Page One

Campus Computing News

Update on SmartForce CBT

Lab-of-the-Month: SCS

FrontPage and Dreamweaver

WebDAV and You

Your First ColdFusion Application

Today's Cartoon

RSS Matters

SAS Corner

The Network Connection

List of the Month

WWW@UNT.EDU

Short Courses

IRC News

Staff Activities

Subscribe to Benchmarks *Online*

Research and Statistical Support
University of North Texas

RSS Matters

Using the Bootstrap with Small Data Sets: The Smoothed Bootstrap

By Dr. Rich Herrington, Research and Statistical Support Consultant

Last month we examined using the bootstrap and robust estimation to calculate statistical power, this month we explore the use of the smoothed bootstrap with small data sets. The GNU S language, "R" is used to implement this procedure. 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 1000 random numbers from a normal distribution, then uses nonparametric density estimation to fit a density curve to the data. To view any text output, scroll to the bottom of the browser window. To view the density curve, 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(1000)
rnorm(1000)
summary(x)
plot(density(x))
```

Run Program

The Disadvantages of Using Small Sample Sizes with the Bootstrap

In the nonparametric bootstrap, samples are drawn from a discrete set. This can be a serious disadvantage in small sample sizes in that spurious fine structure, in the original data, may be faithfully reproduced in the simulated data that has not occurred in the population. Difficulties can arise if the goal of the simulation is to produce samples that have the underlying “true” structure of the observed data without having spurious details arise from random effects. Another concern is that with small samples, with only a few values to select from, the bootstrap samples will underestimate the true variability. Statisticians generally regard the use of the bootstrap with sample sizes less than 10 as too small to rely on (Chernick, 1999).

The Smoothed Bootstrap

One approach to dealing with the discreteness of the empirical distribution function with small sample sizes, is to smooth the empirical distribuion function and then resample from the smoothed empirical distribution function. It has been shown that the nonparametric bootstrap is improved in non-smooth cases, such as the median (Fernholz, 1993). Even though the "smoothed bootstrap" was considered early on by bootstrap researchers, there was little evidence to indicate under which conditions smoothing would be beneficial (Hall P., DiCiccio, T. & Romano, J, 1989; Silverman, B.W., & Young, G.A., 1987). Recent research on the smoothed bootstrap demonstrates that for samll sample sizes, with proper kernel bandwith selection, smoothing the empirical distribution function can yeild a first-order reduction in coverage error for the one-sided percentile method. The one-sided percentile method, based on the smoothed bootstrap with an optimally chosen bandwith, becomes asymptotically as accurate as either the bootstrap t or the accelerated bias correction (Bca) methods (Polansky, A.M. & Schucany, W.R., 1997). Similar arguments show that second-order corrections can be realized for first-order correct confidence intervals such as the two-sided percentile method intervals and bootstrap t intervals (Polansky, 2001). The smoothed bootstrap can also decrease coverage error for finite samples as well (Polansky, 2000). That is, type I error for small sample sizes can be reduced by smoothing the empirical distribution function, when using the Percentile Bootstrap method for calculating confidence intervals. It is important for the present study, to note that Fernholz (1993, 1997) proved that by smoothing the empirical distribution function with an appropriate kernel, the variance and the mean square error of certain statistical functionals can be reduced. A functional is a mapping that assigns a real value to a function. Examples of functionals are the parameters of distribution functions, including the mean, the variance, the skewness and the kurtosis of the distribution. Other examples include sample quantiles, some L-estimators, and M-estimators (Fernholz, 1997). Specifically, Fernholz demonstrates that a smaller variance is achieved when the influence function is either discontinuous (such as in the median) or piecewise linear with convexity towards the x-axis (such as in the Huber and biweight type M-estimators). Essentially, the smoothed bootstrap can be used to improve overall performance (decrease bias, MSE of estimators) in small sample sizes. Brown, Hall, and Young (2001) show that for the median, that smoothing increases efficiency for normal data over that of the conventional median. The algorithm of the smoothed bootstrap is outlined in Silverman, B.W. (1986, page 141). The basic idea is to set:

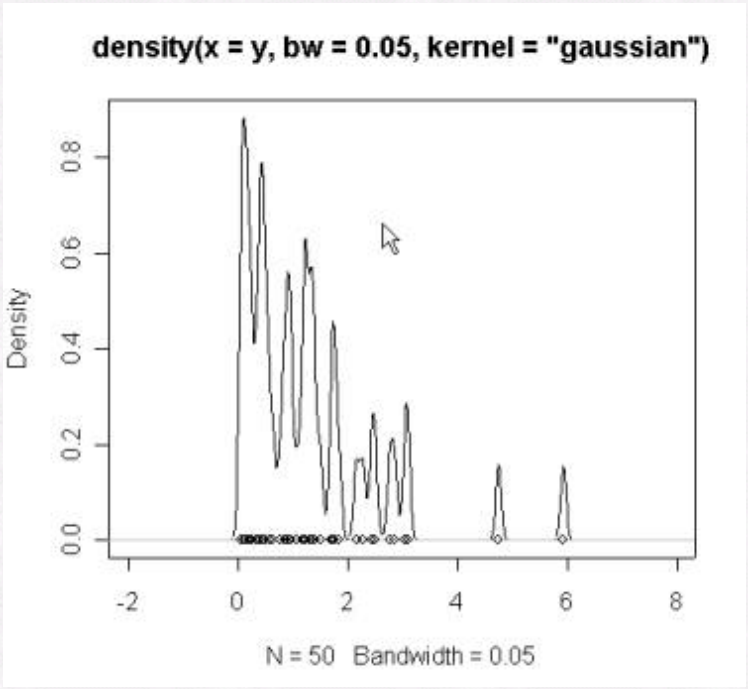
$$Y = X^{*} + h\varepsilon$$

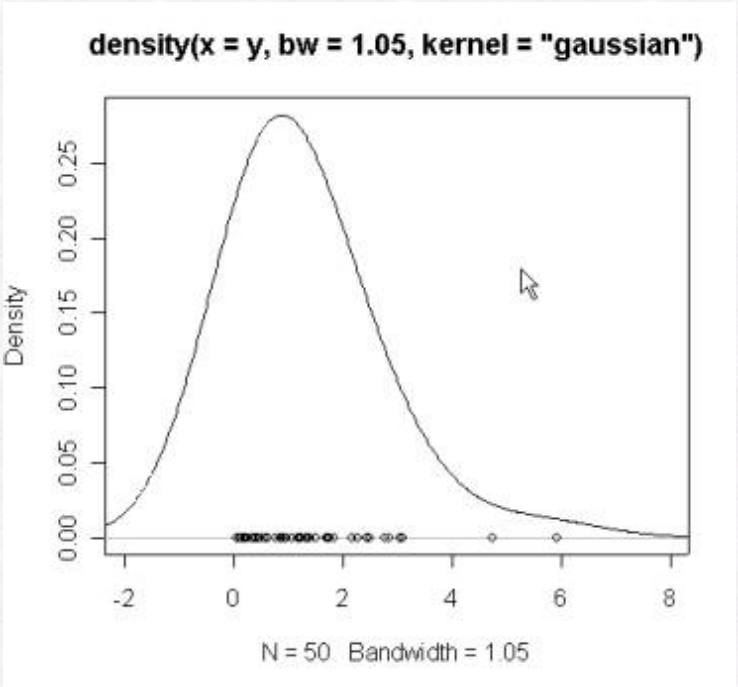
where each X^{*} consist of the bootstrap observations from a bootstrap sample; ε is a random deviate from a probability density function K; and h is a smoothing parameter that can be calculated from the sample moments (e.g. standard deviation). The probability density K is often referred to as the "kernel". A natural candidate for the kernel is the Gaussian distribution (normal distribution). If a Gaussian kernel is selected, then an optimal smoothing parameter can be estimated from the data (a so-called "plug-in" estimate of h):

$$h_{opt} = 1.06 \cdot \sigma \cdot n^{-1/5}$$

where σ is estimated from the sample data. h will work well if the population is normally distributed, but it may oversmooth if the population is either multi-modal or skewed (the sample estimate of σ is not a resistant measure). Silverman (1986, page 46) reports that for heavily skewed data, h will oversmooth, but that the formula is remarkably insensitive to kurtosis within the t family of distributions. To give some sense of how the smoothing operation effects a skew distribution, The figures below, show an exponential distribution of sample size 50. The

top panel displays little or no smoothing, and the bottom panel displays much greater smoothing. It is evident that most of the skewness in the original data set is gone, and tends toward a symmetric distribution resembling a normal distribution. However, if the point is to only smooth local irregularities, but retain the overall shape of the distribution, oversmoothing will mis-represent the underlying population distribution.





The Smoothed Bootstrap Implemented in the "S" Language

The following S code set implements the smoothed bootstrap using two different kernels (*K*), and with various different window estimators (*h*). Twenty numbers are sampled from the normal distribution; this becomes our population that we will resample from. The following code chooses a Gaussian kernel with a standard smoothing parameter for the smoothed bootstrap. The population mean is estimated using the sample mean. One might change the following code

to explore the effects of: 1) using different sample sizes, 2) using different numbers of bootstrap samples; 3) using either the variance corrected or uncorrected versions of the kernels, 3) using robust estimators with the smoothed bootstrap, rather than the mean, and 4) using different window estimators combined with different kernels.

```
#### Set  random number seed
set.seed(98)

#### Simulate 20 numbers from normal distribution
#       with mean zero and variance one
xdata<-rnorm(20)

###Set sample size to draw from smoothed
#     bootstrap
samp.size<-500

### Set number of bootstrap samples to draw
num.boot.samp<-500

### Windowing functions for kernel density estimate
h.window<-function(x, samp.size, window=1){
 if (window==0) {wind.weight<-0}
 if (window==1) {stdev<-var(x)^.5
  wind.weight<-(1.06)*stdev*samp.size^(-1/5)}
 if (window==2) {quar <- diff(quantile(x, c(.25, .75)))
  wind.weight<-(.79)*quar*samp.size^(-1/5)}
 if (window==3) {stdev<-var(x)^.5
  skew<-sum((x-mean(x))^3)/samp.size
  wind.weight<-(((abs(skew)*(1.64^2+2))/(3*stdev*1.64))^.5)^(-1/4)}
 if (window==4) {wind.weight<-samp.size^(-1/2)}
 if (window==5) {stdev<-var(x)^.5
  wind.weight<-(samp.size)^(-3/4)*(stdev)*(2^(1/2))}
 if (window==6){stdev <- var(x)^0.5
  wind.weight <- (1.41)*stdev*samp.size^(-3/4)}
 if (window==7){stdev <- var(x)^0.5
  quar <- diff(quantile(x, c(.25, .75)))
  wind.weight <- min(stdev,(.90)*(quar/1.34)*samp.size^(-1/5))}
return(wind.weight)
}

### Simple Bootstrap, no smoothing
rdensity<-function(orig.data, samp.size, num.boot.samp, window){
 x<-orig.data # orig.data
 x.length<-samp.size*num.boot.samp
 x.unif<-matrix(sample(x, size=x.length, replace=T), nrow=num.boot.samp)
 y<-x.unif
return(y)}

### Corrected estimates for variance of estimated distribution.
#     Correcting for size of windowing function

### Variance corrected estimates of
#    the nonparametric density smoothing
#    parameter - h

#### Epanechnikov kernel - variance corrected
rdensity2<-function(orig.data, samp.size, num.boot.samp, window){
 x<-orig.data # orig.data
 x.length<-samp.size*num.boot.samp
 x.unif<-matrix(sample(x, size=x.length, replace=T), nrow=num.boot.samp)
 x.unif.mean<-apply(x.unif, 1, mean)
 x.unif.mean<-matrix(x.unif.mean, ncol=samp.size,
  nrow=num.boot.samp, byrow=F)
 x.unif.var<-apply(x.unif, 1, var)
 x.h.1<-apply(x.unif, 1, h.window, samp.size, window)
 x.h<-matrix(x.h.1, ncol=samp.size, nrow=num.boot.samp, byrow=F)
 x.kern<-matrix(runif(x.length*3, -1, 1), ncol=3)
 epsilon<-ifelse((abs(x.kern[,3]) >= abs(x.kern[,2])) &
  (abs(x.kern[,3])>= abs(x.kern[,1])), x.kern[,2], x.kern[,3])
 epsilon<-matrix(epsilon, nrow=num.boot.samp)
 kern.var<-.20
 correct.var<-matrix(1+x.h.1^2*(kern.var/x.unif.var)^.5,
  ncol=samp.size, nrow=num.boot.samp, byrow=F)
 y<-x.unif.mean+(x.unif-x.unif.mean+x.h*epsilon)/correct.var
return(y)}

### Gaussian Kernel - variance corrected
rdensity3<-function(orig.data, samp.size, num.boot.samp, window){
 x<-orig.data # orig.data
 x.length<-samp.size*num.boot.samp
 x.unif<-matrix(sample(x, size=x.length, replace=T), nrow=num.boot.samp)
 x.unif.mean<-apply(x.unif, 1, mean)
 x.unif.mean<-matrix(x.unif.mean, ncol=samp.size,
  nrow=num.boot.samp, byrow=F)
 x.unif.var<-apply(x.unif, 1, var)
 x.h.1<-apply(x.unif, 1, h.window, samp.size, window)
 x.h<-matrix(x.h.1, ncol=samp.size, nrow=num.boot.samp, byrow=F)
 x.kern<-matrix(runif(x.length*3, -1, 1), ncol=3)
 epsilon<-matrix(norm(x.length), nrow=num.boot.samp)
 kern.var<-1
 correct.var<-matrix(1+x.h.1^2*(kern.var/x.unif.var)^.5,
  ncol=samp.size, nrow=num.boot.samp, byrow=F)
 y<-x.unif.mean+(x.unif-x.unif.mean+x.h*epsilon)/correct.var
return(y)}

### No variance correction kernels

### No variance correction, Epanechnikov kernel
rdensity4<-function(orig.data, samp.size, num.boot.samp, window){
 x<-orig.data # orig.data
 x.length<-samp.size*num.boot.samp
 x.unif<-matrix(sample(x, size=x.length, replace=T),
  nrow=num.boot.samp)
 x.h<-apply(x.unif, 1, h.window, samp.size, window)
 x.h<-matrix(x.h, ncol=samp.size, nrow=num.boot.samp, byrow=F)
 x.kern<-matrix(runif(x.length*3, -1, 1), ncol=3)
 epsilon<-ifelse((abs(x.kern[,3]) >= abs(x.kern[,2]))
  & (abs(x.kern[,3]) >= abs(x.kern[,1])),x.kern[,2],
  x.kern[,3])
 epsilon<-matrix(epsilon, nrow=num.boot.samp)
 y<-x.unif+x.h*epsilon
return(y)}

### No variance correction, Gaussian kernel
rdensity5<-function(orig.data, samp.size, num.boot.samp, window){
 x<-orig.data # orig.data
 x.length<-samp.size*num.boot.samp
 x.unif<-matrix(sample(x, size=x.length, replace=T),
  nrow=num.boot.samp)
 x.h<-apply(x.unif, 1, h.window, samp.size, window)
 x.h<-matrix(x.h, ncol=samp.size, nrow=num.boot.samp, byrow=F)
 epsilon<-matrix(rnorm(x.length), nrow=num.boot.samp)
 y<-x.unif+x.h*epsilon
return(y)}

### Sample from smoothed bootstrap using
#     variance corrected Gaussian kernel with
#     standard window
x.new.boot<-rdensity3(orig.data=xdata, samp.size,
  num.boot.samp, window=1)

x.new.result<-apply(x.new.boot, 1, mean)
x.new2.result<-apply(x.new.boot, 1, var)

### Mean and variance of the 500 bootstrap means
x.mean.500<-mean(x.new.result)
x.var.500<-mean(x.new2.result)

### Pick a particular bootstrap sample to plot
#     against the original sample
x.new<-x.new.boot[200,]

### Plot graphics on 2 by 2 grid
par(mfrow=c(2,2))

### Attach library of robust regression and
#     covariance methods
library(lqs)

xdata.qq<-qqplot(xdata, x.new, plot=F)
plot(xdata.qq)
abline(lmsreg(xdata.qq$x, xdata.qq$y))
```

```
#### Set  random number seed
set.seed(98)

#### Simulate 20 numbers from normal distribution
#       with mean zero and variance one
xdata<-rnorm(20)

###Set sample size to draw from smoothed
#     bootstrap
samp.size<-500

### Set number of bootstrap samples to draw
num.boot.samp<-500

### Windowing functions for kernel density estimate
h.window<-function(x, samp.size, window=1){
  if (window==0) {wind.weight<-0}
  if (window==1) {stdev<-var(x)^.5
    wind.weight<-(1.06)*stdev*samp.size^(-1/5)}
  if (window==2) {quar <- diff(quantile(x, c(.25, .75)))
    wind.weight<-(.79)*quar*samp.size^(-1/5)}
  if (window==3) {stdev<-var(x)^.5
    skew<-sum((x-mean(x))^3)/samp.size
    wind.weight<-((abs(skew)*(1.64^2+2))/(3*stdev*1.64))^.5)^(-1/4)}
  if (window==4) {wind.weight<-samp.size^(-1/2)}
  if (window==5) {stdev<-var(x)^.5
    wind.weight<-(samp.size)^(-3/4)*(stdev)*(2^(1/2))}
  if (window==6){stdev <- var(x)^0.5
    wind.weight <- (1.41)*stdev*samp.size^(-3/4))
  if (window==7){stdev <- var(x)^0.5
    quar <- diff(quantile(x, c(.25, .75)))
    wind.weight <- min(stdev,(.90)*(quar/1.34)*samp.size^(-1/5))}
  return(wind.weight)
  }

### Simple Bootstrap, no smoothing
rdensity<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T), nrow=num.boot.samp)
  y<-x.unif
  return(y)}

### Corrected estimates for variance of estimated distribution.
#     Correcting for size of windowing function

### Variance corrected estimates of
#     the nonparametric density smoothing
#     parameter - h

#### Epanechnikov kernel - variance corrected
rdensity2<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T), nrow=num.boot.samp)
  x.unif.mean<-apply(x.unif, 1, mean)
  x.unif.mean<-matrix(x.unif.mean, ncol=samp.size,
    nrow=num.boot.samp, byrow=F)
  x.unif.var<-apply(x.unif, 1, var)
  x.h.1<-apply(x.unif, 1, h.window, samp.size, window)
  x.h<-matrix(x.h.1, ncol=samp.size, nrow=num.boot.samp, byrow=F)
  x.kern<-matrix(runif(x.length*3, -1, 1), ncol=3)
  epsilon<-ifelse((abs(x.kern[,3]) >= abs(x.kern[,2])) &
    (abs(x.kern[,3])>= abs(x.kern[,1])), x.kern[,2], x.kern[,3])
  epsilon<-matrix(epsilon, nrow=num.boot.samp)
  kern.var<-.20
  correct.var<-matrix(1+x.h.1^2*(kern.var/x.unif.var)^.5,
    ncol=samp.size, nrow=num.boot.samp, byrow=F)
  y<-x.unif.mean+(x.unif-x.unif.mean+x.h*epsilon)/correct.var
  return(y)}

### Gaussian Kernel - variance corrected
rdensity3<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T), nrow=num.boot.samp)
  x.unif.mean<-apply(x.unif, 1, mean)
  x.unif.mean<-matrix(x.unif.mean, ncol=samp.size,
    nrow=num.boot.samp, byrow=F)
  x.unif.var<-apply(x.unif, 1, var)
  x.h.1<-apply(x.unif, 1, h.window, samp.size, window)
  x.h<-matrix(x.h.1, ncol=samp.size, nrow=num.boot.samp, byrow=F)
  x.kern<-matrix(runif(x.length*3, -1, 1), ncol=3)
  epsilon<-matrix(norm(x.length), nrow=num.boot.samp)
  kern.var<-1
  correct.var<-matrix(1+x.h.1^2*(kern.var/x.unif.var)^.5,
    ncol=samp.size, nrow=num.boot.samp, byrow=F)
  y<-x.unif.mean+(x.unif-x.unif.mean+x.h*epsilon)/correct.var
  return(y)}

### No variance correction kernels

### No variance correction, Epanechnikov kernel
rdensity4<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T),
    nrow=num.boot.samp)
  x.h<-apply(x.unif, 1, h.window, samp.size, window)
  x.h<-matrix(x.h, ncol=samp.size, nrow=num.boot.samp, byrow=F)
  x.kern<-matrix(runif(x.length*3, -1, 1), ncol=3)
  epsilon<-ifelse((abs(x.kern[,3]) >= abs(x.kern[,2]))
    & (abs(x.kern[,3]) >= abs(x.kern[,1])),x.kern[,2],
    x.kern[,3])
  epsilon<-matrix(epsilon, nrow=num.boot.samp)
  y<-x.unif+x.h*epsilon
  return(y)}

### No variance correction, Gaussian kernel
rdensity5<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T),
    nrow=num.boot.samp)
  x.h<-apply(x.unif, 1, h.window, samp.size, window)
  x.h<-matrix(x.h, ncol=samp.size, nrow=num.boot.samp, byrow=F)
  epsilon<-matrix(rnorm(x.length), nrow=num.boot.samp)
  y<-x.unif+x.h*epsilon
  return(y)}

### Sample from smoothed bootstrap using
#       variance corrected Gaussian kernel with
#       standard window
x.new.boot<-rdensity3(orig.data=xdata, samp.size,
    num.boot.samp, window=1)

x.new.result<-apply(x.new.boot, 1, mean)
x.new2.result<-apply(x.new.boot, 1, var)

### Mean and variance of the 500 bootstrap means
x.mean.500<-mean(x.new.result)
x.var.500<-mean(x.new2.result)

### Pick a particular bootstrap sample to plot
#     against the original sample
x.new<-x.new.boot[200,]

### Plot graphics on 2 by 2 grid
par(mfrow=c(2,2))

### Attach library of robust regression and
#     covariance methods
library(lqs)

xdata.qq<-qqplot(xdata, x.new, plot=F)
plot(xdata.qq)
abline(lmsreg(xdata.qq$x, xdata.qq$y))
hist(xdata)
hist(x.new)

### The average Mean and Variance of all 500 bootstrap samples
x.mean.500
x.var.500

### Summarize original data
length(xdata)
summary(xdata)
var(xdata)

### Summarize resampled data from smoothed
#   bootstrap
length(x.new)
```

```
#### Set  random number seed
set.seed(98)

#### Simulate 20 numbers from normal distribution
#       with mean zero and variance one
xdata<-rnorm(20)

###Set sample size to draw from smoothed
#     bootstrap
samp.size<-500

### Set number of bootstrap samples to draw
num.boot.samp<-500

### Windowing functions for kernel density estimate
h.window<-function(x, samp.size, window=1){
  if (window==0) {wind.weight<-0}
  if (window==1) {stdev<-var(x)^.5
    wind.weight<-(1.06)*stdev*samp.size^(-1/5)}
  if (window==2) {quar <- diff(quantile(x, c(.25, .75)))
    wind.weight<-(.79)*quar*samp.size^(-1/5)}
  if (window==3) {stdev<-var(x)^.5
    skew<-sum((x-mean(x))^3)/samp.size
    wind.weight<-((abs(skew)*(1.64^2+2))/(3*stdev*1.64))^.5)^(-1/4)}
  if (window==4) {wind.weight<-samp.size^(-1/2)}
  if (window==5) {stdev<-var(x)^.5
    wind.weight<-(samp.size)^(-3/4)*(stdev)*(2^(1/2))}
  if (window==6){stdev <- var(x)^0.5
    wind.weight <- (1.41)*stdev*samp.size^(-3/4))
  if (window==7){stdev <- var(x)^0.5
    quar <- diff(quantile(x, c(.25, .75)))
    wind.weight <- min(stdev,(.90)*(quar/1.34)*samp.size^(-1/5))}
  return(wind.weight)
  }

### Simple Bootstrap, no smoothing
rdensity<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T), nrow=num.boot.samp)
  y<-x.unif
  return(y)}

### Corrected estimates for variance of estimated distribution.
#       Correcting for size of windowing function

### Variance corrected estimates of
#     the nonparametric density smoothing
#     parameter - h

#### Epanechnikov kernel - variance corrected
rdensity2<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T), nrow=num.boot.samp)
  x.unif.mean<-apply(x.unif, 1, mean)
  x.unif.mean<-matrix(x.unif.mean, ncol=samp.size,
    nrow=num.boot.samp, byrow=F)
  x.unif.var<-apply(x.unif, 1, var)
  x.h.1<-apply(x.unif, 1, h.window, samp.size, window)
  x.h<-matrix(x.h.1, ncol=samp.size, nrow=num.boot.samp, byrow=F)
  x.kern<-matrix(runif(x.length*3, -1, 1), ncol=3)
  epsilon<-ifelse((abs(x.kern[,3]) >= abs(x.kern[,2])) &
    (abs(x.kern[,3])>= abs(x.kern[,1])), x.kern[,2], x.kern[,3])
  epsilon<-matrix(epsilon, nrow=num.boot.samp)
  kern.var<-.20
  correct.var<-matrix(1+x.h.1^2*(kern.var/x.unif.var)^.5,
    ncol=samp.size, nrow=num.boot.samp, byrow=F)
  y<-x.unif.mean+(x.unif-x.unif.mean+x.h*epsilon)/correct.var
  return(y)}

### Gaussian Kernel - variance corrected
rdensity3<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T), nrow=num.boot.samp)
  x.unif.mean<-apply(x.unif, 1, mean)
  x.unif.mean<-matrix(x.unif.mean, ncol=samp.size,
    nrow=num.boot.samp, byrow=F)
  x.unif.var<-apply(x.unif, 1, var)
  x.h.1<-apply(x.unif, 1, h.window, samp.size, window)
  x.h<-matrix(x.h.1, ncol=samp.size, nrow=num.boot.samp, byrow=F)
  x.kern<-matrix(runif(x.length*3, -1, 1), ncol=3)
  epsilon<-matrix(norm(x.length), nrow=num.boot.samp)
  kern.var<-1
  correct.var<-matrix(1+x.h.1^2*(kern.var/x.unif.var)^.5,
    ncol=samp.size, nrow=num.boot.samp, byrow=F)
  y<-x.unif.mean+(x.unif-x.unif.mean+x.h*epsilon)/correct.var
  return(y)}

### No variance correction kernels

### No variance correction, Epanechnikov kernel
rdensity4<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T),
    nrow=num.boot.samp)
  x.h<-apply(x.unif, 1, h.window, samp.size, window)
  x.h<-matrix(x.h, ncol=samp.size, nrow=num.boot.samp, byrow=F)
  x.kern<-matrix(runif(x.length*3, -1, 1), ncol=3)
  epsilon<-ifelse((abs(x.kern[,3]) >= abs(x.kern[,2]))
    & (abs(x.kern[,3]) >= abs(x.kern[,1])),x.kern[,2],
    x.kern[,3])
  epsilon<-matrix(epsilon, nrow=num.boot.samp)
  y<-x.unif+x.h*epsilon
  return(y)}

### No variance correction, Gaussian kernel
rdensity5<-function(orig.data, samp.size, num.boot.samp, window){
  x<-orig.data # orig.data
  x.length<-samp.size*num.boot.samp
  x.unif<-matrix(sample(x, size=x.length, replace=T),
    nrow=num.boot.samp)
  x.h<-apply(x.unif, 1, h.window, samp.size, window)
  x.h<-matrix(x.h, ncol=samp.size, nrow=num.boot.samp, byrow=F)
  epsilon<-matrix(rnorm(x.length), nrow=num.boot.samp)
  y<-x.unif+x.h*epsilon
  return(y)}

### Sample from smoothed bootstrap using
#       variance corrected Gaussian kernel with
#       standard window
x.new.boot<-rdensity3(orig.data=xdata, samp.size,
    num.boot.samp, window=1)

x.new.result<-apply(x.new.boot, 1, mean)
x.new2.result<-apply(x.new.boot, 1, var)

### Mean and variance of the 500 bootstrap means
x.mean.500<-mean(x.new.result)
x.var.500<-mean(x.new2.result)

### Pick a particular bootstrap sample to plot
#     against the original sample
x.new<-x.new.boot[200,]

### Plot graphics on 2 by 2 grid
par(mfrow=c(2,2))

### Attach library of robust regression and
#     covariance methods
library(lqs)

xdata.qq<-qqplot(xdata, x.new, plot=F)
plot(xdata.qq)
abline(lmsreg(xdata.qq$x, xdata.qq$y))
hist(xdata)
hist(x.new)

### The average Mean and Variance of all 500 bootstrap samples
x.mean.500
x.var.500

### Summarize original data
length(xdata)
summary(xdata)
var(xdata)

### Summarize resampled data from smoothed
#   bootstrap
length(x.new)
```

```
#### Set  random number seed
set.seed(98)

#### Simulate 20 numbers from normal distribution
#       with mean zero and variance one
xdata<-rnorm(20)

###Set sample size to draw from smoothed
#     bootstrap
samp.size<-500

### Set number of bootstrap samples to draw
num.boot.samp<-500

### Windowing functions for kernel density estimate
h.window<-function(x, samp.size, window=1){
  if (window==0) {wind.weight<-0}
  if (window==1) {stdev<-var(x)^.5
```

Run Program

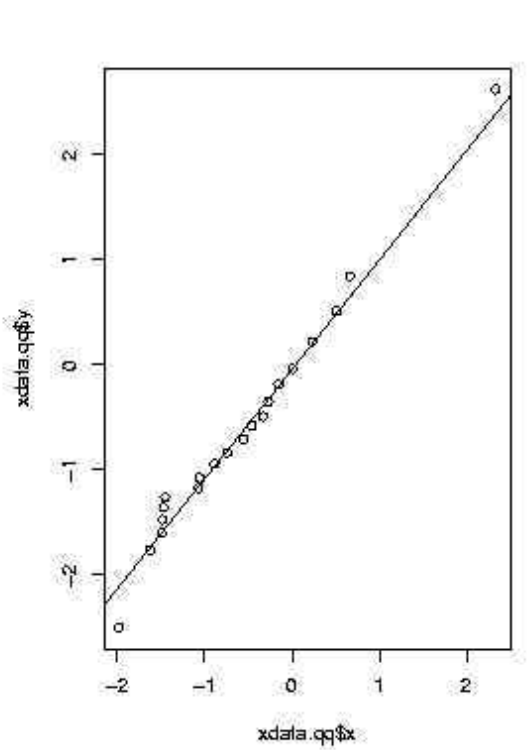
Results

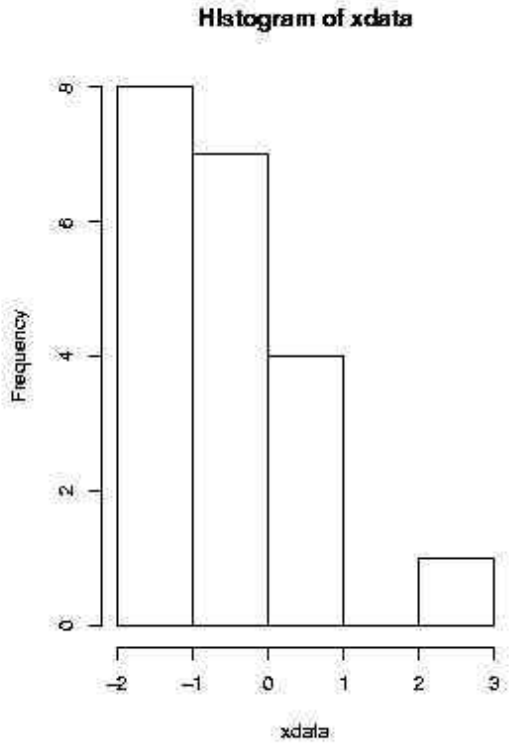
Running the S code listed above produces the following text and graphics:

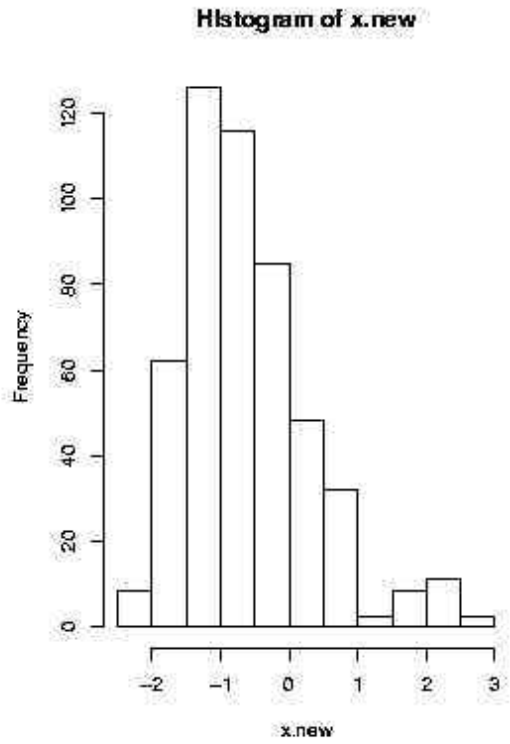
```
> # The average Mean and Variance of all 500 bootstrap samples
>
> x.mean.500
[1] -0.5682712
> x.var.500
[1] 0.8848028
>
> # Summarize original data
>
> length(xdata)
[1] 20
> summary(xdata)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.9820 -1.4540 -0.6479 -0.5646 -0.1187  2.3320
> var(xdata)
[1] 1.020477
>
> # Summarize resampled data from smoothed
> # bootstrap
>
> length(x.new)
[1] 500
> summary(x.new)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.4970 -1.2920 -0.7711 -0.6230 -0.1672  2.6220
> var(x.new)
[1] 0.852574
```

The preceeding code uses the variance adjusted Gaussian kernel to smooth the empirical distribution function (rdensity3). We see that the average mean of the 500 bootstrap samples is virtually identical to the original sample: .568. The average variance of the 500 bootstrap samples, underestimates the original sample variance by a substantial amount: .885 versus the original population variance of 1.00. The 200th bootstrap sample is extracted to plot against the original sample to see how well the shape of the 200th bootstrap sample, with an N=500, captures the shape of the original sample. As can be seen below, the larger sample size smooths the discreetness of the original sample while retaining the overall shape. These results also suggest that the variance corrected version of the Gaussian kernel "overadjusts" for the variance introduced by the smoothing parameter. Further simulation results using the unadjusted version of the Gaussian kernel (rdensity5) show that the unadjusted version does very well in recapturing the population variance. Try re-running the S code above, replacing the sample size with N=10 (use rnorm(10) instead of rnorm(20) in the beginning of the program), and the kernel density estimator with "rdensity5" instead of using "rdensity3" (x.new.boot<-

rdensity3(orig.data=xdata, samp.size,
num.boot.samp, window=1). Compare the mean and variance of the original data (xdata) with the mean and variance of all 500 bootstrap samples. How close are they compared to the results displayed above?







Conclusions

The bootstrap methodology allows the performance of classical and robust estimators to be evaluated in real world data sets. However, much still remains unknown about the finite sample properties of the bootstrap. In particular, small sample sizes can cause the bootstrap to fail, and give poor error coverage for type I errors, for a number of the more popular methods for calculating confidence intervals (Bca, studentized t bootstrap, and the percentile bootstrap). In contrast, the smoothed bootstrap holds promise for a number of robust estimators (median, L-estimators, M-estimators, quantile estimators), in small sample settings (i.e. approximately N< 15). However, it is evident that the proper selection of the smoothing parameter (*h*) is important so that oversmoothing or undersmoothing does not occur. Like robust estimators, smoothing the empirical distribution function can reduce the impact of heavy tails on a location estimatorl. Optimal selection of the smoothing parameter, *h* , is important so that undersmoothing or oversmoothing does not occur. Various approaches have been tried, for example, adaptive estimators (Silverman, 1986, page 48). Robust estimators such as M-estimators, optimally downweight the tails according to a statistical critierion (maximum likelihood) for a given set of tuning constants. Tuning constants are recommended that work well with a wide range of distributions found in real data (Hoaglin, Moesteller, & Tukey, 1983). The combined use of the smoothed bootstrap with an M-estimator as a location estimate, calls for an optimal combination of the tuning constants (e.g. *k*) for robust location, and the smoothing parameter (e.g. *h*) for the smoothed bootstrap. One computational approach towards this goal, would be to use various combinations of *h* and *k* , and choose the combination that produces the shortest possible confidence intervals while minimizing the coverage error under the null hypothesis . Used in this way, the parameters *h* and *k* become calibration coefficients (Polansky, 2001, page 822). Polansky (2001) reports theortetical results and simulation results that support this approach for the choice of *h* in small sample sizes (N<20). In summary, the bootstrap has become such a important tool, both theoretically and application-wise, that it has led Peter Hall, an emminent figure in the bootstrap research field, to comment, “The bootstrap has had a great impact on the practice of statistics, to the extent that the property of being bootstrappable might well be added to those of efficiency, robustness and ease of computation, as a fundamentally desirable property for statistical procedures in general” (Brown, 2001).

References

Brown, B.M., Hall, P., & Young, G.A. (2001). The smoothed median and the bootstrap. *Biometrika, 88(2),* 519-534.

Chernick, Michael, R. (1999). *Bootstrap Methods: A Practitioner' Guide,* John Wiley and Sons Inc.: New York.

Fernholz, L.T. (1993). Smoothed Versions of Statistical Functionals. In: *New Directions in Statistical Data Analysis and Robustness.* Eds: S. Morgenthaler, E. Ronchetti, W.A. Stahel. Birkhauser Verlag, Bosten.

Fernholz, L.T. (1997). Reducing the variance by smoothing. *Journal of Statistical Planning and Inference, 57,* 29-38.

Hall P., DiCiccio, T. & Romano, J. (1989). On smoothing and the bootstrap. *The Annals of Statistics, 2,* 692-704.

Hoaglin, D. C., Mosteller, F., & Tukey, J.W. (1983). *Understanding robust and exploratory data analysis.* New York: Wiley.

Polansky, A.M. (2001). Bandwidth selection for the smoothed bootstrap percentile method. *Computational Statistics and Data Analsysis, 36,* 333-349.

Polansky, A.M. and Schucany, W.R.(1997). Kernel Smoothing to Improve Bootstrap Confidence Intervals. *J.R. Statist. Soc. B, 59(4),* 821-838.

Polansky,A.M. (2000). Stabilizing bootstrap-t confidence intervals for small samples. *The Canadian Journal of Statistics, 28,* 501-516.

Silverman, B.W. (1986). *Density estimation for statistics and data analysis.* Chapman and Hall, London.

Silverman, B.W., & Young, G.A. (1987). The bootstrap: To smooth or not to smooth. *Biometrika*, 74, 469-479.