JOSHUA NGUYEN

R-LIFE: BEGINNER'S GUIDE FOR STATISTICAL COMPUTATION AND GRAPHICS

JOSHUA NGUYEN

Contents

Introduction

This manual contains and provides the fundamental basis of R and RStudio in a rudimentary vernacular for the common layperson. It also archives some of Joshua Nguyen's projects in R and statistical computation.

What is R?

Introduction

R is a standardized programming language that is used for statistical computation and graphics. It is a free software and widely used among statisticians and data miners for analysis.

RStudio

RSTUDIO is an Integrated Development Environment (IDE) [1] for R. As a graphical user interface or GUI, RStudio will essential simplify the tasks commonly done in R. Written in C++ and Java, RStudio uses a Qt framework for its graphical user interface.

[1] An IDE or integrated development environment software application that offers comprehensive facilities and often includes of a source code editor, automation tools and a debugger.

R versus SQL versus Python?

SQL is a standard language for retrieving and manipulating structured databases. On the contrary, MySQL is a relational database management system, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.

Basics of R In RStudio

Objective

This section will introduce the basic syntax and tools of R in RStudio. Our goal is to:

1. Learn how to calculate with R
2. Use R functions
3. Create and manipulate objects
4. Index values

Operators

`+, -, /,*,` Basic arithmetic operators including addition, subtraction, division and multiplication respectively

`<-` Assigns value on the right hand side to the object on the left hand side

`#` Initiates comment line

```
# Addition and subtraction
> 1+1-2
[1] 0
# Multiplication and Division
> 25*4/2
[1] 50
# Storing values to an object
> x <- 16
> x
[1] 16
```

Functions

`c()` Concatenates two or more objects together

`length()` Returns number of elements in a specific vector

`mean()` Returns mean value or average of a specific vector

`log()` Returns the `natural` log of a value

`sort()` Sorts values of a vector in ascending order

`sum()` Returns the summation of two or more numerical values [2]

`sqrt()` Returns the square root of a numerical value

[2] the `sum()` function can also return the count of logical TRUE values in a vector

```
> x <- 16
> x
[1] 16
# Concatenating objects to form a vector
> y = c(x, 1,2)
> y
[1] 16 1 2
# Finding the size of an object
> length(y)
[1] 3
# Finding the mean value of a vector
> mean(y)
[1] 6.333333
# Finding the natural log value of an object
> log(x)
[1] 2.772589
# Sorting the values of a vector in increasing order
> sort(y)
[1] 1 2 16
# Finding the sum of a vector
> sum(y)
[1] 19
# Finding the square root of a value
> sqrt(x)
[1] 4
```

Expressions

`[]` Indexes a vector with numbers or logical expressions

`>` / `<` Greater than / less than

`>= / <=` Greater than or equals to / less than or equals to

`==` Equals to

`!=` Not equal to

`&` And

`|` Or

Indexing is very useful in finding a specific value of a vector. Recall that an index is an element's position in a vector or array, similar to the ID of a row in an excel spreadsheet. We call values inside the square brackets, [], indices.

```
> vector_a <- c(1,2,3,4,5,6,7,8,9,10)
> vector_a[2]
[1] 2
```

In R, it can be extremely useful to apply conditions to our indexes.

```
# Greater than
> vector_a > 5
[1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
↪  TRUE  TRUE
# Returning the true values
> vector_a[vector_a > 5]
[1] 6 7 8 9 10
# Returning values between 4 and 6
> vector_a[vector_a > 4 & vector_a < 6]
[1] 5
```

Recall earlier that the `sum()` function can return the count or "sum" of logical True values in a particular vector. That is, it will count $TRUE$ as a 1 and $FALSE$ as a 0, and then return the sum.

```
> sum (vector_a > 3 & vector < 6)
[1] 2
```

Summary

In this chapter, we introduced how to use basic operators such as arithmetic or assigning an object a value in R using Rstudio. We then learned how to perform basic functions on our objects and returned values of particular interests like the square root of 16. Lastly, indexing our vectors allowed us to find and select very specific values.

Within The Data

Objective

This section will introduce basic R tools in RStudio with an actual database. Our goal is to create objects, index vectors and apply functions in R to our dataset in order to gain a main summary about it. This is called exploratory data analysis. We will learn:

1. Reading databases and examining its contents
2. Creating and manipulating objects with functions
3. Indexing a dataframe

New Functions

`setwd()` Selects a new working directory

`getwd()` Returns current working directory

`read.csv()` Imports .csv file into R's enviornment as a dataframe

`names()` Returns variable names of a dataframe

`head()` Returns the 6 rows of a dataframe

`min()` Returns the minimum value of a vector

`max()` Returns the maximum value of a vector

`levels()` Returns all unique values of a vector

`table()` Creates a frequency table or a a contingency table depending on the number of variables.

Importing the Dataset

This chapter will use the **Heart Failure Prediction Dataset**. The `csv` file can be obtained at https://www.kaggle.com/fedesoriano/heart-failure-prediction. Download the file to follow along.

```
# First set the path
> setwd("C:/Users/MyName/Documents/Data Analyst/RLife")
# Import the dataset
> my_data <- read.csv("heart.csv")
> my_data
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
1	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
2	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
3	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
4	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
5	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
6	39	M	NAP	120	339	0	Normal	170	N	0.0	Up	0
7	45	F	ATA	130	237	0	Normal	170	N	0.0	Up	0
8	54	M	ATA	110	208	0	Normal	142	N	0.0	Up	0
9	37	M	ASY	140	207	0	Normal	130	Y	1.5	Flat	1
10	48	F	ATA	120	284	0	Normal	120	N	0.0	Up	0
11	37	F	NAP	130	211	0	Normal	142	N	0.0	Up	0
12	58	M	ATA	136	164	0	ST	99	Y	2.0	Flat	1
13	39	M	ATA	120	204	0	Normal	145	N	0.0	Up	0
14	49	M	ASY	140	234	0	Normal	140	Y	1.0	Flat	1
15	42	F	NAP	115	211	0	ST	137	N	0.0	Up	0
16	54	F	ATA	120	273	0	Normal	150	N	1.5	Flat	0
17	38	M	ASY	110	196	0	Normal	166	N	0.0	Flat	1
18	43	F	ATA	120	201	0	Normal	165	N	0.0	Up	0
19	60	M	ASY	100	248	0	Normal	125	N	1.0	Flat	1
20	36	M	ATA	120	267	0	Normal	160	N	3.0	Flat	1
21	43	F	TA	100	223	0	Normal	142	N	0.0	Up	0
22	44	M	ATA	120	184	0	Normal	142	N	1.0	Flat	0
23	49	F	ATA	124	201	0	Normal	164	N	0.0	Up	0
24	44	M	ATA	150	288	0	Normal	150	Y	3.0	Flat	1
25	40	M	NAP	130	215	0	Normal	138	N	0.0	Up	0
26	36	M	NAP	130	209	0	Normal	178	N	0.0	Up	0
27	53	M	ASY	124	260	0	ST	112	Y	3.0	Flat	0
28	52	M	ATA	120	284	0	Normal	118	N	0.0	Up	0
29	53	F	ATA	113	468	0	Normal	127	N	0.0	Up	0
30	51	M	ATA	125	188	0	Normal	145	N	0.0	Up	0
31	53	M	NAP	145	518	0	Normal	130	N	0.0	Flat	1
32	56	M	NAP	130	167	0	Normal	114	N	0.0	Up	0
33	54	M	ASY	125	224	0	Normal	122	N	2.0	Flat	1
34	41	M	ASY	130	172	0	ST	130	N	2.0	Flat	1
35	43	F	ATA	150	186	0	Normal	154	N	0.0	Up	0
36	32	M	ATA	125	254	0	Normal	155	N	0.0	Up	0
37	65	M	ASY	140	306	1	Normal	87	Y	1.5	Flat	1
38	41	F	ATA	110	250	0	ST	142	N	0.0	Up	0
39	48	F	ATA	120	177	1	ST	148	N	0.0	Up	0
40	48	F	ASY	150	227	0	Normal	130	Y	1.0	Flat	0
41	54	F	ATA	150	230	0	Normal	130	N	0.0	Up	0
42	54	F	NAP	130	294	0	ST	100	Y	0.0	Flat	1
43	35	M	ATA	150	264	0	Normal	168	N	0.0	Up	0
44	52	M	NAP	140	259	0	ST	170	N	0.0	Up	0
45	43	M	ASY	120	175	0	Normal	120	Y	1.0	Flat	1
46	59	M	NAP	130	318	0	Normal	120	Y	1.0	Flat	0
47	37	M	ASY	120	223	0	Normal	168	N	0.0	Up	0
48	50	M	ATA	140	216	0	Normal	170	N	0.0	Up	0
49	36	M	NAP	112	340	0	Normal	184	N	1.0	Flat	0
50	41	M	ASY	110	289	0	Normal	170	N	0.0	Flat	1
51	50	M	ASY	130	233	0	Normal	121	Y	2.0	Flat	1
52	47	F	ASY	120	205	0	Normal	98	Y	2.0	Flat	1
53	45	M	ATA	140	224	1	Normal	122	N	0.0	Up	0
54	41	F	ATA	130	245	0	Normal	150	N	0.0	Up	0
55	52	F	ASY	130	180	0	Normal	140	Y	1.5	Flat	0
56	51	F	ATA	160	194	0	Normal	170	N	0.0	Up	0
57	31	M	ASY	120	270	0	Normal	153	Y	1.5	Flat	1
58	58	M	NAP	130	213	0	ST	140	N	0.0	Flat	1
59	54	M	ASY	150	365	0	ST	134	N	1.0	Up	0
60	52	M	ASY	112	342	0	ST	96	Y	1.0	Flat	1

```
# Getting variable names
> names(my_data)
[1] "Age"            "Sex"            "ChestPainType"
↪   "RestingBP"      "Cholesterol"    "FastingBS"
↪   "RestingECG"     "MaxHR"          "ExerciseAngina"
↪   "Oldpeak"        "ST_Slope"       "HeartDisease"
```

To obtain a specific variable from our dataframe, we use the $, dollar sign, symbol.

```
# Lets find the min and max age of our dataframe
> min(my_data$Age)
[1] 28
> max(my_data$Age)
[1] 77
```

Lets use the `levels()` function to select all unique values of our "Sex" variable. We do this because sex is categorical and not numerical. Lets find the range.

```
> levels(my_data$Sex)
[1] "M"             "F"
# Using the table function, we can find out the number
↪  of males and females
> table(my_data$Sex)
  F   M
193 725
```

A frequency table is based on one variable. To select two variables, we use a contingency table. [3]

[3] Remember, that contingency tables will require two categorical variables and not numerical.

```
# Contingency table for resting ECG and ST Slop
> table(my_data$ST_Slope, my_data$RestingECG)

      LVH Normal  ST
Down   17     31  15
Flat   97    266  97
Up     74    255  66
```

Notice that we did not choose sex or another simple variable. This is because the two variables have to be the same size. In other words, the count of unique values in each variable have to be equal.

Dataframe Indexing

We can index our data or return specific variables based on a conditional by using **logical indexing**. Recall, dataframes are two-dimensional having rows and columns. Ergo, dataframes require two *indices*. The **rows()** corresponds to the first index and **columns()** corresponds to the second index.

```
# Return Age of row 1
> my_data[1,1]
[1] 40
```

It is useful to create a new dataframe based on a certain criteria or condition. For example, in the next code, we will create a subset of our dataframe such that it only includes males.

```
# Creating new dataframe to include only males but all
↪  columns
> males <- my_data[my_data$Sex== "M",]
> males$Age
  [1] 40 37 54 39 54 37 58 39 49 38 60 36 44 44 40 36
  ↪  53 52 51 53 56 54 41 32 65 35 52 43 59 37 50 36
  ↪  41 50 45 31 58 54 52 49 45 46 32 52 44 57 44 52
  ↪  55 46 32 52 49 55 54 63 52 56 66 65 43 55 39 48
  ↪  58 43 39 56 41 65 51 40 ...
```

Indexing Dataframes with a Variable

Sometimes we can be even more specific with our indexing and select certain values based on a particular variable.

```
# Indexing with a variable
> my_data$Age[my_data$Sex== "M"]
  [1] 40 37 54 39 54 37 58 39 49 38 60 36 44 44 40 36
  ↪  53 52 51 53 56 54 41 32 65 35 52 43 59 37 50 36
  ↪  41 50 45 31 58 54 52 49 45 46 32 52 44 57 44 52
  ↪  55 46 32 52 49 55 54 63 52 56 66 65 43 55 39 48
  ↪  58 43 39 56 41 65 51 40 ...
```

Lets create a new object containing the ages of all males within our dataset and find the mean or average of it.

```
# Return mean of all males' ages
> male_age <- my_data$Age[my_data$Sex== "M"]
> mean(male_age)
[1] 53.78207
```

Summary

You have now learned how to set a directory path for R and used that path to import a `csv` file to begin a rudimentary form of exploratory

data analysis. Next, with our imported dataframe, you learned to index it based on certain variables and conditions. We utilized new the mean function to return the average age of all males in our dataset.

Exploratory Data Analysis

Objective

This section will introduce basics of EDA such as describing and displaying our dataframe. We will learn:

1. Describing and displaying categorical variables
2. Describing and displaying the distributions of numerical variables
3. Describing and displaying the relationship between categorical and numerical variables
4. Describe the relationship between two numerical variables

New Functions

`barplot()` Returns a bar chart based on the frequency table of a categorical variable

`sd()` Returns standard deviation of a numerical vector

`median()` Returns median of a numerical vector

`fivenum()` Returns the five-number[4] summary of a numerical vector

`hist()` Returns histogram of a numerical vector

`boxplot()` Returns boxplot of a numerical vector

`plot()` Returns scatterplot of two numerical variables

[4] min, first quartile, median, third quartile, max

Importing the Dataset

This chapter will use the **Caffeine Content of Drinks** dataset. The `csv` file can be obtained at https://www.kaggle.com/heitornunes/caffeine-content-of-drinks.

```
# Import the dataset
> my_data <- read.csv("caffeine.csv", StringAsFactors = T)
> head(my_data)
                      drink Volume..ml. Calories Caffeine..mg.   type
1                Costa Coffee    256.9937        0           277 Coffee
2 Coffee Friend Brewed Coffee    250.1918        0           145 Coffee
3          Hell Energy Coffee    250.1918      150           100 Coffee
4          Killer Coffee (AU)    250.1918        0           430 Coffee
5                Nescafe Gold    250.1918        0            66 Coffee
6            Espresso Monster    248.4174      170           160 Coffee
```

Analyzing the Data

In this chapter, we will answer the following questions about our dataframe:

1. **How many types of drinks currently exist in our database?**
2. **What is the overall distribution of calories (and caffeine) and does it differ by drink type?**
3. **Are volume and calories related?**

Investing the Data

Notice that there are five variables in our dataset. Two variables, `drink`, `type`, are categorical, and the other three variables, `volume`, `calories` and `caffeine` are numerical. Using the `levels` function, we can find the number of unique elements in our variables. Lets found out how many types of drinks currently exist in our dataframe.

```
# Number of types of drinks in our dataset
> levels(my_data$type)
[1] "Coffee" "Energy Drinks" "Energy Shots" "Soft Drinks"
↪  "Tea" "Water"
```

Therefore, from above, there exists six different drink types in our dataset. Lets found out how many of each type do we have.

```
# Table of types of drinks
> table(my_data$type)
       Coffee Energy Drinks  Energy Shots   Soft Drinks
       173           219           36            90
       Tea         Water
       66            26
```

Lets visualize these counts above using a barplot which plots the frequency or count of the number of types of drinks.

barplot()

xlab=" Specifies label for the x-axis

ylab=" Sepcifies label for the y-axis

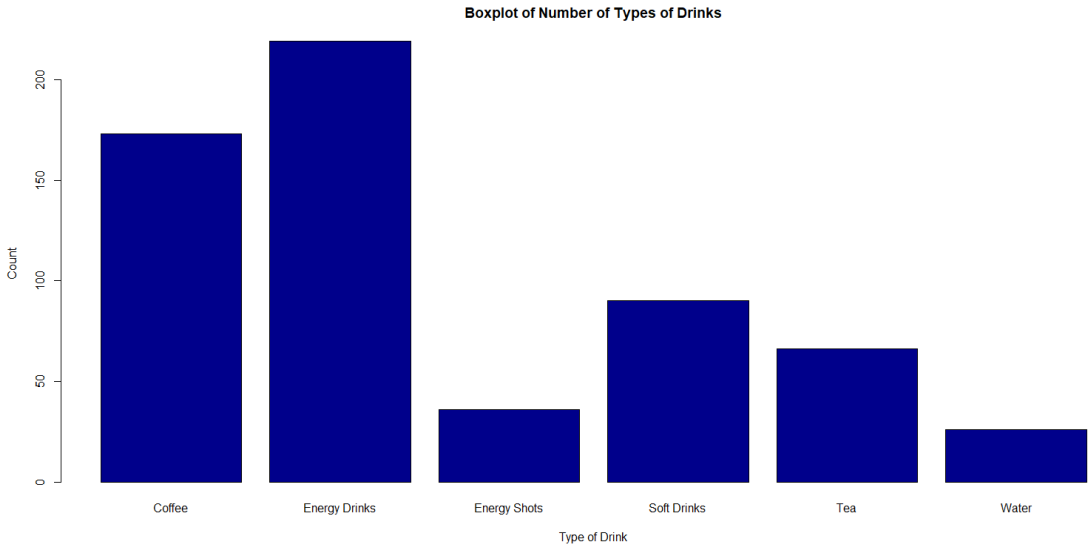
main=" Specifies title

xlim=c(min, max) Sets the minimum and maximum value of the x-axis

ylim=c(min, max) Sets the minimum and maximum value of the y-axis

col= " Sets color of plot

```
# Using Barplot to graph frequency of types of drinks
> drink_type <- table(my_data$type)
> barplot(drink_type, xlab = 'Type of Drink', ylab =
↪  'Count', main = 'Boxplot of Number of Types of
↪  Drinks', col = 'dark blue')
```



Describing Numerical Variables

In order to describe the caloric or caffeine content of our drinks, we need to calculate measures of average, standard deviation and quartile ranges. Lets summarize the calorie contents.

```
# Mean
> mean(my_data$Calories)
[1] 75.52787
# Standard Deviation
> sd(my_data$Calories)
[1] 94.79992
> median(my_data$Calories)
[1] 25
> fivenum(my_data$Calories)
[1]   0   0  25 140 830
```

The mean Calorie of the drinks in our dataset is 75.5 calories with a standard deviation of 94.8 calories. The median value for calorie content is 25 calories. The IQR[5] is 140 calories.

To visualize numerical variables, we use the `hist()` function to create a histogram plot.

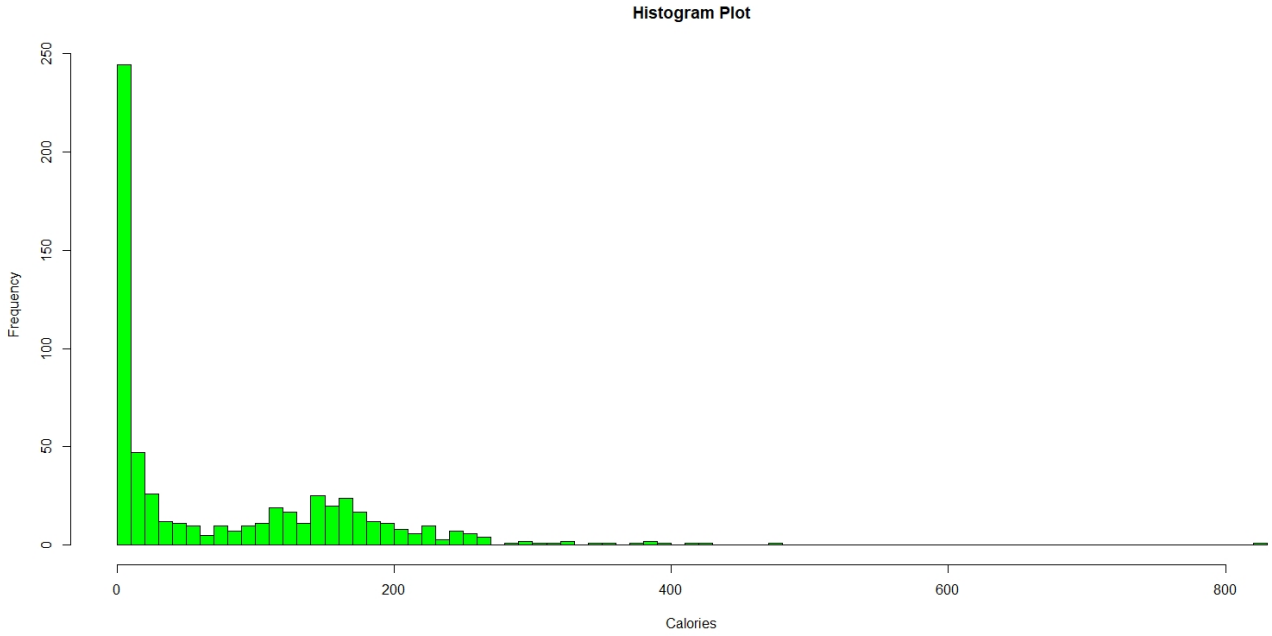
[5] Interquartile Range is a measure of statistical dispersion, which is the spread of the data. The IQR may also be called the midspread, middle 50 percent, or H-spread. It is defined as the difference between the 75th and 25th percentiles of the data

Hist()

R=F Places the values that fall on the bin borders to the right side with default being `left`

breaks = x Contains an approximate x number of bins with default being 10

```
# Histogram Plot
> hist(my_data$Calories, xlab = 'Calories', main =
↪  'Histogram Plot', right = F, breaks  = 100, col =
↪  'green')
```



Here, we see that the calorie content for the drinks of our dataset is heavily skewed to the right. Interestingly, we can also see a binomial distribution!

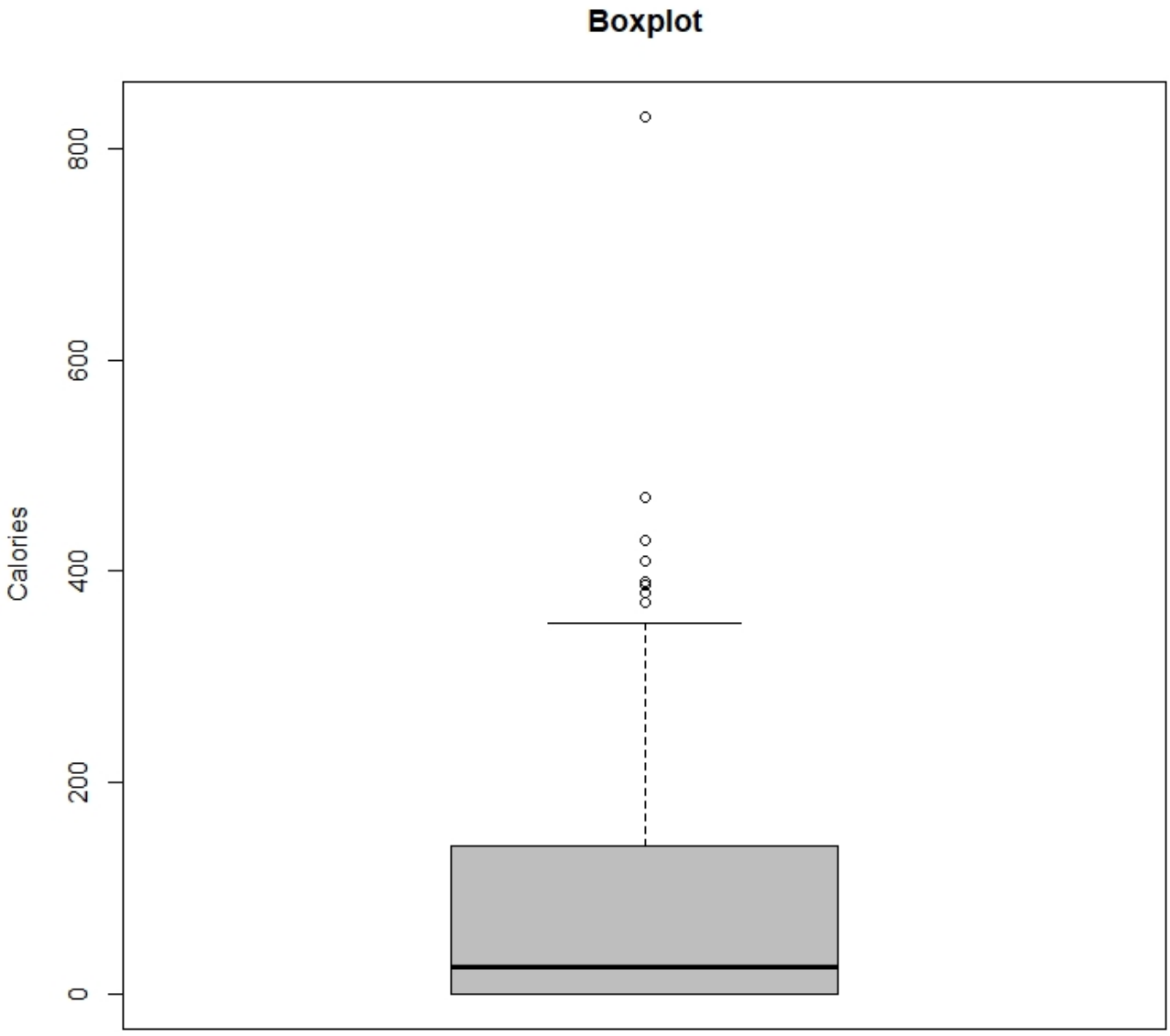
We can also create a boxplot our our data set to get another display of our Calorie variable.

```
# Boxplot
> boxplot(my_data$Calories, ylab = 'Calories', main =
↪  'Boxplot', col = 'grey' )
```

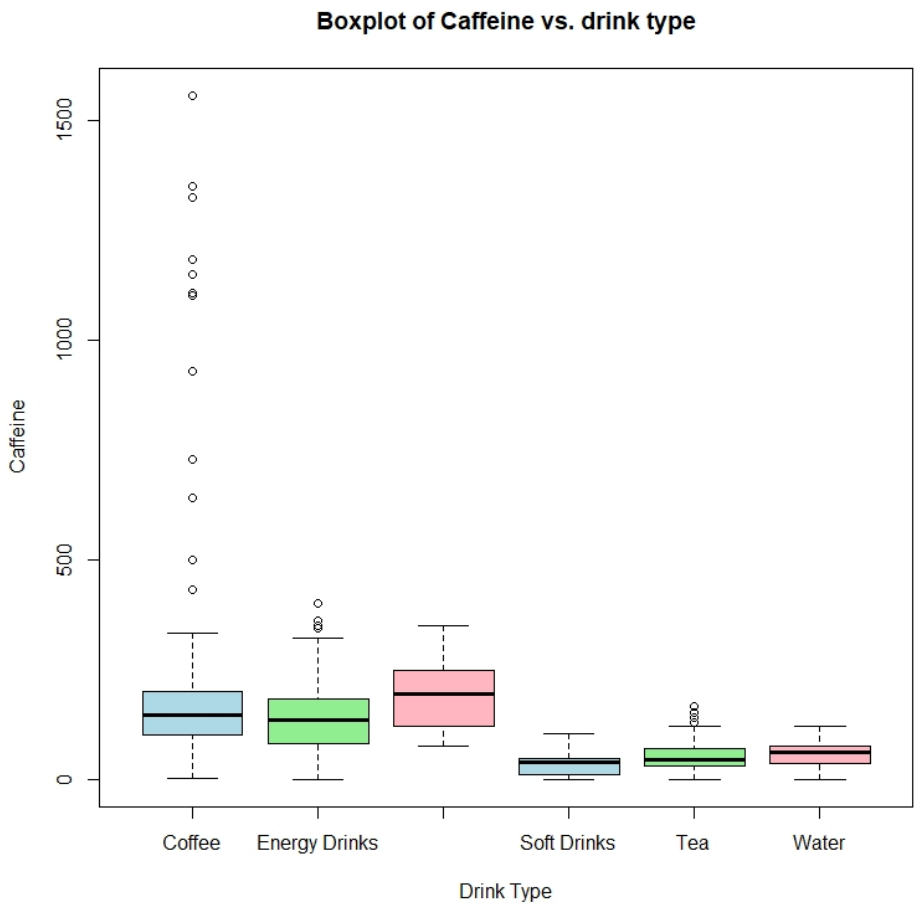
Creating a boxplot can also show the distribution of a numerical variable. In our boxplot we see that the Calorie content for our drinks are not very symmetric. Indubitably, recall from our histogram plot that the data was heavily skewed to the right. This is reflected in our boxplot. Additionally, our boxplot also helps us see that there exists many outliers in our dataset.

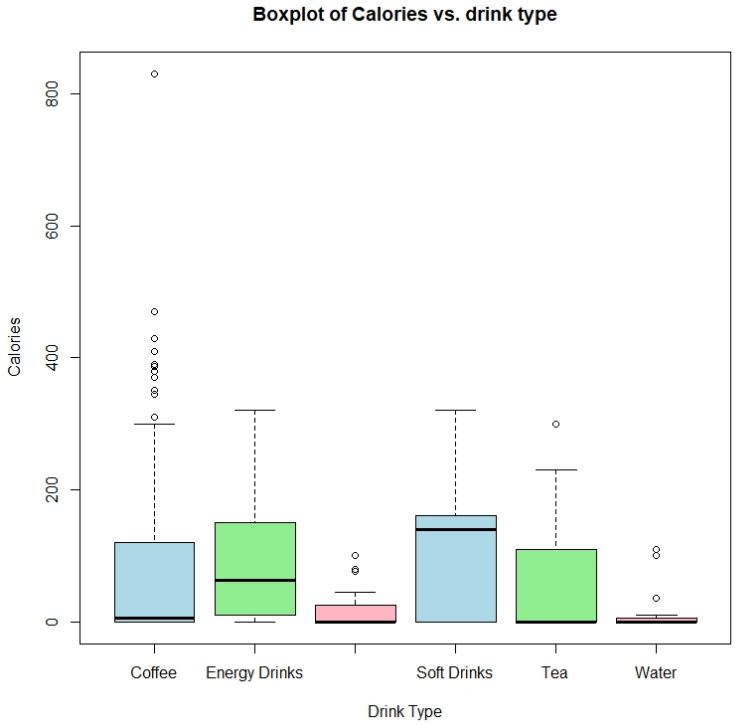
Finding the Relationship Between Categorical and Numerical Variables

Next, we can amalgamate our comparison to both categorical and numerical variables. For example, using the `boxplot()` function, we can compare the distribution of caffeine and calories per drink type. Within our `boxplot()` function, we will use the tilde symbol, ~, to display calories and caffeine as a function of drink type.


Boxplot
Calories
0
200
400
600
800

```
# Calories versus drink type
> boxplot(my_data$Calories ~ my_data$type, xlab = 'Drink
↪  Type', ylab = 'Calories', main = 'Boxplot of Calories
↪  vs. drink type', col = c("light blue", "light green",
↪  "light pink") )
# Caffeine versus drink type
> boxplot(my_data$Caffeine..mg. ~ my_data$type, xlab =
↪  'Drink Type', ylab = 'Calories', main = 'Boxplot of
↪  Caffeine vs. drink type', col = c("light blue", "light
↪  green", "light pink") )
```





Notice that there does not exist a strong distribution of Calorie content for all six drink types. However, on the contrary, see that the distribution of caffeine content is relatively similar for coffee, energy drinks and energy shots but differ from soft drinks, tea and water. In addition, we see that the distribution of caffeine for soft drinks, tea and water are also similar. Therefore, lets use **logical indexing** to create subsets of our dataset. Because the distribution of caffeine content is relatively similar, we can calculate the mean and standard deviation of each drink type.

```
# Descriptive statistics for the Coffee drink type
> coffee <- my_data[my_data$type == "Coffee",]
> mean(coffee$Caffeine..mg.)
[1] 200.5896
> sd(coffee$Caffeine..mg.)
[1] 248.2222
# Descriptive statistics for the Energy Drinks drink type
> ED <- my_data[my_data$type == "Energy Drinks",]
> mean(ED$Caffeine..mg.)
[1] 147.8676
> sd(ED$Caffeine..mg.)
[1] 76.73453
```

```
# Descriptive statistics for the Energy Shots drink type
> ES <- my_data[my_data$type == "Energy Shots",]
> mean(ES$Caffeine..mg.)
[1] 193.4167
> sd(ES$Caffeine..mg.)
[1] 79.53593
# Descriptive statistics for the Soft Drinks drink type
> soft_drinks <- my_data[my_data$type == "Soft Drinks",]
> mean(soft_drinks$Caffeine..mg.)
[1] 33.67778
> sd(soft_drinks$Caffeine..mg.)
[1] 24.91596
# Descriptive statistics for Tea
> tea <- my_data[my_data$type == "Tea",]
> mean(tea$Caffeine..mg.)
[1] 55.86364
> sd(tea$Caffeine..mg.)
[1] 39.33364
# Descriptive statistics for Water
> water <- my_data[my_data$type == "Water",]
> mean(water$Caffeine..mg.)
[1] 53.73077
> sd(water$Caffeine..mg.)
[1] 34.0606
```

As we might have suspected, among the six drink types, coffee has the highest mean average of caffeine content. It also has the highest standard deviation. On the other hand, water has the lowest mean average of caffeine content. However, notice that soft drinks and not water possesses the lowest standard deviation of caffeine content.

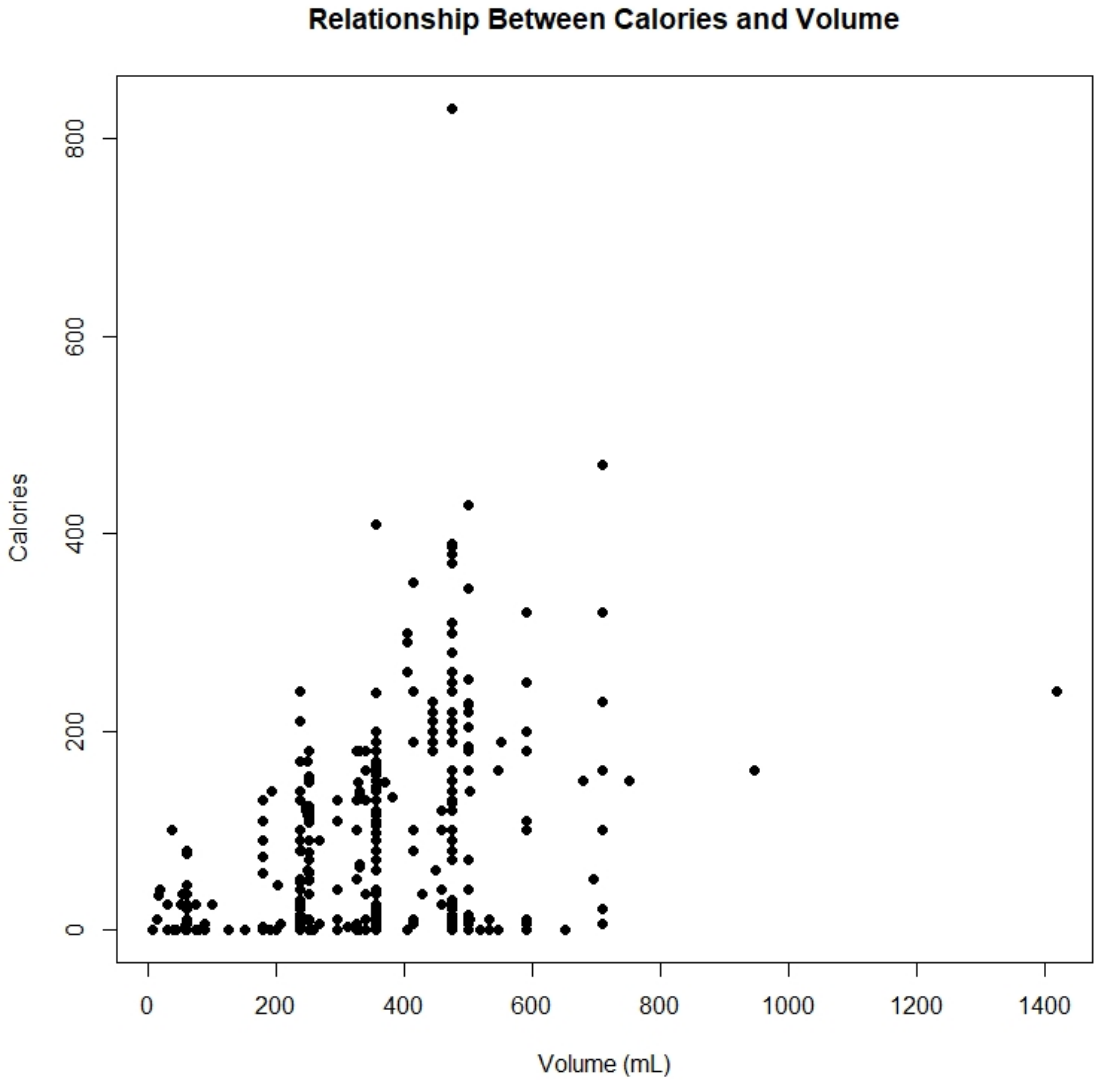
Finding the Relationship Between Two Numerical Variables

In order to visualize the relationship bewteen two numerical variables such as volume and Calorie content, we will use the `plot()` function to create a scatter plot.

```
# Create Scatterplot
> plot(my_data$Volume..ml., my_data$Calories, xlab =
↪  'Volume (mL)', ylab = 'Calories', main = 'Relationship
↪  Between Calories and Volume', pch = 16)
```

The `pch =16` option in our code above tells R to fill in the dots. The default will be unfilled.

Interestingly, there does not seem to be a strong relationship between drink volume and its calorie content. Later in this manual, we will learn further descriptive statistics that can be used to quantitatively measure the relationship of the two variables.



Inferential Statistics

Objective

Before beginning some inferential statistics, we will need to collect **random** samples from our population of interest. Using the smaller sample or subset, we can run statistics on it to judiciously infer an estimate about the larger sample. Moreover, we are interested in illustrating the Central Limit Theorem[6] which states that if you have a population with some mean μ and standard deviation σ and take sufficiently large random samples from the population with replacement each time, then the distribution of the sample means will be approximately normally distributed. Overall, we will learn:

1. Calculating population parameters such as mean and standard deviation
2. Calculating expected mean and expected deviation of a sampling distribution based on the Central Limit Theorem
3. Build sampling distribution of means
4. Compare actual mean and actual standard deviation to their actual values

[6] For any population distribution, drawing large samples of size, n, then the distribution of sample means or sampling distribution, will:

1. have a normal distribution
2. have a sample mean μ_1 that is equal to the population mean μ
3. have a sample standard deviation σ_1 that is equal to the population standard error of $\sigma/\sqrt{n}$, where n is the sample size.

New Functions

`sample()` Returns a random sample of values from a vector

`for(i in 1:z){}` For loop function which performs an action specified on an object inside brackets {} with z iterations

`numeric()` Creates empty numeric vector, commonly used as a placeholder to store values using the `for(){}` function

Importing the Dataset

This chapter will use the **Underwater Surface Temperature Dataset** dataset. The `csv` file can be obtained at https://www.kaggle.com/shivamb/underwater-surface-temperature-dataset.

```
# Import the dataset
> my_data <- read.csv("underwater_temperature.csv",
↪  StringAsFactors = T)
> head(my_data[,c(1,7)])
  ID Temp...C.
1  1    24.448
2  2    24.448
3  3    24.545
4  4    24.448
5  5    24.351
6  6    24.351
```

Analyzing the Data

In this chapter, we will answer the following questions about our dataframe:

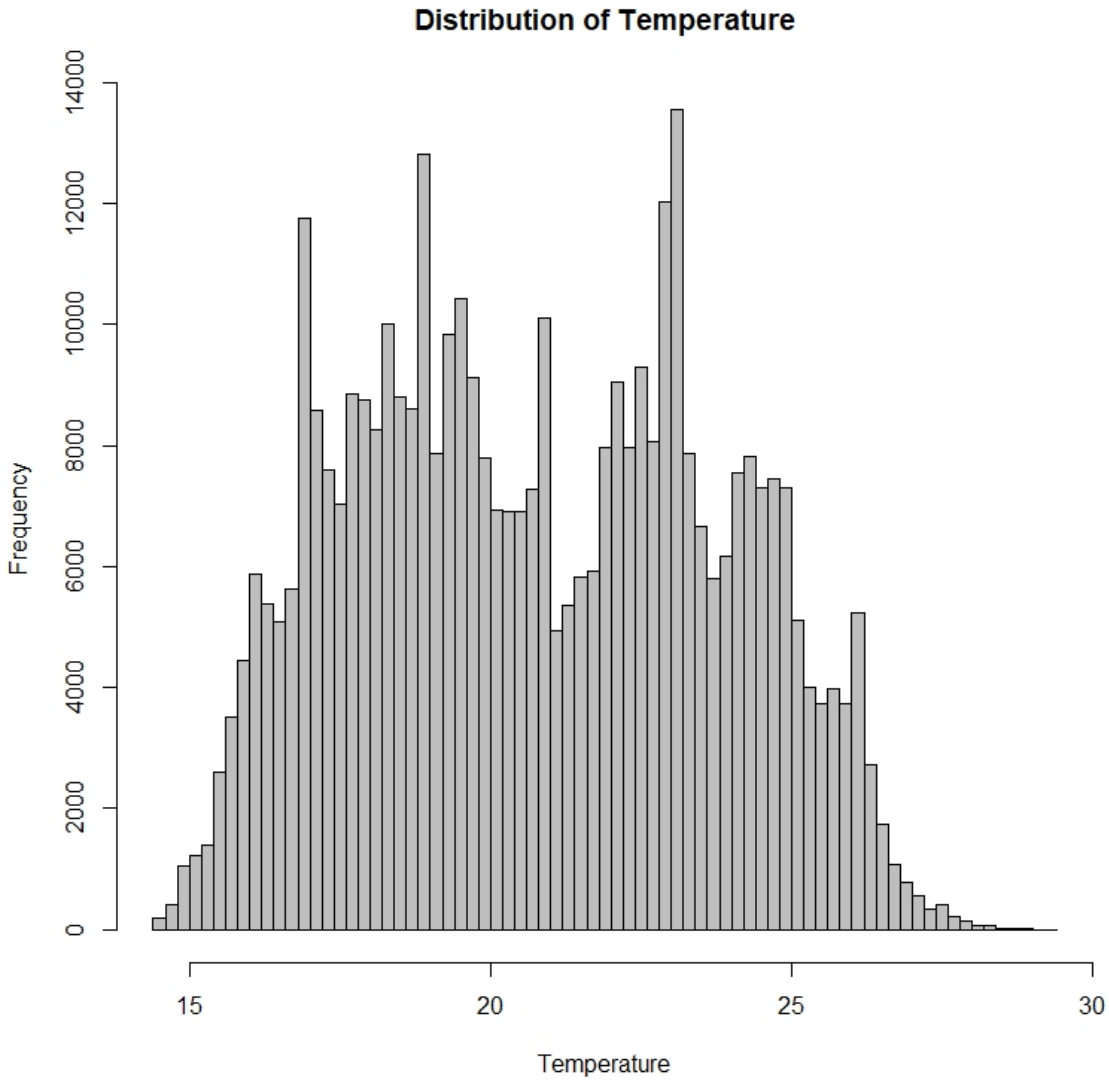
1. **How does changing the sample size, n, affect the distribution of the sample means?**

This dataset includes temperature variable around particular locations which is of high interest to us. Creating and viewing a histogram, we see that temperature is almost normalized, maybe even binomial?

```
# Histogram of Temperature
> hist(my_data$Temp, main = 'Distribution of
↪  Temperature', xlab = 'Temperature', breaks = 100,
↪  right = F, col = 'grey')
```

Calculating for μ and σ gives us:

```
# Removing Null values in the Temperature Columns
> my_data <-
↪  my_data[complete.cases(my_data$Temp...C.),]
# Finding the mean
> mean(my_data$Temp...C.)
[1] 20.75905
# Finding the standard deviation
> sd(my_data$Temp...C.)
[1] 2.980158
```



Therefore, we see that $\mu = 20.75905$ degrees Celcius and $\sigma = 2.980158$ degrees Celcius for the **entire population**. Next, lets create a sampling distribution and see if we can illustrate the Central Limit Theorem. We pick some sample size, n, say $n = 25$. Think about what happens to our sample mean and deviation as we increase the size of n. Calculating the standard deviation for $n = 25$ gives:

```
# standard deviation of n = 25
> sd(my_data$Temp...C./sqrt(25))
[1] 0.5960316
```

Ergo, we should expect for any subset or random sampling of our population, that $\mu = 20.75905$ and $\sigma_1 = 0.5960316$, by the Central

Limit Theorem.

Verifying Central Limit Theorem for $n = 25$

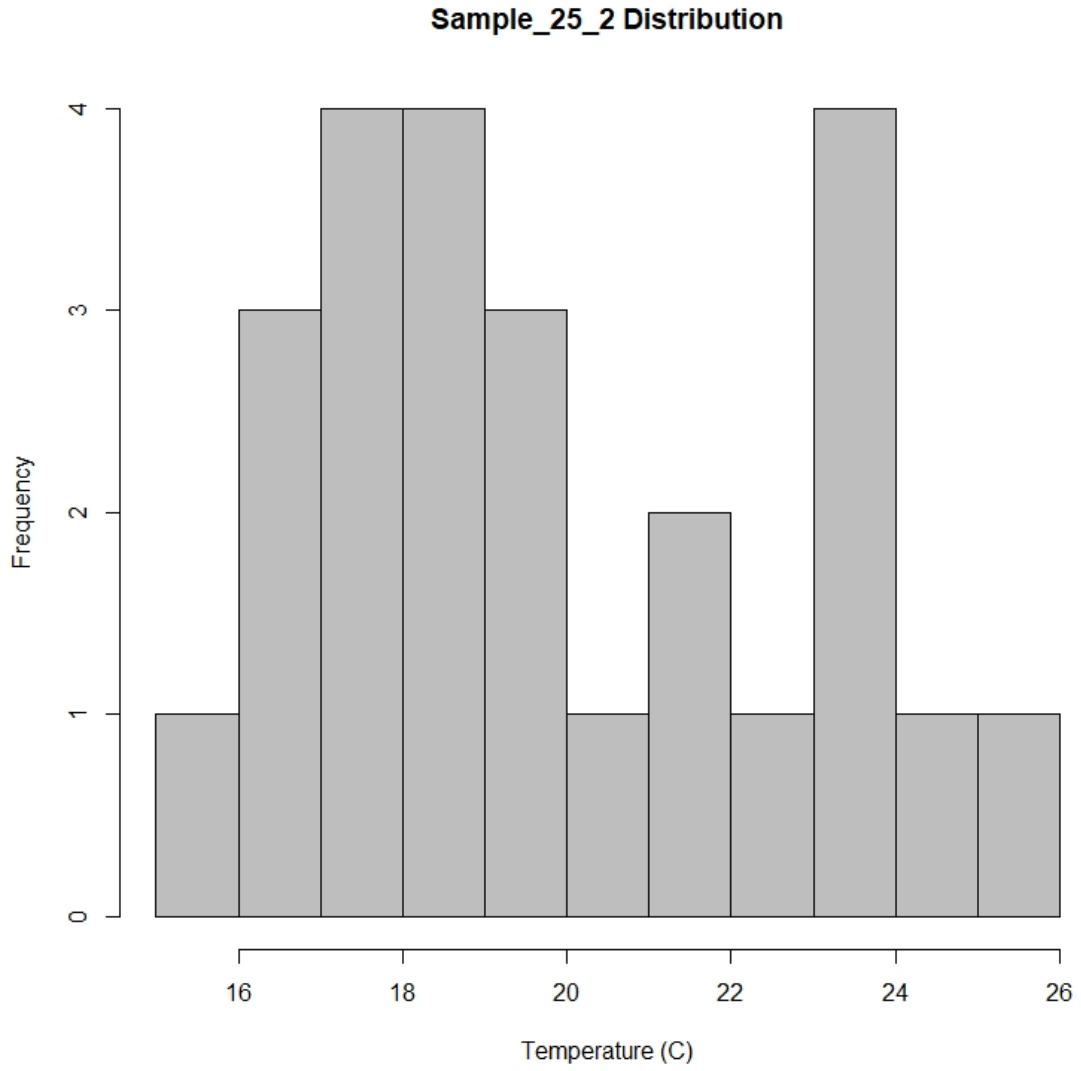
We can select a random sampling with the `sample()` function. Recall, this is a random sample, so each time you use the function, it will generate a new vector set or subset.

```
# Random sampling for the case n = 25
> sample_25 <- sample(my_data$Temp...C., size = 25)
> sample_25
[1] 25.319 17.665 18.426 17.855 22.812 17.760 25.610
↪  24.062 24.062 27.370 17.665 24.448 15.091 16.999
[15] 24.448 24.545 22.142 17.665 16.808 24.641 21.664
↪  25.319 17.950 20.710 16.237
> sample_25_2 <- sample(my_data$Temp...C., size = 25)
> sample_25_2
[1] 25.222 17.855 23.100 15.760 22.333 23.677 23.100
↪  18.331 19.187 21.378 19.282 17.855 17.760 18.711
[15] 21.951 16.523 18.901 24.545 16.332 16.523 23.484
↪  20.424 17.379 19.092 18.806
# Finding the mean values
> mean(sample_25)
[1] 21.09092
> mean(sample_25_2)
[1] 19.90044
# Finding the standard deviation
> sd(sample_25)
[1] 3.726381
> sd(sample_25_2)
[1] 2.811188
```

Notice from these two random samples of our population, that we were able to estimate the mean of the population with our subsets relatively accurately; however, both of our standard deviations were off by a factor of about 7.

Lets visualize the distribution of random samples with a histogram plot.

```
# Histogram for sample_25_2
> hist(sample_25_2, main = 'Sample_25_2 Distribution',
↪  xlab = 'Temperature (C)', breaks = 10, right = F,
↪  col = 'grey')
```

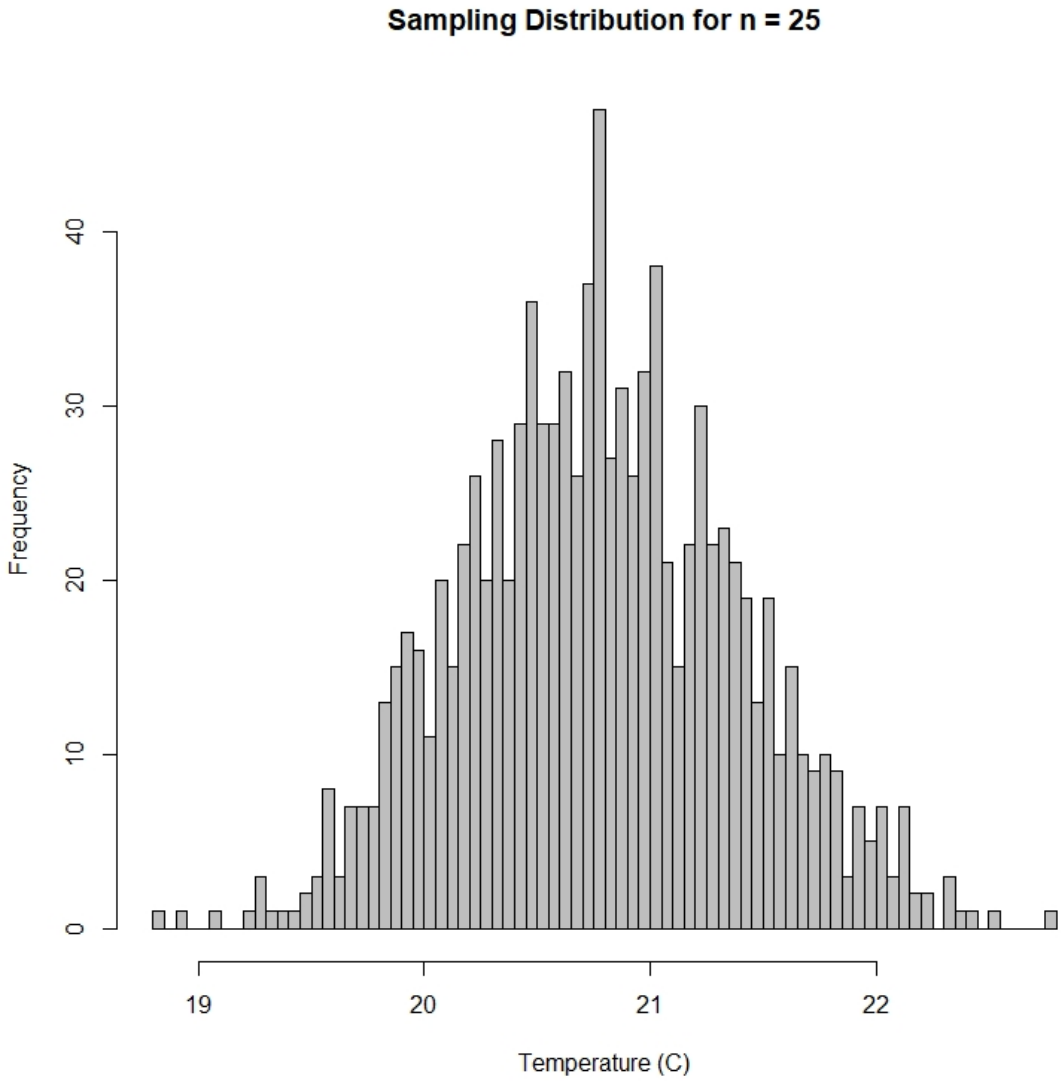


Notice, we get a similar bi-nominal distribution with our random sample instead of a normal distribution as the Central Limit Theorem may have predicted.

Simulation of Sampling Distribution

Lets increase the samples we will take from our population, say a x number of iterations. Our thought process here is to create an empty numeric vector with the `numeric()` function. Using a `for` loop, we will repeat our sampling process an x number of times. With each iteration, we will input the random sampling into our empty vector until it becomes size x. We will end with a vector with x random samples.

```
# Random Sampling
# Creating empty vector
> mymeans <- numeric(0)
# for loop
> for (i in 1:1000) {
+     x <- sample(my_data$Temp...C., size = 25)
+     mymeans <- c(mymeans, mean(x))
+ }
> hist(mymeans, main = 'Sampling Distribution for n =
↪  25', xlab = 'Temperature (C)', right = F, breaks =
↪  100, col = 'grey')
```



Despite our population not having a normal distribution, we see that iterating over random samples gives us a normal distribution as the Central Limit Theorem predicts. Try increasing the number of iterations or the size of our sampling, n, and see what happens to the graph? [7]

[7] Answer: The distribution gets more "normal"

Comparing the Sample Distribution

```
# Sampling mean
> mean(mymeans)
[1] 20.75871
# Sampling mean accuracy to true mean
> mean(mymeans) - mean(my_data$Temp...C.)
[1] -0.0003379119
# Sampling standard deviation
> sd(mymeans)
[1] 0.4326511
# Sampling standard deviation to true standard
↪  deviation
> sd(mymeans) - (sd(my_data$Temp...C.)/sqrt(5000))
[1] 0.3905053
```

As you can see, we have illustrated the Central Limit Theorem because the mean and standard deviation difference of random sampling and the true population is not significant!

One-Sample t Test

Objective

For our first statistical hypothesis testing, we will do a one-sample t-test on a dataset. **One-sample t test** will test if the mean of a population is equal to a claimed value. For example, I estimate the average salary (mean) in the United States (population) is $40,000$ (claimed value). I can use a one-sample t test to validate this hypothesis. Overall, in this chapter, we will learn:

1. Confirming the normality assumption for one-sample t testing
2. Investigating a transformation curves and how this improve the normality of a variable
3. Conducting one-sample t tests and interpreting their results.

New Functions

`qqnorm()` Plots a normal quantile-quantile (Q-Q) plot

`qqline()` Adds reference line to the quantile-quantile lineplot

`t.test()` Runs a t-test on a numeric vector

Importing the Dataset

This chapter will use the **In Hospital Mortality Prediction** dataset. The `csv` file can be obtained at https://www.kaggle.com/saurabhshahane/in-hospital-mortality-prediction.

```
# Import the dataset
> my_data <- read.csv("data01.csv", StringAsFactors =
↪  T)
```

```
> head(my_data[c(5,6,10,24,40)],)
  gendera      BMI diabetes      RBC   glucose
1       1 37.58818        1 2.960000 114.63636
2       2       NA        0 3.138000 147.50000
3       2 26.57263        0 2.620000 149.00000
4       2 83.26463        0 4.277500 128.25000
5       2 31.82484        0 3.286667 145.75000
6       1 24.26229        0 3.235000  98.33333
```

Analyzing the Data

In this chapter, we will answer the following questions about our dataframe:

1. **The average fasting blood glucose levels for healthy non-diabetic adults is between 90 to 100 mg/dL, say on average 95 mg/dL. Is the mean fasting blood glucose levels of diabetic patients at this hospital equal to 95 mg/dL at the time of their hospital stay?**[8]

2. **The average normal male's red blood cell count or RBC is between 4.7 to 6.1 million cells per microliter. Is the mean RBC of diabetic *male* patients at this hospital equal to and inside the range of 4.7 to 6.1 million cells per microliter at the time of their hospital stay?**

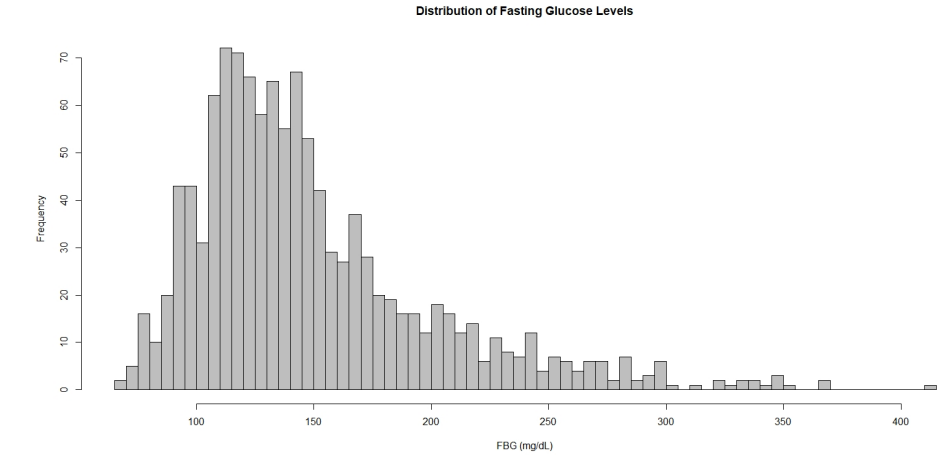
[8] We should expect a higher mean fasting glucose level for diabetic patients compared to non-diabetics, but I think it would be of interest to verify this statistically.

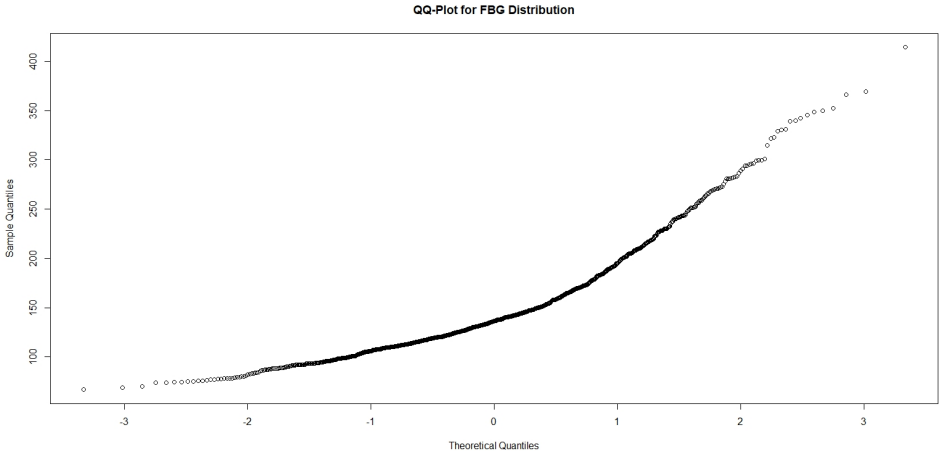
Assessing Normality

The one-sample t test assumes that the variable of interest is approximately normally distributed. Previously, we learned to use the `hist()` function to visually see a normal distribution. Lets use the `qqnorm()` function in this section.

Now, QQ-norms are scatterplots where the quantiles of a normal distribution are plotted as a function of the quantiles of a variable. A variable is normally distributed, if its points on the QQ-Norm plot fall rougly on the `qqline()` which is used as a reference for normality.

```
# Histogram
> hist(my_data$glucose, main = 'Distribution of
↪  Fasting Glucose Levels', xlab = 'FBG (mg/dL)',
↪  right = F, breaks = 100, col = 'grey')
# QQ-Norm
> qqnorm(my_data$glucose, main = 'QQ-Plot for FBG
↪  Distribution')
```





Data Transformations

Sometimes, data will not be normally distributed. For example, in our distribution curves and histogram above, we see that the distribution of Fasting Blood Glucose is right skewed. Thus, we cannot run a one-sample t test or any other test that require a normally distributed population. However, we can use functions that *transforms* data and converts them to a normally distributed. We call this a transformation. Common data transformation functions includes

1. `log()`
2. `sqrt()`
3. `1/x`
4. `exp(x)`

Conducting a One-Sample T Test

We want to verify if the mean fasting blood glucose levels for diabetics equal 96 mg/dL and if the RBC for male patients with diabetes is equal to 4.7 million cells per microliter.

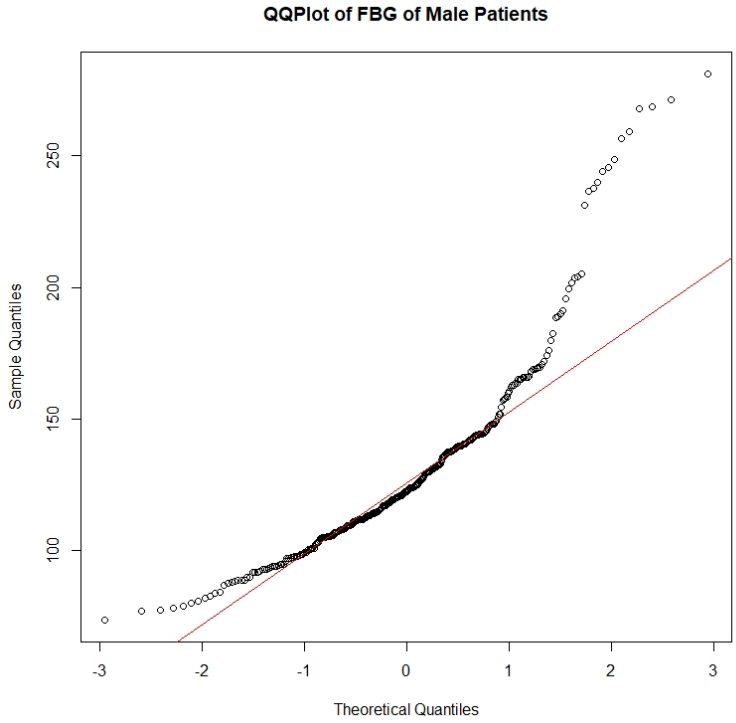
Before we do our sample t test, lets see if our sample subset meets the assumptions. Assuming the sample collection was random and independent, lets check if the population is normally distributed.

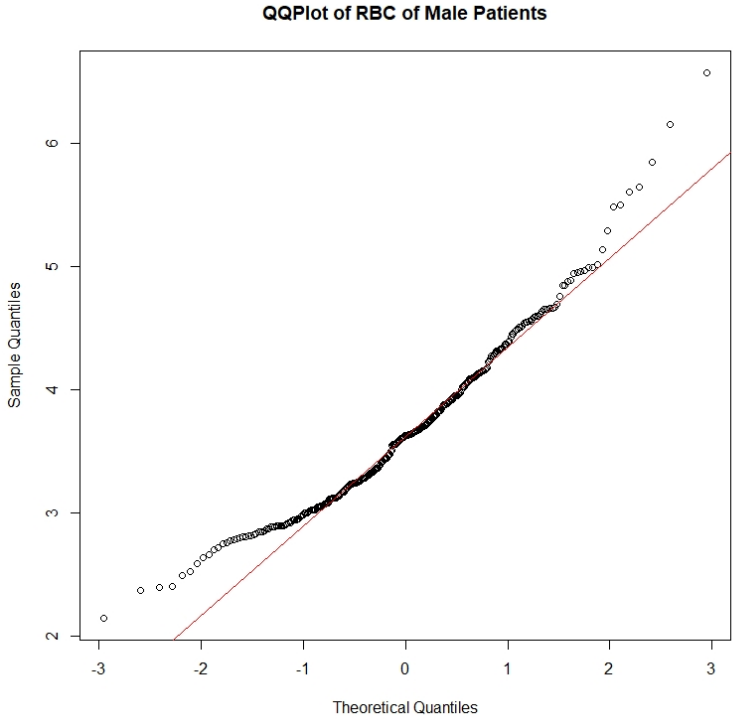
```
# Create FBG vector for male patients with diabetes
> males_glucose <-
↪  my_data[!(my_data[,5]==2|my_data[,10]==1),]
> males_glucose <- males_glucose$glucose
# Create RBC vector for male patients with diabetes
> males_RBC <-
↪  my_data[!(my_data[,5]==2|my_data[,10]==1),]
> males_RBC <- males_RBC$RBC
# Confirm the Normality
> qqnorm(males_glucose, main = 'QQPlot of FBG of Male
↪  Patients')
> qqline(males_glucose, col = 'red')
> qqnorm(males_RBC, main = 'QQPlot of RBC of Male
↪  Patients')
> qqline(males_RBC, col = 'red')
```

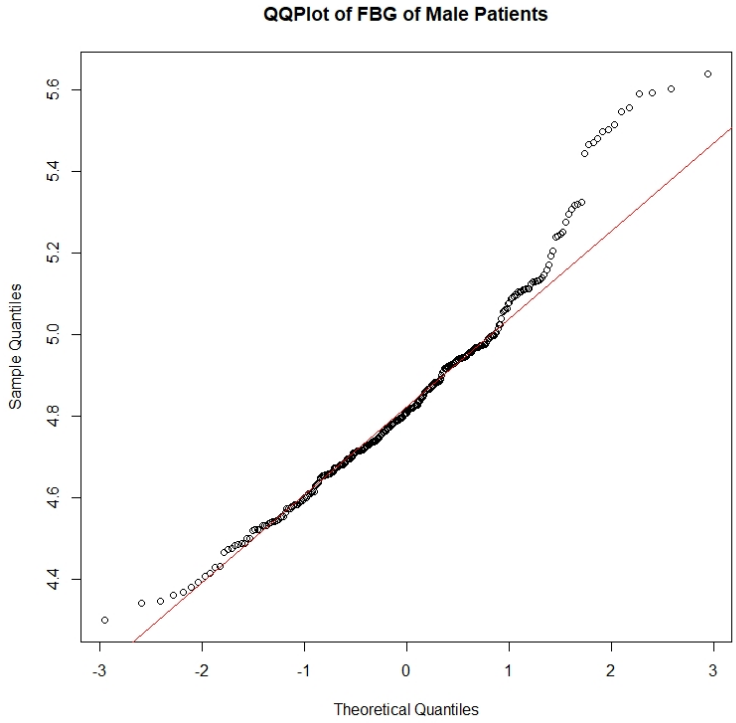
We can see that both of our subsets are right skewed. Therefore, we should normalize them via some transformation. Can you guess which function would normalize our data?

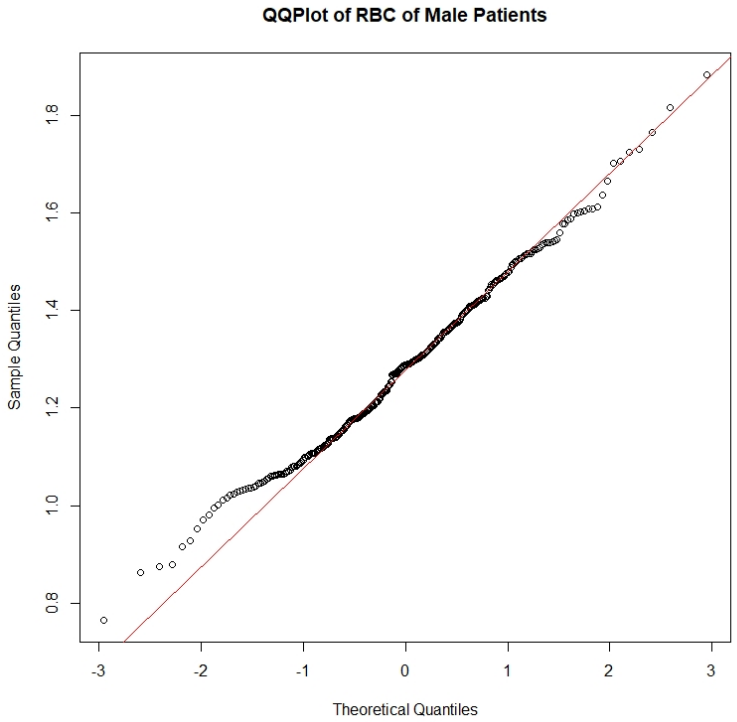
In fact, using a `log(()` transformation, our data gets more normalized. It isn't perfectly normally distributed, but it does fit a normal distribution curve much better.

```
# Log transformation
> qqnorm(log(males_RBC), main = 'QQPlot of RBC of Male
↪  Patients')
> qqline(log(males_RBC), col = 'red')
> qqnorm(log(males_glucose), main = 'QQPlot of FBG of
↪  Male Patients')
> qqline(log(males_glucose), col = 'red')
```


QQPlot of FBG of Male Patients
Sample Quantiles
Theoretical Quantiles
100
150
200
250
-3
-2
-1
0
1
2
3


QQPlot of RBC of Male Patients
Sample Quantiles
Theoretical Quantiles
2
3
4
5
6
-3
-2
-1
0
1
2
3


QQPlot of FBG of Male Patients
Sample Quantiles
Theoretical Quantiles
4.4
4.6
4.8
5.0
5.2
5.4
5.6
-3
-2
-1
0
1
2
3


QQPlot of RBC of Male Patients
Sample Quantiles
Theoretical Quantiles
0.8
1.0
1.2
1.4
1.6
1.8
-3
-2
-1
0
1
2
3

Hypothesis

Since our `log` transformation relatively passes the normalization assumption, we will proceed with the one-sample t test.

We define $\mu =$ the mean fast glucose level of non-diabetic patients.

$H_0 : \mu = 95$ mg/dL

$H_A : \mu \neq 95$ mg/dL

Next, we provide the `t.test()` function with our subset sample vector and its μ value.

```
# T test
> t.test(log(males_glucose), mu = 95)

        One Sample t-test

data:  log(males_glucose)
t = -6334.3, df = 308, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 95
95 percent confidence interval:
 4.811286 4.867301
sample estimates:
mean of x
 4.839293

> t.test(log(males_RBC), mu = 5.5)

        One Sample t-test

data:  log(males_RBC)
t = -404.34, df = 312, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 5.5
95 percent confidence interval:
 1.265290 1.306304
sample estimates:
mean of x
 1.285797
```

Fasting Blood Glucose

We have found evidence that the mean Fasting Blood Glucose content of male patients with Diabetes is significantly different from 95 mg/dL ($t = -6334.3, df = 308, p - value < 2.2e - 16$). The 95 percent

confidence interval for the true mean Fasting Blood Glucose levels of males patients with Diabetes is [4.811286 mg/dL, 4.867301 mg/dL].

Red Blood Count

We have found evidence that the mean Red Blood Cell Couunt of male patients with Diabetes is significantly different from 5.5 million cells per microliterL ($t = -404.34, df = 312, p - value < 2.2e - 16$). The 95 percent confidence interval for the true mean Red Blood Cell Count of male patients with Diabetes is [1.265290 million cells per microliter , 1.306304 million cells per microliter].

One-Sided Alternative Hypothesis

In some cases, we could predict the alternative hypothesis with high probability. For example, I knew with high probability that male patients with Diabetes would tend to have a higher FBG, that is by the very definition of Diabetes [9] itself. Therefore, in this case, we can justify a one-sided alternative hypothesis or one-sided test. In this case:

[9] Diabetes is a group of diseases that result in too much sugar in the blood (high blood glucose)

We define $\mu =$ the mean fast glucose level of non-diabetic patients.

$H_0 : \mu = 95$ mg/dL

$H_A : \mu > 95$ mg/dL

Ergo, running the one-sided test in R gives the following:

```
# One-sided T test
> t.test(log(males_glucose), mu = 95, alternative =
↪  'greater')

        One Sample t-test

data:  log(males_glucose)
t = -6334.3, df = 308, p-value = 1
alternative hypothesis: true mean is greater than 95
95 percent confidence interval:
 4.81581     Inf
sample estimates:
mean of x
 4.839293
```

Two Sample t Tests

Objective

In the previous section, we learned to compare a population mean to the true mean via a one-sample t test. In this chapter, we will compare if two population mean are equal or not. Our goal here is to:

1. Confirming the assumptions for two-sample t testing
2. Conduct a paired two-sample t test and interpret results
3. Conduct a independent two-sample t test and interpret results

New Functions

`install.packages()` Installs external R package

`library()` Calls and opens an external packaged

`laveneTest()` Runs Lavene's Test of equal variance

Importing the Dataset

This chapter will use the **Breast Cancer Wisconsin (Diagnostic) Data Set** dataset. The `csv` file can be obtained at https://www.kaggle.com/uciml/breast-cancer-wisconsin-data.

```
# Importing the Dataset
> my_data <- read.csv("data.csv", StringsAsFactors = T)
> head(my_data[c(1,2,3,4)],)
        id diagnosis radius_mean texture_mean
1   842302         M       17.99        10.38
2   842517         M       20.57        17.77
3 84300903         M       19.69        21.25
4 84348301         M       11.42        20.38
5 84358402         M       20.29        14.34
6   843786         M       12.45        15.70
```

Analyzing the Data

In this chapter, we will answer the following questions about our dataframe:

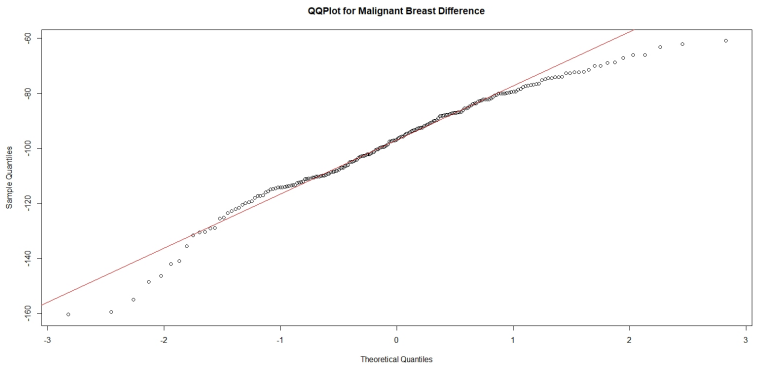
1. **For malignant breasts cancers, are their mean radius greater, on average, than their mean perimeter?**
2. **Several studies have shown that mammographic texture features are associated with breast cancer risk independent of the contribution of breast density. Thus, texture features may provide novel information for risk stratification.[10]Therefore, do malignant and benign breast cancers have the same mean texture?**

[10] https://breast-cancer-research.biomedcentral.com/articles/10.1186/s13058-016-0778-1

Paired T Test

Paired T Test help compare two different numeric variables for a single subject. Assuming the breasts measurements in our dataset were randomly collected and independent, lets check the normality assumption. However, for paired t tests, we need to be sure that we are checking if the *difference* between our two numeric variables are normally distributed. Observe the following:

```
# Indexing malignant breasts
> mal_breasts <- my_data[my_data$diagnosis == "M",]
# Checking normality
> mal_breasts_diff <- mal_breasts$radius_mean -
↪  mal_breasts$perimeter_mean
> qqnorm(mal_breasts_diff, main = 'QQPlot for Malignant
↪  Breast Difference')
> qqline(mal_breasts_diff, col = 'red')
```



We can see our mean difference distribution is relatively normal, recall not all points have to fall on the red line.

Testing our Hypothesis

Let μ_d = mean difference between mean radius and mean perimeter lengths of malignant breast cancer tumors

$H_0 : \mu_d = 0$ cm

$H_A : \mu_d > 0$ cm

Using the `t.test()` function, we need to specify that `paired = T` and use the one-sided alternative function or `alternative = 'greater'`

```
# Two sided t test
> t.test(mal_breasts$radius_mean,
↪  mal_breasts$perimeter_mean, paired = T, alternative =
↪  'greater')

        Paired t-test

data:  mal_breasts$radius_mean and
↪  mal_breasts$perimeter_mean
t = -76.358, df = 211, p-value = 1
alternative hypothesis: true difference in means is
↪  greater than 0
95 percent confidence interval:
 -100.0208       Inf
sample estimates:
mean of the differences
              -97.90255
```

Here, $p < 0.5$, therefore, we can conclude that the mean radius of malignant breast cancer is *not* significantly greater than the mean perimeter length of malignant breast cancers, that is the mean difference are equal to each other.

Independent t Test

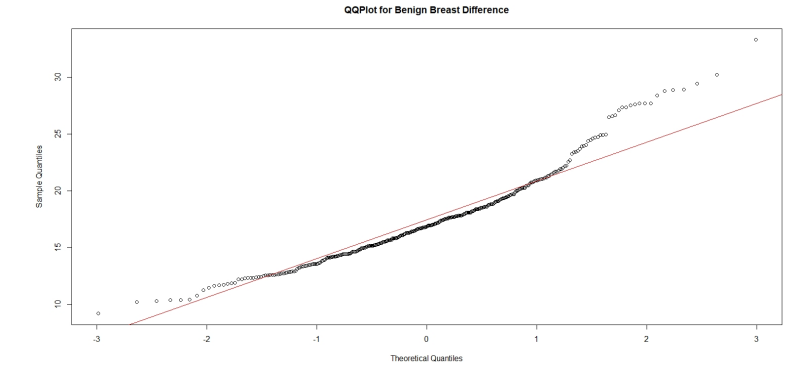
Independent T Test lets us compare a numeric variable for two different populations. First, we should confirm that the mean texture for malignant and benign breasts are normally distributed and have equal variances.

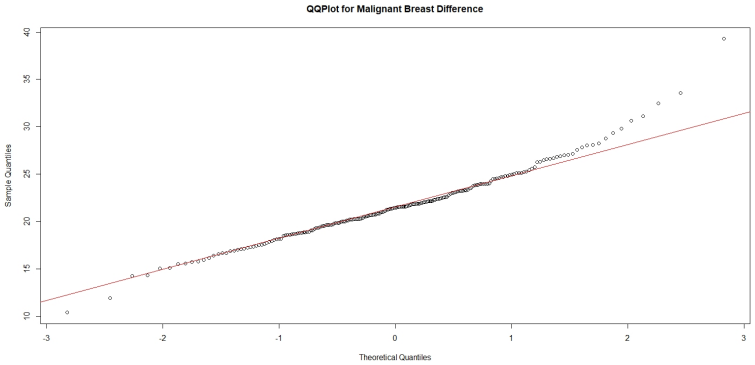
```
# Subset data based on diagnosis and mean texture
> mal_breasts_texture <-
↪  my_data$texture_mean[my_data$diagnosis == "M",]
> ben_breasts_texture <-
↪  my_data$texture_mean[my_data$diagnosis == "B", ]
```

```
# Checking normality
> qqnorm(mal_breasts_texture, main = 'QQPlot for Malignant
↪  Breast Difference')
> qqnorm(ben_breasts_texture, main = 'QQPlot for Malignant
↪  Breast Difference')
> qqline(mal_breasts_texture, col = 'red')
> qqline(ben_breasts_texture, col = 'red')
```

Running the code tells us that our benign breasts are not normally distributed. However, we can apply some transformation to help normalize them better.

```
> qqnorm((ben_breasts_texture)-0.5, main = 'QQPlot for
↪  Benign Breast Difference')
> qqline(ben_breasts_texture, col = 'red')
```





Checking Variances

In order to check if our two populations have equal variances, we can run Levene's Test. However, the `leveneTest()` function is in the external package `cars`.

```
# Installing library
> install.packages("car")
# Opening library
> library(car)
# Running Levene's Test
> leveneTest(texture_mean ~ diagnosis, data = my_data)
Levenes Test for Homogeneity of Variance (center = median)
       Df F value Pr(>F)
group   1   0.684 0.4086
      567
```

We see that the data passes the Levene's Test of equal variance ($p = 0.4$). [11]

[11] If our data failed the equal variance assumption, then we will use the un-pooled version called Welch's Test.

Running Independent T Test

Let μ_M = mean texture of malignant breast cancer

Let μ_B = mean texture of benign breast cancer

$H_0 : \mu_M = \mu_B$

$H_0 : \mu_M \neq \mu_B$

```
# Independent T Test
> t.test(texture_mean ~ diagnosis, data = my_data,
↪  var.equal = T)

        Two Sample t-test

data:  texture_mean by diagnosis
t = -10.867, df = 567, p-value < 2.2e-16
alternative hypothesis: true difference in means between
↪  group B and group M is not equal to 0
95 percent confidence interval:
 -4.357107 -3.023181
sample estimates:
mean in group B mean in group M
       17.91476        21.60491
```

Of note, if our data failed the equal variance assumption, we can either input `var.equal = F` which is the default value anyways.

Therefore, we can conclude that the mean texture of malignant and benign breast cancers are significantly different ($p < 0.05$).

Testing Single Categorical Variables

Objective

In this section, we will learn to test the distribution of a single categorical variables using:

1. **Binomial Test**

2. **Chi-Squared Goodness of Fit Test**

New Functions

`binom.test()` Runs Binomial Test

`chisq.test()` Runs Chi Squared Test

Importing the Dataset

This chapter will use the **Prostate Cancer** dataset. The `csv` file can be obtained at https://www.kaggle.com/sajidsaifi/prostate-cancer.

```
# Importing the Dataset
> my_data <- read.csv("Prostate_Cancer.csv",
↪  StringsAsFactors = T)
> head(my_data)
  id diagnosis_result radius texture perimeter area
  ↪  smoothness compactness symmetry fractal_dimension
1  1                M     23      12       151  954
↪  0.143       0.278    0.242            0.079
2  2                B      9      13       133 1326
↪  0.143       0.079    0.181            0.057
3  3                M     21      27       130 1203
↪  0.125       0.160    0.207            0.060
4  4                M     14      16        78  386
↪  0.070       0.284    0.260            0.097
```

```
5  5                 M       9       19        135 1297
↪  0.141        0.133     0.181             0.059
6  6                 B      25       25         83  477
↪  0.128        0.170     0.209             0.076
```

Analyzing The Data

In this chapter, we will answer the following question:

1. **According to NCBI, Lesion diameter 20 mm, but not 15 mm, was a significant risk factor for lymph node metastasis[12]. Therefore, regarding lesions with radius size of over 10mm, is the proportion of malignant breast equal to 1/2?**

[12] https://pubmed.ncbi.nlm.nih.gov/29291666/

Binomial Test

Lets investigate our dataset first.

```
# Return table of prostate cancers
> table(my_data$diagnosis_result)

 B  M
38 62
# Return number of prostate cancers with radius greater
↪  than 10mm
> sum(my_data$radius > 10)
[1] 85
# Return number of prostate cancers with radius greater
↪  than 7.5mm
> sum(my_data$radius > 7.5)
[1] 100
```

To test if, based on the prostate lesions with radius over 10mm, the proportion of malignant prostate cancers equals one-half, we can run the binomial test and consider a malignant cancers as a "success," or outcome of interest.

Assuming our data collection was random and independent, we will consider the following:

Let π = the proportion of malignant prostate lesions

$H_0 : \pi = 1/2$

$H_A : \pi \neq 1/2$

Providing the `binom.test()` function with three arguments, we define the following variables:

x number of successes in our dataset

n sample size

p π from null hypothesis

Using the `table()` function, we can obtain our x and n:

```
# Creating subset with lesion radius greater than 10mm
> greater_10 <- my_data[(my_data$radius >= 10),]
# Obtaining x
> sum(greater_10$diagnosis == 'M')
[1] 52
# Obtaining n
> table(greater_10$diagnosis_result)

 B  M
33 52
# Conducting Binomial Test
> binom.test(52,85,1/2)

        Exact binomial test

data:  52 and 85
number of successes = 52, number of trials = 85, p-value =
↪  0.05025
alternative hypothesis: true probability of success is not
↪  equal to 0.5
95 percent confidence interval:
 0.4998837 0.7156216
sample estimates:
probability of success
             0.6117647
```

Based on our data with prostate lesions with radius greater than 10mm, we did not find that the actual proportion of malignant prostate cancers is significantly different from one-half (n = 85, p = 0.0505 > 0.05).

Chi-Squared Goodness of Fit Test

Importing the Dataset

This section will use the **Heart Failure Prediction Dataset** dataset. The `csv` file can be obtained at https://www.kaggle.com/fedesoriano/heart-failure-prediction.

```
# Importing the Dataset
> my_data <- read.csv("heart.csv", StringsAsFactors = T)
> head(my_data)
  Age Sex ChestPainType RestingBP Cholesterol FastingBS
  ↪  RestingECG MaxHR ExerciseAngina Oldpeak ST_Slope
  ↪  HeartDisease
1  40   M           ATA       140         289         0
↪  Normal   172             N     0.0       Up
↪  0
2  49   F           NAP       160         180         0
↪  Normal   156             N     1.0     Flat
↪  1
3  37   M           ATA       130         283         0
↪  ST    98              N     0.0       Up            0
4  48   F           ASY       138         214         0
↪  Normal   108             Y     1.5     Flat
↪  1
5  54   M           NAP       150         195         0
↪  Normal   122             N     0.0       Up
↪  0
6  39   M           NAP       120         339         0
↪  Normal   170             N     0.0       Up
↪  0
>
```

Analyzing The Data

In this section, we will answer the following questions:

1. **Are all four types of chest pain represented in equal proportion, considering if the patient has past medical history of heart disease?**

Lets begin with our hypothesis:

H_0 = The ratio of ATA:NAP:ASY:TA is 1:1:1:1

H_A = The ratio of ATA:NAP:ASY:TA is NOT 1:1:1:1

```
# Table of observed counts
> observed_count <-
↪  table(my_data$ChestPainType[my_data$HeartDisease ==
↪  1])
> observed_count
```

```
ASY ATA NAP  TA
392  24  72  20
```

The Chi-square assumes sample collection of sufficient size, in particular, more than five counts per variable item. Fortunately, we can use the `chisq.test()` function to check this assumption for us by adding $expected at the end and using the `p=` option function.

```
# Checking assumption
> chisq.test(observed_count, p=
↪  c(1/4,1/4,1/4,1/4))$expected
ASY ATA NAP  TA
127 127 127 127
```

Because all counts were greater than five, we pass the sufficient sampling assumption. We can proceed with the test by dropping the `expected` line at the end.

```
# Chi Square Test
> chisq.test(observed_count, p= c(1/4,1/4,1/4,1/4))

        Chi-squared test for given probabilities

data:  observed_count
X-squared = 750.46, df = 3, p-value < 2.2e-16
```

The sample data from patients with heart disease provides evidence that the actual ratio of ATA:NAP:ASY:TA chest pain type is significantly different from 1:1:1:1 ($\chi^2 = 750.46, df = 3, p < 0.05$)

Testing Multiple Categorical Variables

Objective

Using the Chi-Squared Test of Independence, we can test the distribution and relationship of two distinct categorical variables.

New Functions

`prop.table()` Calculates row/column proportions in contingency table

Importing the Dataset

This chapter will use the **BMI-Dataset** dataset. The `csv` file can be obtained at https://www.kaggle.com/yasserh/bmidataset.

```
# Importing the Dataset
> my_data <- read.csv("bmi.csv", StringsAsFactors = T)
> head(my_data)
  Gender Height Weight Index
1   Male    174     96     4
2   Male    189     87     2
3 Female    185    110     4
4 Female    195    104     3
5   Male    149     61     3
6   Male    189    104     3
```

Here we have two categorical variables, Gender, and BMI Index.

Index :

0 - Extremely Weak

1 - Weak

2 - Normal

3 - Overweight

4 - Obesity

5 - Extreme Obesity

Analyzing the Data

In this chapter, will answer the following question:

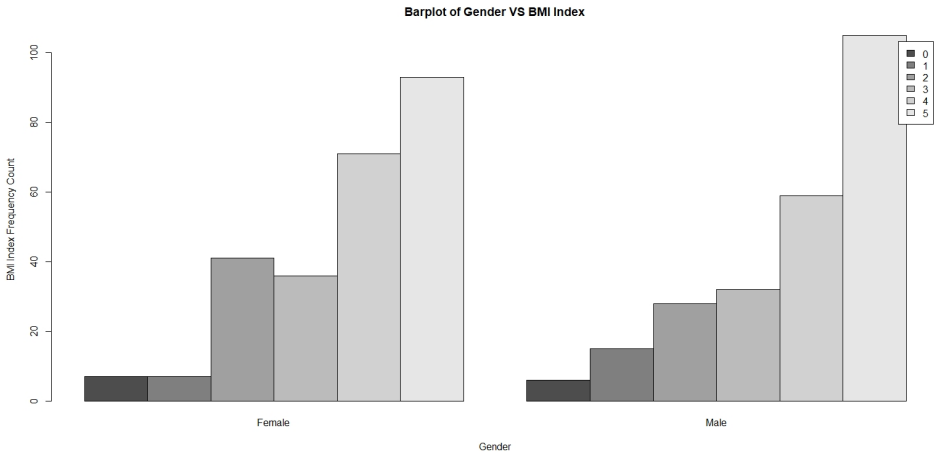
1. **Is the presence of overweight, obesity and extreme obesity for BMI Index related to Gender?**

Creating a contingency table gives us the count or frequency of gender and BMI index in our dataset.

```
# Creating table object
> cont_table <- table(my_data$Index, my_data$Gender)
> cont_table

   Female Male
0       7    6
1       7   15
2      41   28
3      36   32
4      71   59
5      93  105
```

Using the `barplot()` function, we can get a graphical representation of this distribution.

```
# Barplot
> barplot(cont_table, xlab = 'Gender', ylab= 'BMI Index
↪  Frequency Count', main = 'Barplot of Gender VS BMI
↪  Index', legend = T, beside = T)
```



Of note, the `beside = T` code tells R to place the Gender count side by side instead of stacked.

Although the distribution between males (245) and females (255) are relatively equivalent, lets look at the relative frequencies for the distribution of BMI index between the two genders. Here, we will use the `prop.table()` function which will give display the row proportions (1) or column proportions (2).

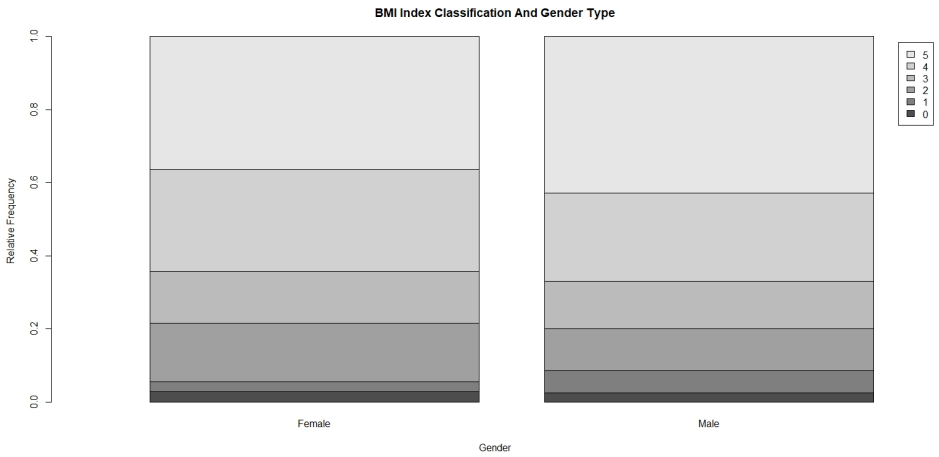
```
# Proportions table
> prop.table(cont_table, 1)

      Female      Male
  0 0.5384615 0.4615385
  1 0.3181818 0.6818182
  2 0.5942029 0.4057971
  3 0.5294118 0.4705882
  4 0.5461538 0.4538462
  5 0.4696970 0.5303030
```

Here we see that, in regards to the extremely weak BMI index, 53 percent were females and 46 percent were males, ect. Lets base our proportions table on columns to get us a better representation for our question of interest.

```
> prop.table(cont_table, 2)
       Female       Male
  0 0.02745098 0.02448980
  1 0.02745098 0.06122449
  2 0.16078431 0.11428571
  3 0.14117647 0.13061224
  4 0.27843137 0.24081633
  5 0.36470588 0.42857143
```

Now, we see that, out of all females, 78 percent of them were classified in the overweight, obesity and extreme obesity group. In addition, out of all males, 80 percent of all males were classified in the said groupings. Lets use the `barplot()` function to get a graphical representation of our proportions.

```
# Creating Mosaic Plot
> barplot(prop.table(cont_table, 2), legend = T, main =
↪  'BMI Index Classification And Gender Type', xlab =
↪  'Gender', ylab = 'Relative Frequency', xlim =
↪  c(0,2.5))
```



From our barplot above, we see that the distribution of overweight, obesity and extreme obesity BMI Index is almost relatively similar when comparing males and females. Using the Chi-Squared Test of Independence, we can test if the proportions of the BMI indexes are truly similar.

Chi Squared Test of Independence

We begin, by defining the following:

H_0 = BMI Index level and Gender are independent

H_A = BMI Index level and Gender are not independent

First, we need to check the sufficient count assumption.

```
# Checking count
> chisq.test(cont_table)$expected
    Female  Male
0   6.63  6.37
1  11.22 10.78
2  35.19 33.81
3  34.68 33.32
4  66.30 63.70
5 100.98 97.02
```

All expected counts are over five, so we can say with certainty that our dataset has passed the sufficient count assumption. Proceeding with the statistical testing gives us the following result:

```
> chisq.test(cont_table)

        Pearsons Chi-squared test
```

```
data:  cont_table
X-squared = 7.3085, df = 5, p-value = 0.1987
```

Overall, we have evidence that the distribution of BMI Index were independent and that BMI Index were not significantly different when comparing males and females ($\chi^2 = 7.309, df = 5, p = 0.2$). Qualitatively, this means that, based on our data, the proportion of overweight, obese and extreme obese did not depend on gender.

Non-Parametric Methods

Objective

Sometimes, data will fail the assumptions. One way to rectify this issue is to run non-parametric methods. In this chapter, we will learn:

1. **Sign Test**
2. **Mann-Whitney U-Test**

New Functions

`wilcox.test()` Runs Man-Whitney U-Test

Importing the Dataset

This chapter will use the **Used Car Data** dataset. The `csv` file can be obtained athttps://www.kaggle.com/sanjeetsinghnaik/used-car-information.

```
# Importing the Dataset
> my_data <- read.csv("dataset.csv", StringsAsFactors = T)
> head(my_data)
  X Id year         brand
   ↪  full_model_name model_name   price
1 0  0 2016         Honda
 ↪  Honda Brio S MT       Brio  425000
2 1  1 2012        Nissan
 ↪  Nissan Sunny XV Diesel      Sunny  325000
3 2  2 2017        Toyota              Toyota Fortuner 2.8
 ↪  4x2 MT [2016-2020]   Fortuner 2650000
4 3  3 2017 Mercedes-Benz Mercedes-Benz E-Class E 220d
 ↪  Expression [2019-2019]    E-Class 4195000
5 4  4 2012       Hyundai                   Hyundai Verna
 ↪  Fluidic 1.6 CRDi SX      Verna  475000
6 5  5 2012       Hyundai                         Hyundai
 ↪  i20 Sportz 1.2 BS-IV        i20  335000
```

Of note, this dataset was collected in India. Lets convert the Rupee to USD. The conversion is that 1 Rupee is 0.013 USD.

```
# Creating new column for USD Price
> my_data$PriceUSD <- my_data$price * 0.013
```

Analyzing the Data

In this chapter, will answer the following question:

1. **Is the average sales price of used cars sold in the year 2020 different from** $\$20,000$**?**
2. **Is there a difference between the distance traveled regarding the different Petrol and Diesel fuel type for used cars sold in the year 2020?**

Sign Test

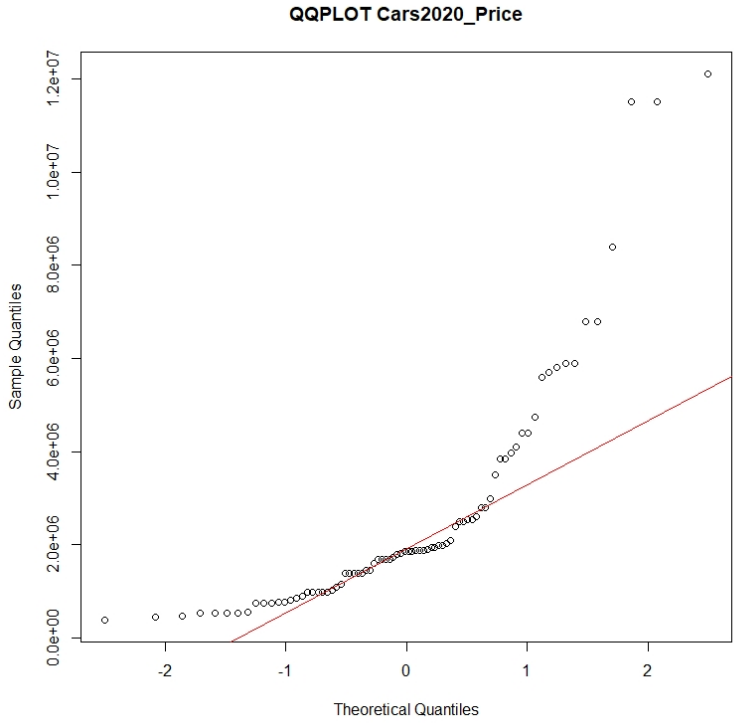
First, we create a subset with cars sold in the year 2020 and then we can view the distribution to test normality.

```
# Creating subset
> cars2020 <- my_data[(my_data$year == 2020),]
# Normality Assumption
> qqnorm(cars2020$price, main = 'QQPLOT Cars2020_Price')
> qqline(priceUSD, col = 'red')
```

Our subset definitely does not pass the normality assumption. We can apply some linear transformation, but lets use a non-parametric test. We define the following:

H_0 : The median sales price of used cars sold in 2020 $= \$20,000$.

H_A : The median sales price of used cars sold in 2020 $\neq \$20,000$.



We let car price *greater* $\$20,000$ as a *success*, and see that:

```
# Number of Successes
> sum(cars2020$PriceUSD > 20000)
[1] 49
# Total Number of Cars
> table(cars2020$year)
2020
  80
```

Using that $x = 49, n = 80, p = 0.5$, we get the following:

```
# Conducting Sign Test
> binom.test(49,80,0.5)

        Exact binomial test

data:  49 and 80
number of successes = 49, number of trials = 80, p-value =
↪  0.05666
alternative hypothesis: true probability of success is not
↪  equal to 0.5
95 percent confidence interval:
 0.4969755 0.7194349
```

```
sample estimates:
probability of success
                0.6125
```

Therefore, we have statistical evidence that the median price of used cars sold in 2020 was significantly different than $\$20,000$.

```
# Median
> median(cars2020$PriceUSD)
[1] 24050
```

Moreover, we can say with certainty that the median price in USD is about $\$24,050$.

Mann-Whitney U-Test

First, we need to create two separate dataframes for the two fuel types from our `cars2020 dataset` used previously.

```
# Diesel and Petrol Dataframes
> Diesel <- cars2020[(cars2020$fuel_type == 'Diesel'),]
> Petrol <- cars2020[(cars2020$fuel_type == 'Petrol'),]
```

Ironically, lets assume that our two datasets do not pass the normality assumption. [13] Using the *Mann-Whitney U-Test*, we define the following:

[13] This is true - our two datasets do not pass normality but checking will be left as an exercise to the reader.

H_0 : The distribution of distance travelled of cars sold in 2020 is similar between the two fuel types.

H_A : The distribution of distance travelled of cars sold in 2020 is not similar between the two fuel types.

Next, we use the `wilcox.test()` function to carry out our Mann-Whitney U-Test:

```
# Mann-Whitney U-Test
> wilcox.test(Diesel$distance_travelled.kms.,
↪  Petrol$distance_travelled.kms.)

        Wilcoxon rank sum test with continuity correction

data:  Diesel$distance_travelled.kms. and
↪  Petrol$distance_travelled.kms.
W = 920, p-value = 0.1258
alternative hypothesis: true location shift is not equal
↪  to 0
```

Ergo, we have statistical evidence that the distribution of distance travelled for used cars sold in 2020 is **not** significantly different between diesel and petrol.

Correlation and Regressional Models

Objective

Correlation and Regressional Analysis are tests that we can use to measure the relationship between two or more numerical variables. In this section, we will learn to:

1. **Conduct Regressional analysis and interpret its results**
2. **Conduct Correlation analysis and interpret its results**

New Functions

cor() Returns Pearson Correlation Coefficient between two distinct variables

cor.test() Runs Correlation test

lm() Defines a linear model object

summary() Displays `lm()` results

abline() Adds a line to an existing scatterplot

Importing the Dataset

This chapter will use the **US Health Insurance Dataset** dataset.
The `csv` file can be obtained athttps://www.kaggle.com/teertha/ushealthinsurancedataset.

```
# Importing the Dataset
> my_data <- read.csv("insurance.csv", stringsAsFactors =
↪  T)
> head(my_data)
  age    sex    bmi children smoker    region   charges
1  19 female 27.900        0    yes southwest 16884.924
2  18   male 33.770        1     no southeast  1725.552
3  28   male 33.000        3     no southeast  4449.462
```

```
4  33   male 22.705        0     no northwest 21984.471
5  32   male 28.880        0     no northwest  3866.855
6  31 female 25.740        0     no southeast  3756.622
```

Analyzing the Dataset

In this chapter, will answer the following question:

1. **Is there a linear relationship between Age and Insurance price in the US for non-smokers?**

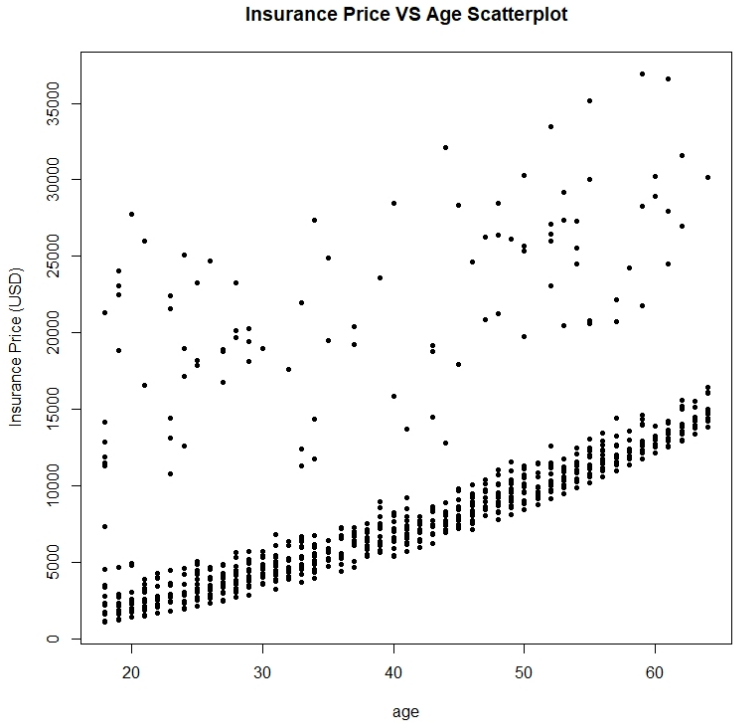
Correlation Analysis

Here, we want to measure or visualize the strength of the linear relation between two numeric variables. Before we do any statistical testing, we can visualize this if this relationship exists somewhat by using a scatterplot.

```
# Creating Subset for nonsmokers
> nonsmoker <- my_data[(my_data$smoker == 'no'),]
# Scatterplot
> plot(nonsmoker$charges ~ nonsmoker$age, xlab = 'Age',
↪  ylab = 'Insurance Price (USD)', main = 'Insurance
↪  Price VS Age ScatterPlot')
```

Here, we can see a somewhat strong linear correlation with some outliers. Lets use the `cor()` function to test our Pearson correlation coefficient.

```
# Pearson Coefficient
> cor(nonsmokers$charges, nonsmokers$age)
[1] 0.6279468
```



Therefore, we see that there exists a strong positive relation between insurance price and age ($r = 0.63$). Lets continue with our statistical testing, we define the following:

Let $\rho =$ the correlation between Insurance Price and Age

$H_0 : \rho = 0$

$H_0 : \rho \neq 0$

Here, the order of the variable does not matter when inputting in the `cor.test()` function.

```
# Correlation test
> cor.test(nonsmokers$age, nonsmokers$charges)

        Pearsons product-moment correlation

data:  nonsmokers$age and nonsmokers$charges
t = 26.294, df = 1062, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.5901183 0.6630239
sample estimates:
      cor
0.6279468
```

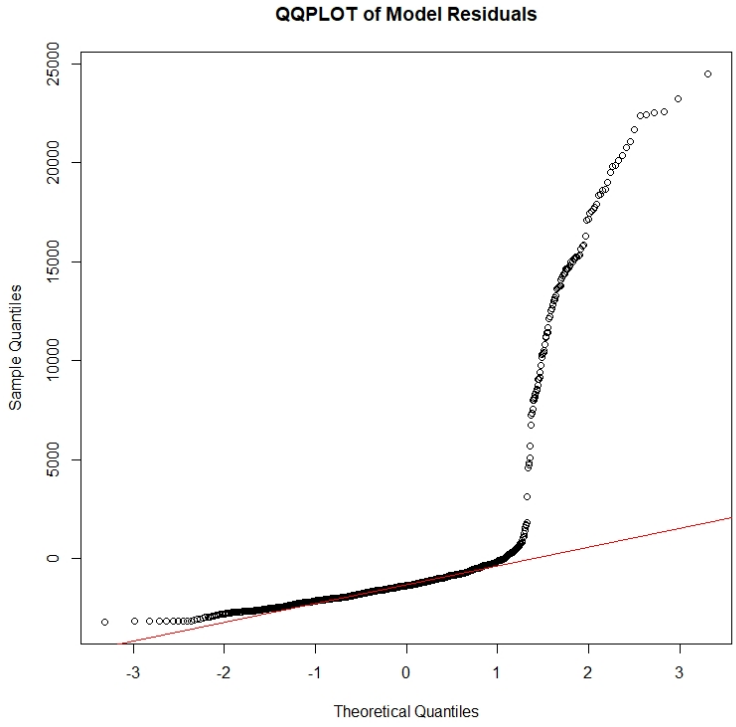
We have statistical evidence that the correlation between Insurance Price and Age is not zero - in other words, there does exist a relation between the two variables. In fact, we are 95% confident that the correlation coefficient is between 0.59 and 0.66.

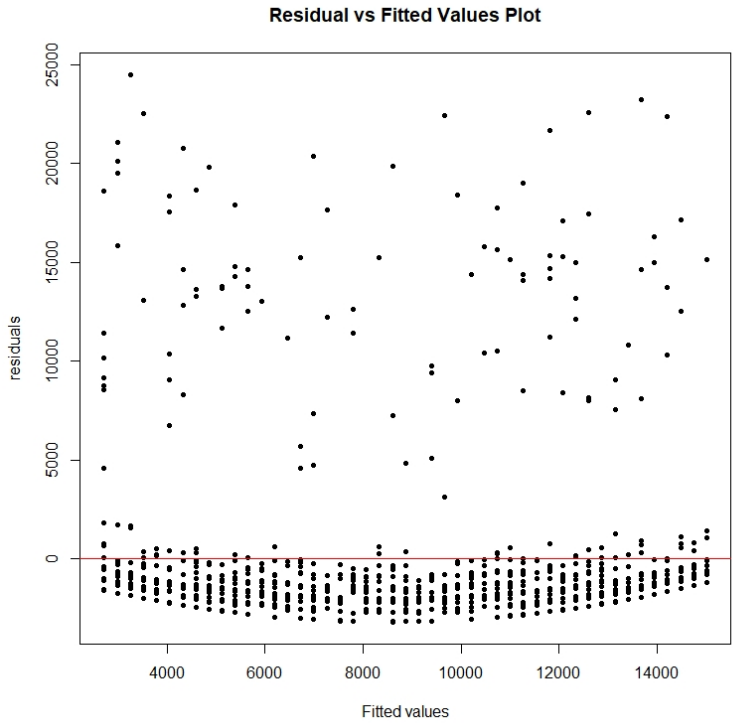
Linear Regression

Linear Regressional Analysis will estimate the linear model parameters and allows one to make predictions of the response variable.

First, we will build our `lm()` model, while remembering to put the response variable first as a function of the explanatory variable which we will input last.

```
# Building Linear Model
> my_model <- lm(nonsmokers$charges ~ nonsmokers$Age)
# Testing normality and equal variance assumptions
> qqnorm(my_model$residuals, main = 'QQPLOT of Model
↪  Residuals')
> qqline(my_model$residuals, col = 'red')
> plot(my_model$fitted.values, my_model$residuals, xlab =
↪  'Fitted values', ylab = 'residuals', main = 'Residual
↪  vs Fitted Values Plot', pch = 20)
> abline(h=0, col = 'red')
```





From above, we have already confirmed that insurance price and age are linearly related. However, we need to confirm that the residuals of

14

14

1. You should not remove outliers just because they make the distribution of the residuals non-normal. You may examine the case that has that high residual and see if there are problems with it (the easiest would be if it is a data entry error) but you must justify your deletion on substantive grounds.
2. Assuming there is no good reason to remove that observation, you can run the regression with and without it and see if there are any large differences in the parameter estimates; if not, you can leave it in and note that removing it made little difference
3. If it makes a big difference, then you could try robust regression, which deals with outliers or quantile regression, which makes no assumptions about the distribution of the residuals.

the model are normally distributed and have equal variance.

Note that the residuals are not normal. In addition, the scatterplot does shows not show a a symmetric cloud of points above and below the line of fitted values versus residuals plot. Thus, not all assumptions have been met. However, it would be detrimental to remove such outliers. Lets run the test - we begin by defining the following:

Let β = slope of Insurance price on Age in nonsmoking US citizens

$H_0 : \beta_1 = 0$

$H_0 : \beta_1 \neq 0$

Using the `summary()` function, we can view our linear model.

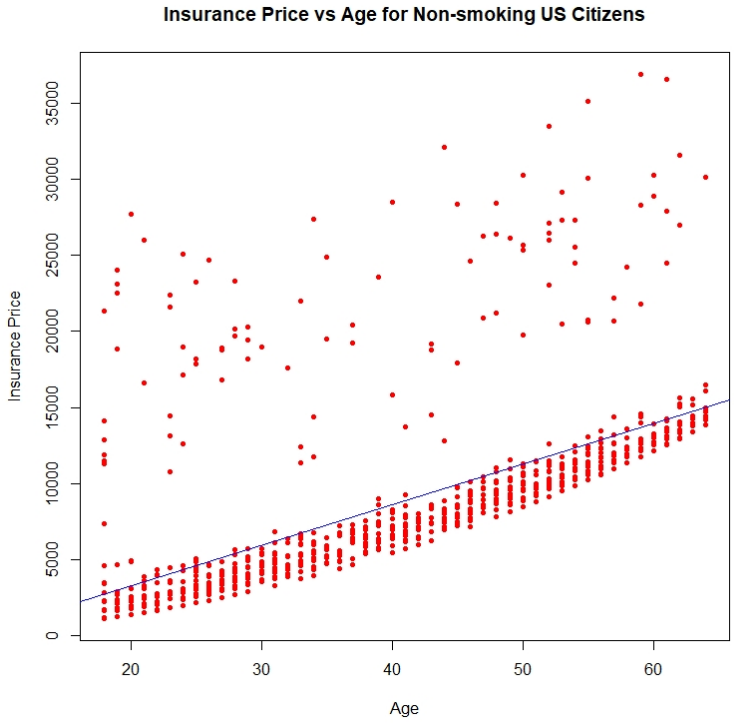
```
# Regresional Analysis Model
> summary(my_model)

Call:
lm(formula = nonsmokers$charges ~ nonsmokers$age)

Residuals:
    Min      1Q  Median      3Q     Max
-3182.9 -1948.6 -1363.8  -665.2 24470.7

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    -2091.42     425.10   -4.92    1e-06 ***
nonsmokers$age   267.25      10.16   26.29   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
↪  ’ 1

Residual standard error: 4667 on 1062 degrees of freedom
Multiple R-squared:  0.3943,        Adjusted R-squared:
↪  0.3937
F-statistic: 691.4 on 1 and 1062 DF,  p-value: < 2.2e-16
```

This data provides evidence that Insurance Price is significantly linearly related to Age for non-smoking US citizens. For every increase in year for age, Insurance Price increases by $267.25, on average.

Next, we can see how well this model fits our data by looking at the Multiple R-squared value, which is appropriate for models with single explanatory variable. Here, we see that 39% of the variation in Insurance Price can be explained by variation in Age.

Next, we will apply some data visualizations to view our model.

```
# Linear Model Scatter Plot
> plot(nonsmokers$charges, nonsmokers$age, xlab = 'Age',
↪   ylab = 'Insurance Price', main = 'Insurance Price vs
↪   Age for Non-smoking US Citizens', pch = 20, col =
↪   'red')
> abline(my_model, col = 'blue')
```



ANOVA Analysis

Objective

ANOVA Analysis or analysis of variance is an analytical tool used to measure the effect of one or more explanatory variables on a numerical response variable. This testing overall measures the differences among means and provides a statistical test of whether two or more population means are equal. It is an extension of the simple *T-test* and extends to multiple means.

New Functions

aov() Defines ANOVA object

TurkeyHSD() Runs post-hoc pairwise comparisons via Tukey adjustment [15]

drop1 Provides correct sums of squares for multi-factor ANOVA models

[15] Tukey Multiple Comparison Test determines if and which means are different from a set of other means.

Importing the Dataset

This chapter will use the **Cirrhosis Prediction Dataset** dataset.
The `csv` file can be obtained at https://www.kaggle.com/fedesoriano/cirrhosis-prediction-dataset.

```
# Importing the Dataset
> my_data <- read.csv("cirrhosis.csv", stringsAsFactors =
↪  T)

# Data Cleaning - Creating new variable for Staging
> mapping <- c("Stage 1","Stage 2", "Stage 3", "Stage 4" )
> names(mapping) <- c(1,2,3,4)
> my_data$Staging <- mapping[as.character(my_data$Stage)]
> head(my_data)
```

```
  ID N_Days Status            Drug   Age Sex Ascites
   ↪  Hepatomegaly Spiders Edema Bilirubin Cholesterol
1  1    400      D D-penicillamine 21464   F       Y
 ↪  Y       Y      Y      14.5         261
2  2   4500      C D-penicillamine 20617   F       N
 ↪  Y       Y      N       1.1         302
3  3   1012      D D-penicillamine 25594   M       N
 ↪  N       N      S       1.4         176
4  4   1925      D D-penicillamine 19994   F       N
 ↪  Y       Y      S       1.8         244
5  5   1504     CL          Placebo 13918   F       N
 ↪  Y       Y      N       3.4         279
6  6   2503      D          Placebo 24201   F       N
 ↪  Y       N      N       0.8         248
  Albumin Copper Alk_Phos   SGOT Tryglicerides Platelets
   ↪  Prothrombin Stage Staging
1    2.60    156   1718.0 137.95           172       190
 ↪  12.2     4 Stage 4
2    4.14     54   7394.8 113.52            88       221
 ↪  10.6     3 Stage 3
3    3.48    210    516.0  96.10            55       151
 ↪  12.0     4 Stage 4
4    2.54     64   6121.8  60.63            92       183
 ↪  10.3     4 Stage 4
5    3.53    143    671.0 113.15            72       136
 ↪  10.9     3 Stage 3
6    3.98     50    944.0  93.00            63        NA
 ↪  11.0     3 Stage 3
```

Analyzing the Data

In this chapter, we will answer the following question:

1. **Does blood Cholesterol levels differ based on presence of edema?**
2. **What about if we consider sex? Hepatomegaly?**
3. **Is the effect of disease staging on blood cholesterol levels the same for sexes? Presence of Edema?**

One-Way ANOVA

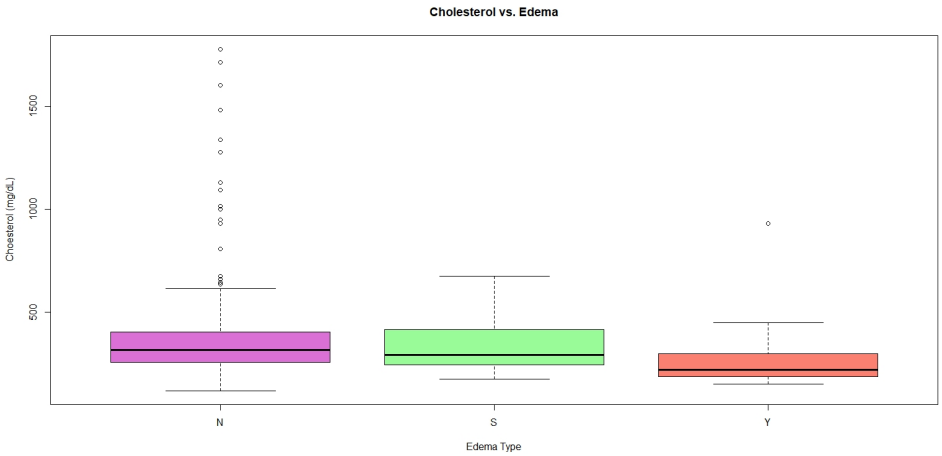
In order to measure the effects of blood cholesterol levels and its variations based on edema presence, via the `aov()` function, we will first

build an ANOVA model object.

```
# ANOVA Model
> my_anova <- aov(my_data$Cholesterol ~ my_data$Edema)
```

Before we proceed with the ANOVA test, we should test the normality and equal variances assumptions.

```
# Confirming Normality
> boxplot(my_data$Cholesterol ~ my_data$Edema, xlab =
↪  'Edema Type', ylab = 'Choesterol (mg/dL)', main =
↪  'Cholesterol vs. Edema', col = c('orchid', 'pale
↪  green', 'salmon'))
```



We see that the boxes are relatively equal in size and that the variances are almost similar. We can run Levene's test to confirm the following.

```
# Levenes Test
> leveneTest(my_data$Cholesterol ~ my_data$Edema)
Levenes Test for Homogeneity of Variance (center = median)
       Df F value Pr(>F)
group   2  0.3681 0.6924
      281
```

Here, we see that the data did passed Levene's Test of equal variance (p=0.69). Lets proceed with the one-way ANOVA; again if your dataset assumption failed, you should not try to alter your data by deleting any outliers; instead, try running the model with and without the outliers keeping the failed assumptions in mind when intepreting results. Of note, you could try running non-parametric versions of ANOVA; for example, we can run Welch and Brown-Forsythe tests

and then a post hoc test of Games-Howell. Moreover, we define the following:

H_0: The mean blood cholesterol levels of all edema types are equal

H_A: The mean blood cholesterol levels of all edema types are not equal

```
# One-Way ANOVA Summary
> summary(my_anova)
               Df   Sum Sq Mean Sq F value Pr(>F)
my_data$Edema   2   178466   89233   1.666  0.191
Residuals     281 15046445   53546
```

Here, we have statistical evidence to show that the mean blood cholesterol levels of patients are equal based on whether if they have edema.

However, notice that blood cholesterol levels is not equal when basing off of their disease staging. Observe the following:

```
# Staging Anova
> my_anova <- aov(my_data$Cholesterol ~ my_data$Staging)
> summary(my_anova)
                 Df   Sum Sq Mean Sq F value Pr(>F)
my_data$Staging   3   493063  164354   3.124 0.0263 *
Residuals       280 14731848   52614
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
↪  ’ 1
```

To find out which disease staging differ from each other, we can run a post-hoc analysis on our data via the `TukeyHSD()` function.

```
# Post-hoc analysis
> TukeyHSD(my_anova)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = my_data$Cholesterol ~ my_data$Staging)

$`my_data$Staging`
                      diff        lwr        upr     p adj
Stage 2-Stage 1  85.33417  -95.76216 266.430509 0.6160276
Stage 3-Stage 1 148.00602  -25.46015 321.472188 0.1243175
Stage 4-Stage 1  69.96437 -105.34648 245.275222 0.7311373
Stage 3-Stage 2  62.67185  -31.22973 156.573424 0.3126421
```

```
Stage 4-Stage 2 -15.36980 -112.63690  81.897300 0.9769615
Stage 4-Stage 3 -78.04165 -160.23360   4.150309 0.0696100
```

Observe that there is a significant difference in blood cholesterol levels when comparing stage 3 and stage 4 liver disease. However, the mean blood cholesterol levels of all the other disease staging are not that significanty different when compared to each other. Using the `summary.lm()$r.squared` $function, we can calculate the R - square value to report for our model.$

```
# R Squared value
> summary.lm(my_anova)$r.squared
[1] 0.03238529
```

Here, disease staging only accounts for 3% of the variation in blood cholesterol levels.

Multi-factor ANOVA

MULTIFACTOR ANOVA allows us to test if blood cholesterol levels differ based on disease staging and sex, ect.

```
# Define two way anova
> multi_anova <- aov(my_data$Cholesterol ~ my_data$Sex +
↪  my_data$Staging)
```

Since we have already checked the assumptions for Edema, you should confirm the assumptions for the other variables. However, lets assume that our variables have passed the relevant assumptions, we can state our hypotheses.

Hypothesis Set 1-

H_0 : Accounting for Sex, the mean blood cholesterol levels of all liver disease stage are equal

H_A : Accounting for Sex, the mean blood cholesterol levels of all liver disease stage are not equal

Hypothesis Set 2-

H_0 : Accounting for liver disease stage, the mean blood cholesterol levels of both sexes are equal

H_A : Accounting for liver disease stage, the mean blood cholesterol levels of both sexes are not equal

```
# Multi ANOVA
> drop1(multi_anova, ~., test = "F")
Single term deletions

Model:
my_data$Cholesterol ~ my_data$Sex + my_data$Staging
                Df Sum of Sq      RSS    AIC F value
                 ↪  Pr(>F)
<none>                       14731785 3093.3
my_data$Sex      1        63 14731848 3091.3  0.0012
 ↪  0.97244
my_data$Staging  3    491140 15222925 3096.6  3.1005
 ↪  0.02716 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
 ↪  ' 1
> summary.lm(multi_anova)$r.squared
[1] 0.03238944
```

Here, while controlling for liver staging, the patient's sex does not impact their blood cholesterol levels, as one should expect trivially. However, while controlling for sex, liver stage does impact mean blood cholesterol levels. Overall, our model explains for only 3% of the variation in blood cholesterol levels, again not a lot!

Interaction Effects

We can also add an interaction term to our model to see if the effects of liver disease stage differ for patients who are either male or female. We will add a third set of a hypothesis:

Hypothesis Set 3-

H_0 : There is an interaction between sex and liver disease stage on mean blood cholesterol

H_A : There is no interaction between sex and liver disease stage on mean blood cholesterol

```
# Interaction Model
> int_anova <- aov(my_data$Cholesterol ~ my_data$Sex *
 ↪  my_data$Staging)
```

```
# Interaction ANOVA
> drop1(int_anova, ~., test="F")
Single term deletions

Model:
my_data$Cholesterol ~ my_data$Sex * my_data$Staging
                            Df Sum of Sq      RSS    AIC F
                             ↪  value  Pr(>F)
<none>                                    14617497 3097.1
my_data$Sex                  1     31231 14648729 3095.7
 ↪  0.5897 0.44320
my_data$Staging              3    491455 15108953 3100.4
 ↪  3.0931 0.02745 *
my_data$Sex:my_data$Staging  3    114287 14731785 3093.3
 ↪  0.7193 0.54118
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
 ↪  ’ 1
> summary.lm(int_anova)$r.squared
[1] 0.03989603
```

Therefore, we can see that there exists no interaction between sex and liver disease stage on mean blood cholesterol levels. Ergo, the differences in mean blood cholesterol levels across the different liver stages are not significantly different for either sexes. Our model again only explains about 4% of the variation of the mean blood cholesterol levels.

General Linear Models

Objective

General Linear Models (GLM) or multiple regression models will test the effects of multiple numerical and categorical explanatory variables on a numeric response variable.
Here, we will learn to :

1. **Conduct General Linear Models and intepret results**
2. **Mean-center numeric explanatory variables**

Importing the Dataset

This chapter will use the **MRI and Alzheimers** dataset. The csv file can be obtained at https://www.kaggle.com/jboysen/mri-and-alzheimers?select=oasis_longitudinal.csv

```
# Import Dataset
> my_data <- read.csv("oasis_longitudinal.csv",
↪  stringsAsFactors = T)
> head(my_data)
  Subject.ID        MRI.ID       Group Visit MR.Delay M.F
  ↪  Hand Age EDUC SES MMSE CDR eTIV  nWBV   ASF
1  OAS2_0001 OAS2_0001_MR1 Nondemented     1        0   M
↪  R  87   14   2   27 0.0 1987 0.696 0.883
2  OAS2_0001 OAS2_0001_MR2 Nondemented     2      457   M
↪  R  88   14   2   30 0.0 2004 0.681 0.876
3  OAS2_0002 OAS2_0002_MR1    Demented     1        0   M
↪  R  75   12  NA   23 0.5 1678 0.736 1.046
4  OAS2_0002 OAS2_0002_MR2    Demented     2      560   M
↪  R  76   12  NA   28 0.5 1738 0.713 1.010
5  OAS2_0002 OAS2_0002_MR3    Demented     3     1895   M
↪  R  80   12  NA   22 0.5 1698 0.701 1.034
6  OAS2_0004 OAS2_0004_MR1 Nondemented     1        0   F
↪  R  88   18   3   28 0.0 1215 0.710 1.444
```

Analyzing the Data

In this chapter, we will answer the following question:

1. **Will Age and years of education significantly explain the variation in whole brain volume?**

General Linear Models

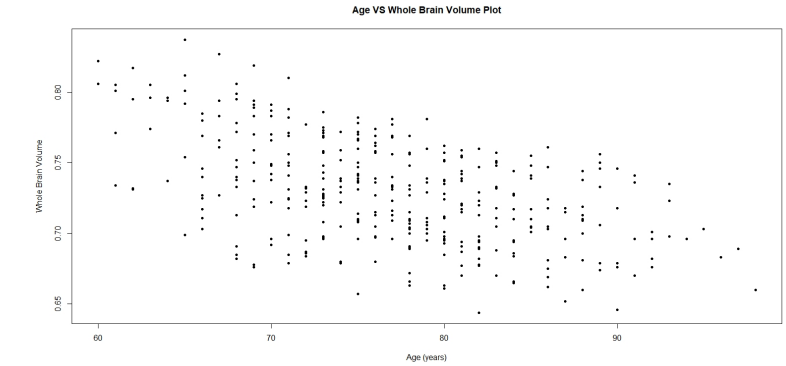
Lets first build a GLM that includes our explanatory variables using the lm() function from the previous chapters.

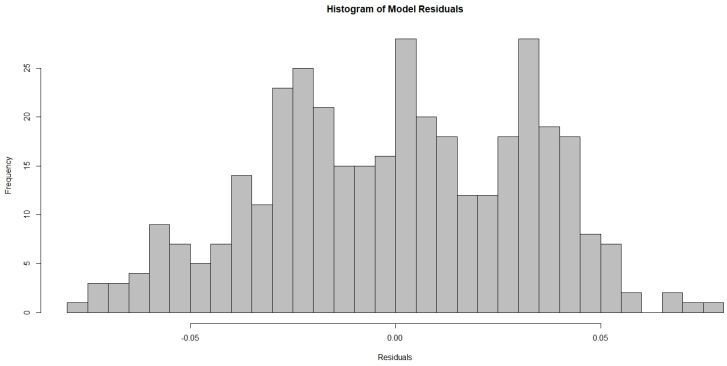
```
# GLM
> multi_model <- lm(my_data$nWBV ~ my_data$Age +
↪  my_data$EDUC)
```

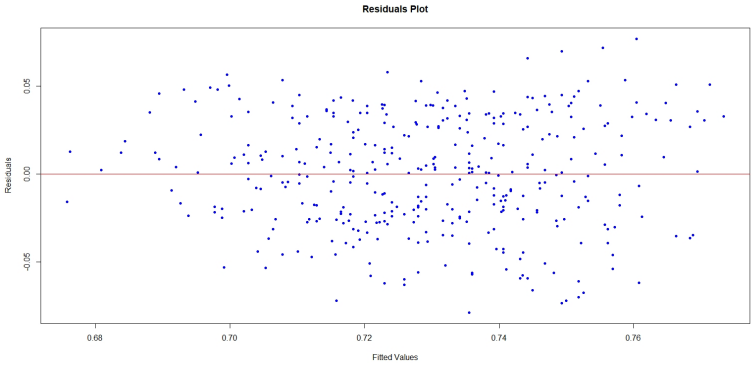
Next, we should check check our normality, equal variances and linearity assumptions.

```
# Linearity
> plot(my_data$EDUC, my_data$nWBV, xlab = 'Age (years)',
↪  ylab = 'Whole Brain Volume', main = 'Age VS Whole
↪  Brain Volume Plot', pch = 20 )

# Normality of Residuals
> hist(multi_model$residuals, main = 'Histogram of Model
↪  Residuals', xlab = 'Residuals', right = F, breaks =
↪  50, col = 'grey')

# Equal Variances
> plot(multi_model$fitted.values, multi_model$residuals,
↪  xlab = 'Fitted Values', ylab = 'Residuals', main =
↪  'Residuals Plot', pch = 20, col = 'blue' )
> abline(h = 0, col = 'red')
```

Interestingly, our dataset actually passes all three assumptions pretty well. Lets proceed with the linear model. I only checked the assumptions for one variable, however, in reality you should check the assumptions for all of the explanatory variables.







Finally, lets define the following:

Hypothesis 1:

H_0 : When controlling for age, years of education does not have any variation on the whole brain volume.

H_A : When controlling for age, years of education does explain some variation on the whole brain volume.

Hypothesis 2:

H_0 : When controlling for years of education, age does not have any variation on the whole brain volume.

H_A : When controlling for years of education, age does explain some variation on the whole brain volume.

Using the sum() function gives us the results of our GLM regressional model.

```
# Summary of Linear Model
> summary(multi_model)

Call:
lm(formula = my_data$nWBV ~ my_data$Age + my_data$EDUC)

Residuals:
      Min        1Q    Median        3Q       Max
-0.078543 -0.023551  0.001001  0.028258  0.076573

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9288881  0.0189080   49.13   <2e-16 ***
my_data$Age  -0.0025228  0.0002160  -11.68   <2e-16 ***
my_data$EDUC -0.0003444  0.0005739   -0.60    0.549
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
↪  ' 1

Residual standard error: 0.03183 on 370 degrees of freedom
Multiple R-squared:  0.2694,        Adjusted R-squared:
↪  0.2655
F-statistic: 68.22 on 2 and 370 DF,  p-value: < 2.2e-16
```

While holding age constant, the amount of education years does not explain any significant variation of whole brain volume.

However, while holding the education variable constant, age does explain a significant variation on whole brain volume. On average, for every one year increase in age, the whole brain volume *decreases* by 0.0025 units, while holding years in education the same.

Because there exists more than one explanatory variables in our model, we will use the *Adjusted R-squared* value to explain how well our model fits the data. In particular, our model explains 26.94% of the variation in whole brain volume.

General Linear Models with Interactions

Similar to a multi-factor ANOVA Model, we can test if the effects of one response variable differs based on an interaction. Adding another set of hypotheses gives us the following:

Hypothesis 3:

H_0 : There is no interaction between age and education on whole brain volume.

H_A : There is an interaction between age and education on whole brain volume.

Before running the summary() function and our GLM, we need to mean-center our numerical explanatory variable to make the results more interpretable. [16]

[16] In order to center a variable at the mean, we take the difference between the mean and every value in the dataset

```
# Centered Age
> my_data$Age_cent <- my_data$Age - mean(my_data$Age)
# Centered Education
> my_data$EDUC_cent <- my_data$EDUC - mean(my_data$EDUC)

# General Linear Model with Interaction
> multi_model_int <- lm(my_data$nWBV ~ my_data$Age_cent *
↪  my_data$EDUC_cent)
> summary(multi_model_int)

Call:
lm(formula = my_data$nWBV ~ my_data$Age_cent *
↪  my_data$EDUC_cent)

Residuals:
      Min        1Q    Median        3Q       Max
-0.078921 -0.022796  0.001086  0.027850  0.075729

Coefficients:
                                    Estimate Std. Error t
                                    ↪  value Pr(>|t|)
(Intercept)                        7.296e-01  1.650e-03
↪  442.088   <2e-16 ***
my_data$Age_cent                  -2.532e-03  2.172e-04
↪  -11.654   <2e-16 ***
my_data$EDUC_cent                 -4.117e-04  5.947e-04
↪  -0.692    0.489
```

```
my_data$Age_cent:my_data$EDUC_cent  3.181e-05  7.256e-05
↪  0.438    0.661
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
↪  ' 1

Residual standard error: 0.03186 on 369 degrees of freedom
Multiple R-squared:  0.2698,        Adjusted R-squared:
↪  0.2639
F-statistic: 45.44 on 3 and 369 DF,  p-value: < 2.2e-16
```

Overall, we found no significant interaction between age and years of education on whole brain volume. Ergo, the effects of age on whole brain volume is the same regardless on how one's educational history. In fact, for every one year increase in age, whole brain volume decreases, on average, a total of 0.002532 units. Our model again only explains 26.4% of the variation in whole brain volume, not a lot!

Index