McCoy_Jason

Test Items (50 points total)

(1) R has probability functions available for use (see Davies, Chapter 16, and Kabacoff, Section 5.2.3). Using one distribution to approximate another is not uncommon.

(1)(a) (6 points) The normal distribution may be used to approximate the binomial distribution if np > 5 and np(1-p) > 5. Find the following binomial probabilities using *dbinom()* and *pbinom()* with a probability, p = 0.5, and n = 100. Then, estimate the same probabilities using the normal approximation **with continuity correction** and *pnorm()*. Show the numerical results of your calculations.

i. The probability of exactly 50 successes.

```
n = 100; p = 0.5
n*p
```

```
## [1] 50
```

```
n*(1-p)
```

```
## [1] 50
```

```
#Since np>5 and n(1-p)>5, normal distribution can be used to approximate binomial distribution

dbinom(50,n,p)
```

```
## [1] 0.07958924
```

```
pnorm(50.5,n*p,sqrt(n*p*(1-p)))-pnorm(49.5,n*p,sqrt(n*p*(1-p)))
```

```
## [1] 0.07965567
```

ii. The probability of fewer than 40 successes.

```
sum(dbinom(1:39,n,p))
```

```
## [1] 0.0176001
```

```
pnorm(39.5,n*p,sqrt(n*p*(1-p)))
```

```
## [1] 0.01786442
```

iii. The probability of 60 or more successes.

```
sum(dbinom(60:n,n,p))
```

```
## [1] 0.02844397
```

```
1-pnorm(59.5,n*p,sqrt(n*p*(1-p)))
```

```
## [1] 0.02871656
```

```
# For all the above, normal distribution seems to be a good approximation for binomial distrib
ution as the values are very close to each other.
```

(1)(b) (4 points) For this problem refer to Sections 5.2 and 5.3 of Business Statistics. With n = 100 and p = 0.02, use the binomial probabilities from *dbinom()* to calculate the expected value and variance for this binomial distribution using the general formula for mean and variance of a discrete distribution (To do this, you will need to use integer values from 0 to 100 as binomial outcomes along with the corresponding binomial probability). Calculate the same using the formulae np and np(1-p). Your answers should match.

```
n=100; p=0.02; x=0:100
Ex = sum(x*dbinom(x,n,p))
Vx = sum(x^2*dbinom(x,n,p))-Ex^2
Ex
```

```
## [1] 2
```

```
Vx
```

```
## [1] 1.96
```

```
n*p
```

```
## [1] 2
```

```
n*p*(1-p)
```

```
## [1] 1.96
```

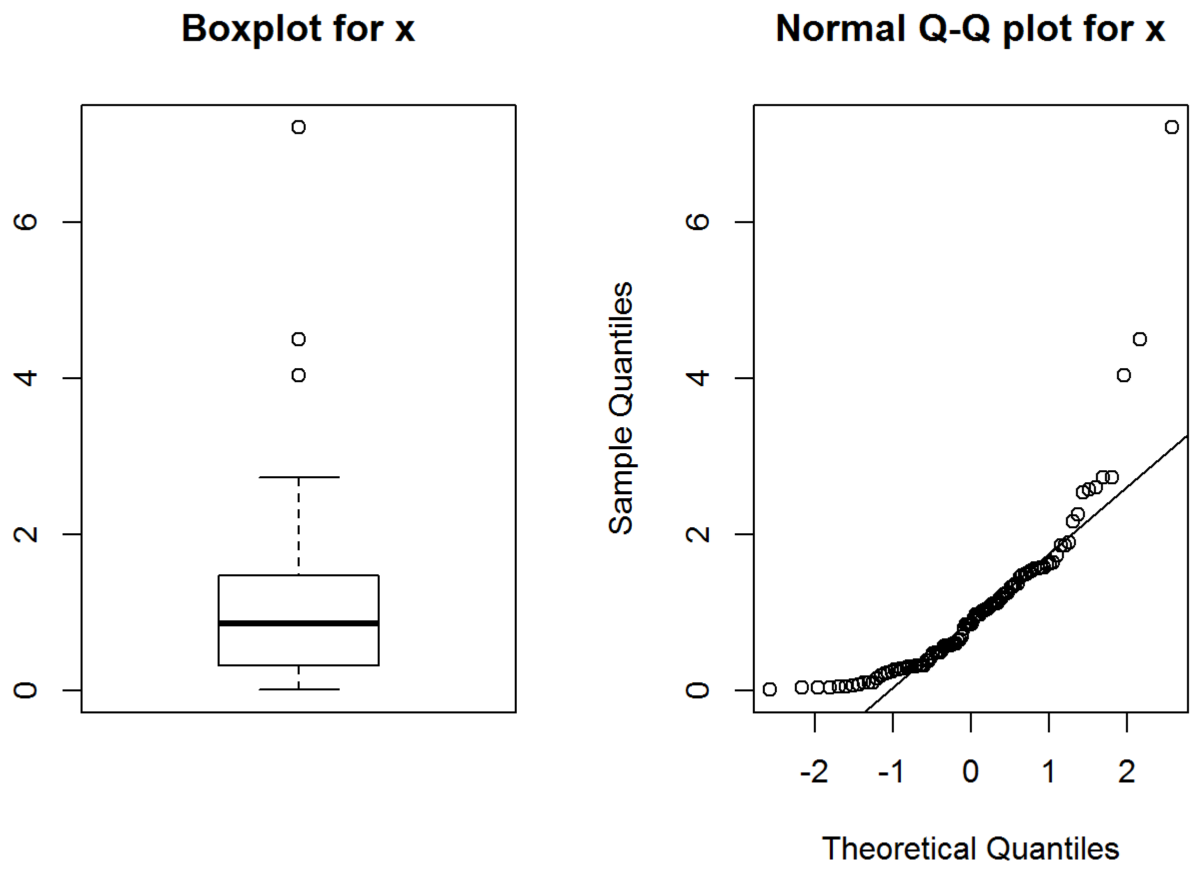
```
#Yes, the answers match!
```

10 points

(2) A recurring problem in statistics is the identification of outliers. This problem involves plotting data to display outliers, and then classiying them.

(2)(a) (5 points) Generate a random sample, “x”, of 100 values using *set.seed(123)* and *rexp(n = 100, rate = 1)*. Do not change this number. If you must draw another sample, start the process with *set.seed(123)* to maintain comparability with the answer sheet. Present “x” in side-by-side box- and QQ-plots, using *boxplot()* and *qqnorm()*, *qqline()*. Use *boxplot.stats()* and/or logical statements to identify the extreme outliers, if any. Present the extreme outlier values.

```
par(mfrow = c(1,2))
set.seed(123)
x = rexp(n=100,rate = 1)
boxplot(x,main="Boxplot for x")
qqnorm(x, main="Normal Q-Q plot for x")
qqline(x)
```



```
boxplot.stats(x)$out   boxplot.stats(x, coef = 3.0, do.conf = TRUE, do.out = TRUE)
```

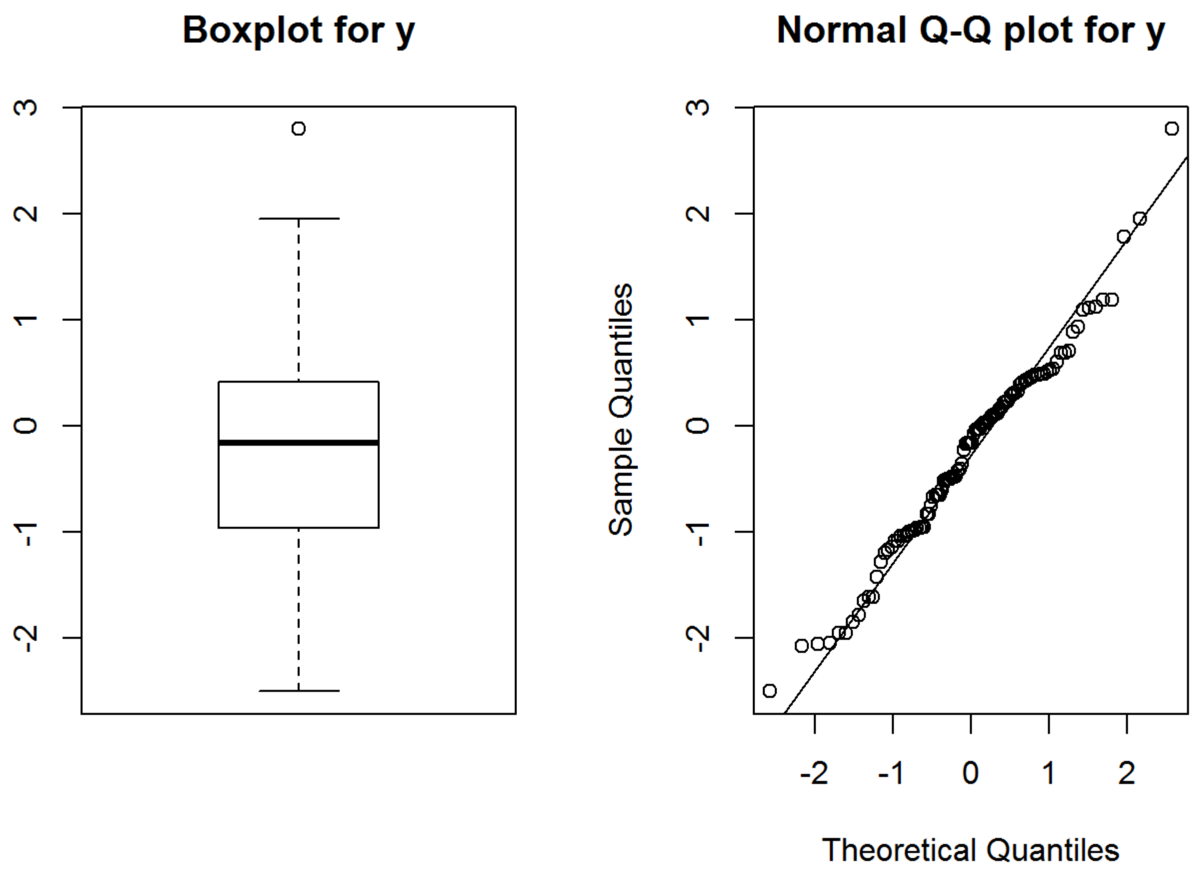
```
## [1] 4.041012 7.211008 4.498673
```

```
# x has 3 outliers which can also be seen on the boxplot as circles
```

(2)(b) (5 points) Transform the random sample, “x”, generated in (a), to form a different variable, designated “y”, using the Box-Cox Transformation: y = 3*(x^(1/3) - 1). Display the values for “y” as in (a), identify and present the extreme outliers, if any.

```
y=3*(x^(1/3)-1)
```

```
par(mfrow = c(1,2))
boxplot(y, main="Boxplot for y")
qqnorm(y, main="Normal Q-Q plot for y")
qqline(y)
```



```
boxplot.stats(y)$out
```

```
## [1] 2.795887
```

Not extreme.

```
#y has one outlier which can also be seen on the boxplot as circles
#Looking at the Q-Q plots, the points in Normal Q-Q plot for x do not seem to be lying approxi
mately along the straight line. For Normal Q-Q plot for y, the situation is better than x. So,
 the transformation from x to y helps to improve on the normality of the distribution.Also, y
has less outliers than x.
```

8 points

(3) Performing hypothesis tests using random samples is fundamental to statistical inference. The first part of this problem involves comparing two different diets. Using “ChickWeight” data available in the base R, “datasets” package, execute the following code to prepare a data frame for analysis.

```
# load "ChickWeight" dataset
data(ChickWeight)

# Create T | F vector indicating observations with Time == 21 and Diet == "1" OR "3"
```

```
index <- ChickWeight$Time == 21 & (ChickWeight$Diet == "1" | ChickWeight$Diet == "3")

# Create data frame, "result," with the weight and Diet of those observations with "TRUE" "ind
ex"" values
result <- subset(ChickWeight[index, ], select = c(weight, Diet))

# Encode "Diet" as a factor
result$Diet <- factor(result$Diet)
str(result)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':   26 obs. of  2
variables:
##  $ weight: num  205 215 202 157 223 157 305 98 124 175 ...
##  $ Diet  : Factor w/ 2 levels "1","3": 1 1 1 1 1 1 1 1 1 1 ...
```

The data frame, “result”, will have chick weights for two diets, identified as diet “1” and “3”. Use the data frame, “result,” to complete the following item.

(3)(a) (4 points) Use the “weight” data for the two diets to test the null hypothesis of equal population mean weights for the two diets. Test at the 95% confidence level with a two-sided t-test. This can be done using *t.test()* in R. Assume equal variances. Display the results of t.test().

```
#Ho: The difference between the population mean weights of the two diets is 0
#Ha: The difference between the population mean weights of the two diets is not 0
# This is NOT a paired sample test
t.test(result$weight[result$Diet==1],result$weight[result$Diet==3],alternative="two.sided",mu=
0,paired=FALSE,var.equal=TRUE,conf.level=0.95)
```

```
##
##  Two Sample t-test
##
## data:  result$weight[result$Diet == 1] and result$weight[result$Diet == 3]
## t = -3.5955, df = 24, p-value = 0.001454
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -145.67581  -39.42419
## sample estimates:
## mean of x mean of y
##    177.75    270.30
```

```
# The p-value<0.05, reject the null hypothesis. There is sufficient evidence at 5% significanc
e level to conclude that the population mean weights for the two diets are not equal.
```

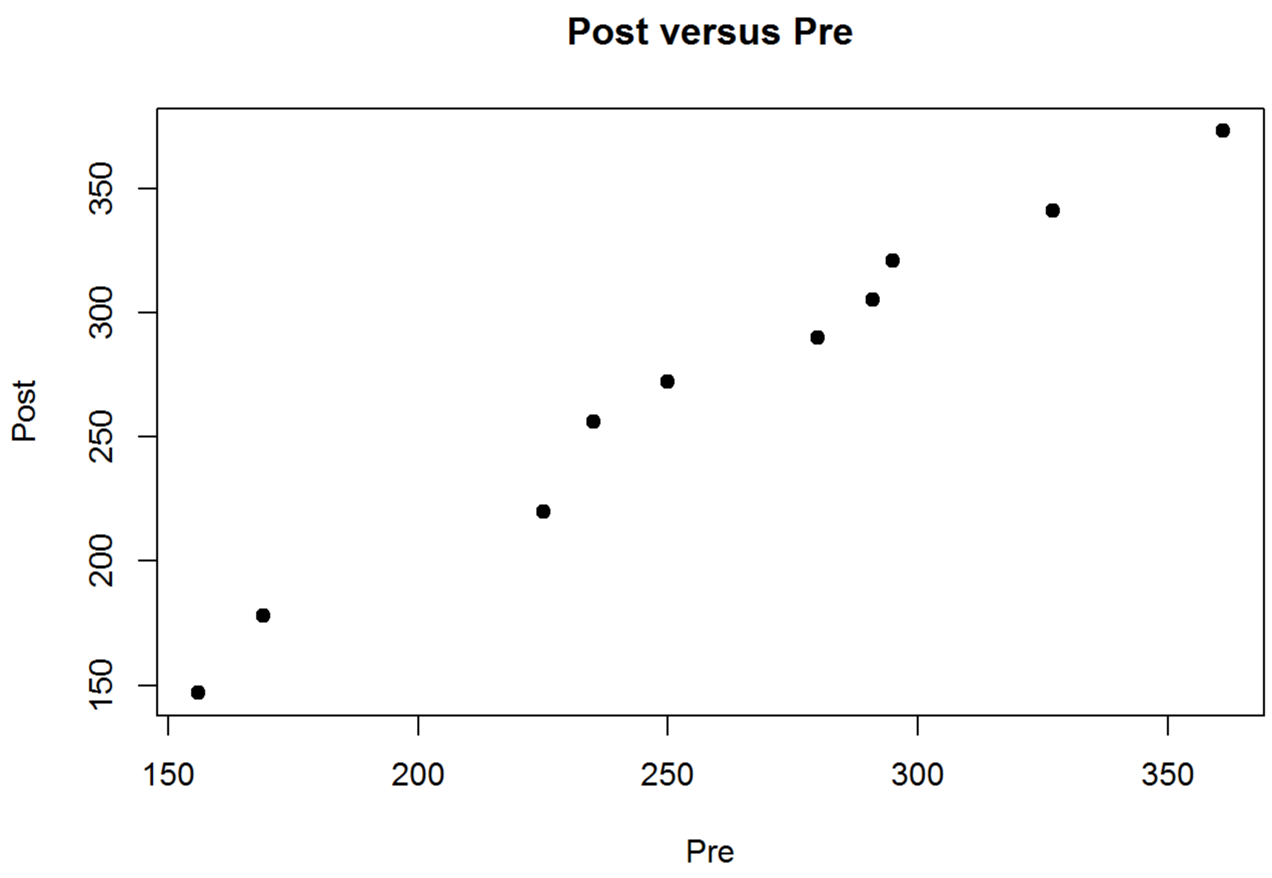
Working with paired data is another common statistical activity. The “ChickWeight” data will be used to illustrate how the weight gain from day 20 to 21 may be analyzed. Use the following code to prepare pre- and post-data from Diet == “3” for analysis.

```
# load "ChickWeight" dataset
data(ChickWeight)

# Create T | F vector indicating observations with Diet == "3"
index <- ChickWeight$Diet == "3"
```

```
# Create vector of "weight" for observations where Diet == "3" and Time == 20
pre <- subset(ChickWeight[index, ], Time == 20, select = weight)$weight

# Create vector of "weight" for observations where Diet == "3" and Time == 21
post <- subset(ChickWeight[index, ], Time == 21, select = weight)$weight
```

(3)(b) (2 points) Present a scatterplot of the variable post as a function of the variable pre. Add a title and label the variables in this scatterplot.

```
par(mfrow = c(1,1))
plot(pre,post,main="Post versus Pre",pch=19,xlab="Pre",ylab="Post")
```



3(c) (4 points) Perform a two-sided paired t-test at the 95% confidence level to test the null hypothesis of zero average weight gain from day 20 to 21. Calculate and present a two-sided, 95% confidence interval for the average weight gain from day 20 to day 21. Write the code for the paired t-test and for determination of the confidence interval endpoints. **Do not use *t.test()* although you may check your answers using this function.** Present the resulting test statistic value, critical value, p-value and confidence interval.

```
#Ho: There is zero average weight gain from day 20 to day 21
#Ha: The average weight gain from day 20 to day 21 is not zero.
# This is a paired sample test

x = post-pre
tstat = (mean(x)-0)*sqrt(length(x))/sd(x)
```

```
pvalue= 2*(1-pt(tstat, df=length(x)-1))
tstat
```

```
## [1] 3.225267
```

```
pvalue
```

```
## [1] 0.01040122
```

```
# The p-value<0.05, reject the null hypothesis. There is sufficient evidence at 5% significanc
e level to conclude that the average weight gain from day 20 to day 21 is not zero.

#Confidence interval
lower_bound = mean(x) - qt(0.975, length(x)-1)*sd(x)/sqrt(length(x))
lower_bound
```

```
## [1] 3.4042
```

```
upper_bound = mean(x) + qt(0.975, length(x)-1)*sd(x)/sqrt(length(x))
upper_bound
```

```
## [1] 19.3958
```

```
#Check:
t.test(post,pre,alternative="two.sided",mu=0,paired=TRUE,conf.level=0.95)
```

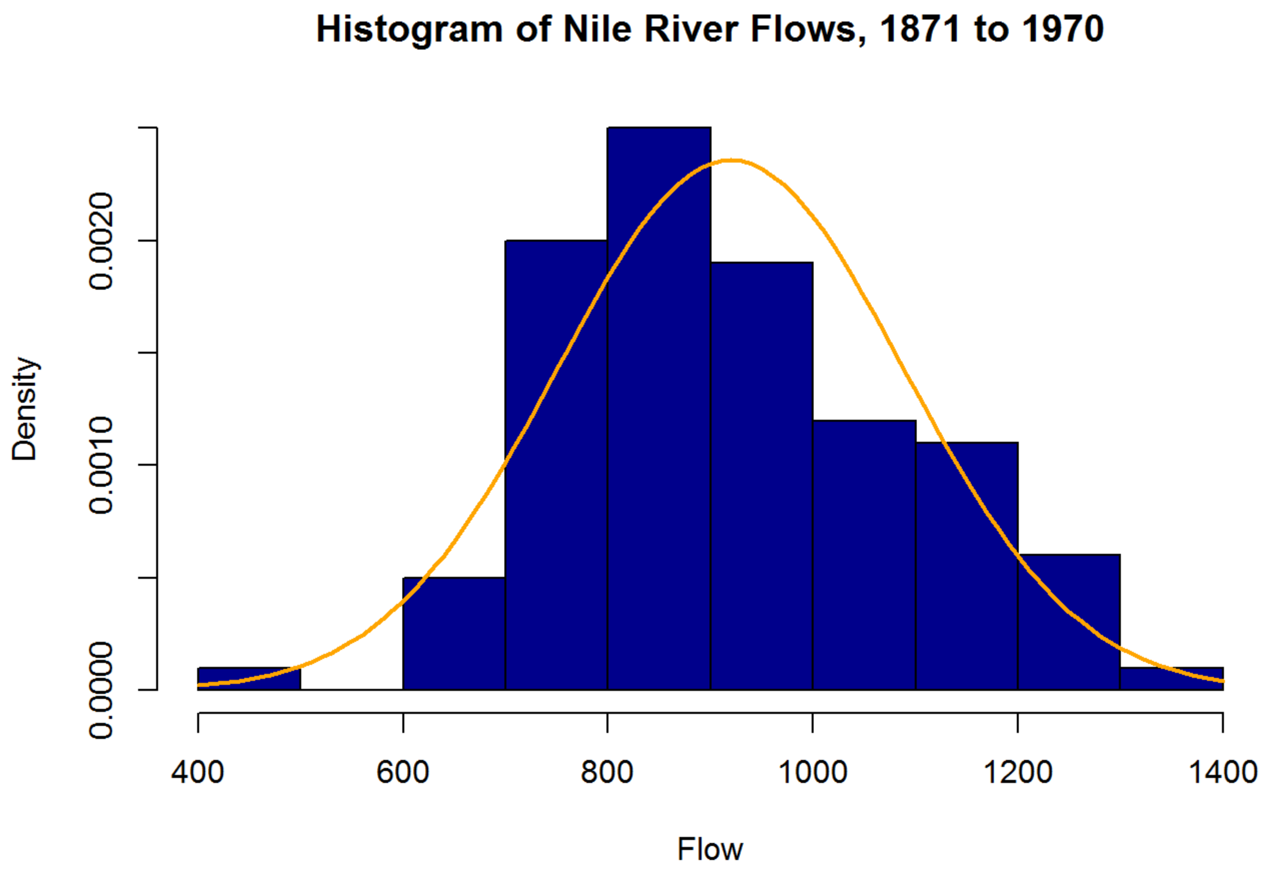
```
##
##  Paired t-test
##
## data:  post and pre
## t = 3.2253, df = 9, p-value = 0.0104
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   3.4042 19.3958
## sample estimates:
## mean of the differences
##                    11.4
```

```
#The answers match exactly!
```

10 points

(4) Statistical inference depends on using a sampling distribution for a statistic in order to make confidence statements about unknown population parameters. The Central Limit Theorem is used to justify use of the normal distribution as a sampling distribution for statistical inference. Using Nile River flow data from 1871 to 1970, this problem demonstrates sampling distribution convergence to normality. Use the code below to prepare the data.

```r
data(Nile)
m <- mean(Nile)
std <- sd(Nile)

x <- seq(from = 400, to = 1400, by = 1)
hist(Nile, freq = FALSE, col = "darkblue", xlab = "Flow",
     main = "Histogram of Nile River Flows, 1871 to 1970")
curve(dnorm(x, mean = m, sd = std), col = "orange", lwd = 2, add = TRUE)
```



(4)(a) (3 points) Using Nile River flow data and the “moments” package, calculate skewness and kurtosis. Present side-by-side displays using *qqnorm()*, *qqline()* and *boxplot()*; i.e *par(mfrow = c(1, 2))*. Add features to these displays as you choose.

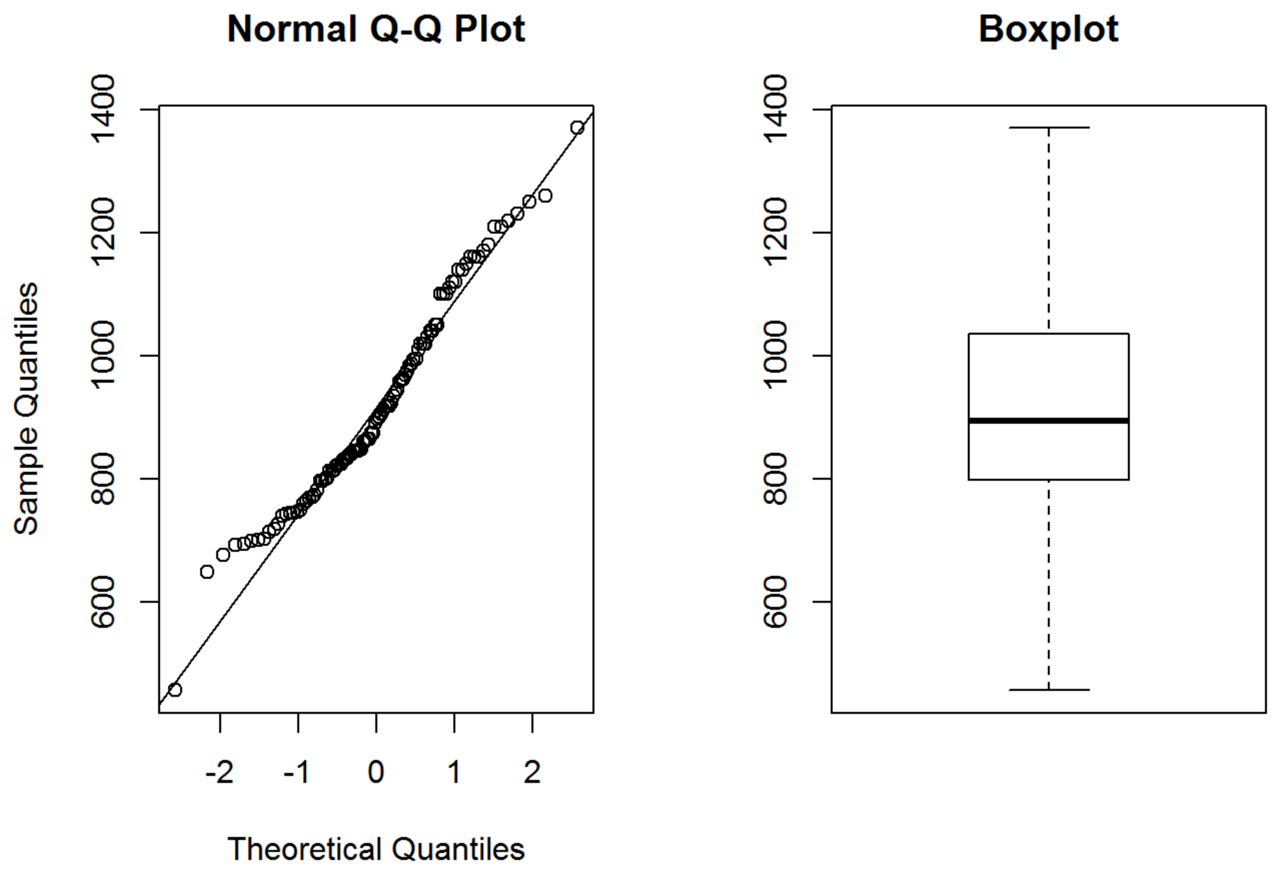
```r
data(Nile)
library(moments)
skewness(Nile)
```

```
## [1] 0.3223697
```

```r
#Skewness is greater than 0, implies positively skewed
kurtosis(Nile)
```

```
## [1] 2.695093
```

```
#Kurtosis is 2.695093 which is less than 3, hence distribution is platykurtic.
par(mfrow = c(1, 2))
qqnorm(Nile, main="Normal Q-Q Plot")
qqline(Nile)
boxplot(Nile, main="Boxplot")
```



```
#Boxplot doesn't show the presence of any outliers. Normal Q-Q plot shows that points are appr
oximately around the striaght line.
```

(4)(b) (3 points) Using *set.seed(124)* and the Nile data, generate 1000 random samples of size n = 16, with replacement. For each sample drawn, calculate and store the sample mean. This will require a for-loop and use of the *sample()* function. Label the resulting 1000 mean values as “sample1”. **Repeat these steps using *set.seed(127)* - a different “seed” - and samples of size n = 64.** Label these 1000 mean values as “sample2”. Compute and present the mean value, sample standard deviation and sample variance for “sample1” and “sample2”.

```
set.seed(124)
sample1=numeric(1000)
for(i in 1:1000)
{
sample1[i]=mean(sample(Nile,16,replace=TRUE))
}

set.seed(127)
sample2=numeric(1000)
```

```
for(i in 1:1000)
{
sample2[i]=mean(sample(Nile,64,replace=TRUE))
}

#Mean
mean(sample1)
```

```
## [1] 918.7349
```

```
mean(sample2)
```

```
## [1] 919.4785
```

```
#Sample standard deviation
sd(sample1)
```

```
## [1] 41.57159
```

```
sd(sample2)
```

```
## [1] 20.07329
```

```
#Sample Variance
var(sample1)
```

```
## [1] 1728.197
```

```
var(sample2)
```

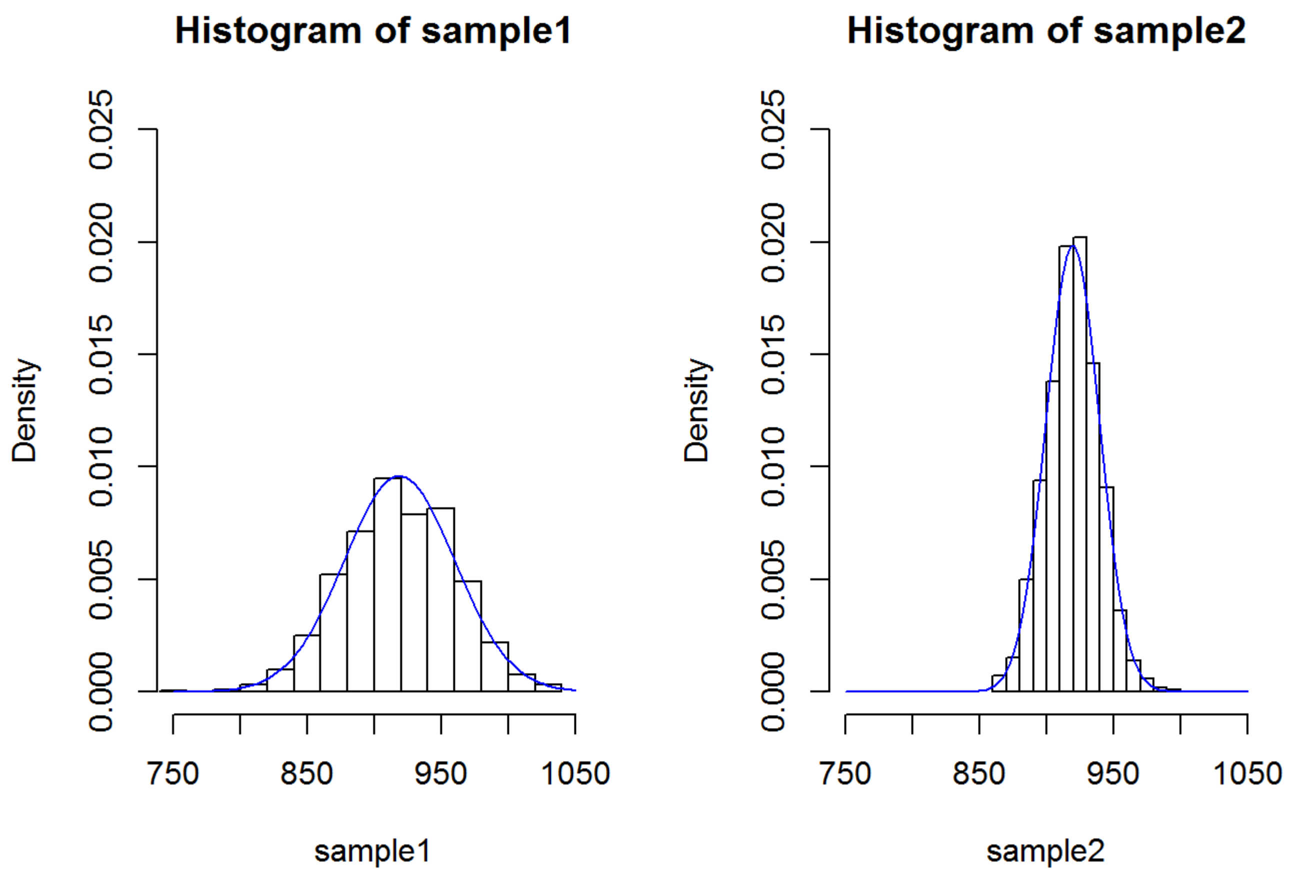
```
## [1] 402.937
```

```
# Note the sample standard deviation and variance is less for Sample2 as compared to Sample1.
This is because Sample2 has mean values of sample sizes of n=64 which is greater than n=16 for
 sample1.
```

(4)(c) (4 points) Using “sample1” and “sample2”, present side-by-side histograms with the normal density curve superimposed (use *par(mfrow = c(1,2))*). To prepare comparable histograms it will be necessary to use “freq = FALSE” and to maintain the same x-axis with “xlim = c(750, 1050)”, and the same y-axis with “ylim = c(0, 0.025).” **To superimpose separate density functions, you will need to use the mean and standard deviation for each “sample” - each histogram - separately.**

```
par(mfrow = c(1,2))
hist(sample1,freq = FALSE,xlim = c(750,1050),ylim=c(0,0.025))
curve(dnorm(x, mean=mean(sample1), sd=sd(sample1)), add=TRUE,col="blue")
```

```
hist(sample2,freq = FALSE,xlim = c(750,1050),ylim=c(0,0.025))
curve(dnorm(x, mean=mean(sample2), sd=sd(sample2)), add=TRUE,col="blue")
```



```
#It can be clearly seen from the histogram that the variance/standard deviation is less for sa
mple2 distribution than for sample1 distribution.
```

10 points

(5) This problem deals with 2 x 2 contingency table analysis. This is an example of categorical data analysis (see Kabacoff, pp. 145-151). The following graphical method, in conjunction with the chi-square test, are ways to screen data for variables exhibiting monotonic association. This method is one of several presented by Quenouille in his book “Rapid Statistical Calculations”.

The “Seatbelts” dataset contains monthly road casualties in Great Britain, 1969 to 1984. Use the code below to organize the data and generate two factor variables: “killed” and “month”. These variables will be used for contingency table analysis.

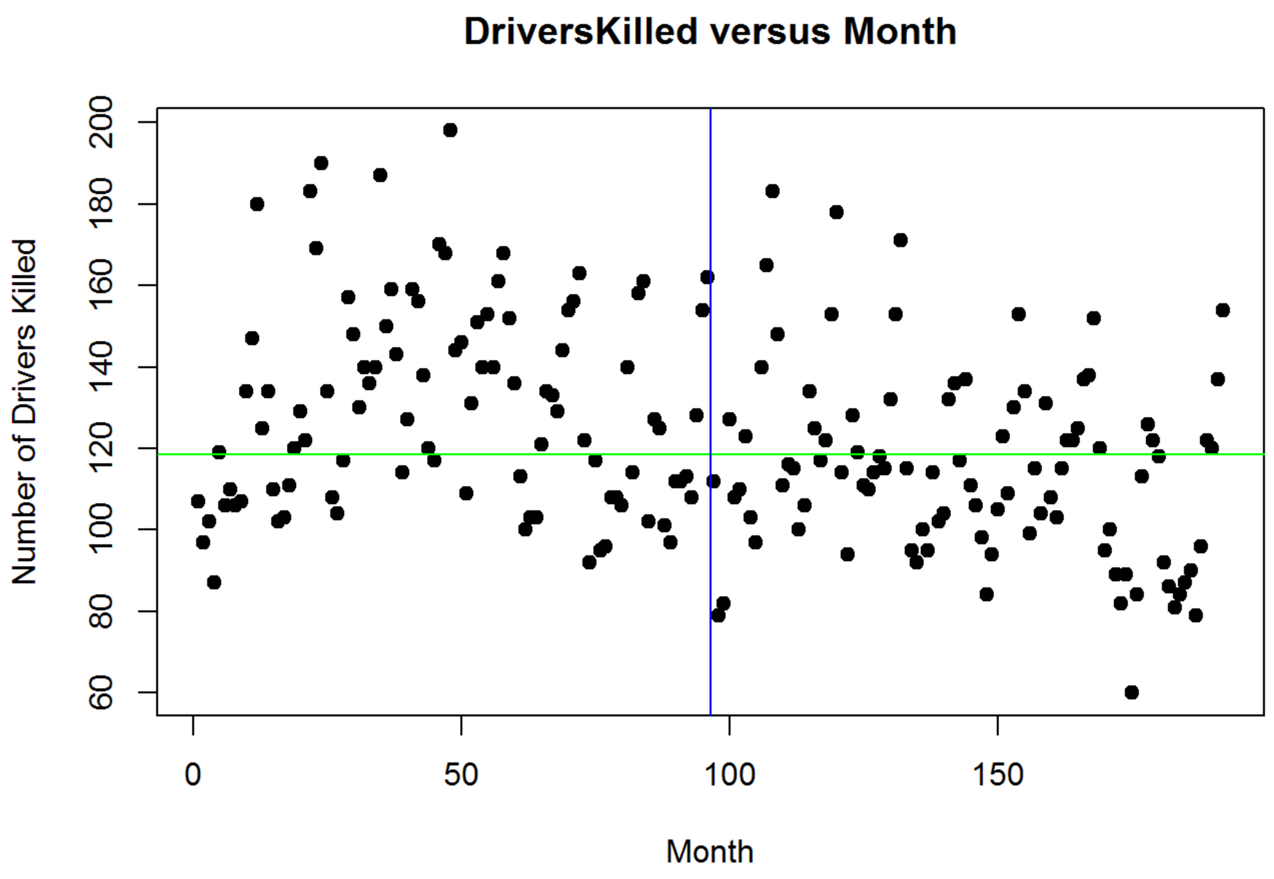
```
data(Seatbelts)
Seatbelts <- as.data.frame(Seatbelts)

Seatbelts$Month <- seq(from = 1, to = nrow(Seatbelts))
Seatbelts <- subset(Seatbelts, select = c(DriversKilled, Month))
summary(Seatbelts)
```

```
##  DriversKilled       Month
##  Min.   : 60.0   Min.   :  1.00
##  1st Qu.:104.8   1st Qu.: 48.75  
##  Median :118.5   Median : 96.50  
##  Mean   :122.8   Mean   : 96.50  
##  3rd Qu.:138.0   3rd Qu.:144.25  
##  Max.   :198.0   Max.   :192.00
```

```
killed <- factor(Seatbelts$DriversKilled > 118.5, labels = c("below", "above"))
month <- factor(Seatbelts$Month > 96.5, labels = c("below", "above"))
```

(5)(a) (3 points) Using “Seatbelts,” generate a scatterplot of the variables DriversKilled versus Month. This is a time series, and Seatbelts$Month should be on the horizontal axis. Show vertical and horizontal lines to indicate the median of each variable. Label as desired.

```
par(mfrow = c(1,1))
plot(Seatbelts$Month,Seatbelts$DriversKilled, main="DriversKilled versus Month", xlab="Month",
 ylab="Number of Drivers Killed",pch=19)
abline(h=median(Seatbelts$DriversKilled), col="green")
abline(v=median(Seatbelts$Month), col="blue")
```



```
#The green and the blue lines on the graph represent median values.
```

(5)(b) (2 points) A chi-square test of independence will be used (see Black, Section 16.2) to test the null hypothesis that the factor variables, “killed” and “month”, are independent. Use *table()* to generate a 2 x 2 contingency table showing the fatality count classified by “killed” and “month”. Use the **uncorrected** *chisq.test()* to test the null hypothesis that “killed” and "month" are independent at the 95% confidence level. Present these results.

```
table(killed, month)
```

```
##        month
## killed  below above
##   below    37    59
##   above    59    37
```

```
chisq.test(table(killed, month), correct = FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(killed, month)
## X-squared = 10.083, df = 1, p-value = 0.001496
```

```
# 95% confidence level implies alpha=0.05. Since the p-value of 0.001496 < 0.05, we reject the
 null hypothesis of independence between months and number of drivers killed. The conclusion i
s that there is sufficient evidence at 5% significance level to say that "killed" and "month"
variables are not independent.
```

(5)(c) (5 points) Add margins to the contingency table from (b) using the function *addmargins()*. Write a function that computes the uncorrected Pearson Chi-squared statistic based on the a 2 x 2 contingency table with margins added (check Davies, Section 11.1.1, pp. 216-219, and Kabacoff, Section 20.1.3, pp. 473-474). Submit this augmented table to the function you have written. Compare the result with (b). You should be able to duplicate the X-squared value (chi-squared) and *p*-value. Present both.

The statements shown below calculate the expected value for each cell in an augmented contingency table with margins added. Using these statements, the Pearson Chi-square statistic may be calculated. Other approaches are acceptable.

e11 <- x[3, 1] * x[1, 3] / x[3, 3]

e12 <- x[3, 2] * x[1, 3] / x[3, 3]

e21 <- x[3, 1] * x[2, 3] / x[3, 3]

e22 <- x[3, 2] * x[2, 3] / x[3, 3]

```
addmargins(table(killed, month))
```

```
##        month
## killed  below above Sum
##   below    37    59  96
##   above    59    37  96
##   Sum      96    96 192
```

```
chisq.test.sample <- function(x) {
e11 = x[3, 1] * x[1, 3] / x[3, 3]
e12 = x[3, 2] * x[1, 3] / x[3, 3]
e21 = x[3, 1] * x[2, 3] / x[3, 3]
e22 = x[3, 2] * x[2, 3] / x[3, 3]
Xsq = ((x[1, 1] - e11)^2)/e11 + ((x[1, 2] - e12)^2)/e12 + ((x[2, 1] - e21)^2)/e21 + ((x[2, 2]
- e21)^2)/e21
p = pchisq(Xsq,df=1,lower.tail = FALSE)
list(Xsq,p)
}
chisq.test.sample(addmargins(table(killed, month)))
```

```
## [[1]]
## [1] 10.08333
##
## [[2]]
## [1] 0.001496164
```

```
#The X-squared value (chi-squared) and p-value are exactly the same as what was calculated pre
viously.
```

10 points

48 points