bayesboot — Bayesian bootstrap estimation

Description

The `bayesboot` prefix performs Bayesian bootstrap estimation of specified statistics or expressions for Stata commands or user-written programs. It functions as a wrapper for two consecutive steps: First, `bayesboot` issues the `rwgen bayes` command to generate importance weights for each replication. Second, it issues the `bootstrap` prefix with the `iweights()` option to compute Bayesian bootstrap estimates.

Quick start

Compute a Bayesian bootstrap estimate of the mean of `v1` returned by `summarize` in `r(mean)`

```
bayesboot mean=r(mean): summarize v1
```

Same as above, but use the values of variable `x1` as the prior powers for each observation

```
bayesboot mean=r(mean), priorpowers(x1): summarize v1
```

Same as above, but use values of both `x2` and `x3` to uniquely identify observations

```
bayesboot mean=r(mean), priorpowers(x1) id(x2 x3): summarize v1
```

Compute a Bayesian bootstrap estimate of the statistic `r(mystat)` returned by program `myprog1`

```
bayesboot stat=r(mystat): myprog1 v1
```

Same as above, but use 100 replications

```
bayesboot stat=r(mystat), reps(100): myprog1 v1
```

Same as above, and generate new variables `iw1` through `iw100` to store importance weights

```
bayesboot stat=r(mystat), reps(100) generate(iw): myprog1 v1
```

Compute Bayesian bootstrap estimates of the coefficients stored in `e(b)` by `myprog2`

```
bayesboot _b: myprog2 y x1 x2 x3
```

Menu

Statistics > Resampling > Bayesian bootstrap estimation

Syntax

`bayesboot` *exp_list* [, *options*] : *command*

options	Description
Options	
`priorpowers(#` | *varname*`)`	specify prior power for each observation; default is `priorpowers(-1)`
`id(`*varlist*`)`	specify variables that uniquely identify observations
`generate(`*stub*`)`	generate new variables to store the importance weights
bootstrap_options	options allowed by `bootstrap`

command is any command that follows standard Stata syntax. *command* syntax must support `iweights`.

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

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

exp_list contains	`(`*name*`:` *elist*`)`
	elist
	eexp
elist contains	*newvar* `=` `(`*exp*`)`
	`(`*exp*`)`
eexp is	*specname*
	`[`*eqno*`]`*specname*
specname is	`_b`
	`_b[]`
	`_se`
	`_se[]`
eqno is	`##`
	name

exp is a standard Stata expression; see **[U] 13 Functions and expressions**.

Distinguish between `[]`, which are to be typed, and [ ], which indicate optional arguments.

Options

Options

`priorpowers(#` | *varname*`)` specifies the power value l_i of the prior distribution for the ith observation, $i = 1, 2, \ldots, n$. The default is `priorpowers(-1)`, which corresponds to an improper noninformative prior. `priorpowers(#)` specifies the same power # for all observations. Alternatively, you can specify different power values in *varname*. See *Methods and formulas* in **[R] rwgen**. The variable specified in `priorpowers()` may not be specified in `id()`.

`id(`*varlist*`)` specifies the variables used to uniquely identify each observation. By default, `bayesboot` considers all variables in the dataset to assess uniqueness. Any variables specified in `id()` may not be specified in `priorpowers()`.

`generate(`*stub*`)` requests that new variables be generated holding the Bayesian bootstrap weights. For # replications, # new variables *stub*1, *stub*2, …, *stub*# will be created. The importance weights are not stored by default.

bootstrap_options: reps(#), saving(*filename*[, *suboptions*]), bca, ties, mse, level(#), notable, noheader, nolegend, verbose, nodots, dots(#), noisily, trace, title(*text*), *display_options*, *eform_option*, nodrop, nowarn, force, reject(*exp*), rseed(#), jackknifeopts(*jkopts*), and coeflegend; see [R] **bootstrap**.

Remarks and examples

The bayesboot prefix performs Bayesian bootstrap estimation of specified statistics or expressions for Stata commands or user-written programs. It implements the technique introduced by Rubin (1981), which modifies the traditional bootstrap technique by adopting a Bayesian approach. Instead of discrete resampling with replacement, it assigns probabilities to each observation, forming an n-vector of weights that follow a specified posterior distribution. This method captures the uncertainty about the representativeness of each observation in the overall distribution.

bayesboot is a wrapper for a two-step process: First, it generates replicate weights, and second, it performs Bayesian bootstrap estimation with those weights. Alternatively, you can generate the replicate weights yourself with the rwgen bayes command and then specify those weight variables in the iweights() option with the bootstrap prefix to perform bootstrap estimation. However, bayesboot consolidates this all into one step. Alternatively, if you already have replicate weights, you can skip the step of generating them with rwgen bayes and simply use bootstrap to perform estimation.

All postestimation tools for bootstrap are supported after bayesboot.

▷ Example 1: Bayesian bootstrap estimation of standard errors

Suppose we want to compute Bayesian bootstrap estimates for the standard errors of the coefficients from the following regression:

```
. use https://www.stata-press.com/data/r19/auto
(1978 automobile data)
. regress price mpg weight length
  (output omitted)
```

To compute the Bayesian bootstrap estimates, we add the `bayesboot` prefix to the above regression command. We specify the random-number seed for reproducibility and request 100 replications. We use the default prior for each observation. This way, if all observations are unique, the posterior distribution of weights will follow a Dirichlet distribution, Dirichlet(1,1,...,1). This implies that each data point has the same posterior probability. This setup is comparable with the classical bootstrap, where each observation has a $1/n$ probability of selection in any given draw.

```
. bayesboot, rseed(19) reps(100): regress price mpg weight length
(running regress on estimation sample)
Bayesian bootstrap replications (100): .........10.........20.........30.......
> ..40.........50.........60.........70.........80.........90.........100 done
Bayesian bootstrap
Observation prior: Improper
Linear regression                                    Number of obs =        74
                                                     Replications  =       100
                                                     Wald chi2(3)  =     35.55
                                                     Prob > chi2   =    0.0000
                                                     R-squared     =    0.3574
                                                     Adj R-squared =    0.3298
                                                     Root MSE      = 2414.5629
```

price	Observed coefficient	Bayesian bootstrap std. err.	z	P>|z|	Normal-based [95% conf.	interval]
mpg	-86.78928	75.87845	-1.14	0.253	-235.5083	61.92975
weight	4.364798	1.618642	2.70	0.007	1.192318	7.537277
length	-104.8682	52.87436	-1.98	0.047	-208.5	-1.236331
_cons	14542.43	6475.149	2.25	0.025	1851.375	27233.49

First, `bayesboot` executes `rwgen bayes` to generate 100 importance weights. Then, it executes `bootstrap`; for each replication, it computes a weighted estimate of the parameters, and then it computes the standard deviation of those weighted estimates to report the standard error.

As with the `bootstrap` prefix, the point estimates are not affected when using the `bayesboot` prefix. ◁

▷ Example 2: Specifying the power values of the prior distribution

By default, `bayesboot` uses an improper noninformative prior, but we can use the `priorpowers()` option to customize the power values of the prior distribution. Below, we generate a new variable, `priorparm`, based on the values of `trunk` and use the resulting values as the power values. This allows us to account for the varying levels of confidence in our observations when performing our analysis. The prior parameters in `priorparm` control the shape of the posterior Dirichlet distribution that samples the importance weights, where higher values indicate stronger prior beliefs about the data.

```
. generate double priorparm = trunk / 10

. bayesboot, rseed(19) priorpowers(priorparm): regress price mpg weight length
(running regress on estimation sample)

Bayesian bootstrap replications (50): .........10.........20.........30........
> .40.........50 done

Bayesian bootstrap
Observation prior: priorparm

Linear regression                                       Number of obs =         74
                                                        Replications  =         50
                                                        Wald chi2(3)  =     157.33
                                                        Prob > chi2   =     0.0000
                                                        R-squared     =     0.3574
                                                        Adj R-squared =     0.3298
                                                        Root MSE      = 2414.5629
```

price	Observed coefficient	Bayesian bootstrap std. err.	z	P>|z|	Normal-based [95% conf.	interval]
mpg	-86.78928	37.19352	-2.33	0.020	-159.6873	-13.89132
weight	4.364798	.9316533	4.69	0.000	2.538791	6.190805
length	-104.8682	30.15013	-3.48	0.001	-163.9613	-45.77501
_cons	14542.43	3401.082	4.28	0.000	7876.436	21208.43

For more information about the Bayesian bootstrap technique and how weights are generated, see *Methods and formulas* in [R] **rwgen**.

◁

▷ Example 3: Determining uniqueness of observations

By default, all variables in the dataset are used to uniquely identify observations, but we can use the `id()` option to specify which variables we want to use.

Suppose we want to compute Bayesian bootstrap estimates for the standard errors of the mean of `price`, and we want cars from the same car brand (for example, Buick, Cadillac, Dodge) to use the same prior parameters. To begin, we generate a new variable, `brand1`, to store the car brand because the `make` variable stores the make and model.

```
. split make, generate(brand) limit(1)
variable created as string:
brand1
. list make brand1 in 1/5

     +--------------------------+
     | make              brand1 |
     |--------------------------|
  1. | AMC Concord          AMC |
  2. | AMC Pacer            AMC |
  3. | AMC Spirit           AMC |
  4. | Buick Century      Buick |
  5. | Buick Electra      Buick |
     +--------------------------+
```

Now we use the `id(brand1)` option with `bayesboot` to specify that we only want to use the values of `brand1` to identify unique observations. `bayesboot` will execute `rwgen bayes` and create temporary variables to store the replicate weights, but we use the `generate()` option to store these variables for later use.

```
. bayesboot _b, rseed(19) id(brand1) generate(iw): mean price
(running mean on estimation sample)
Bayesian bootstrap replications (50): .........10.........20.........30........
> .40.........50 done
Bayesian bootstrap
Observation prior: Improper
Mean estimation                         Number of obs = 74
                                        Replications  = 50

--------------------------------------------------------------
             |              Bayesian
             |  Observed   bootstrap          Normal-based
             |      mean   std. err.     [95% conf. interval]
-------------+------------------------------------------------
       price |  6165.257   281.0588      5614.392    6716.122
--------------------------------------------------------------
Uniqueness ID: brand1
```

We specified the stub `iw`, which means that we now have variables named `iw1`, `iw2`, etc., in our data. Let's examine the replicate weights for all the Buicks in our data:

```
. list brand1 iw1-iw5 if brand1 == "Buick", separator(0)
```

	brand1	iw1	iw2	iw3	iw4	iw5
4.	Buick	.01236759	.00713592	.01340483	.0216436	.01589735
5.	Buick	.01236759	.00713592	.01340483	.0216436	.01589735
6.	Buick	.01236759	.00713592	.01340483	.0216436	.01589735
7.	Buick	.01236759	.00713592	.01340483	.0216436	.01589735
8.	Buick	.01236759	.00713592	.01340483	.0216436	.01589735
9.	Buick	.01236759	.00713592	.01340483	.0216436	.01589735
10.	Buick	.01236759	.00713592	.01340483	.0216436	.01589735

Note that cars from the same car brand have identical importance weights, as defined by our uniqueness criteria.

Because we stored the replicate weights, we can use them going forward with `bootstrap()`, allowing us to reproduce our analysis without having to re-create the weights or specify the random-number seed. For example, you can use the command below to reproduce the results from `bayesboot` above:

```
. bootstrap, iweights(iw1-iw50): mean price
```

◁

Stored results

`bayesboot` 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(k_eq)`	number of equations in `e(b)`
`e(k_exp)`	number of standard expressions
`e(k_eexp)`	number of extended expressions (that is, `_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(cmdname)`	command name from *command*
`e(cmd)`	same as `e(cmdname)` or `bootstrap`
`e(command)`	*command*
`e(cmdline)`	command as typed
`e(prefix)`	`bayesboot`
`e(title)`	title in estimation output
`e(rngstate)`	random-number state used
`e(exp#)`	expression for the #th statistic
`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`
`e(rwb_newvarlist)`	generated weight variables
`e(rwb_method)`	`bayes`
`e(rwb_stub)`	*stub* of new variables

e(rwb_priorpowers)	prior powers
e(rwb_id)	variables for determining uniqueness
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
e(V_modelbased)	model-based variance
Functions	
e(sample)	marks estimation sample

`bayesboot` will carry forward most of the results already in `e()` from `bootstrap`; when *exp_list* is `_b`, `bayesboot` will also carry forward most of the results already in `e()` from *command*.

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

The `bayesboot` prefix performs Bayesian bootstrap estimation by combining the `rwgen bayes` command and the `bootstrap` prefix with the `iweights()` option.

The process consists of two sequential steps. First, `bayesboot` executes the `rwgen bayes` command to generate importance weights for each replication. These weights are temporarily stored as variables and automatically removed after execution. You can store these weights by specifying the `generate()` option. For detailed information about the weight-generation process, see *Methods and formulas* in [R] **rwgen**.

Second, the command executes the `bootstrap` prefix with the importance weights specified in the `iweights()` option to compute and summarize the final results.

`bayesboot` was primarily developed for nonestimation commands, but it also supports estimation commands. When an estimation command is specified, `bayesboot` first executes the routine to obtain `e(sample)` and then uses this sample to filter observations for subsequent `rwgen bayes` and `bootstrap, iweights()` commands.

Reference

Rubin, D. B. 1981. The Bayesian bootstrap. *Annals of Statistics* 9: 130–134. https://doi.org/10.1214/aos/1176345338.

Also see

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

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

[R] **jackknife** — Jackknife estimation

[R] **rwgen** — Generate replicate weights for bootstrap estimation

[U] 20 Estimation and postestimation commands

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