

# A Tutorial on Bootstrapping in the SAS® System

Paul A. Thompson, Ph.D.  
Case Western Reserve University  
Cleveland, OH 44106

## Abstract

Bootstrapping is a non-parametric method for the estimation of variances for poorly-understood or mathematically-intractible parameters. It is quite simple to perform a bootstrap in SAS, although it takes some understanding of the process. In bootstrapping, there are actually three components to the method: drawing a bootstrap sample, computing a certain parameter estimate under the bootstrapping framework, and using the variance estimate from the bootstrap method (or the bootstrap sampling distribution itself). After briefly describing the theory inherent to bootstrapping, the three components to bootstrapping are described in SAS using code to illustrate the technique. Finally, a variety of different applications of bootstrapping methods are described.

## Introduction

Bootstrapping is a non-parametric method which can be used to estimate variability and bias of the parameter estimate. It is a method best used in three situations: 1) when little else is known about a parameter estimate; 2) when the parameter estimation method is mathematically intractible; or 3) when distributional assumptions required for more some methods cannot be met.

The basic ideas of bootstrapping are discussed in a variety of sources (Diaconis and Efron 1983; Efron 1982; Efron and Tibshirani 1986; Mooney and Duval 1993; Thompson 1991). The essential notion of the bootstrap lies in using the natural variation of the actual sample to obtain some approximation of the variation inherent in a parameter estimate. Efron (1982) discusses many types of bootstrap methods, but is rather technically oriented. The discussion in Mooney and Duval (in the Sage *Quantitative Applications in the Social Sciences*) is somewhat more generally aimed. Other somewhat more accessible sources include Efron and Tibshirani (1986) and Diaconis and Efron (1983). Thompson presents a very applied discussion of several more complex bootstrapping applications. This discussion will consider exactly how to do a bootstrap in the SAS® system for data management (SAS), omitting most technical considerations.

## Terminology

Consider a parameter  $\theta$ , which is some function of values drawn from some population  $\mathcal{F}(X)$ . When the parameter is estimated, a sample  $\hat{\mathcal{F}}(X)$  is drawn from the population, and the parameter is estimated from the sample:

$$\hat{\theta} = \theta_o(\hat{\mathcal{F}}(X)) \quad (1)$$

The *variability* of the parameter estimate,  $\sigma_\theta^2$  (or its estimate  $\hat{\sigma}_\theta^2$ ) is important for use in inference. While the variability of many parameters is derived through theoretical analysis, others can prove more difficult. In some cases, the estimates are mathematically intractible (i.e., the variability of loadings for multidimensional scaling estimates or factor analysis approaches). In other cases, the data have unreasonable conditions, such as a dependency problem.

**The bootstrap method.** In using the bootstrap method, the observed sample of  $n$  cases is the source of a number of *resamples* or *bootstrap samples*  $\hat{\mathcal{F}}(X)_b^*$ . These resamples are obtained by sampling  $n$  observations *with replacement* from the original sample of  $n$  cases. If the approach is done systematically, each resample is different (when the order of selection is considered); this approach is generally considered infeasible due to "combinatorial

explosion." Thus, the usual method is to draw resamples randomly; resamples differ in probability, but not with certainty. Then each resample is used to obtain a separate estimate of the parameter:

$$\hat{\theta}_{(b)}^* = \theta_O \left( \hat{\mathcal{F}}(X)_b^* \right) \tag{2}$$

where  $\hat{\mathcal{F}}(X)_b^*$  is a resample. The estimates will differ in value, as each is obtained from a different resample. The frequency distribution of  $\hat{\theta}_{(b)}^*$  is the *bootstrap sampling distribution* (BSD) (Efron 1982).

After obtaining the BSD, we may obtain the following bootstrap estimates:

$$\text{Location: } \tilde{\theta}^* = \frac{1}{B} \sum_{b=1}^B \hat{\theta}_{(b)}^* \tag{3}$$

$$\text{Variance: } \hat{\sigma}_{\tilde{\theta}^*}^2 = \frac{1}{B} \sum_{b=1}^B (\hat{\theta}_{(b)}^* - \tilde{\theta}^*)^2 \tag{4}$$

$$\text{Standard error: } \hat{\sigma}_{\tilde{\theta}^*} = \sqrt{\hat{\sigma}_{\tilde{\theta}^*}^2} \tag{5}$$

$$\text{Bias: BIAS} = \hat{\theta} - \tilde{\theta}^* \tag{6}$$

We are also often interested in the various order statistics.

The number of bootstrap resamples required is dependent on the smoothness of the estimate, the intentions of the analyst and the available time on computers. The minimum number suggested is generally 200. When the entire BSD is itself to be examined, possibly 1000-2000 replications should be used. Fortunately, in many situations, analysis of these numbers of resamples is quite "do-able."

Theoretical justification for bootstrapping is discussed in (Mooney and Duval 1993), Efron 1982, Efron and Tibshirani 1986 and many other sources. For the moment, I will say only that:

$$\hat{\mathcal{F}}(X)_b^* \stackrel{n \rightarrow \infty}{\approx} \mathcal{F}(X) \tag{7}$$

$$\hat{\sigma}_{\tilde{\theta}^*} \stackrel{n \rightarrow \infty}{\approx} \sigma_{\theta} \tag{8}$$

and thus, in the long run and as  $n \rightarrow \infty$ , the bootstrap estimates will approach the true values.

## Types of bootstrapping

**Simple bootstrapping.** In evaluating the variability of the sample mean, for instance, we must consider the individual as varying, because the individual and the score from the individual are indistinguishable. The general algorithm for bootstrap selection in this situation is as follows:

1. Determine the number of bootstrap replicates,  $B$ .
2. Obtain the sample  $\hat{\mathcal{F}}$  with  $n$  cases.
3. Select  $B$  resamples of cases from  $\hat{\mathcal{F}}$ , selecting with replacement. If several independent sub-samples  $\hat{\mathcal{F}}_i$  compose the actual sample  $\hat{\mathcal{F}}$ , each must be resampled independently, ensuring that  $n_i$  observations are obtained for each sub-sample.
4. Perform the analysis on each resample, collecting the value of the statistic from each resample. This is the BSD.
5. Use the BSD to obtain either the standard deviation of the parameter estimate (bootstrap estimate of the standard error) or various order statistics.

This approach will be used for many different types of bootstrap samples, in the most basic of situations.

**Regression-style bootstrapping.** When the situation involves a linear model where the source is broken into a fixed and error component, other approaches are needed. In the regression case, the  $X$  matrix is actually considered fixed (see Neter, Wasserman, and Kutner 1985, p. 238). In some areas of econometrics, we consider  $X$  to be random, but this is not the usual case in most other areas of applied statistics. When  $X$  is fixed, we do not resample the entire vector of independent variables, but rather resample for the errors alone. In this case, we must modify Step 3 above as follows:

1. Using the full model, compute  $\hat{Y}_i$  (predicted value) and  $\hat{\epsilon}_i$  (residual term).
2. Generate the  $B$  resamples from  $\hat{\mathcal{F}}$ . Select the error estimates  $\hat{\epsilon}$  using the bootstrap identifications.

- For each sample, use the complete set of values  $\hat{Y}$ . Take the randomly chosen values of  $\hat{\epsilon}_i$  and compute:

$$Y_i^* = \hat{Y}_i + \hat{\epsilon}_i^* \tag{9}$$

The further analyses are then performed using  $Y^*$ .

Note, however, that Efron (1982) presents an argument for closed-form evaluation of  $\hat{\sigma}_{\hat{\theta}}^2$  (where  $\theta$  is the error variance) in the regression-style context.

### Examples of the Use of Bootstrap Methods

Bootstrapping has been used for regression models (Freedman and Peters 1984), tests of differences between groups (Thompson 1991), non-linear regression (Thompson 1991), correlations (Diaconis and Efron 1983) and unfolding (Heiser 1981), among others.

### Performing Bootstrap Analyses in SAS

The process of bootstrapping variance estimates for sample statistics has three basic components: 1) obtain a dataset containing a number of bootstrap resamples; 2) obtain the estimate of the statistic from each resample; 3) use the BSD to perform inference.

#### 1) Resampling using DATA and SORT steps

Simple bootstrapping with a single sample. In SAS (see Page 1), simple bootstrapping is actually rather ... simple! Figure 1 demonstrates the overall approach.

- Generate a dataset containing a number of bootstrap resamples (Lines 1-4), where each resample contains the IDN value to select out the corresponding value from the dataset containing the sample of data for analysis. Seed control is maintained by retaining the seeds in a seed dataset. The number of bootstrap resamples is 100. The CEIL function produces the integer at or above the decimal chosen, and thus are integers corresponding to the identification of cases in the original sample. A corresponding IDN value must be found in the sample.
- PROC FREQ is a randomization check (Line 5). The frequency count should be uniform, and a simple  $\chi^2$  test can be used to test the randomization process to ensure that no errors have occurred.
- Bootstrap resamples are sorted by IDN and then merged in with the dataset DATASRC, which contains the original sample. The construction IF (INB & ^IND) tests for cases in the bootstrap dataset only, which occurs with incorrectly chosen IDN values.
- The dataset containing the resamples, BRUNA, is sorted by BOOT. This completes the resampling process.
- Any SAS procedure which allows the use of the BY variable mechanism (that is, the set of SAS procedures) can be used to analyse the bootstrap resamples dataset. In this case, we produce the dataset CP with Pearson correlation results and CK with Kendall correlation results.
- PROC UNIVARIATE is used to obtain the standard deviation of Both can be examined to obtain bootstrap  $\hat{\sigma}_{\hat{\theta}}^2$ , by determining the variability of the parameter estimate in the output datasets.

When several sub-samples are involved in the actual sample (as in ANOVA applications), Steps 1-4 are repeated as often as needed, and the final bootstrap reample dataset BRUNA is formed by combining the samples.

```

DATA BSAMPA; SET SEEDA;      KEEP IDN BOOT;          /* Generate bootstrap resample */
DO BOOT=1 TO 100; DO SUBCNT=1 TO 30;          /* 100 bootstrap samples */
CALL RANUNI(SD1, IDN); IDN=CEIL(IDN*30);      /* 30 subjects in this sample */
OUTPUT; END; END;          /* Output to file BSAMPA */
5 PROC FREQ DATA=BSAMPA; TABLES IDN;        /* Fr(IDN) for checking */
PROC SORT DATA=BSAMPA; BY IDN;             /* Order by subject number */
DATA BRUNA; MERGE BSAMPA(IN=INB) DATASRC(IN=IND); BY IDN;
IF (INB & ^IND) THEN PUT "Error";          /* This will not happen */
PROC SORT DATA=BRUNA; BY BOOT;             /* Order by BOOT */
10 PROC CORR DATA=BRUNA OUTP=CP OUTK=CK NOPRINT; /* Compute corr for each group */
VAR VARYA VARYB; BY BOOT; RUN;
PROC UNIVARIATE DATA=CP(WHERE=( _NAME_="VARYA")); VAR VARYB; RUN;
PROC UNIVARIATE DATA=CK(WHERE=( _NAME_="VARYA")); VAR VARYB; RUN;
    
```

Figure 1: Bootstrap sample selection, 1-sample situation

## A macro for bootstrapping in SAS

A macro for bootstrapping in SAS is presented in Figure 1, which selects both simple and regression-style bootstrap resamples.

1. Retain seeds in a file (Line 7).
2. Count the number of cases (Line 8).
3. The dataset containing ID values for the resamples is generated (Lines 9-13). At the end, the seed value is output to the seed dataset, to maintain the integrity of the seed stream. SUBCNT is a sequential subject counter.
4. Sort the bootstrap dataset by the ID variable (Line 14).
5. For regression-style bootstrapping, we now:
  - (a) Compute the residual and predicted values (Lines 16-17).
  - (b) Merge the residual values (R&\_dv.) into the bootstrap sample dataset (Lines 18-19).
  - (c) Sort the bootstrap dataset by the sequential subject counter SUBCNT (Line 20).
  - (d) While renaming the subject counter to &\_idv., merge the predicted values (P&\_dv.) into the bootstrap sample dataset. Thus, each bootstrap resample has the same set of values P&\_dv., while the values of the residual R&\_dv. are selected with replacement. Compute the new dependent variable &\_dv. from the residual and predicted parts. (Lines 21-24).
6. For simple bootstrapping, we now select the resample with a merge (Lines 27-28).
7. The final merge organizes the data correctly for analysis BY &\_bvar.

```

%macro _bootgen(_b=200,_base=WORK,_bootout=BOOTSAMP,_bvar=BOOT,_btype=S,
_dv=,_idv=IDW,_ivs=,_sdhld=SEEDHLD,_seed=SEEDS,_seedv=SD1,_srcinv=SRCSET);
/* Vars _b:# bootstrap resample||_base:database||_bootout:dataset with resamples */
/* _bvar:bootstrap identifier||_btype:type of selection-(S)imple,(R)egression */
/* _dv:dependent variable||_idv:ID variable||_ivs:independent variables */
/* _sdhld:seed archive||_seed:seed file||_seedv:seed var||_srcinv:sample */
PROC APPEND DATA=_base..&_seed. BASE=_base..&_sdhld.;RUN;
PROC MEANS DATA=_base..&_srcinv. NOPRINT;VAR &_idv.;OUTPUT OUT=_CNT N=OCNT;RUN;
DATA _base..&_bootout.(KEEP=&_idv. &_bvar. SUBCNT)/* Results dataset */
10  &_base..&_seed.(KEEP=&_seedv.); MERGE &_base..&_seed. _CNT;
DO &_bvar.=1 TO &b.; DO SUBCNT=1 TO OCNT;
CALL RANUNI(&_seedv.,&_idv.);&_idv.=CEIL(&_idv.*OCNT);OUTPUT &_base..&_bootout.;
END; END; OUTPUT &_base..&_seed.;RUN;
PROC SORT DATA=_base..&_bootout.;BY &_idv.;RUN;/* Order by subject number */
15 %if (&btype. = R) %then %do; /* Regression-style */
PROC REG DATA=_base..&_srcinv.(DROP=R&_dv. P&_dv.) NOPRINT;
MODEL &_dv.=&_ivs.;OUTPUT OUT=_base..&_srcinv. P=P&_dv. R=R&_dv.; RUN;
DATA _base..&_bootout.; /* Merge in residual-error */
MERGE &_base..&_srcinv.(KEEP=R&_dv. &_idv. &_dv.) &_base..&_bootout.;BY &_idv.;RUN;
20 PROC SORT DATA=_base..&_bootout.;BY SUBCNT; /* Order by sequential # */ RUN;
DATA _base..&_bootout.; /* Get in Yhat, and set up */
MERGE &_base..&_srcinv.(KEEP=P&_dv. &_ivs. &_idv. &_dv. RENAME=(&_dv.=H&_dv.))
&_base..&_bootout.(RENAME=(&_idv.=S&_idv. SUBCNT=&_idv.));
BY &_idv.;&_dv.=P&_dv.+R&_dv.;RENAME &_idv.=SUBCNT_S&_idv.=&_idv.; RUN;
25 %end;
%else %do;
DATA _base..&_bootout.; /* Merge source into bootout */
MERGE &_base..&_bootout.(IN=IN_B) &_base..&_srcinv.(IN=IN_D);BY &_idv.;RUN;
%end;
30 PROC SORT DATA=_base..&_bootout.;BY &_bvar.;RUN;/* Order by boot sample count*/
%mend _bootgen;
    
```

Figure 2: Bootstrap resampling using DATA and SORT steps

## 1) Resampling using SQL

It seems quite sensible to consider the use of SQL for the resampling process as well. A macro which performs resampling using SQL is shown in Figure 3 (along with a utility macro for counting items).

1. In Lines 1-4, there is a macro to count the number of items in a macro variable.
2. Generate the identifiers for the bootstrap samples. Don't do this in SQL; it is SLOWWW!!
3. For regression-style bootstrapping, the regression estimates must be set up (Lines 13-15). Count the number of items in the macro variable `_ivs` (the macro with independent variables). Then compute the predicted and residual values.
4. For simple bootstrapping, compute the number of items in the macro `_vlist` (Line 16).
5. Next SQL is used to actually perform the resampling.
  - (a) For regression-style bootstrapping, we must nest queries (nested queries are found inside the parenthesized sections of the FROM clause). The first query (Lines 21-23) matches up values of `&_idv.` in the both the bootstrap file and data source, and pulling in the predicted value `P&_dv.` and then calling it `PREDV.` The next query (Lines 24-25) pulls in the estimate from the regression result, matching up by `RANIDV` (the random ID value) in the bootstrap sample and the `&_idv.` identifier in the data source. Finally, the higher-level query (Lines 19-26, with nested queries on the FROM clause) matches up the two tables using the `_bvar.` and `&_idv.` variables.
  - (b) For simple bootstrapping, we match on the random ID number from the bootstrapping dataset (Lines 27-30).

```

%macro _itemcnt(_items=a b, _delim=1, _n=_nit);%let _delx=%str(,*\);%let &n.=0;
%let _ksep=%quote(%substr(&_delx., &_delim., 1));
%do %while(%length(%scan(%str(&_items.),%eval(&&n.+1), %str(&_ksep.))) > 0);
%let &n.=%eval(&&n. + 1); %end;%mend _itemcnt;
5 %macro _bootsql(_b=200, _base=WORK, _bootout=BOOTSAMP, _bvar=BOOT, _btype=S, _vlist=,
  _dv=, _idv=IDV, _ivs=, _sdhld=SEEDHLD, _seed=SEEDS, _seedv=SD1, _srcinv=SRCSET);
  %* _vlist:var list *;
  PROC APPEND DATA=&_base..&_seed. BASE=&_base..&_sdhld.;RUN;
  PROC MEANS DATA=&_base..&_srcinv. NOPRINT;VAR &_idv.;OUTPUT OUT=_CNT N=OCNT;RUN;
10 DATA BOOTINT(KEEP=&_idv. &_bvar. RANIDV) &_base..&_seed.(KEEP=&_seedv.); MERGE &_base..&_seed. _CNT;
  DO &bvar.=1 TO &b.; DO &_idv.=1 TO OCNT; CALL RANUNI(&_seedv.,RANIDV);
  RANIDV=CEIL(RANIDV/OCNT);OUTPUT BOOTINT; END; END; OUTPUT &_base..&_seed.;RUN;
  %if (&btype. = R) %then %do;%let _ci=%_itemcnt(_items=&_ivs.,_n=_ci);
  PROC REG DATA=&_base..&_srcinv.(DROP=R&_dv. P&_dv.) NOPRINT;
15 MODEL &_dv.=&_ivs.;OUTPUT OUT=&_base..&_srcinv. P=P&_dv. R=R&_dv.; RUN;%end;
  %else %do;%let _ci=%_itemcnt(_items=&_vlist.,_n=_ci);%put _ci[&_ci.];%end;
  PROC SQL; CREATE TABLE &_base..&_bootout. AS ORDER BY 1,2
  %if (&btype. = R) %then %do;
20 SELECT F.&bvar.,F.&_idv.,F.RANIDV,F.YHAT,R.EPSEST,F.YHAT+R.EPSEST AS NEWDEP,
  %do _bi=1 %to &_ci.;F.%scan(&_ivs.,&_bi),%end;F.&_dv.
  FROM (SELECT B.&bvar., B.&_idv., B.RANIDV,D.P&_dv. AS YHAT,
  %do _bi=1 %to &_ci.;D.%scan(&_ivs.,&_bi),%end;D.&_dv.
  FROM &_base..&_srcinv. D, BOOTINT B WHERE D.&_idv.=B.&_idv.) F,
  (SELECT B.&bvar., B.&_idv., B.RANIDV, D.R&_dv. AS EPSEST
25 FROM &_base..&_srcinv. D, BOOTINT B WHERE D.&_idv.=B.RANIDV) R
  WHERE F.&bvar.=R.&bvar. & F.&_idv.=R.&_idv.;QUIT;%end;
  %else %do;
  SELECT B.&bvar., B.&_idv., B.RANIDV,
  %do _bi=1 %to &_ci.;D.%scan(&_vlist.,&_bi),%end;D.&_idv. AS G&_idv.
30 FROM &_base..&_srcinv. D, BOOTINT B WHERE D.&_idv.=B.RANIDV;QUIT;%end;
%mend _bootsql;
    
```

Figure 3: Bootstrap resampling using SQL

## 1) Resampling using IML

The SAS/IML<sup>®</sup> system (IML) offers another very convenient approach for selecting the resamples in the bootstrap context (Figure 4). Additionally, since the bootstrap is primarily an investigative tool with new methods, using IML (see Page 6) is very convenient since analysis will often be performed in IML after selection of the resamples. IML is also very easy to use, once the basics have been mastered (i.e., handling SAS datasets). Figure 4 presents a macro which uses IML for resample selection only.

1. Save the seeds for later references (Line 3).
2. When doing regression-style bootstrapping, read the independent, dependent and ID variables into separate matrices, compute the LS regression estimates, and compute the predicted and residual estimates. IML is very convenient for standard operations! (Lines 5-8)
3. When doing simple bootstrapping, read the data into IML (Line 9).
4. Now expand the single seed into a vector of seeds for the easy approach to the generation of bootstrap ID values. There are several considerations which prompt the manner in which RANUNI is handled: 1) it doesn't support subscripted operations; 2) you can't use  $J(nrx, 1, sdx)$  to create a dataset with seeds, as this would produce a constant seed matrix which would generate the same number; 3) before using a matrix in CALL RANUNI, it must exist (Lines 12-13).
5. Actual bootstrap resample generation is very simple at this point (Lines 14-20). As before, different methods are used with regression-style and simple bootstrapping. First, the vector of ID numbers is generated, called RESAMPN. In regression-style resampling, the numbers are used to select the errors, which are then added to the constant values of YPRED to produce the final result, RESVAL. In simple bootstrapping, the RESAMPN vector of ID numbers selects rows of DATASRC directly.
6. Samples are output to the results dataset (Lines 21-22).
7. The seeds are output the seeds dataset, preserving the integrity of the seed stream (Lines 24-25).

```

%macro _bootiml(_b=200,_base=WORK,_bootout=BOOTSAMP,_bvar=BOOT,_btype=S,
_dv=,_idv=IDN,_ivs=,_sdhld=SEEDHLD,_seed=SEEDS,_seedv=SD1,_srcinv=SRCSET);
PROC APPEND DATA=&_base..&_seed. BASE=&_base..&_sdhld.;RUN;
PROC IML;
5 %if (&_btype.=R) %then %do;USE &_base..&_srcinv.;
  READ ALL VAR {&_dv.} INTO YMAT[COLNAME=CNZY];
  READ ALL VAR {&_ivs.} INTO XMAT[COLNAME=CNZX];READ ALL VAR {&_dv.} INTO YMAT[COLNAME=CNZY];
  READ ALL VAR {&_idv.} INTO IDMAT[COLNAME=CNZI];DATASRC=YMAT||XMAT||IDMAT;CNZ=CNZY||CNZX||CNZI;
  BETA=INV(XMAT'*XMAT)*XMAT'*YMAT;YPRED=XMAT*BETA;EPRED=YMAT-YPRED;%end;
%else %do;USE &_base..&_srcinv.;READ ALL INTO DATASRC[COLNAME=CNZ];%end;
10 CNZ="&_bvar."||CNZ;CLOSE &_base..&_srcinv.;NRX=NROW(DATASRC);
  USE &_base..&_seed.;READ ALL INTO SDXZ[COLNAME=SEEDZ];SDZ=SDXZ[1,1];CLOSE &_base..&_seed.;
  SDV=J(NRX,1,1);DO I=1 TO NRX;A=1;CALL RANUNI(SDZ, A);SDV[I]=A;END;
  SDV=CEIL(SDV*3.141593827*100000)-25;RESAMP=J(NRX,1,1);
  DO I=1 TO &_b.;CALL RANUNI(SDV,RESAMP);RESAMPN=CEIL(RESAMP*NRX);
15 %if (&_btype.=R) %then %do;
  YANAL=EPRED[RESAMPN,]+YPRED;RESVAL=J(NRX,1,1)||YANAL||XMAT||IDMAT;
%end;
%else %do;
  RESVAL=DATASRC[RESAMPN,];RESVAL=J(NRX,1,1)||RESVAL;
20 %end;
  IF (I=1) THEN CREATE &_base..&_bootout. FROM RESVAL[COLNAME=CNZ];
  APPEND FROM RESVAL[COLNAME=CNZ];
  END;CLOSE &_base..&_bootout.;
  CREATE &_base..&_seed. FROM SDXZ[COLNAME=SEEDZ];SDXZ[1,1]=SDZ;
25 APPEND FROM SDXZ[COLNAME=SEEDZ];CLOSE &_base..&_seed.;QUIT;
%mend _bootiml;

```

Figure 4: Bootstrapping using IML

### Comparing the three methods

The methods can all be used for generating bootstrap samples, and will appeal to persons depending on their overall orientation. The methods differ in time required for generation of resamples, as shown in Table 1 (run on Gateway 2000 486DX-33 computer, 8 MEG RAM). The IML method appears to be far quicker than the other two, in either resample selection method. It doesn't have the time penalty involved in the use of regression-style bootstrapping. IML does not appeal to all users of SAS, as it requires the user be conversant with matrix algebra; users familiar with IML will probably wish to use it for bootstrapping, given these results.

Table 1: Time in seconds to generate resamples with three variables

Approach	DATA/SORT			SQL			IML		
	100	500	1000	100	500	1000	100	500	1000
Simple	14	86	155	19	110	195	14	30	55
Regression-style	24	94	211	45	279	300	9	34	43

## 2) Methods for Estimation of Quantities

In SAS, the actual computation of repeated quantities is very easy, if you use BY-variable approaches; merely analyse the dataset containing the bootstrap resamples BY `&_bvar`. Thus, SAS is a very good platform to perform bootstrap estimation, because every PROC save IML is designed to work with BY variables. With IML, you merely perform the analysis in a DO-loop, ensuring that all variables are properly set at the start of the next loop.

Once estimated, there are sometimes complications. When bootstrapping, we are often working with parameter estimates of either a new type or which are not standard components of SAS output datasets. Thus, getting the values into a SAS dataset from the PROCs sometimes poses a difficulty. There are generally three approaches which can be used here: 1) placing output into a file using PRINTTO, and reading it into a dataset using INPUT statements. Although difficult to set up at times, this is always reliable; 2) using IML, and outputting the computed statistics into a dataset; 3) using the Output Delivery System (ODS) when it is available. Eventually, Option 3 will be the method of choice, but this is currently available in only a few PROCs.

## 3) Using the Information in the BSD

The BSD, once obtained, is often used to compute a *confidence interval* (CI) corresponding to some probability level. Such intervals can be used to test hypotheses (by seeing if they contain the value of the hypothesized parameter). There are a number of ways of computing CIs, which often result in closely comparable intervals:

$$\text{Normal approximation: } (\hat{\theta} - z_{(\alpha/2)}\hat{\sigma}_{\hat{\theta}}, \hat{\theta} + z_{(1-\alpha/2)}\hat{\sigma}_{\hat{\theta}}) \tag{10}$$

$$\text{Percentile method: } (\mathcal{O}[\hat{\theta}_{(\cdot)}^*, (\alpha/2)], \mathcal{O}[\hat{\theta}_{(\cdot)}^*, (1 - \alpha/2)]) \tag{11}$$

$$\text{Bias-corrected: } (\mathcal{O}[\hat{\theta}_{(\cdot)}^*, \Phi(2z_0 + z_{(\alpha/2)})], \mathcal{O}[\hat{\theta}_{(\cdot)}^*, \Phi(2z_0 + z_{(1-\alpha/2)})]) \tag{12}$$

Equation 10 returns a confidence interval in a rather standard manner, using the  $\hat{\sigma}_{\hat{\theta}}$  obtained from the BSD, and a deviate value obtained from the normal distribution;  $z_{\alpha/2}$  represents the deviate on the normal distribution corresponding to the  $\alpha/2 + 100$ th percentile. Equation 11 represents the rather straight-forward notion that the  $\alpha/2$ th and  $1 - \alpha/2$ th order statistic comprise a confidence interval within which  $1 - \alpha \cdot 100$  % of all values are found. In this notation,  $\mathcal{O}$  refers to the order statistic process *per se*, the first term the variable for the order statistic process, and the second the cumulative proportion in question. Finally, Equation 12 represents a confidence interval which takes into account the bias of estimation (as measured by bootstrap estimates). In this approach, we determine  $z_0 = \Phi^{-1}[P(\hat{\theta}_{(\cdot)}^* \leq \hat{\theta})]$ , where  $\Phi^{-1}$  is the inverse normal distribution function, returning the deviate corresponding to a certain cumulative percentile. This is a measure of the bias of the estimation of  $\hat{\theta}$ . This is added to the deviate corresponding to the nominal level of the confidence interval desired; it functions to provide a non-linear shift in the end-points of the CI, as affected by the degree and sign of  $z_0$ .  $\Phi$  is the percentile function of the normal distribution (returning a percentile corresponding to a particular deviate). Essentially, Equation 12 is a normal approximation CI, with the ends shifted according to a measure of the bias of the statistic.

A macro which sets up these different confidence intervals is presented in Figure 5. This macro has the following features:

1. The macro variables are all similar to those in other macros. Different ones are defined in the macro (Lines 2-3). Prior to running the macro, the values of the statistic(s) in question must be computed both for the original sample (`_ostat`) and for the bootstrap resamples (`_bstat`).
2. Define the percentile values for the lower and upper ends of the interval. Remember, there is no decimal arithmetic in the macro system!
3. Set up the bias-corrected confidence intervals (Lines 8-17). Determine the proportion of bootstrap values lower than the observed statistic value. At the end of the dataset, compute the bounds, as described in Equation 12. Output the lower and upper limits, and convert them into macro variable values.
4. For each variable in `&vlist.`, run UNIVARIATE to compute various statistics for each variable separately. Since the BC-percentile values are different for each variable, this is done one variable at a time.
5. Compute final CI values, using the normal distribution values and the order statistics obtained from the UNIVARIATE procedure.

```

%macro _calcci(_b=, _base=work, _bfinal=bres, _bvar=boot, _cil=95, _vlist=, _ostat=, _bstat=);
%* _bfinal:Final dataset with CIs||_cil:Level of CI (given as 1-alpha) *;
%* _ostat:original statistics dataset||_bstat:bootstrap statistics dataset *;
%let _nvr=;%itemcnt(_items=&vlist., _n=_nvr);
%let _dv=%eval(100-&cil.); %let _hv=%eval(&dv./2);
%if (%eval(&hv.*2) ^= &dv.) %then %let _pctl=&hv..5 %eval(100-&hv.-1).5;
%else %let _pctl=&hv. %eval(100-&hv.);
DATA BIASCHK; SET &base..&bstat. END=EOF; RETAIN _OS1-_OS&nvr.; RETAIN PC1-PC&nvr. 0;
IF (_N_ = 1) THEN SET &base..&ostat.(
10  RENAME=(%do _inv=1 %to &nvr.; %scan(&vlist.,&inv.)=_OS&inv.%end.);
ARRAY VOS _OS1-_OS&nvr.; ARRAY VBS &vlist; ARRAY VPC PC1-PC&nvr.;
DO I=1 TO &nvr.; VPC{I}=VPC{I}+(VBS{I} < VOS{I}); END;
IF (EOF) THEN DO;DO I=1 TO &nvr.;BCPCT=VPC{I}/_N_;VPC{I}=PROBIT(VPC{I}/_N_);
BCLO=PROBNORM(PROBIT(%scan(&pctl.,1," ")/100)+2*VPC{I});
15  BCHI=PROBNORM(PROBIT(%scan(&pctl.,2," ")/100)+2*VPC{I});
CALL SYMPUT(COMPRESS("_bc"||PUT(I,4.)), PUT(BCLO*100,5.2)||" "||PUT(BCHI*100,5.2));
OUTPUT;END;END;KEEP BCLO BCHI BCPCT;RUN;
%do _i=1 %to &nvr.;
PROC UNIVARIATE DATA=&base..&bstat. NOPRINT; VAR %scan(&vlist.,&i.);
20  OUTPUT OUT=CI&i. PCTLPTS=&pctl. &bc&i. PCTLPRE=PCT_ PCTLNAME=OL OH BL BH
MEAN=BBAR STD=BSTD;RUN;
PROC APPEND DATA=CI&i. BASE=ALLCI;RUN;
%end;
PROC TRANSPOSE DATA=&ostat. OUT=TOSTAT(RENAME=( _NAME_ =VARNM VX1=OSTAT)) PREFIX=VX;VAR &vlist.;RUN;
25  DATA &base..&bfinal.;LENGTH VARNM $8;MERGE ALLCI TOSTAT BIASCHK;
ZV=PROBIT(%scan(&pctl.,2," ")/100);
LABEL PCT_BL="Lower/BC BootEst/&cil. pct CI" PCT_BH="Upper/BC BootEst/&cil. pct CI"
NAP_LO="Lower/ParamEst/&cil. pct CI" NAP_HI="Upper/ParamEst/&cil. pct CI"
PCT_OL="Lower/OrderSt/&cil. pct CI" PCT_OH="Upper/OrderSt/&cil. pct CI";
30  NAP_LO=OSTAT-ZV*BSTD;NAP_HI=OSTAT+ZV*BSTD;
RUN;
%mend _calcci;

```

Figure 5: Computation of bootstrap confidence intervals

## Example

One situation for which no consistently-accepted solution exists is the "Behrens-Fisher" problem, in which small samples are taken from two non-normal populations, and the assumption of equal variances cannot be made. There are well-known solutions when we have: 1) large samples, or 2) small samples from normal populations. Another interesting problem, which can be addressed at the same time, is the difference in variance between two groups. Again, when the populations are markedly non-normal, there are known problems with standard solutions to the problem.

In both of these cases, a bootstrapping solution can be used. Two separate bootstrap resampling processes are used, so that each group is resampled separately. Two statistics are computed: 1)  $\bar{X}_1 - \bar{X}_2$ , the difference between means; and 2)  $s_1 - s_2$ , the difference between standard deviations. The BSD is formed by computing this value for each of the  $B$  resamples. At that point, the various CIs may be computed, and hypotheses may be tested using such CIs.

Table 2 presents results from the analysis of two groups. In this table, the values calculated from the two original samples are presented. Using  $B = 500$ , the bootstrap BSD is then examined to obtain the mean and standard deviation. Four types of CIs are then shown, including the standard  $t$  distribution interval (for the means only), the normal approximation interval (Equation 10), the percentile interval (Equation 11) and the bias-corrected interval (Equation 12). As seen in the table, the bootstrap intervals are often similar to the intervals obtained under theoretical analysis. Using the bootstrapping intervals in Table 2, the conclusion would be drawn that there are no differences between the groups, either in terms of means or variances. This conclusion flows from the CIs which all include 0.

Table 2: Confidence intervals under various rules

Variable	Group	Original Statistic	CIs of four types					
			Bootstrap Mean	St.D.	$t$	Normal approx	% method	Bias- corrected
Mean	Group 1 ( $n = 11$ )	-2.97	-2.99	0.64	(-4.22,-1.73)	(-4.22,-1.73)	(-4.33,-1.72)	(-4.26,-1.71)
	Group 2 ( $n = 35$ )	-2.92	-2.92	0.24	(-3.37,-2.47)	(-3.39,-2.45)	(-3.43,-2.45)	(-3.44,-2.46)
	Difference	-0.05	-0.08	0.65	(-1.04,0.93)	(-1.33,1.22)	(-1.43,1.31)	(-1.45,1.31)
Standard Deviation	Group 1	2.28	2.08	0.72		(0.86,3.70)	(0.56,3.40)	(0.70,3.52)
	Group 2	2.48	1.42	0.32		(0.85,2.12)	(0.94,2.05)	(0.99,2.14)
	Difference	0.80	0.66	0.77		(-0.70,2.30)	(-1.00,2.09)	(-0.70,2.27)

## Discussion

This discussion presented three different ways of drawing bootstrap resamples in SAS. Each method has advantages and disadvantages. The DATA/SORT approach is generally simple and relatively straightforward. The PROC SQL (SQL) method is intellectually interesting, but slow and somewhat limited (in that all variables must be specified, which is not a limitation for the other approaches). The IML method is fast and also offers the great advantage of actually performing the calculations in the same general approach as can be used to perform the resampling. As noted above, bootstrapping applications using conventional SASPROC's suffer from the great difficulty of obtaining the results of peculiar and unusual statistics; when using IML, if the result can be phrased as a matrix equation or algorithm, the results are immediately forthcoming. This is a great advantage to an exploratory technique.

After drawing resamples, the multiple resamples must be analysed using some procedure or method. SAS is a good mechanism for this, as the BY process is implemented in nearly all SAS procedures. Finally, inference must be done; several methods for computing confidence intervals of several sorts are shown, and these are applied in a simple and commonly encountered situation.

## Conclusion

Bootstrapping can be very useful in many situations which are generally exploratory in nature. In these situations, a statistic exists, but little can be said about its overall variability, due to mathematical intractability, distributional difficulty or other problem. In such cases, bootstrapping may be the only approach which is viable. As such, SAS offers a very useful platform to do bootstrapping on. Doing bootstrapping in SAS requires us to consider SAS as a fourth-generation language, and consider it in a programming sense, rather than looking for complete "cook-book" approaches. If you can view SAS in this way, bootstrapping is easy in SAS.

## References

- De Leeuw, J. and J. Meulman (1986). A special jack-knife for multidimensional scaling. *Journal of Classification* 3, 97-112.
- Diaconis, P. M. and B. S. Efron (1983). (may) computer-intensive methods in statistics. *Scientific American* 23, 116-130.

- Efron, B. S. (1982). *The jackknife, the bootstrap and other resampling plans*. Philadelphia, PA: Society for Industrial and Applied Mathematics.
- Efron, B. S. and R. Tibshirani (1986). Bootstrap methods for standard errors, confidence intervals and other measures of statistical accuracy. *Statistical Science* 1, 54-74.
- Freedman, D. A. and S. C. Peters (1984). Bootstrapping a regression equation: Some empirical results. *Journal of the American Statistical Association* 79, 97-106.
- Heiser, W. J. (1981). *Unfolding analysis of proximity data*. Leiden, The Netherlands: Department of Data Theory.
- Mooney, C. and R. Duval (1993). *Bootstrapping*. Newbury Park, CA: Sage.
- Neter, J., W. Wasserman, and M. H. Kutner (1985). *Applied linear statistical models*. Richard D Irwin, Inc: Homewood.
- Thompson, P. A. (1991). Re-sampling approaches to hypothesis testing in structural models. *Multivariate Behavioral Research* 26, 737-763.

**Contacting the author.** For more information about bootstrapping in SAS please contact:

Paul A. Thompson, Ph.D.  
Department of Psychiatry, Laboratory of Biological Psychiatry  
Case Western Reserve University  
Cleveland, OH 44106  
Phone: (216) 844-8946, Fax: (216) 844-5840

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brands and product names are registered trademarks or trademarks of their respective companies.