testnl — Test nonlinear hypotheses after estimation

Description

`testnl` tests (linear or nonlinear) hypotheses about the estimated parameters from the most recently fit model.

`testnl` produces Wald-type tests of smooth nonlinear (or linear) hypotheses about the estimated parameters from the most recently fit model. The p-values are based on the delta method, an approximation appropriate in large samples.

`testnl` can be used with `svy` estimation results; see [SVY] **svy postestimation**.

The format `(`exp_1`=`exp_2`=`exp_3`...)` for a simultaneous-equality hypothesis is just a convenient shorthand for a list `(`exp_1`=`exp_2`)` `(`exp_1`=`exp_3`)`, etc.

`testnl` may also be used to test linear hypotheses. `test` is faster if you want to test only linear hypotheses; see [R] **test**. `testnl` is the only option for testing linear and nonlinear hypotheses simultaneously.

Quick start

After single-equation models

Test that the product of the coefficients for `x1` and `x2` is equal to 4

```
testnl _b[x1]*_b[x2] = 4
```

Test that the ratio of the indicators for the factor variable `a` = 2 and `a` = 3 is 1

```
testnl _b[2.a]/_b[3.a] = 1
```

Test that an expression involving continuous factor variable syntax is equal to 16

```
testnl -_b[x1]/(2*_b[c.x1#c.x1]) = 16
```

Test the equality of two expressions

```
testnl _b[x1]*_b[x2] = _b[x1]*_b[x3]
```

Joint test that two products are both equal to 2

```
testnl (_b[x1]*_b[x2] = 2) (_b[x1]*_b[x3] = 2)
```

Same as above

```
testnl _b[x1]*_b[x2] = _b[x1]*_b[x3] = 2
```

Same as above, but add separate tests for each expression

```
testnl _b[x1]*_b[x2] = _b[x1]*_b[x3] = 2, mtest
```

Same as above, but adjust p-values for multiple comparisons using Holm's method

```
testnl _b[x1]*_b[x2] = _b[x1]*_b[x3] = 2, mtest(holm)
```

Test a linear hypothesis and a nonlinear hypothesis together

```
testnl (_b[x1] =_b[x2]) (_b[x2]^2 =_b[x3])
```

After multiple-equation models

Test that the product of the coefficients for `x1` and `x2` in the equation for `y1` is equal to 1

```
testnl _b[y1:x1]*_b[y1:x2] = 1
```

Test that the product of the coefficients for `x1` and `x2` in the equation for `y2` is equal to 1

```
testnl _b[y2:x1]*_b[y2:x2] = 1
```

Test the equality of expressions involving coefficients from the equations for `y1` and `y4`

```
testnl _b[y1:x1]*_b[y1:x2] = _b[y4:x1]*_b[y4:x2]
```

Menu

Statistics > Postestimation

Syntax

```
testnl exp = exp [= exp ...] [, options]

testnl (exp = exp [= exp ...]) [(exp = exp [= exp ...]) ...] [, options]
```

options	Description
mtest[(*opt*)]	test each condition separately
iterate(#)	use maximum # of iterations to find the optimal step size
df(#)	use F distribution with # denominator degrees of freedom for the reference distribution of the test statistic
nosvyadjust	carry out the Wald test as $W/k \sim F(k,d)$; for use with svy estimation commands when the df() option is also specified

collect is allowed; see **[U] 11.1.10 Prefix commands**.
df(#) and nosvyadjust 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.

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

mtest[(*opt*)] specifies that tests be performed for each condition separately. *opt* specifies the method for adjusting p-values for multiple testing. Valid values for *opt* are

bonferroni	Bonferroni's method
holm	Holm's method
sidak	Šidák's method
noadjust	no adjustment is to be made

Specifying mtest without an argument is equivalent to specifying mtest(noadjust).

`iterate(#)` specifies the maximum number of iterations used to find the optimal step size in the calculation of numerical derivatives of the test expressions. 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.

The following options are available with `testnl` but are 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. With survey data, # is the design degrees of freedom unless `nosvyadjust` is specified.

`nosvyadjust` is for use with `svy` estimation commands when the `df()` option is also specified; see [SVY] **svy estimation**. It specifies that the Wald test be carried out without the default adjustment for the design degrees of freedom. That is, the test is carried out as $W/k \sim F(k,d)$ rather than as $(d-k+1)W/(kd) \sim F(k,d-k+1)$, where $k =$ the dimension of the test and $d =$ the design degrees of freedom specified in the `df()` option.

Remarks and examples

Remarks are presented under the following headings:

Introduction

▷ Example 1

We have just estimated the parameters of an earnings model on cross-sectional time-series data using one of Stata's more sophisticated estimators:

```
. use https://www.stata-press.com/data/r19/earnings
(NLS women 14-24 in 1968)
. xtgee ln_w grade age c.age#c.age, corr(exchangeable) nolog
GEE population-averaged model                   Number of obs    =  1,326
Group variable: idcode                          Number of groups =    269
Family: Gaussian                                Obs per group:
Link:   Identity                                              min =      1
Correlation: exchangeable                                     avg =    4.9
                                                              max =      9
                                                Wald chi2(3)     = 327.33
Scale parameter = .0976738                      Prob > chi2      = 0.0000
```

ln_wage	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
grade	.0749686	.0066111	11.34	0.000	.062011	.0879261
age	.1080806	.0235861	4.58	0.000	.0618526	.1543086
c.age#c.age	-.0016253	.0004739	-3.43	0.001	-.0025541	-.0006966
_cons	-.8788933	.2830899	-3.10	0.002	-1.433739	-.3240473

An implication of this model is that peak earnings occur at age `-_b[age]/(2*_b[c.age#c.age])`, which here is equal to 33.2. Say that we have a theory that peak earnings should occur at age `16 + 1/_b[grade]`.

```
. testnl -_b[age]/(2*_b[c.age#c.age]) = 16 + 1/_b[grade]
  (1)  -_b[age]/(2*_b[c.age#c.age]) = 16 + 1/_b[grade]
               chi2(1) =        1.71
           Prob > chi2 =        0.1914
```

These data do not reject our theory.

◁

Using testnl to perform linear tests

`testnl` may be used to test linear constraints, but `test` is faster; see [R] **test**. You could type

```
. testnl _b[x4] = _b[x1]
```

but it would take less computer time if you typed

```
. test _b[x4] = _b[x1]
```

Specifying constraints

The constraints to be tested can be formulated in many different ways. You could type

```
. testnl _b[mpg]*_b[weight] = 1
```

or

```
. testnl _b[mpg] = 1/_b[weight]
```

or you could express the constraint any other way you wished. (To say that `testnl` allows constraints to be specified in different ways does not mean that the test itself does not depend on the formulation. This point is briefly discussed later.) In formulating the constraints, you must, however, exercise one caution: users of `test` often refer to the coefficient on a variable by specifying the variable name. For example,

```
. test mpg = 0
```

More formally, they should type

```
. test _b[mpg] = 0
```

but `test` allows the `_b[]` surrounding the variable name to be omitted. `testnl` does not allow this shorthand. Typing

```
. testnl mpg=0
```

specifies the constraint that the value of variable `mpg` in the first observation is zero. If you make this mistake, sometimes `testnl` will catch it:

```
. testnl mpg=0
equation (1) contains reference to X rather than _b[X]
r(198);
```

In other cases, testnl may not catch the mistake; then, the constraint will be dropped because it does not make sense:

```
. testnl mpg=0
  Constraint (1) dropped
```

(There are reasons other than this for constraints being dropped.) The worst case, however, is

```
. testnl _b[weight]*mpg = 1
```

when what you mean is not that _b[weight] equals the reciprocal of the value of mpg in the first observation, but rather that

```
. testnl _b[weight]*_b[mpg] = 1
```

Sometimes, this mistake will be caught by the "contains reference to X rather than _b[X]" error, and sometimes it will not. Be careful.

testnl, like test, can be used after any Stata estimation command, including the survey estimators. When you use it after a multiple-equation command, such as mlogit or heckman, you refer to coefficients by using Stata's standard syntax: [*eqname*]_b[*varname*].

Stata's single-equation estimation output looks like this:

```
             |    Coef   ...
-------------+----------------------------------------------------------
      weight |   12.27   ...    <-  coefficient is  _b[weight]
         mpg |    3.21   ...
-------------------------------------------------------------------------
```

Stata's multiple-equation output looks like this:

```
             |    Coef   ...
-------------+----------------------------------------------------------
cat1         |           ...
      weight |   12.27   ...    <-  coefficient is  [cat1]_b[weight]
         mpg |    3.21   ...
-------------+----------------------------------------------------------
8            |           ...
      weight |    5.83   ...    <-  coefficient is  [8]_b[weight]
         mpg |    7.43   ...
-------------------------------------------------------------------------
```

Dropped constraints

testnl automatically drops constraints when

- They are nonbinding, for example, _b[mpg]=_b[mpg]. More subtle cases include

```
_b[mpg]*_b[weight] = 4
_b[weight] = 2
_b[mpg] = 2
```

  In this example, the third constraint is nonbinding because it is implied by the first two.

- They are contradictory, for example, _b[mpg]=2 and _b[mpg]=3. More subtle cases include

```
_b[mpg]*_b[weight] = 4
_b[weight] = 2
_b[mpg] = 3
```

  The third constraint contradicts the first two.

Multiple constraints

▷ Example 2

We illustrate the simultaneous test of a series of constraints using simulated data on labor-market promotion in a given year. We fit a probit model with separate effects for education, experience, and experience-squared for men and women.

```
. use https://www.stata-press.com/data/r19/promotion
(Simulated data on promotions)
. probit promo male male#c.(yedu yexp yexp2), nolog
Probit regression                                 Number of obs =    775
                                                  LR chi2(7)    = 424.42
                                                  Prob > chi2   = 0.0000
Log likelihood = -245.42768                       Pseudo R2     = 0.4637
```

promo	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
male	.6489974	.203739	3.19	0.001	.2496763	1.048318
male#c.yedu						
Female	.9730237	.1056136	9.21	0.000	.7660248	1.180023
Male	1.390517	.1527288	9.10	0.000	1.091174	1.68986
male#c.yexp						
Female	.4559544	.0901169	5.06	0.000	.2793285	.6325803
Male	1.422539	.1544255	9.21	0.000	1.11987	1.725207
male#c.yexp2						
Female	-.1027149	.0573059	-1.79	0.073	-.2150325	.0096026
Male	-.3749457	.1160113	-3.23	0.001	-.6023236	-.1475677
_cons	.9872018	.1148215	8.60	0.000	.7621559	1.212248

```
Note: 1 failure and 2 successes completely determined.
```

The effects of human capital seem to differ between men and women. A formal test confirms this.

```
. test (yedu#0.male = yedu#1.male) (yexp#0.male = yexp#1.male)
> (yexp2#0.male = yexp2#1.male)
 ( 1)  [promo]0b.male#c.yedu - [promo]1.male#c.yedu = 0
 ( 2)  [promo]0b.male#c.yexp - [promo]1.male#c.yexp = 0
 ( 3)  [promo]0b.male#c.yexp2 - [promo]1.male#c.yexp2 = 0
           chi2(  3) =   35.43
         Prob > chi2 =    0.0000
```

How do we interpret this gender difference? It has repeatedly been stressed (see, for example, Long [1997, 47–50]; Allison [1999]) that comparison of groups in binary response models, and similarly in other latent-variable models, is hampered by an identification problem: with β the regression coefficients for the latent variable and σ the standard deviation of the latent residual, only the β/σ are identified. In fact, in terms of the latent regression, the probit coefficients should be interpreted as β/σ, not as the β. If we cannot claim convincingly that the residual standard deviation σ does not vary between the sexes, equality of the regression coefficients β implies that the coefficients of the probit model for men and women are *proportional* but not necessarily equal. This is a nonlinear hypothesis in terms of the probit coefficients, not a linear one.

```
. testnl _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
> = _b[yexp2#1.male]/_b[yexp2#0.male]
  (1)  _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
  (2)  _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp2#1.male]/_b[yexp2#0.male]
               chi2(2) =        9.21
             Prob > chi2 =        0.0100
```

We conclude that we find fairly strong evidence against the proportionality of the coefficients, and hence we have to conclude that success in the labor market is produced in different ways by men and women. (But remember, these were simulated data.)

◁

▷ Example 3

The syntax for specifying the equality of multiple expressions is just a convenient shorthand for specifying a series of constraints, namely, that the first expression equals the second expression, the first expression also equals the third expression, etc. The Wald test performed and the output of `testnl` are the same whether we use the shorthand or we specify the series of constraints.

```
. testnl (_b[yedu#1.male]/_b[yedu#0.male] =
>         _b[yexp#1.male]/_b[yexp#0.male])
>        (_b[yedu#1.male]/_b[yedu#0.male] =
>         _b[yexp2#1.male]/_b[yexp2#0.male])
  (1)  _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
  (2)  _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp2#1.male]/_b[yexp2#0.male]

             chi2(2) =        9.21
           Prob > chi2 =        0.0100
```

Having established differences between men and women, we would like to do multiple testing between the ratios. Because we did not specify hypotheses in advance, we prefer to adjust the p-values of tests using, here, Bonferroni's method.

```
. testnl _b[yedu#1.male]/_b[yedu#0.male] =
>        _b[yexp#1.male]/_b[yexp#0.male] =
>        _b[yexp2#1.male]/_b[yexp2#0.male], mtest(b)
  (1)  _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
  (2)  _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp2#1.male]/_b[yexp2#0.male]

------------------------------------------------------------------------------
             |        chi2     df   p > chi2
-------------+----------------------------------------------------------------
         (1) |        6.89      1     0.0173*
         (2) |        0.93      1     0.6713*
-------------+----------------------------------------------------------------
         All |        9.21      2     0.0100
------------------------------------------------------------------------------
* Bonferroni-adjusted p-values
```

◁

Manipulability

Although `testnl` allows you to specify constraints in different ways that are mathematically equivalent, as noted above, this does not mean that the tests are the same. This difference is known as the manipulability of the Wald test for nonlinear hypotheses; also see [R] **boxcox**. The test might even be significant for one formulation but not significant for another formulation that is mathematically equivalent. Trying out different specifications to find a formulation with the desired p-value is totally inappropriate, though it may actually be fun to try. There is no variance under representation because the nonlinear Wald test is actually a standard Wald test for a linearization of the constraint, which depends on the particular specification. We note that the likelihood-ratio test is not manipulable in this sense.

From a statistical point of view, it is best to choose a specification of the constraints that is as linear is possible. Doing so usually improves the accuracy of the approximation of the null-distribution of the test by a χ^2 or an F distribution. The example above used the nonlinear Wald test to test whether the coefficients of human capital variables for men were proportional to those of women. A specification of proportionality of coefficients in terms of ratios of coefficients is fairly nonlinear if the coefficients in the denominator are close to 0. A more linear version of the test results from a bilinear formulation. Thus, instead of

```
. testnl _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
  (1)  _b[yedu#1.male]/_b[yedu#0.male] = _b[yexp#1.male]/_b[yexp#0.male]
               chi2(1) =        6.89
          Prob > chi2 =        0.0087
```

perhaps

```
. testnl _b[yedu#1.male]*_b[yexp#0.male] = _b[yedu#0.male]*_b[yexp#1.male]
  (1)  _b[yedu#1.male]*_b[yexp#0.male] = _b[yedu#0.male]*_b[yexp#1.male]
               chi2(1) =       13.95
          Prob > chi2 =        0.0002
```

is better, and in fact it has been suggested that the latter version of the test is more reliable. This assertion is confirmed by performing simulations and is in line with theoretical results of Phillips and Park (1988). There is strong evidence against the proportionality of human capital effects between men and women, implying for this example that differences in the residual variances between the sexes can be ruled out as the explanation of the sex differences in the analysis of labor market participation.

Stored results

`testnl` stores the following in `r()`:

Scalars

`r(df)`	degrees of freedom
`r(df_r)`	residual degrees of freedom
`r(chi2)`	χ^2
`r(p)`	p-value for Wald test
`r(F)`	F statistic

Macros

`r(mtmethod)`	method specified in `mtest()`

Matrices

`r(G)`	derivatives of $R(\mathbf{b})$ with respect to **b**; see *Methods and formulas* below
`r(R)`	$R(\mathbf{b}) - \mathbf{q}$; see *Methods and formulas* below
`r(mtest)`	multiple test results

Methods and formulas

After fitting a model, define **b** as the resulting $1 \times k$ parameter vector and **V** as the $k \times k$ covariance matrix. The (linear or nonlinear) hypothesis is given by $R(\mathbf{b}) = \mathbf{q}$, where R is a function returning a $j \times 1$ vector. The Wald test formula is (Greene 2018, 512–513)

$$W = \Big\{R(\mathbf{b}) - \mathbf{q}\Big\}' \Big(\mathbf{GVG}'\Big)^{-1} \Big\{R(\mathbf{b}) - \mathbf{q}\Big\}$$

where **G** is the derivative matrix of $R(\mathbf{b})$ with respect to **b**. W is distributed as χ^2 if **V** is an asymptotic covariance matrix. $F = W/j$ is distributed as F for linear regression.

The adjustment methods for multiple testing are described in [R] **test**. The adjustment for survey design effects is described in [SVY] **svy postestimation**.

References

Allison, P. D. 1999. Comparing logit and probit coefficients across groups. *Sociological Methods and Research* 28: 186–208. https://doi.org/10.1177/0049124199028002003.

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

Long, J. S. 1997. *Regression Models for Categorical and Limited Dependent Variables*. Thousand Oaks, CA: Sage.

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.

Also see

[R] **contrast** — Contrasts and linear hypothesis tests after estimation

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

[R] **lrtest** — Likelihood-ratio test after estimation

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

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

[U] 13.5 Accessing coefficients and standard errors

[U] 20 Estimation and postestimation commands