Title

stata.com

cmroprobit — Rank-ordered probit choice model

Description

`cmroprobit` fits rank-ordered probit (ROP) models by using maximum simulated likelihood (MSL). The model allows you to relax the independence of irrelevant alternatives (IIA) property that is characteristic of the rank-ordered logistic model by estimating covariances between the error terms for the alternatives.

`cmroprobit` allows two types of independent variables: alternative-specific variables, in which the values of each variable vary with each alternative, and case-specific variables, which vary with each case.

The estimation technique of `cmroprobit` is nearly identical to that of `cmmprobit`, and the two routines share many of the same options; see [CM] **cmmprobit**.

Quick start

Rank-ordered probit model of rankings `y` as a function of `x1` using `cmset` data

```
cmroprobit y x1
```

Same as above, but interpret the lowest value of `y` as the best

```
cmroprobit y x1, reverse
```

Same as above, and include case-specific covariate `x2`

```
cmroprobit y x1, casevars(x2) reverse
```

Same as above, but with factor covariance structure of dimension 1

```
cmroprobit y x1, casevars(x2) reverse factor(1)
```

With all correlations of utility errors constrained to 0

```
cmroprobit y x1, correlation(independent)
```

With a common correlation parameter for all pairs of alternatives

```
cmroprobit y x1, correlation(exchangeable)
```

Menu

Statistics > Choice models > Rank-ordered probit model

Syntax

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

options	Description
Model	
casevars(*varlist*)	case-specific variables
reverse	interpret the lowest rank in *depvar* as the best; the default is the highest rank is the best
basealternative(# | *lbl* | *str*)	alternative used for normalizing location
scalealternative(# | *lbl* | *str*)	alternative used for normalizing scale
noconstant	suppress the alternative-specific constant terms
altwise	use alternativewise deletion instead of casewise deletion
constraints(*constraints*)	apply specified linear constraints
Model 2	
correlation(*correlation*)	correlation structure of the utility errors
stddev(*stddev*)	variance structure of the utility errors
factor(#)	use the factor covariance structure with dimension #
structural	use the structural covariance parameterization; default is the differenced covariance parameterization
SE/Robust	
vce(*vcetype*)	*vcetype* may be oim, robust, cluster *clustvar*, opg, bootstrap, or jackknife
Reporting	
level(#)	set confidence level; default is level(95)
notransform	do not transform variance–covariance estimates to the standard deviation and correlation metric
nocnsreport	do not display constraints
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
intmethod(*seqtype*)	type of quasi–uniform or pseudo–uniform sequence
intpoints(#)	number of points in each sequence
intburn(#)	starting index in the Hammersley or Halton sequence
intseed(*code* | #)	pseudo–uniform random-number seed
antithetics	use antithetic draws
nopivot	do not use integration interval pivoting
initbhhh(#)	use the BHHH optimization algorithm for the first # iterations
favor(speed | space)	favor speed or space when generating integration points

Maximization

maximize_options	control the maximization process; seldom used
collinear	keep collinear variables
coeflegend	display legend instead of statistics

correlation	Description
unstructured	one correlation parameter for each pair of alternatives; correlations with the basealternative() are zero; the default
exchangeable	one correlation parameter common to all pairs of alternatives; correlations with the basealternative() are zero
independent	constrain all correlation parameters to zero
pattern *matname*	user-specified matrix identifying the correlation pattern
fixed *matname*	user-specified matrix identifying the fixed and free correlation parameters

stddev	Description
heteroskedastic	estimate standard deviation for each alternative; standard deviations for basealternative() and scalealternative() set to one
homoskedastic	all standard deviations are one
pattern *matname*	user-specified matrix identifying the standard deviation pattern
fixed *matname*	user-specified matrix identifying the fixed and free standard deviations

seqtype	Description
hammersley	Hammersley point set
halton	Halton point set
random	uniform pseudo–random point set

You must cmset your data before using cmroprobit; see [CM] **cmset**.
indepvars and *varlist* may contain factor variables; see [U] **11.4.3 Factor variables**.
bootstrap, by, collect, jackknife, and statsby are allowed; see [U] **11.1.10 Prefix commands**.
Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.
fweights, iweights, and pweights are allowed; see [U] **11.1.6 weight**.
collinear and coeflegend do not appear in the dialog box.
See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

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

reverse directs cmroprobit to interpret the rank in *depvar* that is smallest in value as the most preferred alternative. By default, the rank that is largest in value is the most preferred alternative.

`basealternative(`*# | lbl | str*`)` specifies the alternative used to normalize the level of utility. The base alternative may be specified as a number when the alternatives variable is numeric, as a label when it is numeric and has a value label, or as a string when it is a string variable. The standard deviation for the utility error associated with the base alternative is fixed to one, and its correlations with all other utility errors are set to zero. The default is the first alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` and `correlation()` options, the `basealternative()` will be implied by the fixed standard deviations and correlations in the matrix specifications. `basealternative()` cannot be equal to `scalealternative()`.

`scalealternative(`*# | lbl | str*`)` specifies the alternative used to normalize the scale of the utility. The scale alternative may be specified as a number, label, or string. The default is to use the second alternative when sorted. If a `fixed` or `pattern` matrix is given in the `stddev()` option, the `scalealternative()` will be implied by the fixed standard deviations in the matrix specification. `scalealternative()` cannot be equal to `basealternative()`.

If a `fixed` or `pattern` matrix is given for the `stddev()` option, the base alternative and scale alternative are implied by the standard deviations and correlations in the matrix specifications, and they need not be specified in the `basealternative()` and `scalealternative()` options.

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

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

`constraints(`*constraints*`)`; see [R] **Estimation options**.

Model 2

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

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

`correlation(exchangeable)` provides for one correlation coefficient common to all utilities, except the utility associated with the `basealternative()` option.

`correlation(independent)` assumes that all correlations are zero.

`correlation(pattern` *matname*`)` and `correlation(fixed` *matname*`)` give you more flexibility in defining the correlation structure. See *Covariance structures* in [CM] **cmmprobit** for more information.

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

`stddev(heteroskedastic)` is the most general and has $J-2$ estimable parameters. The standard deviations of the utility errors for the alternatives specified in `basealternative()` and `scalealternative()` are fixed to one.

`stddev(homoskedastic)` constrains all the standard deviations to equal one.

`stddev(pattern` *matname*`)` and `stddev(fixed` *matname*`)` give you added flexibility in defining the standard deviation parameters. See *Covariance structures* in [CM] **cmmprobit** for more information.

`factor(#)` requests that the factor covariance structure of dimension # be used. The `factor()` option can be used with the `structural` option but cannot be used with `stddev()` or `correlation()`. A $\# \times J$ (or $\# \times J-1$) matrix, $\mathbf{C}$, is used to factor the covariance matrix as $I+\mathbf{C}'\mathbf{C}$, where

I is the identity matrix of dimension J (or $J-1$). The column dimension of $\mathbf{C}$ depends on whether the covariance is structural or differenced. The row dimension of $\mathbf{C}$, #, must be less than or equal to floor$[\{J(J-1)/2-1\}/(J-2)]$, because there are only $J(J-1)/2-1$ identifiable covariance parameters. This covariance parameterization may be useful for reducing the number of covariance parameters that need to be estimated.

If the covariance is structural, the column of $\mathbf{C}$ corresponding to the base alternative contains zeros. The column corresponding to the scale alternative has a one in the first row and zeros elsewhere. If the covariance is differenced, the column corresponding to the scale alternative (differenced with the base) has a one in the first row and zeros elsewhere.

`structural` requests the $J \times J$ structural covariance parameterization instead of the default $J-1 \times J-1$ differenced covariance parameterization (the covariance of the utility errors differenced with that of the base alternative). The differenced covariance parameterization will achieve the same MSL regardless of the choice of `basealternative()` and `scalealternative()`. On the other hand, the structural covariance parameterization imposes more normalizations that may bound the model away from its maximum likelihood and thus prevent convergence with some datasets or choices of `basealternative()` and `scalealternative()`.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), that allow for intragroup correlation (`cluster` *clustvar*), and that use bootstrap or jackknife methods (`bootstrap`, `jackknife`); see [R] ***vce_option***.

If specifying `vce(bootstrap)` or `vce(jackknife)`, you must also specify `basealternative()` and `scalealternative()`.

Reporting

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

`notransform` prevents retransforming the Cholesky-factored covariance estimates to the correlation and standard deviation metric. `notransform` may not be specified on replay.

This option has no effect if `structural` is not specified because the default differenced covariance estimates have no interesting interpretation as correlations and standard deviations. `notransform` also has no effect if the `correlation()` and `stddev()` options are specified with anything other than their default values. Here it is generally not possible to factor the covariance matrix, so optimization is already performed using the standard deviation and correlation representations.

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

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

Integration

`intmethod(hammersley | halton | random)` specifies the method of generating the point sets used in the quasi–Monte Carlo integration of the multivariate normal density. `intmethod(hammersley)`, the default, uses the Hammersley sequence; `intmethod(halton)` uses the Halton sequence; and `intmethod(random)` uses a sequence of uniform random numbers.

`intpoints(#)` specifies the number of points to use in the Monte Carlo integration. If option `intmethod(hammersley)` or `intmethod(halton)` is used, the default is 500 +

floor$[2.5\sqrt{N_c\{\ln(k+5)+v\}}]$, where N_c is the number of cases, k is the number of coefficients in the model, and v is the number of variance parameters. If `intmethod(random)` is used, the number of points is the above times 2. Larger values of `intpoints()` provide better approximations of the log likelihood at the cost of additional computation time.

`intburn(#)` specifies where in the Hammersley or Halton sequence to start, which helps reduce the correlation between the sequences of each dimension. The default is `intburn(0)`. This option may not be specified with `intmethod(random)`.

`intseed(`*code*`|#)` specifies the seed to use for generating the uniform pseudo–random sequence. This option may be specified only with `intmethod(random)`. *code* refers to a string that records the state of the random-number generator `runiform()`; see [R] **set seed**. An integer value # may be used also. The default is to use the current seed value from Stata's uniform random-number generator, which can be obtained from `c(rngstate)`.

`antithetics` specifies that antithetic draws be used. The antithetic draw for the $J-1$ vector uniform-random variables, $\mathbf{x}$, is $1-\mathbf{x}$.

`nopivot` turns off integration interval pivoting. By default, `cmroprobit` will pivot the wider intervals of integration to the interior of the multivariate integration. This improves the accuracy of the quadrature estimate. However, discontinuities may result in the computation of numerical second-order derivatives using finite differencing (for the Newton–Raphson optimize technique, `tech(nr)`) when few simulation points are used, resulting in a non–positive-definite Hessian. `cmroprobit` uses the Broyden–Fletcher–Goldfarb–Shanno optimization algorithm, by default, which does not require computing the Hessian numerically using finite differencing.

`initbhhh(#)` specifies that the Berndt–Hall–Hall–Hausman (BHHH) algorithm be used for the initial # optimization steps. This option is the only way to use the BHHH algorithm along with other optimization techniques. The algorithm switching feature of `ml`'s `technique()` option cannot include `bhhh`.

`favor(speed|space)` instructs `cmroprobit` to favor either speed or space when generating the integration points. `favor(speed)` is the default. When favoring speed, the integration points are generated once and stored in memory, thus increasing the speed of evaluating the likelihood. This speed increase can be seen when there are many cases or when the user specifies many integration points, `intpoints(#)`. When favoring space, the integration points are generated repeatedly with each likelihood evaluation.

For unbalanced data, where the number of alternatives varies with each case, the estimates computed using `intmethod(random)` will vary slightly between `favor(speed)` and `favor(space)`. This is because the uniform sequences will not be identical, even when initiating the sequences using the same uniform seed, `intseed(`*code*`|#)`. For `favor(speed)`, `ncase` blocks of `intpoints(#)` $\times$ $J-2$ uniform points are generated, where J is the maximum number of alternatives. For `favor(space)`, the column dimension of the matrices of points varies with the number of alternatives that each case has.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, and `from(`*init_specs*`)`; see [R] **Maximize**.

The following options may be particularly useful in obtaining convergence with `cmroprobit`: `difficult`, `technique(`*algorithm_spec*`)`, `nrtolerance(#)`, `nonrtolerance`, and `from(`*init_specs*`)`.

If `technique()` contains more than one algorithm specification, `bhhh` cannot be one of them. To use the BHHH algorithm with another algorithm, use the `initbhhh()` option, and specify the other algorithm in `technique()`.

Setting the optimization type to `technique(bhhh)` resets the default *vcetype* to `vce(opg)`.

The default optimization technique is `technique(bfgs)`.

When you specify `from(`*matname* [`, copy`]`)`, the values in *matname* associated with the utility error variances must be for the log-transformed standard deviations and inverse-hyperbolic tangent-transformed correlations. This option makes using the coefficient vector from a previously fitted `cmroprobit` model convenient as a starting point.

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

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

Remarks and examples

stata.com

The mathematical description and numerical computations of the rank-ordered probit model are similar to that of the multinomial probit model. The only difference is that the dependent variable of the rank-ordered probit model is ordinal, showing preferences among alternatives, as opposed to the binary dependent variable of the multinomial probit model, indicating a chosen alternative.

Only the order in the ranks, not the magnitude of their differences, is assumed to be relevant. By default, the largest rank indicates the more desirable alternative. Use the `reverse` option if the lowest rank should be interpreted as the more desirable alternative.

Tied ranks are allowed, but they increase the computation time because all permutations of the tied ranks are used in computing the likelihood for each case.

We will describe how the likelihood of a ranking is computed using the utility (latent-variable) framework here, but for details of the utility parameterization of these models and the method of maximum simulated likelihood, see [CM] **cmmprobit**.

Consider the utility of a J alternative rank-ordered probit model. Using the notation from `cmmprobit`, we have utilities (latent variables) η_{ij}, $j = 1, \ldots, J$, such that

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

Here the $\mathbf{x}_{ij}$ are the alternative-specific independent variables, the $\mathbf{z}_i$ are the case-specific variables, and the ξ_{ij} are multivariate normal with mean zero and covariance $\boldsymbol{\Omega}$. Without loss of generality, assume that individual i ranks the alternatives in order of the alternative indices $j = 1, 2, \ldots, J$, so the alternative J is the preferred alternative, and alternative 1 is the least preferred alternative. The probability of this ranking given $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}_j$ is the probability that $\eta_{i,J-1} - \eta_{i,J} \leq 0$ and $\eta_{i,J-2} - \eta_{i,J-1} \leq 0$, …, and $\eta_{i,1} - \eta_{i,2} \leq 0$.

▷ Example 1

Long and Freese (2014, 477) provide an example of a rank-ordered logit model with alternative-specific variables. We use this dataset to demonstrate `cmroprobit`. The data come from the Wisconsin Longitudinal Study. This is a study of 1957 Wisconsin high school graduates who were asked to

report their relative preferences of four job characteristics: esteem, a job other people regard highly; variety, a job that is not repetitive and allows you to do a variety of things; autonomy, a job where your supervisor does not check on you frequently; and security, a job with a low risk of being laid off.

The case-specific covariates are gender, `female`, an indicator variable for females, and `score`, a score on a general mental ability test measured in standard deviations. We also have alternative-specific variables `high` and `low`, which indicate whether the respondent's current job is high or low in esteem, variety, autonomy, or security. When taken together, `high` and `low` provide three states for a respondent's current job status for each alternative—high, low, or neither. From these variables, we create a new variable, `currentjob`, that we include as an alternative-specific variable in our model.

The alternatives are ranked such that 1 is the preferred alternative and 4 is the least preferred.

```
. use https://www.stata-press.com/data/r18/wlsrank
(1992 Wisconsin Longitudinal Study data on job values)
. generate currentjob = 1 if low==1
(10,264 missing values generated)
. replace currentjob = 2 if low==0 & high==0
(6,319 real changes made)
. replace currentjob = 3 if high==1
(3,945 real changes made)
. label define current 1 "Low" 2 "Neither" 3 "High"
. label values currentjob current
. list id jobchar rank female score currentjob in 1/12, sepby(id)
```

	id	jobchar	rank	female	score	curren~b
1.	1	Esteem	3	Female	.0492111	Neither
2.	1	Variety	1	Female	.0492111	Neither
3.	1	Autonomy	4	Female	.0492111	Neither
4.	1	Security	1	Female	.0492111	Neither
5.	5	Esteem	2	Female	2.115012	High
6.	5	Variety	2	Female	2.115012	High
7.	5	Autonomy	1	Female	2.115012	Neither
8.	5	Security	2	Female	2.115012	High
9.	7	Esteem	4	Male	1.701852	Neither
10.	7	Variety	1	Male	1.701852	Low
11.	7	Autonomy	1	Male	1.701852	High
12.	7	Security	1	Male	1.701852	Neither

The three cases listed have tied ranks. `cmroprobit` will allow ties but at the cost of increased computation time. To evaluate the likelihood of the first observation, `cmroprobit` must compute

$$\Pr(\texttt{Esteem} = 3,\ \texttt{Variety} = 1,\ \texttt{Autonomy} = 4,\ \texttt{Security} = 2) +$$
$$\Pr(\texttt{Esteem} = 3,\ \texttt{Variety} = 2,\ \texttt{Autonomy} = 4,\ \texttt{Security} = 1)$$

and both of these probabilities are estimated using simulation. In fact, the full dataset contains 7,237 tied ranks, and `cmroprobit` takes a long time to estimate the parameters. For exposition, we fit the rank-ordered probit model by using the cases without ties. These cases are marked in the variable `noties`.

The model of job preference is

$$\eta_{ij} = \beta_1 \texttt{Neither}_{ij} + \beta_2 \texttt{High}_{ij} \; + \; \alpha_{1j}\texttt{female}_i + \alpha_{2j}\texttt{score}_i + \alpha_{0j} + \xi_{ij}$$

for $j = 1, 2, 3, 4$. The base alternative will be `Esteem`, so $\alpha_{01} = \alpha_{11} = \alpha_{21} = 0$.

Before we can fit our model, we must `cmset` our data. The first argument to `cmset` is the case ID variable. For these data, it is the variable `id`, which identifies respondents. The second argument is the alternatives variable, which identifies the alternatives that could have been chosen. In this instance, it is the variable `jobchar`, which gives the four job characteristics: esteem, variety, autonomy, and security.

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

We fit our model:

```
. cmroprobit rank i.currentjob if noties, casevars(i.female score) reverse
note: variable 2.currentjob has 69 cases that are not alternative-specific;
      there is no within-case variability.
note: variable 3.currentjob has 107 cases that are not alternative-specific;
      there is no within-case variability.
Iteration 0:  Log simulated-likelihood = -1102.9667
Iteration 1:  Log simulated-likelihood = -1089.1146  (backed up)
Iteration 2:  Log simulated-likelihood = -1085.7877  (backed up)
Iteration 3:  Log simulated-likelihood = -1083.0085  (backed up)
Iteration 4:  Log simulated-likelihood = -1082.5081  (backed up)
Iteration 5:  Log simulated-likelihood = -1082.1977  (backed up)
Iteration 6:  Log simulated-likelihood = -1082.1208  (backed up)
Iteration 7:  Log simulated-likelihood = -1082.0995
Iteration 8:  Log simulated-likelihood = -1082.0442
Iteration 9:  Log simulated-likelihood = -1081.8316  (backed up)
Iteration 10: Log simulated-likelihood = -1081.6816
Iteration 11: Log simulated-likelihood = -1081.5777
Iteration 12: Log simulated-likelihood = -1081.5137
Iteration 13: Log simulated-likelihood = -1081.1495
Iteration 14: Log simulated-likelihood = -1081.0503
Iteration 15: Log simulated-likelihood = -1080.7247
Iteration 16: Log simulated-likelihood = -1080.1485
Iteration 17: Log simulated-likelihood = -1080.0215
Iteration 18: Log simulated-likelihood = -1080.0179
Iteration 19: Log simulated-likelihood = -1079.9916
Iteration 20: Log simulated-likelihood = -1079.9733
Iteration 21: Log simulated-likelihood = -1079.9733
```

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

rank	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
jobchar						
currentjob						
Neither	.0694818	.1092531	0.64	0.525	-.1446503	.2836138
High	.4435911	.121678	3.65	0.000	.2051066	.6820757
Esteem	(base alternative)					
Variety						
female						
Female	.1354259	.1843808	0.73	0.463	-.2259537	.4968055
score	.14071	.0977659	1.44	0.150	-.0509077	.3323278
_cons	1.734496	.1449852	11.96	0.000	1.45033	2.018661
Autonomy						
female						
Female	.2562822	.1645938	1.56	0.119	-.0663158	.5788801
score	.189963	.0873586	2.17	0.030	.0187433	.3611827
_cons	.7007559	.1203087	5.82	0.000	.4649553	.9365566
Security						
female						
Female	.2326342	.2055864	1.13	0.258	-.1703079	.6355762
score	-.1779831	.1101985	-1.62	0.106	-.3939682	.0380021
_cons	1.34348	.1598787	8.40	0.000	1.030123	1.656836
/lnl2_2	.1813086	.0756934	2.40	0.017	.0329522	.329665
/lnl3_3	.4843953	.0793885	6.10	0.000	.3287967	.639994
/l2_1	.6060303	.1158474	5.23	0.000	.3789735	.8330871
/l3_1	.4491585	.1435157	3.13	0.002	.167873	.7304441
/l3_2	.2305503	.121713	1.89	0.058	-.0080029	.4691034

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

We specified the `reverse` option because a rank of 1 is the highest preference. The variance–covariance estimates are for the Cholesky-factored variance–covariance for the utility errors differenced with that of alternative `Esteem`. We can view the estimated correlations by typing

```
. estat correlation
```

	Variety	Autonomy	Security
Variety	1.0000		
Autonomy	0.4512	1.0000	
Security	0.2642	0.2402	1.0000

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

and typing

```
. estat covariance
```

	Variety	Autonomy	Security
Variety	2		
Autonomy	.8570563	1.804358	
Security	.6352061	.5485839	2.889653

```
Note: Covariances are for alternatives differenced with Esteem.
```

gives the variances and covariances. In [R] **mprobit**, we explain that if the utility errors are independent, then the correlations in the differenced parameterization should be ∼0.5, and the variances should be ∼2.0, which seems to be the case here.

The coefficient estimates for the probit models can be difficult to interpret because of the normalization for location and scale. The regression estimates for the case-specific variables will be relative to the base alternative, and the regression estimates for both the case-specific and alternative-specific variables are affected by the scale normalization. The more pronounced the heteroskedasticity and correlations, the more pronounced the resulting estimate differences when choosing alternatives to normalize for location and scale. However, when using the differenced covariance structure, you will obtain the same model likelihood regardless of which alternatives you choose as the base and scale alternatives.

For model interpretation, you can obtain predicted probabilities using `predict`; see [CM] **cmroprobit postestimation**. You can also use `margins` to estimate expected probabilities, marginal effects, and more. See [CM] **Intro 1** and [CM] **margins** for more information and examples using `margins` for interpretation of the results of `cm` commands. Also, see [CM] **Intro 6** for an example after `cmroprobit`.

After you have fit what you consider your final model, you should run the same model again, but this time setting `intpoints(#)`, the number of integration points in the simulated likelihood to a larger number. In this example, we see from the header that the default number of points was 642. We would run our model again using, say, 2000 points and see by how much the coefficient or covariance parameter estimates change. If changes are small compared with standard errors, we can have confidence in the numerical soundness of the simulation used to compute the likelihood. See *Setting the number of integration points* in [CM] **Intro 5** for more information.

◁

Stored results

`cmroprobit` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(N_case)`	number of cases
`e(N_ties)`	number of ties
`e(N_ic)`	N for Bayesian information criterion (BIC)
`e(N_clust)`	number of clusters
`e(k)`	number of parameters
`e(k_alt)`	number of alternatives
`e(k_indvars)`	number of alternative-specific variables
`e(k_casevars)`	number of case-specific variables
`e(k_sigma)`	number of variance estimates
`e(k_rho)`	number of correlation estimates
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test

e(df_m)	model degrees of freedom
e(ll)	log simulated-likelihood
e(const)	constant indicator
e(i_base)	base alternative index
e(i_scale)	scale alternative index
e(mc_points)	number of Monte Carlo replications
e(mc_burn)	starting sequence index
e(mc_antithetics)	antithetics indicator
e(reverse)	1 if minimum rank is best, 0 if maximum rank is best
e(chi2)	χ^2
e(p)	p-value for model test
e(fullcov)	unstructured covariance indicator
e(structcov)	1 if structured covariance, 0 otherwise
e(cholesky)	Cholesky-factored covariance indicator
e(alt_min)	minimum number of alternatives
e(alt_avg)	average number of alternatives
e(alt_max)	maximum number of alternatives
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	cmroprobit
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(caseid)	name of case ID variable
e(altvar)	name of alternatives variable
e(alteqs)	alternative equation names
e(alt#)	alternative labels
e(wtype)	weight type
e(wexp)	weight expression
e(marktype)	casewise or altwise, type of markout
e(key_N_ic)	cases, key for N for Bayesian information criterion (BIC)
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(correlation)	correlation structure
e(stddev)	variance structure
e(chi2type)	Wald, type of model χ^2 test
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(mc_method)	technique used to generate sequences
e(mc_rngstate)	random-number state used
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(marginsdefault)	default predict() specification for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(stats)	alternative statistics
e(stdpattern)	variance pattern
e(stdfixed)	fixed and free standard deviations

e(altvals)	alternative values
e(altfreq)	alternative frequencies
e(alt_casevars)	indicators for estimated case-specific coefficients—e(k_alt)×e(k_casevars)
e(corpattern)	correlation structure
e(corfixed)	fixed and free correlations
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

From a computational perspective, cmroprobit is similar to cmmprobit, and the two programs share many numerical tools. Therefore, we will use the notation from *Methods and formulas* in [CM] **cmmprobit** to discuss the rank-ordered probit probability model.

The utilities (latent variables) for a J-alternative model are $\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\alpha}_j + \xi_{ij}$, for $j = 1, \ldots, J$, $i = 1, \ldots, n$, and $\boldsymbol{\xi}_i' = (\xi_{i,1}, \ldots, \xi_{i,\mathrm{J}}) \sim \mathrm{MVN}(\mathbf{0}, \boldsymbol{\Omega})$. Without loss of generality, assume for the ith observation that an individual ranks the alternatives in the order of their numeric indices, $\mathbf{y}_i = (J, J-1, \ldots, 1)$, so the first alternative is the most preferred, and the last alternative is the least preferred. We can then difference the utilities such that

$$\begin{aligned} v_{ik} &= \eta_{i,k+1} - \eta_{i,k} \\ &= (\mathbf{x}_{i,k+1} - \mathbf{x}_{i,k})\boldsymbol{\beta} + \mathbf{z}_i(\boldsymbol{\alpha}_{k+1} - \boldsymbol{\alpha}_k) + \xi_{i,k+1} - \xi_{ik} \\ &= \boldsymbol{\delta}_{ik}\boldsymbol{\beta} + \mathbf{z}_i\boldsymbol{\gamma}_k + \epsilon_{ik} \end{aligned}$$

for $k = 1, \ldots, J-1$ and where $\boldsymbol{\epsilon}_i = (\epsilon_{i1}, \ldots, \epsilon_{i,\mathrm{J}-1}) \sim \mathrm{MVN}(\mathbf{0}, \boldsymbol{\Sigma}_{(i)})$. $\boldsymbol{\Sigma}$ is indexed by i because it is specific to the ranking of individual i. We denote the deterministic part of the model as $\lambda_{ik} = \boldsymbol{\delta}_{ik}\boldsymbol{\beta} + \mathbf{z}_j\boldsymbol{\gamma}_k$, and the probability of this event is

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

The integral has the same form as (3) in [CM] **cmmprobit**. See [CM] **cmmprobit** for details on evaluating this integral numerically by using simulation.

`cmroprobit` handles tied ranks by enumeration. For k tied ranks, it will generate $k!$ rankings, where ! is the factorial operator $k! = k(k-1)(k-2)\cdots(2)(1)$. For two sets of tied ranks of size k_1 and k_2, `cmroprobit` will generate $k_1!k_2!$ rankings. The total probability is the sum of the probability of each ranking. For example, if there are two tied ranks such that $\mathbf{y}_i = (J, J, J-2, \ldots, 1)$, then `cmroprobit` will evaluate $\Pr(\mathbf{y}_i) = \Pr(\mathbf{y}_i^{(1)}) + \Pr(\mathbf{y}_i^{(2)})$, where $\mathbf{y}_i^{(1)} = (J, J-1, J-2, \ldots, 1)$ and $\mathbf{y}_i^{(2)} = (J-1, J, J-2, \ldots, 1)$.

This command supports the clustered version of the Huber/White/sandwich estimator of the variance using `vce(robust)` and `vce(cluster` *clustvar*`)`. See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*. Specifying `vce(robust)` is equivalent to specifying `vce(cluster` *caseid*`)`, where *caseid* is the variable that identifies the cases.

Reference

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

Also see

[CM] **cmroprobit postestimation** — Postestimation tools for cmroprobit

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

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

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

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

[R] **ologit** — Ordered logistic regression

[R] **oprobit** — Ordered probit regression

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