

[CM] **cmchoiceset** — Tabulate choice sets

[CM] **cmsample** — Display reasons for sample exclusion

[CM] **cmset** — Declare data to be choice model data

[CM] **cmtab** — Tabulate chosen alternatives

Title

cmtab — Tabulate chosen alternatives

Description Quick start Menu Syntax
Options Remarks and examples Stored results Also see

Description

`cmtab` tabulates chosen alternatives, either alone in a one-way tabulation or versus another variable in a two-way tabulation.

For panel choice data, `cmtab` can display a two-way tabulation of chosen alternatives by time or a three-way tabulation of time by chosen alternative by another variable.

Quick start

Display a one-way tabulation of chosen alternatives for `cmset` data, where `depvar` is a 0/1 variable

```
cmtab, choice(depvar)
```

Tabulate chosen alternatives versus the values of variable `xvar`

```
cmtab xvar, choice(depvar)
```

Same as above, and report row percentages and Pearson's χ^2 test

```
cmtab xvar, choice(depvar) row chi2
```

Transpose rows and columns in the above tabulation

```
cmtab xvar, choice(depvar) col chi2 transpose
```

For panel choice data, display a two-way tabulation of chosen alternatives versus the time variable

```
cmtab, choice(depvar) time
```

For panel choice data, display tabulations of chosen alternatives versus `x` for each time

```
cmtab x, choice(depvar) time
```

Same as above, but display tabulations of chosen alternatives versus times for each value of `x`

```
cmtab x, choice(depvar) time timelast
```

Same as above, but create a compact display

```
cmtab x, choice(depvar) time timelast compact
```

Menu

Statistics > Choice models > Setup and utilities > Tabulate chosen alternatives

Syntax

```
cmtab [varname] [if] [in] [weight], choice(choicevar) [options]
```

options	Description
Main	
* `choice(`*choicevar*`)`	specify 0/1 variable indicating the chosen alternative
`missing`	include missing values of *varname* in tabulation
`transpose`	transpose rows and columns in tables
`time`	tabulate by time variable (only for panel CM data)
`timelast`	put time variable last in three-way tabulation; tabulate alternatives by time for each level of *varname* (only for panel CM data)
`compact`	display three-way tabulation compactly (only for panel CM data)
`altwise`	use alternativewise deletion instead of casewise deletion
Options	
tab1_options	options for one-way tables
tab2_options	options for two-way tables

* `choice()` is required.

tab1_options	Description
`sort`	display table in descending order of frequency

tab2_options	Description
`chi2`	report Pearson's χ^2
`lrchi2`	report likelihood-ratio χ^2
`column`	report column percentages
`row`	report row percentages
`cell`	report cell percentages
`rowsort`	list rows in order of observed frequency
`colsort`	list columns in order of observed frequency
[`no`]`key`	report or suppress cell contents key

You must `cmset` your data before using `cmtab`; see [CM] **cmset**.
`by` and `collect` are allowed; see [U] **11.1.10 Prefix commands**.
`fweights` and `iweights` are allowed; see [U] **11.1.6 weight**.

Options

Main

`choice(`*choicevar*`)` specifies the variable indicating the chosen alternative. *choicevar* must be coded as 0 and 1, with 0 indicating an alternative that was not chosen and 1 indicating the chosen alternative. `choice()` is required.

`missing` specifies that the missing values of *varname* are to be treated like any other value of *varname*.

`transpose` transposes rows and columns in the tabular displays.

`time` tabulates the chosen alternative versus the time variable when data are panel choice data. See [CM] **cmset**.

`timelast` puts time last in a three-way tabulation when data are panel choice data. Three-way tabulations are created when *varname* is specified as well as the option `time`. By default, the three-way tabulation is *timevar* × chosen alternative × *varname*; that is, for each value of *timevar*, a two-way table of chosen alternative versus *varname* is displayed. When `timelast` is specified, the three-way tabulation is *varname* × chosen alternative × *timevar*; that is, for each value of *varname*, a two-way table of chosen alternative versus *timevar* is displayed. To reverse the order of the two-way tabulations, you can use the option `transpose`.

`compact` creates a compact three-way tabulation when data are panel choice data.

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

Options

`sort` puts the table in descending order of frequency in a one-way table.

`chi2` calculates and displays Pearson's χ^2 for the hypothesis that the rows and columns in a two-way table are independent. `chi2` may not be specified if `iweights` are used. `chi2` is not available when `compact` is specified.

`lrchi2` displays the likelihood-ratio χ^2 statistic for a two-way table. `lrchi2` may not be specified if `iweights` are used. `lrchi2` is not available when `compact` is specified.

`column` displays the relative frequency, as a percentage, of each cell within its column in a two-way table. `column` is not available when `compact` is specified.

`row` displays the relative frequency, as a percentage, of each cell within its row in a two-way table. `row` is not available when `compact` is specified.

`cell` displays the relative frequency, as a percentage, of each cell in a two-way table. `cell` is not available when `compact` is specified.

`rowsort` and `colsort` specify that the rows and columns, respectively, be presented in order of observed frequency in a two-way table. `rowsort` and `colsort` are not available when `compact` is specified.

[`no`]`key` displays or suppresses a key above two-way tables. The default is to display the key if more than one cell statistic is requested. `key` displays the key. `nokey` suppresses its display. [`no`]`key` is not available when `compact` is specified.

Remarks and examples

`cmtab` is a convenience command for tabulating chosen alternatives, either alone or against another variable.

The option `choice(`*choicevar*`)` is required, where *choicevar* is a 0/1 variable. *choicevar* is typically the dependent variable for choice models with 0/1 dependent variables.

For rank-ordered choice models, such as cmroprobit, using a dependent variable of ranks with choice() will give an error message. To use cmtab in this instance, you would have to create a 0/1 variable, such as a variable indicating the highest ranked alternative for each case.

For tabulations of choice sets, see [CM] **cmchoiceset**. For an overview of other descriptive statistics available for choice model data, see [CM] **Intro 3**.

▷ Example 1: Cross-sectional choice data

Here is an example with cross-sectional choice data. First, we cmset our data:

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.
     Case ID variable: consumerid
Alternatives variable: car
```

These fictitious data represent persons who purchased a car with their choices categorized by the nationality of the manufacturer, American, Japanese, European, or Korean. Second, we use cmtab with only the choice() option, which gives a one-way tabulation of the chosen alternatives.

```
. cmtab, choice(purchase)
Tabulation of chosen alternatives (purchase = 1)
```

Nationality of car	Freq.	Percent	Cum.
American	384	43.39	43.39
Japanese	326	36.84	80.23
European	135	15.25	95.48
Korean	40	4.52	100.00
Total	885	100.00	

We see that most people in this dataset purchased American cars more than any other nationality of car.

We can look at associations between chosen alternatives and other variables in the dataset. We wonder whether gender is associated with the nationality of the car purchased:

```
. cmtab gender, choice(purchase)
Tabulation for chosen alternatives (purchase = 1)
    gender is constant within case
```

Nationality of car	Gender: 0 = Female, 1 = Male: Female	Male	Total
American	96	280	376
Japanese	110	206	316
European	22	108	130
Korean	8	32	40
Total	236	626	862

We specify the option `row` to better see the percentages of gender within choices. We also specify `chi2` to get a p-value for the association of gender with the choice of car.

```
. cmtab gender, choice(purchase) row chi2
Tabulation for chosen alternatives (purchase = 1)
    gender is constant within case
```

Key
frequency *row percentage*

Nationality of car	Gender: 0 = Female, 1 = Male Female	Male	Total
American	96 25.53	280 74.47	376 100.00
Japanese	110 34.81	206 65.19	316 100.00
European	22 16.92	108 83.08	130 100.00
Korean	8 20.00	32 80.00	40 100.00
Total	236 27.38	626 72.62	862 100.00

Pearson chi2(3) = 17.6654 Pr = 0.001

There are more male car purchasers than female car purchasers in these data. Purchasers of European cars are even more overwhelmingly male. However, the percentage of Japanese cars purchased by females is greater than the percentage of American, European, or Korean cars purchased by females. The p-value from the Pearson's χ^2 test for association is 0.001.

The `transpose` option transposes rows and columns in the display:

```
. cmtab gender, choice(purchase) row chi2 nokey transpose
Tabulation for chosen alternatives (purchase = 1)
    gender is constant within case
```

Gender: 0 = Female, 1 = Male	Nationality of car American	Japanese	European	Korean	Total
Female	96 40.68	110 46.61	22 9.32	8 3.39	236 100.00
Male	280 44.73	206 32.91	108 17.25	32 5.11	626 100.00
Total	376 43.62	316 36.66	130 15.08	40 4.64	862 100.00

Pearson chi2(3) = 17.6654 Pr = 0.001

◁

Example 2: Panel choice data

When you have panel choice data, `cmtab` is useful to see how chosen alternatives vary by time. Here is an example. First, we `cmset` the data:

```
. use https://www.stata-press.com/data/r18/transport, clear
(Transportation choice data)

. cmset id t alt
note: case identifier _caseid generated from id and t.
note: panel by alternatives identifier _panelaltid generated from id and alt.

                    Panel data: Panels id and time t
              Case ID variable: _caseid
         Alternatives variable: alt
Panel by alternatives variable: _panelaltid (strongly balanced)
                 Time variable: t, 1 to 3
                         Delta: 1 unit

Note: Data have been xtset.
```

Second, we specify the option `time` to look at chosen alternatives by time. The option `column` helps to see whether there is any trend with time.

```
. cmtab, choice(choice) time column chi2
Tabulation of chosen alternatives (choice = 1) by time t
```

Key
frequency *column percentage*

Alternatives	Time variable 1	2	3	Total
Car	234	359	388	981
	46.80	71.80	77.60	65.40
Public	108	81	67	256
	21.60	16.20	13.40	17.07
Bicycle	74	40	31	145
	14.80	8.00	6.20	9.67
Walk	84	20	14	118
	16.80	4.00	2.80	7.87
Total	500	500	500	1,500
	100.00	100.00	100.00	100.00

```
Pearson chi2(6) = 148.9651   Pr = 0.000
```

There is a large time trend for the chosen alternatives in these data. The percentage of persons choosing cars as their mode of transportation increases from 46.8% at time 1 to 77.6% at time 3. All the other choices of modes of transportation decline over time.

Does choice of transportation vary by whether a person has a full-time or part-time job (indicated by the variable `parttime`)? Here is how we could look at that, aggregating across time.

```
. cmtab parttime, choice(choice) column nokey
Tabulation for chosen alternatives (choice = 1)
    parttime is constant within case
```

Alternativ es	Part-time job Full-time	Part-time	Total
Car	503 66.80	478 63.99	981 65.40
Public	132 17.53	124 16.60	256 17.07
Bicycle	72 9.56	73 9.77	145 9.67
Walk	46 6.11	72 9.64	118 7.87
Total	753 100.00	747 100.00	1,500 100.00

Because this tabulation aggregates chosen alternatives across time for the same individual, we did not calculate a Pearson χ^2. However, there does not appear to be an association between choice of transportation and whether the person is employed full time or part time.

Let's look at the choice of transportation by full-time or part-time employment for each time point. To do this, we add the option `time`. We also specify the option `transpose` to make wide tables that take up less vertical space. Because we are not aggregating counts, we also specify the `chi2` option.

```
. cmtab parttime, choice(choice) row chi2 nokey time transpose
Tabulations by chosen alternatives (choice = 1)
    parttime is constant within case
    time t = 1
```

Part-time job	Alternatives: Car	Public	Bicycle	Walk	Total
Full-time	119	53	34	31	237
	50.21	22.36	14.35	13.08	100.00
Part-time	115	55	40	53	263
	43.73	20.91	15.21	20.15	100.00
Total	234	108	74	84	500
	46.80	21.60	14.80	16.80	100.00

Pearson chi2(3) = 5.0154 Pr = 0.171

time t = 2

Part-time job	Alternatives: Car	Public	Bicycle	Walk	Total
Full-time	186	43	18	11	258
	72.09	16.67	6.98	4.26	100.00
Part-time	173	38	22	9	242
	71.49	15.70	9.09	3.72	100.00
Total	359	81	40	20	500
	71.80	16.20	8.00	4.00	100.00

Pearson chi2(3) = 0.8683 Pr = 0.833

time t = 3

Part-time job	Alternatives: Car	Public	Bicycle	Walk	Total
Full-time	198	36	20	4	258
	76.74	13.95	7.75	1.55	100.00
Part-time	190	31	11	10	242
	78.51	12.81	4.55	4.13	100.00
Total	388	67	31	14	500
	77.60	13.40	6.20	2.80	100.00

Pearson chi2(3) = 5.2158 Pr = 0.157

Is there a time trend for choice of transportation for those employed full time? For those employed part time? The tables above can be considered a three-way tabulation: time × `parttime` × chosen alternative. To look for time trends within `parttime`, we note the three-way tabulation `parttime` × chosen alternative × time is better. We can get this three-way tabulation by specifying the option `timelast`.

```
. cmtab parttime, choice(choice) column chi2 nokey time timelast
Tabulations by chosen alternatives (choice = 1)
    parttime is constant within case
    parttime = 0
```

Alternativ es	Time variable 1	2	3	Total
Car	119	186	198	503
	50.21	72.09	76.74	66.80
Public	53	43	36	132
	22.36	16.67	13.95	17.53
Bicycle	34	18	20	72
	14.35	6.98	7.75	9.56
Walk	31	11	4	46
	13.08	4.26	1.55	6.11
Total	237	258	258	753
	100.00	100.00	100.00	100.00

Pearson chi2(6) = 57.2439 Pr = 0.000

parttime = 1

Alternativ es	Time variable 1	2	3	Total
Car	115	173	190	478
	43.73	71.49	78.51	63.99
Public	55	38	31	124
	20.91	15.70	12.81	16.60
Bicycle	40	22	11	73
	15.21	9.09	4.55	9.77
Walk	53	9	10	72
	20.15	3.72	4.13	9.64
Total	263	242	242	747
	100.00	100.00	100.00	100.00

Pearson chi2(6) = 93.5435 Pr = 0.000

Three-way tabulations created by cmtab can be displayed more compactly using the option compact:

```
. cmtab parttime, choice(choice) time timelast compact
Tabulations by chosen alternatives (choice = 1)
    parttime is constant within case
```

Alternatives	Part-time job and Time variable: Full-time 1	Full-time 2	Full-time 3	Part-time 1	Part-time 2	Part-time 3
Car	119	186	198	115	173	190
Public	53	43	36	55	38	31
Bicycle	34	18	20	40	22	11
Walk	31	11	4	53	9	10

Stored results

cmtab stores the following in r():

Scalars

r(N)	number of observations
r(r)	number of rows
r(c)	number of columns
r(chi2)	Pearson's χ^2
r(p)	p-value for Pearson's χ^2 test
r(chi2_lr)	likelihood-ratio χ^2
r(p_lr)	p-value for likelihood-ratio test

Also see

[CM] **cmchoiceset** — Tabulate choice sets

[CM] **cmsample** — Display reasons for sample exclusion

[CM] **cmset** — Declare data to be choice model data

[CM] **cmsummarize** — Summarize variables by chosen alternatives

Title

cmxtmixlogit — Panel-data mixed logit choice model

Description | Quick start | Menu | Syntax
Options | Remarks and examples | Stored results | Methods and formulas
References | Also see

Description

`cmxtmixlogit` fits a mixed logit model (MLM) to choice data in which decision makers make repeated choices, typically at different time periods. Data with repeated cases from the same units are also referred to as panel data. Panel-data mixed logit models use random coefficients to model the correlation of choices across alternatives. The random coefficients are on variables that vary across alternatives (and may vary over individuals and choice sets as well), known as alternative-specific variables.

The correlation of choices across alternatives relaxes the independence of irrelevant alternatives (IIA) assumption required by the conventional multinomial logit model, fit by `mlogit`, and the choice-model conditional logit model, fit by `cmclogit`.

In the context of panel-data applications, the MLM models the probability of selecting each alternative for each time period rather than modeling a single probability for selecting each alternative, as in the case of cross-sectional data. Mixed logit models for cross-sectional data are fit by `cmmixlogit`.

Quick start

Mixed logit regression for panel data of `y` on `x1`, where the coefficients on `x1` are assumed random normal, using `cmset` data

```
cmxtmixlogit y, random(x1)
```

Same as above, and include levels of case-specific covariate `a`

```
cmxtmixlogit y, random(x1) casevars(i.a)
```

Same as above, and include lag of alternative-specific variable `x2` with triangular distributed random coefficients

```
cmxtmixlogit y, random(x1) random(L.x2, triangle) casevars(i.a)
```

Same as above, and include alternative-specific variable `x3` with fixed coefficient

```
cmxtmixlogit y x3, random(x1) random(L.x2, triangle) casevars(i.a)
```

Menu

Statistics > Choice models > Panel-data mixed logit model

Syntax

```
cmxtmixlogit depvar [indepvars] [if] [in] [weight] [, options]
```

depvar equal to 1 identifies the chosen alternatives, whereas a 0 indicates the alternatives that were not chosen. There can be only one chosen alternative for each case.

indepvars specifies the alternative-specific covariates that have fixed coefficients.

options	Description
Model	
`casevars(`*varlist*`)`	choice set-specific variables
`random(`*varlist* [`,` *distribution*]`)`	specify variables that are to have random coefficients and the coefficients' distribution
`corrmetric(`*metric*`)`	correlation metric for correlated random coefficients
`basealternative(`*# | lbl | str*`)`	alternative used for normalizing location
`noconstant`	suppress the alternative-specific constant terms
`altwise`	use alternativewise deletion instead of casewise deletion
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `robust`, `cluster` *clustvar*, `opg`, `bootstrap`, or `jackknife`
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`or`	report odds ratios and relative-risk ratios
`nocnsreport`	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
Integration	
`intmethod(`*seqspec*`)`	specify point set for Monte Carlo integration
`intpoints(#)`	specify number of points in each sequence
`intburn(#)`	specify starting index in the Hammersley or Halton sequence
`intseed(#)`	specify random-number seed for pseudo–random sequence
`favor(speed | space)`	favor speed or space when generating integration points
Maximization	
maximize_options	control the maximization process; seldom used
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

distribution	Description
normal	Gaussian-distributed random coefficients; the default
correlated	correlated Gaussian-distributed random coefficients
lnormal	lognormal distributed random coefficients
tnormal	truncated normal distributed random coefficients
uniform	uniform distributed random coefficients
triangle	triangular distributed random coefficients

metric	Description
correlation	standard deviation and correlation; the default
covariance	variance and covariance
cholesky	Cholesky factor

seqspec is

seqtype[, antithetics | mantithetics]

seqtype	Description
hammersley	Hammersley point set; the default
halton	Halton point set
random	uniform pseudo–random point set

You must cmset your data before using cmxtmixlogit; see [CM] **cmset**.

indepvars and *varlist* may contain factor variables and time-series operators; see [U] **11.4.3 Factor variables** and [U] **11.4.4 Time-series varlists**.

bootstrap, by, collect, jackknife, statsby, and svy are allowed; see [U] **11.1.10 Prefix commands**.

Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.

vce() 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

casevars(*varlist*) specifies the case-specific variables that are constant for each case, that is, choice set. If there are a maximum of A alternatives, there will be $A-1$ sets of coefficients associated with casevars().

random(*varlist* [, *distribution*]) specifies the alternative-specific variables that are to have random coefficients and, optionally, the assumed distribution of the random coefficients. The default distribution is normal, meaning Gaussian-distributed random coefficients. *distribution* may also be correlated, lnormal, tnormal, uniform, or triangle. random() may be specified more than once to specify different sets of variables that correspond to different coefficient distributions.

`corrmetric(`*metric*`)` specifies the estimation metric for correlated random coefficients. `corrmetric(correlation)`, the default, estimates the standard deviations and correlations of the random coefficients. `corrmetric(covariance)` estimates variances and covariances, and `corrmetric(cholesky)` estimates Cholesky factors. `corrmetric()` is allowed only when `random(`*varlist*`, correlated)` is specified.

`basealternative(`*#* | *lbl* | *str*`)` sets the alternative that normalizes the level of utility. The base alternative may be specified as a number when the alternatives variable is numeric, as a label when it is numeric and has a value label, or as a string when it is a string variable. The default is the alternative with the highest frequency of being chosen. This option is ignored if neither alternative-specific constants nor case-specific variables are specified.

`noconstant` suppresses the $A-1$ alternative-specific constant terms.

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

`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***.

Specifying `vce(robust)` is equivalent to specifying `vce(cluster` *caseid*`)`.

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `basealternative()`.

Reporting

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

`or` reports the estimated coefficients transformed to odds ratios for alternative-specific variables and relative-risk ratios for case-specific variables. That is, e^b rather than b is reported. Standard errors and confidence intervals are transformed accordingly. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or when replaying previously estimated results.

`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**.

Integration

`intmethod(`*seqtype*`[ , antithetics | mantithetics ])` specifies the method of generating the point sets used in the Monte Carlo integration. `intmethod(hammersley)`, the default, uses the Hammersley sequence; `intmethod(halton)` uses the Halton sequence; and `intmethod(random)` uses a sequence of uniform random numbers.

`antithetics` and `mantithetics` specify that a unidimensional antithetic sequence or a multidimensional antithetic sequence, respectively, be generated instead of the standard implementation of the requested *seqtype*. These methods improve the accuracy of the Monte Carlo integration at the cost of additional computation time; see *Methods and formulas*.

`intpoints(#)` specifies the number of points to use in the Monte Carlo integration. The default number of points is a function of model complexity and integration method. If `intmethod(hammersley)` or `intmethod(halton)` is used, the default is $500 + \text{floor}[2.5\sqrt{N_c\{\ln(r+5)+v\}}]$, where N_c is the number of panels, r is the number of random coefficients in the model, and v is the number of variance parameters. If `intmethod(hammersley, mantithetics)` or `intmethod(halton, mantithetics)` is used, the number of integration points is $250+\text{floor}[0.5\sqrt{N_c\{\ln(r+5)+v\}}]$. If `intmethod(random)` is used, the number of points is twice the number of points used by `intmethod(hammersley)` and `intmethod(halton)`. Larger values of `intpoints()` provide better approximations of the log likelihood at the cost of additional computation time.

`intburn(#)` specifies where in the Hammersley or Halton sequence to start, which helps reduce the correlation between the sequences of each dimension. The default is to discard the first n initial elements from each sequence, where n is the largest prime used to generate the sequences. This option may not be specified with `intmethod(random)`.

`intseed(#)` specifies the seed to use for generating uniform pseudo–random sequences. This option may be specified only with `intmethod(random)`. # must be an integer between 0 and $2^{31}-1$. The default is to use the current seed value from Stata's uniform random-number generator; see [R] **set seed**.

`favor(speed|space)` instructs `cmxtmixlogit` to favor either speed or space when generating the integration points. `favor(speed)` is the default. When favoring speed, the integration points are generated once and stored in memory, thus increasing the speed of evaluating the likelihood. This speed increase can be seen when there are many cases or when the user specifies many integration points in `intpoints(#)`. When favoring space, the integration points are generated repeatedly with each likelihood evaluation.

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**.

Setting the optimization type to `technique(bhhh)` resets the default *vcetype* to `vce(opg)`.

The following options are available with `cmxtmixlogit` but are not shown in the dialog box:

`collinear`, `coeflegend`; see [R] **Estimation options**.

Remarks and examples

`cmxtmixlogit` fits a mixed logit model to panel data. The mixed logit model is most frequently used to model the probability that an individual chooses one of several unordered alternatives. It is also known as the mixed multinomial logit model (McFadden and Train 2000), the random-parameters logit model (Cameron and Trivedi 2005), the logit kernel model (Ben-Akiva, Bolduc, and Walker 2001), or the hybrid logit model (Ben-Akiva et al. 1997). For an introduction to mixed logit models, see [CM] **cmmixlogit**.

The panel-data mixed logit model is used in the case where a decision maker makes repeated choices over time, and it models the probability of selecting each alternative at each time point. See chapter 12 of Rabe-Hesketh and Skrondal (2022) for applications using Stata.

▷ Example 1: Panel-data mixed logit model with alternative- and case-specific covariates

We have created fictitious data for an individual's choice of transportation mode over multiple occasions. There are also data on his or her cost of transportation and travel time for each mode, as well as information about his or her income and age. We wish to examine how these factors affect the choice of transportation alternative.

`transport.dta` records information about the choices of primary transportation made by 500 individuals to their workplace in a metropolitan area. Each individual chose between four modes of transportation at three points in time. The alternatives are recorded in the variable `alt`.

The binary variable `choice` records the chosen alternative; `choice` is 1 for the chosen alternative and 0 otherwise. For each individual, we have one choice set per time, where `t` is the time variable. In the following, we refer to a choice set as a case. Here are the data for the first individual in the dataset:

```
. use https://www.stata-press.com/data/r18/transport
(Transportation choice data)
. list in 1/12, sepby(t)
```

	id	t	alt	choice	trcost	trtime	age	income	parttime
1.	1	1	Car	1	4.14	0.13	3.0	3	Full-time
2.	1	1	Public	0	4.74	0.42	3.0	3	Full-time
3.	1	1	Bicycle	0	2.76	0.36	3.0	3	Full-time
4.	1	1	Walk	0	0.92	0.13	3.0	3	Full-time
5.	1	2	Car	1	8.00	0.14	3.2	5	Full-time
6.	1	2	Public	0	3.14	0.12	3.2	5	Full-time
7.	1	2	Bicycle	0	2.56	0.18	3.2	5	Full-time
8.	1	2	Walk	0	0.64	0.39	3.2	5	Full-time
9.	1	3	Car	1	1.76	0.18	3.4	5	Part-time
10.	1	3	Public	0	2.25	0.50	3.4	5	Part-time
11.	1	3	Bicycle	0	0.92	1.05	3.4	5	Part-time
12.	1	3	Walk	0	0.58	0.59	3.4	5	Part-time

This person chose to use the car at all three points in time.

Travel time (`trtime`, measured in hours) and cost of travel (`trcost`, measured in $) are alternative-specific variables that vary over alternatives, cases, and individuals. Variables `age` (measured in decades), `income` (annual income measured in $10,000), and `parttime` (variable indicating a part-time job) are case-specific variables that vary both over cases and individuals but are constant across alternatives for a given individual and time.

Before we can fit the model, we will need to `cmset` the data. To specify that `id` identifies the individuals, that `t` identifies time, and that `alt` identifies the alternatives available to each individual, we type the following:

```
. cmset id t alt
note: case identifier _caseid generated from id and t.
note: panel by alternatives identifier _panelaltid generated from id and alt.

                     Panel data: Panels id and time t
               Case ID variable: _caseid
          Alternatives variable: alt
Panel by alternatives variable: _panelaltid (strongly balanced)
                  Time variable: t, 1 to 3
                          Delta: 1 unit

Note: Data have been xtset.
```

The output informs us that the panels are strongly balanced, which is to say that each individual made a choice in each of the same periods. The output also says that two new variables were created, _caseid and _panelaltid. See example 2 in [CM] **cmset** for an explanation of the purpose of these new variables. You need not be concerned about them, just leave them in your dataset.

We can now use cmxtmixlogit to fit our model. We wish to estimate the effect of travel cost (trcost), travel time (trtime), income, and age on the choice of transportation mode. We assume that all individuals have the same preferences with respect to travel cost, but that preferences with respect to travel time are heterogeneous, and we model these heterogeneous preferences with a random coefficient for trtime.

We specify that choice contains the chosen alternatives and that trcost is included in the model with a fixed coefficient. Specifying random(trtime) includes trtime with a normally distributed random coefficient—the default distribution of random coefficients. Specifying casevars(age income) includes the case-specific variables age and income in the model with fixed coefficients.

```
. cmxtmixlogit choice trcost, random(trtime) casevars(age income)
Fitting fixed parameter model:
Fitting full model:
Iteration 0:  Log simulated-likelihood =  -1025.707  (not concave)
Iteration 1:  Log simulated-likelihood = -1014.2513
Iteration 2:  Log simulated-likelihood = -1006.2212
Iteration 3:  Log simulated-likelihood = -1005.9904
Iteration 4:  Log simulated-likelihood = -1005.9899
Iteration 5:  Log simulated-likelihood = -1005.9899
Mixed logit choice model                    Number of obs        =      6,000
                                            Number of cases      =      1,500
Panel variable: id                          Number of panels     =        500
Time variable: t                            Cases per panel: min =          3
                                                             avg =        3.0
                                                             max =          3
Alternatives variable: alt                  Alts per case:   min =          4
                                                             avg =        4.0
                                                             max =          4
Integration sequence:      Hammersley
Integration points:              594        Wald chi2(8)     =     432.68
Log simulated-likelihood = -1005.9899       Prob > chi2      =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
alt						
trcost	-.8388216	.0438587	-19.13	0.000	-.9247829	-.7528602
trtime	-1.508756	.2641554	-5.71	0.000	-2.026492	-.9910212
/Normal						
sd(trtime)	1.945596	.2594145			1.498161	2.526661
Car	(base alternative)					
Public						
age	.1538915	.0672638	2.29	0.022	.0220569	.2857261
income	-.3815444	.0347459	-10.98	0.000	-.4496451	-.3134437
_cons	-.5756547	.3515763	-1.64	0.102	-1.264732	.1134222
Bicycle						
age	.20638	.0847655	2.43	0.015	.0402426	.3725174
income	-.5225054	.0463235	-11.28	0.000	-.6132978	-.4317131
_cons	-1.137393	.4461318	-2.55	0.011	-2.011795	-.2629909
Walk						
age	.3097417	.1069941	2.89	0.004	.1000372	.5194463
income	-.9016697	.0686042	-13.14	0.000	-1.036132	-.7672078
_cons	-.4183279	.5607111	-0.75	0.456	-1.517302	.6806458

The output header shows that our model was fit using data on 500 panels. Each panel is an individual in our example. Each panel/person made a choice in each of 3 time periods, so there are 1,500 cases. Because every case has 4 alternatives, there are 6,000 observations in total.

The estimate of the fixed coefficient on `trcost` is −0.84. The estimated mean of the random coefficients on `trtime` is −1.51. The estimated standard deviation of these random coefficients is 1.95, and there is significant heterogeneity in these coefficients over the individuals in the population.

Each case-specific variable has different coefficients for each alternative. These coefficients are interpreted similarly to conventional multinomial logit. For example, that the estimated coefficients on `age` for the `Public` and `Bicycle` alternatives are positive when `Car` is the base alternative implies that an increase in age makes it more likely that someone will choose public transportation or a bicycle over a car. Similarly, an increase in income makes it less likely that someone will choose public transportation or a bicycle over a car.

After you have fit what you consider your final model, you should run the same model again, but this time setting `intpoints(#)`, the number of integration points in the simulated likelihood to a larger number. In this example, we see from the header that the default number of points was 594. We would run our model again using, say, 1000 points and see by how much the coefficient or covariance parameter estimates change. If changes are small compared with standard errors, we can have confidence in the numerical soundness of the simulation used to compute the likelihood. See *Setting the number of integration points* in [CM] **Intro 5** for more information.

◁

▷ Example 2: Expected choice probabilities

The choice probabilities are a nonlinear function of the estimated parameters, which complicates parameter interpretation. We can, however, use `margins` to estimate the effects of variables.

We wish to estimate expected choice probabilities for specified income levels. We estimate average probabilities of each alternative when every individual has an income of $30,000 and when every individual has an income of $80,000. The observed values are used for all other variables.

```
. margins, at(income=(3 8))
Predictive margins                                      Number of obs = 6,000
Model VCE: OIM
Expression: Pr(alt), predict()
1._at: income = 3
2._at: income = 8
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome#_at						
Car#1	.3331611	.0196734	16.93	0.000	.294602	.3717203
Car#2	.7009364	.0109829	63.82	0.000	.6794103	.7224626
Public#1	.2210964	.0184285	12.00	0.000	.1849772	.2572156
Public#2	.1801593	.0091811	19.62	0.000	.1621646	.1981539
Bicycle#1	.1676081	.0181511	9.23	0.000	.1320325	.2031837
Bicycle#2	.0862692	.0080813	10.68	0.000	.0704302	.1021081
Walk#1	.2781343	.0243791	11.41	0.000	.2303521	.3259166
Walk#2	.0326352	.0058942	5.54	0.000	.0210827	.0441877

If every individual had an income of $30,000, 33% of them would choose a car. On the other hand, if everybody had an income of $80,000, 70% would choose a car. The results for `Walk` are reversed. If every individual had an income of $30,000, 28% of them would choose to walk. If every individual had an income of $80,000, only 3% of them would choose to walk.

In the above example, we averaged the choice probabilities over all time periods. This works well for time-invariant effects, but we could be interested in having our predictions evaluated for certain points in time. In the following example, we estimate the same averaged predictions as in the previous example but only for the first time period. We do this by specifying `subpop(if t==1)`.

```
. margins, at(income=(3 8)) subpop(if t==1)
Predictive margins                                  Number of obs   = 6,000
Model VCE: OIM                                      Subpop. no. obs = 2,000
Expression: Pr(alt), predict()
1._at: income = 3
2._at: income = 8
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome#_at						
Car#1	.3280809	.0193673	16.94	0.000	.2901217	.3660401
Car#2	.6910464	.0110758	62.39	0.000	.6693381	.7127546
Public#1	.2292844	.0188376	12.17	0.000	.1923633	.2662054
Public#2	.1892283	.0094331	20.06	0.000	.1707398	.2077168
Bicycle#1	.1723129	.0184627	9.33	0.000	.1361265	.2084992
Bicycle#2	.0880065	.0083066	10.59	0.000	.0717258	.1042872
Walk#1	.2703219	.0238268	11.35	0.000	.2236222	.3170215
Walk#2	.0317188	.005727	5.54	0.000	.0204942	.0429434

Here the results are relatively similar to the ones we obtained by averaging over all time periods. ◁

▷ Example 3: Contrasts of marginal predictions

The previous example estimated averages for two different scenarios at one point in time. Based on this model, and assuming we have a random or otherwise representative sample, we can interpret the difference in these average probabilities as the effect of having an income of $80,000 instead of an income of $30,000. We can estimate this difference using the `contrast()` option. We request this for each time point separately by including the `over(t)` option.

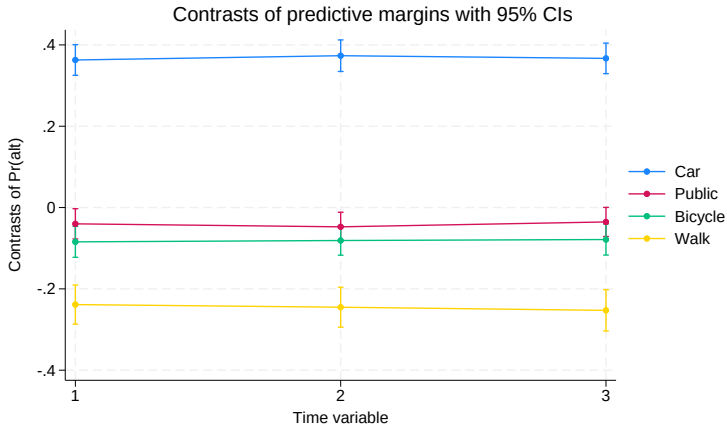
```
. margins, at(income=(3 8)) contrast(at(r) nowald) over(t)
Contrasts of predictive margins                          Number of obs = 6,000
Model VCE: OIM

Expression: Pr(alt), predict()
Over:       t

1._at: 1.t
           income = 3
1._at: 2.t
           income = 3
1._at: 3.t
           income = 3
2._at: 1.t
           income = 8
2._at: 2.t
           income = 8
2._at: 3.t
           income = 8
```

	Contrast	Delta-method std. err.	[95% conf.	interval]
_at@_outcome#t				
(2 vs 1) Car#1	.3629654	.0192268	.3252817	.4006492
(2 vs 1) Car#2	.3734991	.0197989	.334694	.4123042
(2 vs 1) Car#3	.3668613	.0191453	.3293372	.4043854
(2 vs 1) Public#1	-.040056	.0189899	-.0772756	-.0028365
(2 vs 1) Public#2	-.0473747	.0183016	-.0832451	-.0115042
(2 vs 1) Public#3	-.0353807	.0182474	-.071145	.0003835
(2 vs 1) Bicycle#1	-.0843064	.0194643	-.1224557	-.046157
(2 vs 1) Bicycle#2	-.0810573	.018425	-.1171697	-.0449449
(2 vs 1) Bicycle#3	-.0786532	.0194817	-.1168366	-.0404699
(2 vs 1) Walk#1	-.2386031	.0245353	-.2866913	-.1905148
(2 vs 1) Walk#2	-.2450671	.0250718	-.2942069	-.1959274
(2 vs 1) Walk#3	-.2528274	.0259127	-.3036154	-.2020394

For example, at the first time point, we estimate that the effect of making \$80,000 instead of \$30,000 is an increase of 0.36 in the probability of choosing to travel by car.

We can also plot our results by using `marginsplot`.

```
. marginsplot
Variables that uniquely identify margins: t _outcome
```



Here we have the differences in probabilities on the y axis and time on the x axis. Differences over time are shown for each of the alternatives. We see that there is no substantial change over time in the effect of making \$80,000 instead of \$30,000.

◁

▷ Example 4: Marginal predictions with alternative-specific variables

We now estimate the effect of an alternative-specific covariate. Changing the value on one alternative-specific covariate affects the choice probabilities of all the alternatives. For example, an increase in the cost of car travel changes the probabilities of choosing `Car`, `Public`, `Bicycle`, or `Walk`. We estimate expected probabilities of choosing each outcome under two scenarios. The first scenario is the as-is scenario: all variables are as observed. In the second scenario, the cost of car travel increases by 25%. We do this by specifying the option `alternative(Car)` to tell `margins` that we wish to apply our `at()` scenarios for the alternative-specific variable `trcost` to the `Car` alternative.

```
. margins, alternative(Car) at(trcost = generate(trcost))
> at(trcost = generate(1.25*trcost)) subpop(if t==1)
Predictive margins                                  Number of obs   = 6,000
Model VCE: OIM                                      Subpop. no. obs = 2,000

Expression:  Pr(alt), predict()
Alternative: Car

1._at: trcost =      trcost

2._at: trcost = 1.25*trcost
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome#_at						
Car#1	.5439062	.0113994	47.71	0.000	.5215638	.5662486
Car#2	.4405694	.0101017	43.61	0.000	.4207704	.4603683
Public#1	.2010082	.0104382	19.26	0.000	.1805497	.2214668
Public#2	.2548516	.0117988	21.60	0.000	.2317264	.2779769
Bicycle#1	.1255662	.0095539	13.14	0.000	.1068409	.1442914
Bicycle#2	.1566796	.0110237	14.21	0.000	.1350736	.1782856
Walk#1	.1295194	.0101536	12.76	0.000	.1096187	.1494201
Walk#2	.1478994	.0110109	13.43	0.000	.1263185	.1694803

The average probability of choosing car travel falls from 0.54 to 0.44 when we move from the current situation to one in which the cost of car travel increases by 25%. About half of those that would stop selecting car travel would move to public transportation; the expected probability of selecting public transportation increases from 0.20 to 0.25.

◁

▷ Example 5: Average marginal effects

A marginal effect is the change in the probability of choosing a certain alternative as a function of an infinitesimal change in a covariate at a given point. In other words, it is the slope of the choice probability with respect to a covariate at a given point of the covariate. The average marginal effect is the average of the marginal effects over all the observations.

As in Example 4, a change to one alternative-specific covariate affects the choice probabilities of all alternatives. In our example, there are four marginal effects for the cost of `Car`, four marginal effects for the cost of `Public`, four marginal effects for the cost of `Bicycle`, and four marginal effects for the cost of `Walk`.

In the context of discrete choice models with alternative-specific variables, we distinguish between direct and indirect marginal effects. The marginal effects of an alternative-specific variable on its own choice probability is called a direct marginal effect. The marginal effects of an alternative-specific variable on the choice probabilities of the other alternatives are called indirect marginal effects.

We now estimate the direct and the indirect marginal effects of travel time for Car.

```
. margins, dydx(trtime) alternative(Car)
Average marginal effects                                  Number of obs = 6,000
Model VCE: OIM
Expression:  Pr(alt), predict()
Alternative: Car
dy/dx wrt:   trtime
```

	dy/dx	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
trtime						
_outcome						
Car	-.1581844	.0269102	-5.88	0.000	-.2109275	-.1054414
Public	.1055447	.0171745	6.15	0.000	.0718834	.139206
Bicycle	.0374872	.0073318	5.11	0.000	.0231171	.0518573
Walk	.0151526	.0043034	3.52	0.000	.006718	.0235871

The estimate of the direct average marginal effect is around -0.16, suggesting a decrease in the probability of choosing a car if car travel time increases. The estimated indirect marginal effects are all positive, with the effect on Public being larger than those on Bicycle and on Walk.

◁

❑ Technical note

Although cmxtmixlogit follows a random-effects approach, the model does not actually contain a separately identifiable variance component beyond the error components of the random coefficients. Rather, the extra variance due to panel random effects is contained in the error components associated with the random coefficients. No fixed-effects estimator is available for the mixed logit model.

❑

Stored results

cmxtmixlogit stores the following in e():

Scalars

e(N)	number of observations
e(N_case)	number of cases
e(N_panel)	number of panels
e(N_ic)	N for Bayesian information criterion (BIC)
e(N_clust)	number of clusters
e(k)	number of parameters
e(k_alt)	number of alternatives
e(k_casevars)	number of case-specific variables
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(df_m)	model degrees of freedom
e(df_c)	degrees of freedom, comparison test
e(ll)	log simulated-likelihood
e(ll_c)	log likelihood, comparison model
e(const)	constant indicator
e(intpoints)	number of raw integration points
e(lsequence)	length of each integration sequence
e(intburn)	starting sequence index

e(chi2)	χ^2
e(chi2_c)	χ^2, comparison test
e(p)	p-value for model test
e(p_c)	p-value for comparison test
e(alt_min)	minimum number of alternatives
e(alt_avg)	average number of alternatives
e(alt_max)	maximum number of alternatives
e(t_avg)	average number of time points per panel
e(t_min)	minimum number of time points per panel
e(t_max)	maximum number of time points per panel
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	cmxtmixlogit
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(caseid)	name of case ID variable
e(timevar)	name of time variable
e(altvar)	name of alternatives variable
e(alteqs)	alternative equation names
e(alt#)	alternative labels
e(base)	base alternative
e(corrmetric)	correlation metric for correlated random coefficients
e(wtype)	weight type
e(wexp)	weight expression
e(marktype)	casewise or altwise, type of markout
e(key_N_ic)	cases, key for N for Bayesian information criterion (BIC)
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(chi2type)	type of χ^2
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(intmethod)	technique used to generate sequences
e(sequence)	type of sequences
e(mc_rngstate)	random-number state used
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(properties)	b V
e(predict)	program used to implement predict
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(altvals)	alternative values
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 the mixed logit model, the utility that individual i receives from alternative a at time t, denoted by U_{iat}, is

$$U_{iat} = \mathbf{x}_{iat}\boldsymbol{\beta}_i + \mathbf{w}_{iat}\boldsymbol{\alpha} + \mathbf{z}_{it}\boldsymbol{\delta}_a + \epsilon_{iat} \qquad a = 1, 2, \ldots, A$$

$\boldsymbol{\beta}_i$ are random coefficients that vary over individuals in the population, and $\mathbf{x}_{iat}$ is a vector of alternative-specific variables. $\boldsymbol{\alpha}$ are fixed coefficients on $\mathbf{w}_{iat}$, a vector of alternative-specific variables. $\boldsymbol{\delta}_a$ are fixed, alternative-specific coefficients on $\mathbf{z}_{it}$, a vector of case-specific variables. ϵ_{iat} is a random term that follows a type I extreme value distribution. cmxtmixlogit estimates the fixed coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}_a$ and the parameters of $f(\boldsymbol{\beta})$, the distribution of the random coefficients.

cmxtmixlogit estimates the parameters of the mixed logit model by maximum simulated likelihood (MSL). The probability that case i chooses alternative a at time t, conditional on the random parameter $\boldsymbol{\beta}_i$, is

$$P_{iat}(\boldsymbol{\beta}) = \frac{e^{\mathbf{x}_{iat}\boldsymbol{\beta}_i + \mathbf{w}_{iat}\boldsymbol{\alpha} + \mathbf{z}_{it}\boldsymbol{\delta}_a}}{\sum_{a=1}^{A} e^{\mathbf{x}_{iat}\boldsymbol{\beta}_i + \mathbf{w}_{iat}\boldsymbol{\alpha} + \mathbf{z}_{it}\boldsymbol{\delta}_a}}$$

We get the unconditional choice probability, P_{iat}, by integrating over the mixing distribution $f(\boldsymbol{\beta})$:

$$P_{iat} = \int P_{iat}(\boldsymbol{\beta}) f(\boldsymbol{\beta}) d\boldsymbol{\beta} \tag{1}$$

This integral of dimension d, where d equals the number of random parameters, is approximated by simulation. The simulated likelihood for the ith case is

$$L_i = \prod_{t=1}^{T} \sum_{a=1}^{A} d_{iat} \widehat{P}_{iat}$$

where d_{iat} is an indicator that takes on the value 1 for the chosen alternative at time t and 0 otherwise. The overall log simulated-likelihood is then $\sum_{i=1}^{N} \ln(L_i)$. $\widehat{P}_{iat}$ are the simulated probabilities

$$\widehat{P}_{iat} = \frac{1}{M} \sum_{m=1}^{M} P_{iat}(\boldsymbol{\beta}^m) \tag{2}$$

where $\boldsymbol{\beta}^m$ are the random parameters drawn from $f(\boldsymbol{\beta})$ and M is the number of random draws. Equation (2) is the computation used to approximate the probabilities in (1).

Computation of $\widehat{P}_{iat}$ is based on integration sequences where each point of the sequence is a draw from density $f(\boldsymbol{\beta})$. The underlying uniform sequences are either pseudo–random draws from the uniform density or deterministic sequences such as a Halton sequence. Using deterministic sequences leads to better coverage of the probability space and lower variance of the simulator and thus having a smaller approximation error than pseudo–random sequences, given the same number of draws. cmxtmixlogit supports pseudo–random, Halton, and Hammersley sequences; see Drukker and Gates (2006) for details.

Using a higher M in (2) will produce a better approximation to the probabilities in (1), at the cost of increased computation time. M is a function of the number of raw integration points q, which may be specified using the `intpoints()` option. In the default method, $M = q$ is the number of draws used in the approximation in (2). In addition to the default method, `cmxtmixlogit` supports methods in which the draws are symmetric around a midpoint, known as unidimensional and multidimensional antithetic draws. These antithetic methods produce a better approximation to the probabilities in (1), at the cost of additional computation time; see Train (2009, sec. 9.3.1). For unidimensional antithetics, $M = 2q$ draws are used. For multidimensional antithetics on a problem with d random coefficients, $M = 2^d q$ draws are used.

Random coefficients with mean μ and scale parameter σ are simulated as follows:

$$\begin{aligned}
\beta^m_{\text{normal}} &= \mu + \sigma\eta_i & \eta_i &\sim N(0,1)\\
\beta^m_{\text{lognormal}} &= \exp(\mu + \sigma\eta_i) & \eta_i &\sim N(0,1)\\
\beta^m_{\text{truncated_normal}} &= \mu + \sigma\eta_i & \eta_i &\sim \text{TN}(0,1,-1.96,1.96)\\
\beta^m_{\text{uniform}} &= \mu + \sigma\eta_i & \eta_i &\sim U(-1,1)\\
\beta^m_{\text{triangular}} &= \mu + \sigma\eta_i & \eta_i &\sim \triangle(-1,1)
\end{aligned}$$

where $N(\mu,\sigma^2)$ is the normal distribution, $\text{TN}(\mu,\sigma^2,a,b)$ is the truncated normal distribution with lower truncation point a and upper truncation point b, $U(a,b)$ is uniform over $[\,a,b\,]$, and $\triangle(a,b)$ is the triangular distribution over $[\,a,b\,]$.

Correlated random parameters drawn from the multivariate normal distribution are generated as $\beta^m_{\text{MVN}} = \mathbf{M} + \mathbf{L}\eta_i$, where $\mathbf{M}$ is a vector of means, $\eta_i \sim N(\mathbf{0},\mathbf{I})$, and $\mathbf{L}$ is the Cholesky factor such that the variance–covariance matrix $\mathbf{V} = \mathbf{L}\mathbf{L}'$.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster` *clustvar*`)`. See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*. Specifying `vce(robust)` is equivalent to specifying `vce(cluster` *panelvar*`)`, where *panelvar* is the variable that identifies the panels.

References

Ben-Akiva, M., D. Bolduc, and J. Walker. 2001. Specification, identification, and estimation of the logit kernel (or continuous mixed logit) model. Manuscript, University of California, Berkeley. http://eml.berkeley.edu/reprints/misc/multinomial2.pdf.

Ben-Akiva, M., D. L. McFadden, M. Abe, U. Böckenholt, D. Bolduc, D. Gopinath, T. Morikawa, V. Ramaswamy, V. Rao, D. Revelt, and D. Steinberg. 1997. Modeling methods for discrete choice analysis. *Marketing Letters* 8: 273–286. https://doi.org/10.1023/A:1007956429024.

Brownstone, D., and K. E. Train. 1998. Forecasting new product penetration with flexible substitution patterns. *Journal of Econometrics* 89: 109–129. https://doi.org/10.1016/S0304-4076(98)00057-8.

Cameron, A. C., and P. K. Trivedi. 2005. *Microeconometrics: Methods and Applications*. New York: Cambridge University Press.

Drukker, D. M., and R. Gates. 2006. Generating Halton sequences using Mata. *Stata Journal* 6: 214–228.

McFadden, D. L., and K. E. Train. 2000. Mixed MNL models for discrete response. *Journal of Applied Econometrics* 15: 447–470. https://doi.org/10.1002/1099-1255(200009/10)15:5⟨447::AID-JAE570⟩3.0.CO;2-1.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Train, K. E. 2009. *Discrete Choice Methods with Simulation*. 2nd ed. New York: Cambridge University Press.

Also see

[CM] **cmxtmixlogit postestimation** — Postestimation tools for cmxtmixlogit

[CM] **cmmixlogit** — Mixed logit choice model

[CM] **cmset** — Declare data to be choice model data

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

[SVY] **svy estimation** — Estimation commands for survey data

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

Title

cmxtmixlogit postestimation — Postestimation tools for cmxtmixlogit

Postestimation commands predict margins Methods and formulas
Also see

Postestimation commands

The following standard postestimation commands are available after `cmxtmixogit`:

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`	adjusted predictions, predictive margins, and 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`	probabilities, 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 probabilities or linear predictions.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [`,` *statistic*]

`predict` [*type*] *stub*`*` [*if*] [*in*]`,` `scores`

statistic	Description
Main	
`pr`	probability alternative is chosen; the default
`xb`	linear prediction

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`predict` omits missing values casewise if `cmxtmixlogit` used casewise deletion (the default); if `cmxtmixlogit` used alternativewise deletion (option `altwise`), `predict` uses alternativewise deletion.

Options for predict

Main

`pr`, the default, calculates the probability of choosing each alternative.

`xb` calculates the linear prediction.

`scores` calculates the scores for each coefficient in `e(b)`. This option requires a new variable list of length equal to the number of columns in `e(b)`. Otherwise, use the *stub*`*` syntax to have `predict` generate enumerated variables with prefix *stub*.

margins

Description for margins

`margins` estimates margins of response for probabilities and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`pr`	probability alternative is chosen; the default
`xb`	linear prediction
`scores`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For more details, see [CM] **margins**.

Methods and formulas

The predicted probability of case i choosing alternative a at time t is

$$\widehat{P}_{iat} = \frac{1}{M}\sum_{m=1}^{M} P_{iat}(\boldsymbol{\beta}^m)$$

where M is the number of random draws and $P_{iat}(\boldsymbol{\beta}^m)$ are the logistic probabilities

$$P_{iat}(\boldsymbol{\beta}^m) = \frac{e^{\mathbf{x}_{iat}\boldsymbol{\beta}_i^m + \mathbf{w}_{iat}\boldsymbol{\alpha} + \mathbf{z}_{it}\boldsymbol{\delta}_a}}{\sum_{a=1}^{A} e^{\mathbf{x}_{iat}\boldsymbol{\beta}_i^m + \mathbf{w}_{iat}\boldsymbol{\alpha} + \mathbf{z}_{it}\boldsymbol{\delta}_a}}$$

evaluated at the simulated coefficients $\boldsymbol{\beta}^m$. The linear predictions are

$$\frac{1}{M}\sum_{m=1}^{M} \mathbf{x}_{iat}\boldsymbol{\beta}_i^m + \mathbf{w}_{iat}\boldsymbol{\alpha} + \mathbf{z}_{it}\boldsymbol{\delta}_a$$

See *Methods and formulas* in [CM] **cmxtmixlogit** for details.

Also see

[CM] **cmxtmixlogit** — Panel-data mixed logit choice model

[CM] **margins** — Adjusted predictions, predictive margins, and marginal effects

[U] 20 Estimation and postestimation commands

Title

margins — Adjusted predictions, predictive margins, and marginal effects

Description Quick start Menu Syntax
Options Remarks and examples Stored results Also see

Description

`margins` calculates statistics based on predictions of a previously fit model. These statistics can be calculated averaging over all covariates, or at fixed values of some covariates and averaged over the remaining covariates. After you fit a choice model, `margins` provides estimates such as marginal predicted choice probabilities, adjusted predictions, and marginal effects that allow you to easily interpret the results of a choice model.

Many possible margins can be calculated for choice models. Therefore, `margins` has special choice model options to select which outcomes are estimated or to select which alternatives are fixed or averaged within. These options are available after `cmclogit`, `cmmixlogit`, `cmxtmixlogit`, `cmmprobit`, and `cmroprobit`.

`margins` with the `contrast` option or with contrast operators performs contrasts (comparisons) of margins. After you fit a choice model, there are also special options to select contrasts for outcomes or for alternatives.

This entry focuses on the use of the special choice model options with `margins`. For the full capabilities of `margins`, see [R] **margins**.

Quick start

Outcome probabilities, the average predicted probability of selecting each alternative

```
margins
```

Outcome probabilities for each level of factor-variable `a`

```
margins a
```

Same as above, but show results only for the probability of selecting the outcome labeled `Alt1`

```
margins a, outcome(Alt1)
```

Outcome probabilities setting alternative-specific variable `x1` to $50, 75, \ldots, 500$ for each alternative, one at a time

```
margins, at(x1=(50(25)500))
```

Same as above, but set only the observations of `x1` corresponding to the alternative `Alt1` to the specified values

```
margins, at(x1=(50(25)500)) alternative(Alt1)
```

Outcome probabilities when `x1` is set to the specified values simultaneously across each alternative

```
margins, at(x1=(50(25)500)) alternative(simultaneous)
```

Average marginal effect of `x1` on the predicted probabilities of each outcome

```
margins, dydx(x1)
```

Same as above, but show results only for the outcome `Alt1` for a change in the alternative-specific variable `x1` at the observations corresponding to the alternative `Alt2`, and repeat for alternative `Alt3`

```
margins, dydx(x1) outcome(Alt1) alternative(Alt2 Alt3)
```

Contrasts

For each outcome, test equality of outcome probabilities across levels of `a`

```
margins a, contrast
```

Set alternative-specific variable `x1` to 4 at each of the alternatives identified by `altvar`, and test for differences from the previous alternative

```
margins, at(x1=4) contrast(alternativecontrast(ar.altvar))
```

Menu

Statistics > Postestimation

Syntax

```
margins [marginlist] [, options]
```

marginlist is a list of factor variables or interactions that appear in the current estimation results.

For the full syntax, see [R] **margins**.

options	Description
`outcome(`*outcomes*`[, altsubpop])`	estimate margins for specified outcomes
`alternative(`*alts*`)`	estimate margins for specified alternatives for alternative-specific covariates
`alternative(simultaneous)`	estimate margins changing all alternatives simultaneously for alternative-specific covariates
`contrast`	joint tests of differences across levels of the elements of *marginlist*
`contrast(`*contrast_options*`)`	contrast the margins between the outcomes or alternatives as specified by *contrast_options*
`noesample`	do not restrict `margins` to the estimation sample
other_margins_options	see [R] **margins** for more options

contrast_options	Description
`outcomejoint`	joint test of differences across outcomes
`outcomecontrast(`*op*`[._outcome])`	apply the *op*`.` contrast operator to the outcomes
`alternativejoint`	joint test of differences across alternatives for alternative-specific covariates
`alternativecontrast(`*op*`[.altvar])`	apply the *op*`.` contrast operator to the levels of the alternatives for alternative-specific covariates

Options

Main

`outcome(`*outcomes* [`, altsubpop`]`)` specifies that margins be estimated for the specified outcomes only. The default is to estimate margins for all outcomes.

outcomes is a list of one or more outcomes, which are the values of the alternatives variable; see [CM] **cmset**. *outcomes* can be specified by

`#1`, `#2`, ..., where `#1` means the first level of the alternatives variable, `#2` means the second level, etc.;

numeric values of the alternatives variable if it is a numeric variable;

value labels of the alternatives variable, enclosed in quotes if there are spaces in the value labels;

string values of the alternatives variable if it is a string variable, enclosed in quotes if there are spaces in the values; or

`_all` or `*` for all levels of the alternatives variable.

The suboption `altsubpop` applies only to samples with unbalanced choice sets. For balanced samples, the default is the same as specifying `altsubpop`. This option is used in conjunction with alternative-specific covariates and unbalanced choice sets to specify that calculations done for each alternative be restricted to the subpopulation of cases with that alternative in their choice set. The default treats the sample as if it were balanced with alternatives not in a choice set considered as alternatives with zero probability of being chosen. `altsubpop` is appropriate for unbalanced experimental designs in which decision makers were presented with different choice sets.

`alternative(`*alts*`)` applies only when one or more alternative-specific covariates are specified in an element of *marginlist*, in the `at()` option, or in one of the marginal effects options (`dydx()`, etc.). This option specifies that margins be estimated for the specified alternatives only. The default is to estimate margins for all alternatives. *alts* are specified in the same manner as in `outcome(`*outcomes*`)`.

`alternative(simultaneous)`, as with `alternative(`*alts*`)`, applies only when there are alternative-specific covariates in the specification of `margins`. By default, each alternative-specific covariate is changed (for example, set to a specified value) separately for each alternative, giving results for each alternative. This option specifies that each alternative-specific covariate is to be changed across all alternatives simultaneously to produce a single result.

For example, suppose *xvar* is an alternative-specific variable with alternatives *A*, *B*, and *C*, and `margins, at(`*xvar*`=1)` is specified. By default, *xvar* is first set to 1 for alternative *A* and kept at its sample values for *B* and *C*, then similarly for the alternative *B*, and then *C*, producing results for each of the three alternatives. The `alternative(simultaneous)` option sets *xvar* to 1 at each of the alternatives *A*, *B*, and *C* simultaneously, producing a single result for the alternatives as a group.

`contrast` applies only when *marginlist* is specified. If an element of *marginlist* contains only case-specific covariates, this option displays joint tests of differences among predicted probabilities across the levels of the element for each outcome. If the element contains alternative-specific covariates, this option displays joint tests of differences among predicted probabilities across the levels of the element for each outcome and alternative combination. It also displays a joint test of all the differences.

`contrast(outcomejoint)` displays a joint test of differences across all outcomes. It is a test of the null hypothesis: within each alternative, differences among predicted probabilities across levels of an element of *marginlist* are the same for each outcome.

`contrast(outcomecontrast(`*op*`[ ._outcome ]))` applies the contrast operator *op*`.` to the outcomes. See the *op*`.` table in [R] **contrast** for a list of all contrast operators. The optional `._outcome` does nothing, but adding it will produce more readable code, showing what *op*`.` is operating on.

`contrast(alternativejoint)` applies only when there are alternative-specific covariates in the specification of `margins`. This option displays a joint test of differences across all alternatives. It is a test of the null hypothesis: within each outcome, differences among predicted probabilities across levels of an element of *marginlist* are the same for each alternative.

`contrast(alternativecontrast(`*op*`[ .`*altvar*` ]))` applies only when there are alternative-specific covariates in the specification of `margins`. This option applies the contrast operator *op*`.` to the alternatives. *altvar* is the name of the alternatives covariates used with `cmset`. The optional `.`*altvar* does nothing, but adding it will produce more readable code, showing what *op*`.` is operating on.

`noesample` specifies that `margins` not restrict its computations to the estimation sample used by the previous estimation command. If the estimation command used casewise deletion (the default), `margins` with `noesample` also omits missing values casewise. If the estimation command used alternativewise deletion (option `altwise`), alternativewise deletion is also used by `margins`.

other_margins_options; see [R] **margins** for additional options.

Remarks and examples

Remarks are presented under the following headings:

Introduction
Estimating margins for case-specific variables
Estimating margins for alternative-specific variables
The altsubpop suboption for unbalanced choice sets
More on unbalanced choice sets
The outcomecontrast() and alternativecontrast() suboptions
Graphing margins results

Introduction

Before reading this entry, read [CM] **Intro 1**. There you will learn about many of the common questions you can answer using `margins` after choice models and about some of the special options that are specific to `cm` commands. This entry explores even more of the choice model options for `margins` and delves deeper into the types of hypotheses you can test. Here we also provide advice on using the more advanced options, such as those for handling unbalanced choice sets.

The special choice model options for `margins` can be used after

```
cmclogit
cmmixlogit
cmxtmixlogit
cmmprobit
cmroprobit
```

`margins` has the same capabilities when used after any of these commands. All examples of `margins` shown in this entry will work after any of these commands.

The special choice model options described in this entry cannot, however, be used with margins after other choice models. cmrologit does not have explicitly identified alternatives, so you use the standard syntax of margins after this command; see [R] **margins**. nlogit does not allow margins because of the structure of the hierarchical model it fits.

margins produces estimates based on predictions. After cm estimation commands, two types of predictions can be used with margins, predicted probabilities or linear-form predictions. Predicted probabilities are the default and likely the only one you will use.

For choice models, a predicted probability is the probability of a decision maker choosing one of the possible alternatives, and these probabilities sum to one across the alternatives. (Note that for rank-ordered alternatives modeled by cmroprobit, margins bases its results on predictions calculated using the predict option pr1, which gives the probability of ranking an alternative as first. So margins after cmroprobit behaves just as it does after the cm estimators for models in which a single alternative is chosen.)

It is important to understand the difference between the use of the word "outcome" and the use of the word "alternative" in a margins specification or in output from margins. Whenever the word "outcome" is used in margins, it refers to what alternative is chosen.

Whenever the word "alternative" is used in a margins specification, it means do something special by alternative when margins operates on alternative-specific variables. So when you see "alternative" used with margins, do not think of alternatives as outcomes, think of manipulating alternative-specific variables at the observations corresponding to the alternatives. If you specify one of the alternative* options when there are only case-specific variables in the margins specification, it does nothing. It is simply ignored, no error message is given. You will not see the word alternative used in margins output; instead, you will see the name of the alternatives variable that you specified with cmset.

Let's explain what margins does after cm estimators using a simple example. Suppose cost_cat is a case-specific categorical variable, and we included i.cost_cat as a casevar() in our cm estimation. If we now type

```
. margins cost_cat
```

the output will show the average predicted probability of selecting each alternative (as a possible outcome) at each of the levels of cost_cat. If there are k levels of cost_cat and n possible alternatives, there will be $k \times n$ predicted probabilities in the output from margins.

If, however, cost_cat were an alternative-specific variable, then margins would display $k \times n \times n$ predicted probabilities. Alternative-specific variables are variables that vary across both alternatives and cases, and each alternative-specific variable can be thought of as n different variables, one for each alternative. (Indeed, it is only because cm commands require data in long format that each alternative-specific variable is stored as a single Stata variable. If the storage design had been wide format, there would be n Stata variables for each alternative-specific variable.)

Suppose cost_cat is alternative specific and its levels are 1, 2, 3, and 4. Suppose the possible alternatives are car, bus, and train. margins first sets the level of cost_cat to 1 for the observations corresponding to alternative car. At the observations corresponding to the other alternatives, cost_cat is, by default, kept at its observed value. Then average predicted probabilities are calculated for each of the outcomes car, bus, and train. Then cost_cat is set to 2, still at alternative car only, and three more probabilities are calculated. This process continues until all $k \times n \times n = 4 \times 3 \times 3 = 36$ predicted probabilities are estimated. So the default output of margins can contain an overwhelming number of predicted probabilities for alternative-specific variables when n is not small. Some of the special options for margins after the cm estimation commands are there for the purpose of reducing the number of probabilities estimated and displayed.

The outcome(*levels*) option restricts the probabilities estimated by margins to the probabilities of the decision maker choosing only the alternatives in *levels*. This restriction works for both case-specific and alternative-specific variables.

The alternative(*levels*) option applies only to alternative-specific variables. With this option, margins changes only the variable at the observations corresponding to the alternatives in *levels*. For example, if we typed

```
. margins cost_cat, outcome(Car) alternative(Bus)
```

margins would estimate the average predicted probability of choosing car for cost_cat set to each of its levels, 1, 2, 3, and 4, at observations corresponding to the alternative bus only.

margins uses numerical derivatives to calculate standard errors. These computations can take a long time when your data have lots of cases, when there are lots of alternatives, or when there are lots of levels in the covariates in the margins specification. This is another reason to use outcome() and alternative(). Restricting the estimates to a smaller set of possibilities reduces computation time.

When margins is taking a long time to calculate estimates, you may want to first run margins with its nose option, which skips the standard error calculations.

```
. margins ..., ... nose
```

You can check the output and confirm that your specification of the margins command is what you want. Then, you can run it again without nose to get the standard errors.

Estimating margins for case-specific variables

When all the variables in the margins specification are case-specific variables, there is less output and it is easier to interpret.

▷ Example 1: Only case-specific variables

In example 1 of [CM] **cmclogit**, we fit a model of consumer choice of automobile. The alternatives are nationality of the automobile manufacturer: American, Japanese, European, and Korean. There is one alternative-specific variable in the model, dealers, that contains the number of dealerships of each nationality in the consumer's city. The case-specific variables are gender and income, the consumer's income in thousands of dollars.

We load the data and cmset them. For this example, we create a categorical variable, income_cat, that contains quartiles of income. We specify it as a case-specific variable along with gender and fit a cmclogit model:

```
. use https://www.stata-press.com/data/r18/carchoice
(Car choice data)
. cmset consumerid car
note: alternatives are unbalanced across choice sets; choice sets of
      different sizes found.
     Case ID variable: consumerid
Alternatives variable: car
. xtile income_cat = income, nquantiles(4)
```

```
. cmclogit purchase dealers, casevars(i.gender i.income_cat)
Iteration 0:  Log likelihood = -960.62626
Iteration 1:  Log likelihood = -950.15551
Iteration 2:  Log likelihood = -949.74982
Iteration 3:  Log likelihood = -949.74885
Iteration 4:  Log likelihood = -949.74885

Conditional logit choice model                  Number of obs      =      3,075
Case ID variable: consumerid                    Number of cases    =        862

Alternatives variable: car                      Alts per case: min =          3
                                                               avg =        3.6
                                                               max =          4

                                                  Wald chi2(13)    =      48.28
Log likelihood = -949.74885                       Prob > chi2      =     0.0000
```

purchase	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
car						
dealers	.04461	.0263695	1.69	0.091	-.0070733	.0962932
American	(base alternative)					
Japanese						
gender						
Male	-.4224753	.1906288	-2.22	0.027	-.7961009	-.0488497
income_cat						
2	.0946656	.242739	0.39	0.697	-.3810941	.5704254
3	.5016051	.2215662	2.26	0.024	.0673433	.935867
4	.4315078	.2280086	1.89	0.058	-.0153808	.8783964
_cons	-.0483645	.2470234	-0.20	0.845	-.5325214	.4357924
European						
gender						
Male	.6949682	.2814506	2.47	0.014	.1433352	1.246601
income_cat						
2	.496032	.3314079	1.50	0.134	-.1535156	1.14558
3	.8082547	.2960338	2.73	0.006	.2280392	1.38847
4	.9865233	.2939615	3.36	0.001	.4103694	1.562677
_cons	-1.969437	.3723598	-5.29	0.000	-2.699249	-1.239625
Korean						
gender						
Male	.0257312	.4913433	0.05	0.958	-.937284	.9887464
income_cat						
2	-.638151	.5053717	-1.26	0.207	-1.628661	.3523593
3	-.8951082	.5090436	-1.76	0.079	-1.892815	.102599
4	-1.082247	.5441119	-1.99	0.047	-2.148687	-.0158072
_cons	-.8522904	.5679136	-1.50	0.133	-1.965381	.2607998

We now use `margins` to get average predicted probabilities for the different levels of `income_cat`.

```
. margins income_cat
Predictive margins                                      Number of obs = 3,075
Model VCE: OIM
Expression: Pr(car|1 selected), predict()
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome# income_cat						
American#1	.4941932	.0345185	14.32	0.000	.4265382	.5618482
American#2	.4677756	.0369774	12.65	0.000	.3953013	.5402499
American#3	.385185	.0338328	11.38	0.000	.3188739	.4514961
American#4	.3850117	.0335619	11.47	0.000	.3192316	.4507918
Japanese#1	.3237821	.0328847	9.85	0.000	.2593293	.3882349
Japanese#2	.3375662	.0339412	9.95	0.000	.2710427	.4040897
Japanese#3	.4165363	.0340384	12.24	0.000	.3498223	.4832504
Japanese#4	.3891954	.0332832	11.69	0.000	.3239615	.4544294
European#1	.0971881	.0181104	5.37	0.000	.0616924	.1326838
European#2	.1506216	.0282786	5.33	0.000	.0951966	.2060467
European#3	.1698688	.0255855	6.64	0.000	.1197222	.2200154
European#4	.2021599	.0277523	7.28	0.000	.1477663	.2565534
Korean#1	.0848366	.0194847	4.35	0.000	.0466472	.123026
Korean#2	.0440366	.0150923	2.92	0.004	.0144562	.073617
Korean#3	.0284098	.0113046	2.51	0.012	.0062532	.0505665
Korean#4	.023633	.0103463	2.28	0.022	.0033546	.0439114

Rows are labeled first by the outcome category, the alternative that is chosen; and second by the value of `income_cat`. The first column in the body of the table gives predicted probabilities of the outcome. If we have a random or otherwise representative sample, these are the expected probabilities based on our model. The 0.494 next to `American#1` is the expected probability of a consumer with `income_cat` $= 1$ buying an American car. Said differently, we expect 49.4% of individuals to buy American cars if they are in the first income quartile and have the same distribution of `dealers` and `gender` that we observe in the data.

The probability is computed by setting `income_cat` $= 1$ for all persons, leaving the other variables unchanged, calculating the predicted probability for each outcome with these altered data, and then averaging the probabilities. And so on for the other levels of `income_cat`. So, to be precise, what is calculated by `margins` is an average predicted probability.

Examination of the table shows that as income increases, consumers are less likely to buy American cars and less likely to buy Korean cars but more likely to buy European cars. The association between Japanese cars and income categories is less clear. Admittedly, it would be easier to spot these changes if we plotted the probabilities instead of looking at the values reported in this table. To see a plot, you can simply type `marginsplot`, as we demonstrate in [CM] **Intro 1** and in example 4 below.

We can test for the effect of `income_cat` on the probability of selecting each nationality of car. To get the joint test of any difference in expected probabilities across income levels for each outcome, we specify the option `contrast`.

```
. margins income_cat, contrast
Contrasts of predictive margins                       Number of obs = 3,075
Model VCE: OIM
Expression: Pr(car|1 selected), predict()
```

	df	chi2	P>chi2
income_cat@_outcome			
American	3	8.50	0.0368
Japanese	3	5.33	0.1493
European	3	11.93	0.0076
Korean	3	9.00	0.0293
Joint	9	27.52	0.0011

We see that the joint test of any difference across income levels has the smallest p-value for European cars (0.0076). The joint test of the effect of income is nonsignificant for Japanese cars.

We can use an *op*. contrast operator to see differences (contrasts). The contrast operators most typically used in this context are `r.`, differences from the reference level; `a.`, differences from the next level (adjacent contrasts); and `ar.`, differences from the previous level (reverse adjacent contrasts). See the *op*. table in [R] **contrast** for a list of all contrast operators.

Here we use the `ar.` contrast operator to estimate differences in expected probabilities between each level of `income_cat` and its previous level. We also specify the `outcome()` option to restrict the results to the probability of buying a European car.

```
. margins ar.income_cat, outcome(European)
Contrasts of predictive margins                       Number of obs = 3,075
Model VCE: OIM
Expression: Pr(car|1 selected), predict()
Outcome:    European
```

	df	chi2	P>chi2
income_cat			
(2 vs 1)	1	2.42	0.1199
(3 vs 2)	1	0.25	0.6152
(4 vs 3)	1	0.73	0.3934
Joint	3	11.93	0.0076

	Contrast	Delta-method std. err.	[95% conf.	interval]
income_cat				
(2 vs 1)	.0534335	.0343615	-.0139138	.1207808
(3 vs 2)	.0192472	.0382871	-.0557941	.0942885
(4 vs 3)	.032291	.0378334	-.0418611	.1064432

From the second table, the line labeled `(2 vs 1)` estimates that the probability of selecting a European car increases by 0.053 when we go from the first income category to the second. The p-value for this difference is given in the first table on the line `(2 vs 1)`, $p = 0.12$, and we see that this difference is not significant at the 5% level. In fact, none of these reverse adjacent contracts are significant. The joint significance reported in the first table is 0.0076 and is, of course, the same as what was calculated by the previous `margins` command.

Instead of testing whether income has an effect on the expected probability of selecting one nationality of car, we might want to test whether the effects of income are different for different

nationalities of cars. The option `contrast(outcomecontrast(`*op*`._outcome))` can be used to get tests of the differences between outcomes in the differences of expected probabilities across income levels.

For instance, when we typed `margins income_cat`, we saw that the expected probability of selecting a Japanese car was 0.338 for the second income category and was 0.324 for the first income category, a difference of 0.014. If we look at the probabilities of selecting an American car, we get 0.468 for the second income category and 0.494 for the first income category, a difference of −0.026. The differences of 0.014 and −0.026 have opposite signs, but are they statistically different from each other? We could ask the same question for differences in the third versus first income categories and for differences in the fourth versus first income categories. The `contrast(outcomecontrast())` option gives a joint test of all of these differences—a test of whether `income_cat` has the same effect on the probability of selecting a Japanese car as it has on the probability of selecting an American car.

We use the `r.` contrast operator to get differences between outcomes relative to the first level of the alternatives variable, which is `American`.

```
. margins income_cat, contrast(outcomecontrast(r._outcome))
Contrasts of predictive margins                         Number of obs = 3,075
Model VCE: OIM
Expression: Pr(car|1 selected), predict()
```

	df	chi2	P>chi2
_outcome#income_cat			
(Japanese vs American) (joint)	3	8.07	0.0446
(European vs American) (joint)	3	13.16	0.0043
(Korean vs American) (joint)	3	2.28	0.5167
Joint	9	27.52	0.0011

To be clear about what is being tested: The test labeled "`(Japanese vs American) (joint)`" is a test of the null hypothesis,

$$\Pr(\texttt{Japanese\#2}) - \Pr(\texttt{Japanese\#1}) = \Pr(\texttt{American\#2}) - \Pr(\texttt{American\#1})$$
$$\Pr(\texttt{Japanese\#3}) - \Pr(\texttt{Japanese\#1}) = \Pr(\texttt{American\#3}) - \Pr(\texttt{American\#1})$$
$$\Pr(\texttt{Japanese\#4}) - \Pr(\texttt{Japanese\#1}) = \Pr(\texttt{American\#4}) - \Pr(\texttt{American\#1})$$

where `Japanese#2` denotes the outcome of choosing a Japanese car for `income_cat` $= 2$, etc.

At the 5% level, there are significant differences in the effect of income on the probability of selecting Japanese versus American cars and in the probability of selecting European versus American cars.

We do not need to perform all of these tests at once. If we are interested only in testing the difference in effect of income on Japanese versus American outcomes, we can use the `r(1/2).` operator to restrict the outcome levels.

```
. margins income_cat, contrast(outcomecontrast(r(1/2)._out))
Contrasts of predictive margins                         Number of obs = 3,075
Model VCE: OIM
Expression: Pr(car|1 selected), predict()
```

	df	chi2	P>chi2
_outcome#income_cat	3	8.07	0.0446

The result is the same as it was in the previous `margins` output for the test of Japanese versus American outcomes.

◁

Estimating margins for alternative-specific variables

For alternative-specific variables, we can explore even more possibilities using `margins`. For instance, we can estimate the effect of changing the value of an alternative-specific variable at only one of the alternatives, or we could change its value across all alternatives. As we discussed earlier, even when an alternative-specific variable is changed only at one value of the alternatives, it creates changes in the predicted probabilities of selecting an outcome for all the possible outcomes. To handle this additional complexity, the option `alternative()` is extremely useful when we want to test hypotheses about alternative-specific variables that involve only one (or a subset) of the alternatives. We demonstrate this below.

▷ Example 2: Alternative-specific variables

We continue with the same `cmclogit` choice model on the nationality of car purchased. The model was fit with an alternative-specific variable `dealers`, which contains the number of dealerships of each nationality of car in the individual's community.

Would increasing the number of dealerships for a certain nationality of car affect the likelihood of more people buying that car? And at what nationality's expense? For example, if increasing the number of Korean dealerships increases the probability of buying a Korean car, then the probability of buying an American, a Japanese, or a European car must go down—and we would like to know which one has the biggest decrease in probability. (It is possible one of these could also go up, but because the changes in probabilities must sum to zero, one of the changes must be negative if the change for Korean cars is positive.)

`margins` can answer these questions—based on the fitted `cmclogit` model.

`margins` by default produces a lot of output as we discussed earlier. Here there are four outcomes in this model, one for each nationality of car. There are also four ways to change the alternative-specific variable `dealers`. We can change it just for Korean dealerships, or just for American dealerships, or Japanese, or European. Restricting the change to a particular alternative—in this case, Korean—is what we want.

We can run `margins` with two `at()` options, the first with the `dealers` set to the original value and the second with `dealers` increased by one. Because we specify the option `alternative(Korean)`, only `dealers` corresponding to the alternative Korean are increased.

```
. margins, at(dealers=generate(dealers)) at(dealers=generate(dealers+1))
> alternative(Korean)
Predictive margins                                      Number of obs = 3,075
Model VCE: OIM

Expression:  Pr(car|1 selected), predict()
Alternative: Korean

1._at: dealers =   dealers

2._at: dealers = dealers+1
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome#_at						
American#1	.4361949	.0167567	26.03	0.000	.4033524	.4690374
American#2	.4352739	.0167586	25.97	0.000	.4024276	.4681202
Japanese#1	.3665893	.0162132	22.61	0.000	.3348121	.3983666
Japanese#2	.3659044	.016203	22.58	0.000	.3341472	.3976617
European#1	.1508121	.0120258	12.54	0.000	.1272419	.1743822
European#2	.1505359	.0120085	12.54	0.000	.1269998	.1740721
Korean#1	.0464037	.0069344	6.69	0.000	.0328124	.059995
Korean#2	.0482858	.007271	6.64	0.000	.0340348	.0625367

These results indicate that if we add one Korean dealership in each community, we would expect the percentage of individuals purchasing a Korean car to go from 4.64% to 4.82%. Said another way, we expect the probability of buying a Korean car to increase from 0.0464 to 0.0482.

We run `margins` again with the `contrast(atcontrast(`*op*`))` option to estimate each of the differences in expected probabilities. Here *op* represents a contrast operator. There are only two `at()`'s to contrast, so any pairwise operator will do. We use the `r` operator.

```
. margins, at(dealers=generate(dealers)) at(dealers=generate(dealers+1))
> alternative(Korean) contrast(atcontrast(r))
Contrasts of predictive margins                         Number of obs = 3,075
Model VCE: OIM

Expression:  Pr(car|1 selected), predict()
Alternative: Korean

1._at: dealers =   dealers

2._at: dealers = dealers+1
```

	df	chi2	P>chi2
_at@_outcome			
(2 vs 1) American	1	2.65	0.1036
(2 vs 1) Japanese	1	2.63	0.1051
(2 vs 1) European	1	2.49	0.1148
(2 vs 1) Korean	1	2.64	0.1041
Joint	3	2.67	0.4454

	Contrast	Delta-method std. err.	[95% conf.	interval]
_at@_outcome				
(2 vs 1) American	-.000921	.0005658	-.00203	.0001879
(2 vs 1) Japanese	-.0006849	.0004226	-.0015131	.0001433
(2 vs 1) European	-.0002761	.0001751	-.0006194	.0000671
(2 vs 1) Korean	.001882	.001158	-.0003875	.0041516

Based on these results, we expect the probability of buying a Korean car to increase by 0.0019. Most of this increase came at the expense of American and Japanese cars. We expect that the probability of buying an American car will decrease by 0.0009, and the probability of buying a Japanese car will decrease by 0.0007. The probability of buying a European car decreases only by 0.0003.

◁

The altsubpop suboption for unbalanced choice sets

Not everyone in this dataset has Korean as one of his or her possible choices for a car. The choice sets are unbalanced. This can be seen by running cmchoiceset.

```
. cmchoiceset, generate(choiceset)
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2 3	380	42.94	42.94
1 2 3 4	505	57.06	100.00
Total	885	100.00	

```
Note: Total is number of cases.
```

The value of 4 for the alternatives variable car represents Korean. So we see that 380 persons out of a total of 885 do not have Korean as a choice.

How does margins handle the cases in which a particular alternative is not part of the choice set for the case? By default, margins considers an alternative that is missing from a choice set to have zero probability of being chosen. This makes sense if we are looking at the change in buying a Korean car for a change in a variable like income. There are no Korean dealerships in the community, so even if income changes, there is still no way for a person in that community to purchase a Korean car (ignoring the possibility of buying a Korean car in another community).

The outcome() option of margins has a suboption altsubpop, which changes the way margins handles alternatives that are not present in the case's choice set. When altsubpop is specified, the results for each outcome are restricted to those cases that have that outcome in their choice set. Here is what we get when we use altsubpop.

```
. margins, at(dealers=generate(dealers)) at(dealers=generate(dealers+1))
> alternative(Korean) outcome(_all, altsubpop)
Predictive margins                                      Number of obs = 3,075
Model VCE: OIM
Expression:  Pr(car|1 selected), predict()
Alternative: Korean
1._at: dealers =   dealers
2._at: dealers = dealers+1
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome#_at						
American#1	.4361949	.0167567	26.03	0.000	.4033524	.4690374
American#2	.4352739	.0167586	25.97	0.000	.4024276	.4681202
Japanese#1	.3665893	.0162132	22.61	0.000	.3348121	.3983666
Japanese#2	.3659044	.016203	22.58	0.000	.3341472	.3976617
European#1	.1508121	.0120258	12.54	0.000	.1272419	.1743822
European#2	.1505359	.0120085	12.54	0.000	.1269998	.1740721
Korean#1	.0817996	.0122239	6.69	0.000	.0578412	.105758
Korean#2	.0851172	.0128173	6.64	0.000	.0599959	.1102386

We find that for the subpopulation of individuals who had Korean in their choice set, the expected probability of selecting a Korean car is 0.0818. For the same subpopulation, if we increase the number of Korean dealerships by 1, the probability of selecting a Korean car goes to 0.0851. We can use the `contrast()` option to estimate the effect.

```
. margins, at(dealers=generate(dealers)) at(dealers=generate(dealers+1))
> alternative(Korean) contrast(atcontrast(r)) outcome(_all, altsubpop)
Contrasts of predictive margins                         Number of obs = 3,075
Model VCE: OIM

Expression:  Pr(car|1 selected), predict()
Alternative: Korean

1._at: dealers =   dealers

2._at: dealers = dealers+1
```

	df	chi2	P>chi2
_at@_outcome			
(2 vs 1) American	1	2.65	0.1036
(2 vs 1) Japanese	1	2.63	0.1051
(2 vs 1) European	1	2.49	0.1148
(2 vs 1) Korean	1	2.64	0.1041
Joint	3	2.67	0.4454

	Contrast	Delta-method std. err.	[95% conf.	interval]
_at@_outcome				
(2 vs 1) American	-.000921	.0005658	-.00203	.0001879
(2 vs 1) Japanese	-.0006849	.0004226	-.0015131	.0001433
(2 vs 1) European	-.0002761	.0001751	-.0006194	.0000671
(2 vs 1) Korean	.0033176	.0020412	-.0006831	.0073184

The change in the expected probability of buying a Korean car is estimated at 0.0033 for this subpopulation, a considerable difference from the previous estimate of 0.0019. This is not surprising. The earlier estimate considered those cases without Korean in their choice set as having a zero probability of buying a Korean car even when we added a Korean dealer. The estimate of 0.0033 ignores those cases with zero probability.

Note also that changes in probabilities no longer sum to zero. The number of cases for each estimate varies, so we would not expect them to sum to zero. The estimates for the Korean outcome are only for the subpopulation of individuals who had Korean in their choice set, while all other estimates are for the full population.

Another way to handle unbalanced choice sets is to use the `subpop()` option (or the `over()` option) with `margins` and use an indicator variable for the different choice sets. We created such a variable and called it `choiceset` when we ran `cmchoiceset` earlier. See [CM] **cmchoiceset**. Here is `margins` using `subpop()` to restrict the sample to the cases that have all four alternatives.

```
. margins, at(dealers=generate(dealers)) at(dealers=generate(dealers+1))
> alternative(Korean) contrast(atcontrast(r) nowald)
> subpop(if choiceset=="1 2 3 4":choiceset)
Contrasts of predictive margins                         Number of obs   = 3,075
Model VCE: OIM                                          Subpop. no. obs = 1,956

Expression:  Pr(car|1 selected), predict()
Alternative: Korean

1._at: dealers =   dealers

2._at: dealers = dealers+1
```

	Contrast	Delta-method std. err.	[95% conf.	interval]
_at@_outcome				
(2 vs 1) American	-.0016235	.0009974	-.0035784	.0003313
(2 vs 1) Japanese	-.0012073	.0007449	-.0026672	.0002526
(2 vs 1) European	-.0004868	.0003087	-.0010918	.0001183
(2 vs 1) Korean	.0033176	.0020412	-.0006831	.0073184

We now have results only for those cases having all four choices, and the changes in probabilities sum to zero.

For observational data, the default behavior of `margins` is likely what you want. If an alternative was not in a decision maker's choice set, how can changing a covariate make it possible to choose that alternative? The assumption is that the reason the alternative is not in the choice set is that the alternative does not exist for that decision maker under any condition. If the choices for commuters are car, train, or bus, but there is no train service in a commuter's community, then the probability of that commuter taking a train to work is zero.

Imagine, however, a different type of study in which, by design, individuals were not offered a particular alternative. Suppose, for example, a marketeer is testing consumer preferences among six different types of breakfast cereal. He or she thinks that giving each consumer six different cereals to taste would be overwhelming. So the marketer gives each consumer only four cereals. Is it reasonable to keep the probability of picking a cereal not offered fixed at zero when looking at estimates for the entire sample? If it had been offered to someone to whom it was not, he or she might have chosen it. Using `altsubpop` in this case seems not only reasonable but also, perhaps, essential. We want to make comparisons only among those alternatives that persons were able to choose between.

More on unbalanced choice sets

When we are looking at changing the number of Korean dealerships to see its effect on buying cars of different nationalities, neither the default, the `altsubpop` suboption, nor `subpop()` really does what we want. We want to increase the number of Korean dealerships in all communities, including those who currently do not have any Korean dealerships in their community. We now show how this could be done.

First, we use the command `expand` to add observations to the cases that do not have Korean (`car` $= 4$) in their choice set. We generate a variable, `new`, that flags the newly created observations. See [D] **expand** for details.

```
. expand 2 if choiceset=="1 2 3":choiceset & car==3, gen(new)
(380 observations created)
```

Second, for new observations, we set the alternatives variable `car` equal to 4 (representing Korean) and `dealers` equal to 0. Then, we run `cmchoiceset` to confirm we did what we wanted.

```
. replace car = 4 if new==1
(380 real changes made)
. replace dealers = 0 if new==1
(380 real changes made)
. cmchoiceset
Tabulation of choice-set possibilities
```

Choice set	Freq.	Percent	Cum.
1 2 3 4	885	100.00	100.00
Total	885	100.00	

```
Note: Total is number of cases.
```

We can now estimate the probability of selecting each nationality of car after adding a Korean dealership to all communities. We run `margins` with the `noesample` option because we are now doing predictions outside the estimation sample.

```
. margins, at(dealers=generate(dealers+1)) alternative(Korean) noesample
Predictive margins                                      Number of obs = 3,448
Model VCE: OIM
Expression:  Pr(car|1 selected), predict()
Alternative: Korean
At: dealers = dealers+1
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome						
American	.4177746	.0169372	24.67	0.000	.3845782	.450971
Japanese	.3532169	.0160733	21.98	0.000	.3217138	.38472
European	.1454895	.0117044	12.43	0.000	.1225493	.1684298
Korean	.0835189	.0124662	6.70	0.000	.0590857	.1079522

We find that if we add one Korean dealership in each community, including those that had no dealerships originally, the expected probability of selecting a Korean car is 0.0835. When we used `altsubpop` and considered only the subpopulation of communities that had a Korean car in their choice set, this expected probability was just slightly larger, 0.0851.

▷ Example 3: margins, contrast

Testing contrasts with `margins` after the `cm` estimation commands can be overwhelming, especially with alternative-specific variables, because there are so many possibilities. Contrasts can be made between different outcomes (for both case-specific and alternative-specific variables). Contrasts can also be made between the alternatives at which alternative-specific variables are changed.

This example explains how you can read the output from `margins` to understand exactly what is being tested. For those familiar with manually defining and testing contrasts, we also show you how to find the contrast matrix to see exactly what `margins` is testing for any contrast it performs. In this example, we use `margins` after `cmmprobit` but recall what we said earlier. All the special options of `margins` for use after `cm` estimation commands work in the same way after `cmclogit`, `cmmprobit`, `cmroprobit`, `cmmixlogit`, and `cmxtmixlogit`. So this example applies to all of these commands.

We use data from example 1 in [CM] **cmmprobit**. The data represent individuals' choices of travel mode: `air`, `train`, `bus`, or `car`. There are two alternative-specific variables: `travelcost`, a measure of generalized cost of travel; and `termtime`, time spent in the terminal. The variable `income` gives household income and is case specific.

We load the data and cmset them. The command xtile is used to make a categorical variable, cost_cat, that contains tertiles of travelcost. This is the alternative-specific categorical variable we will use with margins. We then fit our cmmprobit model.

```
. use https://www.stata-press.com/data/r18/travel, clear
(Modes of travel)
. cmset id mode
     Case ID variable: id
Alternatives variable: mode
. xtile cost_cat = travelcost, nquantiles(3)
. cmmprobit choice i.cost_cat termtime, casevars(income)
note: variable 2.cost_cat has 70 cases that are not alternative-specific;
      there is no within-case variability.
note: variable 3.cost_cat has 113 cases that are not alternative-specific;
      there is no within-case variability.
  (iteration log omitted)
Multinomial probit choice model                 Number of obs      =        840
Case ID variable: id                            Number of cases    =        210
Alternatives variable: mode                     Alts per case: min =          4
                                                               avg =        4.0
                                                               max =          4
Integration sequence:      Hammersley
Integration points:               601           Wald chi2(6)       =      36.54
Log simulated-likelihood = -190.38007           Prob > chi2        =     0.0000
```

choice	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
mode						
cost_cat						
2	.2155852	.2214832	0.97	0.330	-.2185139	.6496843
3	-.3822223	.2805587	-1.36	0.173	-.9321073	.1676628
termtime	-.043736	.008679	-5.04	0.000	-.0607466	-.0267255
Air	(base alternative)					
Train						
income	-.0340284	.0092383	-3.68	0.000	-.0521352	-.0159216
_cons	.4632755	.3902352	1.19	0.235	-.3015715	1.228122
Bus						
income	-.0136801	.008338	-1.64	0.101	-.0300223	.0026622
_cons	-.1941561	.4585625	-0.42	0.672	-1.092922	.7046098
Car						
income	-.0039416	.0082461	-0.48	0.633	-.0201036	.0122204
_cons	-2.100911	.7742118	-2.71	0.007	-3.618338	-.5834833
/lnl2_2	-.3689019	.3215014	-1.15	0.251	-.999033	.2612293
/lnl3_3	-.4548538	.3229243	-1.41	0.159	-1.087774	.1780661
/l2_1	1.075525	.2157021	4.99	0.000	.6527562	1.498293
/l3_1	.9613768	.2451866	3.92	0.000	.4808199	1.441934
/l3_2	.6096931	.2902669	2.10	0.036	.0407804	1.178606

```
(mode=Air is the alternative normalizing location)
(mode=Train is the alternative normalizing scale)
. estimates store ourmodel
```

Sometimes, we want to test a single, simple hypothesis. For instance, we could test whether changing the cost of the `train` alternative affects the probability of selecting the `air` outcome. We can request this test and also estimate the differences in expected probabilities across cost categories by typing

```
. margins r.cost_cat, alternative(Train) outcome(Air)
Contrasts of predictive margins                               Number of obs = 840
Model VCE: OIM
Expression:  Pr(mode), predict()
Alternative: Train
Outcome:     Air
```

	df	chi2	P>chi2
cost_cat			
(2 vs 1)	1	1.00	0.3184
(3 vs 1)	1	1.52	0.2176
Joint	2	6.49	0.0390

	Contrast	Delta-method std. err.	[95% conf.	interval]
cost_cat				
(2 vs 1)	-.0153519	.015386	-.045508	.0148042
(3 vs 1)	.0226503	.0183692	-.0133527	.0586533

The joint test of the effect of `cost_cat` for the `train` alternative on the probability of selecting `air` travel is significant, with a p-value of 0.0390.

What if we wanted to perform all such tests—all tests of the effect of `cost_cat` for each alternative on the expected probabilities of each outcome? We specify the `contrast` option with `margins`:

```
. margins cost_cat, contrast
Contrasts of predictive margins                               Number of obs = 840
Model VCE: OIM
Expression: Pr(mode), predict()
```

	df	chi2	P>chi2
cost_cat@_outcome#mode			
Air Air	2	15.45	0.0004
Air Train	2	6.49	0.0390
Air Bus	2	5.59	0.0612
Air Car	2	10.96	0.0042
Train Air	2	6.92	0.0315
Train Train	2	9.75	0.0076
Train Bus	2	8.87	0.0119
Train Car	2	7.11	0.0286
Bus Air	2	5.68	0.0584
Bus Train	2	8.36	0.0153
Bus Bus	2	10.45	0.0054
Bus Car	2	7.77	0.0206
Car Air	2	11.09	0.0039
Car Train	2	6.64	0.0361
Car Bus	2	7.47	0.0238
Car Car	2	11.85	0.0027
Joint	14	307.62	0.0000

Now each row of the output gives a test of the effect of cost_cat on the probability of selecting each outcome by each alternative, but we have to look carefully to understand which hypothesis is being tested in each line. Let's look at the second row of the output. This tests the null hypothesis that the cost of the train alternative has no effect on the probability of selecting the air outcome. This is the same hypothesis we tested with our previous margins command.

Having run the previous margins command, we could easily spot the row in this output that tested the same hypothesis. But if we ran only the margins cost_cat, contrast command, how would we determine what the hypothesis on a given row is? Recall what was said earlier about the use of "outcome" and "alternative" in margins specifications. The key for the labels on the table is cost_cat@_outcome#mode. The first part of the key, cost_cat@, means we are testing differences across cost_cat. The second part of the key, _outcome#mode, is where the differences are being tested. _outcome is the alternative hypothetically chosen. mode, which is the alternatives variable, gives the alternative at which the value of cost_cat is being changed.

The joint test shown in the last row is a test of the null hypothesis: within each outcome by alternative group, expected probabilities across levels of cost_cat are the same; that is, cost_cat has no effect anywhere.

We can duplicate the results manually using the test command and the contrast coefficients that margins uses. margins stores in r(L) the matrix of contrasts that are tested.

```
. matrix list r(L)
r(L)[32,48]
                    1._outcome#  1._outcome#          1._outcome#
                       2.mode#      2.mode#              2.mode#
                ...  1.cost_cat   2.cost_cat    ...   3.cost_cat
...
2.cost_cat@
1._outcome#
   2.mode                   -1            1                     0
...
3.cost_cat@
1._outcome#
   2.mode                   -1            0                     1
```

It is a huge matrix. We will again focus on the two-degrees-of-freedom test reported in the second line of the previous margins output that is labeled air train, so we show only the relevant portion of r(L) here. Our test is based on rows of this matrix that include the @1.outcome#2.mode in the row label. We know this because our alternatives variable mode is coded as 1 for air (our _outcome) and 2 for train (our mode). The nonzero elements within each row define a single contrast. In the first row of r(L) that we displayed here, it shows the contrast of the expected probabilities for the first and second levels of cost_cat. The contrast in the second row compares the expected probabilities for the first and third levels of cost_cat. The column labels show the syntax that we can later use to perform our own test.

We now run margins with the post option to save the results of margins as if it were an estimation command.

```
. margins cost_cat, post
Predictive margins                                      Number of obs = 840
Model VCE: OIM
Expression: Pr(mode), predict()
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome# mode# cost_cat						
Air#Air#1	.2713238	.0427612	6.35	0.000	.1875135	.3551341
Air#Air#2	.3165716	.0324305	9.76	0.000	.2530089	.3801342
Air#Air#3	.2004557	.0340447	5.89	0.000	.1337293	.2671821
Air#Train#1	.2764299	.0300901	9.19	0.000	.2174544	.3354053
Air#Train#2	.261078	.0283491	9.21	0.000	.2055147	.3166413
Air#Train#3	.2990802	.0292282	10.23	0.000	.2417939	.3563665
(*output omitted*)						

```
. estimates store ourmargins
```

Our test for an effect of the cost of the train travel on the probability of selecting the air travel is just a test for a difference in the expected probabilities labeled air#train#1, air#train#2, and air#train#3 above.

We use test with the syntax we saw in r(L).

```
. test (1._outcome#2.mode#1.cost_cat = 1._outcome#2.mode#2.cost_cat)
>      (1._outcome#2.mode#1.cost_cat = 1._outcome#2.mode#3.cost_cat)
 ( 1)  1bn._outcome#2.mode#1bn.cost_cat - 1bn._outcome#2.mode#2.cost_cat = 0
 ( 2)  1bn._outcome#2.mode#1bn.cost_cat - 1bn._outcome#2.mode#3.cost_cat = 0
           chi2(  2) =    6.49
         Prob > chi2 =    0.0390
```

We duplicated the second row of output from margins cost_cat, contrast.

◁

The outcomecontrast() and alternativecontrast() suboptions

margins has two other options, contrast(outcomecontrast(*op*)) and contrast(alternativecontrast(*op*)), that perform joint tests of hypothesis after fitting choice models. We use contrast(outcomecontrast(*op*)) to test for differences across outcomes; we demonstrate this below. We use contrast(alternativecontrast(*op*)) to test for differences across alternatives.

Continuing our example, we use the contrast(outcomecontrast(*op*)) option to test for differences in the effects of cost_cat across outcomes. Before we can run margins, we must get our cmmprobit estimation results back (because the estimation results currently active are those from margins).

```
. estimates restore ourmodel
(results ourmodel are active now)
```

Now we can test whether changing the cost of the air travel alternative has the same effect on the probability of selecting the train outcome as it has on the probability of selecting the air outcome.

```
. margins r.cost_cat, alternative(Air) outcome(Train Air)
> contrast(outcomecontrast(r))
Contrasts of predictive margins                         Number of obs = 840
Model VCE: OIM
Expression:  Pr(mode), predict()
Alternative: Air
```

	df	chi2	P>chi2
_outcome#cost_cat			
(Train vs Air) (2 vs 1)	1	1.01	0.3157
(Train vs Air) (3 vs 1)	1	1.87	0.1711
Joint	2	14.12	0.0009

	Contrast	Delta-method std. err.	[95% conf.	interval]
_outcome#cost_cat				
(Train vs Air) (2 vs 1)	-.058856	.0586622	-.1738319	.0561199
(Train vs Air) (3 vs 1)	.0912034	.0666299	-.0393888	.2217957

There is a lot going on in this margins command. By specifying r.cost_cat, we requested differences in expected probabilities when comparing levels of the cost_cat variable. The alternative(Air) option tells margins that we want to estimate these differences only when changing the cost of the air alternative. The outcome(Train Air) option specifies that we want to estimate only these differences in expected probabilities of selecting the train outcome and the air outcome. Finally, contrast(outcomecontrast(r)) says that we want to test whether the differences are the same for train travel and for air travel. Thus, we are testing whether the effect of the cost of air travel is the same on the probability of selecting train travel as it is on the probability of selecting air travel.

In the output from this command, the contrast labeled (train vs air) (2 vs 1) is the difference in the effect of changing the cost_cat of air travel from 1 to 2 on the probability of selecting train versus air travel. The p-value reported in the first table for this test is 0.3157. We do not have evidence that changing the cost of air travel from the first tertile to the second has different effects on the probabilities of selecting train and air travel. Similarly, looking at the lines labeled (train vs air) (3 vs 1), we find no evidence that the effect of changing the cost of air travel from the first tertile to the third tertile has different effects on the probabilities of selecting train and air travel.

The joint test is provided in the last line of the top table in the output. With a p-value of 0.0009, we reject the null hypothesis that the effects of the cost_cat of air travel on the probability of selecting train travel are the same as the effects of the cost_cat of air travel on the probability of selecting air travel.

If we are interested in all tests comparing the effects of the cost_cat of one alternative on the probabilities of selecting two different outcomes, we can run margins again but without the alternative() and outcome() options.

```
. margins cost_cat, contrast(outcomecontrast(r))
Contrasts of predictive margins                         Number of obs = 840
Model VCE: OIM
Expression: Pr(mode), predict()
```

	df	chi2	P>chi2
_outcome#cost_cat@mode			
(Train vs Air) (joint) Air	2	14.12	0.0009
(Train vs Air) (joint) Train	2	9.73	0.0077
(Train vs Air) (joint) Bus	2	3.96	0.1379
(Train vs Air) (joint) Car	2	1.13	0.5683
(Bus vs Air) (joint) Air	2	14.88	0.0006
(Bus vs Air) (joint) Train	2	0.88	0.6440
(Bus vs Air) (joint) Bus	2	10.42	0.0055
(Bus vs Air) (joint) Car	2	1.23	0.5401
(Car vs Air) (joint) Air	2	15.06	0.0005
(Car vs Air) (joint) Train	2	1.47	0.4793
(Car vs Air) (joint) Bus	2	3.30	0.1918
(Car vs Air) (joint) Car	2	12.59	0.0018
Joint	14	1112.36	0.0000

The first line in the output matches the joint test reported in our previous `margins` command. The interpretations of the remaining rows are similar. For instance, in the second row, `(train vs air) (joint) train`, we test whether the effects of the cost of `train` travel on the probability of selecting `train` travel are the same as the effects of the cost of `train` travel on the probability of selecting `air` travel.

Again, we can list the `r(L)` matrix to see how the contrasts for each of these joint tests were formulated.

```
. matrix list r(L)
```

(*output omitted*)

We restore the `margins` estimation results and run `test` using the formulation of the contrasts we saw in `r(L)`.

```
. estimates restore ourmargins
(results ourmargins are active now)
. test (   1._outcome#1.mode#1.cost_cat - 1._outcome#1.mode#2.cost_cat
>        = 2._outcome#1.mode#1.cost_cat - 2._outcome#1.mode#2.cost_cat )
>      (   1._outcome#1.mode#1.cost_cat - 1._outcome#1.mode#3.cost_cat
>        = 2._outcome#1.mode#1.cost_cat - 2._outcome#1.mode#3.cost_cat )
 ( 1)  1bn._outcome#1bn.mode#1bn.cost_cat - 1bn._outcome#1bn.mode#2.cost_cat -
       2._outcome#1bn.mode#1bn.cost_cat + 2._outcome#1bn.mode#2.cost_cat = 0
 ( 2)  1bn._outcome#1bn.mode#1bn.cost_cat - 1bn._outcome#1bn.mode#3.cost_cat -
       2._outcome#1bn.mode#1bn.cost_cat + 2._outcome#1bn.mode#3.cost_cat = 0

           chi2(  2) =   14.12
         Prob > chi2 =    0.0009
```

We have duplicated the first row of the output from `margins`. Its interpretation is now clear. It is a test of the null hypothesis that for the alternative `air` (which is `mode` = 1), differences in predicted probabilities across levels of `cost_cat` are the same for the outcome `train` as they are for the outcome `air`. We say "for the alternative `air`", meaning that the observations corresponding to the alternative `air` are the observations where `cost_cat` is set to 1, 2, or 3, and predicted probabilities are calculated for these values. At the observations corresponding to other alternatives, `cost_cat` is kept at its observed values. We could set `cost_cat` to its mean (or median, etc.) at these other alternatives using the option `at((mean) cost_cat)`. See [R] **margins**.

We can follow the same steps after using the `contrast(alternativecontrast(op))` option with `margins` to be sure we understand what contrasts are being tested there.

Graphing margins results

After any `margins` command, you can use `marginsplot` to create a graph of the estimated probabilities or contrasts. See [R] **marginsplot** for information on using this command.

▷ Example 4: marginsplot

Here we give an example of using `marginsplot` after `margins` to graph the expected probabilities of selecting four outcomes across a range of values of a continuous case-specific variable. The model was fit using `cmmprobit`, but this example is perfectly general and works after any command that supports `margins`.

We continue with the data from example 1 in [CM] **cmmprobit** and use the model fit in the previous example.

We want to see how income affects the choices, controlling for travel cost and terminal time. The range of `income` is 2 to 72, and we call `margins` with an `at()` option that calculates probabilities at many points over the full range of income.

```
. margins, at(income=(2 5(5)70 72))
Predictive margins                                      Number of obs = 840
Model VCE: OIM
Expression: Pr(mode), predict()
1._at:  income =  2
  (output omitted)
16._at: income = 72
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_outcome#_at						
Air# 1	.1721153	.0456076	3.77	0.000	.0827261	.2615045
Air# 2	.1827666	.0443583	4.12	0.000	.0958259	.2697072
(output omitted)						
Car#15	.4052022	.0722242	5.61	0.000	.2636454	.5467589
Car#16	.4100206	.0758821	5.40	0.000	.2612945	.5587467

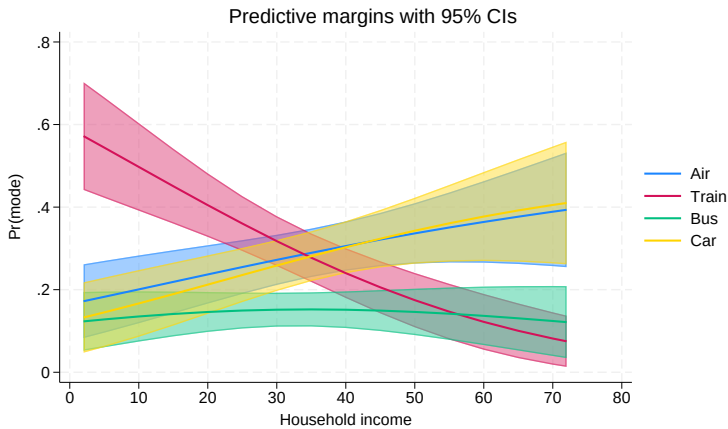
We could simply type

```
. marginsplot
```

to visualize the results.

Here we include the option `recast(line)` to smooth the plotting of the lines and the option `recastci(rarea)` to make the confidence intervals curves with shaded fill. The option `ciopts(color(%50))` makes the fill of the confidence intervals semitransparent.

```
. marginsplot, xlabel(0(10)80) recast(line) recastci(rarea)
> ciopts(color(%50)) plotopts(lwidth(medthick))
Variables that uniquely identify margins: income _outcome
```



From the graph, we see that the expected probability of choosing air travel increases with increasing income. The probability of choosing car travel also increases with increasing income. In fact, its probability is almost the same as the probability for air travel at all values of income. The probability of choosing bus travel changes little by income. The probability of choosing train travel has the biggest change over the range of income. At `income` = 2, the expected probability of choosing train travel is 54%. At `income` = 72, the expected probability of choosing train travel is only 9%.

◁

For more examples of `marginsplot` after CM commands, see [CM] **Intro 1**.

Stored results

In addition to the results shown in [R] **margins**, `margins` after `cm` estimators stores the following in `r()`:

Scalars
- `r(k_alt)` number of levels of alternatives variable

Macros
- `r(altvar)` name of alternatives variable
- `r(alt#)` #th level of alternatives variable

Matrices
- `r(altvals)` vector containing levels of alternatives variable

`margins` with the `post` option also stores the following in `e()`:

Scalars
- `e(k_alt)` number of levels of alternatives variable

Macros
- `e(altvar)` name of alternatives variable
- `e(alt#)` #th level of alternatives variable

Matrices
- `e(altvals)` vector containing levels of alternatives variable

Also see

[R] **contrast** — Contrasts and linear hypothesis tests after estimation

[R] **margins, contrast** — Contrasts of margins

[R] **margins, pwcompare** — Pairwise comparisons of margins

[R] **margins postestimation** — Postestimation tools for margins

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

Title

nlogit — Nested logit regression

Description Quick start Menu Syntax
Options Remarks and examples Stored results Methods and formulas
References Also see

Description

`nlogit` performs full information maximum-likelihood estimation for nested logit models. These models relax the assumption of independently distributed errors and the independence of irrelevant alternatives inherent in conditional and multinomial logit models by clustering similar alternatives into nests.

By default, `nlogit` uses a parameterization that is consistent with a random utility model (RUM). Before version 10 of Stata, a nonnormalized version of the nested logit model was fit, which you can request by specifying the `nonnormalized` option.

You must use `nlogitgen` to generate a new categorical variable to specify the branches of the nested logit tree before calling `nlogit`.

Quick start

Create variables identifying alternatives at higher levels

For a three-level nesting structure, create `l2alt` identifying four alternatives at level two based on the variable `balt`, which identifies eight bottom-level alternatives

```
nlogitgen l2alt = balt(l2alt1: balt1 | balt2, l2alt2: balt3 | balt4,  ///
     l2alt3: balt5 | balt6, l2alt4: balt7 | balt8)
```

Same as above, defining `l2alt` from values of `balt` rather than from value labels

```
nlogitgen l2alt = balt(l2alt1: 1|2, l2alt2: 3|4, l2alt3: 5|6,  ///
     l2alt4: 7|8)
```

Create `talt` identifying top-level alternatives based on the four alternatives in `l2alt`

```
nlogitgen talt = l2alt(talt1: l2alt1 | l2alt2, talt2: l2alt3 | l2alt4)
```

Examine tree structure

Display three-level nesting structure

```
nlogittree balt l2alt talt
```

Also report choice frequencies of indicator `chosen` for each bottom-level alternative

```
nlogittree balt l2alt talt, choice(chosen)
```

Identify potentially problematic observations before fitting the model by specifying case identifier `casevar`

```
nlogittree balt l2alt talt, choice(chosen) case(casevar)
```

Same as above, and generate prob indicating problematic observations

```
nlogittree balt l2alt talt, choice(chosen) case(casevar) generate(prob)
```

Fit nested logit model

Three-level model with alternative-specific covariate x1, case-specific covariate x2 modeling top-level choice, and case-specific covariate x3 modeling level-two choice

```
nlogit chosen x1 || talt: x2 || l2alt: x3 || balt:, case(casevar)
```

Same as above, but do not estimate intercepts for bottom-level alternatives

```
nlogit chosen x1 || talt: x2 || l2alt: x3 || balt:, noconstant  ///
     case(casevar)
```

Same as above, but estimate intercepts for top-level alternatives

```
nlogit chosen x1 || talt: x2, estconst || l2alt: x3  ///
     || balt:, noconstant case(casevar)
```

Specify the base alternative at each level

```
nlogit chosen x1 || talt: x2, base(talt2) || l2alt: x3, base(l2alt4)  ///
     || balt:, base(balt8) case(casevar)
```

Use nonnormalized parameterization

```
nlogit chosen x1 || talt: x2 || l2alt: x3 || balt:, case(casevar)  ///
     nonnormalized
```

Menu

nlogit

Statistics > Choice models > Nested logit model > Nested logit model

nlogitgen

Statistics > Choice models > Nested logit model > Setup for nested logit model

nlogittree

Statistics > Choice models > Nested logit model > Display nested logit tree structure

Syntax

Nested logit regression

```
nlogit depvar [indepvars] [if] [in] [weight] [|| lev1_equation
    [|| lev2_equation ...]] || altvar: [byaltvarlist], case(varname) [nlogit_options]
```

where the syntax of *lev#_equation* is

```
altvar: [byaltvarlist] [, base(# | lbl) estconst]
```

Create variable based on specification of branches

```
nlogitgen newaltvar = altvar (branchlist) [, [no]log]
```

where *branchlist* is

```
branch, branch [, branch ...]
```

and *branch* is

```
[label:] alternative [| alternative [| alternative ...]]
```

Display tree structure

```
nlogittree altvarlist [if] [in] [weight] [, nlogittree_options]
```

nlogit_options	Description
Model	
* `case(`*varname*`)`	use *varname* to identify cases
`base(`*#* | *lbl*`)`	use the specified level or label of *altvar* as the base alternative for the bottom level
`noconstant`	suppress the constant terms for the bottom-level alternatives
`nonnormalized`	use the nonnormalized parameterization
`altwise`	use alternativewise deletion instead of casewise deletion
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `robust`, `cluster` *clustvar*, `bootstrap`, or `jackknife`
Reporting	
`level(`*#*`)`	set confidence level; default is `level(95)`
`notree`	suppress display of tree-structure output; see also `nolabel` and `nobranches`
`nocnsreport`	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

* `case(`*varname*`)` is required.

nlogittree_options	Description
Main	
`choice(`*depvar*`)`	use *depvar* as the choice indicator variable
`case(`*varname*`)`	use *varname* to identify cases
`generate(`*newvar*`)`	create *newvar* to identify invalid observations
`nolabel`	suppress the value labels in tree-structure output
`nobranches`	suppress drawing branches in the tree-structure output

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

`bootstrap`, `by`, `collect`, `fp`, `jackknife`, and `statsby` are allowed; see [U] **11.1.10 Prefix commands**.

Weights are not allowed with the `bootstrap` prefix; see [R] **bootstrap**.

`fweights`, `iweights`, and `pweights` are allowed with `nlogit`, and `fweights` are allowed with `nlogittree`; see [U] **11.1.6 weight**. Weights for `nlogit` must be constant within case.

`collinear` does not appear in the dialog box.

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

Options

Specification and options for lev#_equation

altvar is a variable identifying alternatives at this level of the hierarchy.

byaltvarlist specifies the variables to be used to compute the by-alternative regression coefficients for that level. For each variable specified in the variable list, there will be one regression coefficient for each alternative of that level of the hierarchy. If the variable is constant across each alternative (a case-specific variable), the regression coefficient associated with the base alternative is not identifiable. These regression coefficients are labeled as (base) in the regression table. If the variable varies among the alternatives, a regression coefficient is estimated for each alternative.

`base(`*#* | *lbl*`)` can be specified in each level equation where it identifies the base alternative to be used at that level. The default is the alternative that has the highest frequency.

If `vce(bootstrap)` or `vce(jackknife)` is specified, you must specify the base alternative for each level that has a *byaltvarlist* or whether the constants will be estimated. Doing so ensures that the same model is fit with each call to `nlogit`.

`estconst` applies to all the level equations except the bottom-level equation. Specifying `estconst` requests that constants for each alternative (except the base alternative) be estimated. By default, no constant is estimated at these levels. Constants can be estimated in only one level of the tree hierarchy. If you specify `estconst` for one of the level equations, you must specify `noconstant` for the bottom-level equation.

Options for nlogit

Model

`case(`*varname*`)` specifies the variable that identifies each case. `case()` is required.

`base(`*#* | *lbl*`)` can be specified in each level equation where it identifies the base alternative to be used at that level. The default is the alternative that has the highest frequency.

If `vce(bootstrap)` or `vce(jackknife)` is specified, you must specify the base alternative for each level that has a *byaltvarlist* or whether the constants will be estimated. Doing so ensures that the same model is fit with each call to `nlogit`.

`noconstant` applies only to the equation defining the bottom level of the hierarchy. By default, constants are estimated for each alternative of *altvar*, less the base alternative. To suppress the constant terms for this level, specify `noconstant`. If you do not specify `noconstant`, you cannot specify `estconst` for the higher-level equations.

`nonnormalized` requests a nonnormalized parameterization of the model that does not scale the inclusive values by the degree of dissimilarity of the alternatives within each nest. Use this option to replicate results from older versions of Stata. The default is to use the RUM–consistent parameterization.

`altwise` specifies that alternativewise deletion be used when omitting observations because of missing values in your variables. The default is to use casewise deletion; that is, the entire group of observations making up a case is omitted if any missing values are encountered. This option does not apply to observations that are excluded by the `if` or `in` qualifier or the `by` prefix; these observations are always handled alternativewise regardless of whether `altwise` is specified.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

The inclusive-valued/dissimilarity parameters are parameterized as `ml` ancillary parameters. They are labeled as [*alternative*`_tau`]`_const`, where *alternative* is one of the alternatives defining a branch in the tree. To constrain the inclusive-valued/dissimilarity parameter for alternative `a1` to be, say, equal to alternative `a2`, you would use the following syntax:

```
. constraint 1 [a1_tau]_cons = [a2_tau]_cons
. nlogit ..., constraints(1)
```

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***.

If `vce(robust)` or `vce(cluster` *clustvar*`)` is specified, the likelihood-ratio test for the independence of irrelevant alternatives (IIA) is not computed.

Reporting

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

`notree` specifies that the tree structure of the nested logit model not be displayed. See also `nolabel` and `nobranches` below for when `notree` is not specified. cmd:notree may not be specified on 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**.

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 `technique(bhhh)` option is not allowed. The default optimization technique is `technique(bfgs)`.

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

`collinear` prevents collinear variables from being omitted. Use this option when you know that you have collinear variables and you are applying `constraints()` to handle the rank reduction. See [R] **Estimation options** for details on using `collinear` with `constraints()`.

`nlogit` will not allow you to specify an independent variable in more than one level equation. Specifying the `collinear` option will allow execution to proceed in this case, but it is your responsibility to ensure that the parameters are identified.

Specification and options for nlogitgen

newaltvar and *altvar* are variables identifying alternatives at each level of the hierarchy.

label defines a label to associate with the branch. If no label is given, a numeric value is used.

alternative specifies an alternative, of *altvar* specified in the syntax, to be included in the branch. It is either a numeric value or the label associated with that value. An example of `nlogitgen` is

```
. nlogitgen type = restaurant(fast: 1 | 2,
> family: CafeEccell | LosNortenos | WingsNmore, fancy: 6 | 7)
```

`log` and `nolog` specify whether to display the iteration log. The iteration log is displayed by default unless you used `set iterlog off` to suppress it; see `set iterlog` in [R] ***set iter***.

Specification and options for nlogittree

Main

altvarlist is a list of alternative variables that define the tree hierarchy. The first variable must define bottom-level alternatives, and the order continues to the variable defining the top-level alternatives.

`choice(`*depvar*`)` defines the choice indicator variable and forces `nlogittree` to compute and display choice frequencies for each bottom-level alternative.

`case(`*varname*`)` specifies the variable that identifies each case. When both `case()` and `choice()` are specified, `nlogittree` executes diagnostics on the tree structure and identifies observations that will cause `nlogit` to terminate execution or drop observations.

`generate(`*newvar*`)` generates a new indicator variable, *newvar*, that is equal to 1 for invalid observations. This option requires that both `choice()` and `case()` are also specified.

`nolabel` forces `nlogittree` to suppress value labels in tree-structure output.

`nobranches` forces `nlogittree` to suppress drawing branches in the tree-structure output.

Remarks and examples

Remarks are presented under the following headings:

Introduction
Data setup and the tree structure
Estimation
Testing for the IIA
Nonnormalized model

Introduction

`nlogit` performs full information maximum-likelihood estimation for nested logit models. These models relax the assumption of independently distributed errors and the IIA inherent in conditional and multinomial logit models by clustering similar alternatives into nests. The nested logit model is a direct generalization of McFadden's choice model fit by `cmclogit`. You may want to read [CM] **Intro 5**, [CM] **Intro 8**, and [CM] **cmclogit** before continuing.

Although `nlogit` fits choice models, it is not a `cm` command, and you do not have to `cmset` your data. `nlogit` has its own data requirements for nested alternatives, which are detailed in the following examples.

By default, `nlogit` uses a RUM parameterization. Before version 10 of Stata, a nonnormalized version of the nested logit model was fit, which you can request by specifying the `nonnormalized` option. We recommend using the RUM parameterization for new projects because it is based on a sound model of consumer behavior.

McFadden (1977, 1981) showed how this model can be derived from a rational choice framework. Amemiya (1985, chap. 9) contains a nice discussion of how this model can be derived under the assumption of utility maximization. Hensher, Rose, and Greene (2015) provide a lucid introduction to choice models, including nested logit.

Throughout this entry, we consider a model of restaurant choice. We begin by introducing the data.

▷ Example 1: Families choosing a restaurant

We have fictional data on 300 families and their choice of seven local restaurants. Freebirds and Mama's Pizza are fast food restaurants; Café Eccell, Los Norteños, and Wings 'N More are family restaurants; and Christopher's and Mad Cows are fancy restaurants. We want to model the decision of where to eat as a function of household income (`income`, in thousands of dollars), the number of children in the household (`kids`), the rating of the restaurant according to a local restaurant guide (`rating`, coded 0–5), the average meal cost per person (`cost`), and the distance between the household and the restaurant (`distance`, in miles). `income` and `kids` are attributes of the family, `rating` is an attribute of the alternative (the restaurant), and `cost` and `distance` are attributes of the alternative as perceived by the families—that is, each family has its own cost and distance for each restaurant.

We begin by loading the data and listing some of the variables for the first three families:

```
. use https://www.stata-press.com/data/r18/restaurant
. describe
Contains data from https://www.stata-press.com/data/r18/restaurant.dta
 Observations:         2,100
    Variables:             8                  2 Dec 2022 15:07
-------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
-------------------------------------------------------------------------------
family_id       int     %9.0g                 family ID
restaurant      byte    %12.0g     names      choices of restaurants
income          int     %9.0g                 household income
cost            float   %9.0g                 average meal cost per person
kids            byte    %9.0g                 number of kids in the household
rating          byte    %9.0g                 ratings in local restaurant guide
distance        float   %9.0g                 distance between home and
                                                restaurant
chosen          byte    %9.0g                 0 no 1 yes
-------------------------------------------------------------------------------
Sorted by: family_id
```

```
. list family_id restaurant chosen kids rating distance in 1/21, sepby(fam)
> abbrev(10)
```

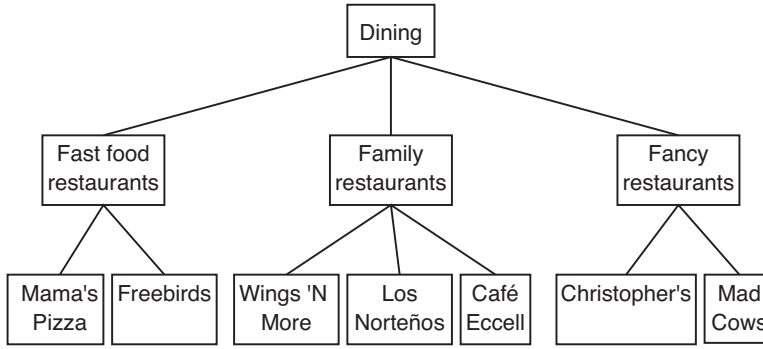
	family_id	restaurant	chosen	kids	rating	distance
1.	1	Freebirds	1	1	0	1.245553
2.	1	MamasPizza	0	1	1	2.82493
3.	1	CafeEccell	0	1	2	4.21293
4.	1	LosNortenos	0	1	3	4.167634
5.	1	WingsNmore	0	1	2	6.330531
6.	1	Christophers	0	1	4	10.19829
7.	1	MadCows	0	1	5	5.601388
8.	2	Freebirds	0	3	0	4.162657
9.	2	MamasPizza	0	3	1	2.865081
10.	2	CafeEccell	0	3	2	5.337799
11.	2	LosNortenos	1	3	3	4.282864
12.	2	WingsNmore	0	3	2	8.133914
13.	2	Christophers	0	3	4	8.664631
14.	2	MadCows	0	3	5	9.119597
15.	3	Freebirds	1	3	0	2.112586
16.	3	MamasPizza	0	3	1	2.215329
17.	3	CafeEccell	0	3	2	6.978715
18.	3	LosNortenos	0	3	3	5.117877
19.	3	WingsNmore	0	3	2	5.312941
20.	3	Christophers	0	3	4	9.551273
21.	3	MadCows	0	3	5	5.539806

Because each family chose among seven restaurants, there are 7 observations in the dataset for each family. The variable `chosen` is coded 0/1, with 1 indicating the chosen restaurant and 0 otherwise. ◁

We could fit a conditional logit model to our data. Because `income` and `kids` are constant within each family, we would use the `cmclogit` command instead of `clogit`. However, the conditional logit may be inappropriate. That model assumes that the random errors are independent, and as a result, it forces the odds ratio of any two alternatives to be independent of the other alternatives, a property known as the IIA. We will discuss the IIA assumption in more detail later.

Assuming that unobserved shocks influencing a decision maker's attitude toward one alternative have no effect on his or her attitude toward the other alternatives may seem innocuous, but often this assumption is too restrictive. Suppose that when a family was deciding which restaurant to visit, they were pressed for time because of plans to attend a movie later. The unobserved shock (being in a hurry) would raise the likelihood that the family goes to either fast food restaurant (Freebirds or Mama's Pizza). Similarly, another family might be choosing a restaurant to celebrate a birthday and therefore be inclined to attend a fancy restaurant (Christopher's or Mad Cows).

Nested logit models relax the independence assumption and allow us to group alternatives for which unobserved shocks may have concomitant effects. Here we suspect that restaurants should be grouped by type (fast, family, or fancy). The tree structure of a family's decision about where to eat might look like this:



At the bottom of the tree are the individual restaurants, indicating that there are some random shocks that affect a family's decision to eat at each restaurant independently. Above the restaurants are the three types of restaurants, indicating that other random shocks affect the type of restaurant chosen. As is customary when drawing nested logit trees, at the top level is one box, representing the family making the decision.

We use the following terms to describe nested logit models.

level, or decision level, is the level or stage at which a decision is made. The example above has only two levels. In the first level, a type of restaurant is chosen—fast food, family, or fancy—and in the second level, a specific restaurant is chosen.

bottom level is the level where the final decision is made. In our example, this is when we choose a specific restaurant.

alternative set is the set of all possible alternatives at any given decision level.

bottom alternative set is the set of all possible alternatives at the bottom level. This concept is often referred to as the choice set in the economics-choice literature. In our example, the bottom alternative set is all seven of the specific restaurants.

alternative is a specific alternative within an alternative set. In the first level of our example, "fast food" is an alternative. In the second or bottom level, "Mad Cows" is an alternative. Not all alternatives within an alternative set are available to someone making a choice at a specific stage, only those that are nested within all higher-level decisions.

chosen alternative is the alternative from an alternative set that we observe someone having chosen.

❑ Technical note

Although the trees in nested logit analysis are often interpreted as implying that the highest-level decisions are made first, followed by decisions at lower levels, and finally the decision among alternatives at the bottom level, no such temporal ordering is implied. See Hensher, Rose, and Greene (2015, chap. 14). In our example, we are not assuming that families first choose whether to attend a fast, family, or fancy restaurant and then choose the particular restaurant; we assume merely that they choose one of the seven restaurants.

❑

Data setup and the tree structure

To fit a nested logit model, you must first create a variable that defines the structure of your nested logit tree.

▷ Example 2: Identifying the first-level set of alternatives

To run `nlogit`, we need to generate a categorical variable that identifies the first-level set of alternatives: fast food, family restaurants, or fancy restaurants. We can do so easily by using `nlogitgen`.

```
. nlogitgen type = restaurant(fast: Freebirds | MamasPizza,
> family: CafeEccell | LosNortenos| WingsNmore, fancy: Christophers | MadCows)
New variable type is generated with 3 groups
label list lb_type
lb_type:
           1 fast
           2 family
           3 fancy
. nlogittree restaurant type, choice(chosen)
Tree structure specified for the nested logit model
 type    N      restaurant    N   k
──────────────────────────────────────
 fast   600 ─┬─ Freebirds    300  12
             └─ MamasPizza   300  15
 family 900 ─┬─ CafeEccell   300  78
             ├─ LosNortenos  300  75
             └─ WingsNmore   300  69
 fancy  600 ─┬─ Christophers 300  27
             └─ MadCows      300  24
──────────────────────────────────────
                    Total  2100 300
k = number of times alternative is chosen
N = number of observations at each level
```

The new categorical variable is `type`, which takes on value 1 (fast) if `restaurant` is Freebirds or Mama's Pizza; value 2 (family) if `restaurant` is Café Eccell, Los Norteños, or Wings 'N More; and value 3 (fancy) otherwise. `nlogittree` displays the tree structure.

◁

❑ Technical note

We could also use values instead of value labels of `restaurant` in `nlogitgen`. Value labels are optional, and the default value labels for `type` are `type1`, `type2`, and `type3`. The vertical bar is also optional.

```
. use https://www.stata-press.com/data/r18/restaurant, clear
. nlogitgen type = restaurant(1 2, 3 4 5, 6 7)
New variable type is generated with 3 groups
label list lb_type
lb_type:
           1 type1
           2 type2
           3 type3
```

```
. nlogittree restaurant type
Tree structure specified for the nested logit model
 type    N      restaurant     N
──────────────────────────────────
 type1 600 ─┬─ Freebirds     300
            └─ MamasPizza    300
 type2 900 ─┬─ CafeEccell    300
            ├─ LosNortenos   300
            └─ WingsNmore    300
 type3 600 ─┬─ Christophers  300
            └─ MadCows       300
──────────────────────────────────
                      Total  2100
N = number of observations at each level
```

❑

In our dataset, every family was able to choose among all seven restaurants. However, in other applications some decision makers may not have been able to choose among all possible alternatives. For example, two cases may have choice hierarchies of

```
      case 1                          case 2
type        restaurant          type        restaurant
──────────────────────          ──────────────────────
fast   ─┬─ Freebirds            fast   ─┬─ Freebirds
        └─ MamasPizza                   └─ MamasPizza
family ─┬─ CafeEccell           family ─┬─ LosNortenos
        ├─ LosNortenos                  └─ WingsNmore
        └─ WingsNmore
fancy  ─┬─ Christophers         fancy  ── Christophers
        └─ MadCows
```

where the second case does not have the restaurant alternatives Café Eccell or Mad Cows available to them. The only restriction is that the relationships between higher- and lower-level alternative sets be the same for all decision makers. In this two-level example, Freebirds and Mama's Pizza are classified as fast food restaurants for both cases; Café Eccell, Los Norteños, and Wings 'N More are family restaurants; and Christopher's and Mad Cows are fancy restaurants. `nlogit` requires only that hierarchy be maintained for all cases.

Estimation

▷ Example 3: Fitting the model

With our `type` variable created that defines the three types of restaurants, we can now examine how the alternative-specific attributes (`cost`, `rating`, and `distance`) apply to the bottom alternative set (the seven restaurants) and how family-specific attributes (`income` and `kid`) apply to the alternative set at the first decision level (the three types of restaurants).

```
. use https://www.stata-press.com/data/r18/restaurant, clear
. quietly nlogitgen type = restaurant(fast: Freebirds | MamasPizza,
> family: CafeEccell | LosNortenos| WingsNmore, fancy: Christophers | MadCows)
. nlogit chosen cost rating distance || type: income kids, base(family) ||
> restaurant:, noconstant case(family_id)
Tree structure specified for the nested logit model
 type     N       restaurant     N   k
-----------------------------------------
 fast    600 ---- Freebirds     300  12
              `-- MamasPizza    300  15
 family  900 ---- CafeEccell    300  78
              |-- LosNortenos   300  75
              `-- WingsNmore    300  69
 fancy   600 ---- Christophers  300  27
              `-- MadCows       300  24
-----------------------------------------
                        Total  2100 300
k = number of times alternative is chosen
N = number of observations at each level
Iteration 0:  Log likelihood = -541.93581
  (output omitted)
Iteration 17: Log likelihood = -485.47331
RUM-consistent nested logit regression          Number of obs      =      2,100
Case variable: family_id                        Number of cases    =        300
Alternative variable: restaurant                Alts per case: min =          7
                                                               avg =        7.0
                                                               max =          7
                                                  Wald chi2(7)    =      46.71
Log likelihood = -485.47331                       Prob > chi2     =     0.0000
------------------------------------------------------------------------------
      chosen | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
restaurant   |
        cost |  -.1843847   .0933975    -1.97   0.048    -.3674404   -.0013289
      rating |    .463694   .3264935     1.42   0.156    -.1762215    1.10361
    distance |  -.3797474   .1003828    -3.78   0.000    -.5764941   -.1830007
-------------+----------------------------------------------------------------
type equations
-------------+----------------------------------------------------------------
fast         |
      income |  -.0266038   .0117306    -2.27   0.023    -.0495952   -.0036123
        kids |  -.0872584   .1385026    -0.63   0.529    -.3587184    .1842016
-------------+----------------------------------------------------------------
family       |
      income |          0  (base)
        kids |          0  (base)
-------------+----------------------------------------------------------------
fancy        |
      income |   .0461827   .0090936     5.08   0.000     .0283595    .0640059
        kids |  -.3959413   .1220356    -3.24   0.001    -.6351267   -.1567559
-------------+----------------------------------------------------------------
dissimilarity parameters
-------------+----------------------------------------------------------------
/type        |
    fast_tau |   1.712878    1.48685                     -1.201295    4.627051
  family_tau |   2.505113   .9646351                       .614463    4.395763
   fancy_tau |   4.099844   2.810123                     -1.407896    9.607583
------------------------------------------------------------------------------
LR test for IIA (tau=1): chi2(3) = 6.87                  Prob > chi2 = 0.0762
```

First, let's examine how we called `nlogit`. The delimiters (||) separate equations. The first equation specifies the dependent variable, `chosen`, and three alternative-specific variables, `cost`, `rating`, and `distance`. We refer to these variables as alternative specific because they vary among the bottom-level alternatives, the restaurants. We obtain one parameter estimate for each variable. These estimates are listed in the equation subtable labeled `restaurant`.

For the second equation, we specify the `type` variable. It identifies the first-level alternatives, the restaurant types. Following the colon after `type`, we specify two case-specific variables, `income` and `kids`. Here we obtain a parameter estimate for each variable for each alternative at this level. That is why we call these variable lists "by-alternative" variables. Because `income` and `kids` do not vary within each case, to identify the model, one must specify the alternative set of parameters as zero. We specified the `base(family)` option with this equation to restrict the parameters for the `family` alternative.

The variable identifying the bottom-level alternatives, `restaurant`, is specified after the second equation delimiter. We do not specify any variables after the colon delimiter at this level. Had we specified variables here, we would have obtained an estimate for each variable in each equation. As we will see below, these variables parameterize the constant term in the utility equation for each bottom-level alternative. The `noconstant` option suppresses bottom-level alternative-specific constant terms.

Near the bottom of the output are the dissimilarity parameters, which measure the degree of correlation of random shocks within each of the three types of restaurants. Dissimilarity parameters greater than one imply that the model is inconsistent with random utility maximization; Hensher, Rose, and Greene (2005, sec. 13.6) discuss this in detail. We will ignore the fact that all our dissimilarity parameters exceed one.

The conditional logit model is a special case of nested logit in which all the dissimilarity parameters are equal to one. At the bottom of the output, we find a likelihood-ratio test of this hypothesis. Here we have mixed evidence of the null hypothesis that all the parameters are one. Equivalently, the property known as the IIA imposed by the conditional logit model holds if and only if all dissimilarity parameters are equal to one. We discuss the IIA in more detail now.

◁

Testing for the IIA

The IIA is a property of the multinomial and conditional logit models that forces the odds of choosing one alternative over another to be independent of the other alternatives. For simplicity, suppose that a family was choosing only between Freebirds and Mama's Pizza, and the family was equally likely to choose either of the restaurants. The probability of going to each restaurant is 50%. Now suppose that Bill's Burritos opens up next door to Freebirds, which is also a burrito restaurant. If the IIA holds, then the probability of going to each restaurant must now be 33.33% so that the family remains equally likely to go to Mama's Pizza or Freebirds.

The IIA may sometimes be a plausible assumption. However, a more likely scenario would be for the probability of going to Mama's Pizza to remain at 50% and the probabilities of going to Freebirds and Bill's Burritos to be 25% each, because the two restaurants are next door to each other and serve the same food. Nested logit analysis would allow us to relax the IIA assumption of conditional logit. We could group Bill's Burritos and Freebirds into one nest that encompasses all burrito restaurants and create a second nest for pizzerias.

The IIA is a consequence of assuming that the errors are independent and identically distributed (i.i.d.). Because the errors are i.i.d., they cannot contain any alternative-specific unobserved information, and therefore adding a new alternative cannot affect the relationship between a pair of existing alternatives.

In the previous example, we saw that a joint test that the dissimilarity parameters were equal to one is one way to test for IIA. However, that test required us to specify a tree for the nested logit model, and different specifications could lead to conflicting results of the test. Hausman and McFadden (1984) suggest that if part of the choice set truly is irrelevant with respect to the other alternatives, omitting that subset from the conditional logit model will not lead to inconsistent estimates. Therefore, Hausman's (1978) specification test can be used to test for IIA, and this test will not be sensitive to the tree structure we specify for a nested logit model.

▷ Example 4: Testing the IIA assumption

We want to test the IIA for the subset of family restaurants against the alternatives of fast food and fancy restaurants. To do so, we need to use Stata's `hausman` command; see [R] **hausman**.

We first run the estimation on the full bottom alternative set, store the results by using `estimates store`, and then run the estimation on the bottom alternative set, excluding the alternatives of family restaurants. We then run the `hausman` test.

```
. generate incFast = (type == 1) * income
. generate incFancy = (type == 3) * income
. generate kidFast = (type == 1) * kids
. generate kidFancy = (type == 3) * kids
. clogit chosen cost rating distance incFast incFancy kidFast kidFancy,
> group(family_id) nolog
Conditional (fixed-effects) logistic regression         Number of obs =  2,100
                                                        LR chi2(7)    = 189.73
                                                        Prob > chi2   = 0.0000
Log likelihood = -488.90834                             Pseudo R2     = 0.1625
```

chosen	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
cost	-.1367799	.0358479	-3.82	0.000	-.2070404	-.0665193
rating	.3066622	.1418291	2.16	0.031	.0286823	.584642
distance	-.1977505	.0471653	-4.19	0.000	-.2901927	-.1053082
incFast	-.0390183	.0094018	-4.15	0.000	-.0574455	-.0205911
incFancy	.0407053	.0080405	5.06	0.000	.0249462	.0564644
kidFast	-.2398757	.1063674	-2.26	0.024	-.448352	-.0313994
kidFancy	-.3893862	.1143797	-3.40	0.001	-.6135662	-.1652061

```
. estimates store fullset
. clogit chosen cost rating distance incFast kidFast if type != 2,
> group(family_id) nolog
note: 222 groups (888 obs) omitted because of all positive or
      all negative outcomes.
Conditional (fixed-effects) logistic regression         Number of obs =    312
                                                        LR chi2(5)    =  44.35
                                                        Prob > chi2   = 0.0000
Log likelihood = -85.955324                             Pseudo R2     = 0.2051
```

chosen	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
cost	-.0616621	.067852	-0.91	0.363	-.1946496	.0713254
rating	.1659001	.2832041	0.59	0.558	-.3891698	.72097
distance	-.244396	.0995056	-2.46	0.014	-.4394234	-.0493687
incFast	-.0737506	.0177444	-4.16	0.000	-.108529	-.0389721
kidFast	.4105386	.2137051	1.92	0.055	-.0083157	.8293928

```
. hausman . fullset
```

	Coefficients (b)	Coefficients (B)	(b-B) Difference	sqrt(diag(V_b-V_B)) Std. err.
	.	fullset		
cost	-.0616621	-.1367799	.0751178	.0576092
rating	.1659001	.3066622	-.1407621	.2451308
distance	-.244396	-.1977505	-.0466456	.0876173
incFast	-.0737506	-.0390183	-.0347323	.015049
kidFast	.4105386	-.2398757	.6504143	.1853533

```
                b = Consistent under H0 and Ha; obtained from clogit.
    B = Inconsistent under Ha, efficient under H0; obtained from clogit.
Test of H0: Difference in coefficients not systematic
    chi2(5) = (b-B)'[(V_b-V_B)^(-1)](b-B)
            =  10.70
Prob > chi2 = 0.0577
(V_b-V_B is not positive definite)
```

Similar to our findings in example 3, the results of the test of the IIA are mixed. We cannot reject the IIA at the commonly used 5% significance level, but we could at the 10% level. Substantively, a significant test result suggests that the odds of going to one of the fancy restaurants versus going to one of the fast food restaurants change if we include the family restaurants in the alternative set and that a nested logit specification may be warranted.

◁

Nonnormalized model

Previous versions of Stata fit a nonnormalized nested logit model that is available via the `nonnormalized` option. The nonnormalized version is presented in, for example, Greene (2018, 837–839). Here we outline the differences between the nonnormalized model and the random utility parameterization of the model. Our discussion follows Heiss (2002) and assumes the nested logit tree has two levels, with M alternatives at the upper level and a total of J alternatives at the bottom level.

In a RUM framework, by consuming alternative j, decision maker i obtains utility

$$U_{ij} = V_{ij} + \epsilon_{ij} = \alpha_j + \mathbf{x}_{ij}\boldsymbol{\beta}_j + \mathbf{z}_i\boldsymbol{\gamma}_j + \epsilon_{ij}$$

where V_{ij} is the deterministic part of utility and ϵ_{ij} is the random part. $\mathbf{x}_{ij}$ are alternative-specific variables and $\mathbf{z}_i$ are case-specific variables. The set of errors $\epsilon_{i1}, \ldots, \epsilon_{iJ}$ are assumed to follow the generalized extreme-value (GEV) distribution, which is a generalization of the type 1 extreme-value distribution that allows for alternatives within nests of the tree structure to be correlated. Let ρ_m denote the correlation in nest m, and define the dissimilarity parameter $\tau_m = \sqrt{1-\rho_m}$. $\tau_m = 0$ implies that the alternatives in nest m are perfectly correlated, whereas $\tau_m = 1$ implies independence.

The *inclusive value* for the mth nest corresponds to the expected value of the utility that decision maker i obtains by consuming an alternative in nest m. Denote this value by IV_m,

$$\text{IV}_m = \ln \sum_{j \in B_m} \exp(V_k/\tau_m) \tag{1}$$

where B_m denotes the set of alternatives in nest m. Given the inclusive values, we can show that the probability that random-utility–maximizing decision maker i chooses alternative j is

$$\Pr_j = \frac{\exp\{V_j/\tau(j)\}}{\exp\{\text{IV}(j)\}} \frac{\exp\{\tau(j)\text{IV}(j)\}}{\sum_m \exp(\tau_m \text{IV}_m)}$$

where $\tau(j)$ and $\text{IV}(j)$ are the dissimilarity parameter and inclusive value for the nest in which alternative j lies.

In contrast, for the nonnormalized model, we have a latent variable

$$\widetilde{V}_{i,j} = \widetilde{\alpha}_j + \mathbf{x}_{i,j}\widetilde{\boldsymbol{\beta}}_j + \mathbf{z}_i\widetilde{\boldsymbol{\gamma}}_j$$

and corresponding inclusive values

$$\widetilde{\text{IV}}_m = \ln \sum_{j \in B_m} \exp(\widetilde{V}_k) \tag{2}$$

The probability of choosing alternative j is

$$\Pr_j = \frac{\exp(\widetilde{V}_j)}{\exp\{\widetilde{\text{IV}}(j)\}} \frac{\exp\{\tau(j)\widetilde{\text{IV}}(j)\}}{\sum_m \exp(\tau_m \widetilde{\text{IV}}_m)}$$

Equations (1) and (2) represent the key difference between the random utility and the nonnormalized models. By scaling the V_{ij} within each nest, the RUM parameterization allows utilities to be compared across nests. Without the rescaling, utilities can be compared only for goods within the same nest. Moreover, adding a constant to each V_{ij} for consumer i will not affect the probabilities of the RUM, but adding a constant to each $\widetilde{V}_{ij}$ will affect the probabilities from the nonnormalized model. Decisions based on utility maximization can depend only on utility differences and not the scale or zero point of the utility function because utility is an ordinal concept, so the nonnormalized model cannot be consistent with utility maximization.

Heiss (2002) showed that the nonnormalized model can be consistent with a random utility parameterization in the special case where all the variables are specified in the bottom-level equation. Then multiplying the nonnormalized coefficients by the respective dissimilarity parameters results in the coefficients that are consistent with a RUM.

❑ Technical note

Degenerate nests occur when there is only one alternative in a branch of the tree hierarchy. The associated dissimilarity parameter of the RUM is not defined. The inclusive-valued parameter of the nonnormalized model will be identifiable if there are alternative-specific variables specified in (1) of the model specification (the *indepvars* in the model syntax). Numerically, you can skirt the issue of nonidentifiable/undefined parameters by setting constraints on them. For the RUM constraint, set the dissimilarity parameter to 1. See the description of `constraints()` in *Options* for details on setting constraints on the dissimilarity parameters.

❑

Stored results

`nlogit` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(N_case)`	number of cases
`e(N_ic)`	N for Bayesian information criterion (BIC)
`e(N_clust)`	number of clusters
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test
`e(k_alt)`	number of alternatives for bottom level
`e(k_alt`*j*`)`	number of alternatives for *j*th level
`e(k_indvars)`	number of independent variables
`e(k_ind2vars)`	number of by-alternative variables for bottom level
`e(k_ind2vars`*j*`)`	number of by-alternative variables for *j*th level
`e(df_m)`	model degrees of freedom
`e(df_c)`	`clogit` model degrees of freedom
`e(ll)`	log likelihood
`e(ll_c)`	`clogit` model log likelihood
`e(chi2)`	χ^2
`e(chi2_c)`	likelihood-ratio test for IIA
`e(p)`	p-value for model Wald test
`e(p_c)`	p-value for IIA test
`e(i_base)`	base index for bottom level
`e(i_base`*j*`)`	base index for *j*th level
`e(levels)`	number of levels
`e(alt_min)`	minimum number of alternatives
`e(alt_avg)`	average number of alternatives
`e(alt_max)`	maximum number of alternatives
`e(const)`	constant indicator for bottom level
`e(const`*j*`)`	constant indicator for *j*th level
`e(rum)`	`1` if RUM, `0` otherwise
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	`1` if converged, `0` otherwise

Macros

`e(cmd)`	`nlogit`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(indvars)`	name of independent variables
`e(ind2vars)`	by-alternative variables for bottom level
`e(ind2vars`*j*`)`	by-alternative variables for *j*th level
`e(case)`	variable defining cases
`e(altvar)`	alternative variable for bottom level
`e(altvar`*j*`)`	alternative variable for *j*th level
`e(alteqs)`	equation names for bottom level
`e(alteqs`*j*`)`	equation names for *j*th level
`e(alt`*i*`)`	*i*th alternative for bottom level
`e(alt`*j*`_`*i*`)`	*i*th alternative for *j*th level
`e(wtype)`	weight type
`e(wexp)`	weight expression
`e(marktype)`	`casewise` or `altwise`, type of markout
`e(key_N_ic)`	`cases`, key for N for Bayesian information criterion (BIC)
`e(title)`	title in estimation output
`e(clustvar)`	name of cluster variable
`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(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(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(k_altern)	number of alternatives at each level
e(k_branch*j*)	number of branches at each alternative of *j*th level
e(stats)	alternative statistics for bottom level
e(stats*j*)	alternative statistics for *j*th level
e(altidx*j*)	alternative indices for *j*th level
e(alt_ind2vars)	indicators for bottom level estimated by-alternative variable—e(k_alt)×e(k_ind2vars)
e(alt_ind2vars*j*)	indicators for *j*th level estimated by-alternative variable—e(k_alt*j*)×e(k_ind2vars*j*)
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:

Two-level nested logit model
Three-level nested logit model

Two-level nested logit model

Consider our two-level nested logit model for restaurant choice. We define $T = \{1, 2, 3\}$ to be the set of indices denoting the three restaurant types and $R_1 = \{1, 2\}$, $R_2 = \{3, 4, 5\}$, and $R_3 = \{6, 7\}$ to be the set of indices representing each restaurant within type $t \in T$. Let C_1 and C_2 be the random variables that represent the choices made for the first level, restaurant type, and second level, restaurant, of the hierarchy, where we observe the choices $C_1 = t, t \in T$, and $C_2 = j, j \in R_t$. Let $\mathbf{z}_t$ and $\mathbf{x}_{tj}$, for $t \in T$ and $j \in R_t$, refer to the row vectors of explanatory variables for the first-level alternatives and bottom-level alternatives for one case, respectively. We write the utilities (latent variables) as $U_{tj} = \mathbf{z}_t\boldsymbol{\alpha}_t + \mathbf{x}_{tj}\boldsymbol{\beta}_j + \epsilon_{tj} = \boldsymbol{\eta}_{tj} + \epsilon_{tj}$, where $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_j$ are column vectors and the ϵ_{tj} are random disturbances. When the $\mathbf{x}_{tj}$ are alternative specific, we can drop the indices from $\boldsymbol{\beta}$, where we estimate one coefficient for each alternative in R_t, $t \in T$. These variables are specified in the first equation of the nlogit syntax (see example 3).

When the random-utility framework is used to describe the choice behavior, the alternative that is chosen is the alternative that has the highest utility. Assume for our restaurant example that we choose restaurant type $t \in T$. For the RUM parameterization of `nlogit`, the conditional distribution of ϵ_{tj} given choice of restaurant type t is a multivariate version of Gumbel's extreme-value distribution,

$$F_{R|T}(\boldsymbol{\epsilon} \,|\, t) = \exp\left[-\left\{\sum_{m \in R_t} \exp(\epsilon_{tm}/\tau_t)\right\}^{\tau_t}\right] \tag{3}$$

where it has been shown that the ϵ_{tj}, $j \in R_t$, are exchangeable with correlation $1-\tau_t^2$, for $\tau_t \in (0,1]$ (Kotz and Nadarajah 2000). For example, the probability of choosing Christopher's, $j = 6$ given type $t = 3$, is

$$\begin{aligned}
\Pr(C_2 = 6 \,|\, C_1 = 3) &= \Pr\left(U_{36} - U_{37} > 0\right) \\
&= \Pr\left(\epsilon_{37} \le \epsilon_{36} + \eta_{36} - \eta_{37}\right) \\
&= \int_{-\infty}^{\infty} \left\{ \int_{-\infty}^{\epsilon_{36}+\eta_{36}-\eta_{37}} f_{R|T}\left(\epsilon_{36}, \epsilon_{37}\right) d\epsilon_{37} \right\} d\epsilon_{36}
\end{aligned}$$

where $f = \dfrac{\partial F}{\partial \epsilon_{36} \partial \epsilon_{37}}$ is the joint density function of $\boldsymbol{\epsilon}$ given t. U_{37} is the utility of eating at Mad Cows, the other fancy ($t = 3$) restaurant. Amemiya (1985) demonstrates that this integral evaluates to the logistic function

$$\begin{aligned}
\Pr(C_2 = 6 \,|\, C_1 = 3) &= \frac{\exp(\boldsymbol{\eta}_{36}/\tau_3)}{\exp(\boldsymbol{\eta}_{36}/\tau_3) + \exp(\boldsymbol{\eta}_{37}/\tau_3)} \\
&= \frac{\exp(\mathbf{x}_{36}\boldsymbol{\beta}_6/\tau_3)}{\exp(\mathbf{x}_{36}\boldsymbol{\beta}_6/\tau_3) + \exp(\mathbf{x}_{37}\boldsymbol{\beta}_7/\tau_3)}
\end{aligned}$$

and in general

$$\Pr(C_2 = j \,|\, C_1 = t) = \frac{\exp(\mathbf{x}_{tj}\boldsymbol{\beta}_j/\tau_t)}{\sum_{m \in R_t} \exp(\mathbf{x}_{tm}\boldsymbol{\beta}_m/\tau_t)} \tag{4}$$

Letting $\tau_t = 1$ in (3) reduces to the product of independent extreme-value distributions, and (4) reduces to the multinomial logistic function.

For the logistic function in (4), we scale the linear predictors by the dissimilarity parameters. Another formulation of the conditional probability of choosing alternative $j \in R_t$ given choice $t \in T$ is the logistic function without this normalization,

$$\Pr(C_2 = j \,|\, C_1 = t) = \frac{\exp(\mathbf{x}_{tj}\boldsymbol{\beta}_j)}{\sum_{m \in R_t} \exp(\mathbf{x}_{tm}\boldsymbol{\beta}_m)}$$

and this is what is used in `nlogit`'s nonnormalized parameterization.

Amemiya (1985) defines the general form for the joint distribution of the ϵ's as

$$F_{T,R}(\boldsymbol{\epsilon}) = \exp\left\{-\sum_{k\in T}\theta_k\left(\sum_{m\in R_k}\exp(-\epsilon_{km}/\tau_k)\right)^{\tau_k}\right\}$$

from which the probability of choice $t, t \in T$ can be derived as

$$\Pr(C_1 = t) = \frac{\theta_t\left\{\sum_{m\in R_t}\exp(\boldsymbol{\eta}_{tm}/\tau_t)\right\}^{\tau_t}}{\sum_{k\in T}\theta_k\left\{\sum_{m\in R_k}\exp(\boldsymbol{\eta}_{km}/\tau_k)\right\}^{\tau_k}} \tag{5}$$

`nlogit` sets $\theta_t = 1$. Noting that

$$\begin{aligned}\left\{\sum_{m\in R_t}\exp(\boldsymbol{\eta}_{tm}/\tau_t)\right\}^{\tau_t} &= \left\{\sum_{m\in R_t}\exp\left(\frac{\mathbf{z}_t\boldsymbol{\alpha}_t + \mathbf{x}_{tm}\boldsymbol{\beta}_m}{\tau_t}\right)\right\}^{\tau_t}\\ &= \exp(\mathbf{z}_t\boldsymbol{\alpha}_t)\left\{\sum_{m\in R_t}\exp\left(\mathbf{x}_{tm}\boldsymbol{\beta}_m/\tau_t\right)\right\}^{\tau_t}\\ &= \exp(\mathbf{z}_t\boldsymbol{\alpha}_t + \tau_t I_t)\end{aligned}$$

we define the inclusive values I_t as

$$I_t = \ln\left\{\sum_{m\in R_t}\exp(\mathbf{x}_{tm}\boldsymbol{\beta}_m/\tau_t)\right\}$$

and we can view

$$\exp(\tau_t I_t) = \left\{\sum_{m\in R_t}\exp(x_{tm}\boldsymbol{\beta}_m)^{1/\tau_t}\right\}^{\tau_t}$$

as a weighted average of the $\exp(x_{tm}\boldsymbol{\beta}_m)$, for $m \in R_t$. For the `nlogit` RUM parameterization, we can express (5) as

$$\Pr(C_1 = t) = \frac{\exp(\mathbf{z}_t\boldsymbol{\alpha}_t + \tau_t I_t)}{\sum_{k\in T}\exp(\mathbf{z}_k\boldsymbol{\alpha}_k + \tau_k I_k)}$$

Next, we define inclusive values for the nonnormalized model to be

$$\widetilde{I}_t = \ln\left\{\sum_{m\in R_t}\exp(\mathbf{x}_{tm}\boldsymbol{\beta}_m)\right\}$$

and we express $\Pr(C_1 = t)$ as

$$\Pr(C_1 = t) = \frac{\exp(\mathbf{z}_t\boldsymbol{\alpha}_t + \tau_t\widetilde{I}_t)}{\sum_{k\in T}\exp(\mathbf{z}_k\boldsymbol{\alpha}_k + \tau_k\widetilde{I}_k)} \tag{6}$$

Equation (5) is consistent with (6) only when $\boldsymbol{\eta}_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta}_j$, so in general the `nlogit` nonnormalized model is not consistent with the RUM.

Now assume that we have N cases where we add a third subscript, i, to denote case i, $i = 1, \ldots, N$. Denote y_{itj} to be a binary variable indicating the choice made by case i so that for each i only one y_{itj} is 1 and the rest are 0 for all $t \in T$ and $j \in R_t$. The log likelihood for the two-level RUM is

$$\begin{aligned}\log\ell &= \sum_{i=1}^{N}\sum_{k\in T}\sum_{m\in R_k} y_{itj}\log\{\Pr(C_{i1}=k)\Pr(C_{i2}=m|C_{i1}=k)\} \\ &= \sum_{i=1}^{N}\sum_{k\in T}\sum_{m\in R_k} y_{itj}\left[\mathbf{z}_{ik}\boldsymbol{\alpha}_k + \tau_k I_{ik} - \log\left\{\sum_{l\in T}\exp(\mathbf{z}_{il}\boldsymbol{\alpha}_l + \tau_l I_{il})\right\} + \right. \\ &\qquad \left. \mathbf{x}_{itj}\boldsymbol{\beta}_m/\tau_k - \log\left\{\sum_{l\in R_k}\exp(\mathbf{x}_{ikl}\boldsymbol{\beta}_l/\tau_k)\right\}\right]\end{aligned}$$

The likelihood for the nonnormalized model has a similar form, replacing I with $\widetilde{I}$ and by not scaling $\mathbf{x}_{ikj}\boldsymbol{\beta}_j$ by τ_k.

Three-level nested logit model

Here we define a three-level nested logit model that can be generalized to the four-level and higher models. As before, let the integer set T be the indices for the first level of choices. Let sets S_t, $t \in T$, be mutually exclusive sets of integers representing the choices of the second level of the hierarchy. Finally, let R_j, $j \in S_t$, be the bottom-level choices. Let $U_{tjk} = \eta_{tjk} + \epsilon_{tjk}$, $k \in R_j$, and the distribution of ϵ_{tjk} be Gumbel's multivariate extreme value of the form

$$F(\boldsymbol{\epsilon}) = \exp\left(-\sum_{t\in T}\left[\sum_{j\in S_t}\left\{\sum_{k\in R_j}\exp(-\eta_{tjk}/\tau_j)\right\}^{\tau_j/\upsilon_t}\right]^{\upsilon_j}\right)$$

Let C_1, C_2, and C_3 represent the choice random variables for levels 1, 2, and the bottom, respectively. Then the set of conditional probabilities is

$$\Pr(C_3 = k \,|\, C_1 = t, C_2 = j) = \frac{\exp(\eta_{tjk}/\tau_j)}{\sum_{l\in R_j}\exp(\eta_{tjl}/\tau_j)}$$

$$\Pr(C_2 = j \,|\, C_1 = t) = \frac{\left\{\sum_{k\in R_j}\exp(\eta_{tjk}/\tau_j)\right\}^{\tau_j/\upsilon_t}}{\sum_{l\in S_t}\left\{\sum_{k\in R_l}\exp(\eta_{tlk}/\tau_l)\right\}^{\tau_l/\upsilon_t}}$$

$$\Pr(C_1 = t) = \frac{\left[\sum_{j\in S_t}\left\{\sum_{k\in R_j}\exp(\eta_{tjk}/\tau_j)\right\}^{\tau_j/\upsilon_t}\right]^{\upsilon_t}}{\sum_{l\in T}\left[\sum_{j\in S_l}\left\{\sum_{k\in R_j}\exp(\eta_{ljk}/\tau_j)\right\}^{\tau_j/\upsilon_l}\right]^{\upsilon_l}}$$

Assume that we can decompose the linear predictor as $\boldsymbol{\eta}_{tjk} = \mathbf{z}_t\boldsymbol{\alpha}_t + \mathbf{u}_{tj}\boldsymbol{\gamma}_j + \mathbf{x}_{tjk}\boldsymbol{\beta}_k$. Here $\mathbf{z}_t$, $\mathbf{u}_{tj}$, and $\mathbf{x}_{tjk}$ are the row vectors of explanatory variables for the first, second, and bottom levels of the hierarchy, respectively, and $\boldsymbol{\alpha}_t$, $\boldsymbol{\gamma}_j$, and $\boldsymbol{\beta}_k$ are the corresponding column vectors of regression coefficients for $t \in T$, $j \in S_t$, and $k \in R_j$. We then can define the inclusive values for the first and second levels as

$$I_{tj} = \log \sum_{k \in R_j} \exp(\mathbf{x}_{tjk}\boldsymbol{\beta}_k/\tau_j)$$

$$J_t = \log \sum_{j \in S_t} \exp(\mathbf{u}_{tj}\boldsymbol{\gamma}_j/\upsilon_t + \frac{\tau_j}{\upsilon_t} I_{tj})$$

and rewrite the probabilities

$$\Pr(C_3 = k \,|\, C_1 = t, C_2 = j) = \frac{\exp(\mathbf{x}_{tjk}\boldsymbol{\beta}_k/\tau_j)}{\sum_{l \in R_j} \exp(\mathbf{x}_{tjl}\boldsymbol{\beta}_l/\tau_j)}$$

$$\Pr(C_2 = j \,|\, C_1 = t) = \frac{\exp(\mathbf{u}_{tj}\boldsymbol{\gamma}_j/\upsilon_t + \frac{\tau_j}{\upsilon_t} I_{tj})}{\sum_{l \in S_t} \exp(\mathbf{u}_{tl}\boldsymbol{\gamma}_l/\upsilon_t + \frac{\tau_l}{\upsilon_t} I_{tl})}$$

$$\Pr(C_1 = t) = \frac{\exp(\mathbf{z}_t\boldsymbol{\alpha}_t + \upsilon_t J_t)}{\sum_{l \in T} \exp(\mathbf{z}_l\boldsymbol{\alpha}_l + \upsilon_l J_l)}$$

We add a fourth index, i, for case and define the indicator variable y_{itjk}, $i = 1, \ldots, N$, to indicate the choice made by case i, $t \in T$, $j \in S_t$, and $k \in R_j$. The log likelihood for the `nlogit` RUM is

$$\ell = \sum_{i=1}^{N} \sum_{t \in T} \sum_{j \in S_t} \sum_{k \in R_j} y_{itjk} \left\{ \mathbf{z}_{it}\boldsymbol{\alpha}_t + \upsilon_t J_{it} - \log \left(\sum_{m \in T} \mathbf{z}_{im}\boldsymbol{\alpha}_m + \upsilon_m J_{im} \right) + \right.$$
$$\mathbf{u}_{itj}\boldsymbol{\gamma}_j/\upsilon_t + \frac{\tau_j}{\upsilon_t} I_{itj} - \log \left(\sum_{m \in S_t} \mathbf{u}_{itm}\boldsymbol{\gamma}_m/\upsilon_t + \frac{\tau_m}{\upsilon_t} I_{itm} \right) +$$
$$\left. \mathbf{x}_{itjk}\boldsymbol{\beta}_k/\tau_k - \sum_{m \in R_t} \exp(\mathbf{x}_{itjm}\boldsymbol{\beta}_m/\tau_k) \right\}$$

and for the nonnormalized `nlogit` model, the log likelihood is

$$\ell = \sum_{i=1}^{N} \sum_{t \in T} \sum_{j \in S_t} \sum_{k \in R_j} y_{itjk} \left\{ \mathbf{z}_{it}\boldsymbol{\alpha}_t + \upsilon_t J_{it} - \log \left(\sum_{m \in T} \mathbf{z}_{im}\boldsymbol{\alpha}_m + \upsilon_m J_{im} \right) + \right.$$
$$\mathbf{u}_{itj}\boldsymbol{\gamma}_j + \tau_j I_{itj} - \log \left(\sum_{m \in S_t} \mathbf{u}_{itm}\boldsymbol{\gamma}_m + \tau_m I_{itm} \right) +$$
$$\left. \mathbf{x}_{itjk}\boldsymbol{\beta}_k - \sum_{m \in R_t} \exp(\mathbf{x}_{itjm}\boldsymbol{\beta}_m) \right\}$$

Extending the model to more than three levels is straightforward, albeit notationally cumbersome.

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*.

References

Amemiya, T. 1985. *Advanced Econometrics*. Cambridge, MA: Harvard University Press.

Greene, W. H. 2018. *Econometric Analysis*. 8th ed. New York: Pearson.

Hausman, J. A. 1978. Specification tests in econometrics. *Econometrica* 46: 1251–1271. https://doi.org/10.2307/1913827.

Hausman, J. A., and D. L. McFadden. 1984. Specification tests for the multinomial logit model. *Econometrica* 52: 1219–1240. https://doi.org/10.2307/1910997.

Heiss, F. 2002. Structural choice analysis with nested logit models. *Stata Journal* 2: 227–252.

Hensher, D. A., J. M. Rose, and W. H. Greene. 2005. *Applied Choice Analysis: A Primer*. New York: Cambridge University Press.

——. 2015. *Applied Choice Analysis*. 2nd ed. Cambridge: Cambridge University Press.

Kotz, S., and S. Nadarajah. 2000. *Extreme Value Distributions: Theory and Applications*. London: Imperial College Press.

Maddala, G. S. 1983. *Limited-Dependent and Qualitative Variables in Econometrics*. Cambridge: Cambridge University Press.

McFadden, D. L. 1977. Quantitative methods for analyzing travel behaviour of individuals: Some recent developments. Working paper 474, Cowles Foundation. http://cowles.yale.edu/cfdp-474.

——. 1981. Econometric models of probabilistic choice. In *Structural Analysis of Discrete Data with Econometric Applications*, ed. C. F. Manski and D. L. McFadden, 198–272. Cambridge, MA: MIT Press.

Also see

[CM] **nlogit postestimation** — Postestimation tools for nlogit

[CM] **cmclogit** — Conditional logit (McFadden's) choice model

[CM] **cmmixlogit** — Mixed logit choice model

[CM] **cmmprobit** — Multinomial probit choice model

[R] **clogit** — Conditional (fixed-effects) logistic regression

[R] **mlogit** — Multinomial (polytomous) logistic regression

[R] **mprobit** — Multinomial probit regression

[R] **ologit** — Ordered logistic regression

[R] **slogit** — Stereotype logistic regression

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

Title

nlogit postestimation — Postestimation tools for nlogit

Postestimation commands predict estat
Remarks and examples Also see

Postestimation commands

The following postestimation command is of special interest after `nlogit`:

Command	Description
`estat alternatives`	alternative summary statistics

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
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients
`predict`	probabilities, linear predictions, inclusive values, 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 probabilities, linear predictions, conditional probabilities, and inclusive values.

Menu for predict

Statistics > Postestimation

Syntax for predict

`predict` [*type*] *newvar* [*if*] [*in*] [`,` *statistic* `hlevel(#)`]

`predict` [*type*] *stub*`*` [*if*] [*in*], `scores`

statistic	Description
Main	
`pr`	predicted probabilities of choosing the alternatives at all levels of the hierarchy or at level #, where # is specified by `hlevel(#)`; the default
`xb`	linear predictors for all levels of the hierarchy or at level #, where # is specified by `hlevel(#)`
`condp`	predicted conditional probabilities at all levels of the hierarchy or at level #, where # is specified by `hlevel(#)`
`iv`	inclusive values for levels 2, ..., `e(levels)` or for `hlevel(#)`

The inclusive value for the first-level alternatives is not used in estimation; therefore, it is not calculated.

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

`predict` omits missing values casewise if `nlogit` used casewise deletion (the default); if `nlogit` used alternativewise deletion (option `altwise`), `predict` uses alternativewise deletion.

Options for predict

Main

`pr` calculates the probability of choosing each alternative at each level of the hierarchy. Use the `hlevel(#)` option to compute the alternative probabilities at level #. When `hlevel(#)` is not specified, j new variables must be given, where j is the number of levels, or use *stub*`*` to have `predict` generate j variables with the prefix *stub* and numbered from 1 to j. The `pr` option is the default, and if one new variable is given, the probability of the bottom-level alternatives are computed. Otherwise, probabilities for all levels are computed, and *stub*`*` is still valid.

`xb` calculates the linear prediction for each alternative at each level. Use the `hlevel(#)` option to compute the linear predictor at level #. When `hlevel(#)` is not specified, j new variables must be given, where j is the number of levels, or use *stub*`*` to have `predict` generate j variables with the prefix *stub* and numbered from 1 to j.

`condp` calculates the conditional probabilities for each alternative at each level. Use the `hlevel(#)` option to compute the conditional probabilities of the alternatives at level #. When `hlevel(#)` is not specified, j new variables must be given, where j is the number of levels, or use *stub*`*` to have `predict` generate j variables with the prefix *stub* and numbered from 1 to j.

`iv` calculates the inclusive value for each alternative at each level. Use the `hlevel(#)` option to compute the inclusive value at level #. There is no inclusive value at level 1. If `hlevel(#)` is not used, $j - 1$ new variables are required, where j is the number of levels, or use *stub*`*` to have `predict` generate $j - 1$ variables with the prefix *stub* and numbered from 2 to j. See *Methods and formulas* in [CM] **nlogit** for a definition of the inclusive values.

`hlevel(#)` calculates the prediction only for hierarchy level #.

`scores` calculates the scores for each coefficient in `e(b)`. This option requires a new-variable list of length equal to the number of columns in `e(b)`. Otherwise, use *stub*`*` to have `predict` generate enumerated variables with prefix *stub*.

estat

Description for estat

`estat alternatives` displays summary statistics about the alternatives in the estimation sample for each level of the tree structure.

Menu for estat

Statistics > Postestimation

Syntax for estat

```
estat alternatives
```

Remarks and examples

`predict` may be used after `nlogit` to obtain the predicted values of the probabilities, the conditional probabilities, the linear predictions, and the inclusive values for each level of the nested logit model. Predicted probabilities for `nlogit` must be interpreted carefully. Probabilities are estimated for each case as a whole and not for individual observations.

▷ Example 1

Continuing with our model in example 3 of [CM] **nlogit**, we refit the model and then examine a summary of the alternatives and their frequencies in the estimation sample.

```
. use https://www.stata-press.com/data/r18/restaurant
. nlogitgen type = restaurant(fast: Freebirds | MamasPizza,
> family: CafeEccell | LosNortenos | WingsNmore, fancy: Christophers | MadCows)
```

(*output omitted*)

```
. nlogit chosen cost rating distance || type: income kids, base(family) ||
> restaurant:, noconst case(family_id)
```

(*output omitted*)

```
. estat alternatives
```

Alternatives summary for type

Alternative index	Alternative value	label	Cases present	Frequency selected	Percent selected
1	1	fast	600	27	9.00
2	2	family	900	222	74.00
3	3	fancy	600	51	17.00

Alternatives summary for restaurant

Alternative index	Alternative value	label	Cases present	Frequency selected	Percent selected
1	1	Freebirds	300	12	4.00
2	2	MamasPizza	300	15	5.00
3	3	CafeEccell	300	78	26.00
4	4	LosNortenos	300	75	25.00
5	5	WingsNmore	300	69	23.00
6	6	Christophers	300	27	9.00
7	7	MadCows	300	24	8.00

Next, we predict `p2` $= \Pr($`restaurant`$)$; `p1` $= \Pr($`type`$)$; `condp` $= \Pr($`restaurant` $\mid$ `type`$)$; `xb2`, the linear prediction for the bottom-level alternatives; `xb1`, the linear prediction for the first-level alternatives; and `iv`, the inclusive values for the bottom-level alternatives.

```
. predict p*
(option pr assumed)
. predict condp, condp hlevel(2)
. sort family_id type restaurant
. list restaurant type chosen p2 p1 condp in 1/14, sepby(family_id) divider
```

	restaurant	type	chosen	p2	p1	condp
1.	Freebirds	fast	1	.0642332	.1189609	.5399519
2.	MamasPizza	fast	0	.0547278	.1189609	.4600481
3.	CafeEccell	family	0	.284409	.7738761	.3675124
4.	LosNortenos	family	0	.3045242	.7738761	.3935051
5.	WingsNmore	family	0	.1849429	.7738761	.2389825
6.	Christophers	fancy	0	.0429508	.107163	.4007991
7.	MadCows	fancy	0	.0642122	.107163	.5992009
8.	Freebirds	fast	0	.0183578	.0488948	.3754559
9.	MamasPizza	fast	0	.030537	.0488948	.6245441
10.	CafeEccell	family	0	.2832149	.756065	.3745907
11.	LosNortenos	family	1	.3038883	.756065	.4019341
12.	WingsNmore	family	0	.1689618	.756065	.2234752
13.	Christophers	fancy	0	.1041277	.1950402	.533878
14.	MadCows	fancy	0	.0909125	.1950402	.466122

```
. predict xb*, xb
. predict iv, iv
```

```
. list restaurant type chosen xb* iv in 1/14, sepby(family_id) divider
```

	restaurant	type	chosen	xb1	xb2	iv
1.	Freebirds	fast	1	-1.124805	-1.476914	-.2459659
2.	MamasPizza	fast	0	-1.124805	-1.751229	-.2459659
3.	CafeEccell	family	0	0	-2.181112	.1303341
4.	LosNortenos	family	0	0	-2.00992	.1303341
5.	WingsNmore	family	0	0	-3.259229	.1303341
6.	Christophers	fancy	0	1.405185	-6.804211	-.745332
7.	MadCows	fancy	0	1.405185	-5.155514	-.745332
8.	Freebirds	fast	0	-1.804794	-2.552233	-.5104123
9.	MamasPizza	fast	0	-1.804794	-1.680583	-.5104123
10.	CafeEccell	family	0	0	-2.400434	.0237072
11.	LosNortenos	family	1	0	-2.223939	.0237072
12.	WingsNmore	family	0	0	-3.694409	.0237072
13.	Christophers	fancy	0	1.490775	-5.35932	-.6796131
14.	MadCows	fancy	0	1.490775	-5.915751	-.6796131

Also see

[CM] **nlogit** — Nested logit regression

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

Glossary

alternatives. The set of alternatives are the possible choices a decision maker can pick or rank.

alternative-specific variable. When a variable varies across alternatives, it is called alternative specific. An alternative-specific variable may vary across alternatives only or across both alternatives and cases.

alternatives variable. A numeric or string variable that identifies the alternatives. Some models require an alternatives variable and some do not.

balanced. When choice sets are the same for every case, we say that they are balanced.

case. This is a Stata term for the set of Stata observations representing a single decision. A case contains one observation for each of the possible alternatives that the decision maker could have chosen or ranked.

case ID variable. A variable that identifies the cases. For independent cross-sectional data, this variable identifies the decision makers.

case-specific variable. When a variable is constant within a case, it is called case specific.

choice set. The set of alternatives a decision maker could have chosen or ranked. The choice sets can vary across cases.

discrete choice. When each decision maker picks a single alternative from his or her set of possible alternatives, it is called a discrete choice.

independence of irrelevant alternatives (IIA). The IIA property is true when adding another alternative to the set of alternatives does not change the relative probabilities of choosing alternatives from the initial set of alternatives.

observation. For choice models in Stata, there is a difference between Stata observations and statistical observations. We call a statistical observation a case. When we refer to an observation, we mean a Stata observation—one row in the dataset.

panel data. When decision makers make multiple choices at different time points, the data are panel data. A panel variable identifies decision makers, and a time variable identifies the time points.

rank-ordered alternatives. When each decision maker ranks his or her possible alternatives, we say we have rank-ordered alternatives.

unbalanced. When choice sets are not the same for every case, we say that they are unbalanced.

utility. Choice models are typically formulated using a latent continuous variable, called the utility, for each alternative. The largest value of the utility for each case represents the alternative chosen for discrete choices. For rank-ordered alternatives, the ranking of the values of the utilities gives the rank ordering of the choices.

Subject and author index

See the combined subject index and the combined author index in the *Stata Index*.