rwgen — Generate replicate weights for bootstrap estimation

Description | Quick start | Menu | Syntax
Options | Remarks and examples | Stored results | Methods and formulas
References | Also see

Description

`rwgen bsample` and `rwgen bayes` generate replicate weights for each observation that can be used by the `bootstrap` prefix to enhance reproducibility during bootstrap estimation.

`rwgen bsample` executes the `bsample` command on the original dataset one or more times, saving the generated frequency weights in newly created variables. These frequency weights can be specified in the `fweights()` option of the `bootstrap` prefix.

`rwgen bayes` generates weights using Bayesian bootstrap methods. These weights can be specified in the `iweights()` option of the `bootstrap` prefix.

Quick start

Generate bootstrap frequency weights for the default 50 replications, and store them in new variables `stubvar1` through `stubvar50`

```
rwgen bsample stubvar
```

Same as above, but sample only five observations for each replication

```
rwgen bsample stubvar, size(5)
```

Generate bootstrap frequency weights for each stratum defined by `x1`

```
rwgen bsample stubvar, strata(x1)
```

Generate bootstrap frequency weights by sampling clusters identified by values of `cvar`

```
rwgen bsample stubvar, cluster(cvar)
```

Same as above, but create a new unique identifier for sampled clusters, and store it in `newcvar`

```
rwgen bsample stubvar, cluster(cvar) idcluster(newcvar)
```

Generate Bayesian bootstrap weights using default prior powers for 100 replications

```
rwgen bayes stubvar, reps(100)
```

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

```
rwgen bayes stubvar, reps(100) priorpowers(x1)
```

Same as above but use a combination of `x2` and `x3` to uniquely identify the observations

```
rwgen bayes stubvar, reps(100) priorpowers(x1) id(x2 x3)
```

Menu

Statistics > Resampling > Generate bootstrap replicate weights

Syntax

Generate replicate weights using random sampling with replacement

```
rwgen bsample stub [if] [in] [, bsample_options]
```

Generate replicate weights using Bayesian bootstrap methods

```
rwgen bayes stub [if] [in] [, bayes_options]
```

bsample_options	Description
Main	
reps(#)	perform # bootstrap replications; default is reps(50)
size(*exp*)	draw samples of size *exp*, where *exp* is a standard Stata expression; default is _N
strata(*varlist*)	specify variables identifying strata
cluster(*varlist*)	specify variables identifying resampling clusters
idcluster(*newvar*)	create new cluster ID variable
rseed(#)	set random-number seed to #

bayes_options	Description
Main	
reps(#)	perform # bootstrap replications; default is reps(50)
priorpowers(# | *varname*)	specify prior powers for each observation; default is priorpowers(-1)
id(*varlist*)	specify variables that uniquely identify observations
rseed(#)	set random-number seed to #

Options

Options are presented under the following headings:

Options for rwgen bsample
Options for rwgen bayes

Options for rwgen bsample

Main

reps(#) specifies the number of replicate weight variables to generate; the default is reps(50). The generated weights are stored in new variables named *stub*1, *stub*2, ..., *stub*#.

size(*exp*) specifies the size of the sample and must be less than or equal to _N when neither the cluster() nor the strata() option is specified; default is _N.

strata(*varlist*) specifies the variables that identify strata. If strata() is specified, bootstrap samples are selected within each stratum, and size() must be less than or equal to _N within the defined strata.

`cluster(`*varlist*`)` specifies the variables identifying resampling clusters. If `cluster()` is specified, the sample drawn during each replication is a bootstrap sample of clusters, and `size()` must be less than or equal to `N_c` (the number of clusters identified by `cluster()`). If `strata()` is also specified, `size()` must be less than or equal to the number of within-strata clusters.

`idcluster(`*newvar*`)` creates a new variable containing a unique identifier for each resampled cluster.

`rseed(`#`)` sets the random-number seed. This option can be used to reproduce results. `rseed(`#`)` is equivalent to typing `set seed` # prior to calling `rwgen`; see [R] **set seed**.

Options for rwgen bayes

Main

`reps(`#`)` specifies the number of replicate weight variables to generate; the default is `reps(50)`. The generated weights are stored in new variables named *stub*`1`, *stub*`2`, …, *stub*`#`.

`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*. The variable specified in `priorpowers()` may not be specified in `id()`.

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

`rseed(`#`)` sets the random-number seed. This option can be used to reproduce results. `rseed(`#`)` is equivalent to typing `set seed` # prior to calling `rwgen`; see [R] **set seed**.

Remarks and examples

The `rwgen bsample` and `rwgen bayes` commands generate replicate weights that can be used by the `bootstrap` prefix to enhance reproducibility during bootstrap estimation.

`rwgen bsample` executes the `bsample` command on the original dataset one or more times to generate replicate weights. You can randomly resample observations, observations within strata, or clusters. The frequency weights generated by `rwgen bsample` can then be specified in the `fweights()` option of the `bootstrap` prefix.

`rwgen bayes` generates weights using Bayesian bootstrap methods. These weights can be specified in the `iweights()` option of the `bootstrap` prefix.

▷ Example 1: Generate replicate weights for bootstrap

We wish to compute bootstrap estimates for the standard errors of the coefficients from a regression of car price on car weight and origin. We first load `auto.dta`:

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

We use `rwgen bsample` to draw random samples of the data, with replacement, and specify that we want 100 replications. For reproducibility, we set the seed using the `rseed()` option. `rwgen bsample` will store the frequency weights in newly generated variables using the stub we specify, `freq_`.

```
. rwgen bsample freq_, reps(100) rseed(20)
```

After we run the command, our dataset now contains 100 newly generated variables named freq_1, freq_2, …, freq_100. Below, we list the first 10 observations of freq_1 through freq_5:

```
. list freq_1-freq_5 in 1/10
```

	freq_1	freq_2	freq_3	freq_4	freq_5
1.	2	0	1	0	0
2.	1	2	2	0	1
3.	2	2	2	0	3
4.	0	0	1	0	2
5.	0	1	1	0	0
6.	1	1	1	0	0
7.	2	1	1	1	4
8.	1	2	1	2	1
9.	1	2	0	2	0
10.	2	0	2	1	0

Each of these frequency weights represents and can be used to generate a random sample with replacement. For instance, in the first bootstrap sample, neither the fourth nor the fifth observation is present. However, in the fifth sample, the fourth observation appears two times.

The sum of the values in any of the newly generated weights will give us the sample size:

```
. quietly summarize freq_1
. display r(sum)
74
```

The result above matches the number of observations in our original dataset.

Now that we have generated the frequency weights, we can specify them in the fweights() option of the bootstrap prefix to perform a regression using the bootstrap sampled datasets:

```
. bootstrap, fweights(freq_1-freq_100): regress price weight i.foreign
(running regress on estimation sample)
Bootstrap replications (100): .........10.........20.........30.........40.....
> ....50.........60.........70.........80.........90.........100 done
Linear regression                               Number of obs =        74
                                                Replications  =       100
                                                Wald chi2(2)  =     50.08
                                                Prob > chi2   =    0.0000
                                                R-squared     =    0.4989
                                                Adj R-squared =    0.4848
                                                Root MSE      = 2117.0206
```

price	Observed coefficient	Bootstrap std. err.	z	P>|z|	Normal-based [95% conf.	interval]
weight	3.320737	.493339	6.73	0.000	2.35381	4.287663
foreign						
Foreign	3637.001	584.5176	6.22	0.000	2491.368	4782.635
_cons	-4942.844	1526.99	-3.24	0.001	-7935.689	-1949.998

We can also perform our analysis using any one of the bootstrap sampled datasets. For example, if we want to refit the model using only the first bootstrap sampled dataset, we can simply type

```
. regress price weight i.foreign [fweight=freq_1]
```

Source	SS	df	MS	Number of obs	=	74
				F(2, 71)	=	42.81
Model	428736736	2	214368368	Prob > F	=	0.0000
Residual	355506733	71	5007137.08	R-squared	=	0.5467
				Adj R-squared	=	0.5339
Total	784243469	73	10743061.2	Root MSE	=	2237.7

price	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
weight	3.377376	.3677504	9.18	0.000	2.644103	4.11065
foreign						
Foreign	3469.655	660.6613	5.25	0.000	2152.334	4786.976
_cons	-4707.386	1212	-3.88	0.000	-7124.045	-2290.726

◁

▷ Example 2: Adjust sample size

By default, `rwgen bsample` generates bootstrap samples of the same size as the original dataset. In our previous example, `auto.dta` contained 74 observations; thus, our frequency weights summed to 74 for each replication. However, we can use the `size()` option to control the number of observations drawn in each bootstrap sample. For example, below we sample 10 observations with replacement from the original `auto` dataset:

```
. use https://www.stata-press.com/data/r19/auto, clear
(1978 automobile data)
. rwgen bsample frq_, reps(1) size(10)
. quietly summarize frq_1
. display r(sum)
10
```

The weights now sum to 10.

◁

▷ Example 3: Resample clusters

If we are interested in generating bootstrap samples of entire clusters rather than individual observations, we can use the `cluster()` option. For example, we have clustered data in `bsample2.dta`.

```
. use https://www.stata-press.com/data/r19/bsample2, clear
. tabulate group
```

group	Freq.	Percent	Cum.
A	15	19.74	19.74
B	10	13.16	32.89
C	11	14.47	47.37
D	11	14.47	61.84
E	29	38.16	100.00
Total	76	100.00	

We have five distinct groups in our data. Below, we resample clusters using `group` as the cluster variable for bootstrap sampling.

```
. rwgen bsample fc_, reps(5) cluster(group)
```

Note that we are resampling clusters rather than individual observations. Let's list the frequencies for the cluster defined by `group == "B"`

```
. list group fc_1-fc_5 if group == "B"
```

	group	fc_1	fc_2	fc_3	fc_4	fc_5
16.	B	1	1	2	1	2
17.	B	1	1	2	1	2
18.	B	1	1	2	1	2
19.	B	1	1	2	1	2
20.	B	1	1	2	1	2
21.	B	1	1	2	1	2
22.	B	1	1	2	1	2
23.	B	1	1	2	1	2
24.	B	1	1	2	1	2
25.	B	1	1	2	1	2

Frequencies remain constant within clusters because the entire cluster is treated as a single unit during the resampling process. This approach ensures that all observations within a cluster are sampled together, maintaining the cluster structure in the bootstrap samples.

◁

▷ Example 4: Stratified samples with unequal sizes

In this example, we demonstrate how to perform stratified sampling with unequal sizes again using `auto.dta`. This method allows us to sample different numbers of observations from distinct groups within the dataset. To sample 10 foreign cars and 5 domestic cars from the original dataset, we first need to create a variable that specifies the desired sample size for each group. We can do this by generating a new variable that assigns the value 10 to foreign cars and 5 to domestic cars:

```
. use https://www.stata-press.com/data/r19/auto, clear
(1978 automobile data)
. generate nsamp = cond(foreign,10,5)
```

Once this variable is created, we can specify it in the `size()` option of the `rwgen bsample` command in addition to specifying the strata variable `foreign` in the `strata()` option:

```
. rwgen bsample fs_, reps(1) strata(foreign) size(nsamp)
. tabulate foreign [fweight=fs_1]
```

Car origin	Freq.	Percent	Cum.
Domestic	5	33.33	33.33
Foreign	10	66.67	100.00
Total	15	100.00	

◁

▷ Example 5: Using matched case–control data with the idcluster() option

In this example, we demonstrate how to resample clusters using matched case–control data. We have a dataset of infants who are paired by being matched on their mother's age. These data were presented in Hosmer, Lemeshow, and Sturdivant (2013) and examined in example 7 of [R] **bootstrap**.

The matching information is stored in the variable pairid. Specifying this matching variable in the group() or strata() option of estimation commands, such as clogit, allows us to account for the inherent dependencies in the data when fitting our models.

When resampling from groups, or clusters, we need to make sure that we are preserving the matching information and that the resampled clusters are uniquely identified in each bootstrap sample. Thus, we use the idcluster() option with rwgen bsample to generate a new variable to uniquely identify these resampled groups. This ensures that the bootstrap samples maintain the correct matching structure, enabling accurate estimation of standard errors for matched case–control studies.

Below, we specify pairid in the cluster() option to preserve the original matching information and generate a new variable named newpairid to uniquely identify the newly resampled groups:

```
. use https://www.stata-press.com/data/r19/lowbirth2, clear
(Applied Logistic Regression, Hosmer & Lemeshow)
. rwgen bsample frq_, reps(50) rseed(1) cluster(pairid) idcluster(newpairid)
```

Now we can specify our newly generated frequency weights in the fweights() option of the bootstrap prefix to reconstruct the resampling process and obtain estimates of the standard errors that account for the matching information.

```
. bootstrap ratio=exp(_b[smoke]-_b[ptd]), fweights(frq_1-frq_50):
> clogit low lwt smoke ptd ht ui i.race, group(newpairid)
(running clogit on estimation sample)
Bootstrap replications (50): .........10.........20.........30.........40......
> ...50 done
Bootstrap results                                       Number of obs = 112
                                                        Replications  =  50
      Command: clogit low lwt smoke ptd ht ui i.race, group(newpairid)
        ratio: exp(_b[smoke]-_b[ptd])
                                 (Replications based on 56 clusters in pairid)
------------------------------------------------------------------------------
             |   Observed   Bootstrap                         Normal-based
             |coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
       ratio |   .6654095   1.156848     0.58   0.565    -1.601972    2.932791
------------------------------------------------------------------------------
```

◁

▷ Example 6: Bayesian bootstrap

In this example, we will demonstrate how to use the Bayesian bootstrap technique with the rwgen bayes command to analyze auto.dta. We first load the data:

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

According to Rubin (1981), the Bayesian bootstrap requires specifying a prior for each unique observation, representing our prior beliefs about the outcome probabilities. By default, the rwgen bayes command assigns a prior power value of −1 to all observations, which corresponds to an improper noninformative prior.

We use the following command to generate 100 replicate weights for each observation, which will be stored in new variables named ival1_1 through ival1_100, and we specify the rseed(20) option for reproducibility:

```
. rwgen bayes ival1_, reps(100) rseed(20)
```

The rwgen bayes command generates weights in the $[0, 1]$ interval, with the summation of weights equal to 1. These weights represent a sample of the posterior probability distribution over the observations, incorporating our prior beliefs. For example, below we display summary statistics for the first set of replicate weights (ival1_1):

```
. summarize ival1_1
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
     ival1_1 |         74    .0135135    .0114578    .0001009   .0443107

. display r(sum)
1
```

The generated weights can be specified in the iweights() option of the bootstrap prefix. iweights() stands for importance weights, and they allow us to perform weighted analyses, accounting for the varying probabilities of each observation as specified by the Bayesian bootstrap. To demonstrate, we will use the regress command with the generated replicate weights:

```
. bootstrap, iweights(ival1_1-ival1_100): regress mpg weight foreign
(running regress on estimation sample)

Bootstrap replications (100): .........10.........20.........30.........40.....
> ....50.........60.........70.........80.........90.........100 done

Linear regression                               Number of obs =     74
                                                Replications  =    100
                                                Wald chi2(2)  = 169.45
                                                Prob > chi2   = 0.0000
                                                R-squared     = 0.6627
                                                Adj R-squared = 0.6532
                                                Root MSE      = 3.4071

------------------------------------------------------------------------------
             |   Observed   Bootstrap                         Normal-based
         mpg | coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
      weight |  -.0065879   .0005116   -12.88   0.000    -.0075906   -.0055852
     foreign |  -1.650029   .9949633    -1.66   0.097    -3.600121    .3000631
       _cons |    41.6797   1.695628    24.58   0.000     38.35633    45.00307
------------------------------------------------------------------------------
```

◁

▷ Example 7: Changing prior power specification in Bayesian bootstrap

Below, we will build upon the previous example to demonstrate how altering the prior power specification affects the estimated results using the Bayesian bootstrap technique.

By default, rwgen bayes assigns a prior power value of -1 for all observations. This corresponds to using a noninformative prior for all observations. First, let's generate a new variable with higher values for the prior powers; these values will influence the posterior importance weights assigned to each observation.

```
. generate double prior_l2 = runiform(5, 15)
```

Next, we use rwgen bayes to generate a new set of replicate weights based on the new prior powers. This will create new variables ival2_1 through ival2_100:

```
. rwgen bayes ival2_, reps(100) priorpowers(prior_l2)
```

Recall that the importance weights ival1_1 through ival1_100 were generated based on the default prior powers of -1. To compare the weights generated from the two different prior power values, we calculate the variance for ival1_1 through ival1_100 and ival2_1 through ival2_100:

```
. mata: variance(st_data(1, "ival1_1-ival1_100")')
  .0001778708
. mata: variance(st_data(1, "ival2_1-ival2_100")')
  .000016411
```

By examining the summary statistics, you will notice that the variance of the generated replicate weights is much smaller when using the higher prior power value. This is expected because the posterior probability represented by these variables follows a Dirichlet distribution. A higher prior power value results in more concentrated weights, reflecting greater certainty in the prior information.

Now, let's use the bootstrap prefix with the newly generated importance weights ival2_1 through ival2_100.

```
. bootstrap, iweights(ival2_1-ival2_100): regress mpg weight foreign
(running regress on estimation sample)
Bootstrap replications (100): .........10.........20.........30.........40.....
> ....50.........60.........70.........80.........90.........100 done
Linear regression                                   Number of obs =      74
                                                    Replications  =     100
                                                    Wald chi2(2)  = 1745.44
                                                    Prob > chi2   =  0.0000
                                                    R-squared     =  0.6627
                                                    Adj R-squared =  0.6532
                                                    Root MSE      =  3.4071
```

mpg	Observed coefficient	Bootstrap std. err.	z	P>|z|	Normal-based [95% conf.	interval]
weight	-.0065879	.0001582	-41.63	0.000	-.006898	-.0062778
foreign	-1.650029	.3246217	-5.08	0.000	-2.286276	-1.013782
_cons	41.6797	.5088586	81.91	0.000	40.68236	42.67705

Notice that the standard errors are smaller because we used a more informative prior.

◁

▷ Example 8: Influence of uniqueness on posterior estimation of importance weights

In this example, we will explain how the definition of uniqueness influences the importance weights in the Bayesian bootstrap. According to Rubin (1981), the posterior distribution follows a Dirichlet distribution, which combines prior information with the frequency of observations in the sample. The frequency count is based on how uniqueness is defined for the observations.

By default, all variables are used to determine whether an observation is unique. However, we also provide the id() option, which allows users to alter the criteria for determining uniqueness.

We continue the previous example using auto.dta. In the following command, we use the foreign variable to determine uniqueness. This means that all observations with the same value for the foreign variable will be treated as identical when calculating the frequency counts for the posterior distribution:

```
. rwgen bayes uiv_, reps(1) id(foreign)
```

By using the foreign variable to determine uniqueness, the generated weights will be the same for all observations where foreign is equal to "Domestic", and similarly, the weights will be the same for all observations where foreign is equal to "Foreign".

```
. by foreign: summarize uiv_1
```

```
-> foreign = Domestic
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
       uiv_1 |         52    .0140812           0   .0140812   .0140812

-> foreign = Foreign
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
       uiv_1 |         22    .0121716           0   .0121716   .0121716
```

Stored results

rwgen bsample stores the following in r():

Scalars

r(N)	sample size
r(N_reps)	number of replications
r(N_strata)	number of strata
r(N_clust)	number of clusters

Macros

r(method)	bsample
r(stub)	*stub* of new variables
r(rngstate)	random-number state used
r(size)	from the size(*exp*) option
r(strata)	strata variables
r(cluster)	cluster variables
r(idcluster)	new cluster ID variable
r(newvarlist)	generated weight variables

rwgen bayes stores the following in r():

Scalars

r(N)	sample size
r(N_reps)	number of replications

Macros

r(method)	bayes
r(stub)	*stub* of new variables
r(rngstate)	random-number state used
r(id)	variables used to uniquely identify observations
r(priorpowers)	prior powers
r(newvarlist)	generated weight variables

Methods and formulas

The classical bootstrap ([R] **bootstrap**) is a resampling method used to approximate the distribution of a statistic across a sample. Consider a sample $(x_1, x_2, \ldots, x_n)$ from a random variable X. A bootstrap replication is a sample of size m drawn with replacement from $(x_1, x_2, \ldots, x_n)$. In most cases, m will be equal to n, the sample size.

Viewed from a probabilistic perspective, this process involves assigning probabilities to each observation in the sample. A bootstrap sample is drawn from a multinomial distribution with n categories and m independent trials, where success (outcome) probabilities are nonnegative and their sum equals 1. In this setup, each point in the sample has a probability $1/n$ of selection in any draw, although the randomness of the process may result in some points being selected multiple times, or not at all, within a single replication.

Introduced by Rubin (1981), the Bayesian bootstrap 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, allowing for greater variability in the analysis.

We follow Rubin's framework outlined in Rubin (1981), particularly focusing on section 4, to describe the procedures for calculating Bayesian bootstrap weights.

Consider a discrete random variable $\widetilde{X}$ that can take on K distinct values $d_1, d_2, \ldots, d_K$. Each unique value in this set has an associated probability $\pi_k = P(\widetilde{X} = d_k|\boldsymbol{\pi})$, where $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_K)$. For an i.i.d. sample $(x_1, x_2, \ldots, x_n)$ drawn from $\widetilde{X}$, let n_k denote the frequency of each d_k in the sample. For continuous random variables, each observation in the sample is unique, so $K = n$ and $n_i = 1$ for all i.

The Bayesian bootstrap introduces a prior distribution for $\boldsymbol{\pi}$, proportional to $\prod_{k=1}^{K} \pi_k^{l_k}$, reflecting prior beliefs about the outcome probabilities. Updating this prior with the categorical data, the posterior distribution of $\boldsymbol{\pi}$ follows a $(K-1)$-variate Dirichlet distribution, Dirichlet$(n_1+l_1+1, n_2+l_2+1, \ldots, n_K+l_K+1)$ proportional to $\prod_{k=1}^{K} \pi_k^{n_k+l_k}$.

For notational simplicity, let's assume $K = n$. Following Rubin (1981) and Hastie, Tibshirani, and Friedman (2009), the default prior is set to an improper noninformative prior proportional to $\prod_{k=1}^{K} \pi_k^{-1}$, with $l_k = -1$ for $k = 1, 2, \ldots, n$. This way, the posterior distribution is Dirichlet(1,1,…,1), meaning each data point has the same posterior probability. For more advanced usage, the `priorpowers()` option allows you to specify your own values for l_k. However, you need to make sure that those values produce an actual posterior distribution (that is, the formula for the posterior distribution function has a finite integral on its domain); a sufficient condition, for example, would be that $n_k + l_k + 1$ be positive for all observations.

`rwgen bayes` uses Mata's `rdirichlet()` function (see `help mata rdirichlet()`) to generate Bayesian bootstrap weights w_k's from the posterior Dirichlet distribution with shape vector $A = (n_1 + l_1 + 1, n_2 + l_2 + 1, \ldots, n_K + l_K + 1)$. Note that for every element a of A, 1e–3 < a < 1e+8. When $K < n$, the generated weights are distributed evenly among duplicate observations with a unique value d_k as w_k/n_k for $k = 1, 2, \ldots, K$. This approach ensures that the sum of the weights across all observations accurately reflects the true posterior probability distribution, integrating both the empirical data and prior beliefs into our analysis.

References

Hastie, T. J., R. J. Tibshirani, and J. H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer. https://doi.org/10.1007/978-0-387-84858-7.

Hosmer, D. W., Jr., S. A. Lemeshow, and R. X. Sturdivant. 2013. *Applied Logistic Regression*. 3rd ed. Hoboken, NJ: Wiley.

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

Also see

[R] **bayesboot** — Bayesian bootstrap estimation

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

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

[D] **sample** — Draw random sample

[D] **splitsample** — Split data into random samples