John Barker, Programming with R Program Assignment 2

```
##### (1) R has probability functions available for use (Kabacoff, Section 5.2.3).

# Using one distribution to approximate another is not uncommon.

# (1)(a) (6 points) The Poisson distribution may be used to approximate the binomial distribution

# if n > 20 and np < 7. Estimate the following binomial probabilities using

# *dpois()* and *ppois()* with probability p = 0.05, and n = 100. Then, estimate

# the same probabilities using *dbinom()* and *pbinom()*.

# Show the numerical results of your calculations.


# (i) The probability of exactly 0 successes.


set.seed(1234)

dpois(0,5)

ppois(0,5)


dbinom(0,100,.05)

pbinom(0, 100,.05)
```

output

```
> set.seed(1234)
> dpois(0,5)
[1] 0.006737947
> ppois(0,5)
[1] 0.006737947
>
> dbinom(0,100,.05)
[1] 0.005920529
> pbinom(0, 100,.05)
[1] 0.005920529
```

#(ii) The probability of fewer than 6 successes.

```
set.seed(1234)

dpois(5,5)

ppois(5,5)-ppois(4,5)

dbinom(5,100,.05)

pbinom(5,100,.05)-pbinom(4,100,.05)
```

output

```
> set.seed(1234)
> dpois(5,5)
[1] 0.1754674
> ppois(5,5)-ppois(4,5)
[1] 0.1754674
> dbinom(5,100,.05)
[1] 0.1800178
> pbinom(5,100,.05)-pbinom(4,100,.05)
[1] 0.1800178
```

(1)(b) (3 points) Generate side-by-side barplots using *par(mfrow = c(1,2))* or

grid.arrange(). The left barplot will show Poisson probabilties for outcomes

ranging from 0 to 10. The right barplot will show binomial probabilities for

outcomes ranging from 0 to 10. Use p = 0.05 and n = 100. Title each plot,

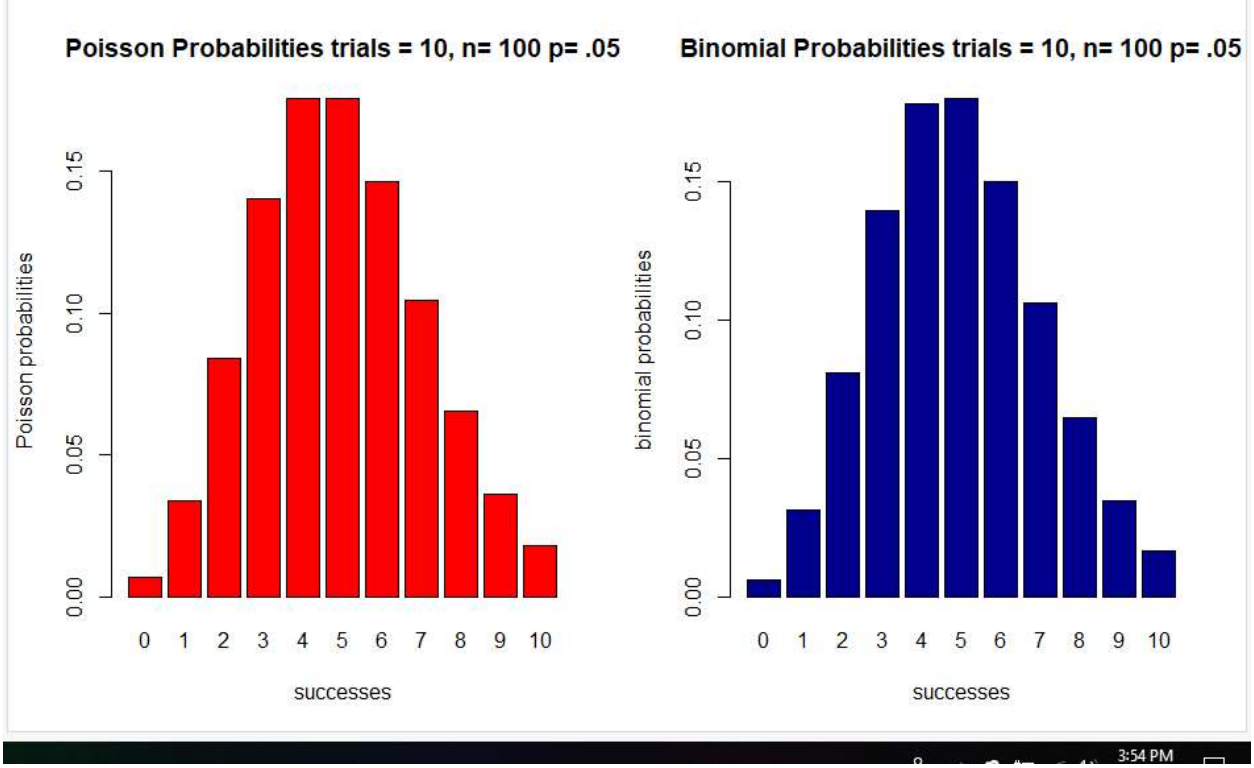
present in color and assign names to the bar; i.e. x-axis value labels.

```
n=100

p=.05

lambda<-n*p

trials <- c(0:10)

successes<-factor(trials)

pprobabilities <- dpois(trials,lambda = lambda)

bprobabilities <- dbinom(trials,size=n,prob=p)
```

```
par(mfrow=c(1,2))
barplot(pprobabilities, names.arg = successes, xlab = "successes", ylab = "Poisson probabilities",
      main = "Poisson Probabilities trials = 10, n= 100 p= .05", col="red")
barplot(bprobabilities, names.arg = successes, xlab = "successes", ylab = "binomial probabilities",
      main = "Binomial Probabilities trials = 10, n= 100 p= .05", col = "darkblue")
par(mfrow=c(1,1))
```

output

```
n=100
> p=.05
> lambda<-n*p
> trials <- c(0:10)
> successes<-factor(trials)
>
>
> pprobabilities <- dpois(trials,lambda = lambda)
> bprobabilities <- dbinom(trials,size=n,prob=p)
>
> par(mfrow=c(1,2))
> barplot(pprobabilities, names.arg = successes, xlab = "successes", ylab = "Poisson probabilities",
+       main = "Poisson Probabilities trials = 10, n= 100 p= .05", col="red")
> barplot(bprobabilities, names.arg = successes, xlab = "successes", ylab = "binomial probabilities",
+       main = "Binomial Probabilities trials = 10, n= 100 p= .05", col = "darkblue")
> par(mfrow=c(1,1))
```



(1)(c) For this problem, refer to Sections 5.2 of Business Statistics. A

discrete random variable has outcomes: 0, 1, 2, 3, 4, 5, 6. The

corresponding probabilities in sequence with the outcomes are:

0.215, 0.230, 0.240, 0.182, 0.130, 0.003, 0.001. In other words,

the probabilty of obtaining "0" is 0.215.

(i) (3 points) Calculate the expected value and variance for this 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 6 as outcomes along with the

corresponding probabilities. Round your answer to 2 decimal places.

```
discretes<-c(0,1,2,3,4,5,6)
discprobs<-c(0.215, 0.230, 0.240, 0.182, 0.130, 0.003, 0.001)
discmean<-round(sum(discretes*discprobs),digits=2)
discmean
```

```
discvar<-round(sum(((discretes-discmean)**2)*discprobs),digits=2)

discvar
```

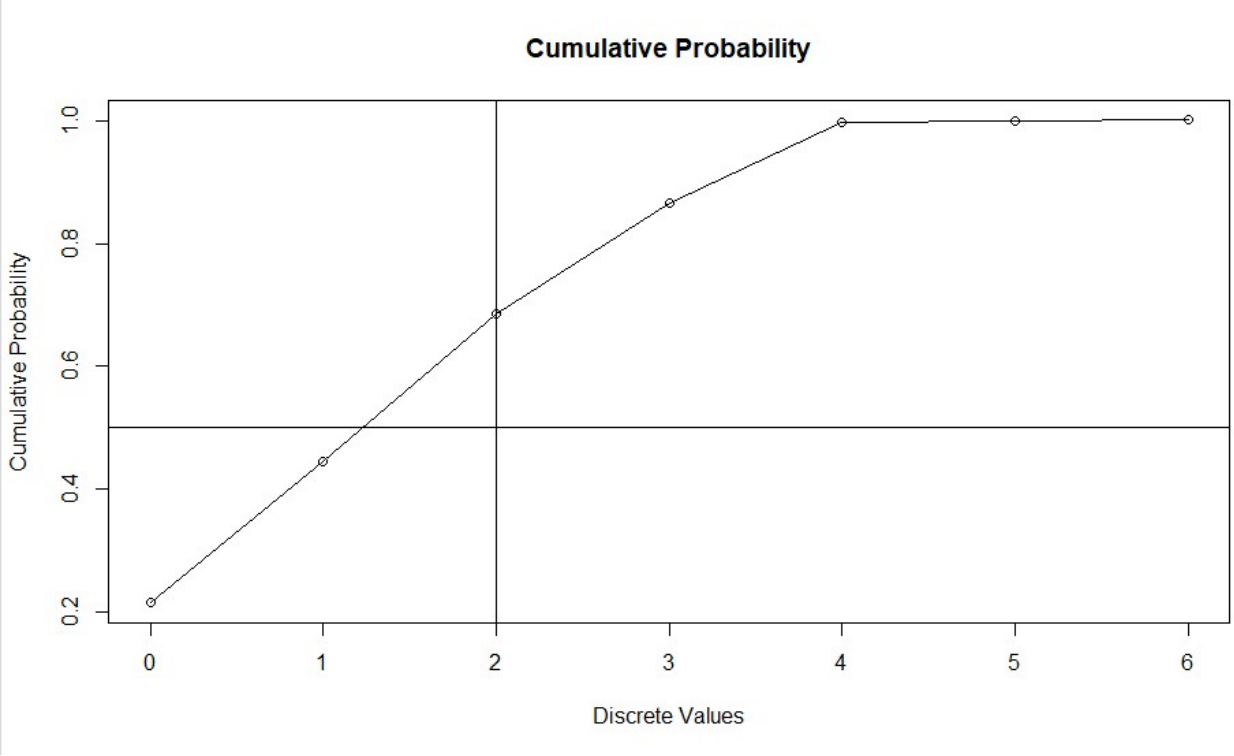
output

```
> discretes<-c(0,1,2,3,4,5,6)
> discprobs<-c(0.215, 0.230, 0.240, 0.182, 0.130, 0.003, 0.001)
> discmean<-round(sum(discretes*discprobs),digits=2)
> discmean
[1] 1.8
> discvar<-round(sum(((discretes-discmean)**2)*discprobs),digits=2)
> discvar
[1] 1.79
```

```
# (ii) (3 points) Use the *cumsum()* function and plot the cumulative

# probabilties versus the corresponding outcomes.  Detemine the value

# of the median for this distribution and show on this plot.


disccum<-c(cumsum(discprobs))

disccum

index<-0

median<-0

median

plot(discretes, disccum,

     main="Cumulative Probability",

     xlab="Discrete Values",

     ylab="Cumulative Probability")

lines(discretes, disccum)

abline(v=2)

abline(h=.5)

disccum
```

output

```
> disccum<-c(cumsum(discprobs))
> disccum
[1] 0.215 0.445 0.685 0.867 0.997 1.000 1.001
> index<-0
> median<-0
> median
[1] 0
> plot(discretes, disccum,
+         main="Cumulative Probability",
+         xlab="Discrete Values",
+         ylab="Cumulative Probability")
> lines(discretes, disccum)
> abline(v=2)
> abline(h=.5)
> disccum
[1] 0.215 0.445 0.685 0.867 0.997 1.000 1.001
```



Section 2: (15 points)

(2) Conditional probabilities appear in many contexts and, in particular,

are used by Bayes' Theorem. Correlations are another means for evaluating dependency

between variables. The dataset "faithful"" is part of the "datasets" package

and may be loaded with the statement *data(faithful)*. It contains 272 observations

of 2 variables; waiting time between eruptions (in minutes) and the duration of the

eruption (in minutes) for the Old Faithful geyser in Yellowstone National Park.

#(2)(a) (3 points) Load the "faithful" dataset and present summary statistics and a histogram

# of waiting times. Additionally, compute the empirical conditional probability of an

eruption less than 3.0 minutes, if the waiting time exceeds 70 minutes.

```
data(faithful)

x<-faithful

summary(x)

a<-hist(x$waiting)

pa<-sum(x$waiting>70)/nrow(x)

pa

(Nab <- nrow(x[which(x$waiting>70 & x$eruptions<3),]))

(Nb <- sum(hist(x$eruptions,plot=F)$counts[hist(x$eruptions,plot=F)$mids<3]))

(Pab <- Nab / Nb)
```

Output

```
> data(faithful)
> x<-faithful
>
> summary(x)
   eruptions        waiting
 Min.   :1.600   Min.   :43.0
 1st Qu.:2.163   1st Qu.:58.0
 Median :4.000   Median :76.0
 Mean   :3.488   Mean   :70.9
 3rd Qu.:4.454   3rd Qu.:82.0
 Max.   :5.100   Max.   :96.0
>
> a<-hist(x$waiting)
> pa<-sum(x$waiting>70)/nrow(x)
> pa
[1] 0.6066176
> (Nab <- nrow(x[which(x$waiting>70 & x$eruptions<3),]))
[1] 1
> (Nb <- sum(hist(x$eruptions,plot=F)$counts[hist(x$eruptions,plot=F)$mids<3]
))
[1] 97
>
> (Pab <- Nab / Nb)
[1] 0.01030928
```

```
#(i) (3 points) Identify any observations in "faithful" for which the waiting
time
# exceeds 70 minutes and the eruptions are less than 3.0 minutes.  List and s
how any
# such observations in a distinct color on a scatterplot of all eruption (ver
tical axis)
# and waiting times (horizontal axis). Include a horizontal line at eruption
= 3.0,
# and a vertical line at waiting time = 70.  Add a title and appropriate text
.
```

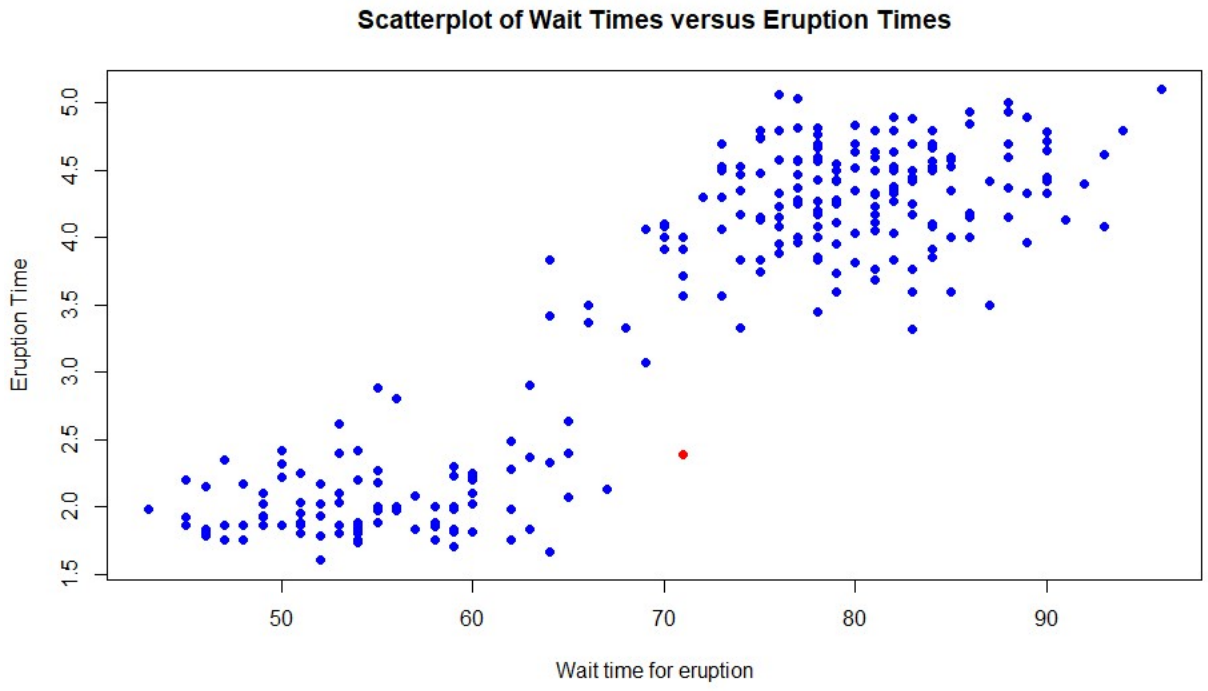
```
pabt<-subset(x, subset = waiting>=70 & eruptions <= 3)

pabt

summary(pabt)

str(pabt)

color = rep(NA, length=length(x))

color[which(x$eruptions<3 & x$waiting > 70)] = "red"

color[which(!(x$eruptions<3 & x$waiting > 70))] = "blue"

color

plot(x$waiting,x$eruptions, col = color, pch=16, xlab = "Wait time for eruption",

    ylab = "Eruption Time", main = "Scatterplot of Wait Times versus Eruption Times")
```

output

```
> pabt<-subset(x, subset = waiting>=70 & eruptions <= 3)
> pabt
    eruptions waiting
211     2.383      71
> summary(pabt)
   eruptions        waiting
 Min.   :2.383   Min.   :71
 1st Qu.:2.383   1st Qu.:71
 Median :2.383   Median :71
 Mean   :2.383   Mean   :71
 3rd Qu.:2.383   3rd Qu.:71
 Max.   :2.383   Max.   :71
> str(pabt)
'data.frame':  1 obs. of  2 variables:
 $ eruptions: num 2.38
 $ waiting  : num 71
>
> color = rep(NA, length=length(x))
> color[which(x$eruptions<3 & x$waiting > 70)] = "red"
> color[which(!(x$eruptions<3 & x$waiting > 70))] = "blue"
> color
  [1] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
 [16] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
 [31] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
 [46] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
```

```
 [61] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
 [76] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
 [91] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[106] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[121] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[136] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[151] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[166] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[181] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[196] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[211] "red"  "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[226] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[241] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[256] "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue" "blue"
[271] "blue" "blue"
```

```
> plot(x$waiting,x$eruptions, col = color, pch=16, xlab = "Wait time for eruption",
+      ylab = "Eruption Time", main = "Scatterplot of Wait Times versus Eruption Times")
```



#(ii) (1.5 point) What does the plot suggest about the relationship between eruption time and waiting time?

#***Answer: (Enter your answer here.) ***

#The plot suggests the longer the wait time for Faithful to erupt the longer the eruption lasts.

#In particular there is only one eruption less than three minutes where the wait time for

#eruption exceeds 70.

#-----

#(2)(b) (4.5 points) Past research indicates that the waiting times between consecutive

eruptions are not independent. This problem will check to see if there is evidence

of this. Form consecutive pairs of waiting times. In other words, pair the first

and second waiting times, pair the third and fourth waiting times, and so forth.

There are 136 resulting consecutive pairs of waiting times. Form a data frame with

the first column containing the first waiting time in a pair and the second column

with the second waiting time in a pair. Plot the pairs with the second member of

a pair on the vertical axis and the first member on the horizontal axis.

One way to do this is to pass the vector of waiting times - faithful$waiting -

to *matrix()*, specifying 2 columns for our matrix, with values organized by row;

i.e. byrow = TRUE.

```
vwaiting<-matrix(x$waiting, ncol = 2, byrow = TRUE)
vwaiting
plot(vwaiting, col="orange", xlab = "Waiting time of First Paired Eruption",
    ylab = "Waiting time of Second Paired Eruption", pch=16,
    main = "Scatter Plot of Eruption Wait Times\nfor Consecutive Pairs of Eruptions ")
```

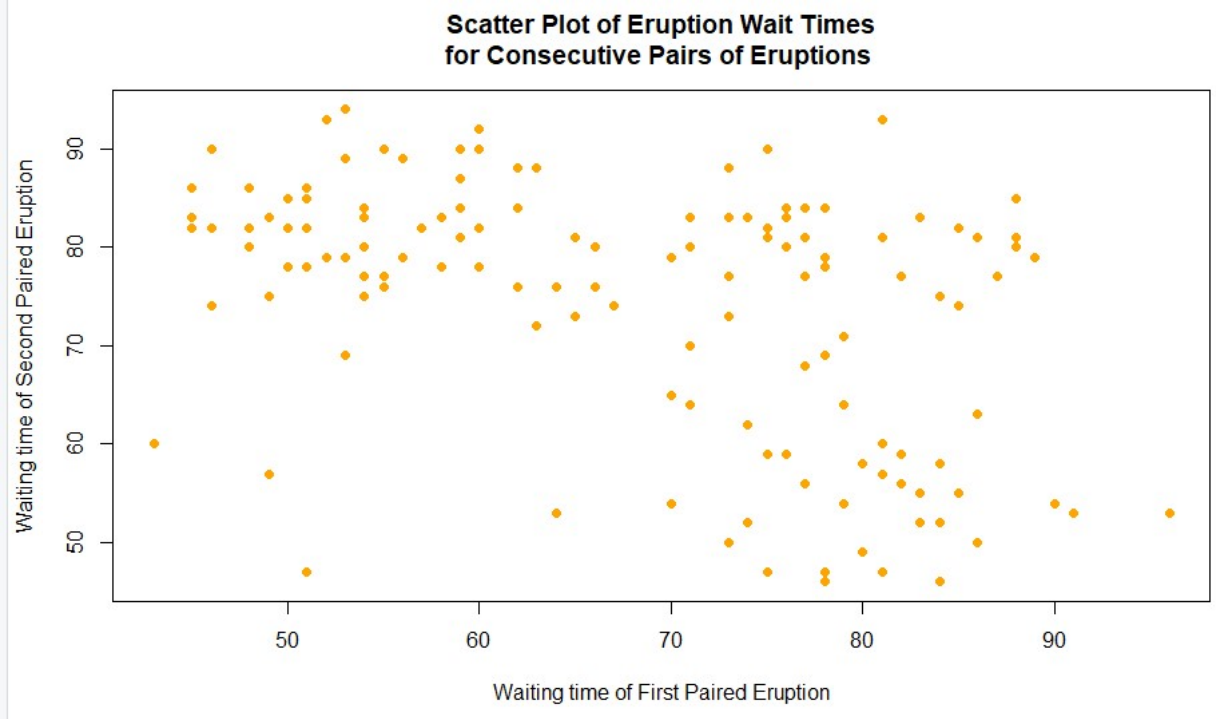
output

```
> vwaiting<-matrix(x$waiting, ncol = 2, byrow = TRUE)
> vwaiting
      [,1] [,2]
 [1,]   79   54
 [2,]   74   62
 [3,]   85   55
 [4,]   88   85
 [5,]   51   85
 [6,]   54   84
 [7,]   78   47
 [8,]   83   52
 [9,]   62   84
[10,]   52   79
[11,]   51   47
[12,]   78   69
[13,]   74   83
[14,]   55   76
[15,]   78   79
[16,]   73   77
```

```
[17,]   66   80
[18,]   74   52
[19,]   48   80
[20,]   59   90
[21,]   80   58
[22,]   84   58
[23,]   73   83
[24,]   64   53
[25,]   82   59
[26,]   75   90
[27,]   54   80
[28,]   54   83
[29,]   71   64
[30,]   77   81
[31,]   59   84
[32,]   48   82
[33,]   60   92
[34,]   78   78
[35,]   65   73
[36,]   82   56
[37,]   79   71
[38,]   62   76
[39,]   60   78
[40,]   76   83
[41,]   75   82
[42,]   70   65
[43,]   73   88
[44,]   76   80
[45,]   48   86
[46,]   60   90
[47,]   50   78
[48,]   63   72
[49,]   84   75
[50,]   51   82
[51,]   62   88
[52,]   49   83
[53,]   81   47
[54,]   84   52
[55,]   86   81
[56,]   75   59
[57,]   89   79
[58,]   59   81
[59,]   50   85
[60,]   59   87
[61,]   53   69
[62,]   77   56
[63,]   88   81
[64,]   45   82
[65,]   55   90
[66,]   45   83
[67,]   56   89
[68,]   46   82
[69,]   51   86
[70,]   53   79
[71,]   81   60
[72,]   82   77
[73,]   76   59
```

```
 [74,]   80   49
 [75,]   96   53
 [76,]   77   77
 [77,]   65   81
 [78,]   71   70
 [79,]   81   93
 [80,]   53   89
 [81,]   45   86
 [82,]   58   78
 [83,]   66   76
 [84,]   63   88
 [85,]   52   93
 [86,]   49   57
 [87,]   77   68
 [88,]   81   81
 [89,]   73   50
 [90,]   85   74
 [91,]   55   77
 [92,]   83   83
 [93,]   51   78
 [94,]   84   46
 [95,]   83   55
 [96,]   81   57
 [97,]   76   84
 [98,]   77   81
 [99,]   87   77
[100,]   51   78
[101,]   60   82
[102,]   91   53
[103,]   78   46
[104,]   77   84
[105,]   49   83
[106,]   71   80
[107,]   49   75
[108,]   64   76
[109,]   53   94
[110,]   55   76
[111,]   50   82
[112,]   54   75
[113,]   78   79
[114,]   78   78
[115,]   70   79
[116,]   70   54
[117,]   86   50
[118,]   90   54
[119,]   54   77
[120,]   79   64
[121,]   75   47
[122,]   86   63
[123,]   85   82
[124,]   57   82
[125,]   67   74
[126,]   54   83
[127,]   73   73
[128,]   88   80
[129,]   71   83
[130,]   56   79
```

```
[131,]   78   84
[132,]   58   83
[133,]   43   60
[134,]   75   81
[135,]   46   90
[136,]   46   74
> plot(vwaiting, col="orange", xlab = "Waiting time of First Paired Eruption"
,
+      ylab = "Waiting time of Second Paired Eruption", pch=16,
+      main = "Scatter Plot of Eruption Wait Times\nfor Consecutive Pairs of
Eruptions ")
```



#(2)(c) (2) Test the hypothesis of independence with a two-sided test at the 5% level

using the Kendall correlation coefficient.

#```{r test2c}

cor(vwaiting, method="kendall", use = "pairwise")

output

```
> cor(vwaiting, method="kendall", use = "pairwise")
           [,1]       [,2]
[1,]  1.0000000 -0.2935579
[2,] -0.2935579  1.0000000
```

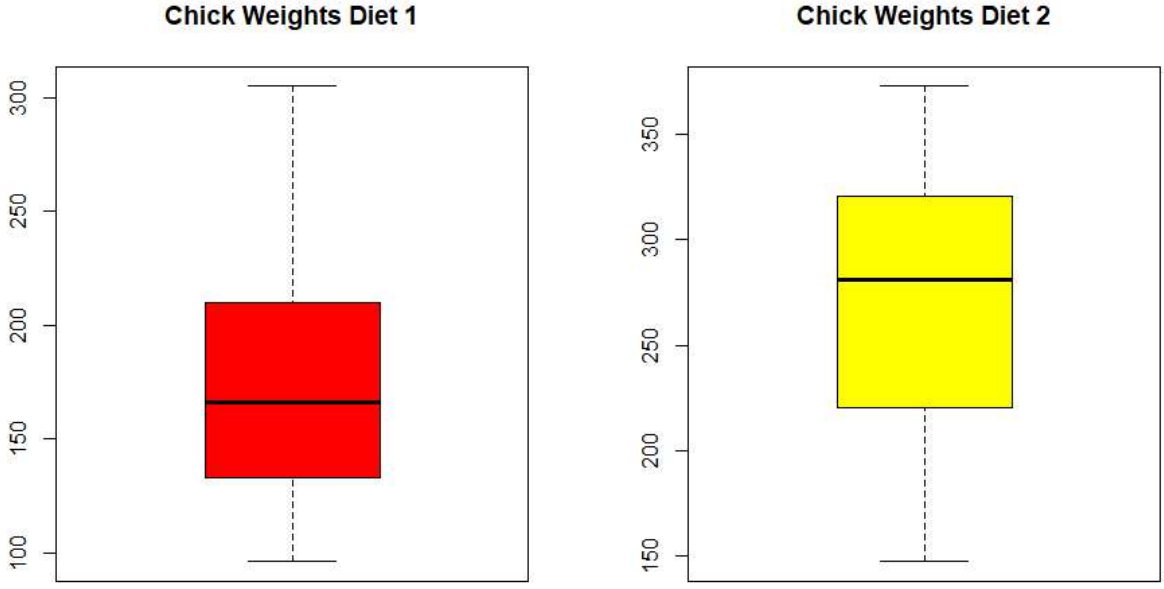
```
##### Section 3: (15 points) ###################################

##### (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.

#```{r test3}

# 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$Diet == "3")
# Create data frame, "result," with the weight and Diet of those observations with "TRUE" "index"" values
result <- subset(ChickWeight[index, ], select = c(weight, Diet))
# Encode "Diet" as a factor
result$Diet <- factor(result$Diet)
str(result)

##### The data frame, "result", has chick weights for two diets, identified as
# diet "1" and "3". Use the data frame, "result," to complete the following item.

#(3)(a) (3 points) Display two side-by-side vertical boxplots using par(mfrow = c(1,2)).
# One boxplot would display diet "1" and the other diet "3".
par(mfrow=c(1,2))
Diet1<-result[which(result$Diet==1),]
Diet3<-result[which(result$Diet==3),]
boxplot(Diet1$weight, main = "Chick Weights Diet 1", col = "red")
```

boxplot(Diet3$weight, main = "Chick Weights Diet 2", col = "yellow")

par(mfrow=c(1,1))

output

```
> data(ChickWeight)
>
> # Create T | F vector indicating observations with Time == 21 and Diet == "
1" OR "3"
> index <- ChickWeight$Time == 21 & (ChickWeight$Diet == "1" | ChickWeight$Di
et == "3")
>
> # Create data frame, "result," with the weight and Diet of those observatio
ns with "TRUE" "index"" values
> result <- subset(ChickWeight[index, ], select = c(weight, Diet))
>
> # Encode "Diet" as a factor
> result$Diet <- factor(result$Diet)
> str(result)
Classes ‘nfnGroupedData’, ‘nfGroupedData’, ‘groupedData’ and 'data.frame':  2
6 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 ...
> par(mfrow=c(1,2))
> Diet1<-result[which(result$Diet==1),]
> Diet3<-result[which(result$Diet==3),]
> boxplot(Diet1$weight, main = "Chick Weights Diet 1", col = "red")
> boxplot(Diet3$weight, main = "Chick Weights Diet 2", col = "yellow")
> par(mfrow=c(1,1))
```



#(3)(b) (3 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().

```
t.test(Diet1$weight, Diet3$weight)
```

output

```
> t.test(Diet1$weight, Diet3$weight)

        Welch Two Sample t-test

data:  Diet1$weight and Diet3$weight
t = -3.4293, df = 16.408, p-value = 0.003337
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -149.64644  -35.45356
sample estimates:
mean of x mean of y
   177.75    270.30
```

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.

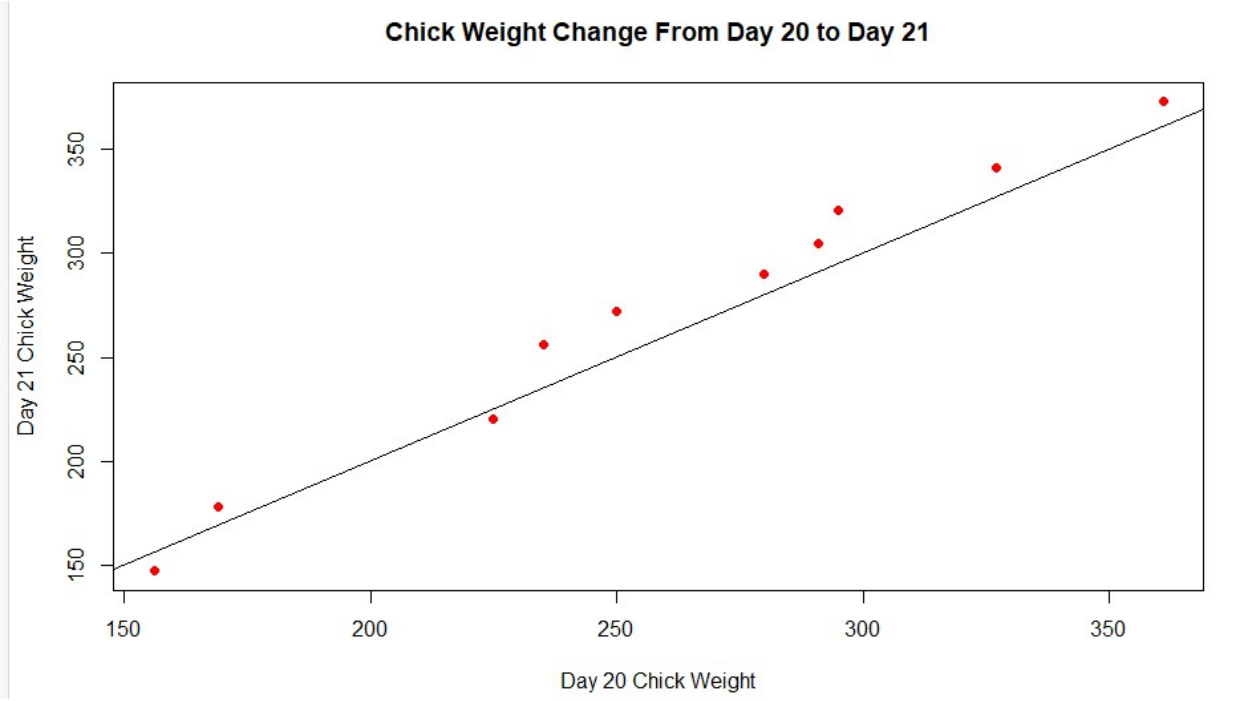
#```{r test3paired}

```
# 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
# The pre and post values are paired, each pair corresponding to an individual chick.
cbind(pre, post)
#(3)(c) (3 points) Present a scatterplot of the variable "post" as a function of the
# variable "pre".  Include a diagonal line with zero intercept and slope equal to one.
# Title and label the variables in this scatterplot.
plot(pre, post, xlab="Day 20 Chick Weight",
    ylab="Day 21 Chick Weight",
    main = "Chick Weight Change From Day 20 to Day 21",
    pch=16, col="red")
abline(a=0,b=1)
```

output

```
> plot(pre, post, xlab="Day 20 Chick Weight",
+      ylab="Day 21 Chick Weight",
+      main = "Chick Weight Change From Day 20 to Day 21",
+      pch=16, col="red")
> abline(a=0,b=1)
```



#(3)(d) (6 points) Calculate and present a one-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.

#```{r test3d}

```
mu<-mean(pre - post)

n<-length(pre)

crit_t<-qt(p=.950,df=n-1)

se<-sd(pre-post)/sqrt(n)

t<-mu/se

list(t.statistic = t, critical.t = crit_t,

    p.value = 2 * pt(q = t, df = n - 1, lower.tail = FALSE),

    conf.interval = c(mu - crit_t * se, mu + crit_t * se))

t.test(x = pre, y = post, paired = TRUE)
```

output

```
> mu<-mean(pre - post)
> n<-length(pre)
> crit_t<-qt(p=.950,df=n-1)
> se<-sd(pre-post)/sqrt(n)
> t<-mu/se
>
> list(t.statistic = t, critical.t = crit_t,
+      p.value = 2 * pt(q = t, df = n - 1, lower.tail = FALSE),
+      conf.interval = c(mu - crit_t * se, mu + crit_t * se))
$t.statistic
[1] -3.225267

$critical.t
[1] 1.833113

$p.value
[1] 1.989599

$conf.interval
[1] -17.879304  -4.920696


>
```

```
> t.test(x = pre, y = post, paired = TRUE)

        Paired t-test

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

#(4)(a) (3 points) Using Nile River flow data and the "moments" package, calculate

skewness and kurtosis. Present a QQ plot and boxplot of the flow data side-by-side

using *qqnorm()*, *qqline()* and *boxplot()*; *par(mfrow = c(1, 2))* may be used

to locate the plots side-by-side. Add features to these displays as you choose.

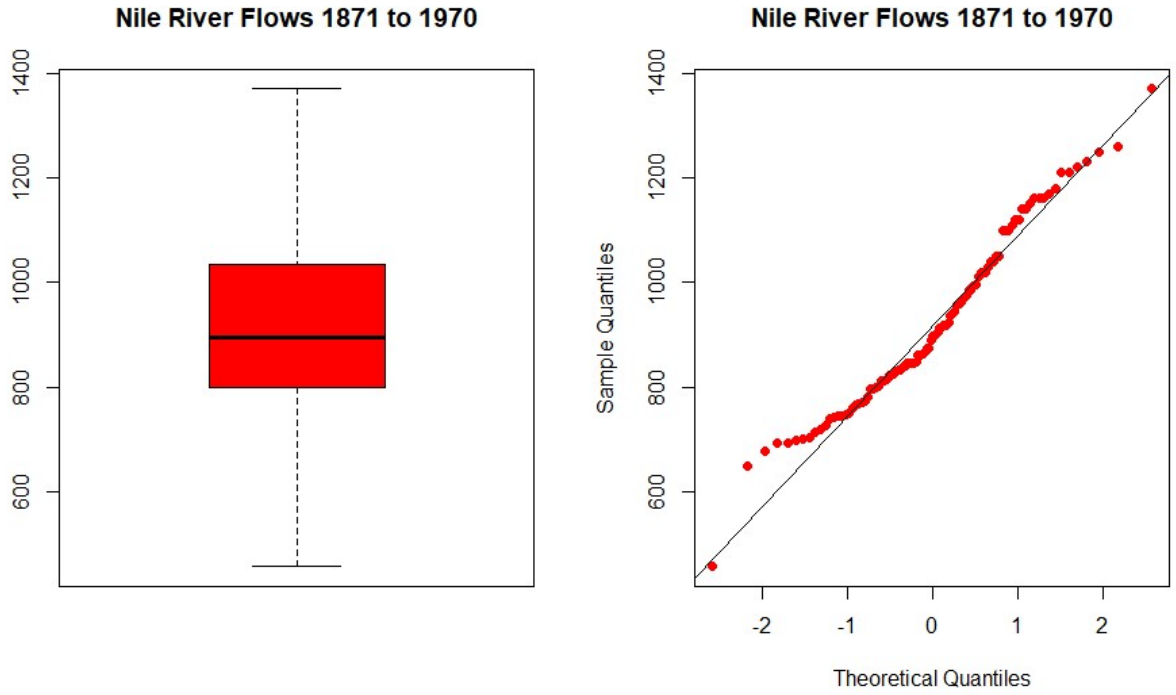
#```{r test4a}

```
library(moments)

str(Nile)

summary(Nile)

skewness(Nile)

kurtosis(Nile)

par(mfrow=c(1,2))

boxplot(Nile, col = 'red', main = 'Nile River Flows 1871 to 1970' )

qqnorm(Nile, col = 'red', pch=16, main = 'Nile River Flows 1871 to 1970' )

qqline(Nile)

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

output

```
> library(moments)
> str(Nile)
 Time-Series [1:100] from 1871 to 1970: 1120 1160 963 1210 1160 1160 813 1230
1370 1140 ...
> summary(Nile)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  456.0   798.5   893.5   919.4  1032.5  1370.0
```

```
> skewness(Nile)
[1] 0.3223697
> kurtosis(Nile)
[1] 2.695093
> par(mfrow=c(1,2))
> boxplot(Nile, col = 'red', main = 'Nile River Flows 1871 to 1970' )
> qqnorm(Nile, col = 'red', pch=16, main = 'Nile River Flows 1871 to 1970' )
> qqline(Nile)
> par(mfrow=c(1,1))
```



#(4)(b) (6 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 can be done with 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 means, sample standard

deviations and sample variances for "sample1" and "sample2" in a table with the first

row for "sample1", the second row for "sample2" and the columns labled for each statistic.

#```{r test4b}

```
set.seed(124)

sample1<-numeric(1000)

for (i in 1:1000) {

  a<-sample(y, 16, replace = T)

  sample1[i]<- mean(a)

}

set.seed(127)

sample2<-matrix(, nrow = 1000, ncol=3)

sample2<-numeric(1000)

for (i in 1:1000) {

  a<-sample(y, 64, replace = T)

  sample2[i]<- mean(a)

}

(samp <- cbind(mean(sample1),sd(sample1),var(sample1)))

(samp <- rbind(samp,cbind(mean(sample2),sd(sample2), var(sample2))))

rownames(samp) <- c("sample1","sample2")

colnames(samp) <- c("Mean", "Std", "Variance")

samp
```

OUTPUT

```
> set.seed(124)
> sample1<-numeric(1000)
> for (i in 1:1000) {
+   a<-sample(y, 16, replace = T)
+   sample1[i]<- mean(a)
+ }
>
> set.seed(127)
> sample2<-matrix(, nrow = 1000, ncol=3)
> sample2<-numeric(1000)
> for (i in 1:1000) {
+   a<-sample(y, 64, replace = T)
+   sample2[i]<- mean(a)
+ }
>
> (samp <- cbind(mean(sample1),sd(sample1),var(sample1)))
          [,1]      [,2]         [,3]
[1,] 0.1878658 0.0469961 0.002208633
> (samp <- rbind(samp,cbind(mean(sample2),sd(sample2), var(sample2))))
          [,1]       [,2]         [,3]
[1,] 0.1878658 0.04699610 0.002208633
```

```
 [2,] 0.1889880 0.02346896 0.000550792
> rownames(samp) <- c("sample1","sample2")
> colnames(samp) <- c("Mean", "Std", "Variance")
>
> samp
             Mean        Std    Variance
sample1 0.1878658 0.04699610 0.002208633
sample2 0.1889880 0.02346896 0.000550792
```

#(4)(c) (6 points) Present side-by-side histograms of "sample1" and "sample2" with the

normal density curve superimposed. 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.**

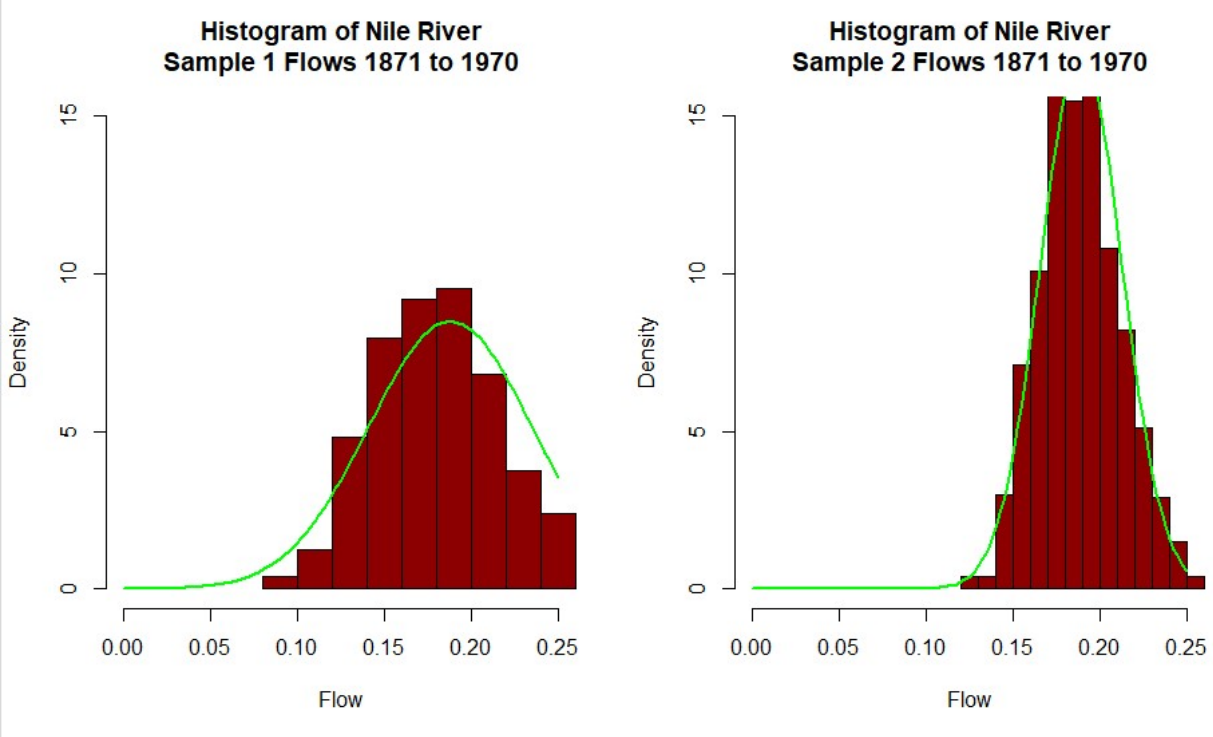
#```{r test4c}

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

x<-seq(from = 700, to = 1100, by = 1)

hist(sample1, freq = FALSE, col = "darkred", xlab = "Flow",

    main = "Histogram of Nile River\nSample 1 Flows 1871 to 1970"

, ylim = c(0, 15),

xlim = c(0, 0.25), plot = TRUE)

curve(dnorm(x, mean = samp[1,1], sd = samp[1,2]), col = "green", lwd = 2, add = TRUE)

hist(sample2, freq = FALSE, col = "darkred", xlab = "Flow",

    main = "Histogram of Nile River\nSample 2 Flows 1871 to 1970"

, ylim = c(0, 15),

xlim = c(0, 0.25), plot = TRUE)

curve(dnorm(x, mean = samp[2,1], sd = samp[2,2]), col = "green", lwd = 2, add = TRUE)

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

output

```
> par(mfrow=c(1,2))
> x<-seq(from = 700, to = 1100, by = 1)
> hist(sample1, freq = FALSE, col = "darkred", xlab = "Flow",
+      main = "Histogram of Nile River\nSample 1 Flows 1871 to 1970"
+ , ylim = c(0, 15),
+ xlim = c(0, 0.25), plot = TRUE)
```

```
> curve(dnorm(x, mean = samp[1,1], sd = samp[1,2]), col = "green", lwd = 2, a
dd = TRUE)
> hist(sample2, freq = FALSE, col = "darkred", xlab = "Flow",
+      main = "Histogram of Nile River\nSample 2 Flows 1871 to 1970"
+ , ylim = c(0, 15),
+ xlim = c(0, 0.25), plot = TRUE)
> curve(dnorm(x, mean = samp[2,1], sd = samp[2,2]), col = "green", lwd = 2, a
dd = TRUE)
> par(mfrow=c(1,1))
```



Section 5: (15 points)

(5) This problem deals with contingency table analysis. This is an example

of categorical data analysis (see Kabacoff, pp. 145-151). The "warpbreaks" dataset

gives the number of warp breaks per loom, where a loom corresponds to a fixed length

of yarn. There are 54 observations on 3 variables: breaks (numeric, the number of breaks),

wool (factor, type of wool: A or B), and tension (factor, low L, medium M and high H).

These data have been studied and used for example elsewhere. For the purposes of this problem,

we will focus on the relationship between breaks and tension using contingency table analysis.

(5)(a)(4.5 points) warpbreaks is part of the "datasets" package and may be loaded via
data(warpbreaks). Load "warpbreaks" and present the structure using *str()*.
Calculate the median number of breaks for the entire dataset, disregarding "tension" and
"wool". Define this median value as "median_breaks". Present a histogram of the number of
breaks with the location of the median indicated.

Create a new variable "number" as follows: for each value of "breaks", classify the number
of breaks as either strictly below "median_breaks", or the alternative. Convert the
"above"|"below" classifications to a factor, and combine with the dataset warpbreaks.
Present a summary of the augmented dataset using *summary()*. Present a contingency
table of the frequency of breaks using the two variables "tension" and "number".
There should be six cells in this table.

#```{r test5a}

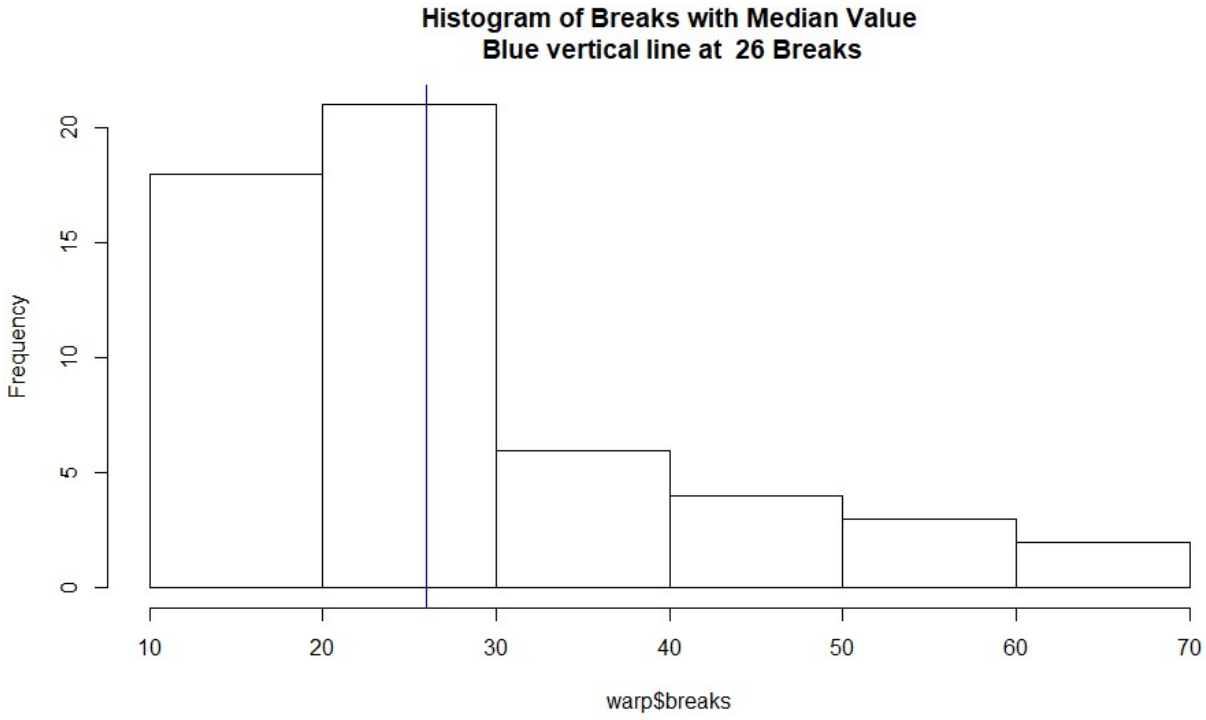
```
data(warpbreaks)
warp<-warpbreaks
str(med)

(median_breaks <- median(warp$breaks))

hist(warp$breaks, main = paste("Histogram of Breaks with Median Value\n Blue vertical line at ",
                               median_breaks, "Breaks"))
abline(v= median_breaks, col="blue")

warp$P50.loc <- factor(warp)
warp$P50.loc <- factor(warp$breaks>median_breaks)
levels(warp$P50.loc <- c("below", "above"))
```

```
summary(warp)


(warptab <- table(warp$tension,warp$P50.loc))


#```

data(warpbreaks)

warp<-warpbreaks

str(med)

(median_breaks <- median(warp$breaks))

hist(warp$breaks, main = paste("Histogram of Breaks with Median Value\n Blue vertical line at ",

                               median_breaks, "Breaks"))

abline(v= median_breaks, col="blue")

warp$P50.loc <- factor(warp)

warp$P50.loc <- factor(warp$breaks>median_breaks)

levels(warp$P50.loc <- c("below", "above"))

summary(warp)

(warptab <- table(warp$tension,warp$P50.loc))
```

Output

```
> data(warpbreaks)
> warp<-warpbreaks
> str(med)
'data.frame':  54 obs. of  4 variables:
 $ breaks : num  26 30 54 25 70 52 51 26 67 18 ...
 $ wool   : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
 $ tension: Factor w/ 3 levels "L","M","H": 1 1 1 1 1 1 1 1 1 2 ...
 $ P50.loc: chr  "below" "above" "below" "above" ...
>
> (median_breaks <- median(warp$breaks))
[1] 26
>
> hist(warp$breaks, main = paste("Histogram of Breaks with Median Value\n Blu
e vertical line at ",
+                                median_breaks, "Breaks"))
> abline(v= median_breaks, col="blue")
>
> warp$P50.loc <- factor(warp)
> warp$P50.loc <- factor(warp$breaks>median_breaks)
> levels(warp$P50.loc <- c("below", "above"))
NULL
```

```
> summary(warp)
     breaks      wool   tension   P50.loc
 Min.   :10.00   A:27   L:18    Length:54
 1st Qu.:18.25   B:27   M:18    Class :character
 Median :26.00          H:18    Mode  :character
 Mean   :28.15
 3rd Qu.:34.00
 Max.   :70.00
>
> (warptab <- table(warp$tension,warp$P50.loc))

    above below
  L     9     9
  M     9     9
  H     9     9
```



#(5)(b)(3 points) Using the table constructed in (5)(a), test at the 5% level

the null hypothesis of independence using the uncorrected *chisq.test()*

(Black, Business Statistics, Section 16.2). Show the results of this test and state your conclusions.

#```{r test5b}

chisq.test(warptab, correct = FALSE)

output

```
> chisq.test(warptab, correct = FALSE)

        Pearson's Chi-squared test

data:  warptab
X-squared = 0, df = 2, p-value = 1
```

The observed value of chi squared 0 is less than the critical value 5.9915 so the null hypothesis tension is independent of number is accepted.

#(5)(c) (7.5 points) Write a function that computes the uncorrected

Pearson Chi-squared statistic.   Apply your function to the table from (5)(a).

You should be able to duplicate the X-squared value (chi-squared) and *p*-value. Present both.

Shown below are examples of the type of function required. These examples will have

to be modified to accomodate the table generated in (5)(a).

#```{r test5c}

```
chi <- function(x) {
 # To be used with 2x2 contingency tables that have margins added.
 # Expected values are calculated.
 e11 <- x[4,1]*x[1,3]/x[4,3]
 e12 <- x[4,2]*x[1,3]/x[4,3]
```

```
  e21 <- x[4,1]*x[2,3]/x[4,3]
  e22 <- x[4,2]*x[2,3]/x[4,3]
  e31 <- x[4,1]*x[3,3]/x[4,3]
  e32 <- x[4,2]*x[3,3]/x[4,3]
  # Value of chi square statistic is calculated.
  chisqStat <- (x[1,1] - e11)^2/e11 + (x[1,2] - e12)^2/e12 +
    (x[2,1] - e21)^2/e21 + (x[2,2] - e22)^2/e22 +
    (x[3,1] - e31)^2/e31 + (x[3,2] - e32)^2/e32
  return(list("chi-squared" = chisqStat,
              "p-value" = pchisq(chisqStat, 1, lower.tail = F)))
}

chisqfun <- function(t) {
  x <- addmargins(t)
  e <- matrix(0, nrow = nrow(t), ncol = ncol(t), byrow = T)
  r <- matrix(0, nrow = nrow(t), ncol = ncol(t), byrow = T)
  for (i in 1:2) {
    for (j in 1:2) {
      e[i,j] = x[nrow(x),j] * x[i,ncol(x)]/x[nrow(x), ncol(x)]
      r[i,j] = ((x[i,j] - e[i,j])^2)/e[i,j]
    }
  }
  chi <- sum(r)
  xdf <- nrow(t) - 2
  pv <- pchisq(chi, df = xdf, lower.tail = FALSE)
  return(cat("Pearson's Chi-squared test \n","Chi sq: ", chi, ";
             Degree of Freedom :",xdf," ; P-value :",pv))
}
```

```
(Y <- matrix(warptab, nrow = 3, byrow = F))

(A <- cbind(Y, apply(Y,1,sum)))

(B <- rbind(A, apply(A, 2, sum)))


chi(B)

chisqfun(B)
```

output

```
> chi <- function(x) {
+   # To be used with 2x2 contingency tables that have margins added.
+   # Expected values are calculated.
+   e11 <- x[4,1]*x[1,3]/x[4,3]
+   e12 <- x[4,2]*x[1,3]/x[4,3]
+   e21 <- x[4,1]*x[2,3]/x[4,3]
+   e22 <- x[4,2]*x[2,3]/x[4,3]
+   e31 <- x[4,1]*x[3,3]/x[4,3]
+   e32 <- x[4,2]*x[3,3]/x[4,3]
+   # Value of chi square statistic is calculated.
+   chisqStat <- (x[1,1] - e11)^2/e11 + (x[1,2] - e12)^2/e12 +
+     (x[2,1] - e21)^2/e21 + (x[2,2] - e22)^2/e22 +
+     (x[3,1] - e31)^2/e31 + (x[3,2] - e32)^2/e32
+   return(list("chi-squared" = chisqStat,
+               "p-value" = pchisq(chisqStat, 1, lower.tail = F)))
+ }
>
> chisqfun <- function(t) {
+   x <- addmargins(t)
+   e <- matrix(0, nrow = nrow(t), ncol = ncol(t), byrow = T)
+   r <- matrix(0, nrow = nrow(t), ncol = ncol(t), byrow = T)
+   for (i in 1:2) {
+     for (j in 1:2) {
+       e[i,j] = x[nrow(x),j] * x[i,ncol(x)]/x[nrow(x), ncol(x)]
+       r[i,j] = ((x[i,j] - e[i,j])^2)/e[i,j]
+     }
+   }
+   chi <- sum(r)
+   xdf <- nrow(t) - 2
+   pv <- pchisq(chi, df = xdf, lower.tail = FALSE)
+   return(cat("Pearson's Chi-squared test \n","Chi sq: ", chi, ";
+              Degree of Freedom :",xdf," ; P-value :",pv))
+ }
>
> (Y <- matrix(warptab, nrow = 3, byrow = F))
     [,1] [,2]
[1,]    9    9
[2,]    9    9
[3,]    9    9
> (A <- cbind(Y, apply(Y,1,sum)))
     [,1] [,2] [,3]
```

```
[1,]    9    9   18
[2,]    9    9   18
[3,]    9    9   18
> (B <- rbind(A, apply(A, 2, sum)))
     [,1] [,2] [,3]
[1,]    9    9   18
[2,]    9    9   18
[3,]    9    9   18
[4,]   27   27   54
>
> chi(B)
$`chi-squared`
[1] 0

$`p-value`
[1] 1

> chisqfun(B)
Pearson's Chi-squared test
 Chi sq:  0 ;
            Degree of Freedom : 2  ; P-value : 1
```