Research and Statistical Support
University of North Texas

RSS Matters

An Introduction to the Percentile Bootstrap Using GNU S

By Dr. Rich Herrington, Research and Statistical Support Consultant

Last month we looked at robust measures of location, this month we demonstrate the Percentile Bootstrap using the GNU S language, "R". R is a statistical programming environment that is a clone of the S and S-Plus language developed at Lucent Technologies. In the following document we illustrate the use of a GNU Web interface to the R engine on the "rss" server, http://rss.acs.unt.edu/cgi-bin/R/Rprog. This GNU Web interface is a derivative of the "Rcgi" Perl scripts available for download from the CRAN Website, http://www.cran.r-project.org (the main "R" Website). Scripts can be submitted interactively, edited, and be re-submitted with changed parameters by selecting the hypertext link buttons that appear below the figures. For example, clicking the "Run Program" button below samples 100 random numbers from a normal distribution, then generates a histogram. To view any text output, scroll to the bottom of the browser window. To view the histogram, select the "Display Graphic" link. The script can be edited and resubmitted by changing the script in the form window and then selecting "Run the R Program". Selecting the browser "back page" button will return the reader to this document.

```
x<-rnorm(100)
hist(x)
```

Run Program

The Bootstrap

Bootstrapping is an approach to statistical inference that makes few assumptions about the underlying probability distribution that describes the data (Efron, 1993). This approach assumes that the empirical cumulative distribution function is a reasonable estimate of the unknown, population cumulative distribution function (or put another way, the empirical density function approximates the population density function). Using the data as an approximation to the population density function, data is re-sampled with replacement from the observed sample to create an empirical sampling distribution for the test statistic under consideration. If we knew the population density function (e.g. normal probability distribution function), we could just sample from this probability density function, as in a Monte Carlo simulation, and generate the sampling distribution to within any degree of precision (or, we could make a normality assumption regarding the

population). Lacking this information then, we assume that the observed data is a good estimate of the population density function. The empirical cumulative distribution function of the observed data is a step function, taking jumps of height 1/n at each of the sample points. As n increases, the function becomes smooth, and looks more like the population cumulative distribution function. As n increase to infinity, the observed cumulative distribution function converges to the population cumulative distribution function. Now, using the observed data sample as a proxy population, we resample with replacement to produce another data set whose length is equal to the length of the original observed data sample. This re-sampling scheme is known as bootstrap re-sampling, and the re-sampled data sets are known as bootstrap samples. In this re-sampling scheme, bootstrap samples may contain duplicate copies of the original data points in the new sample. For example, an observed data set (our proxy population) that consists of the data points: 4,3,5,6,7,2,2,1,7,9 (mean=4.6), could produce the following data with re-sampling (with replacement): 2,4,3,6,1,7,9,5,5,1 (mean=4.3). For each bootstrap sample generated, we would calculate a statistic (e.g. location parameters: mean, median, M-estimator, etc) that is of concern. The empirical distribution of these calculated statistics is referred to as an empirical sampling distribution. This empirical sampling distribution can be used as an approximation to the theoretical population sampling distribution. To calculate scores that correspond the 2.5^{th} and 97.5^{th} confidence intervals, we simply find the scores that correspond to the upper and lower α and (1- α) percentiles of the distribution. For example, for 1000 bootstrap samples of the sample mean, we can calculate confidence intervals by finding the upper and lower 2.5^{th} and 97.5^{th} percentiles using the following approach: round((.05/2)x(1000))=25 for lower percentile; and round((1-(.05/2))x(1000))=975. Moreover, the standard deviation of the bootstrapped test statistics is an approximation of the standard error of the test statistic under consideration. This approach to calculating bootstrap confidence intervals is called the Percentile Bootstrap method (Efron, 1993). The bootstrap distribution, as outlined above is centered around the estimated population value of the test statistic. Other variations of bootstrap re-sampling center the bootstrap distribution around zero, depending on how the null hypothesis is being tested.

Bootstrapping the Difference of M-estimators

Doksum & Sievers (1976) report data on a study designed to assess the effects of ozone on weight gain in rats. The experimental group consisted of 22 seventy-day old rats kept in an ozone environment for 7 days. The control group consisted of 23 rats of the same age, and were kept in an ozone-free environment. Weight gain is measured in grams. The following GNU S code (R code) assigns the data to the vectors x and y and then plots the quantile-quantile plots for the normal distribution for each group. Notice that each group has heavy tails, and do not conform to a normal distribution.

```
### Control

x<-c(41.0,38.4,24.4,25.9,21.9,18.3,13.1,27.3,
      28.5,-16.9,26.0,17.4,21.8,15.4,27.4,19.2,
      22.4,17.7,26.0,29.4,21.4,26.6,22.7)

### Experimental

y<-c(10.1,6.1,20.4,7.3,14.3,15.5,-9.9,6.8,
      28.2,17.9,-9.0,-12.9,14.0,6.6,12.1,15.7,
      39.9,-15.9,54.6,-14.7,44.1,-9.0)

#Library for exploratory data analysis

library(eda)
summary(x)
summary(y)

#Set graphics window for two plots per page

par(mfrow=c(2,1))

#Plot qqplots for each group

qqnorm(x)
qqline(x)
qqnorm(y)
qqline(y)
```

Run Program

The Algorithm

Step 1: Generate the bootstrap alternative distribution. A) Re-sample with replacement from vector x with replacement to generate a bootstrap sample, x1, with length of original vector x. **B)** Re-sample with replacement from vector y with replacement to generate a bootstrap sample, y1, with length of original vector y. **C)** Calculate measures of location for both bootstrap samples x1 and y1. **D)** Subtract the two measures of location. This is one bootstrap difference, and represents the difference between measures of location under the empirical alternate distribution. This empirical distribution is centered on the population difference under the alternate hypothesis. **Step 2: Calculate the critical scores that correspond to the 2.5^{th} and 97.5^{th} critical alpha regions under the empirical alternate distribution**. The critical scores are the scores that correspond to the 2.5^{th} and 97.5^{th} percentiles of the empirical alternate distribution. We can calculate the percentiles using the following approach: round((.05/2)x(#bootstrap samples)) for lower percentile; and round((1-(.05/2))x(#bootstrap samples)). Next, locate the scores that correspond to those

percentiles. The following GNU S code implements this algorithm:

```
N1<-length(x)
N2<-length(y)

#Number of bootstrap samples

nboot<-399

#Alpha criterion

alpha<-.05

#Non-directional test

empTAIL<-2

#Sample with replacement from x and y data vector
#nboot times for sample size of N1 and N2

h1datax <-matrix(sample(x, size=N1 * nboot, replace=T), nrow=nboot)
h1datay <- matrix(sample(y, size=N2 * nboot, replace=T), nrow=nboot)

#Apply M-estimator to each data vector and
#accumulate in new data vectors

h1bvecx <- apply(h1datax, 1, est, k)
h1bvecy <- apply(h1datay, 1, est, k)

#Subtract Bootstrap M-estimate data vectors and sort;
#Accumulate into a M-estimator difference vector;

h1bvec <- sort(h1bvecx-h1bvecy)

#Mean of all Bootstrap M-estimator differences

mean(h1bvec)

#Calculate the upper and lower cuttoff percentiles
#for the lower and upper alpha criterion

effectlow <- round((alpha/empTAIL)*nboot)
effectup <- round((1-alpha/empTAIL)*nboot)

#Display M-estimate differences that correspond to the
#upper and lower alpha criterion cuttoffs

h1.ci<-list(ci=c(h1bvec[effectlow], h1bvec[effectup]))
h1.ci
```

Run Program

The 95% confidence interval that is reported represents our best estimate for the lower and upper bounds of the difference in M-estimators for the two groups. This interval captures the true population difference in M-estimators 95% of the time assuming that the null hypothesis is true (no difference between the groups). If this interval contains zero, we fail to reject the null hypothesis of no difference in the population. If the interval doesn't contain zero, we take this as a rejection of no difference in the population. Additionally, the narrower the confidence interval the more precise our estimate is (less error in our estimation).

A Cautionary Note on Computer Intensive Procedures

The Bootstrap re-sampling process is a very computer intensive procedure. Additionally, the iteratively re-weighted least squares algorithm that the M-estimator procedure uses is also computer intensive. Run times for the GNU S code may take a few minutes to run. Allowing the program to finishes before resubmitting will keep the GNU S server from slowing down. In general, having larger numbers of bootstrap samples increases the accuracy of the confidence intervals. However, the cost of increasing the number of bootstrap samples can be substantial.

References

Doksum, K.A. & Sievers, G.L. (1976). Plotting with confidence: graphical comparisons of two populations. Biometrika 63, 421-434.

Efron, B. (1993). An Introduction to the Bootstrap. New York: Chapman and Hall.