Title

stata.com

nlcom — Nonlinear combinations of estimators

Description

`nlcom` computes point estimates, standard errors, test statistics, significance levels, and confidence intervals for (possibly) nonlinear combinations of parameter estimates after any Stata estimation command, including survey estimation. Results are displayed in the usual table format used for displaying estimation results. Calculations are based on the "delta method", an approximation appropriate in large samples.

Quick start

Estimate the ratio of the coefficient of `x2` to the coefficient of `x1`

```
nlcom _b[x2]/_b[x1]
```

Also estimate the ratio of the coefficient of `x3` to coefficient of `x1`

```
nlcom (_b[x2]/_b[x1]) (_b[x3]/_b[x1])
```

Add labels to the ratios

```
nlcom (r21:_b[x2]/_b[x1]) (r31:_b[x3]/_b[x1])
```

As above, but post estimates and use the `test` command to test that both ratios are equal to 1

```
nlcom (r21:_b[x2]/_b[x1]) (r31:_b[x3]/_b[x1]), post
test (r21 = 1) (r31 = 1)
```

Estimate the ratio of the coefficients of factor indicators `2.a` and `3.a`

```
nlcom _b[2.a]/_b[3.a]
```

Estimate the ratio of the coefficients of `x1` in the equations for `y1` and `y2` in a multiequation model

```
nlcom _b[y1:x1]/_b[y2:x1]
```

Menu

Statistics > Postestimation

Syntax

Nonlinear combination of estimators—one expression

nlcom [*name*:]*exp* [, *options*]

Nonlinear combinations of estimators—more than one expression

nlcom ([*name*:]*exp*) [([*name*:]*exp*) ...] [, *options*]

options	Description
level(#)	set confidence level; default is level(95)
iterate(#)	maximum number of iterations
post	post estimation results
display_options	control column formats and line width
noheader	suppress output header
df(#)	use t distribution with # degrees of freedom for computing p-values and confidence intervals

noheader and df(#) do not appear in the dialog box.

The second syntax means that if more than one expression is specified, each must be surrounded by parentheses. The optional *name* is any valid Stata name and labels the transformations.

exp is a possibly nonlinear expression containing

_b[*coef*]
_b[*eqno*:*coef*]
[*eqno*]*coef*
[*eqno*]_b[*coef*]

eqno is

##
name

coef identifies a coefficient in the model. *coef* is typically a variable name, a level indicator, an interaction indicator, or an interaction involving continuous variables. Level indicators identify one level of a factor variable and interaction indicators identify one combination of levels of an interaction; see [U] **11.4.3 Factor variables**. *coef* may contain time-series operators; see [U] **11.4.4 Time-series varlists**.

Distinguish between [], which are to be typed, and [ ], which indicate optional arguments.

Options

level(#) specifies the confidence level, as a percentage, for confidence intervals. The default is level(95) or as set by set level; see [U] **20.7 Specifying the width of confidence intervals**.

iterate(#) specifies the maximum number of iterations used to find the optimal step size in calculating numerical derivatives of the transformation(s) with respect to the original parameters. 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.

post causes nlcom to behave like a Stata estimation (eclass) command. When post is specified, nlcom will post the vector of transformed estimators and its estimated variance–covariance matrix to e(). This option, in essence, makes the transformation permanent. Thus you could, after posting, treat the transformed estimation results in the same way as you would treat results from other Stata estimation commands. For example, after posting, you could redisplay the results by typing nlcom without any arguments, or use test to perform simultaneous tests of hypotheses on linear combinations of the transformed estimators; see [R] **test**.

Specifying post clears out the previous estimation results, which can be recovered only by refitting the original model or by storing the estimation results before running nlcom and then restoring them; see [R] **estimates store**.

display_options: cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **estimation options**.

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

noheader suppresses the output header.

df(#) specifies that the t distribution with # degrees of freedom be used for computing p-values and confidence intervals.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Introduction

nlcom and predictnl both use the delta method. They take nonlinear transformations 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 the transformations. nlcom is designed for functions of the parameters, and predictnl is designed for functions of the parameters and of the data, that is, for predictions.

nlcom generalizes lincom (see [R] **lincom**) in two ways. First, nlcom allows the transformations to be nonlinear. Second, nlcom can be used to simultaneously estimate many transformations (whether linear or nonlinear) and to obtain the estimated variance–covariance matrix of these transformations.

Basics

In [R] **lincom**, the following regression was performed:

```
. use http://www.stata-press.com/data/r14/regress
. regress y x1 x2 x3
      Source |       SS           df       MS      Number of obs   =       148
-------------+----------------------------------   F(3, 144)       =     96.12
       Model |  3259.3561         3  1086.45203   Prob > F        =    0.0000
    Residual |  1627.56282       144  11.3025196   R-squared       =    0.6670
-------------+----------------------------------   Adj R-squared   =    0.6600
       Total |  4886.91892       147  33.2443464   Root MSE        =    3.3619

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   1.457113    1.07461     1.36   0.177     -.666934    3.581161
          x2 |   2.221682   .8610358     2.58   0.011     .5197797    3.923583
          x3 |   -.006139   .0005543   -11.08   0.000    -.0072345   -.0050435
       _cons |   36.10135   4.382693     8.24   0.000     27.43863    44.76407
------------------------------------------------------------------------------
```

Then `lincom` was used to estimate the difference between the coefficients of `x1` and `x2`:

```
. lincom _b[x2] - _b[x1]
 ( 1)  - x1 + x2 = 0

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         (1) |   .7645682   .9950282     0.77   0.444     -1.20218    2.731316
------------------------------------------------------------------------------
```

It was noted, however, that nonlinear expressions are not allowed with `lincom`:

```
. lincom _b[x2]/_b[x1]
not possible with test
r(131);
```

Nonlinear transformations are instead estimated using `nlcom`:

```
. nlcom _b[x2]/_b[x1]
       _nl_1:  _b[x2]/_b[x1]

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _nl_1 |   1.524714   .9812848     1.55   0.120    -.3985686    3.447997
------------------------------------------------------------------------------
```

❑ Technical note

The notation `_b[`*name*`]` is the standard way in Stata to refer to regression coefficients; see [U] **13.5 Accessing coefficients and standard errors**. Some commands, such as `lincom` and `test`, allow you to drop the `_b[]` and just refer to the coefficients by *name*. `nlcom`, however, requires the full specification `_b[`*name*`]`.

❑

Returning to our linear regression example, `nlcom` also allows simultaneous estimation of more than one combination:

```
. nlcom (_b[x2]/_b[x1]) (_b[x3]/_b[x1]) (_b[x3]/_b[x2])

       _nl_1:  _b[x2]/_b[x1]
       _nl_2:  _b[x3]/_b[x1]
       _nl_3:  _b[x3]/_b[x2]
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
_nl_1	1.524714	.9812848	1.55	0.120	-.3985686	3.447997
_nl_2	-.0042131	.0033483	-1.26	0.208	-.0107756	.0023494
_nl_3	-.0027632	.0010695	-2.58	0.010	-.0048594	-.000667

We can also label the transformations to produce more informative names in the estimation table:

```
. nlcom (ratio21:_b[x2]/_b[x1]) (ratio31:_b[x3]/_b[x1]) (ratio32:_b[x3]/_b[x2])

     ratio21:  _b[x2]/_b[x1]
     ratio31:  _b[x3]/_b[x1]
     ratio32:  _b[x3]/_b[x2]
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
ratio21	1.524714	.9812848	1.55	0.120	-.3985686	3.447997
ratio31	-.0042131	.0033483	-1.26	0.208	-.0107756	.0023494
ratio32	-.0027632	.0010695	-2.58	0.010	-.0048594	-.000667

`nlcom` stores the vector of estimated combinations and its estimated variance–covariance matrix in `r()`.

```
. matrix list r(b)

r(b)[1,3]
       ratio21     ratio31     ratio32
c1   1.5247143  -.00421315  -.00276324

. matrix list r(V)

symmetric r(V)[3,3]
             ratio21     ratio31     ratio32
ratio21    .96291982
ratio31   -.00287781   .00001121
ratio32   -.00014234   2.137e-06   1.144e-06
```

Using the post option

When used with the post option, nlcom stores the estimation vector and variance–covariance matrix in e(), making the transformation permanent:

```
. quietly nlcom (ratio21:_b[x2]/_b[x1]) (ratio31:_b[x3]/_b[x1])
>  (ratio32:_b[x3]/_b[x2]), post

. matrix list e(b)

e(b)[1,3]
        ratio21     ratio31     ratio32
y1    1.5247143  -.00421315  -.00276324

. matrix list e(V)

symmetric e(V)[3,3]
             ratio21     ratio31     ratio32
ratio21    .96291982
ratio31   -.00287781   .00001121
ratio32   -.00014234   2.137e-06   1.144e-06
```

After posting, we can proceed as if we had just run a Stata estimation (eclass) command. For instance, we can replay the results,

```
. nlcom

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     ratio21 |   1.524714   .9812848     1.55   0.120    -.3985686    3.447997
     ratio31 |  -.0042131   .0033483    -1.26   0.208    -.0107756    .0023494
     ratio32 |  -.0027632   .0010695    -2.58   0.010    -.0048594    -.000667
------------------------------------------------------------------------------
```

or perform other postestimation tasks in the transformed metric, this time making reference to the new "coefficients":

```
. display _b[ratio31]
-.00421315

. estat vce, correlation

Correlation matrix of coefficients of nlcom model

        e(V) |  ratio21   ratio31   ratio32
-------------+------------------------------
     ratio21 |   1.0000
     ratio31 |  -0.8759    1.0000
     ratio32 |  -0.1356    0.5969    1.0000

. test _b[ratio21] = 1

 ( 1)  ratio21 = 1

           chi2(  1) =    0.29
         Prob > chi2 =    0.5928
```

We see that testing _b[ratio21]=1 in the transformed metric is equivalent to testing using testnl _b[x2]/_b[x1]=1 in the original metric:

```
. quietly regress y x1 x2 x3

. testnl _b[x2]/_b[x1] = 1

  (1)  _b[x2]/_b[x1] = 1

               chi2(1) =        0.29
           Prob > chi2 =        0.5928
```

We needed to refit the regression model to recover the original parameter estimates.

❑ Technical note

In a previous technical note, we mentioned that commands such as `lincom` and `test` permit reference to *name* instead of `_b[`*name*`]`. This is not the case when `lincom` and `test` are used after `nlcom, post`. In the above, we used

```
. test _b[ratio21] = 1
```

rather than

```
. test ratio21 = 1
```

which would have returned an error. Consider this a limitation of Stata. For the shorthand notation to work, you need a variable named *name* in the data. In `nlcom`, however, *name* is just a coefficient label that does not necessarily correspond to any variable in the data.

❑

Reparameterizing ML estimators for univariate data

When run using only a response and no covariates, Stata's maximum likelihood (ML) estimation commands will produce ML estimates of the parameters of some assumed univariate distribution for the response. The parameterization, however, is usually not one we are used to dealing with in a nonregression setting. In such cases, `nlcom` can be used to transform the estimation results from a regression model to those from a maximum likelihood estimation of the parameters of a univariate probability distribution in a more familiar metric.

▷ Example 1

Consider the following univariate data on Y = # of traffic accidents at a certain intersection in a given year:

```
. use http://www.stata-press.com/data/r14/trafint
. summarize accidents
```

Variable	Obs	Mean	Std. Dev.	Min	Max
accidents	12	13.83333	14.47778	0	41

A quick glance of the output from `summarize` leads us to quickly reject the assumption that Y is distributed as Poisson because the estimated variance of Y is much greater than the estimated mean of Y.

Instead, we choose to model the data as univariate negative binomial, of which a common parameterization is

$$\Pr(Y=y)=\frac{\Gamma(r+y)}{\Gamma(r)\Gamma(y+1)}p^r(1-p)^y \qquad 0\le p\le 1, \quad r>0, \quad y=0,1,\ldots$$

with

$$E(Y)=\frac{r(1-p)}{p} \qquad \mathrm{Var}(Y)=\frac{r(1-p)}{p^2}$$

There exist no closed-form solutions for the maximum likelihood estimates of p and r, yet they may be estimated by the iterative method of Newton–Raphson. One way to get these estimates would be to write our own Newton–Raphson program for the negative binomial. Another way would be to write our own ML evaluator; see [R] **ml**.

The easiest solution, however, would be to use Stata's existing negative binomial ML regression command, `nbreg`. The only problem with this solution is that `nbreg` estimates a different parameterization of the negative binomial, but we can worry about that later.

```
. nbreg accidents
Fitting Poisson model:
Iteration 0:   log likelihood = -105.05361
Iteration 1:   log likelihood = -105.05361
Fitting constant-only model:
Iteration 0:   log likelihood = -43.948619
Iteration 1:   log likelihood = -43.891483
Iteration 2:   log likelihood =  -43.89144
Iteration 3:   log likelihood =  -43.89144
Fitting full model:
Iteration 0:   log likelihood =  -43.89144
Iteration 1:   log likelihood =  -43.89144
Negative binomial regression                    Number of obs     =         12
                                                LR chi2(0)        =       0.00
Dispersion     = mean                           Prob > chi2       =          .
Log likelihood =  -43.89144                     Pseudo R2         =     0.0000

------------------------------------------------------------------------------
   accidents |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |   2.627081   .3192233     8.23   0.000     2.001415    3.252747
-------------+----------------------------------------------------------------
    /lnalpha |   .1402425   .4187147                     -.6804233    .9609083
-------------+----------------------------------------------------------------
       alpha |   1.150553   .4817534                      .5064026     2.61407
------------------------------------------------------------------------------
LR test of alpha=0: chibar2(01) = 122.32               Prob >= chibar2 = 0.000
. nbreg, coeflegend
Negative binomial regression                    Number of obs     =         12
                                                LR chi2(0)        =       0.00
Dispersion     = mean                           Prob > chi2       =          .
Log likelihood =  -43.89144                     Pseudo R2         =     0.0000

------------------------------------------------------------------------------
   accidents |      Coef.  Legend
-------------+----------------------------------------------------------------
       _cons |   2.627081  _b[accidents:_cons]
-------------+----------------------------------------------------------------
    /lnalpha |   .1402425  _b[lnalpha:_cons]
-------------+----------------------------------------------------------------
       alpha |   1.150553
------------------------------------------------------------------------------
LR test of alpha=0: chibar2(01) = 122.32               Prob >= chibar2 = 0.000
```

From this output, we see that, when used with univariate data, `nbreg` estimates a regression intercept, β_0, and the logarithm of some parameter α. This parameterization is useful in regression models: β_0 is the intercept meant to be augmented with other terms of the linear predictor, and α is an overdispersion parameter used for comparison with the Poisson regression model.

However, we need to transform $(\beta_0, \ln\alpha)$ to (p, r). Examining *Methods and formulas* of [R] **nbreg** reveals the transformation as

$$p = \{1 + \alpha \exp(\beta_0)\}^{-1} \qquad r = \alpha^{-1}$$

which we apply using `nlcom`:

```
. nlcom (p:1/(1 + exp([lnalpha]_b[_cons] + _b[_cons])))
> (r:exp(-[lnalpha]_b[_cons]))
            p:  1/(1 + exp([lnalpha]_b[_cons] + _b[_cons]))
            r:  exp(-[lnalpha]_b[_cons])
```

accidents	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
p	.0591157	.0292857	2.02	0.044	.0017168	.1165146
r	.8691474	.3639248	2.39	0.017	.1558679	1.582427

Given the invariance of maximum likelihood estimators and the properties of the delta method, the above parameter estimates, standard errors, etc., are precisely those we would have obtained had we instead performed the Newton–Raphson optimization in the (p, r) metric.

◁

❑ Technical note

Note how we referred to the estimate of $\ln\alpha$ above as `[lnalpha]_b[_cons]`. This is not entirely evident from the output of `nbreg`, which is why we redisplayed the results using the `coeflegend` option so that we would know how to refer to the coefficients; **[U] 13.5 Accessing coefficients and standard errors**.

❑

nlcom versus eform

Many Stata estimation commands allow you to display exponentiated regression coefficients, some by default, some optionally. Known as "`eform`" in Stata terminology, this reparameterization serves many uses: it gives odds ratios for logistic models, hazard ratios in survival models, incidence-rate ratios in Poisson models, and relative-risk ratios in multinomial logit models, to name a few.

For example, consider the following estimation taken directly from the technical note in [R] **poisson**:

```
. use http://www.stata-press.com/data/r14/airline
. generate lnN = ln(n)
. poisson injuries XYZowned lnN
Iteration 0:   log likelihood = -22.333875
Iteration 1:   log likelihood = -22.332276
Iteration 2:   log likelihood = -22.332276
Poisson regression                              Number of obs     =          9
                                                LR chi2(2)        =      19.15
                                                Prob > chi2       =     0.0001
Log likelihood = -22.332276                     Pseudo R2         =     0.3001
```

injuries	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
XYZowned	.6840667	.3895877	1.76	0.079	-.0795111	1.447645
lnN	1.424169	.3725155	3.82	0.000	.6940517	2.154285
_cons	4.863891	.7090501	6.86	0.000	3.474178	6.253603

When we replay results and specify the `irr` (incidence-rate ratios) option,

```
. poisson, irr
Poisson regression                              Number of obs     =          9
                                                LR chi2(2)        =      19.15
                                                Prob > chi2       =     0.0001
Log likelihood = -22.332276                     Pseudo R2         =     0.3001
```

injuries	IRR	Std. Err.	z	P>|z|	[95% Conf.	Interval]
XYZowned	1.981921	.7721322	1.76	0.079	.9235678	4.253085
lnN	4.154402	1.547579	3.82	0.000	2.00181	8.621728
_cons	129.5272	91.84126	6.86	0.000	32.2713	519.8828

we obtain the exponentiated regression coefficients and their estimated standard errors.

Contrast this with what we obtain if we exponentiate the coefficients manually by using `nlcom`:

```
. nlcom (E_XYZowned:exp(_b[XYZowned])) (E_lnN:exp(_b[lnN]))
  E_XYZowned:  exp(_b[XYZowned])
       E_lnN:  exp(_b[lnN])
```

injuries	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
E_XYZowned	1.981921	.7721322	2.57	0.010	.4685701	3.495273
E_lnN	4.154402	1.547579	2.68	0.007	1.121203	7.187602

There are three things to note when comparing `poisson, irr` (and `eform` in general) with `nlcom`:

1. The exponentiated coefficients and standard errors are identical. This is certainly good news.
2. The Wald test statistic (`z`) and level of significance are different. When using `poisson, irr` and other related `eform` options, the Wald test does not change from what you would have obtained without the `eform` option, and you can see this by comparing both versions of the `poisson` output given previously.

   When you use `eform`, Stata knows that what is usually desired is a test of

   $$H_0\colon\ \exp(\beta) = 1$$

   and not the uninformative-by-comparison

   $$H_0\colon\ \exp(\beta) = 0$$

   The test of $H_0\colon\ \exp(\beta) = 1$ is asymptotically equivalent to a test of $H_0\colon \beta = 0$, the Wald test in the original metric, but the latter has better small-sample properties. Thus if you specify `eform`, you get a test of $H_0\colon \beta = 0$.

   `nlcom`, however, is general. It does not attempt to infer the test of greatest interest for a given transformation, and so a test of

   $$H_0\colon \text{transformed coefficient} = 0$$

   is always given, regardless of the transformation.
3. You may be surprised to see that, even though the coefficients and standard errors are identical, the confidence intervals (both 95%) are different.

eform confidence intervals are standard confidence intervals with the endpoints transformed. For example, the confidence interval for the coefficient on lnN is $[0.694, 2.154]$, whereas the confidence interval for the incidence-rate ratio due to lnN is $[\exp(0.694), \exp(2.154)] = [2.002, 8.619]$, which, except for some roundoff error, is what we see from the output of poisson, irr. For exponentiated coefficients, confidence intervals based on transform-the-endpoints methodology generally have better small-sample properties than their asymptotically equivalent counterparts.

The transform-the-endpoints method, however, gives valid coverage only when the transformation is monotonic. nlcom uses a more general and asymptotically equivalent method for calculating confidence intervals, as described in *Methods and formulas*.

Stored results

nlcom stores the following in r():

Scalars

r(N)	number of observations
r(df_r)	residual degrees of freedom

Matrices

r(b)	vector of transformed coefficients
r(V)	estimated variance–covariance matrix of the transformed coefficients

If post is specified, nlcom also stores the following in e():

Scalars

e(N)	number of observations
e(df_r)	residual degrees of freedom
e(N_strata)	number of strata L, if used after svy
e(N_psu)	number of sampled PSUs n, if used after svy
e(rank)	rank of e(V)

Macros

e(cmd)	nlcom
e(predict)	program used to implement predict
e(properties)	b V

Matrices

e(b)	vector of transformed coefficients
e(V)	estimated variance–covariance matrix of the transformed coefficients
e(V_srs)	simple-random-sampling-without-replacement (co)variance $\widehat{V}_{\rm srswor}$, if svy
e(V_srswr)	simple-random-sampling-with-replacement (co)variance $\widehat{V}_{\rm srswr}$, if svy and fpc()
e(V_msp)	misspecification (co)variance $\widehat{V}_{\rm msp}$, if svy and available

Functions

e(sample)	marks estimation sample

Methods and formulas

Given a $1 \times k$ vector of parameter estimates, $\widehat{\boldsymbol{\theta}} = (\widehat{\theta}_1, \dots, \widehat{\theta}_k)$, consider the estimated p-dimensional transformation

$$g(\widehat{\boldsymbol{\theta}}) = [g_1(\widehat{\boldsymbol{\theta}}), g_2(\widehat{\boldsymbol{\theta}}), \dots, g_p(\widehat{\boldsymbol{\theta}})]$$

The estimated variance–covariance of $g(\widehat{\boldsymbol{\theta}})$ is given by

$$\widehat{\rm Var}\left\{g(\widehat{\boldsymbol{\theta}})\right\} = \mathbf{GVG}'$$

where $\mathbf{G}$ is the $p \times k$ matrix of derivatives for which

$$\mathbf{G}_{ij} = \left.\frac{\partial g_i(\boldsymbol{\theta})}{\partial \theta_j}\right|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}} \qquad i = 1, \dots, p \qquad j = 1, \dots, k$$

and $\mathbf{V}$ is the estimated variance–covariance matrix of $\widehat{\boldsymbol{\theta}}$. Standard errors are obtained as the square roots of the variances.

The Wald test statistic for testing

$$H_0\colon g_i(\boldsymbol{\theta}) = 0$$

versus the two-sided alternative is given by

$$Z_i = \frac{g_i(\widehat{\boldsymbol{\theta}})}{\left[\widehat{\text{Var}}_{ii}\left\{g(\widehat{\boldsymbol{\theta}})\right\}\right]^{1/2}}$$

When the variance–covariance matrix of $\widehat{\boldsymbol{\theta}}$ is an asymptotic covariance matrix, Z_i is approximately distributed as Gaussian. For linear regression, Z_i is taken to be approximately distributed as $t_{1,r}$ where r is the residual degrees of freedom from the original fitted model.

A $(1-\alpha) \times 100\%$ confidence interval for $g_i(\boldsymbol{\theta})$ is given by

$$g_i(\widehat{\boldsymbol{\theta}}) \pm z_{\alpha/2}\left[\widehat{\text{Var}}_{ii}\left\{g(\widehat{\boldsymbol{\theta}})\right\}\right]^{1/2}$$

for those cases where Z_i is Gaussian and

$$g_i(\widehat{\boldsymbol{\theta}}) \pm t_{\alpha/2,r}\left[\widehat{\text{Var}}_{ii}\left\{g(\widehat{\boldsymbol{\theta}})\right\}\right]^{1/2}$$

for those cases where Z_i is t 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

Feiveson, A. H. 1999. FAQ: What is the delta method and how is it used to estimate the standard error of a transformed parameter? http://www.stata.com/support/faqs/stat/deltam.html.

Gould, W. W. 1996. crc43: Wald test of nonlinear hypotheses after model estimation. *Stata Technical Bulletin* 29: 2–4. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 15–18. College Station, TX: Stata Press.

Oehlert, G. W. 1992. A note on the delta method. *American Statistician* 46: 27–29.

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

Also see

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

[R] **predictnl** — Obtain nonlinear predictions, standard errors, etc., after estimation

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

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

[SVY] **svy postestimation** — Postestimation tools for svy

[U] 20 Estimation and postestimation commands