Title

stata.com

bstat — Report bootstrap results

Description

`bstat` is a programmer's command that computes and displays estimation results from bootstrap statistics. For each variable in *varlist*, `bstat` computes a covariance matrix, estimates bias, and constructs normal confidence intervals (CIs), percentile CIs, bias-corrected (BC) CIs, and bias-corrected and accelerated (BC_a) CIs using a bootstrap dataset in memory or on disk. The computed CIs can be displayed using `estat bootstrap`; see [R] **bootstrap postestimation**.

`bstat` without *varlist* replays results from the last bootstrap estimation when results are stored in `e()`.

Menu

Statistics > Resampling > Report bootstrap results

Syntax

Bootstrap statistics from variables

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

Bootstrap statistics from file

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

options	Description
Main	
* `stat(`*vector*`)`	observed values for each statistic
* `accel(`*vector*`)`	acceleration values for each statistic
* `ties`	adjust BC/BCa confidence intervals for ties
* `mse`	use MSE formula for variance estimation
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`n(#)`	# of observations from which bootstrap samples were taken
`notable`	suppress table of results
`noheader`	suppress table header
`nolegend`	suppress table legend
`verbose`	display the full table legend
`title(`*text*`)`	use *text* as title for bootstrap results
display_options	control column formats and line width

*Starred options and qualifiers `using`, `if`, and `in` require a bootstrap dataset.

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

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 BC$_a$ 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

stata.com

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/r15/auto
(1978 Automobile Data)
. regress mpg weight length
```

Source	SS	df	MS	Number of obs	=	74
				F(2, 71)	=	69.34
Model	1616.08062	2	808.040312	Prob > F	=	0.0000
Residual	827.378835	71	11.653223	R-squared	=	0.6614
				Adj R-squared	=	0.6519
Total	2443.45946	73	33.4720474	Root MSE	=	3.4137

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
```

Source	SS	df	MS	Number of obs	=	74
				F(2, 71)	=	100.11
Model	1695.70314	2	847.851568	Prob > F	=	0.0000
Residual	601.341031	71	8.46959199	R-squared	=	0.7382
				Adj R-squared	=	0.7308
Total	2297.04417	73	31.4663585	Root MSE	=	2.9103

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
weight	-.0029676	.0013521	-2.19	0.031	-.0056636	-.0002716
length	-.1158425	.047194	-2.45	0.017	-.2099446	-.0217404
_cons	51.72451	5.190075	9.97	0.000	41.3758	62.07323

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.

```
program mysim_r
        version 15.0
        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'
        generate 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
				F(2, 71)	=	100.11
Model	1695.70314	2	847.851568	Prob > F	=	0.0000
Residual	601.341031	71	8.46959199	R-squared	=	0.7382
				Adj R-squared	=	0.7308
Total	2297.04417	73	31.4663585	Root MSE	=	2.9103

__000000	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
weight	-.0029676	.0013521	-2.19	0.031	-.0056636	-.0002716
length	-.1158425	.047194	-2.45	0.017	-.2099446	-.0217404
_cons	51.72451	5.190075	9.97	0.000	41.3758	62.07323

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

	Observed Coef.	Bootstrap Std. Err.	z	P>|z|	Normal-based [95% Conf.	Interval]
_b_weight	-.0038515	.0014673	-2.62	0.009	-.0067274	-.0009756
_b_length	-.0795935	.0509772	-1.56	0.118	-.1795069	.0203199
_b_cons	47.88487	5.650947	8.47	0.000	36.80922	58.96053

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[rngstate]:             XAA000000000000d431c5e5401775ee9b9e24b2604d4885..
  _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)
```

	Observed Coef.	Bootstrap Std. Err.	z	P>|z|	Normal-based [95% Conf.	Interval]
weight	-.0038515	.0014673	-2.62	0.009	-.0067274	-.0009756
length	-.0795935	.0509772	-1.56	0.118	-.1795069	.0203199
_cons	47.88487	5.650947	8.47	0.000	36.80922	58.96053

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`)
`e(k_extra)`	number of extra equations 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

Ng, E. S.-W., R. Grieve, and J. R. Carpenter. 2013. Two-stage nonparametric bootstrap sampling with shrinkage correction for clustered data. *Stata Journal* 13: 141–164.

Poi, B. P. 2004. From the help desk: Some bootstrapping techniques. *Stata Journal* 4: 312–328.

Also see

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

[R] **bootstrap** — Bootstrap sampling and estimation

[R] **bsample** — Sampling with replacement