nl postestimation — Postestimation tools for nl

Postestimation commands predict margins Remarks and examples Also see

Postestimation commands

The following postestimation commands are available after `nl`:

Command	Description
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `forecast`	dynamic forecasts and simulations
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
† `margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	fitted values, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`forecast`, `hausman`, and `lrtest` are not appropriate with `svy` estimation results.

†You must specify the `variables()` option with `nl`.

predict

Description for predict

`predict` creates a new variable containing predictions such as fitted values, residuals, probabilities, and expected values. It can also create multiple new variables containing predicted, named substitutable expressions.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for predicting without named substitutable expressions (parameters)

`predict` [*type*] *newvar* [*if*] [*in*] [, *statistic*]

`predict` [*type*] *stub*`*` [*if*] [*in*], `scores`

Syntax for predicting with named substitutable expressions (parameters)

Predict all parameters

`predict` [*type*] {*stub*`*`|*newvarlist*} [*if*] [*in*], `parameters`

Predict specific parameters

`predict` [*type*] (*newvar* = {*param*:}) [(*newvar* = {*param*:}) [...]] [*if*] [*in*]

`predict` [*type*] {*stub*`*`|*newvarlist*} [*if*] [*in*], `parameters(`*paramnames*`)`

statistic	Description
Main	
`yhat`	fitted values; the default
`residuals`	residuals
`pr(`*a*`,`*b*`)`	$\Pr(y_j \mid a < y_j < b)$
`e(`*a*`,`*b*`)`	$E(y_j \mid a < y_j < b)$
`ystar(`*a*`,`*b*`)`	$E(y_j^*)$, $y_j^* = \max\{a, \min(y_j, b)\}$

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

Options for predict

Main

`yhat`, the default, calculates the fitted values.

`residuals` calculates the residuals.

`pr(`*a*`,`*b*`)` calculates $\Pr(a < f(\mathbf{x}_j,\mathbf{b}) + u_j < b)$, the probability that $y_j|\mathbf{x}_j$ would be observed in the interval (a,b).

a and *b* may be specified as numbers or variable names; *lb* and *ub* are variable names;
`pr(20,30)` calculates $\Pr(20 < f(\mathbf{x}_j,\mathbf{b}) + u_j < 30)$;
`pr(`*lb*`,`*ub*`)` calculates $\Pr(lb < f(\mathbf{x}_j,\mathbf{b}) + u_j < ub)$; and
`pr(20,`*ub*`)` calculates $\Pr(20 < f(\mathbf{x}_j,\mathbf{b}) + u_j < ub)$.

a missing ($a \geq .$) means $-\infty$; `pr(.,30)` calculates $\Pr(-\infty < f(\mathbf{x}_j,\mathbf{b}) + u_j < 30)$;
`pr(`*lb*`,30)` calculates $\Pr(-\infty < f(\mathbf{x}_j,\mathbf{b}) + u_j < 30)$ in observations for which $lb \geq .$
and calculates $\Pr(lb < f(\mathbf{x}_j,\mathbf{b}) + u_j < 30)$ elsewhere.

b missing ($b \geq .$) means $+\infty$; `pr(20,.)` calculates $\Pr(+\infty > f(\mathbf{x}_j,\mathbf{b}) + u_j > 20)$;
`pr(20,`*ub*`)` calculates $\Pr(+\infty > f(\mathbf{x}_j,\mathbf{b}) + u_j > 20)$ in observations for which $ub \geq .$
and calculates $\Pr(20 < f(\mathbf{x}_j,\mathbf{b}) + u_j < ub)$ elsewhere.

`e(`*a*`,`*b*`)` calculates $E(f(\mathbf{x}_j,\mathbf{b}) + u_j \mid a < f(\mathbf{x}_j,\mathbf{b}) + u_j < b)$, the expected value of $y_j|\mathbf{x}_j$ conditional on $y_j|\mathbf{x}_j$ being in the interval (a,b), meaning that $y_j|\mathbf{x}_j$ is truncated. *a* and *b* are specified as they are for `pr()`.

`ystar(`*a*`,`*b*`)` calculates $E(y_j^*)$, where $y_j^* = a$ if $f(\mathbf{x}_j,\mathbf{b}) + u_j \leq a$, $y_j^* = b$ if $f(\mathbf{x}_j,\mathbf{b}) + u_j \geq b$, and $y_j^* = f(\mathbf{x}_j,\mathbf{b}) + u_j$ otherwise, meaning that y_j^* is censored. *a* and *b* are specified as they are for `pr()`.

`scores` calculates the scores. The *j*th new variable created will contain the score for the *j*th parameter in `e(b)`.

`parameters` and `parameters(`*paramnames*`)` calculate predictions for all or a subset of the named substitutable expressions in the model. `parameters()` does not appear in the dialog box.

paramnames is *param* [*param* [. . .]], and *param* is a name of a substitutable expression as specified in one of `nl`'s `define()` options.

margins

Description for margins

`margins` estimates margins of response for fitted values.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [, *options*]

`margins` [*marginlist*] , `predict(`*statistic* ...`)` [*options*]

statistic	Description
`yhat`	fitted values; the default
`pr(`*a*`,`*b*`)`	not allowed with `margins`
`e(`*a*`,`*b*`)`	not allowed with `margins`
`ystar(`*a*`,`*b*`)`	not allowed with `margins`
`residuals`	not allowed with `margins`
`parameters`	predicted parameters
`parameters(`*param*`)`	predicted, named substitutable expression *param*

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Example 1: Basic usage of `predict` and `margins`

Obtaining predictions after fitting a nonlinear regression model with `nl` is no more difficult than obtaining predictions after fitting a linear regression model with `regress`. Here we fit a model of `mpg` on `weight`, allowing for a nonlinear relationship:

```
. use https://www.stata-press.com/data/r19/auto
(1978 automobile data)
. nl (mpg = {b0} + {b1}*weight^{gamma=-.5}), variables(weight) nolog
```

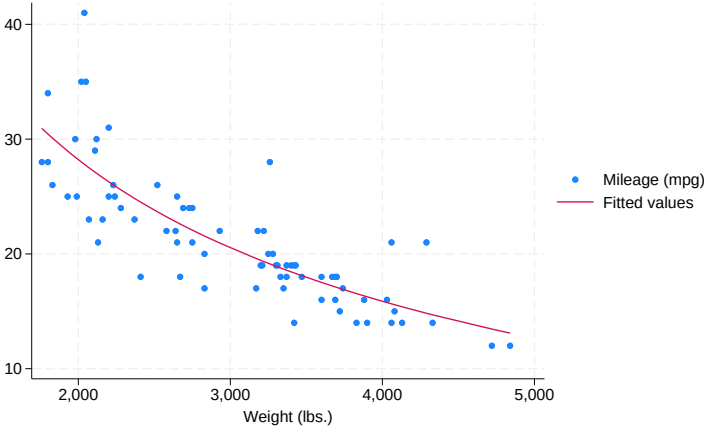
Source	SS	df	MS		
				Number of obs =	74
Model	1646.4376	2	823.218805	R-squared =	0.6738
Residual	797.02185	71	11.2256599	Adj R-squared =	0.6646
				Root MSE =	3.350472
Total	2443.4595	73	33.4720474	Res. dev. =	385.8874

mpg	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
/b0	-18.17687	60.62086	-0.30	0.765	-139.0514	102.6977
/b1	1377.163	5291.956	0.26	0.795	-9174.697	11929.02
/gamma	-.4460793	.6763682	-0.66	0.512	-1.794719	.9025606

Note: Parameter **b0** is used as a constant term during estimation.

Now, we obtain the predicted values of `mpg` and plot them in a graph along with the observed values:

```
. predict mpghat
(option yhat assumed; fitted values)
. scatter mpg weight || line mpghat weight, sort
```



Suppose we wanted to know how sensitive mpg is to changes in weight for cars that weigh 3,000 pounds. We can use margins to find out:

```
. margins, eyex(weight) at(weight = 3000)
Conditional marginal effects                                    Number of obs = 74
Model VCE: GNR

Expression: Fitted values, predict()
ey/ex wrt:  weight
At: weight = 3000
```

	ey/ex	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
weight	-.8408143	.0804327	-10.45	0.000	-.9984596	-.683169

With the eyex() option, margins reports elasticities. These results show that if we increase weight by 1%, then mpg decreases by about 0.84%.

◁

❑ Technical note

Observant readers will notice that margins issued a warning message stating that it could not perform its usual check for estimable functions. In the case of nl, as long as you do not specify the predict() option of margins or specify the default predict(yhat), you can safely ignore that message. The predicted values that nl produces are suitable for use with margins. However, if you specify any predict() options other than yhat, then the output from margins after using nl will not be correct.

❑

▷ Example 2: Predictions from named expressions

Continuing with example 1, consider now a logistic regression that explains the origin of the cars as a function of their price, their fuel efficiency, and their length.

```
. logit foreign price mpg length
Iteration 0:  Log likelihood =  -45.03321
Iteration 1:  Log likelihood = -27.130212
Iteration 2:  Log likelihood = -23.885718
Iteration 3:  Log likelihood = -23.468167
Iteration 4:  Log likelihood = -23.460272
Iteration 5:  Log likelihood = -23.460253
Iteration 6:  Log likelihood = -23.460253
Logistic regression                                     Number of obs =     74
                                                        LR chi2(3)    =  43.15
                                                        Prob > chi2   = 0.0000
Log likelihood = -23.460253                             Pseudo R2     = 0.4790
```

foreign	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
price	.0005602	.0001889	2.97	0.003	.0001899	.0009306
mpg	-.0606663	.0854276	-0.71	0.478	-.2281012	.1067687
length	-.1718435	.0538215	-3.19	0.001	-.2773316	-.0663553
_cons	27.7666	9.860407	2.82	0.005	8.440557	47.09264

After fitting this model, the `predict` command computes predicted probabilities by default or the linear prediction if we use the `xb` option:

```
. predict p_logit
(option pr assumed; Pr(foreign))
. predict xb_logit, xb
```

Next recall that logistic regression fits the logistic cumulative density function to our data via maximum likelihood. We could instead fit this (nonlinear) function via nonlinear least squares using the `nl` command:

```
. nl (foreign = 1/(1 + exp(-{lp: price mpg length})))
Iteration 0:  Residual SS =       18.5
Iteration 1:  Residual SS =  7.7385558
Iteration 2:  Residual SS =  7.6199595
Iteration 3:  Residual SS =  7.6149448
Iteration 4:  Residual SS =  7.6143961
Iteration 5:  Residual SS =  7.6143036
Iteration 6:  Residual SS =  7.6142876
Iteration 7:  Residual SS =  7.6142848
Iteration 8:  Residual SS =  7.6142843
```

Source	SS	df	MS		
				Number of obs =	74
Model	7.8451752	3	2.61505839	R-squared =	0.5075
Residual	7.6142843	70	.10877549	Adj R-squared =	0.4864
				Root MSE =	.3298113
Total	15.459459	73	.211773417	Res. dev. =	41.72401

foreign	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
price	.0006768	.0002826	2.39	0.019	.0001131	.0012405
mpg	-.0703169	.0704894	-1.00	0.322	-.2109036	.0702698
length	-.2241772	.0912304	-2.46	0.016	-.4061305	-.042224
_cons	36.6067	15.28359	2.40	0.019	6.124531	67.08886

```
Note: Parameter _cons is used as a constant term during estimation.
```

Computing predicted probabilities from this model is still the default for `predict` because these are simply the values our model predicts for the dependent variable:

```
. predict p_nl
(option yhat assumed; fitted values)
```

To obtain the linear prediction, we need the predicted values of the named linear combination `{lp:}` we created. We can compute those using the `parameters()` option:

```
. predict xb_nl, parameters(lp)
```

◁

Also see

[R] **nl** — Nonlinear least-squares estimation

[U] 20 Estimation and postestimation commands