Title

stata.com

asmprobit — Alternative-specific multinomial probit regression

Syntax

```
asmprobit depvar [indepvars] [if] [in] [weight], case(varname)
    alternatives(varname) [options]
```

options	Description
Model	
* `case(`*varname*`)`	use *varname* to identify cases
* `alternatives(`*varname*`)`	use *varname* to identify the alternatives available for each case
`casevars(`*varlist*`)`	case-specific variables
`constraints(`*constraints*`)`	apply specified linear constraints
`collinear`	keep collinear variables
Model 2	
`correlation(`*correlation*`)`	correlation structure of the latent-variable errors
`stddev(`*stddev*`)`	variance structure of the latent-variable errors
`structural`	use the structural covariance parameterization; default is the differenced covariance parameterization
`factor(#)`	use the factor covariance structure with dimension #
`noconstant`	suppress the alternative-specific constant terms
`basealternative(#|`*lbl*`|`*str*`)`	alternative used for normalizing location
`scalealternative(#|`*lbl*`|`*str*`)`	alternative used for normalizing scale
`altwise`	use alternativewise deletion instead of casewise deletion
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 column formats and line width

Integration

intmethod(*seqtype*)	type of quasi- or pseudouniform point set
intpoints(#)	number of points in each sequence
intburn(#)	starting index in the Hammersley or Halton sequence
intseed(*code* | #)	pseudouniform 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
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 pseudorandom point set

*case(*varname*) and alternatives(*varname*) are required.

bootstrap, by, jackknife, statsby, and xi 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**.

coeflegend does not appear in the dialog box.

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

Menu

Statistics > Categorical outcomes > Alternative-specific multinomial probit

Description

`asmprobit` fits multinomial probit (MNP) models by using maximum simulated likelihood (MSL) implemented by the Geweke–Hajivassiliou–Keane (GHK) algorithm. By estimating the variance–covariance parameters of the latent-variable errors, the model allows you to relax the independence of irrelevant alternatives (IIA) property that is characteristic of the multinomial logistic model.

`asmprobit` requires multiple observations for each case (decision), where each observation represents an alternative that may be chosen. The cases are identified by the variable specified in the `case()` option, whereas the alternatives are identified by the variable specified in the `alternative()` option. The outcome (chosen alternative) is identified by a value of 1 in *depvar*, with 0 indicating the alternatives that were not chosen; only one alternative may be chosen for each case.

`asmprobit` allows two types of independent variables: alternative-specific variables and case-specific variables. Alternative-specific variables vary across both cases and alternatives and are specified in *indepvars*. Case-specific variables vary only across cases and are specified in the `casevars()` option.

Options

Model

`case(`*varname*`)` specifies the variable that identifies each case. This variable identifies the individuals or entities making a choice. `case()` is required.

`alternatives(`*varname*`)` specifies the variable that identifies the alternatives for each case. The number of alternatives can vary with each case; the maximum number of alternatives is 20. `alternatives()` is required.

`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()`.

`constraints(`*constraints*`)`, `collinear`; see [R] **estimation options**.

Model 2

`correlation(`*correlation*`)` specifies the correlation structure of the latent-variable errors.

`correlation(unstructured)` is the most general and has $J(J-3)/2+1$ unique correlation parameters. This is the default unless `stdev()` or `structural` are specified.

`correlation(exchangeable)` provides for one correlation coefficient common to all latent variables, except the latent variable 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 *Variance structures* later in this entry for more information.

`stddev(`*stddev*`)` specifies the variance structure of the latent-variable errors.

`stddev(heteroskedastic)` is the most general and has $J-2$ estimable parameters. The standard deviations of the latent-variable 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 *Variance structures* later in this entry for more information.

`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 latent 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()`.

`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 variance–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.

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

`basealternative(#` | *lbl* | *str*`)` specifies the alternative used to normalize the latent-variable location (also referred to as the level of utility). The base alternative may be specified as a number, label, or string. The standard deviation for the latent-variable error associated with the base alternative is fixed to one, and its correlations with all other latent-variable 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 latent-variable scale (also referred to as the scale of 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.

`altwise` specifies that alternativewise deletion be used when marking out 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 deleted if any missing values are encountered. This option does not apply to observations that are marked out by the `if` or `in` qualifier or the `by` prefix.

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 variance–covariance estimates to the correlation and standard deviation metric.

This option has no effect if `structural` is not specified because the default differenced variance–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 variance–covariance matrix, so optimization is already performed using the standard deviation and correlation representations.

`nocnsreport`; see [R] **estimation options**.

display_options: `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 quasi–Monte Carlo integration. If this option is not specified, the number of points is $50 \times J$ if `intmethod(hammersley)` or `intmethod(halton)` is used and $100 \times J$ if `intmethod(random)` is used. Larger values of `intpoints()` provide better approximations of the log likelihood, but at the cost of added 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 0. This option may not be specified with `intmethod(random)`.

`intseed(`*code* | `#)` specifies the seed to use for generating the uniform pseudorandom 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(seed)`.

`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, `asmprobit` 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. `asmprobit` 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 `asmprobit` 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 a large number of 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 `asmprobit`: `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 following option is available with `asmprobit` but is not shown in the dialog box:

`coeflegend`; see [R] **estimation options**.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction

The MNP model is used with discrete dependent variables that take on more than two outcomes that do not have a natural ordering. The stochastic error terms are assumed to have a multivariate normal distribution that is heteroskedastic and correlated. Say that you have a set of J unordered alternatives that are modeled by a regression of both case-specific and alternative-specific covariates. A "case" refers to the information on one decision maker. Underlying the model is the set of J latent variables (utilities),

$$\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j + \xi_{ij} \tag{1}$$

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 multivariate normal with mean zero and covariance matrix $\boldsymbol{\Omega}$. The decision maker selects the alternative whose latent variable is highest.

Because the MNP model allows for a general covariance structure in ξ_{ij}, it does not impose the IIA property inherent in multinomial logistic and conditional logistic models. That is, the MNP model permits the odds 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 odds 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.

An alternative to MNP that will allow a nested correlation structure in ξ_{ij} is the nested logit model (see [R] **nlogit**).

The added flexibility of the MNP model does impose a significant computation burden because of the need to evaluate probabilities from the multivariate normal distribution. These probabilities are evaluated using simulation techniques 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 latent variable are irrelevant. Increasing the latent variables 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, indexed by k, say, and take the difference between the latent variable k and the $J-1$ others,

$$\begin{aligned} v_{ijk} &= \eta_{ij} - \eta_{ik} \\ &= (\mathbf{x}_{ij} - \mathbf{x}_{ik})\boldsymbol{\beta} + \mathbf{z}_i(\boldsymbol{\alpha}_j - \boldsymbol{\alpha}_k) + \xi_{ij} - \xi_{ik} \\ &= \boldsymbol{\delta}_{ij'}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\gamma}_{j'} + \epsilon_{ij'} \\ &= \lambda_{ij'} + \epsilon_{ij'} \end{aligned} \tag{2}$$

where $j' = j$ if $j < k$ and $j' = j-1$ if $j > k$, so that $j' = 1, \ldots, J-1$. One 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 `asmprobit`. From (2), 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 `asmprobit`, 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 *Variance structures* below for more on this issue.

In fact, the model is slightly more general in that not all cases need to have faced all J alternatives. The model allows for situations in which some cases chose among all possible alternatives, whereas other cases were given a choice among a subset of them, and perhaps other cases were given a choice among a different subset. The number of observations for each case is equal to the number of alternatives faced.

The MNP model is often motivated using a random-utility consumer-choice framework. Equation (1) represents the utility that consumer i receives from good j. The consumer purchases the good for which the utility is highest. Because utility is ordinal, all that matters is the ranking of the utilities from the alternatives. Thus one must normalize for location and scale.

▷ Example 1

Application of MNP models is common in the analysis of transportation data. Greene (2012, sec. 18.2.9) 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 http://www.stata-press.com/data/r13/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

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 set 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 also the defaults.

```
. asmprobit choice travelcost termtime, case(id) alternatives(mode)
> casevars(income)
```

(*output omitted*)

```
Alternative-specific multinomial probit        Number of obs      =       840
Case variable: id                              Number of cases    =       210
Alternative variable: mode                     Alts per case: min =         4
                                                              avg =       4.0
                                                              max =         4
Integration sequence:      Hammersley
Integration points:               200          Wald chi2(5)       =     32.05
Log simulated-likelihood = -190.09418          Prob > chi2        =    0.0000
```

choice	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
mode						
travelcost	-.00977	.0027834	-3.51	0.000	-.0152253	-.0043146
termtime	-.0377095	.0094088	-4.01	0.000	-.0561504	-.0192686
air	(base alternative)					
train						
income	-.0291971	.0089246	-3.27	0.001	-.046689	-.0117052
_cons	.5616376	.3946551	1.42	0.155	-.2118721	1.335147
bus						
income	-.0127503	.0079267	-1.61	0.108	-.0282863	.0027857
_cons	-.0571364	.4791861	-0.12	0.905	-.9963239	.882051
car						
income	-.0049086	.0077486	-0.63	0.526	-.0200957	.0102784
_cons	-1.833393	.8186156	-2.24	0.025	-3.43785	-.2289357
/lnl2_2	-.5502039	.3905204	-1.41	0.159	-1.31561	.2152021
/lnl3_3	-.6005552	.3353292	-1.79	0.073	-1.257788	.0566779
/l2_1	1.131518	.2124817	5.33	0.000	.7150612	1.547974
/l3_1	.9720669	.2352116	4.13	0.000	.5110606	1.433073
/l3_2	.5197214	.2861552	1.82	0.069	-.0411325	1.080575

```
(mode=air is the alternative normalizing location)
(mode=train is the alternative normalizing scale)
. estimates store full
```

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, `asmprobit` 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 the 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.

Although the transformed parameters of the differenced covariance parameterization are difficult to interpret, you can view them untransformed by using the `estat` command. Typing

```
. estat correlation
```

	train	bus	car
train	1.0000		
bus	0.8909	1.0000	
car	0.7895	0.8951	1.0000

```
Note: correlations are for alternatives differenced with air
```

gives the correlations, and typing

```
. estat covariance
```

	train	bus	car
train	2		
bus	1.600208	1.613068	
car	1.37471	1.399703	1.515884

```
Note: covariances are for alternatives differenced with air
```

gives the (co)variances.

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.

```
. asmprobit choice travelcost termtime, case(id) alternatives(mode)
> casevars(income) factor(1)
```

(*output omitted*)

```
Alternative-specific multinomial probit        Number of obs      =       840
Case variable: id                              Number of cases    =       210

Alternative variable: mode                     Alts per case: min =         4
                                                              avg =       4.0
                                                              max =         4

Integration sequence:      Hammersley
Integration points:              200           Wald chi2(5)       =    107.85
Log simulated-likelihood = -196.85094          Prob > chi2        =    0.0000
```

choice	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
mode						
travelcost	-.0093696	.0036329	-2.58	0.010	-.01649	-.0022492
termtime	-.0593173	.0064585	-9.18	0.000	-.0719757	-.0466589
air	(base alternative)					
train						
income	-.0373511	.0098219	-3.80	0.000	-.0566018	-.0181004
_cons	.1092322	.3949529	0.28	0.782	-.6648613	.8833257
bus						
income	-.0158793	.0112239	-1.41	0.157	-.0378777	.0061191
_cons	-1.082181	.4678732	-2.31	0.021	-1.999196	-.1651666
car						
income	.0042677	.0092601	0.46	0.645	-.0138817	.0224171
_cons	-3.765445	.5540636	-6.80	0.000	-4.851389	-2.6795
/c1_2	1.182805	.3060299	3.86	0.000	.5829972	1.782612
/c1_3	1.227705	.3401237	3.61	0.000	.5610747	1.894335

```
(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	1.182805	2.399027	
car	1.227705	1.452135	2.507259

```
Note: covariances are for alternatives differenced with air
```

◁

Variance 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 reduces the number of estimable parameters by J. Moreover, selecting a scale alternative and normalizing its error variance to one reduces the number by one, as well. 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 `asmprobit` 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, `asmprobit` 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 $\boldsymbol{\Omega}$, 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 fitting a complex model, it may be advantageous to first fit a simple one and then proceed with removing the restrictions one at a time.

Advanced users may wish to specify alternative variance 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: 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. `asmprobit` considers only the elements of the matrix *below* the main diagonal.

Suppose that you have a model with four alternatives, numbered 1–4, and alternative 1 is the base. The unstructured and exchangeable correlation structures identified in the 4×4 lower triangular matrices are

unstructured

	1	2	3	4
1	·			
2	0	·		
3	0	1	·	
4	0	2	3	·

exchangeable

	1	2	3	4
1	·			
2	0	·		
3	0	1	·	
4	0	1	1	·

`asmprobit` labels these correlation structures unstructured and exchangeable, 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 $\boldsymbol{\Sigma}_{(k)}$ defined in the *Introduction* 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:

	1	2	3	4
1	·			
2	0.5	·		
3	0.5	·	·	
4	0.5	·	·	·

The order of the elements of the `pattern` or `fixed` matrices must be the same as the numeric order of the alternative levels.

To specify the structure of the standard deviations—the diagonal elements of $\boldsymbol{\Omega}$—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. In the four-alternative example mentioned above, suppose that you wish to set the first and second standard deviations to one and that you wish to constrain the third and fourth standard deviations to be equal; the following `pattern` matrix will do that:

$$\begin{array}{c} \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} \cdot & \cdot & 1 & 1 \end{pmatrix} \end{array}$$

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 one and to allow the third and fourth to be estimated, you would use this `fixed` matrix:

$$\begin{array}{c} \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 1 & \cdot & \cdot \end{pmatrix} \end{array}$$

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 `asmprobit` detects that the model is not scaled.

The order of the elements of the `pattern` or `fixed` matrices must be the same as the numeric order of the alternative levels.

▷ Example 2

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. Here we will fix the standard deviations for the `air` and `train` alternatives to 1 and the correlations between `air` and the rest of the alternatives to 0.

```
. asmprobit choice travelcost termtime, case(id) alternatives(mode)
> casevars(income) structural
```

(*output omitted*)

```
Alternative-specific multinomial probit        Number of obs      =       840
Case variable: id                              Number of cases    =       210
Alternative variable: mode                     Alts per case: min =         4
                                                              avg =       4.0
                                                              max =         4
Integration sequence:      Hammersley
Integration points:               200          Wald chi2(5)       =     32.05
Log simulated-likelihood = -190.09418          Prob > chi2        =    0.0000
```

choice	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
mode						
travelcost	-.0097703	.0027834	-3.51	0.000	-.0152257	-.0043149
termtime	-.0377103	.0094092	-4.01	0.000	-.056152	-.0192687
air	(base alternative)					
train						
income	-.0291975	.0089246	-3.27	0.001	-.0466895	-.0117055
_cons	.5616448	.3946529	1.42	0.155	-.2118607	1.33515
bus						
income	-.01275	.0079266	-1.61	0.108	-.0282858	.0027858
_cons	-.0571664	.4791996	-0.12	0.905	-.9963803	.8820476
car						
income	-.0049085	.0077486	-0.63	0.526	-.0200955	.0102785
_cons	-1.833444	.8186343	-2.24	0.025	-3.437938	-.22895
/lnsigma3	-.2447428	.4953363	-0.49	0.621	-1.215584	.7260985
/lnsigma4	-.3309429	.6494493	-0.51	0.610	-1.60384	.9419543
/atanhr3_2	1.01193	.3890994	2.60	0.009	.249309	1.774551
/atanhr4_2	.5786576	.3940461	1.47	0.142	-.1936586	1.350974
/atanhr4_3	.8885204	.5600561	1.59	0.113	-.2091693	1.98621
sigma1	1	(base alternative)				
sigma2	1	(scale alternative)				
sigma3	.7829059	.3878017			.2965368	2.067
sigma4	.7182462	.4664645			.2011227	2.564989
rho3_2	.766559	.1604596			.244269	.9441061
rho4_2	.5216891	.2868027			-.1912734	.874283
rho4_3	.7106622	.277205			-.2061713	.9630403

```
(mode=air is the alternative normalizing location)
(mode=train is the alternative normalizing scale)
```

When comparing this output to 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	.6001436	.6129416	
car	0	.3747012	.399619	.5158776

```
. 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  1.6001436  1.6129416
r3  1.3747012   1.399619  1.5158776
```

The slight difference in the regression coefficients between the example 1 and example 2 coefficient tables reflects the accuracy of the [M-5] **ghk( )** algorithm using 200 points from the Hammersley sequence.

We now fit the model using the exchangeable correlation matrix and compare the models with a likelihood-ratio test.

```
. asmprobit choice travelcost termtime, case(id) alternatives(mode)
> casevars(income) correlation(exchangeable)
```

(*output omitted*)

```
Alternative-specific multinomial probit        Number of obs      =       840
Case variable: id                              Number of cases    =       210
Alternative variable: mode                     Alts per case: min =         4
                                                              avg =       4.0
                                                              max =         4
Integration sequence:      Hammersley
Integration points:               200          Wald chi2(5)       =     53.60
Log simulated-likelihood =  -190.4679          Prob > chi2        =    0.0000
```

choice	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
mode						
travelcost	-.0084636	.0020452	-4.14	0.000	-.012472	-.0044551
termtime	-.0345394	.0072812	-4.74	0.000	-.0488103	-.0202684
air	(base alternative)					
train						
income	-.0290357	.0083226	-3.49	0.000	-.0453477	-.0127237
_cons	.5517445	.3719913	1.48	0.138	-.177345	1.280834
bus						
income	-.0132562	.0074133	-1.79	0.074	-.0277859	.0012735
_cons	-.0052517	.4337932	-0.01	0.990	-.8554708	.8449673
car						
income	-.0060878	.006638	-0.92	0.359	-.0190981	.0069224
_cons	-1.565918	.6633007	-2.36	0.018	-2.865964	-.265873
/lnsigmaP1	-.3557589	.1972809	-1.80	0.071	-.7424222	.0309045
/lnsigmaP2	-1.308596	.8872957	-1.47	0.140	-3.047663	.4304719
/atanhrP1	1.116589	.3765488	2.97	0.003	.3785667	1.854611
sigma1	1	(base alternative)				
sigma2	1	(scale alternative)				
sigma3	.7006416	.1382232			.4759596	1.031387
sigma4	.2701992	.2397466			.0474697	1.537983
rho3_2	.8063791	.131699			.3614621	.9521783
rho4_2	.8063791	.131699			.3614621	.9521783
rho4_3	.8063791	.131699			.3614621	.9521783

```
(mode=air is the alternative normalizing location)
(mode=train is the alternative normalizing scale)
. lrtest full .
Likelihood-ratio test                                 LR chi2(2)  =      0.75
(Assumption: . nested in full)                        Prob > chi2 =    0.6882
```

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 `asmprobit`. The pattern matrices are stored in `e(stdpattern)` and `e(corpattern)`.

❑ Technical note

Another way to fit the model with the exchangeable correlation structure in example 2 is to use the `constraint` command to define the constraints on the `rho` parameters manually and then apply those.

```
. constraint 1 [atanhr3_2]_cons = [atanhr4_2]_cons
. constraint 2 [atanhr3_2]_cons = [atanhr4_3]_cons
. asmprobit choice travelcost termtime, case(id) alternatives(mode)
> casevars(income) constraints(1 2) structural
```

With this method, however, we must keep track of what parameterization of the `rho`s is used in estimation, and that depends on the options specified.

❑

▷ Example 3

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–correlation parameter model by specifying a `pattern` matrix in the `correlation()` option. Suppose that we wish to have the correlation between train and bus be equal to the correlation between bus and car and to have the standard deviations for the bus and car equations be equal. We will use air as the base category and train as the scale category.

```
. matrix define corpat = J(4, 4, .)
. matrix corpat[3,2] = 1
. matrix corpat[4,3] = 1
. matrix corpat[4,2] = 2
. matrix define stdpat = J(1, 4, .)
. matrix stdpat[1,3] = 1
. matrix stdpat[1,4] = 1
. asmprobit choice travelcost termtime, case(id) alternatives(mode)
> casevars(income) correlation(pattern corpat) stddev(pattern stdpat)
```

(*output omitted*)

```
Alternative-specific multinomial probit        Number of obs      =       840
Case variable: id                              Number of cases    =       210
Alternative variable: mode                     Alts per case: min =         4
                                                              avg =       4.0
                                                              max =         4
Integration sequence:      Hammersley
Integration points:               200          Wald chi2(5)       =     41.67
Log simulated-likelihood = -190.12871          Prob > chi2        =    0.0000
```

choice	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
mode						
travelcost	-.0100335	.0026203	-3.83	0.000	-.0151692	-.0048979
termtime	-.0385731	.008608	-4.48	0.000	-.0554445	-.0217018
air	(base alternative)					
train						
income	-.029271	.0089739	-3.26	0.001	-.0468595	-.0116824
_cons	.56528	.4008037	1.41	0.158	-.2202809	1.350841
bus						
income	-.0124658	.0080043	-1.56	0.119	-.0281539	.0032223
_cons	-.0741685	.4763422	-0.16	0.876	-1.007782	.859445
car						
income	-.0046905	.0079934	-0.59	0.557	-.0203573	.0109763
_cons	-1.897931	.7912106	-2.40	0.016	-3.448675	-.3471867
/lnsigmaP1	-.197697	.2751269	-0.72	0.472	-.7369359	.3415418
/atanhrP1	.9704403	.3286981	2.95	0.003	.3262038	1.614677
/atanhrP2	.5830923	.3690419	1.58	0.114	-.1402165	1.306401
sigma1	1	(base alternative)				
sigma2	1	(scale alternative)				
sigma3	.8206185	.2257742			.4785781	1.407115
sigma4	.8206185	.2257742			.4785781	1.407115
rho3_2	.7488977	.1443485			.3151056	.9238482
rho4_2	.5249094	.2673598			-.1393048	.863362
rho4_3	.7488977	.1443485			.3151056	.9238482

```
(mode=air is the alternative normalizing location)
(mode=train is the alternative normalizing scale)
```

In the call to `asmprobit`, we did not need to specify the `basealternative()` and `scalealternative()` options because they are implied by the specifications of the pattern matrices.

◁

❑ Technical note

If you experience convergence problems, try specifying `nopivot`, increasing `intpoints()`, specifying `antithetics`, specifying `technique(nr)` with `difficult`, or specifying a switching algorithm in the `technique()` option. As a last resort, you can use the `nrtolerance()` and `showtolerance` options. Changing the base and scale alternative in the model specification can also affect convergence if the `structural` option is used.

Because simulation methods are used to obtain multivariate normal probabilities, the estimates obtained have a limited degree of precision. Moreover, the solutions are particularly sensitive to the starting values used. Experimenting with different starting values may help in obtaining convergence, and doing so is a good way to verify previous results.

If you wish to use the BHHH algorithm along with another maximization algorithm, you must specify the `initbhhh(#)` option, where # is the number of BHHH iterations to use before switching to the algorithm specified in `technique()`. The BHHH algorithm uses an outer-product-of-gradients approximation for the Hessian, and `asmprobit` must perform the gradient calculations differently than for the other algorithms.

❑

❑ Technical note

If there are no alternative-specific variables in your model, the variance–covariance matrix parameters are not identifiable. For such a model to converge, you would therefore 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**.

❑

Stored results

asmprobit stores the following in e():

Scalars

e(N)	number of observations
e(N_case)	number of cases
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(N_clust)	number of clusters
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)	significance
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)	asmprobit
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(indvars)	alternative-specific independent variable
e(casevars)	case-specific variables
e(case)	variable defining cases
e(altvar)	variable defining alternatives
e(alteqs)	alternative equation names
e(alt#)	alternative labels
e(wtype)	weight type
e(wexp)	weight expression
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(correlation)	correlation structure
e(stddev)	variance structure
e(cov_class)	class of the covariance 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_seed)	random-number generator seed
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(mfx_dlg)	program used to implement estat mfx dialog
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
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(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

Methods and formulas

The simulated maximum likelihood estimates for the MNP are obtained using ml; see [R] **ml**. The likelihood evaluator implements the 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 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, $\boldsymbol{\epsilon}_i = (\epsilon_{i1}, \ldots, \epsilon_{i,\mathrm{J}-1}) \sim \mathrm{MVN}(\mathbf{0}, \boldsymbol{\Sigma}_{(k)})$. $\boldsymbol{\Sigma}$ 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

$$\begin{aligned} \Pr(y_i = k) &= \Pr(v_{i1} \le 0, \ldots, v_{i,J-1} \le 0) \\ &= \Pr(\epsilon_{i1} \le -\lambda_{i1}, \ldots, \epsilon_{i,J-1} \le -\lambda_{i,J-1}) \\ &= (2\pi)^{-(J-1)/2} \left|\boldsymbol{\Sigma}_{(k)}\right|^{-1/2} \int_{-\infty}^{-\lambda_{i1}} \cdots \int_{-\infty}^{-\lambda_{i,J-1}} \exp\left(-\tfrac{1}{2}\mathbf{z}'\boldsymbol{\Sigma}_{(k)}^{-1}\mathbf{z}\right) d\mathbf{z} \end{aligned} \tag{3}$$

Simulated likelihood

For clarity in the discussion that follows, we drop the index denoting case so that for an arbitrary observation $\boldsymbol{v}' = (v_1, \ldots, v_{J-1})$, $\boldsymbol{\lambda}' = (\lambda_1, \ldots, \lambda_{J-1})$, 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 latent-variable 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 $\mathrm{N}(0, 1)$. We now have $\boldsymbol{v} = \boldsymbol{\lambda} + \mathbf{L}\boldsymbol{\zeta}$, and by defining

$$z_j = \begin{cases} -\dfrac{\lambda_1}{l_{11}} & \text{for } j = 1 \\ -\dfrac{\lambda_j + \sum_{i=1}^{j-1} l_{ji}\zeta_i}{l_{jj}} & \text{for } j = 2, \ldots, J-1 \end{cases} \tag{4}$$

we can express the probability statement (3) as the product of conditional probabilities

$$\begin{aligned} \Pr(y_i = k) = \Pr\left(\zeta_1 \le z_1\right) \Pr\left(\zeta_2 \le z_2 \mid \zeta_1 \le z_1\right) \cdots \\ \Pr\left(\zeta_{J-1} \le z_{J-1} \mid \zeta_1 \le z_1, \ldots, \zeta_{J-2} \le z_{J-2}\right) \end{aligned}$$

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 (4) 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 simulated log 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

$$|\widehat{L}_i - L_i| \le V(L_i) D_N\{(\delta_i)\}$$

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 pseudorandom sequence, δ_i, the discrepancy is of order $O\{(\log\log N/N)^{1/2}\}$. The order of discrepancy can be improved by using quasirandom sequences.

Quasi–Monte Carlo integration is carried out by `asmprobit` 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. `asmprobit` 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` *casevar*`)`, where *casevar* 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.

Bunch, D. S. 1991. Estimability of the multinomial probit model. *Transportation Research Part B* 25: 1–12.

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 & 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.

Geweke, J. 1989. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica* 57: 1317–1339.

Geweke, J., and M. P. Keane. 2001. Computationally intensive methods for integration in econometrics. In Vol. 5 of *Handbook of Econometrics*, ed. J. Heckman and E. Leamer, 3463–3568. Amsterdam: Elsevier.

Greene, W. H. 2012. *Econometric Analysis*. 7th ed. Upper Saddle River, NJ: Prentice Hall.

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.

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.

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

Also see

[R] **asmprobit postestimation** — Postestimation tools for asmprobit

[R] **asclogit** — Alternative-specific conditional logit (McFadden's choice) model

[R] **asroprobit** — Alternative-specific rank-ordered probit regression

[R] **mlogit** — Multinomial (polytomous) logistic regression

[R] **mprobit** — Multinomial probit regression

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