Gesheva_Programming with R Assignment #2

Instructions

R markdown is a plain-text file format for integrating text and R code, and creating transparent, reproducible and interactive reports. An R markdown file (.Rmd) contains metadata, markdown and R code "chunks", and can be "knit" into numerous output types. Answer the test questions by adding R code to the fenced code areas below each item. Once completed, you will "knit" and submit the resulting .html file, as well the .Rmd file. The .html will include your R code *and* the output.

Before proceeding, look to the top of the .Rmd for the (YAML) metadata block, where the *title* and *output* are given. Please change *title* from 'Programming with R Test #2' to your name, with the format 'lastName_firstName.'

If you encounter issues knitting the .html, please send an email via Canvas to your TA.

Each code chunk is delineated by six (6) backticks; three (3) at the start and three (3) at the end. After the opening ticks, arguments are passed to the code chunk and in curly brackets. **Please do not add or remove backticks, or modify the arguments or values inside the curly brackets**.

Depending on the problem, grading will be based on: 1) the correct result, 2) coding efficiency and 3) graphical presentation features (labeling, colors, size, legibility, etc). I will be looking for well-rendered displays. In the "knit" document, only those results specified in the problem statements should be displayed. For example, do not output - i.e. send to the Console - the contents of vectors or data frames unless requested by the problem. You should be able to display each solution in fewer than ten lines of code.

Submit both the .Rmd and .html files for grading.

Please delete the Instructions shown above prior to submitting your .Rmd and .html files.

Test Items starts from here - There are 5 sections - 75 points total

Section 1: (15 points)

(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.

```
dpois(0, lambda = 5)
```

```
## [1] 0.006737947
```

```
ppois(0, lambda = 5)
```

```
## [1] 0.006737947
```

```r
dbinom(0, 100, .05)
```

```
## [1] 0.005920529
```

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

```
## [1] 0.005920529
```

(ii) The probability of fewer than 6 successes.

```r
sum(dpois(0:5,lambda=5))
```

```
## [1] 0.6159607
```

```r
ppois(5,lambda=5)
```

```
## [1] 0.6159607
```

```r
sum(dbinom(0:5,100,.05))
```

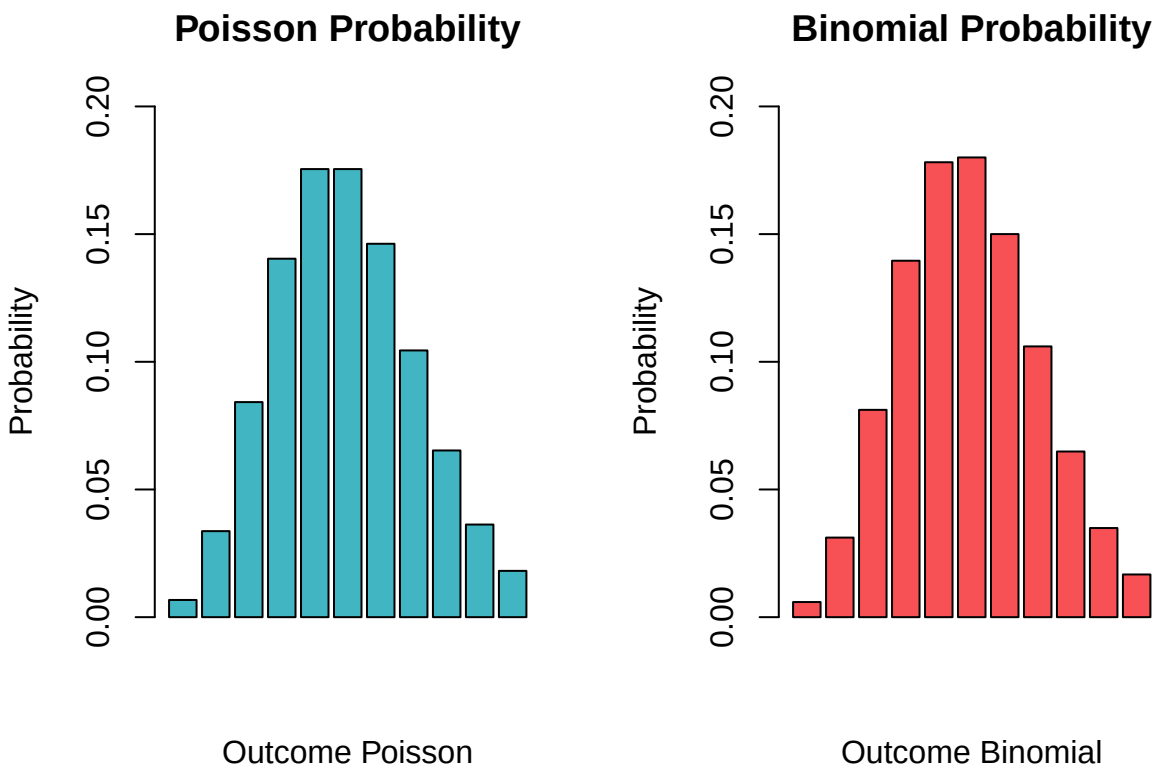
```
## [1] 0.6159991
```

```r
pbinom(5,100,.05)
```

```
## [1] 0.6159991
```

(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.

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

a<-(dpois(0:10,lambda = 5))
b<-(dbinom(0:10,100,0.05))

barplot(a,main ="Poisson Probability",col="#41b5c2",xlab = "Outcome Poisson",ylab ="Probability", ylim=
barplot(b,main = "Binomial Probability",col = "#f85155",xlab = "Outcome Binomial",ylab = "Probability",
```



(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.

```
o<-0:6
p<-c(0.215,0.230,0.240,0.182,0.130,0.003,0.001)

mean_p<-round(sum(o*p),digits=2)
var_p<-round(((sum((o^2)*p))-(mean_p^2)),digits=2)

mean_p
```

```
## [1] 1.8
```

```
var_p
```

```
## [1] 1.78
```

(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.

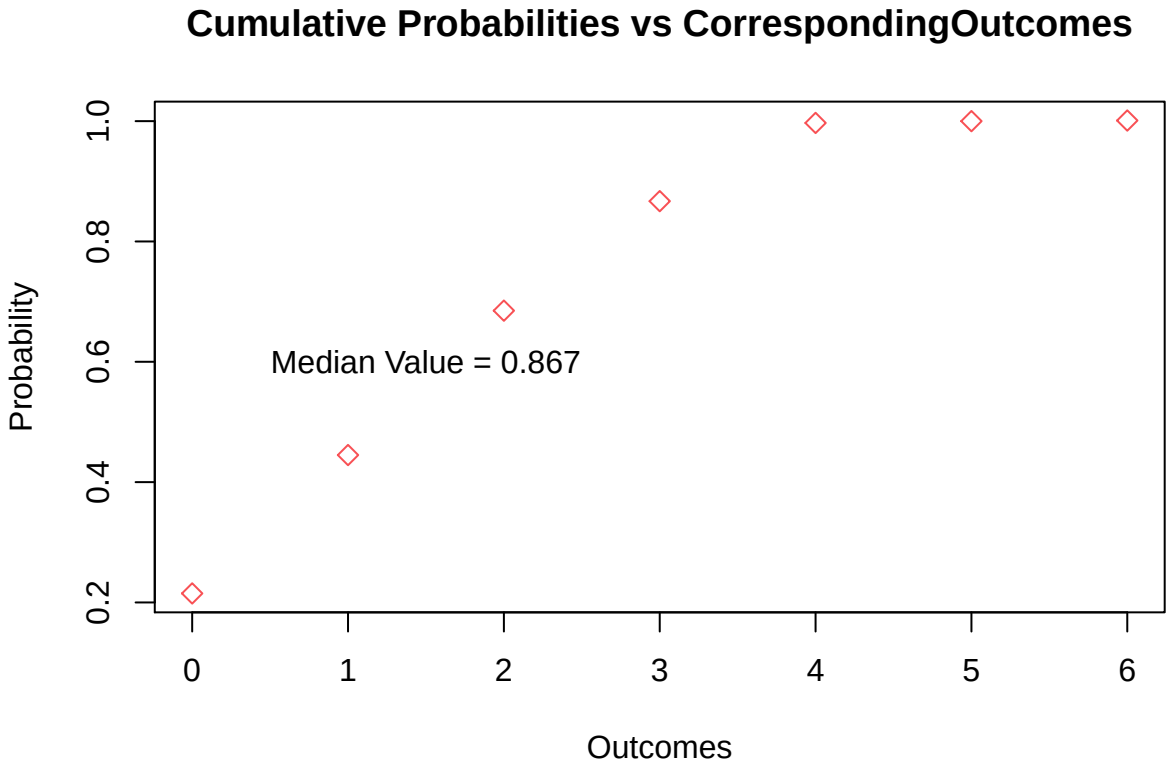
```
cum_p<-cumsum(p)
cum_p
```

```
## [1] 0.215 0.445 0.685 0.867 0.997 1.000 1.001
```

```
median(cum_p)
```

```
## [1] 0.867
```

```
plot(o,cum_p,pch=5,col="#f85155",main = "Cumulative Probabilities vs CorrespondingOutcomes",xlab="Outcor
text(1.5,0.6,"Median Value = 0.867")
```



Section 2: (15 points)

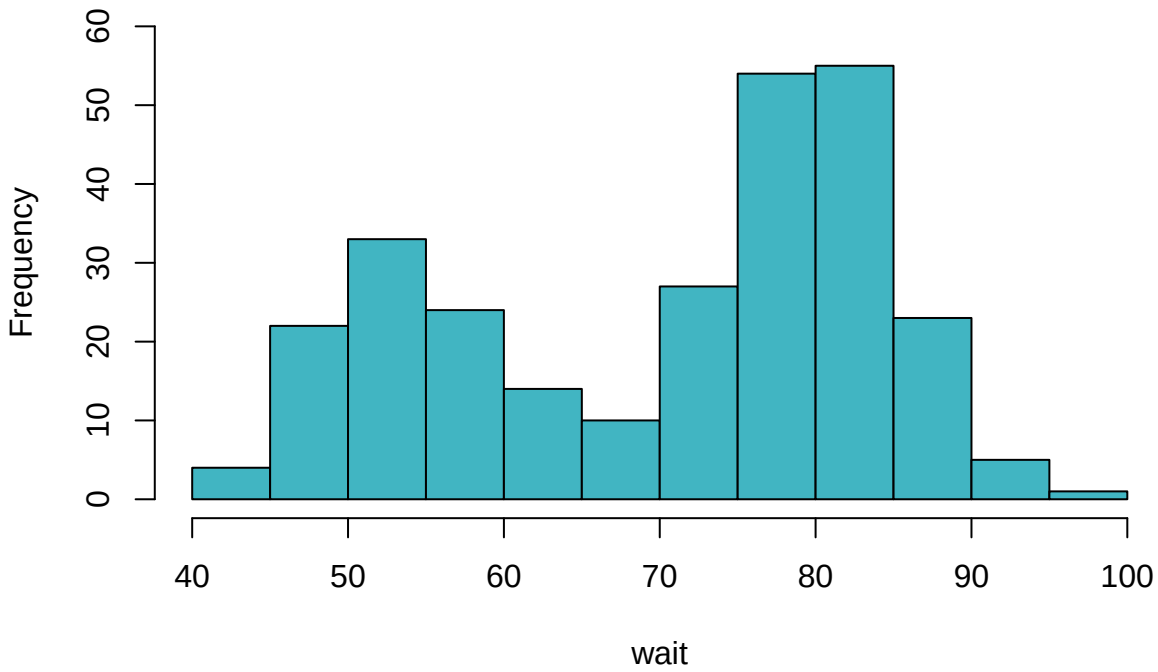
(2) Conditional probabilities appear in many contexts and, in particular, are used by Bayes' Theorem. Correlations are another means for evaluating dependency between variables. The dataset "faithful"" is part of the "datasets" package and may be loaded with the statement *data(faithful)*. It contains 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)
summary(faithful)
```

```
##    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
```

```
wait<-faithful$waiting

hist(wait, main = "Histogrm of Waiting Times",col="#41b5c2", ylim=c(0,60))
```

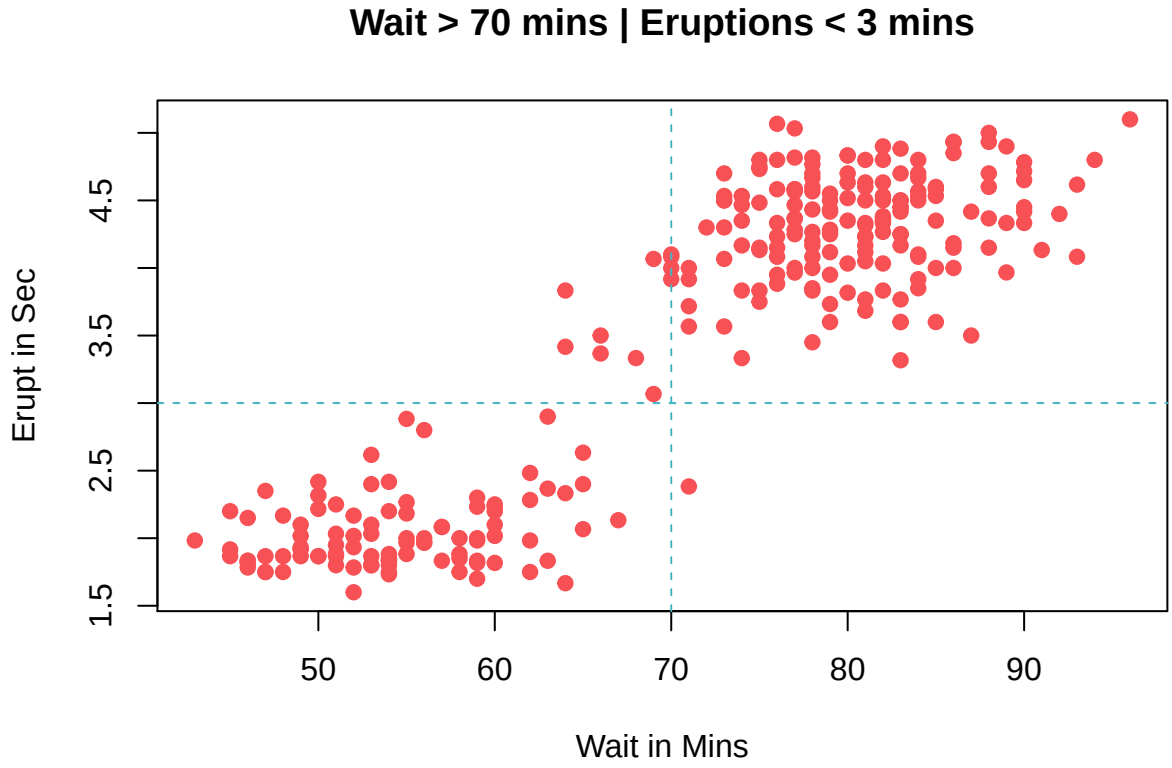



```
wait_70 = subset(faithful, waiting > 70)
erupt_3min = subset(wait_70, eruptions < 3)

nrow(erupt_3min)/nrow(wait_70)
```

```
## [1] 0.006060606
```

(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 show any such observations in a distinct color on a scatterplot of all eruption (vertical 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.

```
erupt<-faithful$eruptions

plot(wait, erupt, pch=19, col="#f85155", main="Wait > 70 mins | Eruptions < 3 mins", xlab = "Wait in Mi

abline(h=3,v=70,lty=2,col="#41b5c2")
```



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

Answer: (The plot revels a rough positive linear correlation. Lower eruption times associate with a lower wait time, longer eruption times associate with longer eruption times.)

---

(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.

```
pair_faith <- matrix(wait, ncol=2, byrow=TRUE)
pair_faith
```

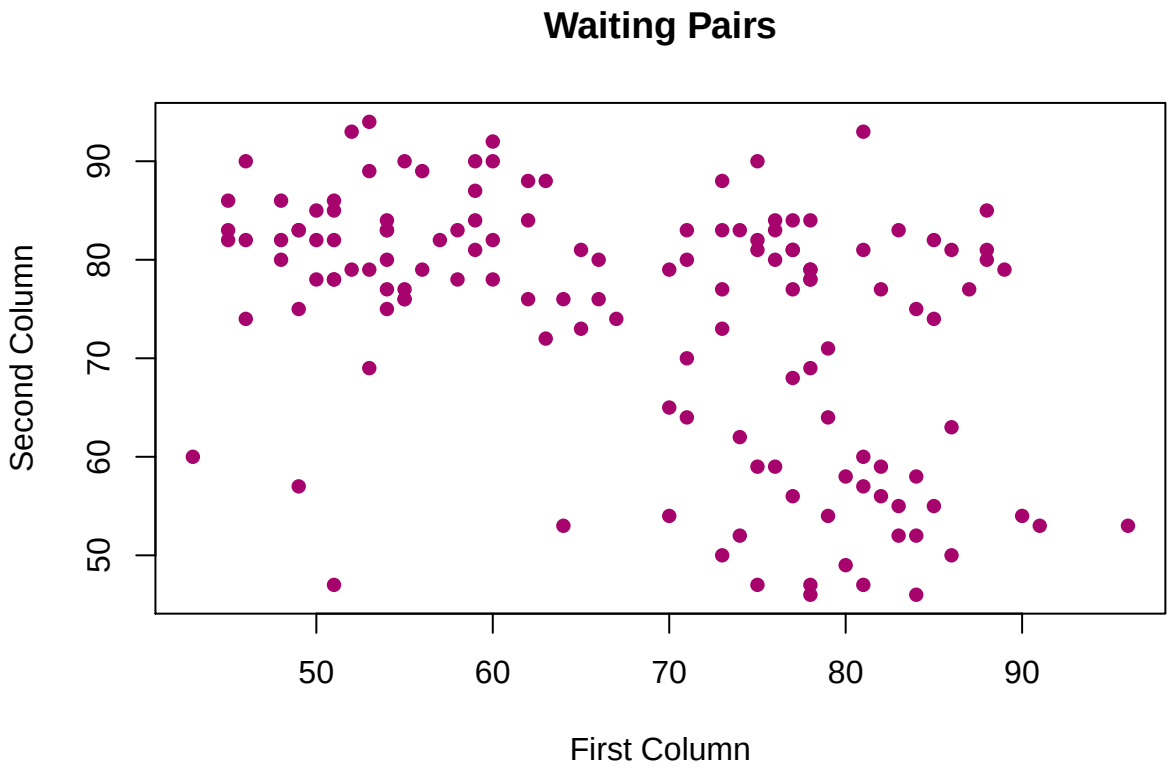
```
##        [,1] [,2]
##   [1,]   79   54
##   [2,]   74   62
##   [3,]   85   55
##   [4,]   88   85
##   [5,]   51   85
##   [6,]   54   84
##   [7,]   78   47
##   [8,]   83   52
##   [9,]   62   84
##  [10,]   52   79
##  [11,]   51   47
```

```
##  [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(y=pair_faith[,2],x=pair_faith[,1], pch=16, col="#a6036d", main="Waiting Pairs", xlab="First Column
```



(2)(c) (2) Test the hypothesis of independence with a two-sided test at the 5% level using the Kendall correlation coefficient.

```
cor.test(pair_faith[,1],pair_faith[,2],method = "kendall", conf.level = 0.95)
```

```
##
##  Kendall's rank correlation tau
##
## data:  pair_faith[, 1] and pair_faith[, 2]
## z = -4.9482, p-value = 7.489e-07
## alternative hypothesis: true tau is not equal to 0
```

```
## sample estimates:
##        tau
## -0.2935579
```

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.

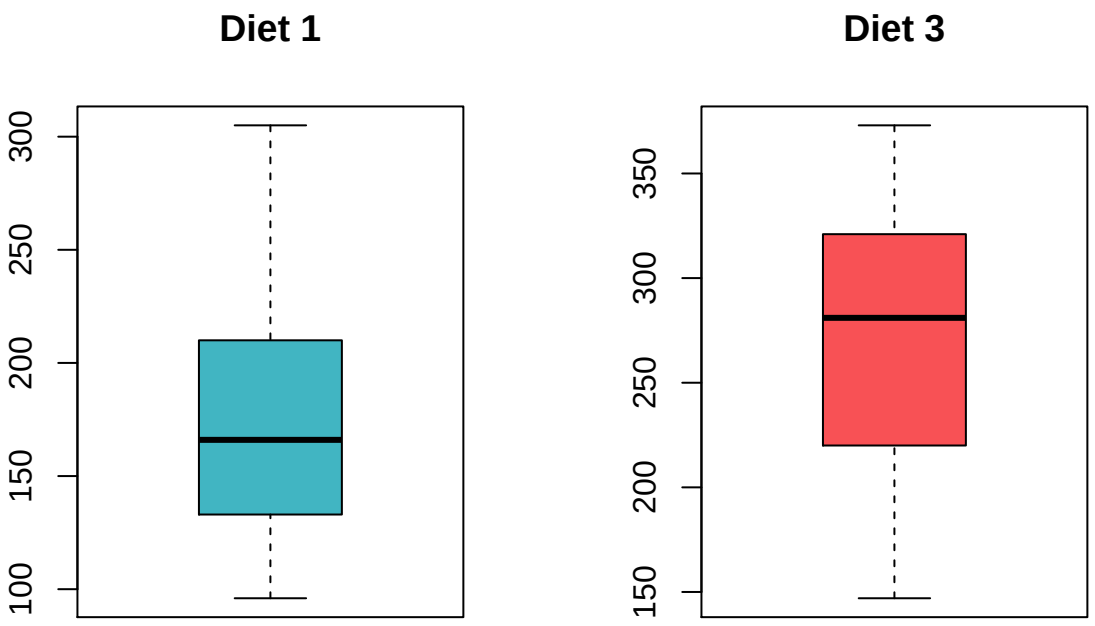
```
# load "ChickWeight" dataset
data(ChickWeight)

# 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"" valu
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", 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<-subset(result,Diet == "1")
diet3<-subset(result,Diet == "3")

boxplot(diet1$weight, main="Diet 1", col="#41b5c2")
boxplot(diet3$weight, main="Diet 3", col="#f85155")
```



(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,alternative = "two.sided", conf.level = 0.95)
```

```
##
##  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.

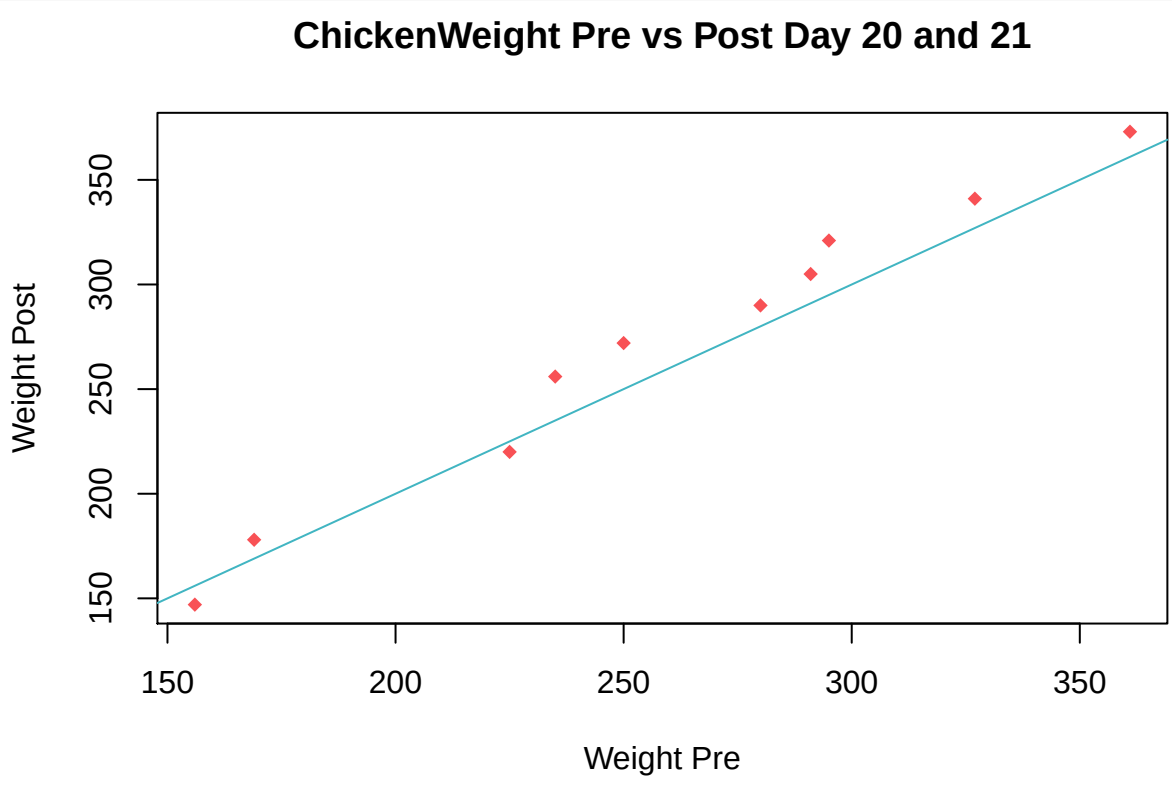
```
# load "ChickWeight" dataset
data(ChickWeight)

# 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.
```

```r
cbind(pre, post)
```

```
##       pre post
##  [1,] 235  256
##  [2,] 291  305
##  [3,] 156  147
##  [4,] 327  341
##  [5,] 361  373
##  [6,] 225  220
##  [7,] 169  178
##  [8,] 280  290
##  [9,] 250  272
## [10,] 295  321
```

(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.

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

plot(pre, post, pch=18, col="#f85155", xlab="Weight Pre", ylab="Weight Post",main="ChickenWeight Pre vs

abline(0,1,col="#41b5c2")
```



(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
t.test(pre,post,alternate="two.sided",conf.level = 0.95)
```

```
##
```

```
##  Welch Two Sample t-test
##
## data:  pre and post
## t = -0.37209, df = 17.846, p-value = 0.7142
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -75.8069  53.0069
## sample estimates:
## mean of x mean of y
##     258.9     270.3
```

```
v1 <- var(pre)
v2 <- var(post)
n1 <- length(pre)
n2 <- length(post)

se <- sqrt(v1/n1 + v2/n2)
nu <- se^4 / ((v1^2 /(n1^2*(n1 -1))) + (v2^2/(n2^2*(n2-1))))
nu
```

```
## [1] 17.84563
```

```
#t-test
s <- (v1/n1 + v2/n2)^(1/2)
v <- ((v1/n1 + v2/n2)^2)/((1/(n1 - 1))*((v1/n1)^2) + (1/(n2 - 1))*((v2/n2)^2)) - 2
s
```

```
## [1] 30.63749
```

```
v
```

```
## [1] 15.84563
```

```
#95% Confidence Interval
mean(pre)
```

```
## [1] 258.9
```

```
mean(post)
```

```
## [1] 270.3
```

```
mean(pre) - mean(post) + c(1, -1)* qt(.975, nu)*se
```

```
## [1]  53.0069 -75.8069
```

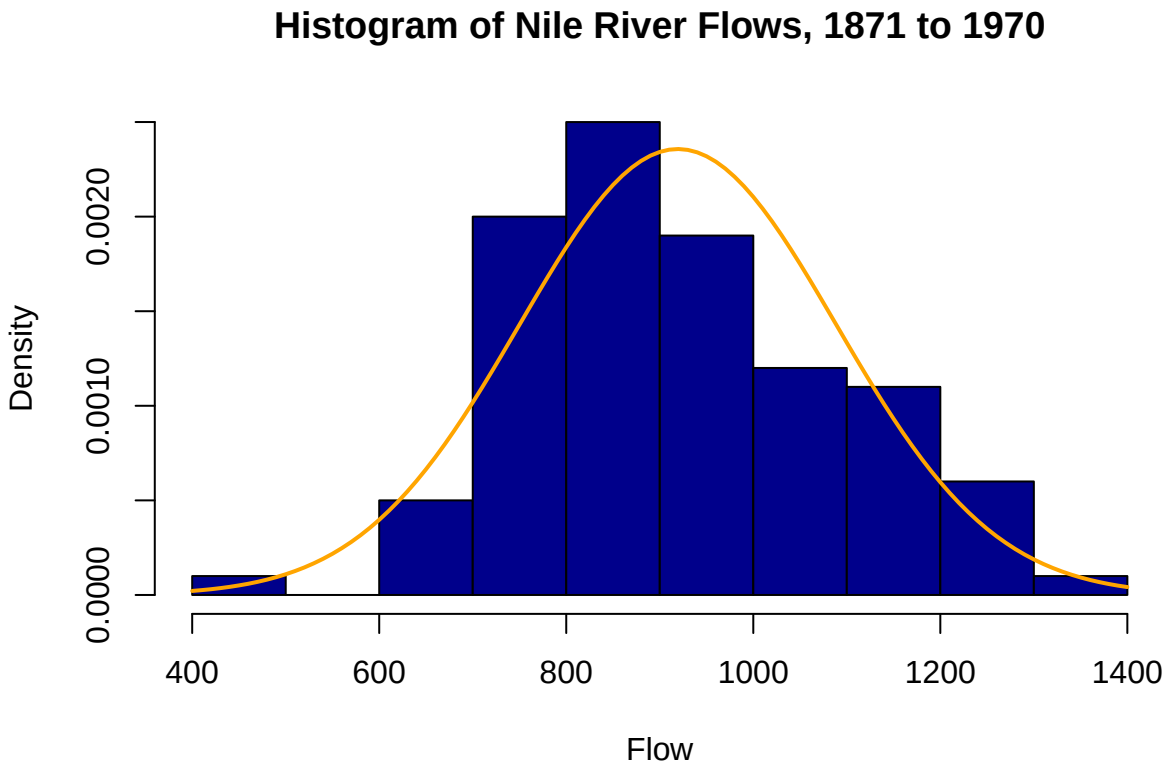
```
#p-value
2*pt(q= -.37209, df = nu)
```

```
## [1] 0.7142076
```

Section 4: (15 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. Refer to this example when completing (4)(c) below.

```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 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.

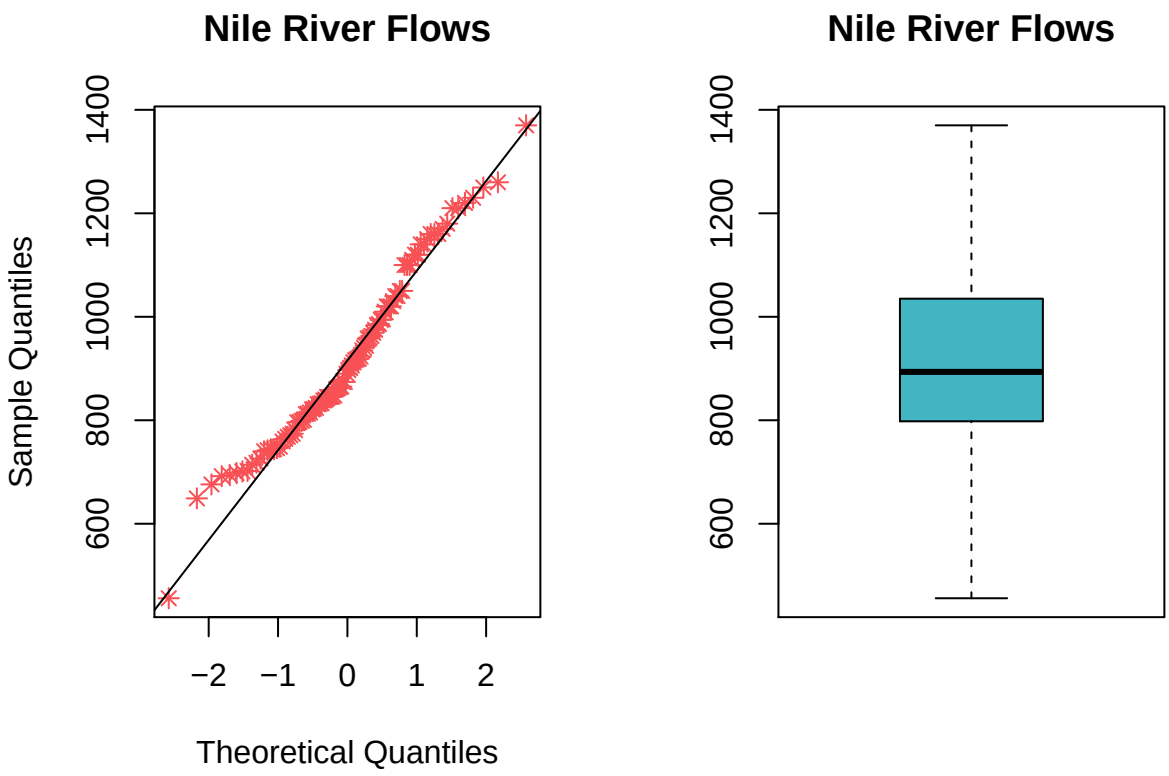
```r
library(moments)

ns=skewness(Nile)
nk=kurtosis(Nile)
ns
```

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

```r
nk
```

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

```r
par(mfrow=c(1,2))
qqnorm(Nile,pch=8, col="#f85155", main = "Nile River Flows")
qqline(Nile)
boxplot(Nile, col="#41b5c2", main = "Nile River Flows")
```



(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.

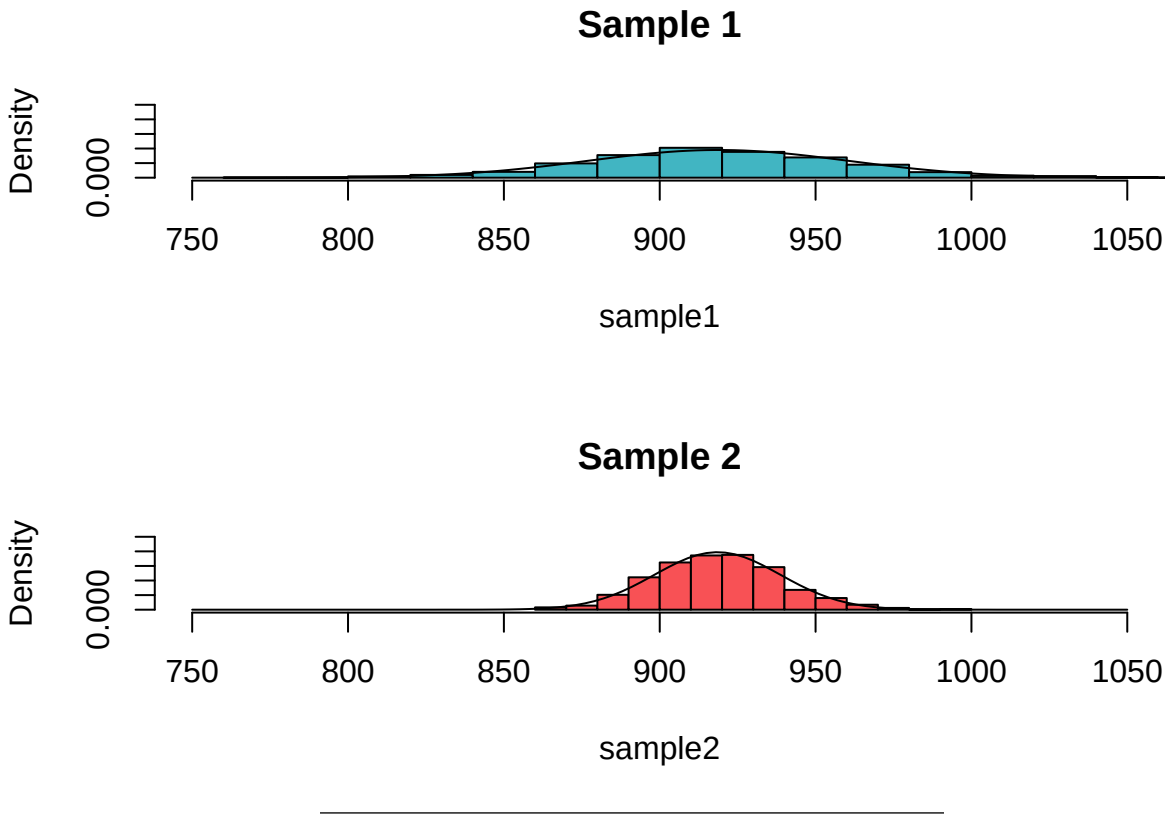
```
set.seed(124)
sample1<-rep(1:1000,0)
for(x in 1:1000) {
  sample1[x]<-mean(sample(Nile,16,replace=TRUE))
}

set.seed(127)
sample2<-rep(1:1000,0)

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

row_names<-c("sample1","sample2")
col_names<-c("Mean","Sample SD"," Sample Var")
matrix(nrow = 2,ncol = 3, c(mean(sample1),mean(sample2),sd(sample1),sd(sample2),var(sample1),var(sample
```

```
##             Mean Sample SD  Sample Var
## sample1 918.7364  42.00156   1764.1312
## sample2 918.5149  20.22883    409.2054
```

(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.**

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

hist(sample1,col="#41b5c2", main="Sample 1", xlim = c(750,1050),ylim = c(0, 0.025), freq = FALSE)
curve(dnorm(x, mean=mean(sample1),sd=sd(sample1)), add=TRUE)

hist(sample2,col="#f85155", main="Sample 2", xlim = c(750,1050),ylim = c(0, 0.025), freq = FALSE)
curve(dnorm(x,mean=mean(sample2),sd=sd(sample2)),add=TRUE)
```



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.

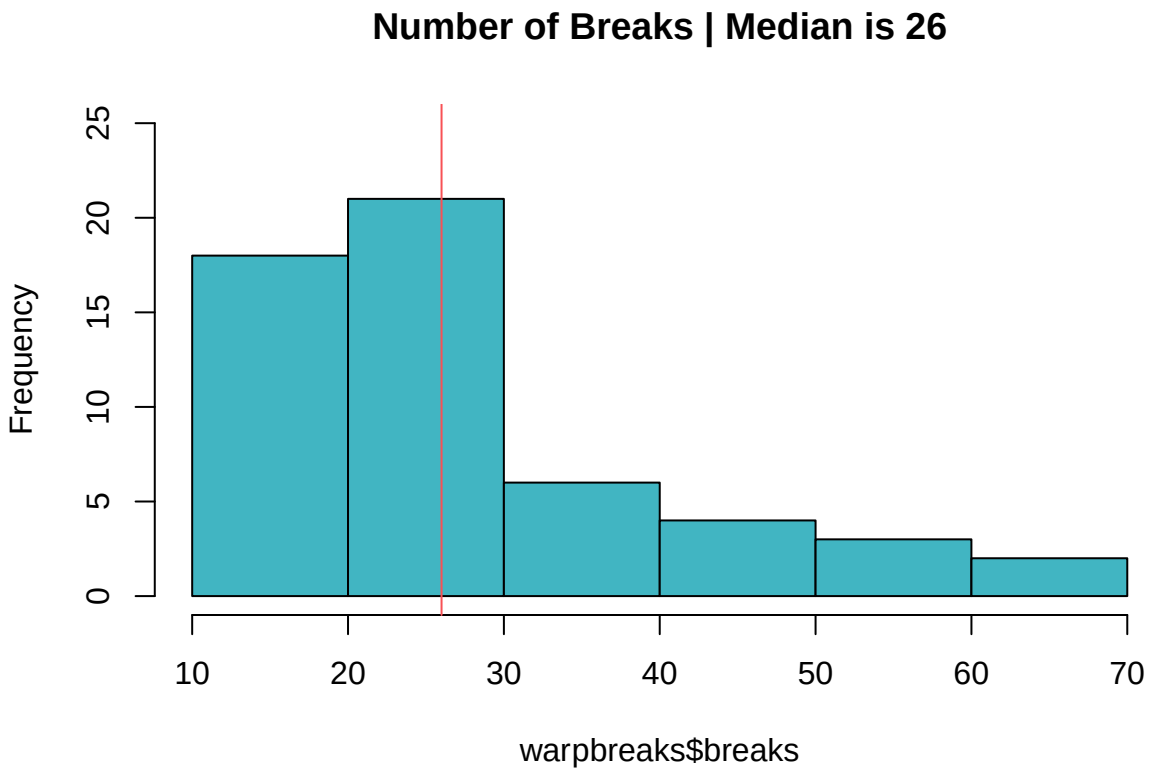
```
data(warpbreaks)
str(warpbreaks)
```

```
## 'data.frame':    54 obs. of  3 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 ...
```

```
median_breaks<-median(warpbreaks$breaks)
median_breaks
```

```
## [1] 26
```

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

hist(warpbreaks$breaks, col="#41b5c2", main="Number of Breaks | Median is 26", ylim= c(0,25))
abline(v = median(warpbreaks$breaks), col = "#f85155")
```




```
number<-ifelse(warpbreaks$breaks<median_breaks,"below","above")
number
```

```
##  [1] "above" "above" "above" "below" "above" "above" "above" "above" "above"
## [10] "below" "below" "above" "below" "below" "below" "above" "above" "above"
## [19] "above" "below" "below" "below" "below" "above" "above" "below" "above"
## [28] "above" "below" "above" "below" "above" "above" "above" "below" "above"
## [37] "above" "above" "below" "below" "above" "above" "below" "above" "above"
## [46] "below" "below" "below" "below" "below" "below" "below" "below" "above"
```

```
warpbreaks2<-cbind(warpbreaks,number)
warpbreaks2
```

```
##    breaks wool tension number
## 1      26    A       L  above
## 2      30    A       L  above
## 3      54    A       L  above
## 4      25    A       L  below
## 5      70    A       L  above
## 6      52    A       L  above
## 7      51    A       L  above
## 8      26    A       L  above
## 9      67    A       L  above
## 10     18    A       M  below
## 11     21    A       M  below
## 12     29    A       M  above
## 13     17    A       M  below
## 14     12    A       M  below
## 15     18    A       M  below
## 16     35    A       M  above
## 17     30    A       M  above
## 18     36    A       M  above
## 19     36    A       H  above
## 20     21    A       H  below
## 21     24    A       H  below
## 22     18    A       H  below
## 23     10    A       H  below
## 24     43    A       H  above
## 25     28    A       H  above
## 26     15    A       H  below
## 27     26    A       H  above
## 28     27    B       L  above
## 29     14    B       L  below
## 30     29    B       L  above
## 31     19    B       L  below
## 32     29    B       L  above
## 33     31    B       L  above
## 34     41    B       L  above
## 35     20    B       L  below
## 36     44    B       L  above
## 37     42    B       M  above
## 38     26    B       M  above
## 39     19    B       M  below
## 40     16    B       M  below
## 41     39    B       M  above
## 42     28    B       M  above
## 43     21    B       M  below
## 44     39    B       M  above
## 45     29    B       M  above
## 46     20    B       H  below
## 47     21    B       H  below
## 48     24    B       H  below
## 49     17    B       H  below
## 50     13    B       H  below
```

```
## 51     15    B       H  below
## 52     15    B       H  below
## 53     16    B       H  below
## 54     28    B       H  above
```

```
summary(warpbreaks2)
```

```
##      breaks      wool   tension    number
##  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
```

```
cont_table<-table(warpbreaks2$tension,warpbreaks2$number)
cont_table
```

```
##
##     above below
##   L    14     4
##   M    10     8
##   H     5    13
```

(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.

```
chisq.test(cont_table)
```

```
##
##  Pearson's Chi-squared test
##
## data:  cont_table
## X-squared = 9.0869, df = 2, p-value = 0.01064
```

```
#The p-value, .01064, is small and this is why we reject the null hypothesis. The variables are dependes
```

(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).

```
data(warpbreaks)
number<-ifelse(warpbreaks$breaks<median_breaks,"below","above")
x<-cbind(warpbreaks,number)
tablex<-table(x$tension,x$number)

chi <- function(x) {

     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]
```

```
   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, 2, lower.tail = F)))
     }

x<-addmargins(tablex)
chi(x)
```

```
## $`chi-squared`
## [1] 9.086897
##
## $`p-value`
## [1] 0.01063667
```

```
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 1:3) {
       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) - 1
  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))
}

chisqfun(cont_table)
```

```
## Pearson's Chi-squared test \n Chi sq:  9.086897 ;
##             Degree of Freedom : 2  ; P-value : 0.01063667
```