Title

stata.com

_rmcoll — Remove collinear variables

Syntax Description Options Remarks and examples Stored results Also see

Syntax

Identify variables to be omitted because of collinearity

```
_rmcoll varlist [if] [in] [weight] [, noconstant collinear expand forcedrop]
```

Identify independent variables to be omitted because of collinearity

```
_rmdcoll depvar indepvars [if] [in] [weight] [, noconstant collinear expand
  normcoll]
```

varlist and *indepvars* may contain factor variables; see **[U] 11.4.3 Factor variables**.
varlist, *depvar*, and *indepvars* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.
`fweights`, `aweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**.

Description

`_rmcoll` returns in `r(varlist)` an updated version of *varlist* that is specific to the sample identified by `if`, `in`, and any missing values in *varlist*. `_rmcoll` flags variables that are to be omitted because of collinearity. If *varlist* contains factor variables, then `_rmcoll` also enumerates the levels of factor variables, identifies the base levels of factor variables, and identifies empty cells in interactions.

The following message is displayed for each variable that `_rmcoll` flags as omitted because of collinearity:

```
note: ______ omitted because of collinearity
```

The following message is displayed for each empty cell of an interaction that `_rmcoll` encounters:

```
note: ______ identifies no observations in the sample
```

`ml` users: it is not necessary to call `_rmcoll` because `ml` flags collinear variables for you, assuming that you do not specify `ml model`'s `collinear` option. Even so, `ml` programmers sometimes use `_rmcoll` because they need the sample-specific set of variables, and in such cases, they specify `ml model`'s `collinear` option so that `ml` does not waste time looking for collinearity again. See [R] **ml**.

`_rmdcoll` performs the same task as `_rmcoll` and checks that *depvar* is not collinear with the variables in *indepvars*. If *depvar* is collinear with any of the variables in *indepvars*, then `_rmdcoll` reports the following message with the 459 error code:

```
______ collinear with ______
```

Options

noconstant specifies that, in looking for collinearity, an intercept not be included. That is, a variable that contains the same nonzero value in every observation should not be considered collinear.

collinear specifies that collinear variables not be flagged.

expand specifies that the expanded, level-specific variables be posted to r(varlist). This option will have an effect only if there are factor variables in the variable list.

forcedrop specifies that collinear variables be dropped from the variable list instead of being flagged. This option is not allowed when the variable list already contains flagged variables, factor variables, or interactions.

normcoll specifies that collinear variables have already been flagged in *indepvars*. Otherwise, _rmcoll is called first to flag any such collinearity.

Remarks and examples

stata.com

_rmcoll and _rmdcoll are typically used when writing estimation commands.

_rmcoll is used if the programmer wants to flag the collinear variables from the independent variables.

_rmdcoll is used if the programmer wants to detect collinearity of the dependent variable with the independent variables.

▷ Example 1: Flagging variables because of collinearity

Let's load auto.dta and add a variable called tt that is collinear with variables turn and trunk. The easiest way to do this is to generate tt as the sum of turn and trunk.

```
. use http://www.stata-press.com/data/r13/auto
(1978 Automobile Data)
. generate tt = turn + trunk
```

Now we can use _rmcoll to identify that we have a collinearity and flag a variable because of it.

```
. _rmcoll turn trunk tt
note: tt omitted because of collinearity
. display r(varlist)
turn trunk o.tt
```

_rmcoll reported that tt was being flagged because of collinearity and attached the omit operator to tt resulting in "o.tt" being returned in r(varlist).

◁

▷ Example 2: Factor variables

_rmcoll works with factor variables. Let's pass rep78 as a factor variable to _rmcoll.

```
. _rmcoll i.rep78
. display r(varlist)
i(1 2 3 4 5)b1.rep78
```

The updated variable list now contains the enumerated levels of rep78 and identifies its base level. Use the expand option if you want to be able to loop over the level-specific, individual variables in r(varlist).

```
. _rmcoll i.rep78, expand
. display r(varlist)
1b.rep78 2.rep78 3.rep78 4.rep78 5.rep78
```

◁

▷ Example 3: Interactions

_rmcoll works with interactions and reports when it encounters empty cells. An empty cell is a combination of factor levels that does not occur in the dataset. Let's use the table command with factor variables rep78 and foreign to see that there are two empty cells:

```
. table rep78 foreign
```

Repair Record 1978	Car type Domestic	Foreign
1	2	
2	8	
3	27	3
4	9	9
5	2	9

Now let's pass the interaction of factor variables rep78 and foreign to _rmcoll.

```
. _rmcoll rep78#foreign
note: 1.rep78#1.foreign identifies no observations in the sample
note: 2.rep78#1.foreign identifies no observations in the sample
. display r(varlist)
i(1 2 3 4 5)b1o(1 1 2).rep78#i(0 1)b0o(0 1 1).foreign
```

◁

▷ Example 4: Coding fragment for standard variables

A code fragment for a program that uses _rmcoll might read

```
...
syntax varlist [fweight iweight] ... [, noCONStant ... ]
marksample touse
if "`weight'" != "" {
        tempvar w
        quietly gen double `w' = `exp' if `touse'
        local wgt [`weight'=`w']
}
else   local wgt /* is nothing */
gettoken depvar xvars : varlist
_rmcoll `xvars' `wgt' if `touse', `constant'
local xvars `r(varlist)'
...
```

In this code fragment, varlist contains one dependent variable and zero or more independent variables. The dependent variable is split off and stored in the local macro depvar. Then the remaining variables are passed through _rmcoll, and the resulting updated independent variable list is stored in the local macro xvars.

◁

▷ Example 5: Coding fragment for factor variables and time-series operators

Here we modified the above code fragment to allow for factor variables and time-series operators.

```
...
syntax varlist(fv ts) [fweight iweight] ... [, noCONStant ... ]
marksample touse
if "‘weight’" != "" {
        tempvar w
        quietly gen double ‘w’ = ‘exp’ if ‘touse’
        local wgt [‘weight’=‘w’]
}
else   local wgt /* is nothing */
gettoken depvar xvars : varlist
_rmcoll ‘xvars’ ‘wgt’ if ‘touse’, expand ‘constant’
local xvars ‘r(varlist)’
...
```

The varlist argument in the syntax command contains the fv specifier to allow factor variables and the ts specifier to allow time-series operators. We also added the expand option in case the remaining code needs to loop over the level-specific, individual variables in the xvars macro.

◁

Stored results

_rmcoll and _rmdcoll store the following in r():

Scalars

r(k_omitted)	number of omitted variables in r(varlist)

Macros

r(varlist)	the flagged and expanded variable list

Also see

[R] **ml** — Maximum likelihood estimation

[U] **18 Programming Stata**