Thakur_Prabhat_Alternative Programming with R Assignment #2

Assignment Items (50 points total)

(1)

Skewness and kurtosis statistics are used to assess normality or lack thereof. These statistics are affected by distributional shape and sample size. This exercise will investigate how variable the skewness and kurtosis statistics are when sampling from a standard normal distribution. The results will provide a baseline for evaluating data samples in the future. The skewness for the standard normal distribution is 0.0. The kurtosis for the standard normal distribution is 3.0. Samples from a normal distribution will depart from these values.

Load the "moments" package which supplies *skewness()* and *kurtosis()* functions. Use three different samples size: 10, 40 and 160. For each sample size, draw 1000 random samples, and compute the skewness for each. Use *par(mfrow = c(3, 1))* and present histograms for these results. Additionally, compute the 2.5% (0.025) and 97.5% (0.975) quantiles for each vector of skewnesses and kurtoses and present.

The function defined below may be used to generate the samples for a given *n*. This function produces a 1000 x *n* dimensional array. Use *apply()* to compute the skewness for each row.

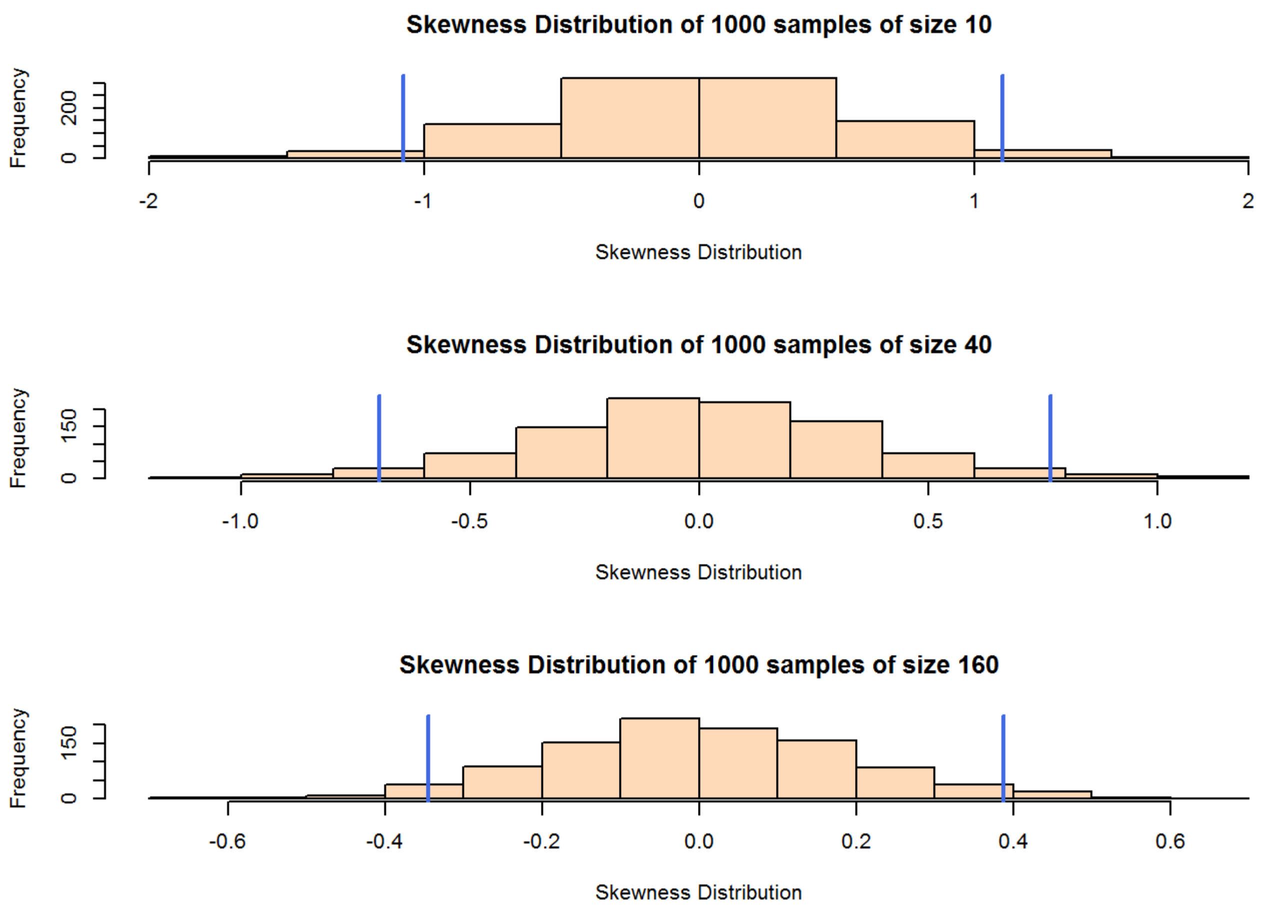
```
generate <- function(n){
  set.seed(123)
  result <- matrix(0, nrow = 1000, ncol = n)
  for (k in 1:1000) {
    result[k, ] <- rnorm(n, mean = 0, sd = 1)
  }
  return(result)
}

sample10 <- generate(10)
sample40 <- generate(40)
sample160 <-generate(160)
```

1(a) (4 points) In the following code 'chunk,' you will need to add code passing each 1000 x *n* matrix to an instance of *apply()*. Your matrices will be passed to *apply()* as X. You must specify the appropriate MARGIN and FUN to have skewness calculated for each row. The documentation page for *apply()* describes both arguments and their possible values. You will present the skewness results for n = 10, 40 and 160 as histograms, determine the first and third quartiles using *quantile()*, and display these quantiles on the histograms.

Use the function *sd()* to calculate the sample standard deviation for each of the three vectors of skewnesses. Additionally, the theoretical formula for the standard deviation of skewness based on a random sample from a normal distribution is *sqrt(6*(n-2)/((n+1)*(n+3)))*. Use this formula and compare to the sample standard deviation values. Show both sets of results side by side in a summary table.

```
library(moments)  # install.packages("moments")
library(knitr)

#Lets define a function to calculate standard deviation of skewness using
```

```
#theoretical formula for a given sample size.
sd.ske <- function (n) {return(sqrt(6*(n-2)/((n+1)*(n+3))))}

#Now calculate skewness for each row of the matrix created above using apply().
sample10.ske <- apply(sample10,MARGIN = 1,FUN = skewness)
sample40.ske <- apply(sample40,MARGIN = 1,FUN = skewness)
sample160.ske <- apply(sample160,MARGIN = 1,FUN = skewness)

#Present above calculated skewness results for sample size = 10, 40 and 160 on histograms.
par(mfrow = c(3, 1))

hist(sample10.ske,col = "peachpuff",main="Skewness Distribution of 1000 samples of size 10",xl
ab = "Skewness Distribution")
abline(v = quantile(sample10.ske,c(0.025,0.975)), col = "royalblue", lwd = 2)

hist(sample40.ske,col = "peachpuff",main="Skewness Distribution of 1000 samples of size 40",xl
ab = "Skewness Distribution")
abline(v = quantile(sample40.ske,c(0.025,0.975)), col = "royalblue", lwd = 2)

hist(sample160.ske,col = "peachpuff",main="Skewness Distribution of 1000 samples of size 160",
xlab = "Skewness Distribution")
abline(v = quantile(sample160.ske,c(0.025,0.975)), col = "royalblue", lwd = 2)
```



```
par(mfrow = c(1, 1))

#Calculate sample standard deviation of skewness for each of the three samples calcualted skew
```

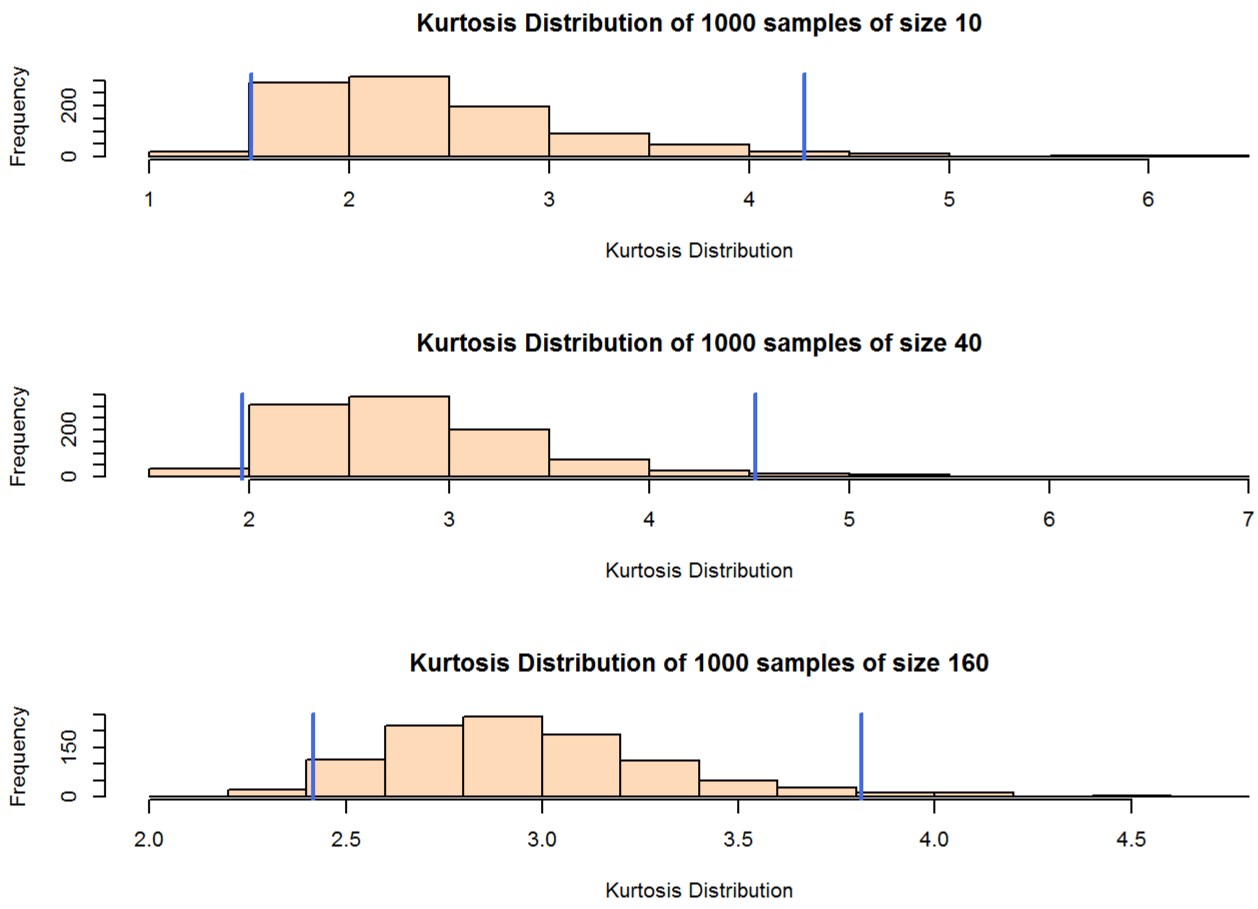
```
ness.
skewness_StandDeviation <- c(sd(sample10.ske),sd(sample40.ske),sd(sample160.ske))

#calculate sample standard deviation of skewness using theoretical formula.
skewnessTF_StandDeviation <- c(sd.ske(10),sd.ske(40),sd.ske(160))

#Show both values side by side by creating data frame.
compare.ske <- data.frame(c("n=10","n=40","n=160"), skewness_StandDeviation,skewnessTF_StandDe
viation)
colnames(compare.ske) <- c("Sample Size", "SD_SkewnessFromSample","SD_SkewnessFromFormula")
kable(compare.ske)
```

Sample Size	SD_SkewnessFromSample	SD_SkewnessFromFormula
n=10	0.5594200	0.5793655
n=40	0.3520808	0.3596179
n=160	0.1877718	0.1900629

1(b)(4 points) Follow the steps outlined in 1(a) above for kurtosis. Kurtosis for a standard normal distribution is 3.0 using the "moments" package *kurtosis()* function.

The theoretical formula for the standard deviation of kurtosis based on a random sample from a normal distribution is *sqrt(24*n*(n-2)*(n-3)/((n+1)*(n+1)*(n+3)*(n+5)))*. Using *sd()*, compute the sample standard deviations for the three random samples, and compare to the results you compute using the theoretical formula for the three sample sizes. Show both sets of result side by side in a summary table.

```
#Lets define a function to calculate standard deviation of Kurtosis using
#theoretical formula for a given sample size.
sd.kur <- function (n) {return(sqrt(24*n*(n-2)*(n-3)/((n+1)*(n+1)*(n+3)*(n+5))))}

#Now calculate kurtosis for each row of the matrix created above using apply().
sample10.kur <- apply(sample10,MARGIN = 1,FUN = kurtosis)
sample40.kur <- apply(sample40,MARGIN = 1,FUN = kurtosis)
sample160.kur <- apply(sample160,MARGIN = 1,FUN = kurtosis)

#Present above calculated kurtosis results for sample size = 10, 40 and 160 on histograms.
par(mfrow = c(3, 1))

hist(sample10.kur,col = "peachpuff",main="Kurtosis Distribution of 1000 samples of size 10",xl
ab = "Kurtosis Distribution")
abline(v = quantile(sample10.kur,c(0.025,0.975)), col = "royalblue", lwd = 2)

hist(sample40.kur,col = "peachpuff",main="Kurtosis Distribution of 1000 samples of size 40",xl
ab = "Kurtosis Distribution")
abline(v = quantile(sample40.kur,c(0.025,0.975)), col = "royalblue", lwd = 2)

hist(sample160.kur,col = "peachpuff",main="Kurtosis Distribution of 1000 samples of size 160",
xlab = "Kurtosis Distribution")
abline(v = quantile(sample160.kur,c(0.025,0.975)), col = "royalblue", lwd = 2)
```



```
par(mfrow = c(1, 1))

#Calculate sample standard deviation of kurtosis for each of the three samples calcualted kurt
osis.
kurtosis_StandDeviation <- c(sd(sample10.kur),sd(sample40.kur),sd(sample160.kur))

#calculate sample standard deviation of kurtosis using theoretical formula.
kurtosisTF_StandDeviation <- c(sd.kur(10),sd.kur(40),sd.kur(160))

#Show both values side by side by creating data frame.
compare.kur <- data.frame(c("n=10","n=40","n=160"), kurtosis_StandDeviation,kurtosisTF_StandDe
viation)
colnames(compare.kur) <- c("Sample Size", "SD_KurtosisFromSample","SD_KurtosisFromFormula")
kable(compare.kur)
```

Sample Size	SD_KurtosisFromSample	SD_KurtosisFromFormula
n=10	0.7344158	0.7547266
n=40	0.6402491	0.6441751
n=160	0.3570882	0.3696429

1(c)(2 points) Evaluate the results you have obtained. What do you observe regarding the convergence of the sampling distributions in terms of their shapes as a function of sample size, and accuracy in comparison to the “true”" skewness value of 0.0 and kurtosis value of 3.0.

Answer: From above exercise, the skewness and kurtosis statistics appear to be very dependent on the sample size. For both skewness and kurtosis, the range between 2.5% and 97.5% quantiles has shrunk with increase in

size. As sample size is increasing both sampling distributions are converging to their true value of 0.0 and 3 respectively and showing approximately normal distribution. From this test, we can conclude that, smaller sample sizes can give results that can mislead about the population distribution.

(2) Yes, and just imagine what happens with non-normal distributions! These metrics are not the best.

This problem requires quartile calculations using random samples of different sizes from the standard normal distribution. The quartiles for the standard normal distribution are (-0.6745, 0.0 and +0.6745), obtainable by *qnorm(c(0.25, 0.5, 0.75), mean = 0, sd = 1, lower.tail = TRUE)*. This is illustrated below.

```
qnorm(c(0.25, 0.5, 0.75), mean = 0, sd = 1, lower.tail = TRUE)
```

```
## [1] -0.6744898  0.0000000  0.6744898
```

(2)a (4 points) Use *set.seed(127)* and *rnorm(n, mean = 0, sd = 1)* with *n* = 25, *n* = 50, *n* = 100 and draw three random samples. Reset *set.seed(127)* prior to drawing each of the three samples. For each sample, calculate the first, second and third quartile using *quantile()*. Use “type = 2” (Business Statistics) and “type = 7” (R default). Generate quartiles for each method. Display the results in a table for comparison purposes. Take note of the results for the first and third quartile in particular.

```
# Add your set.seed(), rnorm() and quantile() code to this code 'chunk'.
# Use set.seed(127) prior to drawing a random sample each time.

#Lets define a function to return first, second and third quartile for a requested sample and
type.
rsquntile <- function(x,t) {return(quantile(x,c(0.25,0.5,0.75),type=t))}

# Create first random sample of size 25.
set.seed(127)
rsampel1 <- rnorm(25, mean = 0, sd = 1)

# Create second random sample of size 50.
set.seed(127)
rsampel2 <- rnorm(50, mean = 0, sd = 1)

# Create third random sample of size 100.
set.seed(127)
rsampel3 <- rnorm(100, mean = 0, sd = 1)

#Calculate quartile for all three sampels with type 2 and 7 and add as columns to a matrix.
sampels.qtile <- data.frame(c("FirsQtile","SecondQtile","ThirdQtile"),rsquntile(rsampel1,2),rs
quntile(rsampel1,7),rsquntile(rsampel2,2),rsquntile(rsampel2,7),rsquntile(rsampel3,2),rsquntil
e(rsampel3,7))
colnames(sampels.qtile) <- c("Quartiles", "n=25,T=2","n=25,T=7","n=50,T=2","n=50,T=7","n=100,T
=2","n=100,T=7")
kable(sampels.qtile)
```

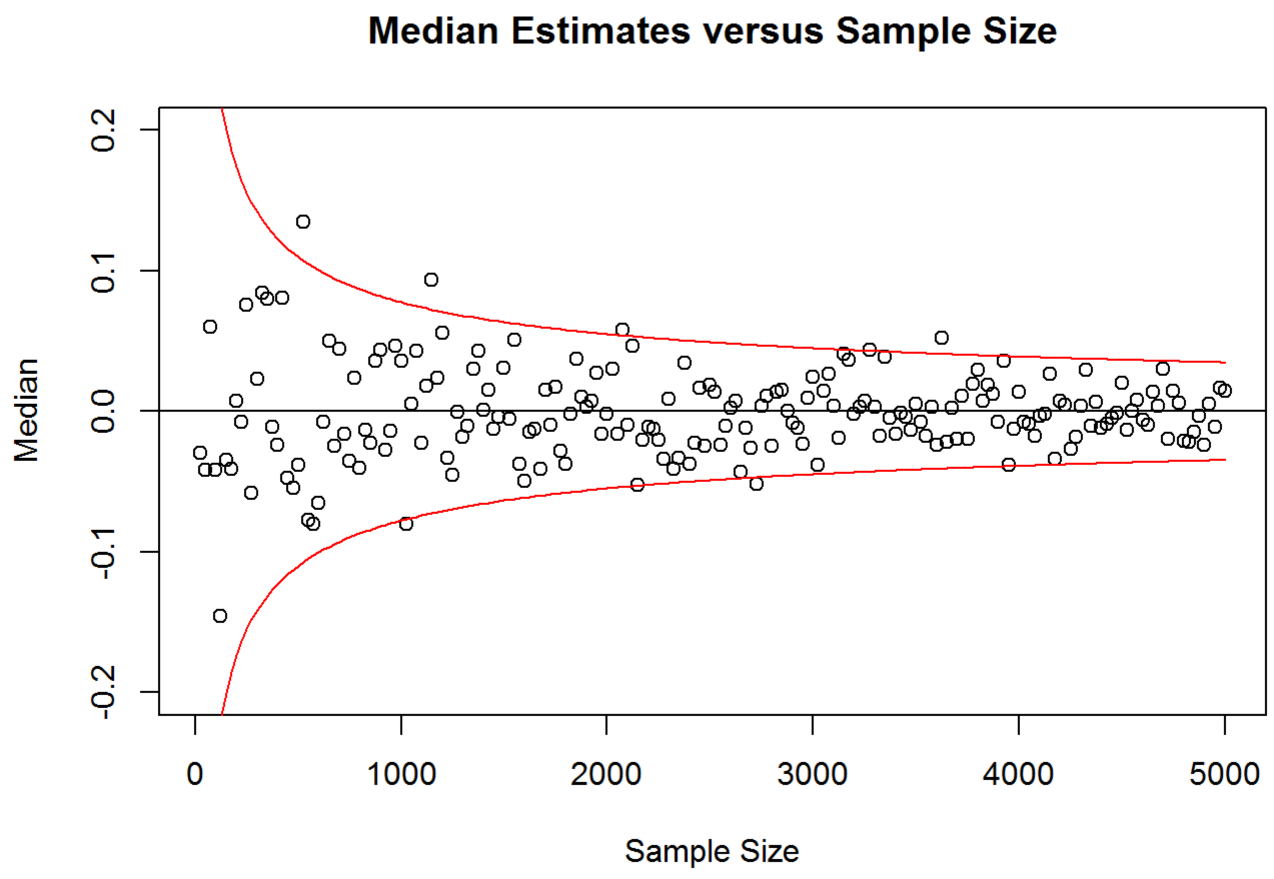
	Quartiles	n=25,T=2	n=25,T=7	n=50,T=2	n=50,T=7	n=100,T=2	n=100,T=7
25%	FirsQtile	-0.4939396	-0.4939396	-0.7493557	-0.7476301	-0.6714935	-0.6479629
50%	SecondQtile	-0.0298100	-0.0298100	-0.1306109	-0.1306109	0.0024713	0.0024713
75%	ThirdQtile	0.6059296	0.6059296	0.5646199	0.5336772	0.7256884	0.7126415

The central limit theorem can be used to derive a quantile estimate standard deviation formula. It estimates the standard deviation of the sampling distribution of any given sample quantile. This formula is valid for a known continuous density function. It is a function of that density function, quantile probabilies and sample size.

In the limit, as the sample size increases, a quantile estimate will have a normal sampling distribution with an expected value equal to the true quantile value q and a variance equal to p(1-p)/n(f(q)**2). In this statement, f() denotes the formula for the density function, and p is the probability (percentile) corresponding to the quantile. For example, in the case of a standard normal density, for the median, q = 0 and f(q) = *dnorm(0, 0, 1)* and p = *pnorm(q, 0, 1, lowertail = TRUE)*.

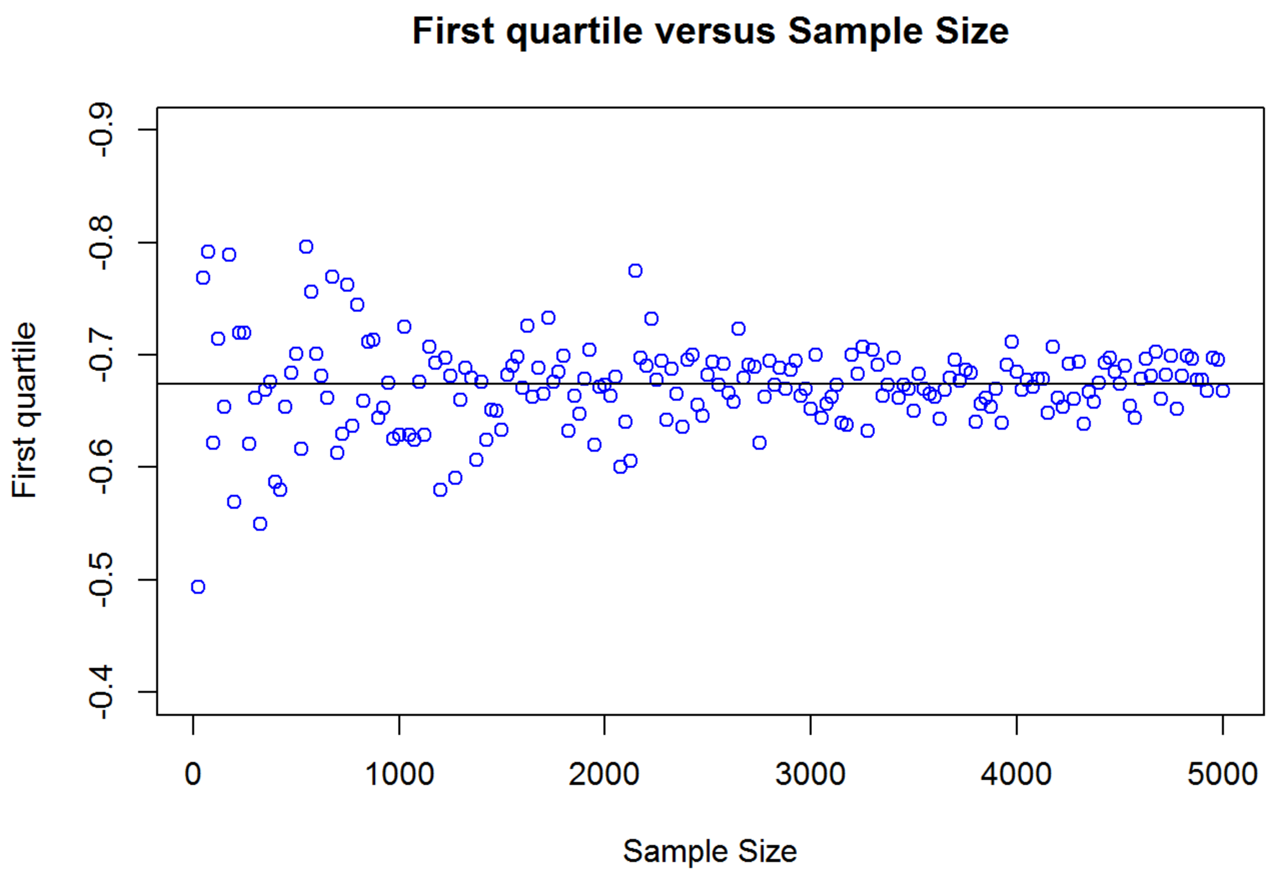
For a selected coverage level, such formulas may be used to superimpose boundary curves on the plots produced in Part 1. Execute the code below which illustrates this for the estimate of the median. Since the central limit theorem promises convergence to normality, these curves are constructed to contain 95% of the median estimates for each sample size. Similar curves may be constructed for quartile estimates.

```r
set.seed(127)
quant <- numeric(0)
for (k in seq(from = 25, to = 5000, by = 25)){
  x <- rnorm(k, mean = 0, sd = 1)
  quant <- rbind(quant, quantile(x, probs = 0.5, type = 7))
}

size <- seq(from = 25, to = 5000, by = 25)

c1 <- qnorm(0.975,0,1,lower.tail = T)
c2 <- qnorm(0.025,0,1,lower.tail = T)

plot(size, quant, main = "Median Estimates versus Sample Size", xlab = "Sample Size", ylab = "
Median", ylim = c(-0.2,0.2))
abline(h = 0.0)
lines(size, c1*sqrt(0.5*0.5/(size*dnorm(0,0,1)**2)), col = "red")
lines(size, c2*sqrt(0.5*0.5/(size*dnorm(0,0,1)**2)), col = "red")
```

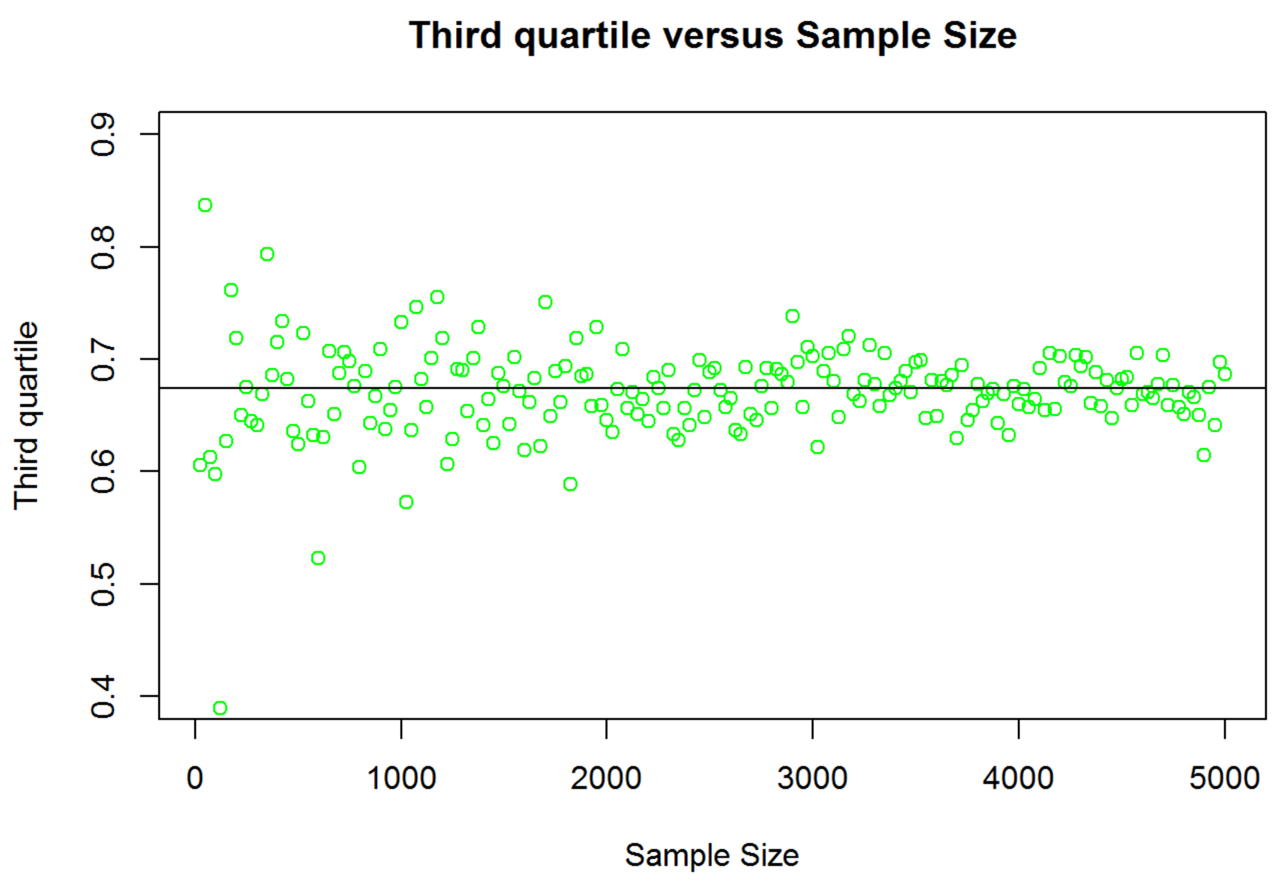


2(b) (3 points) Using code based on the example above, generate scatterplots showing the sample quartile values for the first and third quartiles calculated using type 7 for the sample sizes in *seq(from = 25, to = 5000, by = 25)*. Do not add the curved boundary lines. For these plots use *ylim = c(0.4,0.9)* for the first quartile, and *ylim = c(-0.4, -0.9)* for the third quartile. Give consideration to color, titles and legends.

```r
# Add your set.seed(), rnorm() and quantile() code to this code 'chunk'.
# Only use set.seed(127) once at the beginning of your code 'chunk'.
set.seed(127)

#Calculate first and third quartiles using type 7.
quant <- numeric(0)
for (k in seq(from = 25, to = 5000, by = 25)){
  x <- rnorm(k, mean = 0, sd = 1)
  quant <- rbind(quant, quantile(x, probs = c(0.25,0.75), type = 7))
}

#Create vector of sample sizes.
size <- seq(from = 25, to = 5000, by = 25)

#Scatterplots for sample quartile values for the first and third quartiles
plot(size, quant[,1], main = "First quartile versus Sample Size", xlab = "Sample Size", ylab =
 "First quartile", ylim = c(-0.4,-0.9),col="blue")
abline(h = qnorm(0.25,0,1,lower.tail = TRUE))
```



```
plot(size, quant[,2], main = "Third quartile versus Sample Size", xlab = "Sample Size", ylab =
 "Third quartile", ylim = c(0.4, 0.9),col="green" )
abline(h = qnorm(0.75,0,1,lower.tail = TRUE))
```



2(c)(3 points) Comment. What do these displays indicate about convergence of quartile estimates to the true values? Make at least two observations.

Answer: 1. AS the sample size is increasing, both 1st and 3rd quartile estimate are converging towards the true 1st and 3rd quartile value -0.674 and 0.674 respectively and there shapes are approximately normally distributed. 2. In another observation, we can see that for very small sample size few 3nd quartile estimate quite far from the true value as compared to 1st quartile estimate but for small sample size 1st quartile estimates are more spread-out compared to 3nd quartile estimate.

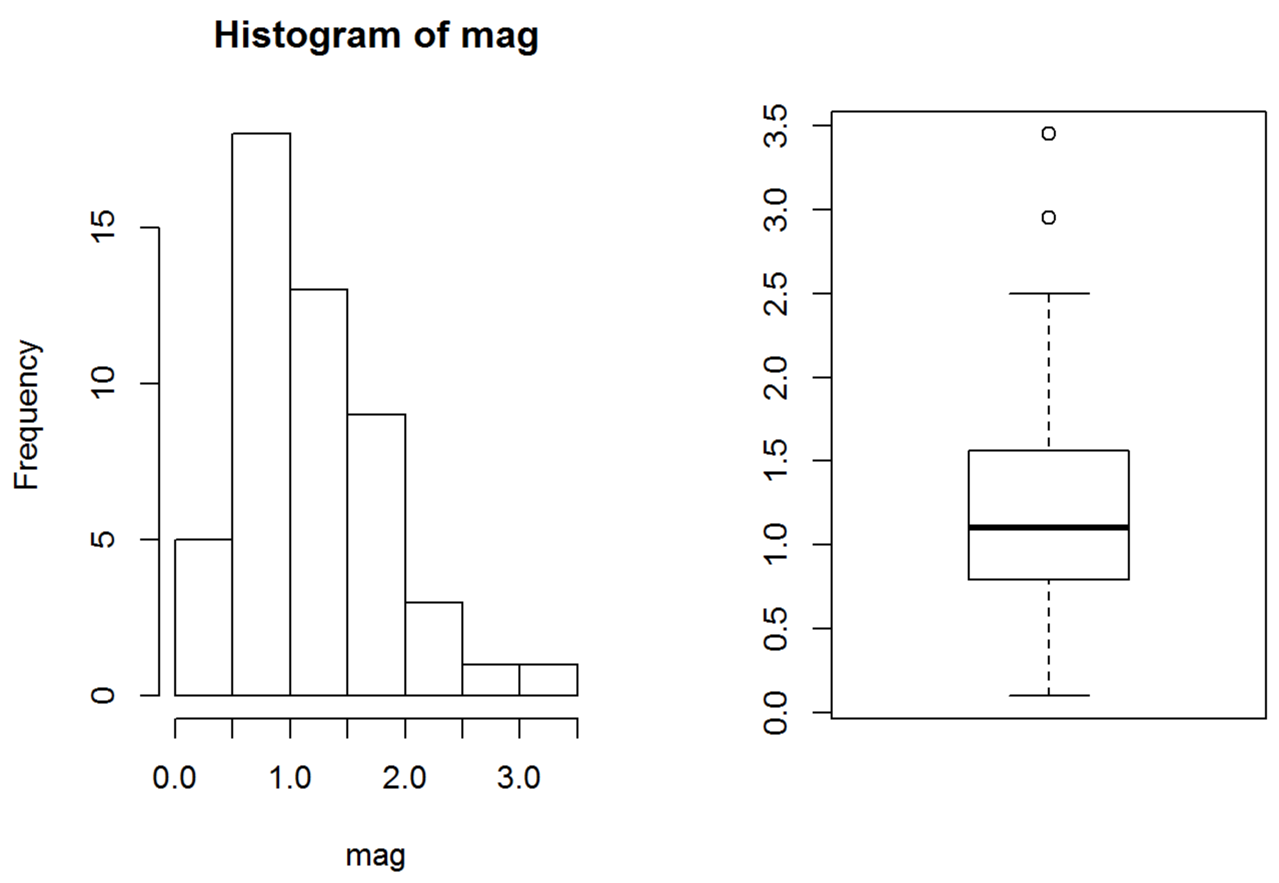
(3) Convergence takes time. Even with n = 50,000 there is visible variation.

Bootstrapping will be used to estimate confidence intervals for the variance of a non-normal distribution. Earthquake magnitude data will be used. Results will be compared to confidence intervals constructed using the traditional chi-square method that assumes normality. The earthquake magnitude data are right-skewed and are not derived from a normal distribution.

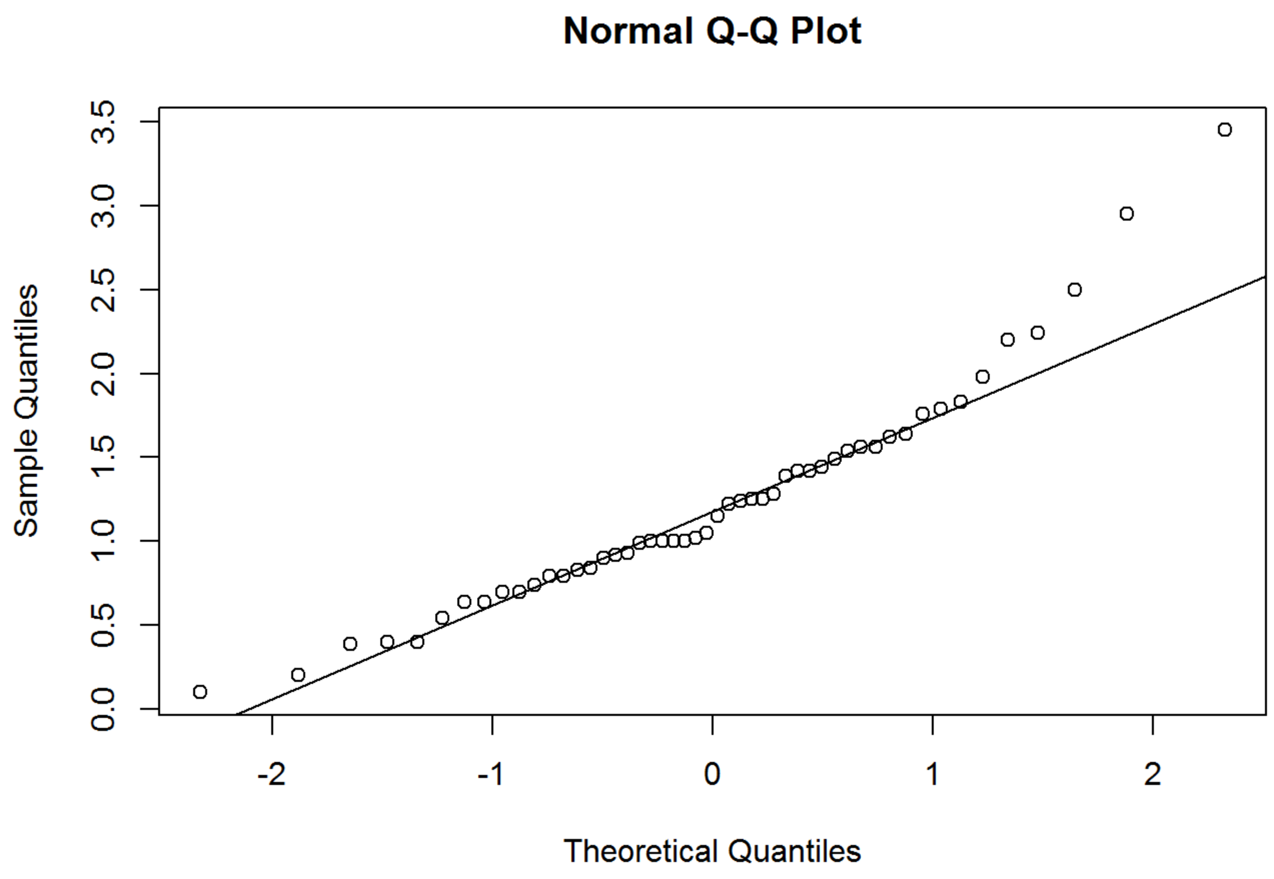
The following code 'chunk' defines the vector "mag" with the earthquake magnitudes, and illustrates evidence of non-normality.

```
mag <- c(0.70, 0.74, 0.64, 0.39, 0.70, 2.20, 1.98, 0.64, 1.22, 0.20, 1.64, 1.02,
         2.95, 0.90, 1.76, 1.00, 1.05, 0.10, 3.45, 1.56, 1.62, 1.83, 0.99, 1.56,
         0.40, 1.28, 0.83, 1.24, 0.54, 1.44, 0.92, 1.00, 0.79, 0.79, 1.54, 1.00,
         2.24, 2.50, 1.79, 1.25, 1.49, 0.84, 1.42, 1.00, 1.25, 1.42, 1.15, 0.93,
         0.40, 1.39)

par(mfrow = c(1,2))
hist(mag)
```

```
boxplot(mag)
```



```
par(mfrow = c(1,1))

qqnorm(mag)
qqline(mag)
```



3(a)(2 points)

Compute the confidence interval for the population variance using the traditional chi-square method which assumes normality.The sample variance point estimate may be calculated using the *var()* function. You will need to add code calculating a 95% confidence interval for the sample variance estimate, based on a chi-square distribution. This assumes normality.

```
# Sample size n , degree of freedom df and sample variance s^2
smp.size <- length(mag)
df <- smp.size -1
smp.var <- var(mag)

cat("Sample Size : ", smp.size, " ,Degree of freedom: ",df, " and Sample variance : ",smp.var,
"\n")
```

```
## Sample Size :  50  ,Degree of freedom:  49  and Sample variance :  0.4401215
```

```
# Lets find the Chi-square value for left and right tail of 95% confidence interval.
chi.values <- qchisq(c(0.025, 0.975), df, lower.tail = TRUE)
cat("Chai Values of left and right tail: ",chi.values," \n")
```

```
## Chai Values of left and right tail:  31.55492 70.22241
```

```
# Now the interval can be constructed for population variance .
```

```
result <- c(signif(df*smp.var/chi.values[2], digits = 4),
         signif(df*smp.var/chi.values[1], digits = 5))

cat("With 95% confidence interval,Using chi-square method the population variance is somewhere
 between ",result[1]," and ",result[2],".\n")
```

```
## With 95% confidence interval,Using chi-square method the population variance is somewhere b
etween  0.3071  and  0.68344 .
```

There is an extensive literature on bootstrapping. The methods shown here give an indication of the possibilities for estimating confidence intervals for a wide range of parameters. Some literature citations will be mentioned.

3(b)(5 points)

Use the Percentile Bootstrap Method for estimating a 95% confidence interval for the variance. This requires drawing 10000 random samples of size *n* = 50 with replacement from the earthquake data “mag”. The sample variance will be computed for each, and the 2.5% (0.025) and 97.5% (0.975) quantiles computed for these data using *quantile()*. Please keep *set.seed(123)*. The *replicate()* function can be used to easily “replicate” the sampling and sample variance steps, with *n* = 10000. Quantiles are then determined from the distribution of results.

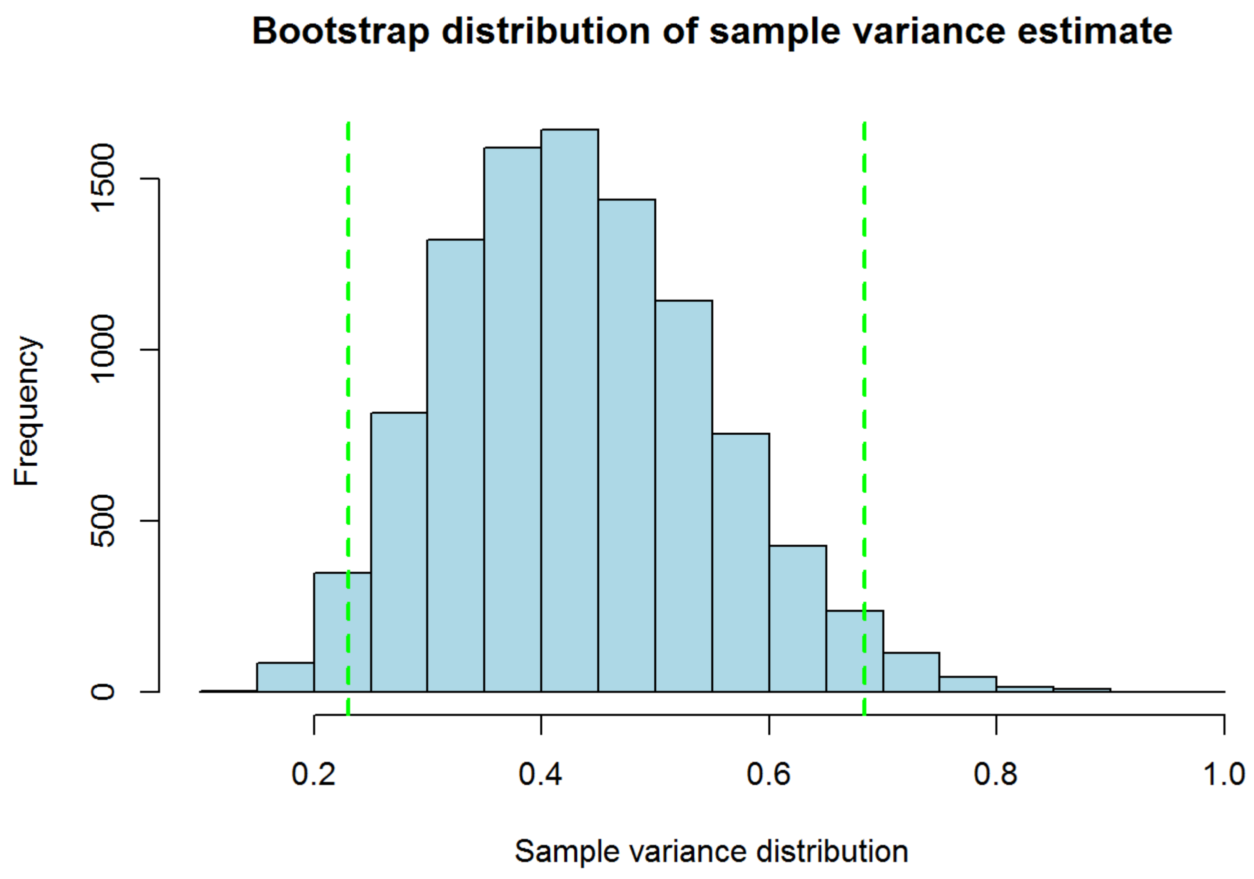
Present a histogram of the 10,000 sample variances along with the confidence interval. Show the quantiles on a histogram of the 10,000 sample variances. The quantiles provide a 95% percentile bootstrap confidence interval.

Present a table displaying the chi-square confidence interval and the bootstrap confidence interval.

```
set.seed(123)

# Define number of resamples drawn.
N <- 10000

# Define vectors for storage purposes.
samples.vars <- numeric(N)

# Calculate variance for each bootstrap sample of size 50 drawn from mag dataset.
for (i in 1:N)
{
  samples.vars[i] <- var(sample(mag, 50, replace = TRUE))
}

# Percentile bootstrapping on variance confidence interval.
result1 <- round(quantile(samples.vars, prob = c(0.025,0.975)), digits = 3)
cat("Using Percentile Bootstrap Method, with 95% confidence interval, the population variance
is somewhere between ",result1[1]," and ",result1[2],".\n")
```

```
## Using Percentile Bootstrap Method, with 95% confidence interval, the population variance is
 somewhere between  0.23  and  0.684 .
```

```
hist(samples.vars, main = "Bootstrap distribution of sample variance estimate", col = "lightbl
ue",xlab = "Sample variance distribution")
abline(v= quantile(samples.vars, probs = c(0.025,0.975)),col = "green", lty = 2, lwd = 2)
```



```r
#Confidence interval comparison
mag.variance <- rbind(result,result1)
rownames(mag.variance) <- c("chi-square" ,"bootstrap")
cat("\nConfidence interval comparison\n")
```

```
##
## Confidence interval comparison
```

```r
mag.variance
```

```
##               2.5%   97.5%
## chi-square 0.3071 0.68344
## bootstrap  0.2300 0.68400
```

3(c) (2 points) Compare and comment upon the traditional (i.e. chi-square) and bootstrap confidence interval results. Which would you use? Why?

Answer: Result from the chi-square method is close to the true value of mag dataset variance which is around 0.44. Though the population variance is non-normal distribution and chi-square method assumes normality, we could still get a better result from because the sample size is 50. The histogram of bootstrap methods is right skewed. I will prefer chi-square method as its simple and the result is close to the true value. Central Limit Theorem: If random samples of size n are repeatedly drawn from a population, the sample statistic are approximately normally distributed for sufficiently large sample sizes (n ??? 30) regardless of the shape of the

The chi-square method is known to be sensitive to non-normality. The bootstrap method will adjust for this. A study using many different samples of size 50 would be needed to understand the coverage rates.

population distribution.

This part requires using the “boot” package discussed in Kabacoff Section 12.6, pages 292-298. The “boot” package requires a function be written to return the sample variance values for each individual resample drawn. Use the following function in *boot()* for the argument “statistic.” This function is defined for you below.

```
f <- function(data, i){
  d <- data[i]
  return(var(d))
}
```

3(d) (6 points) The user-defined function is passed to *boot()* with “mag” along with the number of samples to be drawn with replacement. Confidence bounds follow. The “boot” package has a variety of options for determining confidence intervals. See *boot.ci()*, shown below, with the percentile option. The different computational options may produce slightly different results depending on the number of samples drawn during bootstrapping. For this problem, use 10,000 samples drawn with replacement. Again, please keep *set.seed(123)*.

```
library(boot)  # install.packages("boot")
```

```
## Warning: package 'boot' was built under R version 3.3.3
```

```
set.seed(123)

# Here, you will need to add code defining an object created by boot() with data = mag, statis
tic = f,
# and R = 10000. For calculating the quantiles, you will need to refer to the vector of result
s via "$t".

mag.boot <- boot(data=mag, statistic=f, R=10000)

# Calculating variance confidence interval using boot() function.
result2 <- round(quantile(mag.boot$t, prob = c(0.025,0.975)), digits = 3)
cat("With 95% confidence interval Using boot() function, the population variance is somewhere
between ",result2[1]," and ",result2[2],".\n")
```

```
## With 95% confidence interval Using boot() function, the population variance is somewhere be
tween  0.228  and  0.689 .
```

Alternatively, we could use *boot.ci()*. To do this, we need to pass the object defined above by *boot()*, specifying “conf = 0.95” and “type =”perc“. *boot.ci()* calculates confidence intervals and stores them at”$percent" of the output.

```
# Add your code here.

# Calculating variance confidence interval using boot.ci() function.
mag.boot.ci <- boot.ci(mag.boot, conf = 0.95,type = "perc")
result3 <- round(mag.boot.ci$percent, digits = 3)
cat("With 95% confidence interval Using boot.ci() function, the population variance is somewhe
re between ",result3[1,4]," and ",result3[1,5],".\n")
```

```
## With 95% confidence interval Using boot.ci() function, the population variance is somewhere
```

```
between  0.228  and  0.689 .
```

(4)

Fisher proposed a method for calculating a confidence interval for the Pearson Correlation Coefficient. This method involves transformation of r, the sample Pearson Correlation Coefficient. A z-score is calculated and confidence interval is constructed. This confidence interval is then transformed back to the original scale of measurement. This method is explained in the links:

http://www2.sas.com/proceedings/sugi31/170-31.pdf

http://onlinestatbook.com/2/estimation/correlation_ci.html

Use the data provided and construct the data frame “test” with the code below. The data frame contains test results for 49 students on two standardized tests. Each student took both tests. Do not change the order of the two test entries or the matching per student will not be correct

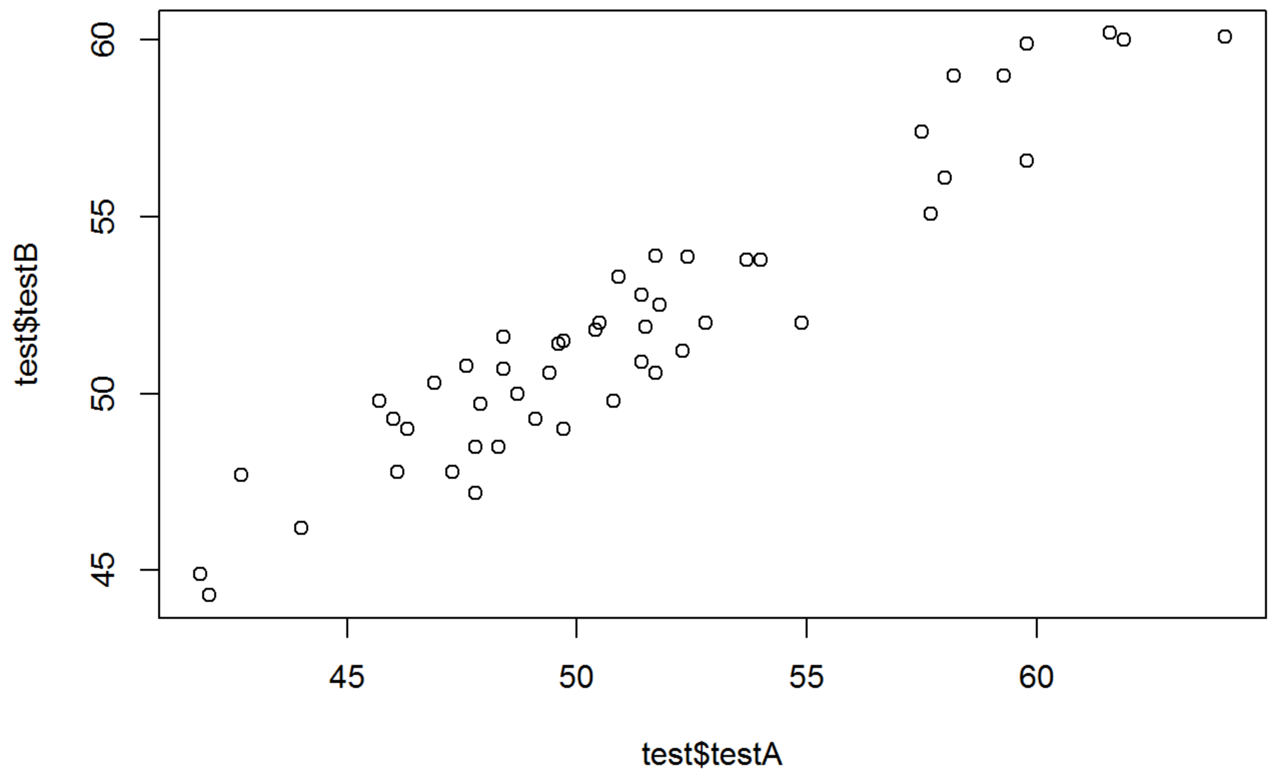
```
testA <- c(58,49.7,51.4,51.8,57.5,52.4,47.8,45.7,51.7,46,50.4,61.9,49.6,61.6,54,54.9,49.7,
           47.9,59.8,52.3,48.4,49.1,53.7,48.4,47.6,50.8,58.2,59.8,42.7,47.8,51.4,50.9,49.4,
           64.1,51.7,48.7,48.3,46.1,47.3,57.7,41.8,51.5,46.9,42,50.5,46.3,44,59.3,52.8)
testB <- c(56.1,51.5,52.8,52.5,57.4,53.86,48.5,49.8,53.9,49.3,51.8,60,51.4,60.2,53.8,52,
           49,49.7,59.9,51.2,51.6,49.3,53.8,50.7,50.8,49.8,59,56.6,47.7,47.2,50.9,53.3,
           50.6,60.1,50.6,50,48.5,47.8,47.8,55.1,44.9,51.9,50.3,44.3,52,49,46.2,59,52)

test <- as.data.frame(cbind(testA,testB))

str(test)
```

```
## 'data.frame':    49 obs. of  2 variables:
##  $ testA: num  58 49.7 51.4 51.8 57.5 52.4 47.8 45.7 51.7 46 ...
##  $ testB: num  56.1 51.5 52.8 52.5 57.4 ...
```

```
summary(test)
```

```
##      testA           testB      
##  Min.   :41.80   Min.   :44.30  
##  1st Qu.:47.80   1st Qu.:49.30  
##  Median :50.50   Median :51.40  
##  Mean   :51.25   Mean   :51.95  
##  3rd Qu.:53.70   3rd Qu.:53.80  
##  Max.   :64.10   Max.   :60.20  
```

```
plot(test$testA,test$testB)
```



4(a)(3 points) Determine a 95% confidence interval for the Pearson Correlation Coefficient of the data in “test” using Fisher’s method. Present the code and the confidence interval for rho, the Pearson Correlation Coefficient. Calculations can be simplified using *tanh()* and *atanh()*. Also, submit the data to *cor.test()* and present those results as well.

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:boot':
##
##     logit
```

```
#Calculate Pearson Correlation Coefficient between testA and testB
test.rho <- cor(test$testA,test$testB)
cat("Pearson Correlation Coefficient for testA and testB:", test.rho, "\n" )
```

```
## Pearson Correlation Coefficient for testA and testB: 0.9492478
```

```
#Calculate confidence interval for the Pearson Correlation Coefficient
result4 <- r.con(test.rho,49,p=.95,twotailed=TRUE)
cat("With 95% confidence interval, the Pearson Correlation Coefficient is somewhere between ",
result4[1]," and ",result4[2],".\n\n")
```

```
## With 95% confidence interval, the Pearson Correlation Coefficient is somewhere between  0.9
113003  and  0.9712052 .
```

```
cor.test(test$testA,test$testB)
```

```
##
##  Pearson's product-moment correlation
##
## data:  test$testA and test$testB
## t = 20.69, df = 47, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9113003 0.9712052
## sample estimates:
##       cor
## 0.9492478
```

4(b)(5 points)

Bootstrapping can be used to arrive at an estimated confidence interval. The process involves resampling with replacement of rows from “test.” The first step is to randomly sample with replacement the 49 rows of “test”. Each sample will consist of 49 rows for which a sample correlation coefficient is calculated. For this purpose, the function *sample.int()* may be used to determine a sample of row numbers to be used. The function *cor()* should be used to determine the sample correlation coefficient. This step is repeated 10,000 times resulting in 10,000 sample correlation coefficients. The 10,000 calculated sample correlation coefficients are written to a vector. *quantile()* is passed this vector to calculate the 2.5% (0.025) and 97.5% (0.975) quantiles which determines the 95% Percentile Bootstrap confidence interval.

Refer to the course site library reserves and read the listing for Section 9.5 of “Mathematical Statistics with Resampling and R,” by Chihara and Hesterberg.

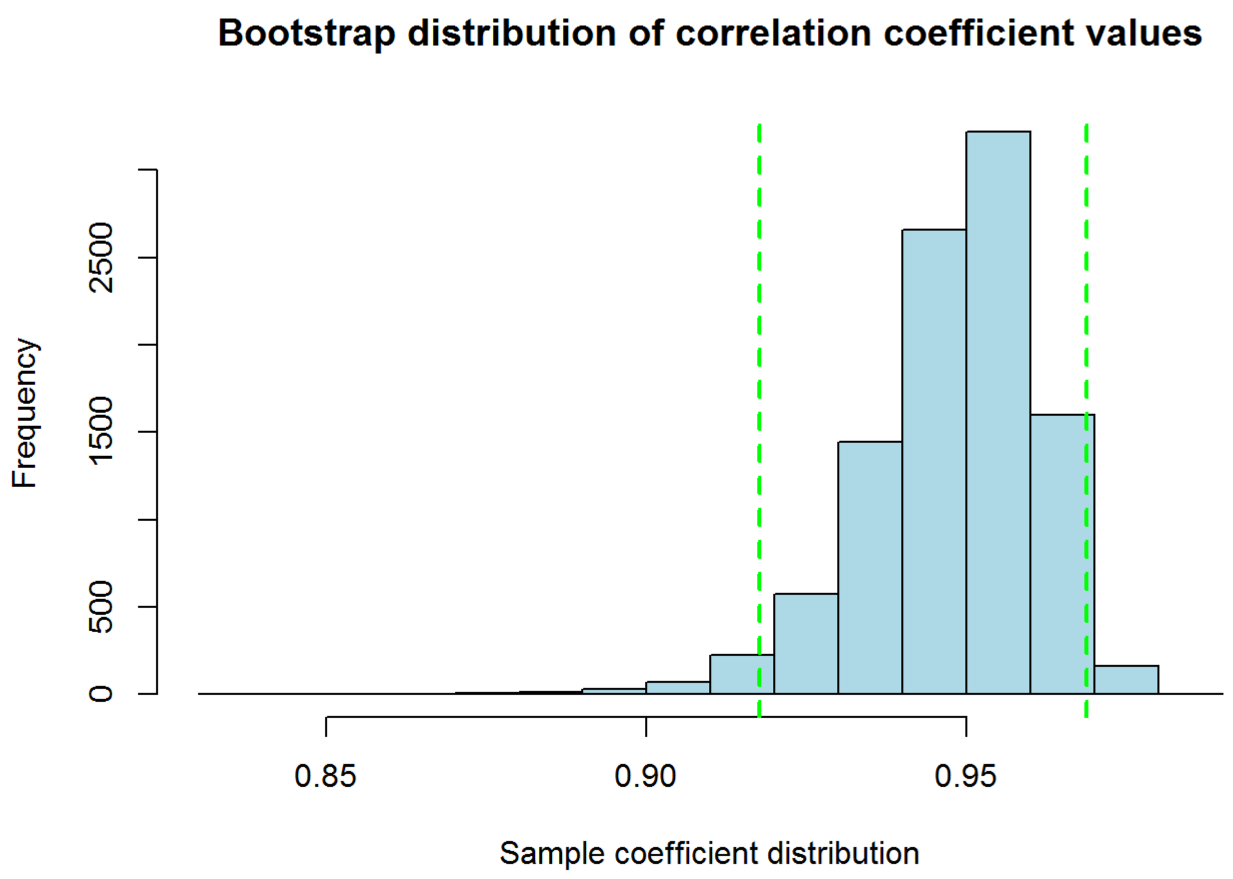
You will write code which does this work. Use *set.seed(123)*. A “for” loop may be used to repeat the sampling and correlation coefficient calculations. Plot a histogram and report a two-sided 95% confidence interval.

```
set.seed(123)

# Define number of resamples drawn.
N <- 10000

# Define vectors for storage purposes.
test.coef <- numeric(N)

# Calculate variance for each bootstrap sample of size 49.
for (i in 1:N)
{
 index <- sample.int(49,49, replace = TRUE) # sample from 1,2,...,n
test.boot <- test[index, ] # resampled data
test.coef[i] <- cor(test.boot$testA, test.boot$testB)
}

# confidence interval for sample correlation coefficient using Bootstrapping .
```

```
result5 <- round(quantile(test.coef, prob = c(0.025,0.975)), digits = 3)
cat("With 95% Percentile Bootstrap confidence interval, the population correlation coefficient
 is somewhere between ",result5[1]," and ",result5[2],".\n")
```

```
## With 95% Percentile Bootstrap confidence interval, the population correlation coefficient i
s somewhere between  0.918  and  0.969 .
```

```
hist(test.coef, main = "Bootstrap distribution of correlation coefficient values", col = "ligh
tblue",xlab = "Sample coefficient distribution")
abline(v= quantile(test.coef, probs =c(0.025,0.975)),col = "green", lty = 2, lwd = 2)
```



4(c)(5 points)

Bootstrapping can also be used to arrive at confidence intervals for regression coefficients. This process is similar to the process in 4(b). Using the current data frame, rows of “test” are randomly sampled with replacement. Each sample is passed to *lm()* and the coefficients extracted. Section 9.5 of Chihara and Hesterberg give an example R script showing how this may be accomplished.

Write code using “test” to produce histograms and 95% two-sided bootstrap confidence intervals for the intercept and slope of a simple linear regression. Please keep *set.seed(123)*. A “for” loop can be written to sample via *sample.int()*, a linear regression model fit via *lm()* and the coefficients extracted via *coef()*. Note that we pass our fitted model object to *coef()* to return the coefficients. You can use brackets after *coef()* to specify particular elements in the output. For example, *coef(model)[1]* - assuming our fitted linear model was “model” - would return the 1st element in the *coef()* output.

Present two histograms, one for the bootstrapped intercept results, and one for the bootstrapped slope results showing the

2.5% and 97.5% quantiles on each. In addition, show the location on the histograms of the estimated intercept and slope of test using *lm()* without bootstrapping.

Lastly, generate a scatter plot of the estimated bootstrap slopes versus the estimated bootstrap intercepts. There will be 10,000 points appearing in this plot, one for each bootstrap sample. Place the intercepts on the x-axis and the slopes on the y-axis.

```
set.seed(123)
# Number of resamples drawn.
N <- 10000

# Define vectors for storage purposes.
alpha.boot <- numeric(N)
beta.boot <- numeric(N)


# Calculate variance for each bootstrap sample
for (i in 1:N)
{
 index <- sample.int(49, 49, replace = TRUE) # sample from 1,2,...,n
test.boot <- test[index, ] # resampled data
test.lm <- lm(testB ~ testA, data = test.boot)
alpha.boot[i] <- coef(test.lm)[1] # new intercept
beta.boot[i] <- coef(test.lm)[2] # new slope
}

# confidence interval for bootstrapped intercept and slop  .
result6 <- round(quantile(alpha.boot, prob = c(0.025,0.975)), digits = 3)
cat("With 95% Percentile Bootstrap confidence interval, the population intercept is somewhere
between ",result6[1]," and ",result6[2],".\n")
```

```
## With 95% Percentile Bootstrap confidence interval, the population intercept is somewhere be
tween  11.953  and  18.953 .
```

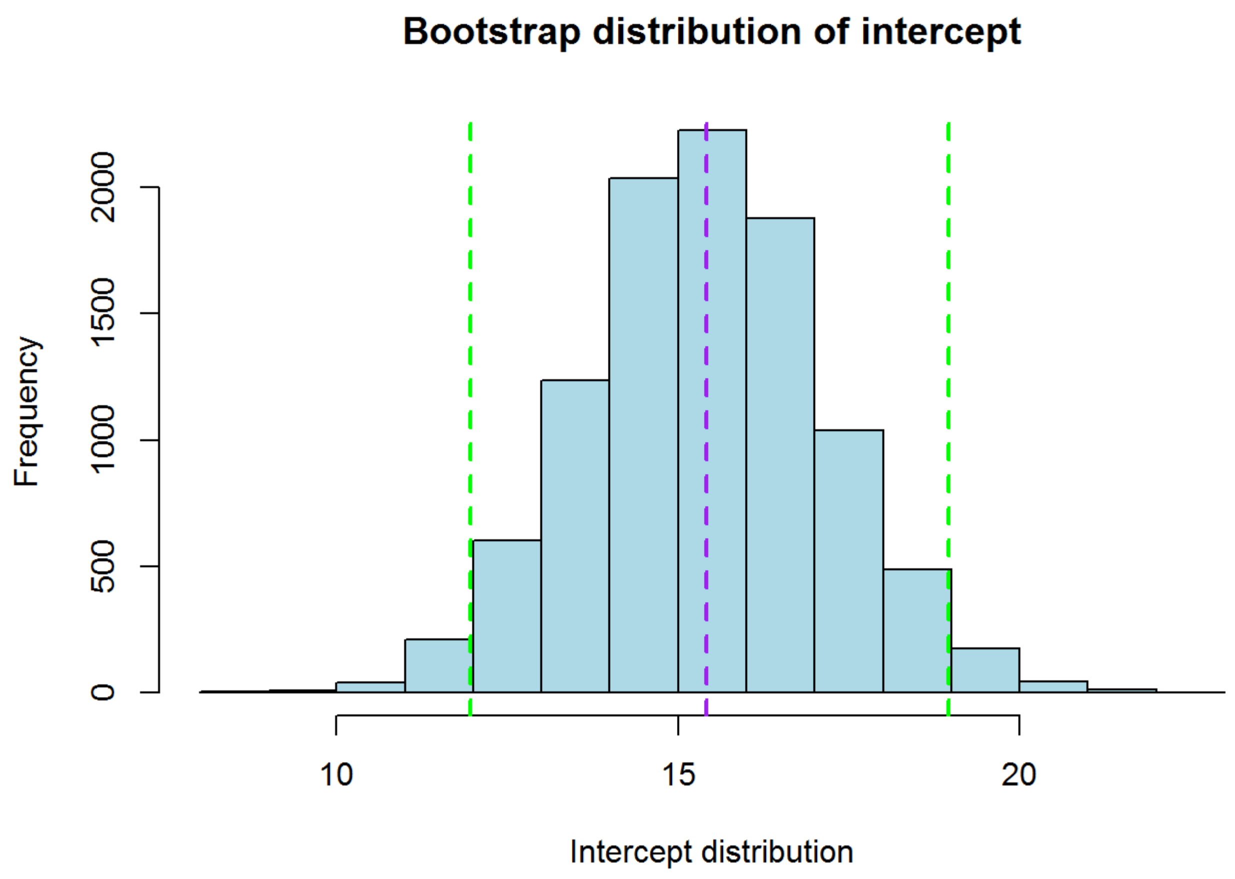
```
result7 <- round(quantile(beta.boot, prob = c(0.025,0.975)), digits = 3)
cat("With 95% Percentile Bootstrap confidence interval, the population slope is somewhere betw
een ",result7[1]," and ",result7[2],".\n")
```

```
## With 95% Percentile Bootstrap confidence interval, the population slope is somewhere betwee
n  0.643  and  0.779 .
```

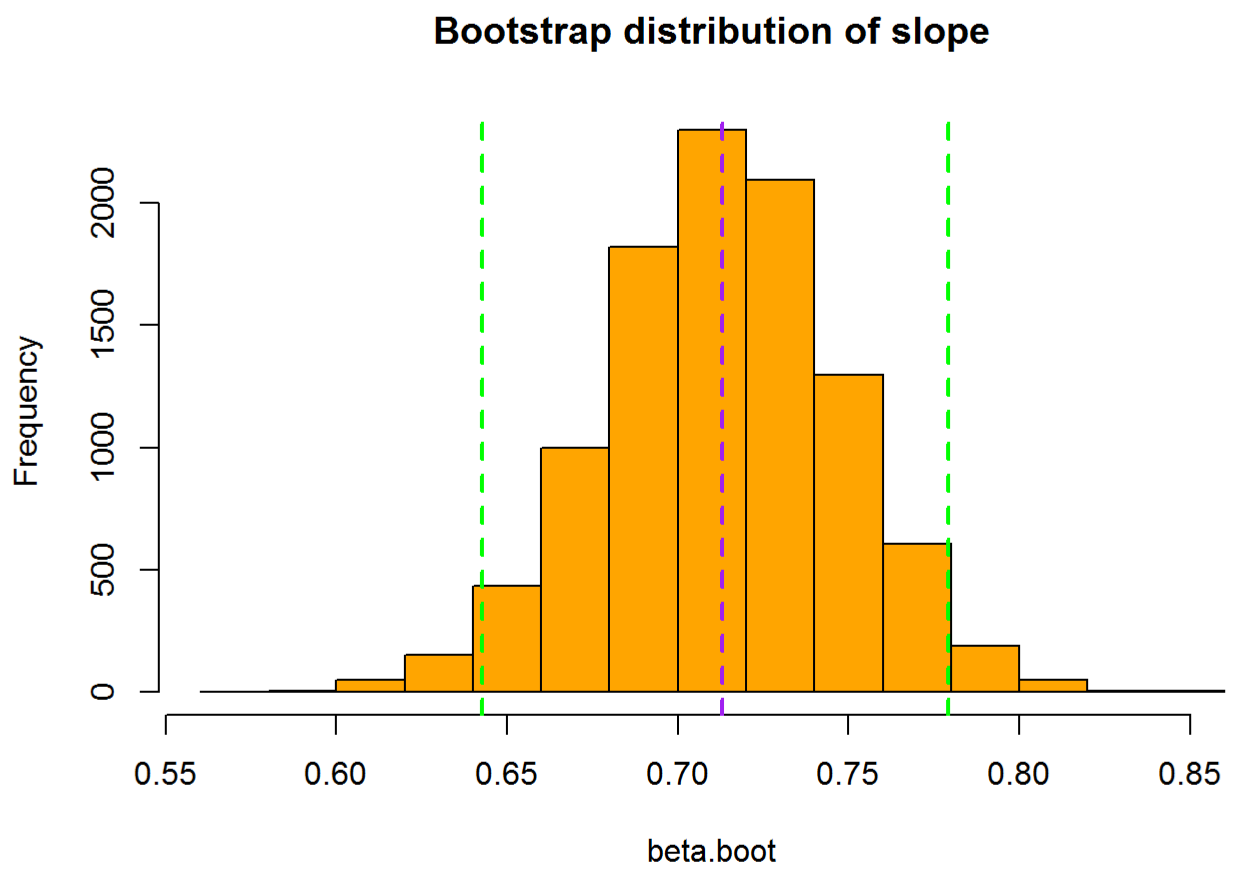
```
observed <- coef(lm(testB ~ testA, data = test))
cat("Observed values of intercept and slop :" ,observed[1]," and ",observed[2],".\n")
```

```
## Observed values of intercept and slop : 15.40882  and  0.7129485 .
```

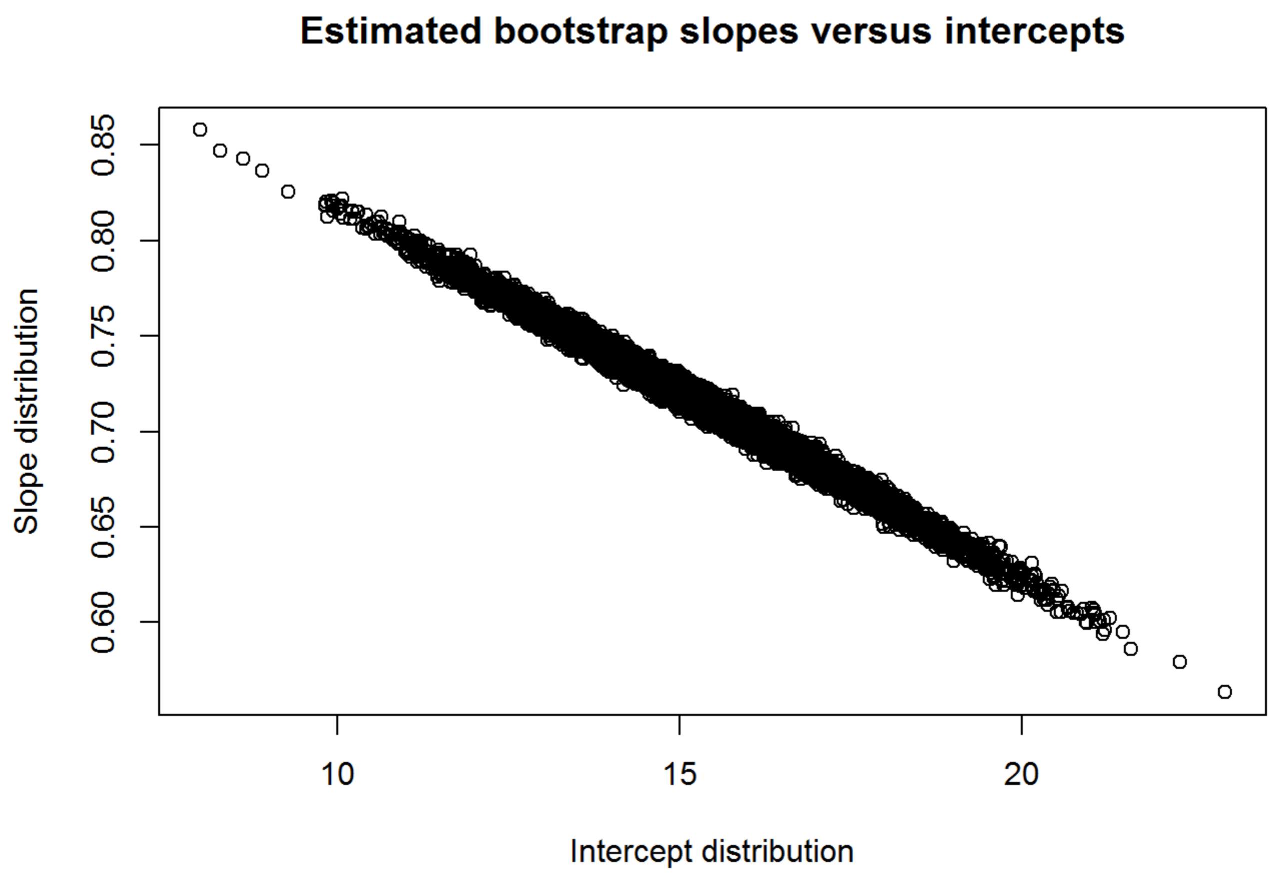
```
hist(alpha.boot, main = "Bootstrap distribution of intercept", col = "lightblue",xlab = "Inter
cept distribution")
abline(v= quantile(alpha.boot, prob = c(0.025,0.975)),col = "green", lty = 2, lwd = 2)
abline(v= observed[1],col = "purple", lty = 2, lwd = 2)
```



```
hist(beta.boot, main = "Bootstrap distribution of slope", col = "orange")
abline(v= quantile(beta.boot, prob = c(0.025,0.975)),col = "green", lty = 2, lwd = 2,xlab = "S
lope distribution")
abline(v= observed[2],col = "purple", lty = 2, lwd = 2)
```



```
#scatter plot of the estimated bootstrap slopes versus the estimated bootstrap intercepts
plot(alpha.boot,beta.boot,main = "Estimated bootstrap slopes versus intercepts",xlab = "Interc
ept distribution",ylab = "Slope distribution")
```



4(d)(2 points) What does the plot of the estimated bootstrap slopes versus the estimated bootstrap intercepts plot indicate about these estimates?

Answer: Looking at the distribution of results , both the estimates has linear and strong relationship, highly correlated and have negative association that means the slope is negative which suggests that estimated slope is decreasing with increase in estimated intercept.

The “boot” package provides a considerable capability for bootstrapping. Other discussions are given at: http://www.ats.ucla.edu/stat/r/faq/boot.htm and http://www.statmethods.net/advstats/bootstrapping.html.

This last exercise is a demonstration of a basic result. Estimates will be correlated depending on how the model is defined. Check the following link for a discussion.
https://stats.stackexchange.com/questions/104704/are-estimates-of-regression-coefficients-uncorrelated

A comparison of bootstrapping and traditional methods like with the chi-square can be accomplished using a simulation study. The mag data are only slightly skewed. As the asymmetry increases, with limited sample sizes of 50 or less, I would expect bootstrapping to provide superior confidence interval coverage compared to normal theory based statistics. The central limit theorem is always applicable. But the sample size requirements may be very large.

50 points