Kumar Ritesh Programming with R Assignment #2 (75 points)

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()* or *ppois()* with probability p = 0.05, and n = 100. Then, estimate the same probabilities using *dbinom()* or *pbinom()*. Show the numerical results of your calculations.

i. The probability of exactly 0 successes.

```
p = 0.05
n = 100
lambda = n*p
x = 0
sprintf('dpois: %5f', dpois(x,lambda))
```

```
## [1] "dpois: 0.006738"
```

```
sprintf('ppois: %5f', ppois(x,lambda))
```

```
## [1] "ppois: 0.006738"
```

```
sprintf('dbinom: %5f', dbinom(x, n, p))
```

```
## [1] "dbinom: 0.005921"
```

```
sprintf('pbinom: %5f', pbinom(x, n, p))
```

```
## [1] "pbinom: 0.005921"
```

ii. The probability of fewer than 7 successes. Please note the following, taken from the Binomial Distribution R Documentation page, regarding the “lower.tail” argument:

lower.tail logical; if TRUE (default), probabilities are P[X ??? x], otherwise, P[X > x].

```
p = 0.05
n = 100
x = 0:6
prob_lt7 = pbinom(x, n, p, lower.tail=TRUE)
prob_6 = pbinom(x, n, p, lower.tail=FALSE)
cat('P[x < 7], If lower.tail=TRUE, probabilities are: ', prob_lt7, '\n')
```

```
## P[x < 7], If lower.tail=TRUE, probabilities are:  0.005920529 0.03708121 0.118263 0.2578387
0.4359813 0.6159991 0.766014
```

```
cat('Otherwise, for P[x >= 7], the probabilities are: ', prob_6)
```

```
## Otherwise, for P[x >= 7], the probabilities are:  0.9940795 0.9629188 0.881737 0.7421613 0.56
40187 0.3840009 0.233986
```

The binomial may also be approximated via the normal distribution. Estimate the following binomial probabilities using *dnorm()* or *pnorm()*, this time with probability p = 0.2 and n = 100. Then, calculate the same probabilities using *dbinom()* and *pbinom()*. Use continuity correction. Show the numerical results of your calculations.

iii. The probability of exactly 25 successes.

```
x = 25
n = 100
p = 0.2
mu = n*p
sigma = sqrt(n*p*(1-p))
binom_probs = dbinom(x, n, p)
binom_cumulative_probs = pbinom(x, n, p)
cat('Binomial: ', binom_probs,'\n')
```

```
## Binomial:  0.04387783
```

```
cat('Cumulative Binomial: ', binom_cumulative_probs,'\n')
```

```
## Cumulative Binomial:  0.9125246
```

```
normal_probs = dnorm(x+0.5, mu, sigma) # 0.5 is the continuity correction
normmal_cumulative_probs = pnorm(x+0.5, mu, sigma)
cat('Normal Approximation: ', normal_probs,'\n')
```

```
## Normal Approximation:  0.03875307
```

```
cat('Cumulative Normal Approximation: ', normmal_cumulative_probs,'\n')
```

```
## Cumulative Normal Approximation:  0.9154343
```

iv. The probability of fewer than 25 successes. Please note the following, taken from the Normal Distribution R Documentation page, regarding the “lower.tail” argument:

lower.tail logical; if TRUE (default), probabilities are P[X ??? x], otherwise, P[X > x].

```
x = 0:24
n = 100
p = 0.2
mu = n*p
sigma = sqrt(n*p*(1-p))
cat('P[x < 25], If lower.tail=TRUE, probabilities are: ', pnorm(x, mean=mu, sd=sigma, lower.tail
=TRUE), '\n')
```

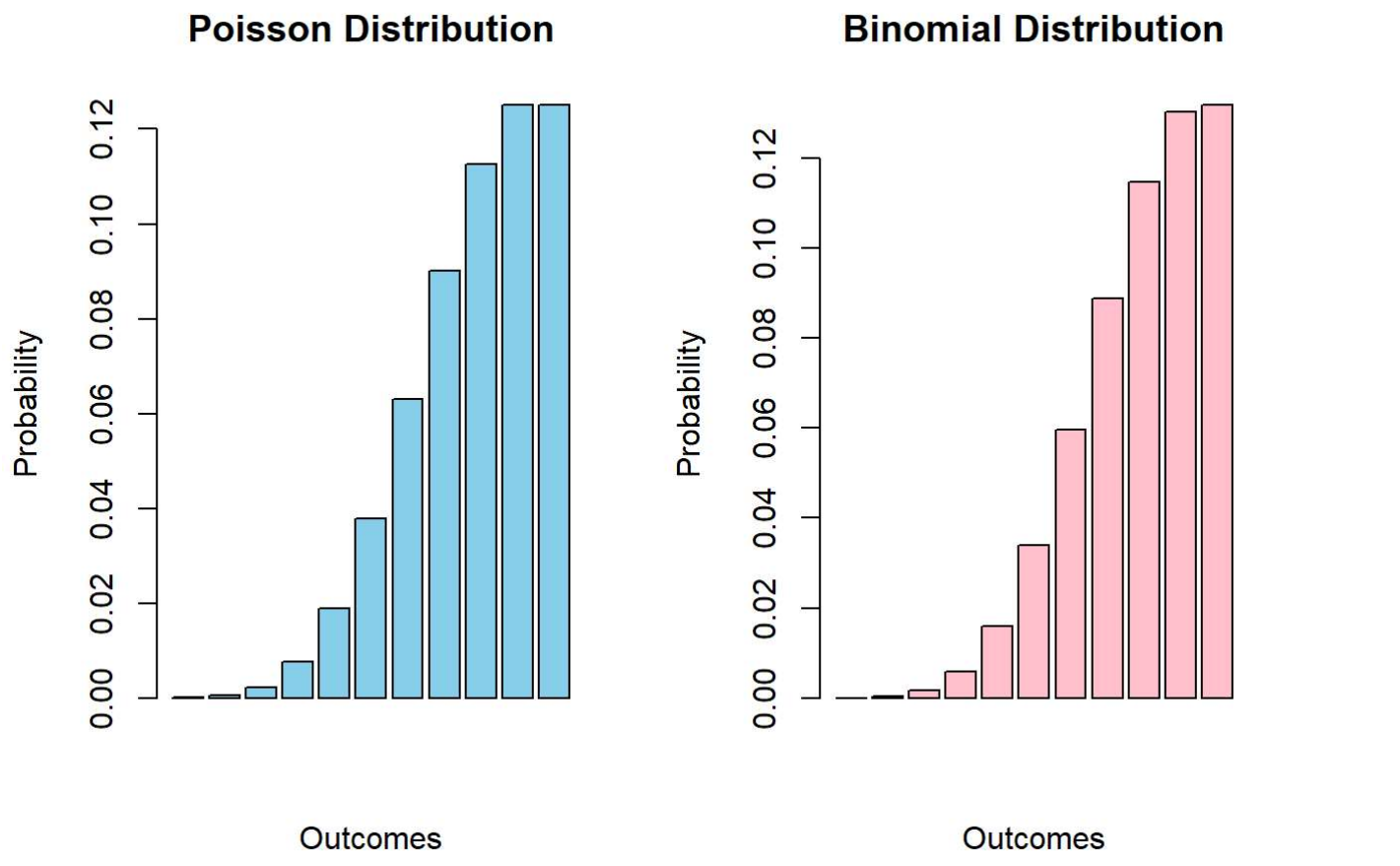
```
## P[x < 25], If lower.tail=TRUE, probabilities are:  2.866516e-07 1.017083e-06 3.397673e-06 1.0
68853e-05 3.167124e-05 8.841729e-05 0.0002326291 0.000577025 0.001349898 0.002979763 0.006209665
0.01222447 0.02275013 0.04005916 0.0668072 0.1056498 0.1586553 0.2266274 0.3085375 0.4012937 0.5
0.5987063 0.6914625 0.7733726 0.8413447
```

```
cat('Otherwise, for P[x >= 25], the probabilities are: ', pnorm(x, mean=mu, sd=sigma, lower.tail
=FALSE))
```

```
## Otherwise, for P[x >= 25], the probabilities are:  0.9999997 0.999999 0.9999966 0.9999893 0.9
999683 0.9999116 0.9997674 0.999423 0.9986501 0.9970202 0.9937903 0.9877755 0.9772499 0.9599408
0.9331928 0.8943502 0.8413447 0.7733726 0.6914625 0.5987063 0.5 0.4012937 0.3085375 0.2266274 0.
1586553
```

(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 = 0.1 and n = 100. Title each plot, present in color and assign names to the bar; i.e. x-axis value labels.

```
n = 100
p = 0.1
lambda = n*p
x = 0:10
poisson_probs = dpois(x, lambda)
binom_probs = dbinom(x, n, p)
par(mfrow = c(1,2))
barplot(poisson_probs, main ="Poisson Distribution", col="skyblue", xlab = "Outcomes", ylab ="Pr
obability")
barplot(binom_probs, main = "Binomial Distribution", col = "pink", xlab = "Outcomes", ylab = "Pr
obability")
```



(1)(c) (6 points) 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. 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 1 decimal place.

```
x = c(0, 1, 2, 3, 4, 5, 6)
p = c(0.215, 0.230, 0.240, 0.182, 0.130, 0.003, 0.001)
Expected_Value = sum(x*p)
Variance = sum(((x-Expected_Value)^2)*p)
cat("Expected Value: ", round(Expected_Value, 1), '\n')
```

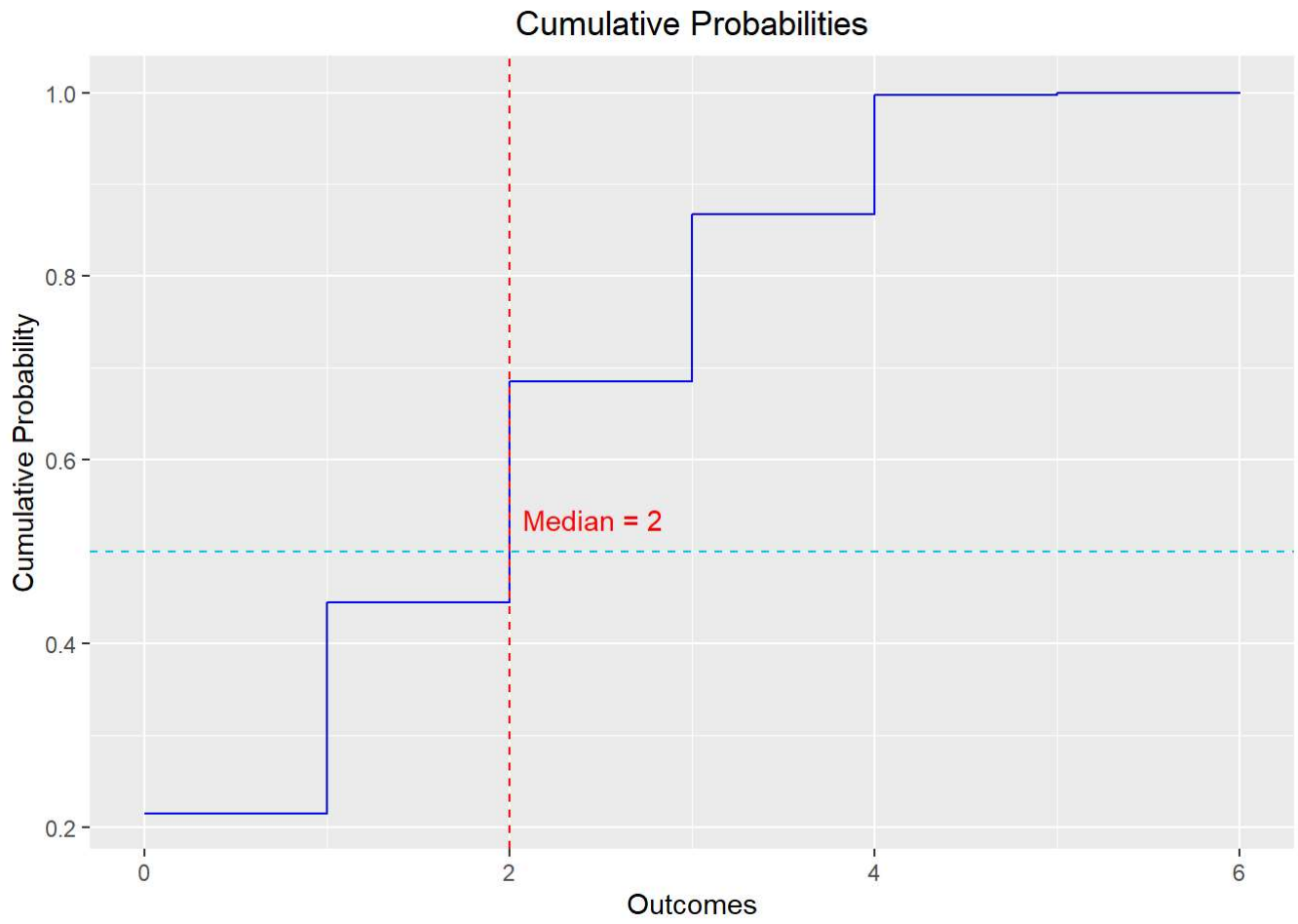
```
## Expected Value:  1.8
```

```
cat("Variance (rounded to 1 decimal place): ", round(Variance, 1), "\n")
```

```
## Variance (rounded to 1 decimal place):  1.8
```

ii. 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. Note that there are methods for interpolating a median. However, we can identify an appropriate median from our set of our outcomes - 0 through 6 - that satisfies the definition. Creating a stair-step plot of the cumulative probability as a function of the outcomes may be helpful in identifying it.

```
x = c(0, 1, 2, 3, 4, 5, 6)
p = c(0.215, 0.230, 0.240, 0.182, 0.130, 0.003, 0.001)
cumulative_p = cumsum(p)
median_x <- x[min(which(cumulative_p >= 0.5))]
df <- data.frame(x, Cumulative_Probability = cumulative_p)
ggplot(df, aes(x, Cumulative_Probability)) + geom_step(color = "blue") + geom_hline(yintercept =
0.5, color='deepskyblue', linetype = "dashed") + geom_vline(xintercept = median_x, linetype = "d
ashed", color = "red") +  annotate("text", x = median_x, y = 0.5, label = paste("Median =", medi
an_x), hjust = -0.1, vjust = -1, color='red')+labs(title = "Cumulative Probabilities", x = "Outc
omes", y = "Cumulative Probability") + theme(plot.title = element_text(hjust = 0.5))
```



Section 2: (15 points)

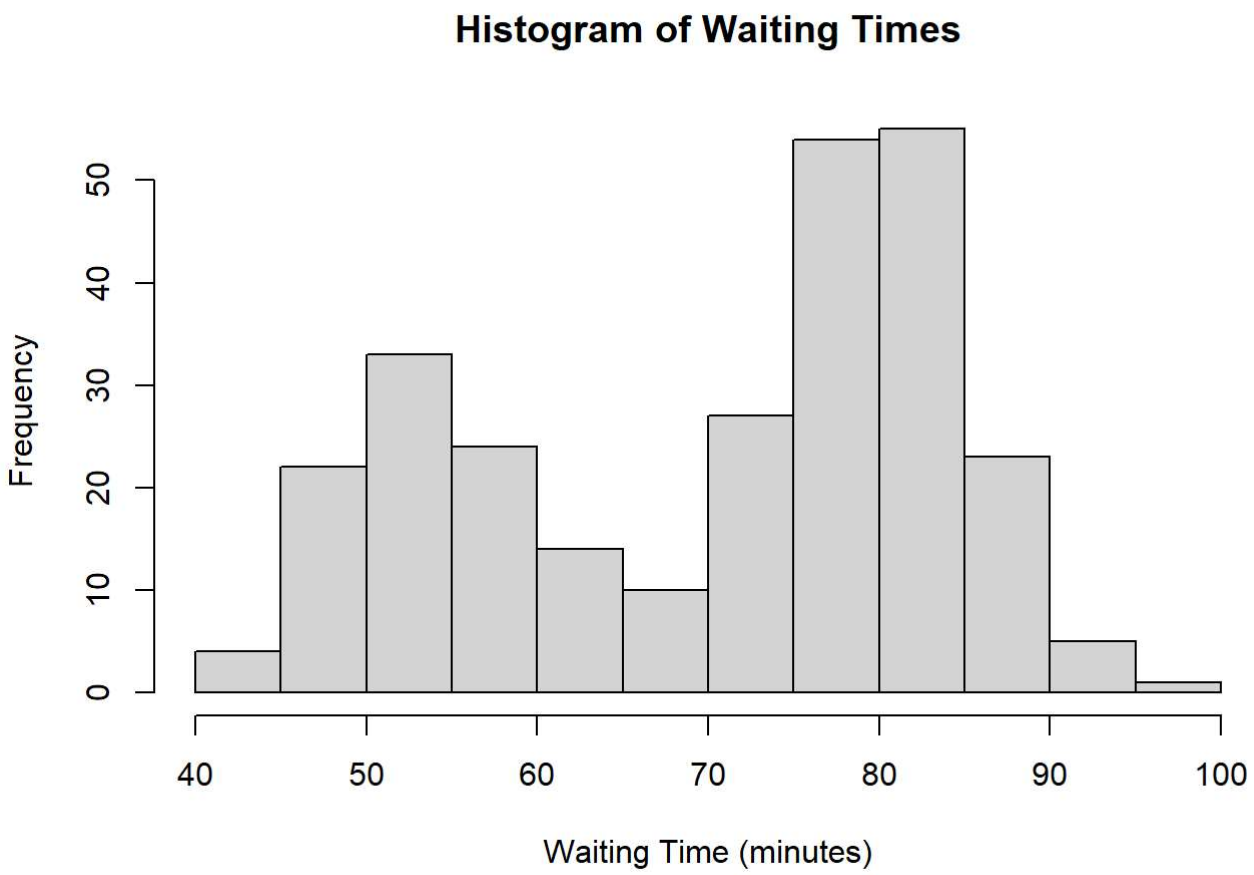
(2) Conditional probabilities appear in many contexts and, in particular, are used by Bayes' Theorem. Correlations are another means for evaluating dependency between variables. The dataset "faithful"" is part of the "datasets" package and may be loaded with the statement *data(faithful)*. It contains 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) (6 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 3.5 minutes, if the waiting time exceeds 70 minutes.

```
data(faithful, package = "datasets")
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
```

```
hist(faithful$waiting, main = "Histogram of Waiting Times", xlab = "Waiting Time (minutes)")
```

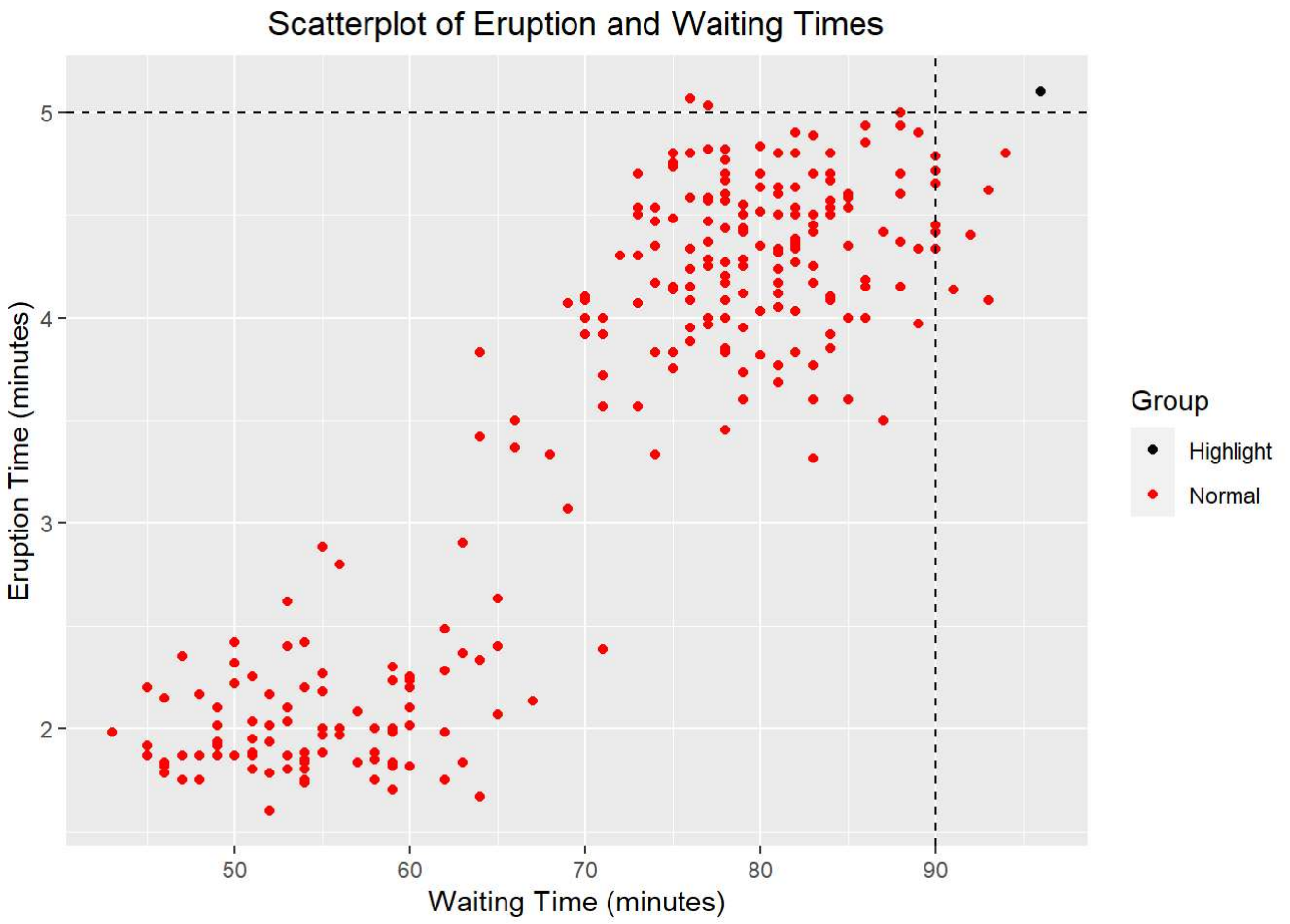



```
total_cases = sum(faithful$waiting > 70)
favorable_cases = sum(faithful$waiting > 70 & faithful$eruptions < 3.5)
conditional_probability = favorable_cases/total_cases
cat('\n','Conditional Probability: ', round(conditional_probability, 6))
```

```
##
##  Conditional Probability:  0.024242
```

i. Identify any observations in “faithful” for which the waiting time exceeds 90 minutes and the eruptions last longer than 5 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 = 5.0, and a vertical line at waiting time = 90. Add a title and appropriate text.

```
faithful$group <- ifelse(faithful$waiting > 90 & faithful$eruptions > 5, "Highlight", "Normal")
ggplot(faithful, aes(x = waiting, y = eruptions, color = group)) + geom_point() +  geom_hline(yi
ntercept = 5, linetype = "dashed", color = "black") + geom_vline(xintercept = 90, linetype = "da
shed", color = "black") + scale_color_manual(values = c("black", "red")) + labs(title = "Scatter
plot of Eruption and Waiting Times", x = "Waiting Time (minutes)", y = "Eruption Time (minute
s)", color = "Group") + theme(plot.title = element_text(hjust = 0.5))
```



ii. What does the plot suggest about the relationship between eruption time and waiting time?

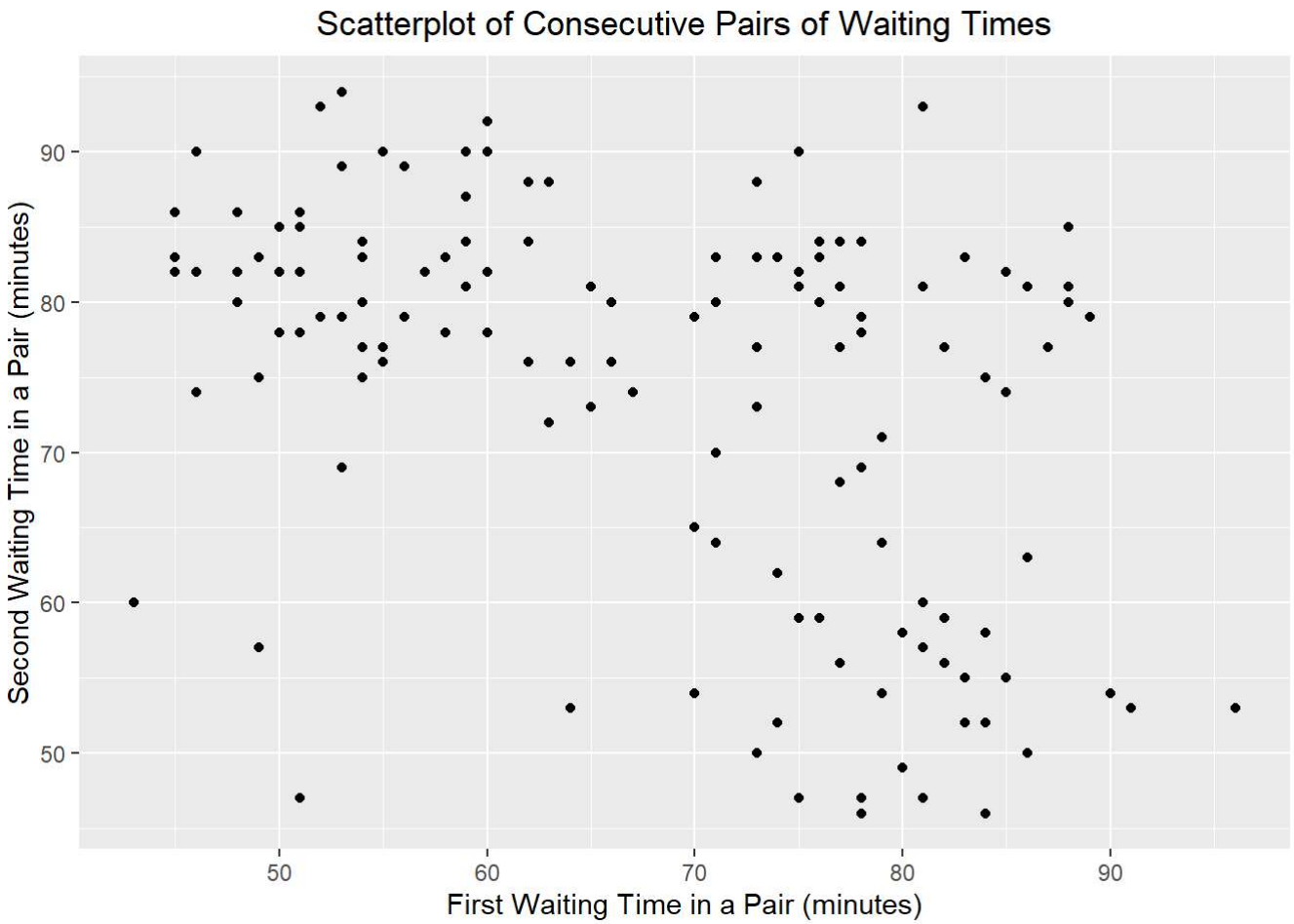
Answer: A strong positive correlation between eruption time and waiting time is evident from the plot. It suggests that longer wait times are typically associated with longer eruptions, and shorter wait times with shorter eruptions. One data point showcases a waiting time over 90 minutes with an eruption lasting more than 5 minutes. While this strong correlation implies a consistent relationship, it does not confirm causation, and other factors may influence both eruption and waiting times.

---

(2)(b) (6 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.

```
waiting_pairs_matrix = matrix(faithful$waiting, ncol=2, byrow=TRUE)
waiting_pairs = data.frame(waiting_pairs_matrix)
names(waiting_pairs) = c("waiting1", "waiting2")
ggplot(waiting_pairs, aes(x = waiting1, y = waiting2)) + geom_point() + labs(title = "Scatterplo
t of Consecutive Pairs of Waiting Times", x = "First Waiting Time in a Pair (minutes)", y = "Sec
ond Waiting Time in a Pair (minutes)") + theme(plot.title = element_text(hjust = 0.5))
```



(2)(c) (3 points) Test the hypothesis of independence with a two-sided test at the 99% confidence level using the Kendall correlation coefficient. The *cor.test()* function can be used to structure this test and specify the appropriate - Kendall's tau - method.

```
cor.test(waiting_pairs$waiting1, waiting_pairs$waiting2, method = "kendall", alternative = "two.
sided", conf.level = 0.99)
```

```
##
##  Kendall's rank correlation tau
##
## data:  waiting_pairs$waiting1 and waiting_pairs$waiting2
## 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, we will create a subset of the "ChickWeight" data frame. Specifically, we want to create a data frame that includes only those rows where Time == 21 AND Diet == 1 or 3.

```
# load "ChickWeight" dataset
data(ChickWeight, package = "datasets")
# There are multiple ways to approach the subsetting task. The method you choose is up
# to you.
result =  subset(ChickWeight, Time == 21 & (Diet == 1 | Diet == 3))
head(result)
```

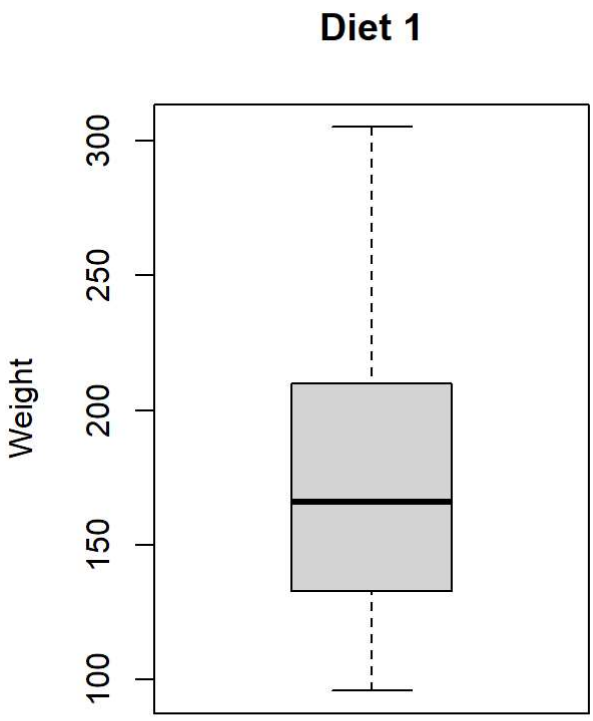
```
##    weight Time Chick Diet
## 12    205   21     1    1
## 24    215   21     2    1
## 36    202   21     3    1
## 48    157   21     4    1
## 60    223   21     5    1
## 72    157   21     6    1
```

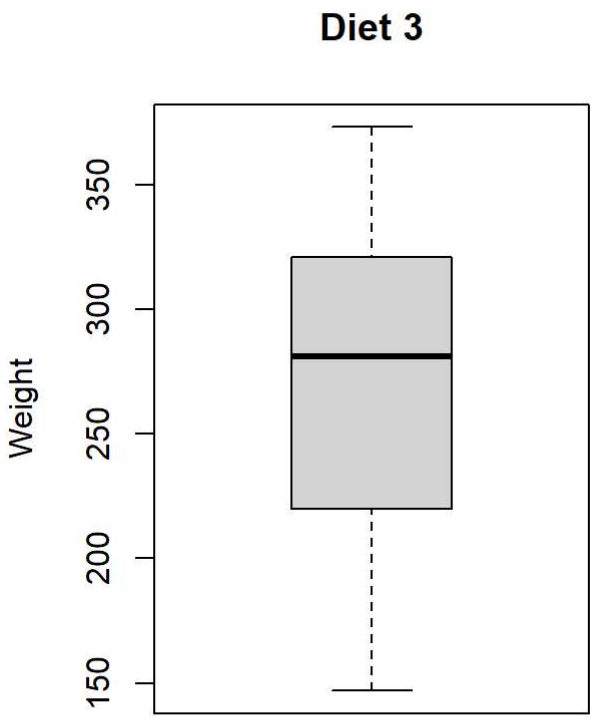
```
# The values in your subsetted data frame should match those below:
# > head(df)
#    weight Time Chick Diet
# 12    205   21     1    1
# 24    215   21     2    1
# 36    202   21     3    1
# 48    157   21     4    1
# 60    223   21     5    1
# 72    157   21     6    1
```

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))
boxplot(result$weight[result$Diet == 1], main="Diet 1", ylab = "Weight")
boxplot(result$weight[result$Diet == 3], main="Diet 3", ylab = "Weight")
```





(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. Do not assume equal variances. Display the results of *t.test()*.

```
t.test(result$weight[result$Diet == 1], result$weight[result$Diet == 3], alternative = "two.side
d", conf.level = 0.95, var.equal = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  result$weight[result$Diet == 1] and result$weight[result$Diet == 3]
## 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. This time, we will look only at those individuals on Diet == “3”. You will need to add code below creating two (2) vectors. One (1) vector should include all the Time == 20 weights of those individuals on Diet == “3”; a second should include all the Time == 21 weights of those individuals on Diet == “3”.

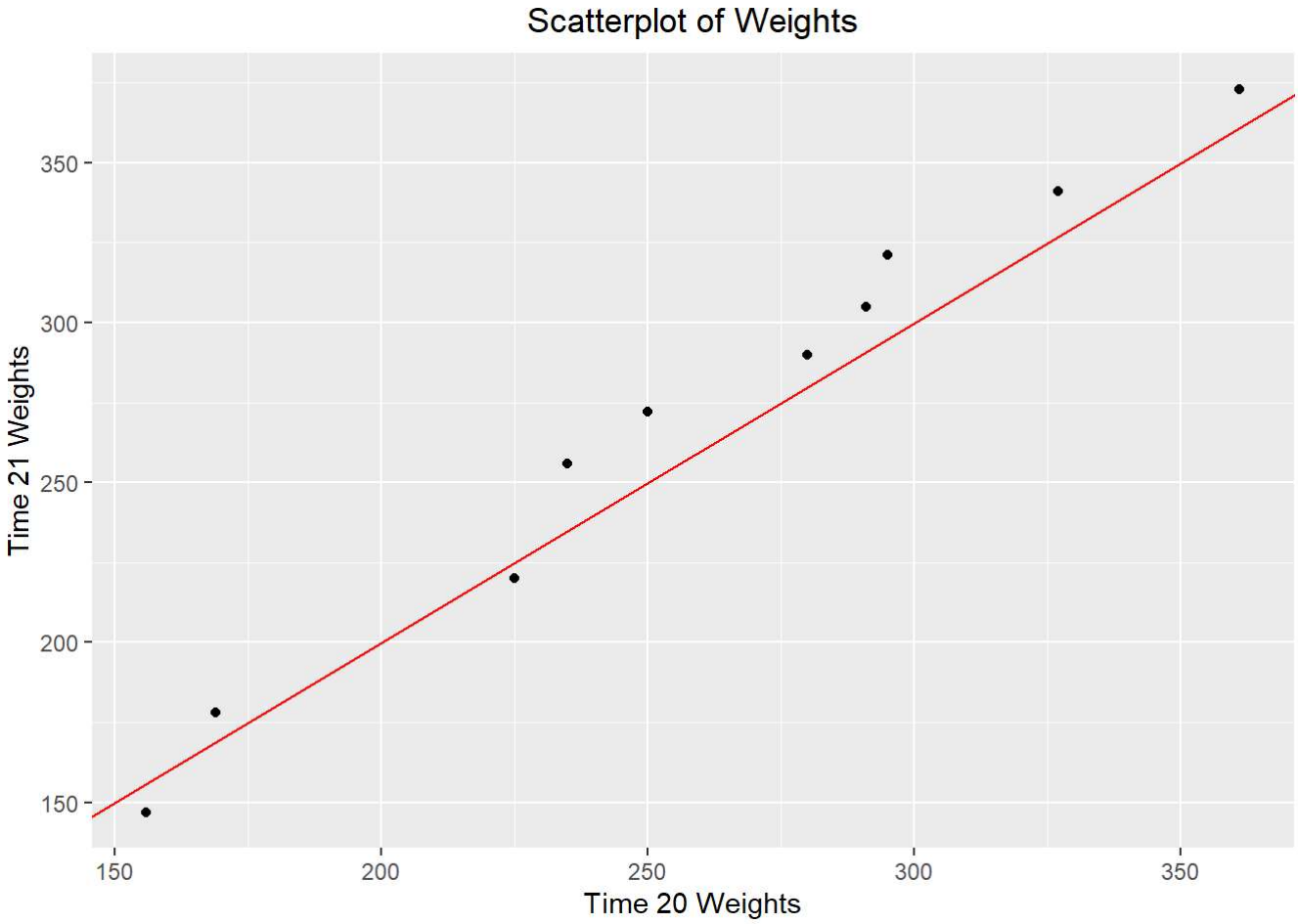
```
# There are multiple ways to approach the subsetting task. The method you choose is up
# to you.
weights_time20_diet3 = ChickWeight$weight[ChickWeight$Time == 20 & ChickWeight$Diet == 3]
weights_time21_diet3 = ChickWeight$weight[ChickWeight$Time == 21 & ChickWeight$Diet == 3]
head(weights_time20_diet3)
```

```
## [1] 235 291 156 327 361 225
```

```
# The first six (6) elements of your Time == 20 vector should match those below:
# [1] 235 291 156 327 361 225
```

(3)(c) (3 points) Present a scatterplot of the Time == 21 weights as a function of the Time == 20 weights. Include a diagonal line with zero intercept and slope equal to one. Title and label the variables in this scatterplot.

```
ggplot() + geom_point(aes(x = weights_time20_diet3, y = weights_time21_diet3)) + geom_abline(int
ercept = 0, slope = 1, color = "red") + labs(title = "Scatterplot of Weights", x = "Time 20 Weig
hts", y = "Time 21 Weights") + theme(plot.title = element_text(hjust = 0.5))
```



(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.

```
n = length(weights_time21_diet3) # number of pairs
d = weights_time21_diet3 - weights_time20_diet3 # sample difference in pairs
d_bar = mean(d) # Mean Sample Difference
sd = sd(d) # Standard Deviation of Sample Difference
df = n-1 # Degrees of Freedom
crit_value = qt(0.95, df) # Critical Value
std_err <- sd/sqrt(n) # Standard Error
t_stat = d_bar/std_err
p_value <- pt(t_stat, df, lower.tail = FALSE)
conf_int <- c(d_bar - crit_value*std_err, Inf)
cat("Test statistic value: ", round(t_stat,4), "\n")
```

```
## Test statistic value:  3.2253
```

```
cat("Critical value: ", round(crit_value,4), "\n")
```

```
## Critical value:  1.8331
```

```
cat("P-value: ", round(p_value,6), "\n")
```

```
## P-value:  0.005201
```

```
cat("Confidence interval: (", conf_int[1], ", Inf)", '\n')
```

```
## Confidence interval: ( 4.920696 , Inf)
```

```
t.test(weights_time21_diet3, weights_time20_diet3, alternative='greater', paired=TRUE, conf.leve
l=0.95)
```

```
##
##  Paired t-test
##
## data:  weights_time21_diet3 and weights_time20_diet3
## t = 3.2253, df = 9, p-value = 0.005201
## alternative hypothesis: true mean difference is greater than 0
## 95 percent confidence interval:
##  4.920696      Inf
## sample estimates:
## mean difference
##            11.4
```

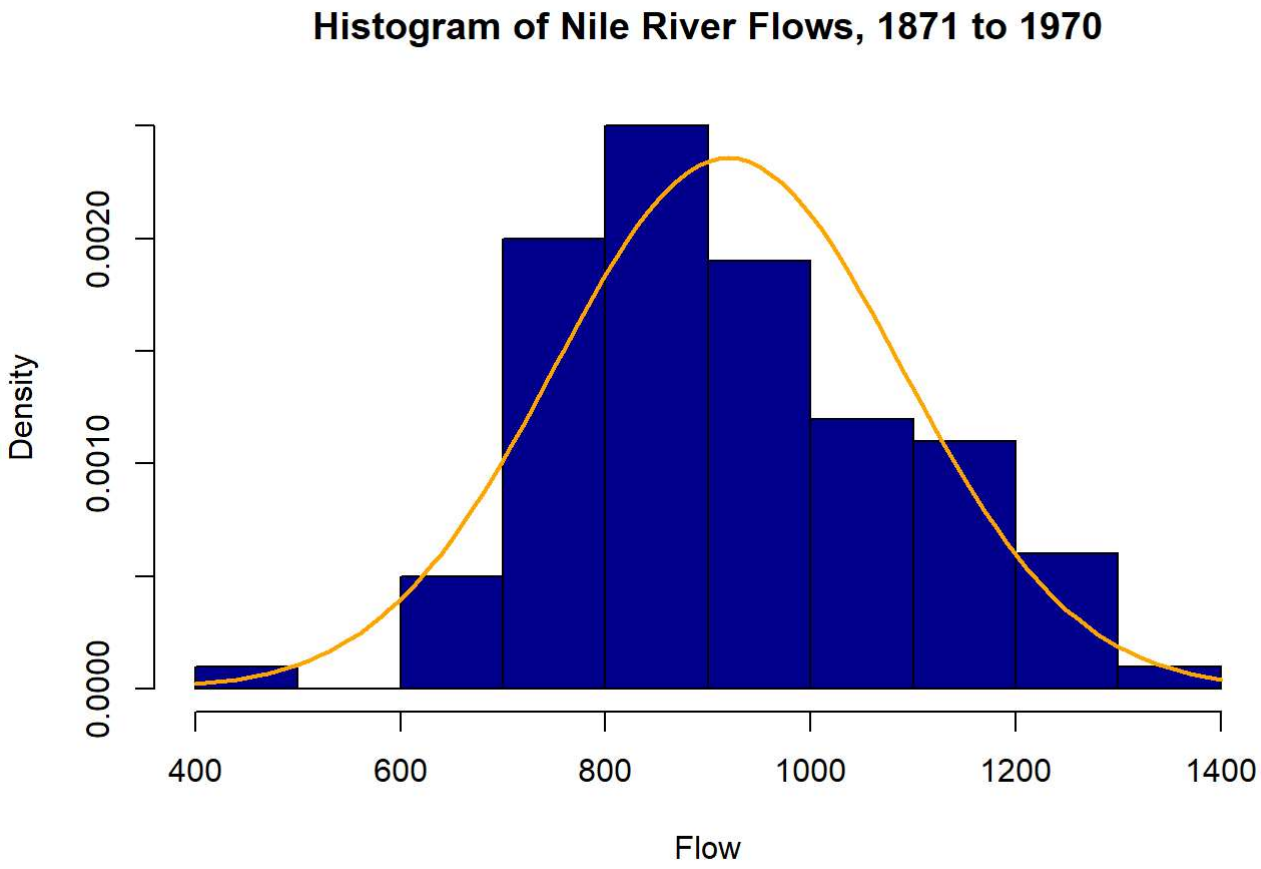
Section 4: (15 points)

(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.

```
data(Nile, package = "datasets")
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.

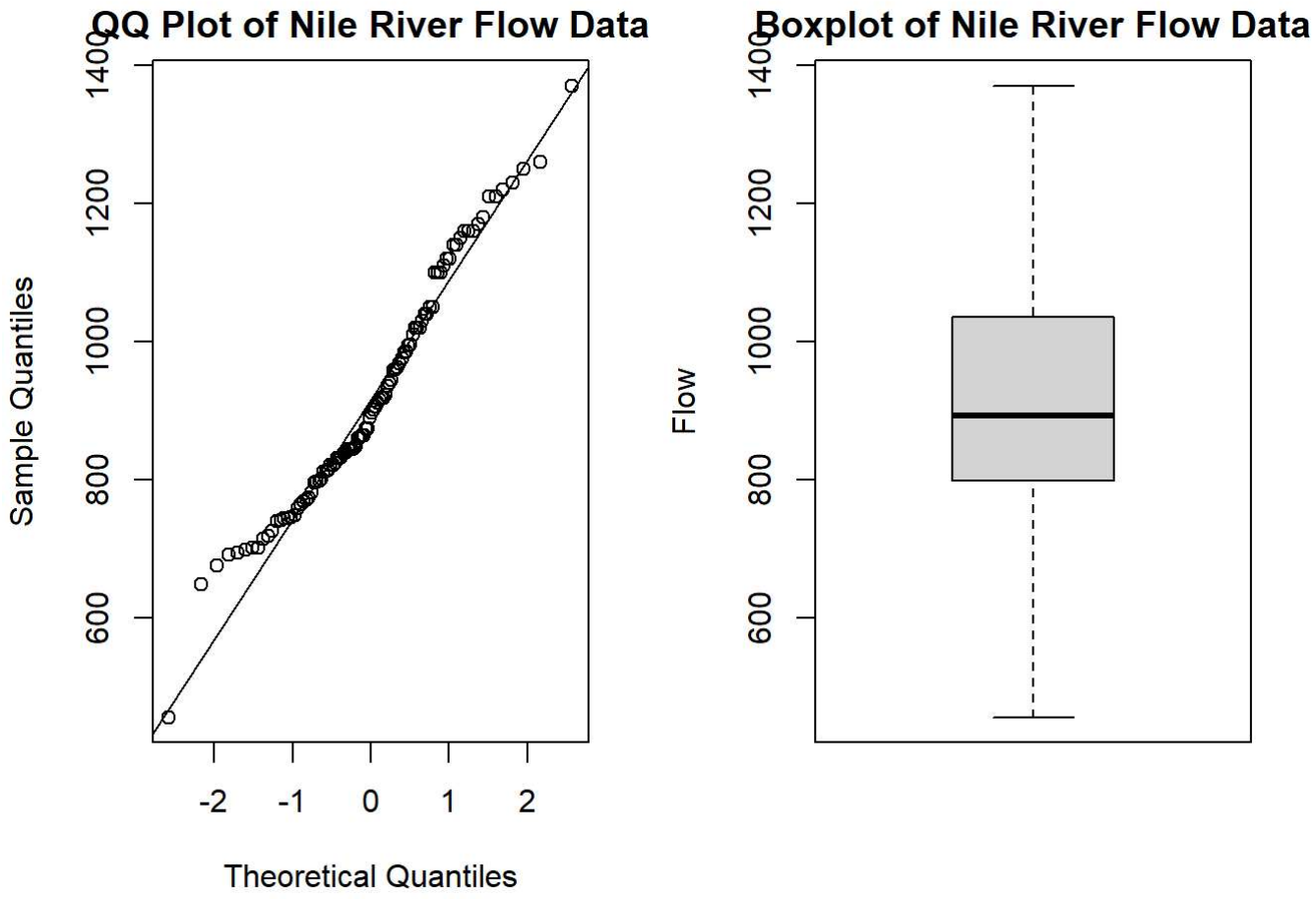
```
nile_skewness = skewness(Nile)
print(paste("Skewness of Nile River flow data: ", nile_skewness))
```

```
## [1] "Skewness of Nile River flow data:  0.322369681723753"
```

```
nile_kurtosis = kurtosis(Nile)
print(paste("Kurtosis of Nile River flow data: ", nile_kurtosis))
```

```
## [1] "Kurtosis of Nile River flow data:  2.69509315497952"
```

```
par(mfrow = c(1, 2), mar = c(5, 4, 2, 2))
qqnorm(Nile, main = "QQ Plot of Nile River Flow Data")
qqline(Nile)
boxplot(Nile, main = "Boxplot of Nile River Flow Data", ylab = "Flow")
```



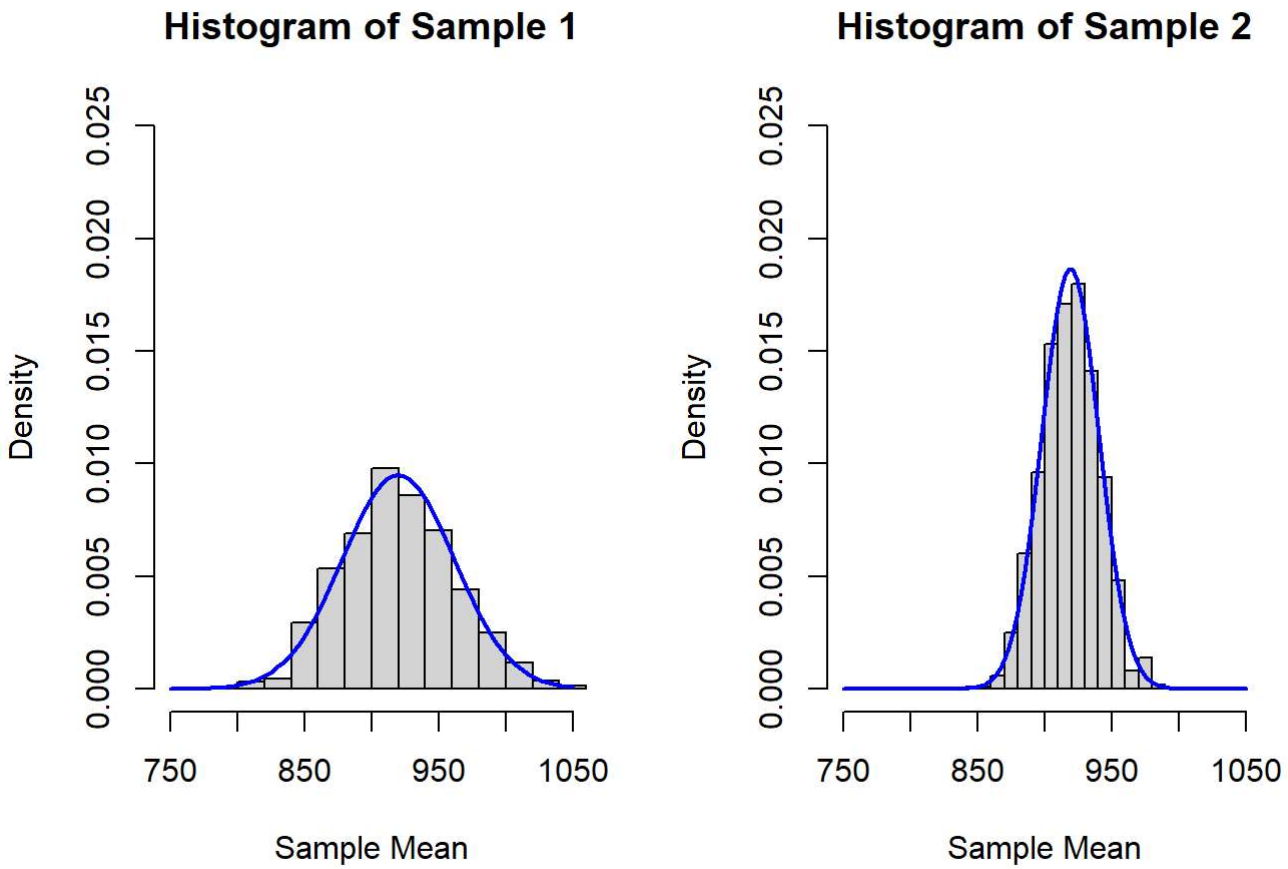
(4)(b) (6 points) Using *set.seed(456)* and the Nile data, generate 1000 random samples of size n = 16, with replacement. For each sample drawn, calculate and store the sample mean. This can be done with a for-loop and use of the *sample()* function. Label the resulting 1000 mean values as “sample1”. **Repeat these steps using *set.seed(789)* - 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(456)
sample1 <- replicate(1000, mean(sample(Nile, size = 16, replace = TRUE)))
set.seed(789)
sample2 <- replicate(1000, mean(sample(Nile, size = 64, replace = TRUE)))
stats <- data.frame( Sample = c("Sample1", "Sample2"), Mean = c(mean(sample1), mean(sample2)), S
D = c(sd(sample1), sd(sample2)), Variance = c(var(sample1), var(sample2)))
print(stats)
```

```
##    Sample     Mean       SD  Variance
## 1 Sample1 919.7673 41.95021 1759.8201
## 2 Sample2 919.0662 21.35711  456.1263
```

(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.**

```
# Create histograms of "sample1" and "sample2" with normal density curves superimposed
par(mfrow = c(1, 2))
hist(sample1, freq = FALSE, xlim = c(750, 1050), ylim = c(0, 0.025), main = "Histogram of Sample
1", xlab = "Sample Mean")
curve(dnorm(x, mean = mean(sample1), sd = sd(sample1)), col = "blue", lwd = 2, add = TRUE)
hist(sample2, freq = FALSE, xlim = c(750, 1050), ylim = c(0, 0.025), main = "Histogram of Sample
2", xlab = "Sample Mean")
curve(dnorm(x, mean = mean(sample2), sd = sd(sample2)), col = "blue", lwd = 2, 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)(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 “wool” and “number”. There should be four cells in this table.

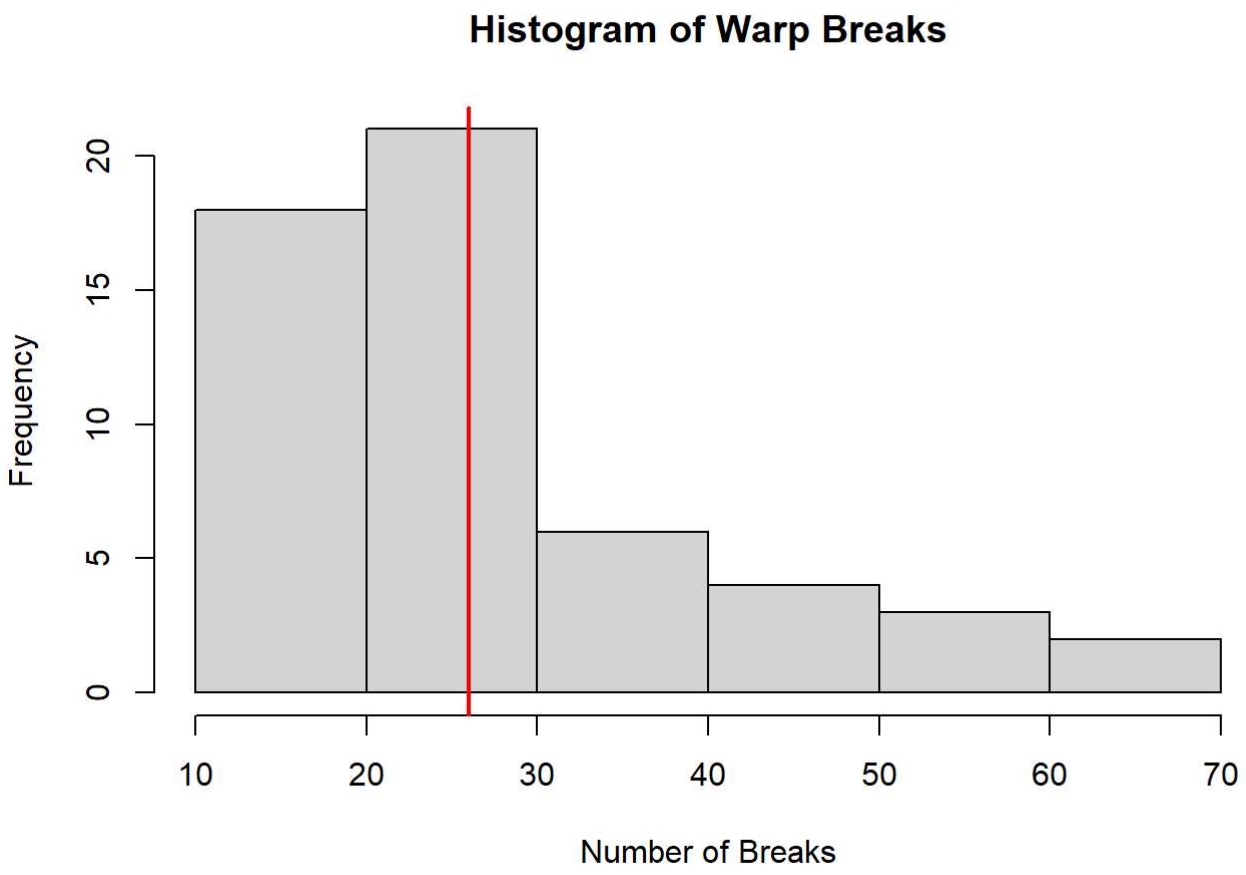
```
data(warpbreaks, package = "datasets")
str(warpbreaks)
```

```
## '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 ...
```

```
cat('\n', 'Median Breaks: ', median(warpbreaks$breaks), '\n')
```

```
##
##  Median Breaks:  26
```

```
hist(warpbreaks$breaks, main = "Histogram of Warp Breaks", xlab = "Number of Breaks")
abline(v = median(warpbreaks$breaks), col = "red", lwd = 2)
```




```
warpbreaks$number = ifelse(warpbreaks$breaks < median(warpbreaks$breaks), "below", "above")
warpbreaks$number = as.factor(warpbreaks$number)
cat('\n')
```

```
summary(warpbreaks)
```

```
##      breaks      wool   tension   number
##  Min.   :10.00   A:27   L:18    above:29
##  1st Qu.:18.25   B:27   M:18    below:25
##  Median :26.00          H:18
##  Mean   :28.15
##  3rd Qu.:34.00
##  Max.   :70.00
```

```
cat('\n')
```

```
table(warpbreaks$wool, warpbreaks$number)
```

```
##
##     above below
##   A    16    11
##   B    13    14
```

(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.

```
contingency_table = table(warpbreaks$wool, warpbreaks$number)
chisq.test(contingency_table)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table
## X-squared = 0.29793, df = 1, p-value = 0.5852
```

```
### Given the p-value of 0.5852, which is greater than the significance level (alpha) of 0.05, w
e do not have sufficient evidence to reject the null hypothesis. Therefore, we conclude that the
data does not provide strong evidence of a relationship between the type of wool and the number
of breaks. In other words, the type of wool seems to have no significant impact on whether the n
umber of breaks is above or below the median, according to this dataset and at the 5% significan
ce level.###
```

(5)(c) (3 points) 'Manually' calculate the chi-squared statistic and p-value of the table from (5)(a). The *addmargins()* function can be used to add row and column sums to the table; useful for calculating the expected values for each cell. You should be able to match the chi-squared and p-values from (5)(b). The underlying code for the *chisq.test()* function can be viewed by entering *chisq.test* - without parentheses - in the Console. You are given code below to create the table, add row and column sums and calculate the expected values for the for the first of two (2) rows. You will need to add code to calculate the expected values for the second row and the chi-squared. The *pchisq()* function can be used to return the p-value.

```
tbl <- table(warpbreaks$wool, warpbreaks$number)
mar_tbl <- addmargins(tbl)

mar_tbl
```

```
##
##       above below Sum
##   A      16    11  27
##   B      13    14  27
##   Sum    29    25  54
```

```
e11 <- mar_tbl[3, 1] * mar_tbl[1, 3] / mar_tbl[3, 3]
e12 <- mar_tbl[3, 2] * mar_tbl[1, 3] / mar_tbl[3, 3]

e21 <- mar_tbl[3, 1] * mar_tbl[2, 3] / mar_tbl[3, 3]
e22 <- mar_tbl[3, 2] * mar_tbl[2, 3] / mar_tbl[3, 3]

chi_squared <- sum((tbl - matrix(c(e11, e21, e12, e22), nrow = 2))^2 / matrix(c(e11, e21, e12, e
22), nrow = 2))

cat('\n', 'Chi-Squared Value is:', chi_squared)
```

```
##
##  Chi-Squared Value is: 0.6703448
```

```
p_value = pchisq(chi_squared, df = 1, lower.tail=F)

cat('\n', 'p-value is:', p_value)
```

```
##
##  p-value is: 0.4129314
```

(5)(d) (4 points) Build a user-defined function, using your code for (5)(c).We want to pass our (5)(a) table to our function and have it return the chi-squared statistic and p-value. You're provided with the 'shell' of a function and will need to add code to calculate the expected values, the chi-squared statistic, the p-value and return (i.e. output) the chi-squared and p-value.

```
chisq_function <- function(x) {
  # Code for calculating the expected values
  mar_tbl <- addmargins(x)
  e11 <- mar_tbl[3, 1] * mar_tbl[1, 3] / mar_tbl[3, 3]
  e12 <- mar_tbl[3, 2] * mar_tbl[1, 3] / mar_tbl[3, 3]
  e21 <- mar_tbl[3, 1] * mar_tbl[2, 3] / mar_tbl[3, 3]
  e22 <- mar_tbl[3, 2] * mar_tbl[2, 3] / mar_tbl[3, 3]

  # Code for calculating the chi-squared
  chi_squared <- sum((x - matrix(c(e11, e21, e12, e22), nrow = 2))^2 / matrix(c(e11, e21, e12, e
22), nrow = 2))

  # Code for calculating the degrees of freedom and p-value
  df = (length(row.names(x)) - 1) * (length(colnames(x)) - 1)
  p_value = pchisq(chi_squared, df = 1, lower.tail=F)


  # Code to ouput the chi-squared, degrees of freedom and p-value
  return(list(chi_squared = chi_squared, df = df, p_value = p_value))

}

chisq_function(tbl)
```

```
## $chi_squared
## [1] 0.6703448
##
## $df
## [1] 1
##
## $p_value
## [1] 0.4129314
```

You do not need to do anything with the below. It is provided only for demonstration purposes. In (5)(d), we know the size of the table - 2 x 2 - and write a function to match. Often, though, we'll want to write functions that are flexible in some way.

```
# Below is a function that should return the same values as chisq.test() and your
# function from (5)(d). Here, though, the function loops over the rows and columns
# to calculate the expected values. Ideally, this function would work for any sized
# table.

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:dim(t)[1]) {
       for (j in 1:dim(t)[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) * (ncol(t) - 1)
  pv <- pchisq(chi, df = xdf, lower.tail = FALSE)
  return(list("chi-squared" = chi, "degrees_of_freedom" = xdf, "p-value" = pv))
  }

chisqfun(tbl)
```

```
## $`chi-squared`
## [1] 0.6703448
##
## $degrees_of_freedom
## [1] 1
##
## $`p-value`
## [1] 0.4129314
```