cnsreg — Constrained linear regression

Description

cnsreg fits constrained linear regression models.

Quick start

Linear regression with coefficients for x1 and x2 constrained to equality

```
constraint 1 x1 = x2
cnsreg y x1 x2 x3, constraints(1)
```

Add constraint x2 = x3 to impose x1 = x2 = x3

```
constraint 2 x2 = x3
cnsreg y x1 x2 x3, constraints(1 2)
```

Constrain the coefficient for x4 to be −1

```
constraint 3 x4 = -1
cnsreg y x1 x2 x3 x4, constraints(1-3)
```

Menu

Statistics > Linear models and related > Constrained linear regression

Syntax

cnsreg *depvar indepvars* [*if*] [*in*] [*weight*] , constraints(*constraints*) [*options*]

options	Description
Model	
* constraints(*constraints*)	apply specified linear constraints
noconstant	suppress constant term
SE/Robust	
vce(*vcetype*)	*vcetype* may be ols, robust, cluster *clustvar*, bootstrap, or jackknife
Reporting	
level(#)	set confidence level; default is level(95)
nocnsreport	do not display constraints
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
mse1	force MSE to be 1
collinear	keep collinear variables
coeflegend	display legend instead of statistics

*constraints(*constraints*) is required.

indepvars may contain factor variables; see [U] **11.4.3 Factor variables**.

depvar and *indepvars* may contain time-series operators; see [U] **11.4.4 Time-series varlists**.

bayesboot, bootstrap, by, collect, fp, jackknife, mi estimate, rolling, statsby, and svy are allowed; see [U] **11.1.10 Prefix commands**.

vce(bootstrap) and vce(jackknife) are not allowed with the mi estimate prefix; see [MI] **mi estimate**.

With the fp prefix (see [R] **fp**), constraints cannot be specified for the variable containing fractional polynomial terms.

Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.

aweights are not allowed with the jackknife prefix; see [R] **jackknife**.

vce(), mse1, and weights are not allowed with the svy prefix; see [SVY] **svy**.

aweights, fweights, iweights, and pweights are allowed; see [U] **11.1.6 weight**.

mse1, collinear, and coeflegend do not appear in the dialog.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

constraints(*constraints*), noconstant; see [R] **Estimation options**.

SE/Robust

vce(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory (ols), that are robust to some kinds of misspecification (robust), that allow for intragroup correlation (cluster *clustvar*), and that use bootstrap or jackknife methods (bootstrap, jackknife); see [R] ***vce_option***.

vce(ols), the default, uses the standard variance estimator for ordinary least-squares regression.

Reporting

`level(#)`, `nocnsreport`; see [R] **Estimation options**.

display_options: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **Estimation options**.

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

`mse1` is used only in programs and ado-files that use `cnsreg` to fit models other than constrained linear regression. `mse1` sets the mean squared error to 1, thus forcing the variance–covariance matrix of the estimators to be $(\mathbf{X}'\mathbf{D}\mathbf{X})^{-1}$ (see *Methods and formulas* in [R] **regress**) and affecting calculated standard errors. Degrees of freedom for t statistics are calculated as n rather than $n - p + c$, where p is the total number of parameters (prior to restrictions and including the constant) and c is the number of constraints.

`mse1` is not allowed with the `svy` prefix.

`collinear`, `coeflegend`; see [R] **Estimation options**.

Remarks and examples

For a discussion of constrained linear regression, see Greene (2018, 126–127); Hill, Griffiths, and Lim (2018, 271–273); or Davidson and MacKinnon (1993, 17).

▷ Example 1: One constraint

In principle, we can obtain constrained linear regression estimates by modifying the list of independent variables. For instance, if we wanted to fit the model

$$\texttt{mpg} = \beta_0 + \beta_1\,\texttt{price} + \beta_2\,\texttt{weight} + u$$

and constrain $\beta_1 = \beta_2$, we could write

$$\texttt{mpg} = \beta_0 + \beta_1(\texttt{price} + \texttt{weight}) + u$$

and run a regression of `mpg` on `price + weight`. The estimated coefficient on the sum would be the constrained estimate of β_1 and β_2. Using `cnsreg`, however, is easier:

```
. use https://www.stata-press.com/data/r19/auto
(1978 automobile data)

. constraint 1 price = weight

. cnsreg mpg price weight, constraint(1)

Constrained linear regression                   Number of obs =        74
                                                F(1, 72)      =     37.59
                                                Prob > F      =    0.0000
                                                Root MSE      =    4.7220

 ( 1)  price - weight = 0

------------------------------------------------------------------------------
         mpg | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
       price |  -.0009875   .0001611    -6.13   0.000    -.0013086   -.0006664
      weight |  -.0009875   .0001611    -6.13   0.000    -.0013086   -.0006664
       _cons |   30.36718   1.577958    19.24   0.000     27.22158    33.51278
------------------------------------------------------------------------------
```

We define constraints by using the `constraint` command; see [R] **constraint**. We fit the model with `cnsreg` and specify the constraint number or numbers in the `constraints()` option.

Just to show that the results above are correct, here is the result of applying the constraint by hand:

```
. generate x = price + weight
. regress mpg x
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(1, 72)        =     37.59
       Model |  838.065767         1  838.065767   Prob > F        =    0.0000
    Residual |  1605.39369        72  22.2971346   R-squared       =    0.3430
-------------+----------------------------------   Adj R-squared   =    0.3339
       Total |  2443.45946        73  33.4720474   Root MSE        =     4.722

------------------------------------------------------------------------------
         mpg | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
           x |  -.0009875   .0001611    -6.13   0.000    -.0013086   -.0006664
       _cons |   30.36718   1.577958    19.24   0.000     27.22158    33.51278
------------------------------------------------------------------------------
```

◁

▷ Example 2: Multiple constraints

Models can be fit subject to multiple simultaneous constraints. We simply define the constraints and then include the constraint numbers in the `constraints()` option. For instance, say that we wish to fit the model

$$\texttt{mpg} = \beta_0 + \beta_1\,\texttt{price} + \beta_2\,\texttt{weight} + \beta_3\,\texttt{displ} + \beta_4\,\texttt{gear\_ratio} + \beta_5\,\texttt{foreign} + \beta_6\,\texttt{length} + u$$

subject to the constraints

$$\beta_1 = \beta_2 = \beta_3 = \beta_6$$
$$\beta_4 = -\beta_5 = \beta_0/20$$

(This model, like the one in example 1, is admittedly senseless.) We fit the model by typing

```
. constraint 1 price=weight
. constraint 2 displ=weight
. constraint 3 length=weight
. constraint 5 gear_ratio = -foreign
. constraint 6 gear_ratio = _cons/20
```

```
. cnsreg mpg price weight displ gear_ratio foreign length, c(1-3,5-6)
Constrained linear regression                   Number of obs =      74
                                                F(2, 72)      =  785.20
                                                Prob > F      =  0.0000
                                                Root MSE      =  4.6823
 ( 1)  price - weight = 0
 ( 2)  - weight + displacement = 0
 ( 3)  - weight + length = 0
 ( 4)  gear_ratio + foreign = 0
 ( 5)  gear_ratio - .05*_cons = 0
```

mpg	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
price	-.000923	.0001534	-6.02	0.000	-.0012288	-.0006172
weight	-.000923	.0001534	-6.02	0.000	-.0012288	-.0006172
displacement	-.000923	.0001534	-6.02	0.000	-.0012288	-.0006172
gear_ratio	1.326114	.0687589	19.29	0.000	1.189046	1.463183
foreign	-1.326114	.0687589	-19.29	0.000	-1.463183	-1.189046
length	-.000923	.0001534	-6.02	0.000	-.0012288	-.0006172
_cons	26.52229	1.375178	19.29	0.000	23.78092	29.26365

There are many ways we could have specified the `constraints()` option (which we abbreviated `c()` above). We typed `c(1-3,5-6)`, meaning that we want constraints 1 through 3 and 5 and 6; those numbers correspond to the constraints we defined. The only reason we did not use the number 4 was to emphasize that constraints do not have to be consecutively numbered. We typed `c(1-3,5-6)`, but we could have typed `c(1,2,3,5,6)` or `c(1-3,5,6)` or `c(1-2,3,5,6)` or even `c(1-6)`, which would have worked as long as constraint 4 was not defined. If we had previously defined a constraint 4, then `c(1-6)` would have included it.

◁

Stored results

`cnsreg` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(df_m)`	model degrees of freedom
`e(df_r)`	residual degrees of freedom
`e(F)`	*F* statistic
`e(p)`	*p*-value for model test
`e(rmse)`	root mean squared error
`e(ll)`	log likelihood
`e(N_clust)`	number of clusters
`e(rank)`	rank of `e(V)`

Macros

`e(cmd)`	`cnsreg`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(wtype)`	weight type
`e(wexp)`	weight expression
`e(title)`	title in estimation output
`e(clustvar)`	name of cluster variable
`e(vce)`	*vcetype* specified in `vce()`
`e(vcetype)`	title used to label Std. err.
`e(properties)`	`b V`

`e(predict)`	program used to implement `predict`
`e(marginsok)`	predictions allowed by `margins`
`e(asbalanced)`	factor variables `fvset` as `asbalanced`
`e(asobserved)`	factor variables `fvset` as `asobserved`
Matrices	
`e(b)`	coefficient vector
`e(Cns)`	constraints matrix
`e(V)`	variance–covariance matrix of the estimators
`e(V_modelbased)`	model-based variance
Functions	
`e(sample)`	marks estimation sample

In addition to the above, the following is stored in `r()`:

Matrices	
`r(table)`	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Let n be the number of observations, p be the total number of parameters (prior to restrictions and including the constant), and c be the number of constraints. The coefficients are calculated as $\mathbf{b}' = \mathbf{T}\{(\mathbf{T}'\mathbf{X}'\mathbf{W}\mathbf{X}\mathbf{T})^{-1}(\mathbf{T}'\mathbf{X}'\mathbf{W}\mathbf{y} - \mathbf{T}'\mathbf{X}'\mathbf{W}\mathbf{X}\mathbf{a}')\} + \mathbf{a}'$, where $\mathbf{T}$ and $\mathbf{a}$ are as defined in [P] **makecns**. $\mathbf{W} = \mathbf{I}$ if no weights are specified. If weights are specified, let $\mathbf{v}$: $1 \times n$ be the specified weights. If `fweight` frequency weights are specified, $\mathbf{W} = \text{diag}(\mathbf{v})$. If `aweight` analytic weights are specified, then $\mathbf{W} = \text{diag}[\mathbf{v}/(\mathbf{1}'\mathbf{v})(\mathbf{1}'\mathbf{1})]$, meaning that the weights are normalized to sum to the number of observations.

The mean squared error is $s^2 = (\mathbf{y}'\mathbf{W}\mathbf{y} - 2\mathbf{b}'\mathbf{X}'\mathbf{W}\mathbf{y} + \mathbf{b}'\mathbf{X}'\mathbf{W}\mathbf{X}\mathbf{b})/(n - p + c)$. The variance–covariance matrix is $s^2\mathbf{T}(\mathbf{T}'\mathbf{X}'\mathbf{W}\mathbf{X}\mathbf{T})^{-1}\mathbf{T}'$.

This command supports the Huber/White/sandwich estimator of the variance and its clustered version using `vce(robust)` and `vce(cluster` *clustvar*`)`, respectively. See [P] **_robust**, particularly *Introduction* and *Methods and formulas*.

`cnsreg` also supports estimation with survey data. For details on VCEs with survey data, see [SVY] **Variance estimation**.

References

Christodoulou, D. 2020. Stata tip 137: Interpreting constraints on slopes of rank-deficient design matrices. *Stata Journal* 20: 493–498.

Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.

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

Hill, R. C., W. E. Griffiths, and G. C. Lim. 2018. *Principles of Econometrics*. 5th ed. Hoboken, NJ: Wiley.

Also see

[R] **cnsreg postestimation** — Postestimation tools for cnsreg

[R] **regress** — Linear regression

[MI] **Estimation** — Estimation commands for use with mi estimate

[SVY] **svy estimation** — Estimation commands for survey data

[U] 20 Estimation and postestimation commands

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