hetprobit — Heteroskedastic probit model

Description

`hetprobit` fits a maximum-likelihood heteroskedastic probit model.

`hetprob` is a synonym for `hetprobit`.

Quick start

Heteroskedastic probit model of `y` on `x1`, using `x2` to model the variance

```
hetprobit y x1, het(x2)
```

With robust standard errors

```
hetprobit y x1, het(x2) vce(robust)
```

After fitting a model, reprint the table as a coefficient legend

```
hetprobit, coeflegend
```

Menu

Statistics > Binary outcomes > Heteroskedastic probit regression

Syntax

`hetprobit` *depvar* [*indepvars*] [*if*] [*in*] [*weight*] ,
  `het(`*varlist*[, `offset(`*varname$_o$*`)`]`)` [*options*]

options	Description
Model	
* `het(`*varlist*[...]`)`	independent variables to model the variance and possible offset variable
`noconstant`	suppress constant term
`offset(`*varname*`)`	include *varname* in model with coefficient constrained to 1
`asis`	retain perfect predictor variables
`constraints(`*constraints*`)`	apply specified linear constraints
`collinear`	keep collinear variables
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `robust`, `cluster` *clustvar*, `opg`, `bootstrap`, or `jackknife`
Reporting	
`level(`*#*`)`	set confidence level; default is `level(95)`
`lrmodel`	perform the likelihood-ratio model test instead of the default Wald test
`waldhet`	perform Wald test on variance
`nocnsreport`	do not display constraints
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Maximization	
maximize_options	control the maximization process; seldom used
`coeflegend`	display legend instead of statistics

*`het()` is required. The full specification is `het(`*varlist* [, `offset(`*varname$_o$*`)`]`)`.

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

depvar, *indepvars*, and *varlist* may contain time-series operators; see [U] 11.4.4 Time-series varlists.

`bayes`, `bootstrap`, `by`, `fp`, `jackknife`, `rolling`, `statsby`, and `svy` are allowed; see [U] 11.1.10 Prefix commands. For more details, see [BAYES] bayes: hetprobit.

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

`vce()`, `lrmodel`, and weights are not allowed with the `svy` prefix; see [SVY] svy.

`fweight`s, `iweight`s, and `pweight`s 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

`het(`*varlist* [, `offset(`*varname$_o$*`)`]`)` specifies the independent variables and the offset variable, if there is one, in the variance function. `het()` is required.

`offset(`*varname*$_o$`)` specifies that selection offset *varname*$_o$ be included in the model with the coefficient constrained to be 1.

`noconstant`, `offset(`*varname*`)`; see [R] **estimation options**.

`asis` forces the retention of perfect predictor variables and their associated perfectly predicted observations and may produce instabilities in maximization; see [R] **probit**.

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

SE/Robust

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

Reporting

`level(`*#*`)`, `lrmodel`; see [R] **estimation options**.

`waldhet` specifies that a Wald test of whether `lnsigma2` = 0 be performed instead of the LR test.

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

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

Maximization

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

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

The following option is available with `hetprobit` 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

`hetprobit` fits a maximum-likelihood heteroskedastic probit model, which is a generalization of the probit model. Let $y_j, j = 1, \dots, N$, be a binary outcome variable taking on the value 0 (failure) or 1 (success). In the probit model, the probability that y_j takes on the value 1 is modeled as a nonlinear function of a linear combination of the k independent variables $\mathbf{x}_j = (x_{1j}, x_{2j}, \dots, x_{kj})$,

$$\Pr(y_j = 1) = \Phi(\mathbf{x}_j \mathbf{b})$$

in which $\Phi()$ is the cumulative distribution function (CDF) of a standard normal random variable, that is, a normally distributed (Gaussian) random variable with mean 0 and variance 1. The linear combination of the independent variables, $\mathbf{x}_j\mathbf{b}$, is commonly called the *index function*, or *index*. Heteroskedastic probit generalizes the probit model by generalizing $\Phi()$ to a normal CDF with a variance that is no longer fixed at 1 but can vary as a function of the independent variables. `hetprobit` models the variance as a multiplicative function of these m variables $\mathbf{z}_j = (z_{1j}, z_{2j}, \dots, z_{mj})$, following Harvey (1976):

$$\sigma_j^2 = \left\{ \exp(\mathbf{z}_j\boldsymbol{\gamma})\right\}^2$$

Thus the probability of success as a function of all the independent variables is

$$\Pr(y_j = 1) = \Phi\Big\{\mathbf{x}_j\mathbf{b}/\exp(\mathbf{z}_j\boldsymbol{\gamma})\Big\}$$

From this expression, it is clear that, unlike the index $\mathbf{x}_j\mathbf{b}$, no constant term can be present in $\mathbf{z}_j\boldsymbol{\gamma}$ if the model is to be identifiable.

Suppose that the binary outcomes y_j are generated by thresholding an unobserved random variable, w, which is normally distributed with mean $\mathbf{x}_j\mathbf{b}$ and variance 1 such that

$$y_j = \begin{cases} 1 & \text{if } w_j > 0 \\ 0 & \text{if } w_j \le 0 \end{cases}$$

This process gives the probit model:

$$\Pr(y_j = 1) = \Pr(w_j > 0) = \Phi(\mathbf{x}_j\mathbf{b})$$

Now suppose that the unobserved w_j are heteroskedastic with variance

$$\sigma_j^2 = \left\{ \exp(\mathbf{z}_j\boldsymbol{\gamma})\right\}^2$$

Relaxing the homoskedastic assumption of the probit model in this manner yields our multiplicative heteroskedastic probit model:

$$\Pr(y_j = 1) = \Phi\Big\{\mathbf{x}_j\mathbf{b}/\exp(\mathbf{z}_j\boldsymbol{\gamma})\Big\}$$

▷ Example 1

For this example, we generate simulated data for a simple heteroskedastic probit model and then estimate the coefficients with `hetprobit`:

```
. set obs 1000
number of observations (_N) was 0, now 1,000
. set seed 1234567
. generate x = 1-2*runiform()
. generate xhet = runiform()
. generate sigma = exp(1.5*xhet)
. generate p = normal((0.3+2*x)/sigma)
. generate y = cond(runiform()<=p,1,0)
. hetprobit y x, het(xhet)
Fitting probit model:
Iteration 0:   log likelihood = -688.33746
Iteration 1:   log likelihood = -610.48362
Iteration 2:   log likelihood =  -610.3626
Iteration 3:   log likelihood =  -610.3626
Fitting full model:
Iteration 0:   log likelihood =  -610.3626
Iteration 1:   log likelihood =  -600.8767
Iteration 2:   log likelihood = -600.10154
Iteration 3:   log likelihood = -600.01544
Iteration 4:   log likelihood = -600.01521
Iteration 5:   log likelihood = -600.01521
Heteroskedastic probit model                    Number of obs     =      1,000
                                                Zero outcomes     =        451
                                                Nonzero outcomes  =        549
                                                Wald chi2(1)      =      54.20
Log likelihood = -600.0152                      Prob > chi2       =     0.0000
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
y						
x	1.782479	.2421117	7.36	0.000	1.307949	2.257009
_cons	.3140616	.0871121	3.61	0.000	.1433249	.4847982
lnsigma2						
xhet	1.31152	.3011689	4.35	0.000	.7212402	1.901801

```
LR test of lnsigma2=0: chi2(1) = 20.69                    Prob > chi2 = 0.0000
```

Above we created two variables, `x` and `xhet`, and then simulated the model

$$\Pr(\mathtt{y}=1) = F\Big\{(\beta_0+\beta_1\mathtt{x})/\exp(\gamma_1\mathtt{xhet})\Big\}$$

for $\beta_0 = 0.3$, $\beta_1 = 2$, and $\gamma_1 = 1.5$. According to `hetprobit`'s output, all coefficients are significant, and, as we would expect, the Wald test of the full model versus the constant-only model—for example, the index consisting of $\beta_0 + \beta_1\mathtt{x}$ versus that of just β_0—is significant with $\chi^2(1) = 54$. Likewise, the likelihood-ratio test of heteroskedasticity, which tests the full model with heteroskedasticity against the full model without, is significant with $\chi^2(1) = 21$. See [R] **maximize** for more explanation of the output. For this simple model, `hetprobit` took five iterations to converge. As stated elsewhere (Greene 2018, 764), this is a difficult model to fit, and it is not uncommon for it to require many iterations or for the optimizer to print out warnings and informative messages during the optimization. Slow convergence is especially common for models in which one or more of the independent variables appear in both the index and variance functions. ◁

❑ Technical note

Stata interprets a value of 0 as a negative outcome (failure) and treats all other values (except missing) as positive outcomes (successes). Thus if your dependent variable takes on the values 0 and 1, then 0 is interpreted as failure and 1 as success. If your dependent variable takes on the values 0, 1, and 2, then 0 is still interpreted as failure, but both 1 and 2 are treated as successes.

❑

Robust standard errors

If you specify the `vce(robust)` option, `hetprobit` reports robust standard errors as described in [U] **20.22 Obtaining robust variance estimates**. To illustrate the effect of this option, we will reestimate our coefficients by using the same model and data in our example, this time adding `vce(robust)` to our `hetprobit` command.

▷ Example 2

```
. hetprobit y x, het(xhet) vce(robust) nolog
Heteroskedastic probit model                      Number of obs     =      1,000
                                                  Zero outcomes     =        451
                                                  Nonzero outcomes  =        549
                                                  Wald chi2(1)      =      50.49
Log pseudolikelihood = -600.0152                  Prob > chi2       =     0.0000
```

y	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
y						
x	1.782479	.2508447	7.11	0.000	1.290832	2.274126
_cons	.3140616	.087195	3.60	0.000	.1431625	.4849607
lnsigma2						
xhet	1.31152	.3059137	4.29	0.000	.7119406	1.9111

```
Wald test of lnsigma2=0: chi2(1) = 18.38                  Prob > chi2 = 0.0000
```

The `vce(robust)` standard errors for two of the three parameters are larger than the previously reported conventional standard errors. This is to be expected, even though (by construction) we have perfect model specification because this option trades off efficient estimation of the coefficient variance–covariance matrix for robustness against misspecification.

◁

Specifying the `vce(cluster` *clustvar*`)` option relaxes the usual assumption of independence between observations to the weaker assumption of independence just between clusters; that is, `hetprobit, vce(cluster` *clustvar*`)` is robust with respect to within-cluster correlation. This option is less efficient than the `xtgee` population-averaged models because `hetprobit` inefficiently sums within cluster for the standard-error calculation rather than attempting to exploit what might be assumed about the within-cluster correlation.

Stored results

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

Scalars

`e(N)`	number of observations
`e(N_f)`	number of zero outcomes
`e(N_s)`	number of nonzero outcomes
`e(k)`	number of parameters
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test
`e(k_dv)`	number of dependent variables
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(ll_0)`	log likelihood, constant-only model
`e(ll_c)`	log likelihood, comparison model
`e(N_clust)`	number of clusters
`e(chi2)`	χ^2
`e(chi2_c)`	χ^2 for heteroskedasticity test
`e(p_c)`	p-value for heteroskedasticity test
`e(df_m_c)`	degrees of freedom for heteroskedasticity test
`e(p)`	p-value for model test
`e(rank)`	rank of `e(V)`
`e(rank0)`	rank of `e(V)` for constant-only model
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	`1` if converged, `0` otherwise

Macros

`e(cmd)`	`hetprobit`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(wtype)`	weight type
`e(wexp)`	weight expression
`e(title)`	title in estimation output
`e(clustvar)`	name of cluster variable
`e(offset1)`	offset for probit equation
`e(offset2)`	offset for variance equation
`e(chi2type)`	`Wald` or `LR`; type of model χ^2 test
`e(chi2_ct)`	`Wald` or `LR`; type of model χ^2 test corresponding to `e(chi2_c)`
`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(method)`	`ml`
`e(ml_method)`	type of `ml` method
`e(user)`	name of likelihood-evaluator program
`e(technique)`	maximization technique
`e(properties)`	`b V`
`e(predict)`	program used to implement `predict`
`e(asbalanced)`	factor variables `fvset` as `asbalanced`
`e(asobserved)`	factor variables `fvset` as `asobserved`

Matrices

`e(b)`	coefficient vector
`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

Methods and formulas

The heteroskedastic probit model is a generalization of the probit model because it allows the scale of the inverse link function to vary from observation to observation as a function of the independent variables.

The log-likelihood function for the heteroskedastic probit model is

$$\ln L = \sum_{j \in S} w_j \ln\Phi\{\mathbf{x}_j\boldsymbol{\beta}/\exp(\mathbf{z}\boldsymbol{\gamma})\} + \sum_{j \notin S} w_j \ln\big[1 - \Phi\{\mathbf{x}_j\boldsymbol{\beta}/\exp(\mathbf{z}\boldsymbol{\gamma})\}\big]$$

where S is the set of all observations j such that $y_j \neq 0$ and w_j denotes the optional weights. $\ln L$ is maximized as described in [R] **maximize**.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster` *clustvar*`)`, respectively. See [P] **_robust**, particularly *Maximum likelihood estimators* and *Methods and formulas*.

`hetprobit` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] **variance estimation**.

References

Blevins, J. R., and S. Khan. 2013. Distribution-free estimation of heteroskedastic binary response models in Stata. *Stata Journal* 13: 588–602.

Greene, W. H. 2018. *Econometric Analysis*. 8th ed. New York: Pearson.

Harvey, A. C. 1976. Estimating regression models with multiplicative heteroscedasticity. *Econometrica* 44: 461–465.

Also see

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

[R] **logistic** — Logistic regression, reporting odds ratios

[R] **probit** — Probit regression

[BAYES] **bayes: hetprobit** — Bayesian heteroskedastic probit regression

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtprobit** — Random-effects and population-averaged probit models

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