Title

stata.com

predictnl — Obtain nonlinear predictions, standard errors, etc., after estimation

Description

`predictnl` calculates (possibly) nonlinear predictions after any Stata estimation command and optionally calculates the variances, standard errors, Wald test statistics, p-values, and confidence limits for these predictions. Unlike its companion nonlinear postestimation commands `testnl` and `nlcom`, `predictnl` generates functions of the data (that is, predictions), not scalars. The quantities generated by `predictnl` are thus vectorized over the observations in the data.

Consider some general prediction, $g(\boldsymbol{\theta}, \mathbf{x}_i)$, for $i = 1, \ldots, n$, where $\boldsymbol{\theta}$ are the model parameters and $\mathbf{x}_i$ are some data for the ith observation; $\mathbf{x}_i$ is assumed fixed. Typically, $g(\boldsymbol{\theta}, \mathbf{x}_i)$ is estimated by $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$, where $\widehat{\boldsymbol{\theta}}$ are the estimated model parameters, which are stored in `e(b)` following any Stata estimation command.

In its most common use, `predictnl` generates two variables: one containing the estimated prediction, $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$, the other containing the estimated standard error of $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$. The calculation of standard errors (and other obtainable quantities that are based on the standard errors, such as test statistics) is based on the delta method, an approximation appropriate in large samples; see *Methods and formulas*.

`predictnl` can be used with `svy` estimation results (assuming that `predict` is also allowed), see [SVY] **svy postestimation**.

The specification of $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$ is handled by specifying *pnl_exp*, and the values of $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$ are stored in the new variable *newvar* of storage type *type*. *pnl_exp* is any valid Stata expression and may also contain calls to two special functions unique to `predictnl`:

1. `predict(`[*predict_options*]`)`: When you are evaluating *pnl_exp*, `predict()` is a convenience function that replicates the calculation performed by the command

   ```
   predict ..., predict_options
   ```

   As such, the `predict()` function may be used either as a shorthand for the formula used to make this prediction or when the formula is not readily available. When used without arguments, `predict()` replicates the default prediction for that particular estimation command.

2. `xb(`[*eqno*]`)`: The `xb()` function replicates the calculation of the linear predictor $\mathbf{x}_i\mathbf{b}$ for equation *eqno*. If `xb()` is specified without *eqno*, the linear predictor for the first equation (or the only equation in single-equation estimation) is obtained.

   For example, `xb(#1)` (or equivalently, `xb()` with no arguments) translates to the linear predictor for the first equation, `xb(#2)` for the second, and so on. You could also refer to the equations by their names, such as `xb(income)`.

   When specifying *pnl_exp*, both of these functions may be used repeatedly, in combination, and in combination with other Stata functions and expressions. See *Remarks and examples* for examples that use both of these functions.

Quick start

After `regress`, create `yhat` containing the default linear prediction

```
predictnl yhat = predict()
```

After `probit y x1 x2`, create `pr1` containing predicted probability that `y` = 1

```
predictnl pr1 = predict(pr)
```

Same as above, and create `lb1` and `ub1` containing the upper and lower bounds of the 95% confidence interval for the prediction

```
predictnl pr1 = predict(pr), ci(lb1 ub1)
```

Same as above, specified as a function of the linear prediction

```
predictnl pr1 = normal(xb()), ci(lb1 ub1)
```

Same as above, specified as a function of the coefficients

```
predictnl pr1 = normal(_b[_cons]+_b[x1]*x1+_b[x2]*x2), ci(lb1 ub1)
```

Same as above, but create the prediction using `x1` equal to 0

```
predictnl pr1 = normal(_b[_cons]+_b[x1]*0+_b[x2]*x2), ci(lb1 ub1)
```

Same as above, but create variable `se1` containing the standard error of the prediction

```
predictnl pr1 = normal(_b[_cons]+_b[x1]*0+_b[x2]*x2), se(se1)
```

After a multiple-equation model, create `ratio` as the ratio of the linear predictions from the second and third equations

```
predictnl ratio = xb(#2)/xb(#3)
```

Menu

Statistics > Postestimation

Syntax

```
predictnl [type] newvar = pnl_exp [if] [in] [, options]
```

options	Description
Main	
`se(`*newvar*`)`	create *newvar* containing standard errors
`variance(`*newvar*`)`	create *newvar* containing variances
`wald(`*newvar*`)`	create *newvar* containing the Wald test statistic
`p(`*newvar*`)`	create *newvar* containing the p-value for the Wald test
`ci(`*newvars*`)`	create *newvars* containing lower and upper confidence intervals
`level(#)`	set confidence level; default is `level(95)`
`g(`*stub*`)`	create *stub*`1`, *stub*`2`, ..., *stub*`k` variables containing observation-specific derivatives
Advanced	
`iterate(#)`	maximum iterations for finding optimal step size; default is 100
`force`	calculate standard errors, etc., even when possibly inappropriate
`df(#)`	use F distribution with # denominator degrees of freedom for the reference distribution of the test statistic

`df(#)` does not appear in the dialog box.

Options

Main

`se(`*newvar*`)` adds *newvar* of storage type *type*, where for each `i` in the prediction sample, *newvar*`[i]` contains the estimated standard error of $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$.

`variance(`*newvar*`)` adds *newvar* of storage type *type*, where for each `i` in the prediction sample, *newvar*`[i]` contains the estimated variance of $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$.

`wald(`*newvar*`)` adds *newvar* of storage type *type*, where for each `i` in the prediction sample, *newvar*`[i]` contains the Wald test statistic for the test of the hypothesis $H_0\colon g(\boldsymbol{\theta}, \mathbf{x}_i) = 0$.

`p(`*newvar*`)` adds *newvar* of storage type *type*, where *newvar*`[i]` contains the p-value for the Wald test of $H_0\colon g(\boldsymbol{\theta}, \mathbf{x}_i) = 0$ versus the two-sided alternative.

`ci(`*newvars*`)` requires the specification of two *newvars*, such that the ith observation of each will contain the left and right endpoints (respectively) of a confidence interval for $g(\boldsymbol{\theta}, \mathbf{x}_i)$. The level of the confidence intervals is determined by `level(#)`.

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see **[U] 20.8 Specifying the width of confidence intervals**.

`g(`*stub*`)` specifies that new variables, *stub*`1`, *stub*`2`, ..., *stub*`k` be created, where `k` is the dimension of $\boldsymbol{\theta}$. *stub*`1` will contain the observation-specific derivatives of $g(\boldsymbol{\theta}, \mathbf{x}_i)$ with respect to the first element, θ_1, of $\boldsymbol{\theta}$; *stub*`2` will contain the derivatives of $g(\boldsymbol{\theta}, \mathbf{x}_i)$ with respect to θ_2, etc.; If the derivative of $g(\boldsymbol{\theta}, \mathbf{x}_i)$ with respect to a particular coefficient in $\boldsymbol{\theta}$ equals zero for all observations in the prediction sample, the *stub* variable for that coefficient is not created. The ordering of the parameters in $\boldsymbol{\theta}$ is precisely that of the stored vector of parameter estimates `e(b)`.

Advanced

`iterate(#)` specifies the maximum number of iterations used to find the optimal step size in the calculation of numerical derivatives of $g(\boldsymbol{\theta}, \mathbf{x}_i)$ with respect to $\boldsymbol{\theta}$. By default, the maximum number of iterations is 100, but convergence is usually achieved after only a few iterations. You should rarely have to use this option.

`force` forces the calculation of standard errors and other inference-related quantities in situations where `predictnl` would otherwise refuse to do so. The calculation of standard errors takes place by evaluating (at $\widehat{\boldsymbol{\theta}}$) the numerical derivative of $g(\boldsymbol{\theta}, \mathbf{x}_i)$ with respect to $\boldsymbol{\theta}$. If `predictnl` detects that $g(\cdot)$ is possibly a function of random quantities other than $\widehat{\boldsymbol{\theta}}$, it will refuse to calculate standard errors or any other quantity derived from them. The `force` option forces the calculation to take place anyway. If you use the `force` option, there is no guarantee that any inference quantities (for example, standard errors) will be correct or that the values obtained can be interpreted.

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

`df(#)` specifies that the F distribution with # denominator degrees of freedom be used for the reference distribution of the test statistic.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction

`predictnl` and `nlcom` both use the delta method. They take a nonlinear transformation of the estimated parameter vector from some fitted model and apply the delta method to calculate the variance, standard error, Wald test statistic, etc., of this transformation. `nlcom` is designed for scalar functions of the parameters, and `predictnl` is designed for functions of the parameters and of the data, that is, for predictions.

Nonlinear transformations and standard errors

We begin by fitting a probit model to the low-birthweight data of Hosmer, Lemeshow, and Sturdivant (2013, 24). The data are described in detail in example 1 of [R] **logistic**.

```
. use https://www.stata-press.com/data/r18/lbw
(Hosmer & Lemeshow data)
. probit low lwt smoke ptl ht
Iteration 0:  Log likelihood =   -117.336
Iteration 1:  Log likelihood = -106.75886
Iteration 2:  Log likelihood = -106.67852
Iteration 3:  Log likelihood = -106.67851
Probit regression                                 Number of obs =    189
                                                  LR chi2(4)    =  21.31
                                                  Prob > chi2   = 0.0003
Log likelihood = -106.67851                       Pseudo R2     = 0.0908
```

low	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
lwt	-.0095164	.0036875	-2.58	0.010	-.0167438	-.0022891
smoke	.3487004	.2041772	1.71	0.088	-.0514794	.7488803
ptl	.365667	.1921201	1.90	0.057	-.0108815	.7422154
ht	1.082355	.410673	2.64	0.008	.2774503	1.887259
_cons	.4238985	.4823224	0.88	0.379	-.5214361	1.369233

After we fit such a model, we first would want to generate the predicted probabilities of a low birthweight, given the covariate values in the estimation sample. This is easily done using `predict` after `probit`, but it doesn't answer the question, "What are the standard errors of those predictions?"

For the time being, we will consider ourselves ignorant of any automated way to obtain the predicted probabilities after `probit`. The formula for the prediction is

$$\Pr(y \neq 0|\mathbf{x}_i) = \Phi(\mathbf{x}_i\boldsymbol{\beta})$$

where Φ is the standard cumulative normal. Thus for this example, $g(\boldsymbol{\theta}, \mathbf{x}_i) = \Phi(\mathbf{x}_i\boldsymbol{\beta})$. Armed with the formula, we can use `predictnl` to generate the predictions and their standard errors:

```
. predictnl phat = normal(_b[_cons] + _b[ht]*ht + _b[ptl]*ptl +
> _b[smoke]*smoke + _b[lwt]*lwt), se(phat_se)
. list phat phat_se lwt smoke ptl ht in -10/l
```

	phat	phat_se	lwt	smoke	ptl	ht
180.	.2363556	.042707	120	Nonsmoker	0	0
181.	.6577712	.1580714	154	Nonsmoker	1	1
182.	.2793261	.0519958	106	Nonsmoker	0	0
183.	.1502118	.0676339	190	Smoker	0	0
184.	.5702871	.0819911	101	Smoker	1	0
185.	.4477045	.079889	95	Smoker	0	0
186.	.2988379	.0576306	100	Nonsmoker	0	0
187.	.4514706	.080815	94	Smoker	0	0
188.	.5615571	.1551051	142	Nonsmoker	0	1
189.	.7316517	.1361469	130	Smoker	0	1

Thus, subject 180 in our data has an estimated probability of low birthweight of 23.6% with standard error 4.3%.

Used without options, `predictnl` is not much different from `generate`. By specifying the `se(phat_se)` option, we were able to obtain a variable containing the standard errors of the predictions; therein lies the utility of `predictnl`.

Using xb() and predict()

As was the case above, a prediction is often not a function of a few isolated parameters and their corresponding variables but instead is some (possibly elaborate) function of the entire linear predictor. For models with many predictors, the brute-force expression for the linear predictor can be cumbersome to type. An alternative is to use the inline function `xb()`. `xb()` is a shortcut for having to type `_b[_cons] + _b[ht]*ht + _b[ptl]*ptl + ...`,

```
. drop phat phat_se
. predictnl phat = norm(xb()), se(phat_se)
. list phat phat_se lwt smoke ptl ht in -10/l
```

	phat	phat_se	lwt	smoke	ptl	ht
180.	.2363556	.042707	120	Nonsmoker	0	0
181.	.6577712	.1580714	154	Nonsmoker	1	1
182.	.2793261	.0519958	106	Nonsmoker	0	0
183.	.1502118	.0676339	190	Smoker	0	0
184.	.5702871	.0819911	101	Smoker	1	0
185.	.4477045	.079889	95	Smoker	0	0
186.	.2988379	.0576306	100	Nonsmoker	0	0
187.	.4514706	.080815	94	Smoker	0	0
188.	.5615571	.1551051	142	Nonsmoker	0	1
189.	.7316517	.1361469	130	Smoker	0	1

which yields the same results. This approach is easier, produces more readable code, and is less prone to error, such as forgetting to include a term in the sum.

Here we used `xb()` without arguments because we have only one equation in our model. In multiple-equation (ME) settings, `xb()` (or equivalently `xb(#1)`) yields the linear predictor from the first equation, `xb(#2)` from the second, etc. You can also refer to equations by their names, for example, `xb(income)`.

❑ Technical note

Most estimation commands in Stata allow the postestimation calculation of linear predictors and their standard errors via `predict`. For example, to obtain these for the first (or only) equation in the model, you could type

```
predict xbvar, xb
predict stdpvar, stdp
```

Equivalently, you could type

```
predictnl xbvar = xb(), se(stdpvar)
```

but we recommend the first method, because it is faster. As we demonstrated above, however, `predictnl` is more general.

❑

Returning to our `probit` example, we can further simplify the calculation by using the inline function `predict()`. `predict(`*pred_options*`)` works by substituting, within our `predictnl` expression, the calculation performed by

```
predict ..., pred_options
```

In our example, we are interested in the predicted probabilities after a probit regression, normally obtained via

```
predict ..., p
```

We can obtain these predictions (and standard errors) by using

```
. drop phat phat_se
. predictnl phat = predict(p), se(phat_se)
. list phat phat_se lwt smoke ptl ht in -10/l
```

	phat	phat_se	lwt	smoke	ptl	ht
180.	.2363556	.042707	120	Nonsmoker	0	0
181.	.6577712	.1580714	154	Nonsmoker	1	1
182.	.2793261	.0519958	106	Nonsmoker	0	0
183.	.1502118	.0676339	190	Smoker	0	0
184.	.5702871	.0819911	101	Smoker	1	0
185.	.4477045	.079889	95	Smoker	0	0
186.	.2988379	.0576306	100	Nonsmoker	0	0
187.	.4514706	.080815	94	Smoker	0	0
188.	.5615571	.1551051	142	Nonsmoker	0	1
189.	.7316517	.1361469	130	Smoker	0	1

which again replicates what we have already done by other means. However, this version did not require knowledge of the formula for the predicted probabilities after a probit regression—`predict(p)` took care of that for us.

Because the predicted probability is the default prediction after `probit`, we could have just used `predict()` without arguments, namely,

```
. predictnl phat = predict(), se(phat_se)
```

Also, the expression *pnl_exp* can be inordinately complicated, with multiple calls to `predict()` and `xb()`. For example,

```
. predictnl phat = normal(invnormal(predict()) + predict(xb)/xb() - 1),
> se(phat_se)
```

is perfectly valid and will give the same result as before, albeit a bit inefficiently.

❑ Technical note

When using `predict()` and `xb()`, the *formula* for the calculation is substituted within *pnl_exp*, not the values that result from the application of that formula. To see this, note the subtle difference between

```
. predict xbeta, xb
. predictnl phat = normal(xbeta), se(phat_se)
```

and

```
. predictnl phat = normal(xb()), se(phat_se)
```

Both sequences will yield the same `phat`, yet for the first sequence, `phat_se` will equal zero for all observations. The reason is that, once evaluated, `xbeta` will contain the values of the linear predictor, yet these values are treated as fixed and nonstochastic as far as `predictnl` is concerned. By contrast, because `xb()` is shorthand for the formula used to calculate the linear predictor, it contains not values, but references to the estimated regression coefficients and corresponding variables. Thus, the second method produces the desired result. ❑

Multiple-equation (ME) estimators

In [R] **mlogit**, data on insurance choice (Tarlov et al. 1989; Wells et al. 1989) were examined, and a multinomial logit was used to assess the effects of age, gender, race, and site of study (one of three sites) on the type of insurance:

```
. use https://www.stata-press.com/data/r18/sysdsn1, clear
(Health insurance data)
. mlogit insure age male nonwhite i.site, nolog
Multinomial logistic regression                 Number of obs =    615
                                                LR chi2(10)   =  42.99
                                                Prob > chi2   = 0.0000
Log likelihood = -534.36165                     Pseudo R2     = 0.0387
```

insure	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
Indemnity	(base outcome)					
Prepaid						
age	-.011745	.0061946	-1.90	0.058	-.0238862	.0003962
male	.5616934	.2027465	2.77	0.006	.1643175	.9590693
nonwhite	.9747768	.2363213	4.12	0.000	.5115955	1.437958
site						
2	.1130359	.2101903	0.54	0.591	-.2989296	.5250013
3	-.5879879	.2279351	-2.58	0.010	-1.034733	-.1412433
_cons	.2697127	.3284422	0.82	0.412	-.3740222	.9134476
Uninsure						
age	-.0077961	.0114418	-0.68	0.496	-.0302217	.0146294
male	.4518496	.3674867	1.23	0.219	-.268411	1.17211
nonwhite	.2170589	.4256361	0.51	0.610	-.6171725	1.05129
site						
2	-1.211563	.4705127	-2.57	0.010	-2.133751	-.2893747
3	-.2078123	.3662926	-0.57	0.570	-.9257327	.510108
_cons	-1.286943	.5923219	-2.17	0.030	-2.447872	-.1260134

Of particular interest is the estimation of the relative risk, which, for a given selection, is the ratio of the probability of making that selection to the probability of selecting the base category (`Indemnity` here), given a set of covariate values. In a multinomial logit model, the relative risk (when comparing to the base category) simplifies to the exponentiated linear predictor for that selection.

Using this example, we can estimate the observation-specific relative risks of selecting a prepaid plan over the base category (with standard errors) by either referring to the `Prepaid` equation by name or number,

```
. predictnl RRppaid = exp(xb(Prepaid)), se(SERRppaid)
```

or

```
. predictnl RRppaid = exp(xb(#1)), se(SERRppaid)
```

because `Prepaid` is the first equation in the model.

Those of us for whom the simplified formula for the relative risk does not immediately come to mind may prefer to calculate the relative risk directly from its definition, that is, as a ratio of two predicted probabilities. After `mlogit`, the predicted probability for a category may be obtained using `predict`, but we must specify the category as the outcome:

```
. predictnl RRppaid = predict(outcome(Prepaid))/predict(outcome(Indemnity)),
> se(SERRppaid)
(1 missing value generated)
. list RRppaid SERRppaid age male nonwhite site in 1/10
```

	RRppaid	SERRpp~d	age	male	nonwhite	site
1.	.6168578	.1503759	73.722107	0	0	2
2.	1.056658	.1790703	27.89595	0	0	2
3.	.8426442	.1511281	37.541397	0	0	1
4.	1.460581	.3671465	23.641327	0	1	3
5.	.9115747	.1324168	40.470901	0	0	2
6.	1.034701	.1696923	29.683777	0	0	2
7.	.9223664	.1344981	39.468857	0	0	2
8.	1.678312	.4216626	26.702255	1	0	1
9.	.9188519	.2256017	63.101974	0	1	3
10.	.5766296	.1334877	69.839828	0	0	1

The "(1 missing value generated)" message is not an error; further examination of the data would reveal that `age` is missing in one observation and that the offending observation (among others) is not in the estimation sample. Just as with `predict`, `predictnl` can generate predictions in or out of the estimation sample.

Thus, we estimate (among other things) that a white, female, 73-year-old from site 2 is less likely to choose a prepaid plan over an indemnity plan—her relative risk is about 62% with standard error 15%.

Test statistics and p-values

Often, a standard error calculation is just a means to an end, and what is really desired is a test of the hypothesis,

$$H_0 : g(\boldsymbol{\theta}, \mathbf{x}_i) = 0$$

versus the two-sided alternative.

We can use `predictnl` to obtain the Wald test statistics or p-values (or both) for the above tests, whether or not we want standard errors. To obtain the Wald test statistics, we use the `wald()` option; for p-values, we use `p()`.

Returning to our `mlogit` example, suppose that we wanted for each observation a test of whether the relative risk of choosing a prepaid plan over an indemnity plan is different from one. One way to do this would be to define $g(\cdot)$ to be the relative risk minus one and then test whether $g(\cdot)$ is different from zero.

```
. predictnl RRm1 = exp(xb(Prepaid)) - 1, wald(W_RRm1) p(sig_RRm1)
(1 missing value generated)
note: p-values are with respect to the chi-squared(1) distribution.
. list RRm1 W_RRm1 sig_RRm1 age male nonwhite in 1/10
```

	RRm1	W_RRm1	sig_RRm1	age	male	nonwhite
1.	-.3831422	6.491778	.0108375	73.722107	0	0
2.	.0566578	.100109	.7516989	27.89595	0	0
3.	-.1573559	1.084116	.2977787	37.541397	0	0
4.	.4605812	1.573743	.2096643	23.641327	0	1
5.	-.0884253	.4459299	.5042742	40.470901	0	0
6.	.0347015	.0418188	.8379655	29.683777	0	0
7.	-.0776336	.3331707	.563798	39.468857	0	0
8.	.6783119	2.587788	.1076906	26.702255	1	0
9.	-.0811482	.1293816	.719074	63.101974	0	1
10.	-.4233705	10.05909	.001516	69.839828	0	0

The newly created variable `W_RRm1` contains the Wald test statistic for each observation, and `sig_RRm1` contains the p-value. Thus, our 73-year-old white female represented by the first observation would have a relative risk of choosing prepaid over indemnity that is significantly different from 1, at least at the 5% level. For this test, it was not necessary to generate a variable containing the standard error of the relative risk minus 1, but we could have done so had we wanted. We could have also omitted specifying `wald(W_RRm1)` if all we cared about were, say, the p-values for the tests.

In this regard, `predictnl` acts as an observation-specific version of `testnl`, with the test results vectorized over the observations in the data. The p-values are pointwise—they are not adjusted to reflect any simultaneous testing over the observations in the data.

Manipulability

There are many ways to specify $g(\boldsymbol{\theta}, \mathbf{x}_i)$ to yield tests such that, for multiple specifications of $g(\cdot)$, the theoretical conditions for which

$$H_0 : g(\boldsymbol{\theta}, \mathbf{x}_i) = 0$$

is true will be equivalent. However, this does not mean that the tests themselves will be equivalent. This is known as the manipulability of the Wald test for nonlinear hypotheses; also see [R] **boxcox**.

As an example, consider the previous section where we defined $g(\cdot)$ to be the relative risk between choosing a prepaid plan over an indemnity plan, minus 1. We could also have defined $g(\cdot)$ to be the risk difference—the probability of choosing a prepaid plan minus the probability of choosing an indemnity plan. Either specification of $g(\cdot)$ yields a mathematically equivalent specification of $H_0 : g(\cdot) = 0$; that is, the risk difference will equal zero when the relative risk equals one. However, the tests themselves do not give the same results:

```
. predictnl RD = predict(outcome(Prepaid)) - predict(outcome(Indemnity)),
> wald(W_RD) p(sig_RD)
(1 missing value generated)
note: p-values are with respect to the chi-squared(1) distribution.
. list RD W_RD sig_RD RRm1 W_RRm1 sig_RRm1 in 1/10
```

	RD	W_RD	sig_RD	RRm1	W_RRm1	sig_RRm1
1.	-.2303744	4.230243	.0397097	-.3831422	6.491778	.0108375
2.	.0266902	.1058542	.7449144	.0566578	.100109	.7516989
3.	-.0768078	.9187646	.3377995	-.1573559	1.084116	.2977787
4.	.1710702	2.366535	.1239619	.4605812	1.573743	.2096643
5.	-.0448509	.4072922	.5233471	-.0884253	.4459299	.5042742
6.	.0165251	.0432816	.835196	.0347015	.0418188	.8379655
7.	-.0391535	.3077611	.5790573	-.0776336	.3331707	.563798
8.	.22382	4.539085	.0331293	.6783119	2.587788	.1076906
9.	-.0388409	.1190183	.7301016	-.0811482	.1293816	.719074
10.	-.2437626	6.151558	.0131296	-.4233705	10.05909	.001516

In certain cases (such as subject 8), the difference can be severe enough to potentially change the conclusion. The reason for this inconsistency is that the nonlinear Wald test is actually a standard Wald test of a first-order Taylor approximation of $g(\cdot)$, and this approximation can differ according to how $g(\cdot)$ is specified.

As such, keep in mind the manipulability of nonlinear Wald tests when drawing scientific conclusions.

Confidence intervals

We can also use `predictnl` to obtain confidence intervals for the observation-specific $g(\boldsymbol{\theta}, \mathbf{x}_i)$ by using the `ci()` option to specify two new variables to contain the left and right endpoints of the confidence interval, respectively. For example, we could generate confidence intervals for the risk differences calculated previously:

```
. drop RD
. predictnl RD = predict(outcome(Prepaid)) - predict(outcome(Indemnity)),
> ci(RD_lcl RD_rcl)
(1 missing value generated)
note: confidence intervals calculated using Z critical values.
. list RD RD_lcl RD_rcl age male nonwhite in 1/10
```

	RD	RD_lcl	RD_rcl	age	male	nonwhite
1.	-.2303744	-.4499073	-.0108415	73.722107	0	0
2.	.0266902	-.1340948	.1874752	27.89595	0	0
3.	-.0768078	-.2338625	.080247	37.541397	0	0
4.	.1710702	-.0468844	.3890248	23.641327	0	1
5.	-.0448509	-.1825929	.092891	40.470901	0	0
6.	.0165251	-.1391577	.1722078	29.683777	0	0
7.	-.0391535	-.177482	.099175	39.468857	0	0
8.	.22382	.0179169	.4297231	26.702255	1	0
9.	-.0388409	-.2595044	.1818226	63.101974	0	1
10.	-.2437626	-.4363919	-.0511332	69.839828	0	0

The confidence level, here, 95%, is either set using the `level()` option or obtained from the current default level, `c(level)`; see [U] **20.8 Specifying the width of confidence intervals**.

From the above output, we can see that, for subjects 1, 8, and 10, a 95% confidence interval for the risk difference does not contain zero, meaning that, for these subjects, there is some evidence of a significant difference in risks.

The confidence intervals calculated by `predictnl` are pointwise; there is no adjustment (such as a Bonferroni correction) made so that these confidence intervals may be considered jointly at the specified level.

Methods and formulas

For the ith observation, consider the transformation $g(\boldsymbol{\theta}, \mathbf{x}_i)$, estimated by $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$, for the $1 \times k$ parameter vector $\boldsymbol{\theta}$ and data $\mathbf{x}_i$ ($\mathbf{x}_i$ is assumed fixed). The variance of $g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)$ is estimated by

$$\widehat{\text{Var}}\left\{g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)\right\} = \mathbf{GVG}'$$

where $\mathbf{G}$ is the vector of derivatives

$$\mathbf{G} = \left\{ \left. \frac{\partial g(\boldsymbol{\theta}, \mathbf{x}_i)}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}} \right\}_{(1\times k)}$$

and $\mathbf{V}$ is the estimated variance–covariance matrix of $\widehat{\boldsymbol{\theta}}$. Standard errors, $\widehat{\text{se}}\{g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)\}$, are obtained as the square roots of the variances.

The Wald test statistic for testing

$$H_0 \colon g(\boldsymbol{\theta}, \mathbf{x}_i) = 0$$

versus the two-sided alternative is given by

$$W_i = \frac{\left\{g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)\right\}^2}{\widehat{\text{Var}}\left\{g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)\right\}}$$

When the variance–covariance matrix of $\widehat{\boldsymbol{\theta}}$ is an asymptotic covariance matrix, W_i is approximately distributed as χ^2 with 1 degree of freedom. For linear regression, W_i is taken to be approximately distributed as $F_{1,r}$, where r is the residual degrees of freedom from the original model fit. The p-values for the observation-by-observation tests of H_0 versus the two-sided alternative are given by

$$p_i = \Pr(T > W_i)$$

where T is either a χ^2- or F-distributed random variable, as described above.

A $(1 - \alpha) \times 100\%$ confidence interval for $g(\boldsymbol{\theta}, \mathbf{x}_i)$ is given by

$$g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i) \pm z_{\alpha/2} \left[\widehat{\text{se}}\left\{g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)\right\}\right]$$

when W_i is χ^2-distributed, and

$$g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i) \pm t_{\alpha/2,r} \left[\widehat{\text{se}}\left\{g(\widehat{\boldsymbol{\theta}}, \mathbf{x}_i)\right\}\right]$$

when W_i is F-distributed. z_p is the $1-p$ quantile of the standard normal distribution, and $t_{p,r}$ is the $1-p$ quantile of the t distribution with r degrees of freedom.

References

Drukker, D. M. 2016a. Probability differences and odds ratios measure conditional-on-covariate effects and population-parameter effects. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/07/26/probability-differences-and-odds-ratios-measure-conditional-on-covariate-effects-and-population-parameter-effects/.

——. 2016b. Quantile regression allows covariate effects to differ by quantile. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/09/27/quantile-regression-allows-covariate-effects-to-differ-by-quantile/.

Hosmer, D. W., Jr., S. A. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.

Phillips, P. C. B., and J. Y. Park. 1988. On the formulation of Wald tests of nonlinear restrictions. *Econometrica* 56: 1065–1083. https://doi.org/10.2307/1911359.

Tarlov, A. R., J. E. Ware, Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930. https://doi.org/10.1001/jama.1989.03430070073033.

Wells, K. B., R. D. Hays, M. A. Burnam, W. H. Rogers, S. Greenfield, and J. E. Ware, Jr. 1989. Detection of depressive disorder for patients receiving prepaid or fee-for-service care. Results from the Medical Outcomes Survey. *Journal of the American Medical Association* 262: 3298–3302. https://doi.org/10.1001/jama.1989.03430230083030.

Also see

[R] **lincom** — Linear combinations of parameters

[R] **nlcom** — Nonlinear combinations of parameters

[R] **predict** — Obtain predictions, residuals, etc., after estimation

[R] **test** — Test linear hypotheses after estimation

[R] **testnl** — Test nonlinear hypotheses after estimation

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