DOCUMENTATION FOR AN S BOOTSTRAP PACKAGE

BY

JERRY HALPERN

TECHNICAL REPORT NO. 163

JUNE 1993

PREPARED UNDER THE AUSPICES

OF

PUBLIC HEALTH SERVICE GRANTS

5 R01 GM21215-17 AND 5 R01 CA55325

DIVISION OF BIOSTATISTICS

STANFORD UNIVERSITY

STANFORD, CALIFORNIA
Documentation For An S Bootstrap Package

By

Jerry Halpern

Technical Report No. 163

June 1993

Prepared Under the Auspices

Of

Public Health Service Grants

5 R01 GM21215-17 and 5 R01 CA55325

Division of Biostatistics

Stanford University

Stanford, California
DOCUMENTATION FOR AN S BOOTSTRAP PACKAGE

by Jerry Halpern

Abstract

This technical report describes a software package from the point of view of the user. The package allows the practicing statistician to "bootstrap" a wide variety of statistics. The package is designed to be run in an S programming environment which supports dynamic loading of user routines. Running times for practical analyses are seconds or a few minutes. The program offers multiple linear regression, multiple logistic regression (including Efron's bias corrected confidence limits), the ability to bootstrap any S function (either one intrinsic to S or one created by the user) which takes one or two matrices as arguments and for which the resampling of the data is pairwise, separate, or suitable for an S regression analysis. There is also a facility for more sophisticated users to attach their own statistical routines. The examples given in this report show how to use the package and that, for its practical use, the user needs only a rudimentary knowledge of S.

The program is invoked on the Sequoia Hall machine named “playfair” with the following commands beginning at the unix prompt. Also see the commented examples below:

1. S
2. library(“bootstrap”)
3. Boot(x) or Boot(x,y) or z_Boot(x) or z_Boot(x,y)

In 3 above x and y are S data sets (eg. x_c(1:10)) each of which is an S vector or an S matrix as is appropriate for your specific application. z stores the output for later display and use.

Any suggestions you may have about this package (for example: useful common resampling schemes, particular kinds of output you would like, comments about the organization of the package) are welcome. please send them to the above mailbox.

Currently there are three categories of functions:

a) The prototype functions available to the user such as the logistic regression. These are implemented in C. The first time in a session they take some appreciable fraction of a minute to load, but on subsequent calls they execute rapidly.

b) S-functions inputted by the user (both those supplied directly by S and those created by the user). These suffer from the sluggishness with which S performs loops. They work and work well enough to be useful, but are far slower than the routines in a) above.

c) See Appendix. The more ambitious user can make his own FORTRAN or C routine available to this bootstrap package.
Instructions, by commented examples, for using the S bootstrap program, called "Boot".

In each of the examples A through I, below, the output from all the bootstrap replications is stored in z by the statement, "z_Boot(...)", which calls Boot and stores its output in the S list, z. In practice one might do 50 or 500 bootstrap replications, and then summarize the results in z using, for instance, means, standard deviations, percentiles, graphs, and tabulations of z by applying the appropriate S functions to z. In the examples below a listing of z, produced with the S statement,"z", is given. "%%" is the unix prompt and > is the S prompt in what follows.

\[
\text{student2}\%\ S
\]

> library("'bootstrap'")

This attaches the bootstrap library. x1, x2, x012, y, and u, which follow, are matrices used below in the examples A through I.

> x1

[,1]
[1,] 1
[2,] 2
[3,] 3
[4,] 4
[5,] 5
[6,] 6
[7,] 7
[8,] 8
[9,] 9
[10,] 10

> x2

[,1] [,2]
[1,] 1  1
[2,] 2 10
[3,] 3  2
[4,] 4  9
[5,] 5  3
[6,] 6  8
[7,] 7  4
[8,] 8  7
[9,] 9  5
[10,] 10 6

> x012

[,1] [,2] [,3]
[1,] 1  1  1
[2,] 1  2 10
[3,] 1  3  2
[4,] 1  4  9
[5,] 1  5  3
[6,] 1  6  8
[7,] 1  7  4
[8,] 1  8  7
[9,] 1  9  5
[10,] 1 10  6
> y
  [,1]
[1,]  6.5
[2,]  33.8
[3,]  11.8
[4,]  34.3
[5,]  15.8
[6,]  33.6
[7,]  18.9
[8,]  31.1
[9,]  26.4
[10,] 28.7

> u
  [,1]
[1,]  0
[2,]  1
[3,]  0
[4,]  0
[5,]  1
[6,]  1
[7,]  1
[8,]  0
[9,]  1
[10,]  1

EXAMPLE A: A single univariate sample.

> z_Boot(y)

Here Boot is called with one argument having a single column, the univariate sample, y. Boot is the function that starts the bootstrap process. It may be called with either one argument, “Boot(a1)”, or with two arguments, “Boot(a1,a2)”. Examples using two arguments are given below. Each of the arguments, a1 and a2, may be either a vector or a matrix. They need not have either the same row or column dimensions nor do both of them need to be matrices. It is only required that they be appropriate for the job to be done. For example, for a linear regression a1 and a2 must have the same number of rows, but for a t-test a1 and a2 may have different numbers of rows. When a1 or a2 is a vector or a 1 by k matrix, a copy of it is changed to a k by 1 matrix by the program before being used.

How many bootstrap replications do you want?
1: 5

Select a statistic to bootstrap
1: trimmed mean
2: personal source code
3: other
Selection: 3
Enter the function call in the form ’<S function name>(x,arg1,...,argk)’

The selections “1: trimmed mean”, and “2: personal source code”, are put in by way of examples of options that can be offered via a menu. 3 allows the user to enter any valid S function which operates on a single matrix (in this case, since y has one column, it must make sense for the function to operate on such a matrix). The rows of the first argument, y, will be sampled with replacement. Enter the function call in the form “< S function name > (x,arg1,...,argk)”. The first argument, x, must be identified with the matrix specified in Boot, in this case y; arg1,...,argk, are k (possibly 0) further arguments that may be required by f.
1: var(y)

> z
"summary N-reps=5 N-y=10"
  mean std 5% 95%
[1,] 95.71987 33.91215 61.15433 148.3566

$Reps:
[1,] 79.47289 148.3566 81.22267 108.3929 61.15433

$Original:
$Original[[1]]:
[1] 102.3121

$Original$Wx:
[1] 10

"N-reps":
[1] 5

The result of Boot is a list, z:

z[1]=z$summary always has 4 elements: the mean of the bootstrap replications, the standard deviation of the bootstrap replications, the lower and upper linearly interpolated 5% points of the replications. z[1] may also have other information in its name, in this case N-reps, the number of replications successfully performed and N-y the number of rows of the matrix argument.


z[3]=z$Original contains the results, as may be appropriate to the analysis (in this case the variance of the data in y, and the number of data points in y), of the actual sample data.

z[4]=z$"N-reps" always contains the number of replications successfully performed.

z may contain arbitrarily more additional elements; but in this case has only the required 4.

EXAMPLE B: A single bivariate sample, x2.

> z_Boot(x2)

Boot is called with a single argument which has two columns.

How many bootstrap replications do you want?
1: 3

Select a statistic to bootstrap
1: trimmed mean
2: personal source code
3: other
Selection: 3
Enter the function call in the form 'f(x, arg1, ..., argk)'
1: var(x2)

The S function var when called with a matrix, x2, produces the covariance matrix corresponding to the pairwise comparison of the columns of x2. In the printout of the replications below, row1 is the variance of the values in column1 of x2, row2 is the covariance of column1 of x2 with column2 of x2, row3 is the covariance of column2 of x2 with column1 of x2, and row4 is the variance of column 2 of x2.
> z
> summary N-reps=3 N-x2=10:
>       mean     std  5%    95%
> [1,] 8.374074 2.158427 6.455556 10.711111
> [2,] 4.566667 2.676440 1.700000  7.000000
> [3,] 4.566667 2.676440 1.700000  7.000000
> [4,] 6.366667 2.155441 4.055556  8.322222

> Reps:
>       [,1]   [,2]   [,3]
> [1,] 6.455556 7.955556 10.711111
> [2,] 1.700000 5.000000  7.000000
> [3,] 1.700000 5.000000  7.000000
> [4,] 8.322222 4.055556  6.722222

> Original:
> Original[[1]]:

> Original$Nx:
> [1] 10

> N-reps:
> [1] 3

EXAMPLE C: Multiple regression with fixed effects. x2 is the k by p design matrix; y is the k by 1 column of observations.

> z_Boot(x2,y)

Boot is applied with two arguments.

How many bootstrap replications do you want?
1: 3

1: Resample from (x1i,...,xik,yi1,...,yik) x and y are paired
2: Resample from (x1i,...,xik) and (yi1,...,yik) separately
3: Other

If “1.”, the first alternative, is selected then x and y are paired as for a paired t test or for a regression analysis with multivariate (as in a random effects model) sampling. Note that x2 and y must have the same number of rows. Note also that in our example y is k x 1, but some functions can have two arguments of dimensions k1 x k2 and k3 x k4 with ki > 1 for i=1,...,4; e.g. “cor” can correlate each column of argument 1 with each column of argument 2.

If alternative “2.” is selected then resampling is done as for an unpaired t test.

Here, for alternative “3.”, neither the rows of x2 nor the rows of y, separately or paired are necessarily resampled. Another menu appears and presents alternatives to selections 1 and 2. In this example we chose “3.”.
Selection: 3
Select a statistic to bootstrap
1: Regression coeefs--Any S Regression function with a fixed x; resampled residuals
2: Fast least squares regression coeefs--fixed x; resampled residuals.
3: Fast logistic regression coeofs--fixed x; binomially resampled outcomes,y.
Selection: 1
Enter a call to any S regression function which returns the coefs
as the first component of a list and the residuals as the second
component of the list. The call must be of the form
<regression function name>=<(x,y,arg1,...,argk).

Here, for selection 1, the user can enter a call to any S regression function which returns the coefs as the first component of a list and the residuals as the second component of the list. The call must be of the form 'f(x,y,arg1...argk)'. Examples of "f" in S are lsfit, l1fit, rreg, rbiwt. An appropriate user defined function could also be entered at this point.

1: lsfit(x2,y)

> z

$summary N-reps=3 N=8=10 N-y=10"

  mean std 5% 95%
[1,] 3.2081874 0.5629707 2.7717078 3.8436205
[2,] 0.7726917 0.1353109 0.6539987 0.9200325
[3,] 3.0239999 0.1272859 2.9359010 3.1699348

$Reps:

[,1] [,2] [,3]
[1,] 3.8436205 3.0092341 2.7717078
[2,] 0.7440497 0.6539987 0.9200325
[3,] 2.9661640 3.1699348 2.9359010

$Original:

$Original[[1]]:
[1] 2.4605263 0.8413158 3.0913158

$Original$Ny:
[1] 10

$"N-reps":
[1] 3

In the rows of z$summary are the parameter estimates from the bootstrap process. 3.208 is the estimated constant; it is the mean of the first row of z$Reps. .773 is the estimate of the slope associated with the data in the first column of x2, and 3.024 is the estimate of the slope associated with the data in the second column of x2. The least squares estimates from the original data set are 2.461, .0841, and 3.091 respectively.

EXAMPLE D: Regression under multivariate sampling (as in a random effects model)

> z_Boot(x2,y)

How many bootstrap replications do you want?
1: 3

1: Resample from (x1,...,xik,y1,...,yik); x and y are paired
2: Resample from (x1,...,xik) and (y1,...,yik) separately
3: Other
Select a statistic to bootstrap
1: Cor
2: Difference in means of x and y for pairs, (x, y)
3: Other
Selection: 3
Call must be of the form '<S function name>(x,y, arg1, ..., argk)'.
First component of list returned by the function will be bootstrapped.
1: lsfit(x2,y)

> z
$s"summary N-reps=3 N-x2=10 N-y=10":$
  mean     std    5%    95%
[1,]  2.5543872 0.10349069 2.4655312 2.6680163
[2,]  0.8544349 0.02572838 0.8264384 0.8770411
[3,]  3.0616927 0.04371848 3.0112476 3.0885810

$Reps:
[1,]  2.6650163 2.4655312 2.5296142
[2,]  0.8770411 0.8264384 0.8598251
[3,]  3.0112476 3.0885810 3.0852495

$Original:
$Original[[1]]:
[1]  2.4605263 0.8413158 3.0913158

$Original$N_y:
[1]  10

$s"N-reps":$
[1]  3

EXAMPLE E: Matched pair analysis of mean difference. A simple example using the menus that are provided.

> z_Boot(x1,y)
How many bootstrap replications do you want?
1: 3

1: Resample from (x1i,...,xik,yi1,...,yik); x and y are paired
2: Resample from (x1i,...,xik) and (yi1,...,yik) separately
3: Other
Selection: 1

Select a statistic to bootstrap
1: Cor
2: Difference in means of x and y for pairs, (x, y)
3: Other
Selection: 2
EXAMPLE F: Analysis of the difference in the means of two independent samples.

> z_Boot(x1,y)
How many bootstrap replications do you want?
1: 3

1: Resample from (x1i,...,xik,yi1,...,yik); x and y are paired
2: Resample from (x1i,...,xik) and (yi1,...,yik) separately
3: Other
Selection: 2

Select a statistic to bootstrap
1: Difference of means of x and y. x and y independent samples.
2: Other
Selection: 1

> z
"summary N-reps=3 N-x1=10 N-y=10":  
      mean  std  5%  95%
[1,] -23.36667 1.82237 -25.38 -21.83

$Reps:
   [,1] [,2] [,3]
[1,] -22.89 -25.38 -21.83

$Original:
$Original$difmean:
[1] -18.59

$Original$NxNy:
[1] 10 10

"N-reps":
[1] 3
EXAMPLE G: Defining and using a statistic for paired data. In all of the above examples the statistics being bootstrapped are all directly provided by the S package. But a user can create a function in the S language and use that function. Here is a simple demonstration.

```r
> sxy_function(x,y)sum(x)-sum(y)
```

The function, sxy, is now defined and can be called. Note that a user defined function must be created before the invocation of Boot.

```r
> z_Boot(x1,y)
```

How many bootstrap replications do you want?
1: 3

1: Resample from \((x_1i,\ldots,x_{ik},y_1i,\ldots,y_{ik})\); \(x\) and \(y\) are paired
2: Resample from \((x_1i,\ldots,x_{ik})\) and \((y_1i,\ldots,y_{ik})\) separately
3: Other
Selection: 1

Select a statistic to bootstrap
1: Cor
2: Difference in means of \(x\) and \(y\) for pairs, (\(x,y\))
3: Other
Selection: 3
Call must be of the form `<S function name>(x,y, arg1,\ldots, argk)`. First component of list returned by the function will be bootstrapped.
1: sxy(x1,y)

```r
> z
```

```r
$"summary N-reps=3 N-x1=10 N-y=10":
   mean std  5%  95%
[1,] -216.8 48.99602 -257.2 -162.3
```

```r
$Reps:
   [,1]  [,2]  [,3]
[1,] -162.3 -257.2 -230.9
```

```r
$Original:
$Original[[1]]:
   [1] -185.9
```

```r
$Original$N_y:
   [1] 10
```

```r
$"N-reps":
   [1] 3
```

EXAMPLE H: Logistic regression.

```r
> z_Boot(x012,u,c(.975,.95,.925,.9))
```

Here an extra parameter, a vector of upper percentage points, is optionally specified. The default is (.975,.95,.90,.85).

As implemented for “Fast logistic regression” (also for “Fast least squares regression”), see below, the constant term is not assumed. To include a constant term, the first column of \(x\), in this case \(x012\), must be all ones’s.
How many bootstrap replications do you want?
1: 3

1: Resample from (xi1,...,xik,yi1,...,yik); x and y are paired
2: Resample from (xi1,...,xik) and (yi1,...,yik) separately
3: Other
Selection: 3

Select a statistic to bootstrap
1: Regression coefs—Any S Regression function.fixed x; resampled residuals
2: Fast least squares regression coefs—fixed x; resampled residuals.
3: Fast logistic regression coefs—fixed x; binomially resampled outcomes,y.
Selection: 3

Second order calcs for one of the b? —enter 0
Second order calcs for 1/(1+exp(-sum(x'b)))?—enter 1
Second order calcs for ratio of two b's? —enter 2
To exit enter -1.
->0
Enter i for beta_i of interest.—>1

Here a method (cf. T. Diciccio and B.Efron(1990)), Better Approximate Confidence Intervals in Exponential Families, Bio-statistics Technical Report #137, Division of Biostatistics, Stanford University) for finding the bootstrap confidence limits, the "BCa" limits, for various of the coefficients is made available. This method avoids the usual replications.

<table>
<thead>
<tr>
<th>alph</th>
<th>BCa-low</th>
<th>BCa-high</th>
<th>standard.low</th>
<th>standard.high</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9750</td>
<td>-4.4997314</td>
<td>6.0867153</td>
<td>-6.0655376</td>
<td>2.0622760</td>
</tr>
<tr>
<td>0.9500</td>
<td>-4.1660957</td>
<td>4.3249927</td>
<td>-5.4121609</td>
<td>1.4088994</td>
</tr>
<tr>
<td>0.9250</td>
<td>-3.9234410</td>
<td>3.3231387</td>
<td>-4.9863431</td>
<td>0.9830816</td>
</tr>
<tr>
<td>0.9000</td>
<td>-3.7217140</td>
<td>2.6205331</td>
<td>-4.6586598</td>
<td>0.6553983</td>
</tr>
</tbody>
</table>

Second order calcs for one of the b? —enter 0
Second order calcs for 1/(1+exp(-sum(x'b)))?—enter 1
Second order calcs for ratio of two b's? —enter 2
To exit enter -1.
->-1

> z
$s"summary N-reps=3 N-x012=10 N-u=10":$

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>std</th>
<th>5%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>-1.12209360</td>
<td>1.88342476</td>
<td>-2.56376521</td>
<td>1.0088756</td>
</tr>
<tr>
<td>B2</td>
<td>0.30591763</td>
<td>0.19038700</td>
<td>0.17058989</td>
<td>0.5236215</td>
</tr>
<tr>
<td>B3</td>
<td>0.02245298</td>
<td>0.09880912</td>
<td>-0.08776388</td>
<td>0.1031040</td>
</tr>
</tbody>
</table>

$Reps: [ ,1] [ ,2] [ ,3]$
B1 1.00887655 -1.8113911 -2.5637652
B2 0.17058989 0.2235415 0.5236215
B3 -0.08776388 0.1031040 0.0520188
EXAMPLE I: Using this package at the command level without invoking the menu.

The following functions other than Bdesc can be called from the command line as
[variable].[function name]([matrix name],[number of replications wanted]) or
[variable].[function name]([matrix name],[matrix name],[numbr of rep reps wanted]),
depending on whether [function name] is a function of one or two matrices.

It is important to note that when a function is called directly rather than through Boot() and its associated menus,
the argument of the functions called must explicitly exist as S matrices; that is, they must have a non-null dim
attribute. This point is illustrated at the beginning of the following example.

BDESC is invoked by Bdesc([variable]) where variable is the output list from any one of the other functions.
    B1func  Bootstraps S functions of one matrix.
    C      B1lmean Bootstraps trimmed mean of a vector.
    C      B1ufc   Bootstraps some user defined function of one matrix resampling from the rows of the matrix.
    C      B2corr  Bootstraps correlation of two vectors.
    C      B2fcreg Bootstraps fixed effects linear regression coefficients.
    C      B2func  Bootstraps S functions of two matrices for which the rows are unpaired.
    C      B2lgt   Bootstraps logistic regression coefficients.
    C      B2ls    Bootstraps fixed effects linear regression coefficients.
    C      B2mean  Bootstraps means of two vectors for which the rows are unpaired.
    C      B2pmean Bootstraps means of two vectors for which the rows are paired.
    C      Bdesc   Takes the output of one of the above functions and creates the list, z, demonstrated in the examples
                      A-H above.

A "C" by a function name means that the main work of the function is done by a program that I wrote in C and
have made available to S by compiling and linking for dyn.load() as described in the S manual(1988), section 7.2.

Now here is an example:

> u_c(u)
> x1_c(x1)
> z_B2lgt(x1,u,10)
Error in double(nn * (d2 + intc)): Trying to allocate a vector with too
many elements (2147483647)

The error occurred because B2lgt is being called directly, and u and x1 as created here are not recognized by S as
being matrices; their dim attribute is null. See my comments above.
Here again is the example:
> u_matrix(u,10,1)
> x1_matrix(x1,10,1)
> z_B2lgt(x1,u,10)

Second order calcs for one of the b?  -enter 0
Second order calcs for 1/(1+exp(-sum(x'b)))?  -enter 1
Second order calcs for ratio of two b's?  -enter 2
To exit enter -1.
->0
Enter i for beta_i of interest.  ->1

<table>
<thead>
<tr>
<th>alph</th>
<th>BCa-low</th>
<th>BCa-high</th>
<th>standard.low</th>
<th>standard.high</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9750</td>
<td>-0.1920622</td>
<td>0.3003640</td>
<td>-0.00972059</td>
<td>0.3575620</td>
</tr>
<tr>
<td>0.9500</td>
<td>-0.1263012</td>
<td>0.2777608</td>
<td>-0.0606481</td>
<td>0.3210043</td>
</tr>
<tr>
<td>0.9000</td>
<td>-0.0620387</td>
<td>0.2491626</td>
<td>-0.0184882</td>
<td>0.2788443</td>
</tr>
<tr>
<td>0.8500</td>
<td>-0.0212656</td>
<td>0.2281843</td>
<td>0.0099635</td>
<td>0.2503926</td>
</tr>
</tbody>
</table>

Second order calcs for one of the b?  -enter 0
Second order calcs for 1/(1+exp(-sum(x'b)))?  -enter 1
Second order calcs for ratio of two b's?  -enter 2
To exit enter -1.
->-1

> z
$Reps:
[1] 0.15859139 0.14398550 0.63082732 -0.02607067 -0.01560521 0.43514062

[7] 0.17415760 0.06929098 0.38104390

$Original:
$Original$coeff:
[1] 0.1301781

$Original$std:
[1] 0.1159889

$Original$My:
[1] 10

"N-reps":
[1] 9
There is an incipient capability for users to supply their own FORTRAN or C routines. At present it only exists under response “2.personal source code” (after beginning with Boot(x)). The routine must be named “myprog”; first line must be either subroutine myprog(...) for FORTRAN or myprog(...) for C. The source code for myprog must then be stored in the user’s directory from which S was invoked under whatever name the user wants to call the routine; i.e. it must be in a file called [rname].f for FORTRAN or [rname].c for C. The function will be known to S as “rname”, not as “myprog”.

Look at driv1.c in /home/funn/bstdir on the Stanford Statististics department machine, playfair, to discover the arguments and types that are required. When more fully developed, I will describe them in this space. When the argument x is a matrix, it is passed without alteration (i.e. in the order x(1,1),...x(n,1),x(1,2),...x(1,n),...,x(m,n)) from S to FORTRAN routines, but for C it is passed as *x in the order x(1,1),x(1,2),x(1,3),...,x(2,1),x(2,2),...,x(m,n). This is so that the user may without further ado process the x matrix along the lines suggested in C-Numerical Recipes, pp.16-20.

Should you be moved to write C source programs be careful to use the S_alloc function mentioned in the S manual. The use of malloc or calloc followed by free is problematical and should not be used. The S code which takes care of compiling, linking, and loading into S your personal routines can be found in the S function B1ufc. It depends on the structure and content of S, C, FORTRAN and other libraries that exist on the particular machine being used, and so some code in B1ufc may have to be changed to use ones own routines. Notice particularly that FORTRAN subroutines are linked using the -lc library, but C subroutines are linked without the -lc library, if the C routine does any io with say printf, scanf, etc. THIS IS IMPORTANT.