asroprobit — Alternative-specific rank-ordered probit regression

Description

`asroprobit` 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 the variance–covariance parameters of the latent-variable errors. Each unique identifier in the `case()` variable has multiple alternatives identified in the `alternatives()` variable, and *depvar* contains the ranked alternatives made by each case. 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. `asroprobit` 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 `asroprobit` is nearly identical to that of `asmprobit`, and the two routines share many of the same options; see [R] **asmprobit**.

Quick start

Alternative-specific rank-ordered probit model of rankings `y` from alternatives `alts` as a function of `x1` for cases identified by `idvar`

```
asroprobit y x1, case(idvar) alternatives(alts)
```

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

```
asroprobit y x1, case(idvar) alternatives(alts) reverse
```

As above, and include case-specific covariate `x2`

```
asroprobit y x1, case(idvar) alternatives(alts) casevars(x2) reverse
```

With all correlations of latent-variable errors constrained to 0

```
asroprobit y x1, case(idvar) alternatives(alts) correlation(independent)
```

With a common correlation parameter for all pairs of alternatives

```
asroprobit y x1, case(idvar) alternatives(alts) correlation(exchangeable)
```

Menu

Statistics > Ordinal outcomes > Rank-ordered probit regression

Syntax

`asroprobit` *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
`reverse`	interpret the lowest rank in *depvar* as the best; the default is the highest rank is the best
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- or pseudouniform sequence
`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.

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

bootstrap, by, 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**.

coeflegend does not appear in the dialog box.

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

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 available 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 `stddev()` or `structural` are specified.

`correlation(exchangeable)` provides for one correlation coefficient common to all latent variables, except the latent variable associated with the `basealternative()`.

`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* in [R] **asmprobit** 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* in [R] **asmprobit** 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.

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

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: `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 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(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, `asroprobit` 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. `asroprobit` 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 `asroprobit` 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 asroprobit: 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).

When specifying from(*matname* [, copy]), the values in *matname* associated with the latent-variable 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 asroprobit model convenient as a starting point.

The following option is available with asroprobit but is not shown in the dialog box:

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

Consider the latent-variable parameterization of a J alternative rank-ordered probit model. Using the notation from asmprobit, we have variables η_{ij}, $j = 1, \dots, J$, 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 $j = 1, 2, \dots, J$, 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 $\eta_{i,J-1} - \eta_{i,J} \le 0$ and $\eta_{i,J-2} - \eta_{i,J-1} \le 0$, ..., and $\eta_{i,1} - \eta_{i,2} \le 0$.

▷ 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 asroprobit. The data come from the Wisconsin Longitudinal Study. This is a study of 1957 Wisconsin high school graduates that were asked to rate their relative preference 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. The alternative-specific variables are high and low, which indicate whether the respondent's current job is high or low in esteem, variety, autonomy, or security. This approach provides three states for a respondent's current job status for each alternative, $(1,0)$, $(0,1)$, and $(0,0)$, using the notation (high, low). The score $(1,1)$ is omitted because the respondent's current job cannot be considered both high and low in one of the job characteristics. The $(0,0)$ score would indicate that the respondent's current job does not rank high or low (is neutral) in a job characteristic. The alternatives are ranked such that 1 is the preferred alternative and 4 is the least preferred.

```
. use http://www.stata-press.com/data/r15/wlsrank
(1992 Wisconsin Longitudinal Study data on job values)

. list id jobchar rank female score high low in 1/12, sepby(id)

     ┌────────────────────────────────────────────────────────┐
     │ id    jobchar   rank   female       score   high   low │
     ├────────────────────────────────────────────────────────┤
  1. │  1   security      1        1    .0492111      0     0 │
  2. │  1   autonomy      4        1    .0492111      0     0 │
  3. │  1    variety      1        1    .0492111      0     0 │
  4. │  1     esteem      3        1    .0492111      0     0 │
     ├────────────────────────────────────────────────────────┤
  5. │  5   security      2        1    2.115012      1     0 │
  6. │  5    variety      2        1    2.115012      1     0 │
  7. │  5     esteem      2        1    2.115012      1     0 │
  8. │  5   autonomy      1        1    2.115012      0     0 │
     ├────────────────────────────────────────────────────────┤
  9. │  7   autonomy      1        0    1.701852      1     0 │
 10. │  7    variety      1        0    1.701852      0     1 │
 11. │  7     esteem      4        0    1.701852      0     0 │
 12. │  7   security      1        0    1.701852      0     0 │
     └────────────────────────────────────────────────────────┘
```

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

$$\begin{aligned}&\Pr(\texttt{esteem} = 3,\ \texttt{variety} = 1,\ \texttt{autonomy} = 4,\ \texttt{security} = 2)+\\&\Pr(\texttt{esteem} = 3,\ \texttt{variety} = 2,\ \texttt{autonomy} = 4,\ \texttt{security} = 1)\end{aligned}$$

and both of these probabilities are estimated using simulation. In fact, the full dataset contains 7,237 tied ranks and asroprobit takes a great deal of time to estimate the parameters. For exposition, we estimate 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

$$\eta_{ij} = \beta_1 \texttt{high}_{ij} + \beta_2 \texttt{low}_{ij} \ + \ \alpha_{1j}\texttt{female}_i + \alpha_{2j}\texttt{score}_i + \alpha_{0j} + \xi_{ij}$$

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

```
. asroprobit rank high low if noties, case(id) alternatives(jobchar)
> casevars(female score) reverse
note: variable high has 107 cases that are not alternative-specific: there is
      no within-case variability
note: variable low has 193 cases that are not alternative-specific: there is
      no within-case variability
Iteration 0:   log simulated-likelihood = -1103.2768
Iteration 1:   log simulated-likelihood = -1089.3361  (backed up)
  (output omitted )
Alternative-specific rank-ordered probit       Number of obs      =      1,660
Case variable: id                              Number of cases    =        415
Alternative variable: jobchar                  Alts per case: min =          4
                                                              avg =        4.0
                                                              max =          4
Integration sequence:      Hammersley
Integration points:               200             Wald chi2(8)    =      34.01
Log simulated-likelihood = -1080.2206             Prob > chi2     =     0.0000

------------------------------------------------------------------------------
        rank |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
jobchar      |
        high |   .3741029   .0925685     4.04   0.000      .192672    .5555337
         low |  -.0697443   .1093317    -0.64   0.524    -.2840305    .1445419
-------------+----------------------------------------------------------------
esteem       |  (base alternative)
-------------+----------------------------------------------------------------
variety      |
      female |   .1351487   .1843088     0.73   0.463    -.2260899    .4963873
       score |   .1405482   .0977567     1.44   0.151    -.0510515    .3321479
       _cons |   1.735016   .1451343    11.95   0.000     1.450558    2.019474
-------------+----------------------------------------------------------------
autonomy     |
      female |   .2561828   .1679565     1.53   0.127    -.0730059    .5853715
       score |   .1898853   .0875668     2.17   0.030     .0182575     .361513
       _cons |   .7009797   .1227336     5.71   0.000     .4604262    .9415333
-------------+----------------------------------------------------------------
security     |
      female |    .232622   .2057547     1.13   0.258    -.1706497    .6358938
       score |  -.1780076   .1102115    -1.62   0.106    -.3940181     .038003
       _cons |   1.343766   .1600059     8.40   0.000     1.030161    1.657372
-------------+----------------------------------------------------------------
     /lnl2_2 |   .1805151   .0757296     2.38   0.017     .0320878    .3289424
     /lnl3_3 |   .4843091   .0793343     6.10   0.000     .3288168    .6398014
-------------+----------------------------------------------------------------
       /l2_1 |   .6062037   .1169368     5.18   0.000     .3770117    .8353957
       /l3_1 |   .4509217   .1431183     3.15   0.002     .1704151    .7314283
       /l3_2 |   .2289447   .1226081     1.87   0.062    -.0113627    .4692521
------------------------------------------------------------------------------
(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 latent-variable errors differenced with that of alternative `esteem`. We can view the estimated correlations by entering

```
. estat correlation

             │   variety   autonomy   security
─────────────┼──────────────────────────────────
     variety │    1.0000
    autonomy │    0.4516     1.0000
    security │    0.2652     0.2399     1.0000

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

and typing

```
. estat covariance

             │   variety   autonomy   security
─────────────┼──────────────────────────────────
     variety │         2
    autonomy │  .8573015    1.80229
    security │  .6376996   .5475882   2.890048

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

gives the (co)variances. [R] **mprobit** explains that if the latent-variable 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 examine the estimated probabilities and marginal effects by using postestimation routines `predict` and `estat mfx`. See [R] **asroprobit postestimation**.

◁

Stored results

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

Scalars	
`e(N)`	number of observations
`e(N_case)`	number of cases
`e(N_ties)`	number of ties
`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(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)`	`asroprobit`
`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(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)`	`Hammersley`, `Halton`, or `uniform random`; technique 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(mfx_dlg)`	program used to implement `estat mfx` dialog
`e(predict)`	program used to implement `predict`
`e(marginsnotok)`	predictions disallowed by `margins`
`e(asbalanced)`	factor variables `fvset` as `asbalanced`
`e(asobserved)`	factor variables `fvset` as `asobserved`
Matrices	
`e(b)`	coefficient vector
`e(Cns)`	constraints matrix
`e(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

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

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 $j = 1,\dots,J$, $i = 1,\dots,n$, and $\boldsymbol{\xi}_i' = (\xi_{i,1},\dots,\xi_{i,J}) \sim \text{MVN}(\mathbf{0},\boldsymbol{\Omega})$. Without loss of generality, assume for the ith observation that an individual ranks the alternatives in the order of their numeric indices, $\mathbf{y}_i = (J, J-1,\dots,1)$, so the first alternative is the most preferred and the last alternative is the least preferred. We can then difference the latent variables 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 $k = 1,\dots,J-1$ and where $\boldsymbol{\epsilon}_i = (\epsilon_{i1},\dots,\epsilon_{i,J-1}) \sim \text{MVN}(\mathbf{0},\boldsymbol{\Sigma}_{(i)})$. $\boldsymbol{\Sigma}$ 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

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

The integral has the same form as (3) of *Methods and formulas* in [R] **asmprobit**. See [R] **asmprobit** for details on evaluating this integral numerically by using simulation.

`asroprobit` 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, `asroprobit` 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 $\mathbf{y}_i = (J, J, J-2, \dots, 1)$, then `asroprobit` will evaluate $\Pr(\mathbf{y}_i) = \Pr(\mathbf{y}_i^{(1)}) + \Pr(\mathbf{y}_i^{(2)})$, where $\mathbf{y}_i^{(1)} = (J, J-1, J-2, \dots, 1)$ and $\mathbf{y}_i^{(2)} = (J-1, J, J-2, \dots, 1)$.

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` *casevar*`)`, where *casevar* 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

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

[R] **asmixlogit** — Alternative-specific mixed logit regression

[R] **asmprobit** — Alternative-specific multinomial probit regression

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

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

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

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