mi test — Test hypotheses after mi estimate

Description

`mi test` performs joint tests of coefficients.

`mi testtransform` performs joint tests of transformed coefficients as specified with `mi estimate` or `mi estimate using` (see [MI] **mi estimate** or [MI] **mi estimate using**).

Menu

Statistics > Multiple imputation

Syntax

Test that coefficients are zero

mi test *coeflist*

Test that coefficients within a single equation are zero

mi test [*eqno*] [: *coeflist*]

Test that subsets of coefficients are zero (full syntax)

mi test (*spec*) [(*spec*) ...] [, *test_options*]

Test that subsets of transformed coefficients are zero

mi testtransform *name* [(*name*) ...] [, *transform_options*]

test_options	Description
Test	
ufmitest	perform unrestricted FMI model test
nosmall	do not apply small-sample correction to degrees of freedom
constant	include the constant in coefficients to be tested

transform_options	Description
Test	
ufmitest	perform unrestricted FMI model test
nosmall	do not apply small-sample correction to degrees of freedom
nolegend	suppress transformation legend

coeflist may contain factor variables and time-series operators; see [U] **11.4.3 Factor variables** and [U] **11.4.4 Time-series varlists**.
collect is allowed with mi test; see [U] **11.1.10 Prefix commands**.

coeflist is

coef [*coef* ...]
[*eqno*]*coef* [[*eqno*]*coef* ...]
[*eqno*]_b[*coef*] [[*eqno*]_b[*coef*] ...]

eqno is

##
eqname

spec is

coeflist
[*eqno*] [: *coeflist*]

coef identifies a coefficient in the model; see the description in [R] **test** for details. *eqname* is an equation name.

name is an expression name as specified with mi estimate or mi estimate using (see [MI] **mi estimate** or [MI] **mi estimate using**).

Options

Test

`ufmitest` specifies that the unrestricted fraction missing information (FMI) model test be used. The default test performed assumes equal fractions of information missing due to nonresponse for all coefficients. This is equivalent to the assumption that the between-imputation and within-imputation variances are proportional. The unrestricted test may be preferable when this assumption is suspect provided that the number of imputations is large relative to the number of estimated coefficients.

`nosmall` specifies that no small-sample adjustment be made to the degrees of freedom. By default, individual tests of coefficients (and transformed coefficients) use the small-sample adjustment of Barnard and Rubin (1999), and the overall model test uses the small-sample adjustment of Reiter (2007).

`constant` specifies that `_cons` be included in the list of coefficients to be tested when using the `[`*eqno*`]` form of *spec* with `mi test`. The default is to not include `_cons`.

`nolegend`, specified with `mi testtransform`, suppresses the transformation legend.

Remarks and examples

Remarks are presented under the following headings:

Introduction

The major issue arising when performing tests after MI estimation is the validity of the variance–covariance estimator (VCE) of the MI estimates. MI variance consists of two sources of variation: within-imputation variation and between-imputation variation. With a small number of imputations, the estimate of the between-imputation variance–covariance matrix is imprecise. In fact, when the number of imputations is less than or equal to the number of estimated parameters, the between-imputation matrix does not even have a full rank. As such, the estimated VCE may not be a valid variance–covariance matrix and thus not suitable for joint inference.

One solution to this problem was proposed by Rubin (1987) and Li et al. (1991). The idea is to assume that the between-imputation variance is proportional to the within-imputation variance. This assumption implies equal FMIs for all jointly tested parameters. Li et al. (1991) found that the procedure performs well in terms of power and maintaining the significance level even with moderately variable FMIs. `mi test` and `mi testtransform`, by default, perform tests using this procedure.

When the number of imputations is large enough relative to the number of tested parameters so that the corresponding VCE is trustworthy, you can request the unrestricted FMI test by specifying the `ufmitest` option. The unrestricted FMI test is the conventional test described by Rubin (1987, 77).

For testing nonlinear hypotheses, direct application of the conventional delta method to the estimated coefficients may not be feasible when the number of imputations is small enough that the VCE of the MI estimates cannot be used for inference. To test these hypotheses, one can first obtain MI estimates of the transformed coefficients by applying Rubin's combination rules to the transformed completed-data estimates and then apply the above MI-specific hypotheses tests to the combined transformed estimates.

The first step can be done by specifying expressions with `mi estimate` (or `mi estimate using`). The second step is performed with `mi testtransform`. `mi testtransform` uses the same method to test transformed coefficients as `mi test` uses to test coefficients.

Overview

Use `mi test` to perform joint tests that coefficients are equal to zero:

```
. mi estimate: regress y x1 x2 x3 x4
. mi test x2 x3 x4
```

Use `mi testtransform`, however, to perform tests of more general linear hypotheses, such as `_b[x1]=_b[x2]`, or `_b[x1]=_b[x2]` and `_b[x1]=_b[x3]`. Testing general linear hypotheses requires estimation of between and within variances corresponding to the specific hypotheses and requires recombining the imputation-specific estimation results. One way you could do that would be to refit the model and include the additional parameters during the estimation step. To test `_b[x1]=_b[x2]`, you could type

```
. mi estimate (diff: _b[x1]-_b[x2]): regress y x1 x2 x3 x4
. mi testtransform diff
```

A better approach, however, is to save each of the imputation-specific results at the time the original model is fit and then later recombine results using `mi estimate using`. To save the imputation-specific results, specify `mi estimate`'s `saving()` option when the model is originally fit:

```
. mi estimate, saving(myresults): regress y x1 x2 x3 x4
```

To test `_b[x1]=_b[x2]`, you type

```
. mi estimate (diff: _b[x1]-_b[x2]) using myresults
. mi testtransform diff
```

The advantage of this approach is that you can test additional hypotheses without refitting the model. For instance, if we now wanted to test `_b[x1]=_b[x2]` and `_b[x1]=_b[x3]`, we could type

```
. mi estimate (diff1: _b[x1]-_b[x2]) (diff2: _b[x1]=_b[x3]) using myresults
. mi testtransform diff1 diff2
```

To test nonlinear hypotheses, such as `_b[x1]/_b[x2]=_b[x3]/_b[x4]`, we could then type

```
. mi estimate (diff: _b[x1]/_b[x2]-_b[x3]/_b[x4]) using myresults
. mi testtransform diff
```

Example 1: Testing subsets of coefficients equal to zero

We are going to test that `tax`, `sqft`, `age`, `nfeatures`, `ne`, `custom`, and `corner` are in the regression analysis of house resale prices we performed in *Example 1: Completed-data logistic analysis* of [MI] **mi estimate**. Following the advice above, when we fit the model, we are going to save the imputation-specific results even though we will not need them in this example; we will need them in the following examples.

```
. use https://www.stata-press.com/data/r19/mhouses1993s30
(Albuquerque home prices Feb15-Apr30, 1993)

. mi estimate, saving(miest): regress price tax sqft age nfeatures ne custom
> corner

Multiple-imputation estimates                   Imputations       =         30
Linear regression                               Number of obs     =        117
                                                Average RVI       =     0.0648
                                                Largest FMI       =     0.2533
                                                Complete DF       =        109
DF adjustment:   Small sample                   DF:     min       =      69.12
                                                        avg       =      94.02
                                                        max       =     105.51
Model F test:       Equal FMI                   F(   7,  106.5)   =      67.18
Within VCE type:          OLS                   Prob > F          =     0.0000

------------------------------------------------------------------------------
       price | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
         tax |   .6768015   .1241568     5.45   0.000     .4301777    .9234253
        sqft |   .2118129    .069177     3.06   0.003     .0745091    .3491168
         age |   .2471445   1.653669     0.15   0.882    -3.051732    3.546021
   nfeatures |   9.288033   13.30469     0.70   0.487    -17.12017    35.69623
          ne |   2.518996   36.99365     0.07   0.946    -70.90416    75.94215
      custom |   134.2193   43.29755     3.10   0.002     48.35674    220.0818
      corner |  -68.58686    39.9488    -1.72   0.089    -147.7934    10.61972
       _cons |   123.9118   71.05816     1.74   0.085    -17.19932    265.0229
------------------------------------------------------------------------------
```

In the above `mi estimate` command, we use the `saving()` option to create a Stata estimation file called `miest.ster`, which contains imputation-specific estimation results.

`mi estimate` reports the joint test of all coefficients equal to zero in the header. We can reproduce this test with `mi test` by typing

```
. mi test tax sqft age nfeatures ne custom corner
note: assuming equal fractions of missing information.

 ( 1)  tax = 0
 ( 2)  sqft = 0
 ( 3)  age = 0
 ( 4)  nfeatures = 0
 ( 5)  ne = 0
 ( 6)  custom = 0
 ( 7)  corner = 0

       F(  7, 106.5) =   67.18
            Prob > F =    0.0000
```

We obtain results identical to those from `mi estimate`.

We can test that a subset of coefficients, say, `sqft` and `tax`, are equal to zero by typing

```
. mi test sqft tax
note: assuming equal fractions of missing information.

 ( 1)  sqft = 0
 ( 2)  tax = 0

       F(  2, 105.7) =  114.75
            Prob > F =    0.0000
```

Example 2: Testing linear hypotheses

Now we want to test the equality of the coefficients for `sqft` and `tax`. Following our earlier suggestion, we use `mi estimate using` to estimate the difference between coefficients (and avoid refitting the models) and then use `mi testtransform` to test that the difference is zero:

```
. mi estimate (diff: _b[tax]-_b[sqft]) using miest, nocoef

Multiple-imputation estimates                   Imputations       =         30
Linear regression                               Number of obs     =        117
                                                Average RVI       =     0.1200
                                                Largest FMI       =     0.1100
                                                Complete DF       =        109
DF adjustment:   Small sample                   DF:     min       =      92.10
                                                        avg       =      92.10
Within VCE type:          OLS                           max       =      92.10

      command: regress price tax sqft age nfeatures ne custom corner
         diff: _b[tax]-_b[sqft]

------------------------------------------------------------------------------
       price | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
        diff |   .4649885   .1863919     2.49   0.014     .0948037    .8351733
------------------------------------------------------------------------------

. mi testtransform diff
note: assuming equal fractions of missing information.

         diff: _b[tax]-_b[sqft]

 ( 1)  diff = 0

       F(  1,  92.1) =    6.22
            Prob > F =    0.0144
```

We suppress the display of the coefficient table by specifying the `nocoef` option with `mi estimate using`. We obtain the same results from the F test as those of the t test reported in the transformation table.

Similarly, we can test whether three coefficients are jointly equal:

```
. mi estimate (diff1: _b[tax]-_b[sqft]) (diff2: _b[custom]-_b[tax]) using miest,
> nocoef

Multiple-imputation estimates                   Imputations       =         30
Linear regression                               Number of obs     =        117
                                                Average RVI       =     0.0748
                                                Largest FMI       =     0.1100
                                                Complete DF       =        109
DF adjustment:   Small sample                   DF:     min       =      92.10
                                                        avg       =      97.95
Within VCE type:          OLS                           max       =     103.80

      command: regress price tax sqft age nfeatures ne custom corner
        diff1: _b[tax]-_b[sqft]
        diff2: _b[custom]-_b[tax]

------------------------------------------------------------------------------
       price | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
       diff1 |   .4649885   .1863919     2.49   0.014     .0948037    .8351733
       diff2 |   133.5425   43.30262     3.08   0.003     47.66984    219.4151
------------------------------------------------------------------------------

. mi testtr diff1 diff2
note: assuming equal fractions of missing information.

        diff1: _b[tax]-_b[sqft]
        diff2: _b[custom]-_b[tax]

 ( 1)  diff1 = 0
 ( 2)  diff2 = 0

       F(  2, 105.6) =    7.34
            Prob > F =    0.0010
```

We estimate two differences, `_b[tax]-_b[sqft]` and `_b[custom]-_b[tax]`, using `mi estimate using` and test whether they are jointly equal to zero by using `mi testtransform`.

We can perform tests of other hypotheses similarly by reformulating the hypotheses of interest such that we are testing equality to zero.

Example 3: Testing nonlinear hypotheses

In the examples above, we tested linear hypotheses. Testing nonlinear hypotheses is no different. We simply replace the specification of linear expressions in `mi estimate using` with the nonlinear expressions corresponding to the tests of interest.

For example, let's test that the ratio of the coefficients for `tax` and `sqft` is one, an equivalent but less efficient way of testing whether the two coefficients are the same. Similarly to the earlier example, we specify the corresponding nonlinear expression with `mi estimate using` and then use `mi testtransform` to test that the ratio is one:

```
. mi estimate (rdiff: _b[tax]/_b[sqft] - 1) using miest, nocoef
Multiple-imputation estimates                   Imputations       =         30
Linear regression                               Number of obs     =        117
                                                Average RVI       =     0.0951
                                                Largest FMI       =     0.0892
                                                Complete DF       =        109
DF adjustment:   Small sample                   DF:     min       =      95.33
                                                        avg       =      95.33
Within VCE type:          OLS                           max       =      95.33
      command: regress price tax sqft age nfeatures ne custom corner
        rdiff: _b[tax]/_b[sqft] - 1

------------------------------------------------------------------------------
       price | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
       rdiff |     2.2359   1.624546     1.38   0.172    -.9890876    5.460888
------------------------------------------------------------------------------

. mi testtr rdiff
note: assuming equal fractions of missing information.
        rdiff: _b[tax]/_b[sqft] - 1
 ( 1)  rdiff = 0
       F(  1,   95.3) =    1.89
            Prob > F =    0.1719
```

We do not need to use mi testtransform (or mi test) to test one transformation (or coefficient) because the corresponding test is provided in the output from mi estimate using.

Stored results

mi test and mi testtransform store the following in r():

Scalars

r(df)	test constraints degrees of freedom
r(df_r)	residual degrees of freedom
r(p)	two-sided p-value
r(F)	F statistic
r(drop)	1 if constraints were dropped, 0 otherwise
r(dropped_i)	index of ith constraint dropped

Methods and formulas

mi test and mi testtransform use the methodology described in *Multivariate case* under *Methods and formulas* of [MI] **mi estimate**, where we replace $\mathbf{q}$ with $\mathbf{Rq} - \mathbf{r}$ and $\mathbf{q}_0 = \mathbf{0}$ for the test H_0: $\mathbf{Rq} = \mathbf{r}$.

References

Barnard, J., and D. B. Rubin. 1999. Small-sample degrees of freedom with multiple imputation. *Biometrika* 86: 948–955. https://doi.org/10.1093/biomet/86.4.948.

Li, K.-H., X.-L. Meng, T. E. Raghunathan, and D. B. Rubin. 1991. Significance levels from repeated p-values with multiply-imputed data. *Statistica Sinica* 1: 65–92.

Reiter, J. P. 2007. Small-sample degrees of freedom for multi-component significance tests with multiple imputation for missing data. *Biometrika* 94: 502–508. https://doi.org/10.1093/biomet/asm028.

Rubin, D. B. 1987. *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley.

Also see

[MI] **mi estimate postestimation** — Postestimation tools for mi estimate

[MI] **mi estimate** — Estimation using multiple imputations

[MI] **mi estimate using** — Estimation using previously saved estimation results

[MI] **Intro** — Introduction to mi

[MI] **Intro substantive** — Introduction to multiple-imputation analysis

[MI] **Glossary**

For suggested citations, see the FAQ on citing Stata documentation.