Research and Statistical Support
University of North Texas

# RSS Matters

## Dealing with Outliers in Bivariate Data: Robust Correlation

**By Dr. Rich Herrington, Research and Statistical Support Consultant**

Last time we examined the smoothed bootstrap, this month we demonstrate the calculation of a robust correlation measure. 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 creates 100 random numbers from a bivariate normal distribution; uses linear regression to fit a least squares line to two of the columns of data; and calculates person's product moment correlation. To view any text output, scroll to the bottom of the browser window. To view the scatterplot, 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.

```
# n=sample size; p=number of columns
# u=mean of columns; s=standard deviation
# of columns; S=correlation matrix

mvrnorm <- function(n, p, u, s, S) {
Z <- matrix(rnorm(n*p), p, n)
t(u + s*t(chol(S))%*% Z)
}

# Sample from Bivariate Normal

z<-mvrnorm(n=100, p=2, u=c(100,100), s=c(15,15), S=matrix(c(1, .4, .4, 1), ncol=2,
nrow=2,byrow=T))

# Fit least squares regression

z.fit<-lm(z[,1]~z[,2])
z.fit

# Plot scatterplot
plot(z[,1], z[,2])

# Plot best fit line
abline(z.fit)

# Calculate Pearson's Correlation
cor(z[,1], z[,2])
```

Run Program

# Pearson's Product Moment Correlation

Let $(X_1,Y_1),\ldots..(X_n,Y_n)$ be a random sample from a bivariate distribution. The sample estimate of the population correlation $\rho$ is given by:

$$r=\frac{\sum(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum(X_i-\bar{X})^2\sum(Y_i-\bar{Y})^2}}$$

If X and Y are independent, $\rho=0$. A test of $H_0:\rho=0$ is based on the test statistic:

$$T=r\sqrt{\frac{n-2}{1-r^2}}$$

where n is the number of X,Y pairs of scores. If $H_0$ is true, T has a Student's t distribution with df = n-2 degrees of freedom if at least one of the marginal distributions is normal. Reject $H_0:\rho=0$ if $|T|>t_{1-\alpha/2}$, for the $1-\alpha/2$ quantile of Student's t distribution with n-2

degrees of freedom. When the null hypothesis is true, the distribution of sample correlation coefficients tends to be normally distributed for increasing sample size. The standard error of this distribution of sample correlations is approximately:

$$\sigma_r = \frac{1}{\sqrt{N}}$$

When the sample size is reasonably large ($N \geq 50$), then it is possible to test the significance of the sample correlation coefficient by forming the usual z statistic and referring it to the normal distribution. In situations where one wishes to test for a non-zero value of the sample correlation coefficient, many sources have recommended Fisher's r-toZ transformation when computing confidence intervals, but it is not asymptotically correct when sampling from nonnormal distributions. Moreover, simulation studies do not support Fisher's r-to-Z transformation for small sample sizes. The S script below samples from a bivariate normal distribution whose population correlation is .40. A scatterplot is plotted with a best fit regression line added to the scatterplot. A classical t-test of the correlation coefficient is performed, and the standard error under the null hypothesis is calculated. Change the script below to have different sample sizes and see the effect of sample size on the standard error; also see how the sample correlation fluctuates from sample to sample around the true population correlation value. With smaller sample sizes, the sample correlation coefficient can be quite different from the population value.

```
mvrnorm <- function(n, p, u, s, S) {
Z <- matrix(rnorm(n * p), p, n)
t(u + s*t(chol(S))%*% Z)
}

z<-mvrnorm(n=100, p=2, u=c(100,100), s=c(15,15), S=matrix(c(1,.4, .4, 1), ncol=2,
nrow=2, byrow=T))

z.fit<-lm(z[,1]~z[,2])

plot(z[,1], z[,2])
abline(z.fit)

z.cor<-cor.test(z[,1], z[,2])
stderr<-1/(length(z[,1]))^.5
stderr
z.cor
```

Run Program

# Robust Correlation: The Biweight Midcorrelation

For an introduction to robust statistics and robust measures of location, see the July issue of Benchmarks. Let $(X_1, Y_1), \ldots (X_n, Y_n)$ be a random sample from a bivariate distribution. Following Wilcox (1997, page 197), let:

$$U_i = \frac{X_i - M_x}{9 \cdot MAD_x}$$

where $M_x$ is the median, and $MAD_x$ is the median absolute deviation. A sample estimate of the median absolute deviation is given by:

$$MAD_x = MEDIAN\left\{\left|X_1 - M_x\right|, \ldots\ldots \left|X_i - M_x\right|\right\}$$

where $M_x$ is the sample median. MAD has a finite sample breakdown point of approximately .5. Let $V_i$ be a function of the $Y_i$ scores as is $U_i$ is for the $X_i$ scores. The sample biweight midcovariance between $X$ and $Y$ is given by:

$$s_{bxy} = \frac{n\sum a_i (X_i - M_x)(1-U_i^2)^2 b_i (Y_i - M_y)(1-V_i^2)^2}{(\sum a_i (1-U_i^2)(1-5U_i^2))(\sum b_i (1-V_i^2)(1-5V_i^2))}$$

where,

$$a_i = 1 \text{ if } -1 \le U_i \le 1, \text{ otherwise } a_i = 0$$

and

$$b_i = 1 \text{ if } -1 \le V_i \le 1, \text{ otherwise } b_i = 0.$$

An estimate of the biweight midcorrelation between $X$ and $Y$ is given by:

$$r_b = \frac{s_{bxy}}{\sqrt{s_{bxx} s_{byy}}}$$

where $s_{bxx}$ and $s_{byy}$ are the biweight midvariance for the $X$ and $Y$ scores. The S script below calculates both the Pearson product-moment correlation and the biweight midcorrelation for a bivariate normal distribution whose population correlation is .40. An outlier is added to the sample, and then the two correlation coefficients are recalculated and compared again. A scatterplot with a best fit line is plotted after the outlier is added. Try changing various parameters and resubmitting the S script.

```
#### Calculate biweight midcovariance
####   from Wilcox (1997).

bicov<-function(x, y){
mx <- median(x)
my <-median(y)
ux <- abs((x - mx)/(9 * qnorm(0.75) * mad(x)))
uy <- abs((y - my)/(9 * qnorm(0.75) * mad(y)))
aval <- ifelse(ux <= 1, 1, 0)
bval <- ifelse(uy <= 1, 1, 0)
top <- sum(aval * (x - mx) * (1 - ux^2)^2 * bval * (y - my) * (1 - uy^2)^2)
top <- length(x) * top
botx <- sum(aval * (1 - ux^2) * (1 - 5 * ux^2))
boty <- sum(bval * (1 - uy^2) * (1 - 5 * uy^2))
bi <- top/(botx * boty)
bi
}

#### Calculate biweight midcorrelation.

bicor<-function(x,y){
x <-as.numeric(x)
y<-as.numeric(y)
bicov(x,y)/(sqrt(bicov(x,x)*bicov(y,y)))
}

#### Sample from multivariate normal

mvrnorm <- function(n, p, u, s, S) {
Z <- matrix(rnorm(n * p), p, n)
t(u + s*t(chol(S))%*% Z)
}

z <- mvrnorm(n=100, p=2, u=c(100,100), s=c(15,15), S=matrix(c(1, .4, .4, 1), ncol=2, nrow=2, byrow=T))
```

```
#### Calculate biweight midcovariance
####   from Wilcox (1997).

bicov<-function(x, y){
mx <- median(x)
my <-median(y)
ux <- abs((x - mx)/(9 * qnorm(0.75) * mad(x)))
uy <- abs((y - my)/(9 * qnorm(0.75) * mad(y)))
aval <- ifelse(ux <= 1, 1, 0)
bval <- ifelse(uy <= 1, 1, 0)
top <- sum(aval * (x - mx) * (1 - ux^2)^2 * bval * (y - my) * (1 - uy^2)^2)
top <- length(x) * top
botx <- sum(aval * (1 - ux^2) * (1 - 5 * ux^2))
boty <- sum(bval * (1 - uy^2) * (1 - 5 * uy^2))
bi <- top/(botx * boty)
bi
}

#### Calculate biweight midcorrelation.

bicor<-function(x,y){
x <-as.numeric(x)
y<-as.numeric(y)
bicov(x,y)/(sqrt(bicov(x,x)*bicov(y,y)))
}

#### Sample from multivariate normal

mvrnorm <- function(n, p, u, s, S) {
Z <- matrix(rnorm(n * p), p, n)
t(u + s*t(chol(S))%*% Z)
}

z <- mvrnorm(n=100, p=2, u=c(100,100), s=c(15,15), S=matrix(c(1, .4, .4, 1), ncol=2,
nrow=2, byrow=T))

# Pearson Correlation and Robust Correlation
# before outlier added

cor(z[,1], z[,2])
bicor(z[,1], z[,2])

# Add outlier to Y vector for smallest X value

z[z[,1]==min(z[,1]),2]<-z[z[,1]==min(z[,1]),2]*2.5
z.fit<-lm(z[,1]~z[,2])
plot(z[,1], z[,2])
abline(z.fit)

# Pearson Correlation and Robust Correlation
# after outlier added

cor(z[,1], z[,2])
bicor(z[,1], z[,2])
```

```
#### Calculate biweight midcovariance
####   from Wilcox (1997).
```

Run Program

# Calculating Empirical P-values, Empirical Power, and Confidence Intervals for a Robust Correlation Coefficient

The S script below calculates observed p-values, power, and confidence intervals for the biweight midcorrelation coefficient. A comparison is made between the Pearson correlation coefficient and the biweight midcorrelation before and after an outlier is added. Confidence intervals are calculated by simulating the null sampling distribution, and also by sampling from the alternate sampling distribution (Hall & Wilson, 1991). For details on calculating the percentile bootstrap, and calculating power using the percentile bootstrap, see the September issue of Benchmarks. For details on the percentile bootstrap see the August issue of Benchmarks. After running the following S script, try changing various parameters: sample size, population correlation, correlation coefficient (assign "est" to have value "bicor" or "cor") and the size of the outlier (change the multiplying factor 2.5 to a smaller value). Changing "est" to have value "cor" will allow one to calculate the p-value and power of the observed sample for the Pearson product-moment correlation.

```
# Function for extracting M-Estimator from the MASS Library
# Calculating Empirical P-values,
# Empirical Power, and Confidence
# Intervals for a Robust Correlation
# Coefficient Population parameters

pop.cor<-.40
pop1.mean<-100
pop1.stdev<-15
pop2.mean<-100
pop2.stdev<-15
sample.size<-40

# Sample from bivariate normal distribution
# with given parameters

# Set random number generator seed
set.seed(12)

z<-mvrnorm(n=sample.size, p=2, u=c(pop1.mean,pop2.mean), s=c(pop1.stdev,pop2.stdev), S=matrix(c(1, pop.cor, pop.cor, 1), ncol=2, nrow=2, byrow=T))

# Correlation before addition of outlier

cor.nolie<-cor(z[,1], z[,2])
cor.nolie.test<-cor.test(z[,1], z[,2])
bicor.nolie<-bicor(z[,1], z[,2])

# Add outlier to Y vector for smallest X value

z[z[,1]==min(z[,1]),2]<-z[z[,1]==min(z[,1]),2]*2.5

#####################################################
# Assign data vectors

z.x<-z[,1]
z.y<-z[,2]

### Set number of bootstrap samples
nboot<-1000

### Set Estimation Method
est<-bicor

# Select "cor" for Pearson r
# Select "bicor" for
# Robust correlation based on bi-weight covariance

# Calculate observed effect

cor.empirical<-cor(z.x, z.y)
diff.empirical<-est(z.x, z.y)

#################################
# Sampling from H0: sample columns independently
# to simulate correlation coefficient under H0

h0datax.list<-vector("list", nboot)

for (i in 1:nboot){
h0datax.list[[i]]<-sample(z.x, size = length(z.x), replace=T)
}

h0datay.list<-vector("list", nboot)

for (i in 1:nboot){
h0datay.list[[i]]<-sample(z.y, size = length(z.y), replace=T)
}

# Calculate correlation for each bootstrap sample
# under H0 sampling distribution

h0bvec<-unlist(lapply(1:nboot, function(i, x, y)est(x[[i]], y[[i]]), x = h0datax.list, y = h0datay.list))
mean.h0bvec<-mean(h0bvec)
stdev.h0bvec<-var(h0bvec)^.5

# Calculate Critical cutoffs on H0 Sampling distribution

critlow<-quantile(h0bvec,.025)
critup<-quantile(h0bvec,.975)

# Calculate Confidence Intervals from H0;
# shift null sampling distribution by observed effect

h0.ci <- 0
h0.ci[1] <- critlow + diff.empirical
h0.ci[2] <- critup + diff.empirical

#### Calculate P-value for H0
count<-length(h0bvec[abs(h0bvec)>=abs(diff.empirical)])
pvalue.empirical<-count/nboot

#################################
# Sample from H1: Sample with rows
# intact; bootstrap sampling of row indices

data.index <- matrix(sample(length(z.x), size = length(z.x) * nboot, replace = T), nrow = nboot)

# Use bootstrap sampled row indices # to extract the same row from both X
```

```
# Function for extracting M-Estimator from the MASS Library
# Calculating Empirical P-values,
# Empirical Power, and Confidence
# Intervals for a Robust Correlation
# Coefficient Population parameters

pop.cor<-.40
pop1.mean<-100
pop1.stdev<-15
pop2.mean<-100
pop2.stdev<-15
sample.size<-40

# Sample from bivariate normal distribution
# with given parameters

# Set random number generator seed
set.seed(12)

z<-mvrnorm(n=sample.size, p=2, u=c(pop1.mean,pop2.mean), s=c(pop1.stdev,pop2.stdev), S=matrix(c(1, pop.cor, pop.cor, 1), ncol=2, nrow=2, byrow=T))

# Correlation before addition of outlier

cor.nolie<-cor(z[,1], z[,2])
cor.nolie.test<-cor.test(z[,1], z[,2])
bicor.nolie<-bicor(z[,1], z[,2])

# Add outlier to Y vector for smallest X value

z[z[,1]==min(z[,1]),2]<-z[z[,1]==min(z[,1]),2]*2.5

#######################################################
# Assign data vectors

z.x<-z[,1]
z.y<-z[,2]

### Set number of bootstrap samples
nboot<-1000

### Set Estimation Method
est<-bicor

# Select "cor" for Pearson r
# Select "bicor" for
# Robust correlation based on bi-weight covariance

# Calculate observed effect

cor.empirical<-cor(z.x, z.y)
diff.empirical<-est(z.x, z.y)

#################################
# Sampling from H0: sample columns independently
# to simulate correlation coefficient under H0

h0datax.list<-vector("list", nboot)

for (i in 1:nboot){
h0datax.list[[i]]<-sample(z.x, size = length(z.x), replace=T)
}

h0datay.list<-vector("list", nboot)

for (i in 1:nboot){
h0datay.list[[i]]<-sample(z.y, size = length(z.y), replace=T)
}

# Calculate correlation for each bootstrap sample
# under H0 sampling distribution

h0bvec<-unlist(lapply(1:nboot, function(i, x, y)est(x[[i]], y[[i]]), x = h0datax.list, y = h0datay.list))
mean.h0bvec<-mean(h0bvec)
stdev.h0bvec<-var(h0bvec)^.5

# Calculate Critical cutoffs on H0 Sampling distribution

critlow<-quantile(h0bvec,.025)
critup<-quantile(h0bvec,.975)

# Calculate Confidence Intervals from H0;
# shift null sampling distribution by observed effect

h0.ci <- 0
h0.ci[1] <- critlow + diff.empirical
h0.ci[2] <- critup + diff.empirical

#### Calculate P-value for H0
count<-length(h0bvec[abs(h0bvec)>=abs(diff.empirical)])
pvalue.empirical<-count/nboot

#################################
# Sample from H1: Sample with rows
# intact; bootstrap sampling of row indices

data.index <- matrix(sample(length(z.x), size = length(z.x) * nboot, replace = T), nrow = nboot)

# Use bootstrap sampled row indices # to extract the same row from both X
# and Y vector

datax <- matrix(z.x[data.index], nrow = nboot, ncol=length(z.x))
datay <- matrix(z.y[data.index], nrow = nboot, ncol=length(z.y))
h1datax.list<-vector("list", nboot)

for (i in 1:nboot){h1datax.list[[i]]<-datax[i,]}

h1datay.list<-vector("list", nboot)

for (i in 1:nboot){h1datay.list[[i]]<-datay[i,]}

# Calculate the correlation under H1 sampling
# distribution

h1bvec<-unlist(lapply(1:nboot, function(i, x, y) est(x[[i]], y[[i]]), x = h1datax.list, y = h1datay.list))
mean.h1bvec<-mean(h1bvec)
stdev.h1bvec<-var(h1bvec)^.5

# Calculate confidence Intervals from H1

effectlow<-quantile(h1bvec,.025)
effectup<-quantile(h1bvec,.975)
h1.ci <- 0
h1.ci[1] <- effectlow
h1.ci[2] <- effectup

# Calculate Bootstrap Empirical Power
countup<-length(h1bvec[h1bvec>=critup])
countlow<-length(h1bvec[h1bvec<=critlow])
power.twotail<-(countup+countlow)/nboot

##### Report Results
# Pearson Correlation and bi-weight midcorrelation
# before outlier is added

cor.nolie
cor.nolie.test
bicor.nolie

# Results after adding outlier
# Mean and Standard Deviation (standard error)
# of Bootstrap Samples for Robust Correlation under H0

mean.h0bvec
stdev.h0bvec

# Normal theory standard error

1/(length(z.x)^.5)

# Mean and Standard Deviation (standard error)
# of Bootstrap Samples for Robust Correlation under H1

mean.h1bvec
stdev.h1bvec
mean.h1bvec-pop.cor

### Pearson r and Bias

cor.empirical
cor.empirical-pop.cor

### Robust Correlation and Bias
```

```
# Function for extracting M-Estimator from the MASS Library
# Calculating Empirical P-values,
# Empirical Power, and Confidence
# Intervals for a Robust Correlation
# Coefficient Population parameters

pop.cor<-.40
pop1.mean<-100
pop1.stdev<-15
pop2.mean<-100
pop2.stdev<-15
sample.size<-40

# Sample from bivariate normal distribution
# with given parameters

# Set random number generator seed
set.seed(12)

z<-mvrnorm(n=sample.size, p=2, u=c(pop1.mean,pop2.mean), s=c(pop1.stdev,pop2.stdev), S=matrix(c(1, pop.cor, pop.cor, 1), ncol=2, nrow=2, byrow=T))

# Correlation before addition of outlier

cor.nolie<-cor(z[,1], z[,2])
cor.nolie.test<-cor.test(z[,1], z[,2])
bicor.nolie<-bicor(z[,1], z[,2])

# Add outlier to Y vector for smallest X value

z[z[,1]==min(z[,1]),2]<-z[z[,1]==min(z[,1]),2]*2.5

#######################################################
# Assign data vectors

z.x<-z[,1]
z.y<-z[,2]

### Set number of bootstrap samples
nboot<-1000

### Set Estimation Method
est<-bicor

# Select "cor" for Pearson r
# Select "bicor" for
# Robust correlation based on bi-weight covariance

# Calculate observed effect

cor.empirical<-cor(z.x, z.y)
diff.empirical<-est(z.x, z.y)

#################################
# Sampling from H0: sample columns independently
# to simulate correlation coefficient under H0

h0datax.list<-vector("list", nboot)

for (i in 1:nboot){
h0datax.list[[i]]<-sample(z.x, size = length(z.x), replace=T)
}

h0datay.list<-vector("list", nboot)

for (i in 1:nboot){
h0datay.list[[i]]<-sample(z.y, size = length(z.y), replace=T)
}

# Calculate correlation for each bootstrap sample
# under H0 sampling distribution

h0bvec<-unlist(lapply(1:nboot, function(i, x, y)est(x[[i]], y[[i]]), x = h0datax.list, y = h0datay.list))
mean.h0bvec<-mean(h0bvec)
stdev.h0bvec<-var(h0bvec)^.5

# Calculate Critical cutoffs on H0 Sampling distribution

critlow<-quantile(h0bvec,.025)
critup<-quantile(h0bvec,.975)

# Calculate Confidence Intervals from H0;
# shift null sampling distribution by observed effect

h0.ci <- 0
h0.ci[1] <- critlow + diff.empirical
h0.ci[2] <- critup + diff.empirical

#### Calculate P-value for H0
count<-length(h0bvec[abs(h0bvec)>=abs(diff.empirical)])
pvalue.empirical<-count/nboot

#################################
# Sample from H1: Sample with rows
# intact; bootstrap sampling of row indices

data.index <- matrix(sample(length(z.x), size = length(z.x) * nboot, replace = T), nrow = nboot)

# Use bootstrap sampled row indices # to extract the same row from both X
# and Y vector

datax <- matrix(z.x[data.index], nrow = nboot, ncol=length(z.x))
datay <- matrix(z.y[data.index], nrow = nboot, ncol=length(z.y))
h1datax.list<-vector("list", nboot)

for (i in 1:nboot){h1datax.list[[i]]<-datax[i,]}

h1datay.list<-vector("list", nboot)

for (i in 1:nboot){h1datay.list[[i]]<-datay[i,]}

# Calculate the correlation under H1 sampling
# distribution

h1bvec<-unlist(lapply(1:nboot, function(i, x, y) est(x[[i]], y[[i]]), x = h1datax.list, y = h1datay.list))
mean.h1bvec<-mean(h1bvec)
stdev.h1bvec<-var(h1bvec)^.5

# Calculate confidence Intervals from H1

effectlow<-quantile(h1bvec,.025)
effectup<-quantile(h1bvec,.975)
h1.ci <- 0
h1.ci[1] <- effectlow
h1.ci[2] <- effectup

# Calculate Bootstrap Empirical Power
countup<-length(h1bvec[h1bvec>=critup])
countlow<-length(h1bvec[h1bvec<=critlow])
power.twotail<-(countup+countlow)/nboot

##### Report Results
# Pearson Correlation and bi-weight midcorrelation
# before outlier is added

cor.nolie
cor.nolie.test
bicor.nolie

# Results after adding outlier
# Mean and Standard Deviation (standard error)
# of Bootstrap Samples for Robust Correlation under H0

mean.h0bvec
stdev.h0bvec

# Normal theory standard error

1/(length(z.x)^.5)

# Mean and Standard Deviation (standard error)
# of Bootstrap Samples for Robust Correlation under H1

mean.h1bvec
stdev.h1bvec
mean.h1bvec-pop.cor

### Pearson r and Bias

cor.empirical
cor.empirical-pop.cor

### Robust Correlation and Bias
```

```
# Function for extracting M-Estimator from the MASS Library
# Calculating Empirical P-values,
# Empirical Power, and Confidence
# Intervals for a Robust Correlation
# Coefficient Population parameters

pop.cor<-.40
pop1.mean<-100
pop1.stdev<-15
pop2.mean<-100
pop2.stdev<-15
sample.size<-40

# Sample from bivariate normal distribution
# with given parameters

# Set random number generator seed
set.seed(12)

z<-mvrnorm(n=sample.size, p=2, u=c(pop1.mean,pop2.mean), s=c(pop1.stdev,pop2.stdev), S=matrix(c(1, pop.cor, pop.cor, 1), ncol=2, nrow=2, byrow=T))

# Correlation before addition of outlier

cor.nolie<-cor(z[,1], z[,2])
cor.nolie.test<-cor.test(z[,1], z[,2])
bicor.nolie<-bicor(z[,1], z[,2])

# Add outlier to Y vector for smallest X value

z[z[,1]==min(z[,1]),2]<-z[z[,1]==min(z[,1]),2]*2.5

#######################################################
# Assign data vectors

z.x<-z[,1]
z.y<-z[,2]

### Set number of bootstrap samples
nboot<-1000

### Set Estimation Method
est<-bicor

# Select "cor" for Pearson r
# Select "bicor" for
# Robust correlation based on bi-weight covariance

# Calculate observed effect

cor.empirical<-cor(z.x, z.y)
diff.empirical<-est(z.x, z.y)

#################################
# Sampling from H0: sample columns independently
# to simulate correlation coefficient under H0

h0datax.list<-vector("list", nboot)

for (i in 1:nboot){
h0datax.list[[i]]<-sample(z.x, size = length(z.x), replace=T)
}

h0datay.list<-vector("list", nboot)

for (i in 1:nboot){
h0datay.list[[i]]<-sample(z.y, size = length(z.y), replace=T)
}

# Calculate correlation for each bootstrap sample
# under H0 sampling distribution

h0bvec<-unlist(lapply(1:nboot, function(i, x, y)est(x[[i]], y[[i]]), x = h0datax.list, y = h0datay.list))
mean.h0bvec<-mean(h0bvec)
stdev.h0bvec<-var(h0bvec)^.5

# Calculate Critical cutoffs on H0 Sampling distribution

critlow<-quantile(h0bvec,.025)
critup<-quantile(h0bvec,.975)

# Calculate Confidence Intervals from H0;
# shift null sampling distribution by observed effect

h0.ci <- 0
h0.ci[1] <- critlow + diff.empirical
h0.ci[2] <- critup + diff.empirical

#### Calculate P-value for H0
count<-length(h0bvec[abs(h0bvec)>=abs(diff.empirical)])
pvalue.empirical<-count/nboot

#################################
# Sample from H1: Sample with rows
# intact; bootstrap sampling of row indices

data.index <- matrix(sample(length(z.x), size = length(z.x) * nboot, replace = T), nrow = nboot)

# Use bootstrap sampled row indices # to extract the same row from both X
# and Y vector

datax <- matrix(z.x[data.index], nrow = nboot, ncol=length(z.x))
datay <- matrix(z.y[data.index], nrow = nboot, ncol=length(z.y))
h1datax.list<-vector("list", nboot)

for (i in 1:nboot){h1datax.list[[i]]<-datax[i,]}

h1datay.list<-vector("list", nboot)

for (i in 1:nboot){h1datay.list[[i]]<-datay[i,]}

# Calculate the correlation under H1 sampling
# distribution

h1bvec<-unlist(lapply(1:nboot, function(i, x, y) est(x[[i]], y[[i]]), x = h1datax.list, y = h1datay.list))
mean.h1bvec<-mean(h1bvec)
stdev.h1bvec<-var(h1bvec)^.5

# Calculate confidence Intervals from H1

effectlow<-quantile(h1bvec,.025)
effectup<-quantile(h1bvec,.975)
h1.ci <- 0
h1.ci[1] <- effectlow
h1.ci[2] <- effectup

# Calculate Bootstrap Empirical Power
countup<-length(h1bvec[h1bvec>=critup])
countlow<-length(h1bvec[h1bvec<=critlow])
power.twotail<-(countup+countlow)/nboot

##### Report Results
# Pearson Correlation and bi-weight midcorrelation
# before outlier is added

cor.nolie
cor.nolie.test
bicor.nolie

# Results after adding outlier
# Mean and Standard Deviation (standard error)
# of Bootstrap Samples for Robust Correlation under H0

mean.h0bvec
stdev.h0bvec

# Normal theory standard error

1/(length(z.x)^.5)

# Mean and Standard Deviation (standard error)
# of Bootstrap Samples for Robust Correlation under H1

mean.h1bvec
stdev.h1bvec
mean.h1bvec-pop.cor

### Pearson r and Bias

cor.empirical
cor.empirical-pop.cor

### Robust Correlation and Bias
```

```
# Function for extracting M-Estimator from the MASS Library
# Calculating Empirical P-values,
# Empirical Power, and Confidence
# Intervals for a Robust Correlation
# Coefficient Population parameters

pop.cor<-.40
pop1.mean<-100
pop1.stdev<-15
pop2.mean<-100
pop2.stdev<-15
sample.size<-40

# Sample from bivariate normal distribution
# with given parameters

# Set random number generator seed
set.seed(12)

z<-mvrnorm(n=sample.size, p=2, u=c(pop1.mean,pop2.mean), s=c(pop1.stdev,pop2.stdev), S=matrix(c(1, pop.cor, pop.cor, 1), ncol=2, nrow=2, byrow=T))

# Correlation before addition of outlier

cor.nolie<-cor(z[,1], z[,2])
cor.nolie.test<-cor.test(z[,1], z[,2])
bicor.nolie<-bicor(z[,1], z[,2])

# Add outlier to Y vector for smallest X value

z[z[,1]==min(z[,1]),2]<-z[z[,1]==min(z[,1]),2]*2.5

#######################################################
# Assign data vectors

z.x<-z[,1]
z.y<-z[,2]

### Set number of bootstrap samples
nboot<-1000

### Set Estimation Method
est<-bicor

# Select "cor" for Pearson r
# Select "bicor" for
# Robust correlation based on bi-weight covariance

# Calculate observed effect

cor.empirical<-cor(z.x, z.y)
diff.empirical<-est(z.x, z.y)

#################################
# Sampling from H0: sample columns independently
# to simulate correlation coefficient under H0

h0datax.list<-vector("list", nboot)

for (i in 1:nboot){
h0datax.list[[i]]<-sample(z.x, size = length(z.x), replace=T)
}

h0datay.list<-vector("list", nboot)

for (i in 1:nboot){
h0datay.list[[i]]<-sample(z.y, size = length(z.y), replace=T)
}

# Calculate correlation for each bootstrap sample
# under H0 sampling distribution

h0bvec<-unlist(lapply(1:nboot, function(i, x, y)est(x[[i]], y[[i]]), x = h0datax.list, y = h0datay.list))
mean.h0bvec<-mean(h0bvec)
stdev.h0bvec<-var(h0bvec)^.5

# Calculate Critical cutoffs on H0 Sampling distribution

critlow<-quantile(h0bvec,.025)
critup<-quantile(h0bvec,.975)

# Calculate Confidence Intervals from H0;
# shift null sampling distribution by observed effect

h0.ci <- 0
h0.ci[1] <- critlow + diff.empirical
h0.ci[2] <- critup + diff.empirical

#### Calculate P-value for H0
count<-length(h0bvec[abs(h0bvec)>=abs(diff.empirical)])
pvalue.empirical<-count/nboot

#################################
# Sample from H1: Sample with rows
# intact; bootstrap sampling of row indices

data.index <- matrix(sample(length(z.x), size = length(z.x) * nboot, replace = T), nrow = nboot)

# Use bootstrap sampled row indices # to extract the same row from both X
# and Y vector

datax <- matrix(z.x[data.index], nrow = nboot, ncol=length(z.x))
datay <- matrix(z.y[data.index], nrow = nboot, ncol=length(z.y))
h1datax.list<-vector("list", nboot)

for (i in 1:nboot){h1datax.list[[i]]<-datax[i,]}

h1datay.list<-vector("list", nboot)

for (i in 1:nboot){h1datay.list[[i]]<-datay[i,]}

# Calculate the correlation under H1 sampling
# distribution

h1bvec<-unlist(lapply(1:nboot, function(i, x, y) est(x[[i]], y[[i]]), x = h1datax.list, y = h1datay.list))
mean.h1bvec<-mean(h1bvec)
stdev.h1bvec<-var(h1bvec)^.5

# Calculate confidence Intervals from H1

effectlow<-quantile(h1bvec,.025)
effectup<-quantile(h1bvec,.975)
h1.ci <- 0
h1.ci[1] <- effectlow
h1.ci[2] <- effectup

# Calculate Bootstrap Empirical Power
countup<-length(h1bvec[h1bvec>=critup])
countlow<-length(h1bvec[h1bvec<=critlow])
power.twotail<-(countup+countlow)/nboot

##### Report Results
# Pearson Correlation and bi-weight midcorrelation
# before outlier is added

cor.nolie
cor.nolie.test
bicor.nolie

# Results after adding outlier
# Mean and Standard Deviation (standard error)
# of Bootstrap Samples for Robust Correlation under H0

mean.h0bvec
stdev.h0bvec

# Normal theory standard error

1/(length(z.x)^.5)

# Mean and Standard Deviation (standard error)
# of Bootstrap Samples for Robust Correlation under H1

mean.h1bvec
stdev.h1bvec
mean.h1bvec-pop.cor

### Pearson r and Bias

cor.empirical
cor.empirical-pop.cor

### Robust Correlation and Bias
```

```
# Function for extracting M-Estimator from the MASS Library
# Calculating Empirical P-values,
```

Run Program

# Results

The output from the S script above are listed below.

```
> #####   Report Results
>
> # Pearson Correlation  and  bi-weight midcorrelation
> # before outlier is added
>
> cor.nolie
[1] 0.4094627
> cor.nolie.test

        Pearson's product-moment correlation

data:  z[, 1] and z[, 2]
t = 2.7667, df = 38, p-value = 0.008698
alternative hypothesis: true correlation is not equal to 0
sample estimates:
      cor
0.4094627

> bicor.nolie
[1] 0.4222197
```

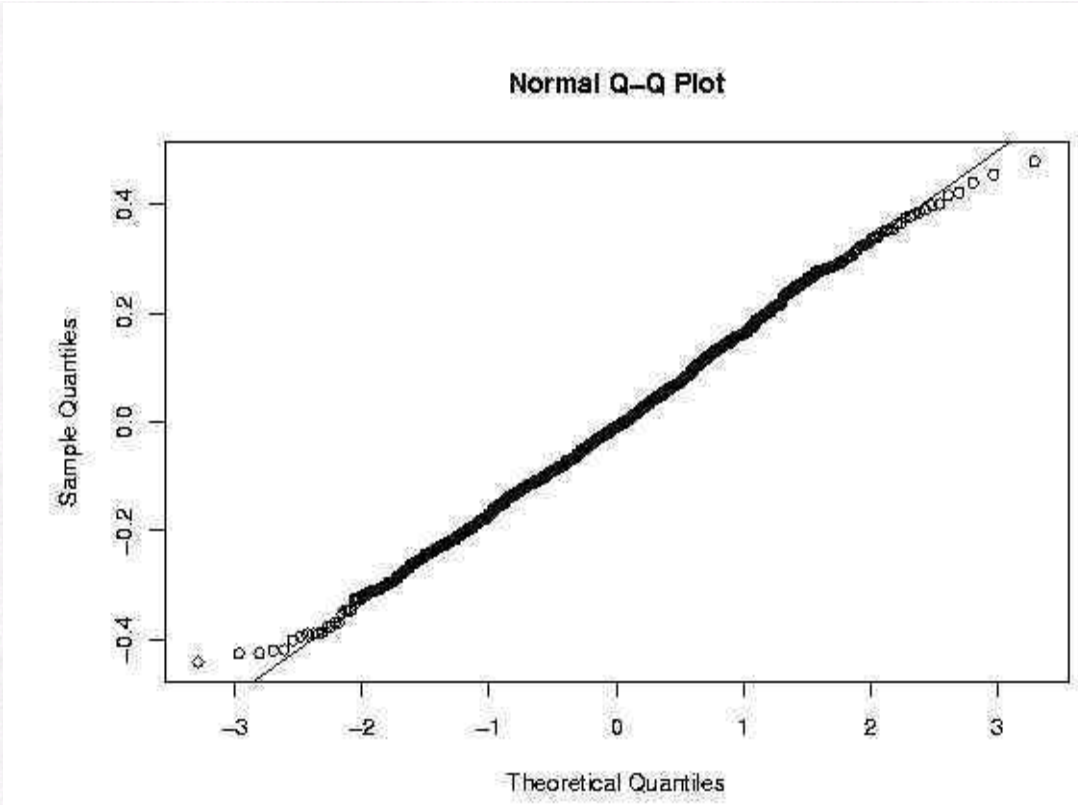
```
> # Results after adding outlier
>
> # Mean and Standard Deviation (standard error)
> # of Bootstrap Samples for Robust Correlation under HO
>
> mean.hObvec
[1] -0.003922042
> stdev.hObvec
[1] 0.1657244
>
> # normal theory standard error
> 1/(length(z.x)^.5)
[1] 0.1581139
>
> # Mean and Standard Deviation (standard error)
> # of Bootstrap Samples for Robust Correlation under H1
>
> mean.h1bvec
[1] 0.3206665
> stdev.h1bvec
[1] 0.1531287
> mean.h1bvec-pop.cor
[1] -0.07933351
```

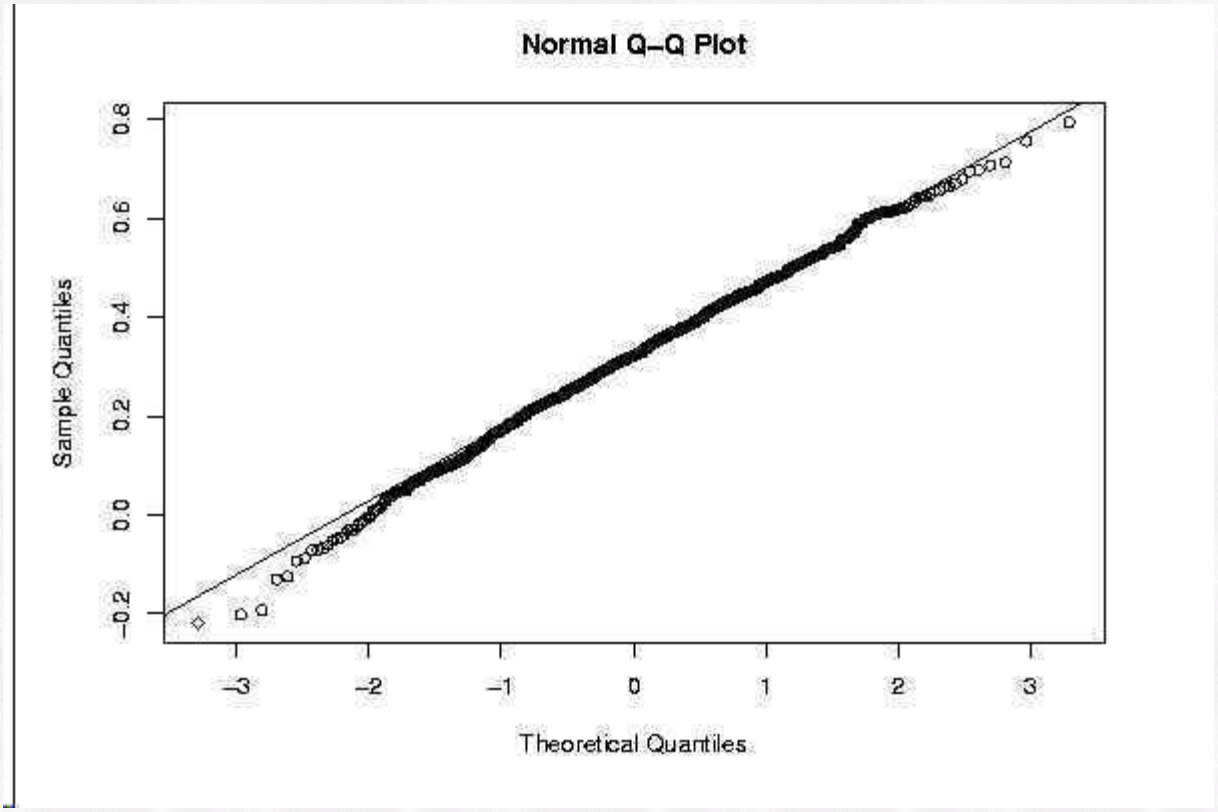
```
> ### Pearson r and Bias
>
> cor.empirical
[1] -0.07661869
> cor.empirical-pop.cor
[1] -0.4766187
>
> ### Robust Correlation and Bias
>
> diff.empirical
[1] 0.3373465
> diff.empirical-mean.h1bvec
[1] 0.01667996
> diff.empirical-pop.cor
[1] -0.06265355
```

```
> # Bootstrap empirical power for two-tail test
> # of the robust correlation
>
> power.twotail
[1] 0.494
>
> # Bootstrap empirical p-value for the
> # robust correlation
>
> pvalue.empirical
[1] 0.041
```

```
> # Two alternative ways of calculating confidence
> # intervals for Robust Correlation (H0 is preferred)
>
> # Confidence intervals based on the
> # bootstrap sampling distribution for the null
> # and alternate sampling distribution
>
> list(h0.ci = h0.ci)
$h0.ci
[1] 0.02247948 0.66213757

> list(h1.ci=h1.ci)
$h1.ci
[1] 0.009956474 0.614598842
```


Normal Q–Q Plot
Sample Quantiles
Theoretical Quantiles
−0.4
−0.2
0.0
0.2
0.4
−3
−2
−1
0
1
2
3


Normal Q–Q Plot
Sample Quantiles
Theoretical Quantiles
−0.2
0.0
0.2
0.4
0.6
0.8
−3
−2
−1
0
1
2
3

# References

Hall P., & Wilson S.R. (1991). Two guidelines for bootstrap hypothesis testing. *Biometrics, 47,* 757-762.

Wilcox, Rand (1997). *Introduction to Robust Estimation and Hypothesis Testing*. Academic Press, New York.