Syntax

Bootstrap statistics from variables

```
bstat [ `varlist` ] [ `if` ] [ `in` ] [, `options` ]
```

Bootstrap statistics from file

```
bstat [ `namelist` ] [ using `filename` ] [ `if` ] [ `in` ] [, `options` ]
```

`options` Description

<table>
<thead>
<tr>
<th>Main</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>stat(vector)</code></td>
<td>observed values for each statistic</td>
</tr>
<tr>
<td><code>accel(vector)</code></td>
<td>acceleration values for each statistic</td>
</tr>
<tr>
<td><code>ties</code></td>
<td>adjust BC/BCa confidence intervals for ties</td>
</tr>
<tr>
<td><code>mse</code></td>
<td>use MSE formula for variance estimation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reporting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>level(#)</code></td>
<td>set confidence level; default is <code>level(95)</code></td>
</tr>
<tr>
<td><code>n(#)</code></td>
<td># of observations from which bootstrap samples were taken</td>
</tr>
<tr>
<td><code>notable</code></td>
<td>suppress table of results</td>
</tr>
<tr>
<td><code>noheader</code></td>
<td>suppress table header</td>
</tr>
<tr>
<td><code>nolegend</code></td>
<td>suppress table legend</td>
</tr>
<tr>
<td><code>verbose</code></td>
<td>display the full table legend</td>
</tr>
<tr>
<td><code>title(text)</code></td>
<td>use <code>text</code> as title for bootstrap results</td>
</tr>
<tr>
<td><code>display_options</code></td>
<td>control column formats and line width</td>
</tr>
</tbody>
</table>

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

Menu

Statistics > Resampling > Report bootstrap results

Description

`bstat` is a programmer’s command that computes and displays estimation results from bootstrap statistics.

For each variable in `varlist` (the default is all variables), then `bstat` computes a covariance matrix, estimates bias, and constructs several different confidence intervals (CIs). The following CIs are constructed by `bstat`:
1. Normal CIs (using the normal approximation)
2. Percentile CIs
3. Bias-corrected (BC) CIs
4. Bias-corrected and accelerated (BCa) CIs (optional)

`estat bootstrap` displays a table of one or more of the above confidence intervals; see [R] bootstrap postestimation.

If there are bootstrap estimation results in `e()`, `bstat` replays them. If given the `using` modifier, `bstat` uses the data in `filename` to compute the bootstrap statistics while preserving the data currently in memory. Otherwise, `bstat` uses the data in memory to compute the bootstrap statistics.

The following options may be used to replay estimation results from `bstat`:

```plaintext
level(#) notable noheader nolegend verbose title(text)
```

For all other options and the qualifiers `using`, `if`, and `in`, `bstat` requires a bootstrap dataset.

Options

**Main**

- `stat(vector)` specifies the observed value of each statistic (that is, the value of the statistic using the original dataset).
- `accel(vector)` specifies the acceleration of each statistic, which is used to construct BCa CIs.
- `ties` specifies that `bstat` adjust for ties in the replicate values when computing the median bias used to construct BC and BCa CIs.
- `mse` specifies that `bstat` compute the variance by using deviations of the replicates from the observed value of the statistics. By default, `bstat` computes the variance by using deviations from the average of the replicates.

**Reporting**

- `level(#); see [R] estimation options.`
- `n(#)` specifies the number of observations from which bootstrap samples were taken. This value is used in no calculations but improves the table header when this information is not saved in the bootstrap dataset.
- `notable` suppresses the display of the output table.
- `noheader` suppresses the display of the table header. This option implies `nolegend`.
- `nolegend` suppresses the display of the table legend.
- `verbose` specifies that the full table legend be displayed. By default, coefficients and standard errors are not displayed.
- `title(text)` specifies a title to be displayed above the table of bootstrap results; the default title is Bootstrap results.

*display_options: cformat(%fmt), pformat(%fmt), sformat(%fmt), and nolstretch; see [R] estimation options.*
Remarks and examples

Remarks are presented under the following headings:

Bootstrap datasets
Creating a bootstrap dataset

Bootstrap datasets

Although `bstat` allows you to specify the observed value and acceleration of each bootstrap statistic via the `stat()` and `accel()` options, programmers may be interested in what `bstat` uses when these options are not supplied.

When working from a bootstrap dataset, `bstat` first checks the data characteristics (see `[P] char`) that it understands:

- `_dta[bs_version]` identifies the version of the bootstrap dataset. This characteristic may be empty (not defined), 2, or 3; otherwise, `bstat` will quit and display an error message. This version tells `bstat` which other characteristics to look for in the bootstrap dataset.

`bstat` uses the following characteristics from version 3 bootstrap datasets:

- `_dta[N]`
- `_dta[N_strata]`
- `_dta[N_cluster]`
- `_dta[command]`
- `varname[observed]`
- `varname[acceleration]`
- `varname[expression]`

`bstat` uses the following characteristics from version 2 bootstrap datasets:

- `_dta[N]`
- `_dta[N_strata]`
- `_dta[N_cluster]`
- `varname[observed]`
- `varname[acceleration]`

An empty bootstrap dataset version implies that the dataset was created by the `bstrap` command in a version of Stata earlier than Stata 8. Here `bstat` expects `varname[bstrap]` to contain the observed value of the statistic identified by `varname` (`varname[observed]` in version 2). All other characteristics are ignored.

- `_dta[N]` is the number of observations in the observed dataset. This characteristic may be overruled by specifying the `n()` option.
- `_dta[N_strata]` is the number of strata in the observed dataset.
- `_dta[N_cluster]` is the number of clusters in the observed dataset.
- `_dta[command]` is the command used to compute the observed values of the statistics.
- `varname[observed]` is the observed value of the statistic identified by `varname`. To specify a different value, use the `stat()` option.
- `varname[acceleration]` is the estimate of acceleration for the statistic identified by `varname`. To specify a different value, use the `accel()` option.
- `varname[expression]` is the expression or label that describes the statistic identified by `varname`. 
Creating a bootstrap dataset

Suppose that we are interested in obtaining bootstrap statistics by resampling the residuals from a regression (which is not possible with the bootstrap command). After loading some data, we run a regression, save some results relevant to the bstat command, and save the residuals in a new variable, res.

```
use http://www.stata-press.com/data/r13/auto
(1978 Automobile Data)
regress mpg weight length
```

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs = 74</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F( 2, 71) = 69.34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>1616.08062</td>
<td>2</td>
<td>808.040312</td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>827.378835</td>
<td>71</td>
<td>11.653223</td>
<td>Adj R-squared = 0.6614</td>
</tr>
<tr>
<td>Total</td>
<td>2443.45946</td>
<td>73</td>
<td>33.4720474</td>
<td>Root MSE = 3.4137</td>
</tr>
</tbody>
</table>

| mpg     | Coef. | Std. Err. | t     | P>|t| | [95% Conf. Interval] |
|---------|-------|-----------|-------|-----|------------------|
| weight  | -.0038515 | .001586 | -2.43 | 0.018 | -.0070138 -.0006891 |
| length  | -.0795935 | .0553577 | -1.44 | 0.155 | -.1899736 .0307867 |
| _cons   | 47.88487  | 6.08787  | 7.87  | 0.000 | 35.746 60.02374 |

```
matrix b = e(b)
local n = e(N)
predict res, residuals
```

We can resample the residual values in res by generating a random observation ID (rid), generate a new response variable (y), and run the original regression with the new response variables.

```
set seed 54321
generate rid = int(_N*runiform())+1
matrix score double y = b
replace y = y + res[rid]
```

(74 real changes made)

```
regress y weight length
```

<table>
<thead>
<tr>
<th>Source</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>Number of obs = 74</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F( 2, 71) = 103.41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>1773.23548</td>
<td>2</td>
<td>886.617741</td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>608.747322</td>
<td>71</td>
<td>8.57391172</td>
<td>Adj R-squared = 0.7372</td>
</tr>
<tr>
<td>Total</td>
<td>2381.98321</td>
<td>73</td>
<td>32.6299074</td>
<td>Root MSE = 2.9281</td>
</tr>
</tbody>
</table>

| y       | Coef.  | Std. Err. | t     | P>|t| | [95% Conf. Interval] |
|---------|--------|-----------|-------|-----|------------------|
| weight  | -.0059938 | .0013604 | -4.41 | 0.000 | -.0087064 -.0032813 |
| length  | -.0127875 | .0474837 | -0.27 | 0.788 | -.1074673 .0818924 |
| _cons   | 42.23195  | 5.22194  | 8.09  | 0.000 | 31.8197 52.6442 |

Instead of programming this resampling inside a loop, it is much more convenient to write a short program and use the simulate command; see [R] simulate. In the following, mysim_r requires the user to specify a coefficient vector and a residual variable. mysim_r then retrieves the list of predictor variables (removing _cons from the list), generates a new temporary response variable with the resampled residuals, and regresses the new response variable on the predictors.
# bstat — Report bootstrap results

**program** `{mysim_r}`

**syntax** `name(name=bvector), res(varname)`

**tempvar** `y` `rid`

**local** `xvars` : `colnames ‘bvector’`

**local** `cons` `_cons`

**local xvars** : `list xvars - cons`

**matrix** `score double ‘y’ = ‘bvector’`

**gen long ‘rid’ = int(_N*runiform()) + 1**

**replace ‘y’ = ‘y’ + ‘res’['rid']**

**regress ‘y’ ‘xvars’**

**end**

We can now give `{mysim_r}` a test run, but we first set the random-number seed (to reproduce results).

```
. set seed 54321
. mysim_r b, res(res)
(74 real changes made)
```

```
Source | SS   | df   | MS    | Number of obs = 74  
Model  | 1773.23548 | 2  | 886.617741 | F(  2,  71) = 103.41
Residual | 608.747732 | 71  | 8.57391172 | Prob > F = 0.0000
Total | 2381.98321 | 73  | 32.629907 | R-squared = 0.7372

|     | Coef.  | Std. Err. | t    | P>|t| | [95% Conf. Interval] |
|-----|--------|-----------|------|-----|---------------------|
| weight | -.0059938 | .0013604 | -4.41 | 0.000 | -.0087064 -.0032813 |
| length | -.0127875 | .0474837 | -0.27 | 0.788 | -.1074673 .0818924 |
| _cons  | 42.23195 | 5.22194 | 8.09 | 0.000 | 31.8197 52.6442 |
```

Now that we have a program that will compute the results we want, we can use `simulate` to generate a bootstrap dataset and `bstat` to display the results.

```
. set seed 54321
. simulate, reps(200) nodots: mysim_r b, res(res)
```

```
command: mysim_r b, res(res)
.bstat, stat(b) n('n')
```

```
Bootstrap results Number of obs = 74  
Replications = 200  

|     | Coef.  | Std. Err. | z    | P>|z| | [95% Conf. Interval] |
|-----|--------|-----------|------|-----|---------------------|
| _b_weight | -.0038515 | .0015715 | -2.45 | 0.014 | -.0069316 -.0007713 |
| _b_length | -.0795935 | .0552415 | -1.44 | 0.150 | -.1878649 .0286779 |
| _b_cons  | 47.88487 | 6.150069 | 7.79 | 0.000 | 35.83096 59.93879 |
```

Finally, we see that `simulate` created some of the data characteristics recognized by `bstat`. All we need to do is correctly specify the version of the bootstrap dataset, and `bstat` will automatically use the relevant data characteristics.
. char list
  _dta[seed]: X681014b5c43f462544a474abacbd93d00042842
  _dta[command]: mysim_r b, res(res)
  _b_weight[is_eexp]: 1
  _b_weight[colname]: weight
  _b_weight[coleq]: _
  _b_weight[expression]: _b[weight]
  _b_length[is_eexp]: 1
  _b_length[colname]: length
  _b_length[coleq]: _
  _b_length[expression]: _b[length]
  _b_cons[is_eexp]: 1
  _b_cons[colname]: _cons
  _b_cons[coleq]: _
  _b_cons[expression]: _b[_cons]
.
. char _dta[bs_version] 3
.
. bstat, stat(b) n('n')

Bootstrap results

Number of obs = 74
Replications = 200

command: mysim_r b, res(res)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>-.0038515</td>
<td>.0015715</td>
<td>-.0038515</td>
<td>.0015715</td>
<td>-2.45</td>
<td>0.014</td>
<td>-.0069316</td>
<td>-.0007713</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>length</td>
<td>-.0795935</td>
<td>.0552415</td>
<td>-.0795935</td>
<td>.0552415</td>
<td>-1.44</td>
<td>0.150</td>
<td>-.1878649</td>
<td>.0286779</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_cons</td>
<td>47.88487</td>
<td>6.150069</td>
<td>47.88487</td>
<td>6.150069</td>
<td>7.79</td>
<td>0.000</td>
<td>35.83096</td>
<td>59.93879</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See Poi (2004) for another example of residual resampling.
**Stored results**

*bstat* stores the following in *e()*:

**Scalars**

- `e(N)` sample size
- `e(N_reps)` number of complete replications
- `e(N_misreps)` number of incomplete replications
- `e(N_strata)` number of strata
- `e(N_clust)` number of clusters
- `e(k_aux)` number of auxiliary parameters
- `e(k_eq)` number of equations in *e(b)*
- `e(k_exp)` number of standard expressions
- `e(k_eexp)` number of extended expressions (i.e., `._b`) beyond the original ones from *e(b)*
- `e(level)` confidence level for bootstrap CIs
- `e(bs_version)` version for bootstrap results
- `e(rank)` rank of *e(V)*

**Macros**

- `e(cmd)` *bstat*
- `e(command)` from _dta[command]*
- `e(cmdline)` command as typed
- `e(title)` title in estimation output
- `e(exp#)` expression for the #th statistic
- `e(prefix)` bootstrap
- `e(ties)` ties, if specified
- `e(mse)` mse, if specified
- `e(vce)` bootstrap
- `e(vcetype)` title used to label Std. Err.
- `e(properties)` b V

**Matrices**

- `e(b)` observed statistics
- `e(b_bs)` bootstrap estimates
- `e(reps)` number of nonmissing results
- `e(bias)` estimated biases
- `e(se)` estimated standard errors
- `e(z0)` median biases
- `e(accel)` estimated accelerations
- `e(ci_normal)` normal-approximation CIs
- `e(ci_percentile)` percentile CIs
- `e(ci_bc)` bias-corrected CIs
- `e(ci_bca)` bias-corrected and accelerated CIs
- `e(V)` bootstrap variance–covariance matrix

**References**


**Also see**

- [R] bootstrap postestimation — Postestimation tools for bootstrap
- [R] bootstrap — Bootstrap sampling and estimation
- [R] bsample — Sampling with replacement