**Description**

cnsreg fits constrained linear regression models.

**Quick start**

Linear regression with coefficients for \(x_1\) and \(x_2\) constrained to equality

\[
\text{constraint 1 } x_1 = x_2 \\
cnsreg y x_1 x_2 x_3, \text{ constraints(1)}
\]

Add constraint \(x_2 = x_3\) to impose \(x_1 = x_2 = x_3\)

\[
\text{constraint 2 } x_2 = x_3 \\
cnsreg y x_1 x_2 x_3, \text{ constraints(1 2)}
\]

Constrain the coefficient for \(x_4\) to be \(-1\)

\[
\text{constraint 3 } x_4 = -1 \\
cnsreg y x_1 x_2 x_3 x_4, \text{ constraints(1-3)}
\]

**Menu**

Statistics > Linear models and related > Constrained linear regression
Syntax

\texttt{cnsreg depvar indepvars [if] [in] [weight], constraints(\textit{constraints}) \ [options]} \\

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{constraints(\textit{constraints})}</td>
<td>apply specified linear constraints</td>
</tr>
<tr>
<td>\texttt{collinear}</td>
<td>keep collinear variables</td>
</tr>
<tr>
<td>\texttt{noconstant}</td>
<td>suppress constant term</td>
</tr>
</tbody>
</table>

\texttt{SE/Robust}

\texttt{vce(vcetype)} \hspace{1cm} \texttt{vcetype may be ols, robust, cluster clustvar, bootstrap, or jackknife}

\texttt{Reporting}

\texttt{level(\#)} \hspace{1cm} set confidence level; default is \texttt{level(95)}

\texttt{nocnsreport} \hspace{1cm} do not display constraints

\texttt{display_options} \hspace{1cm} control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling

\texttt{mse1} \hspace{1cm} force MSE to be 1

\texttt{coeflegend} \hspace{1cm} display legend instead of statistics

\*\texttt{constraints(\textit{constraints})} is required.

\textit{indepvars} may contain factor variables; see \texttt{[U] 11.4.3 Factor variables}.

\textit{depvar} and \textit{indepvars} may contain time-series operators; see \texttt{[U] 11.4.4 Time-series varlists}.

\texttt{bootstrap}, \texttt{by}, \texttt{fp}, \texttt{jackknife}, \texttt{mi estimate}, \texttt{rolling}, \texttt{statsby}, and \texttt{svy} are allowed; see \texttt{[U] 11.1.10 Prefix commands}.

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

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

Weights are not allowed with the \texttt{bootstrap} prefix; see \texttt{[R] bootstrap}.

\texttt{aweights} are not allowed with the \texttt{jackknife} prefix; see \texttt{[R] jackknife}.

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

\texttt{aweights}, \texttt{fweights}, \texttt{iweights}, and \texttt{pweights} are allowed; see \texttt{[U] 11.1.6 weight}.

\texttt{mse1} and \texttt{coeflegend} do not appear in the dialog.

See \texttt{[U] 20 Estimation and postestimation commands} for more capabilities of estimation commands.

Options

\texttt{Model}

\texttt{constraints(\textit{constraints}), collinear, noconstant; see \texttt{[R] estimation options}}.

\texttt{SE/Robust}

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

\texttt{vce(ols)}, the default, uses the standard variance estimator for ordinary least-squares regression.
Reporting level(#); see [R] estimation options.
nocnsreport; see [R] estimation options.
display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), 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 $(X'DX)^{-1}$ (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.

doestlegend; see [R] estimation options.

Remarks and examples


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

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

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

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

and run a regression of mpg on price + weight. The estimated coefficient on the sum would be the constrained estimate of $\beta_1$ and $\beta_2$. Using cnsreg, however, is easier:

```
. use http://www.stata-press.com/data/r15/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

| Coef.  | 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:

```stata
. generate x = price + weight
. regress mpg x
```

```
Source | SS df MS Number of obs = 74
--------+---------------------------------------------
Model | 838.065767 1 838.065767 Prob > F = 0.0000
Residual | 1605.39369 72 22.2971346 Adj R-squared = 0.3430
--------+---------------------------------------------
Total | 2443.45946 73 33.4720474 Root MSE = 4.722
--------+---------------------------------------------
mpg | Coef. 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

\[ mpg = \beta_0 + \beta_1 \text{price} + \beta_2 \text{weight} + \beta_3 \text{displ} + \beta_4 \text{gear\_ratio} + \beta_5 \text{foreign} + \beta_6 \text{length} + u \]

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

```stata
. 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

|        | Coef.     | Std. Err. | t     | P>|t|  | [95% Conf. Interval] |
|--------|-----------|-----------|-------|-------|----------------------|
| mpg    |           |           |       |       |                      |
| price  | -.000923  | .0001534  | -6.02 | 0.000 | -.0012288 -.0006172  |
| weight | -.000923  | .0001534  | -6.02 | 0.000 | -.0012288 -.0006172  |
| displac| -.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
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 $b' = T\{ (T'X'WXT)^{-1} (T'X'Wy - T'X'WXa') \} + a'$, where $T$ and $a$ are as defined in [P] makecns. $W = I$ if no weights are specified. If weights are specified, let $v: 1 \times n$ be the specified weights. If $fweight$ frequency weights are specified, $W = \text{diag}(v)$. If $aweight$ analytic weights are specified, then $W = \text{diag}[v/(1'v)(1'1)]$, meaning that the weights are normalized to sum to the number of observations.

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

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


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