Title

stata.com

cmmprobit — Multinomial probit choice model

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

stata.com

Remarks are presented under the following headings:

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 $\Sigma_{(k)}$ 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

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 $\zeta_j \sim$ 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

$$\widetilde{\zeta}_j = \begin{cases} \Phi^{-1}\left\{\delta_1 \Phi\left(-\frac{\lambda_1}{l_{11}}\right)\right\} & \text{for } j = 1 \\ \Phi^{-1}\left\{\delta_j \Phi\left(\frac{-\lambda_j - \sum_{i=1}^{j-1} l_{ji}\widetilde{\zeta}_i}{l_{jj}}\right)\right\} & \text{for } j = 2, \ldots, J-1 \end{cases}$$

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