

SEMINAR OUTLINE

Data Analysis Overview

Introduction to R Studio

Statistics, Data and Variable

Data Analysis and Reporting with R

Visualization Techniques with R

Conclusion

2021 Seminar for Final Year Students

Data Analysis and Reporting

Olakunle Joshua (olakunle4impact@yahoo.com (<mailto:olakunle4impact@yahoo.com>))

3/6/2021

SEMINAR OUTLINE

- Data Analysis Overview
- Introduction to R Studio
- Statistics, Data and Variable
- Analysis and Reporting with R
- Visualization Techniques with R
- Conclusion

Data Analysis Overview

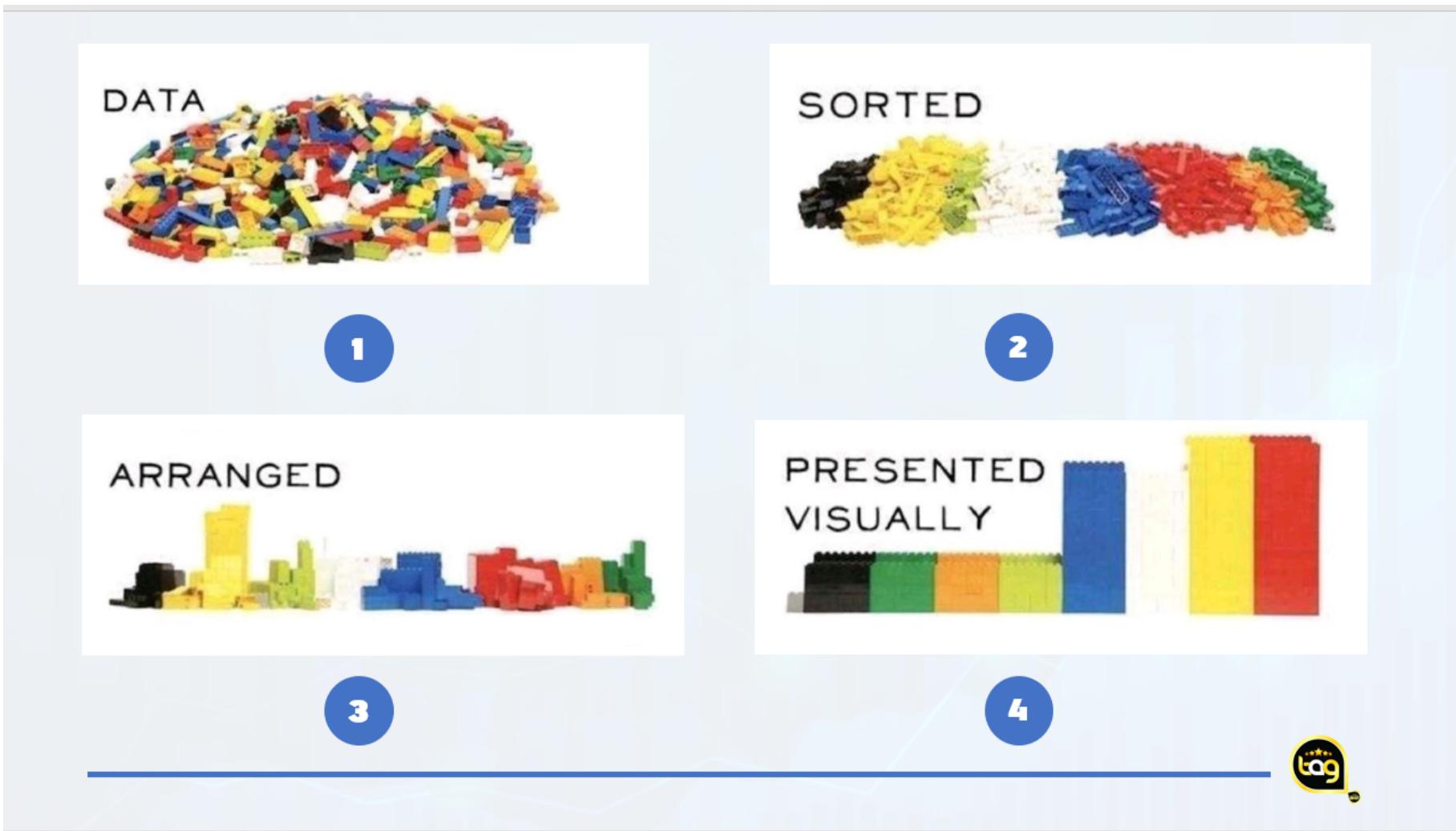


Figure 1: Simple Analogy to describe Data Analysis

The above figure has completely simplified the task we have for today. Looking at the figure very well, we will notice that everything revolves round the very important word that we will first consider in today's seminar. The word **DATA**.

- **Defining a Data**

Data is a common word for every analyst but what exactly does it mean. It is a **collection of facts**, such as numbers, words, measurements, observations or just descriptions of things.

- **Data Collection**

Getting some of these information has become easy in our modern world with the advent of smartphones and computers that has ability to do myriads of things such as gathering data on emails, sales, heartbeat, distance covered, tree heights etc

- **What then is Data Analysis?**

With respect to the figure above, **data analytics** is the analysis of data, whether huge or small, in order to understand it and see how to use the knowledge hidden within it.

Data analysis is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making.

A simple example I used often in most of my trainings is on recorded information on the depth of a river and the possible knowledge to derive from it for swimmers.

Another example is what happens when we study. Consciously or unconsciously, we feed our system with data, that is the content of any materials or textbooks we are reading at that moment and by default our brain process this information, refine and it gradually begin to guide our actions and decisions.

In the next topic, we will learn more on R Studio, how to generate random data in R and finally, how to import and export data in R. Let's go friends.

Introduction to R Studio

There are many data analysis tools that can be used in analysing data with the aim of deriving useful insight from it. Some of which are: SPSS, Excel, EasyFit, Stata, PSPP, Octave, Matlab, Minitab, Statistica, SAS, Tableau, Python etc.

For the purpose of this seminar, we will be sticking close to the use of R programming software. This is because of the following advantages;

- Flexible, easy to learn and friendly graphical capabilities.
- Effective data storage facilities.
- Large number of free packages available for data analysis. We can also build ours.
- Provides all the capabilities of a programming language.
- Supports getting data from different types of sources.
- Interesting of all, it is **FREE**. Isn't this amazing?

R Studio is an Integrated Development Environment (**IDE**) for the base R. Let us have a look at a typical R Studio environment as shown below;

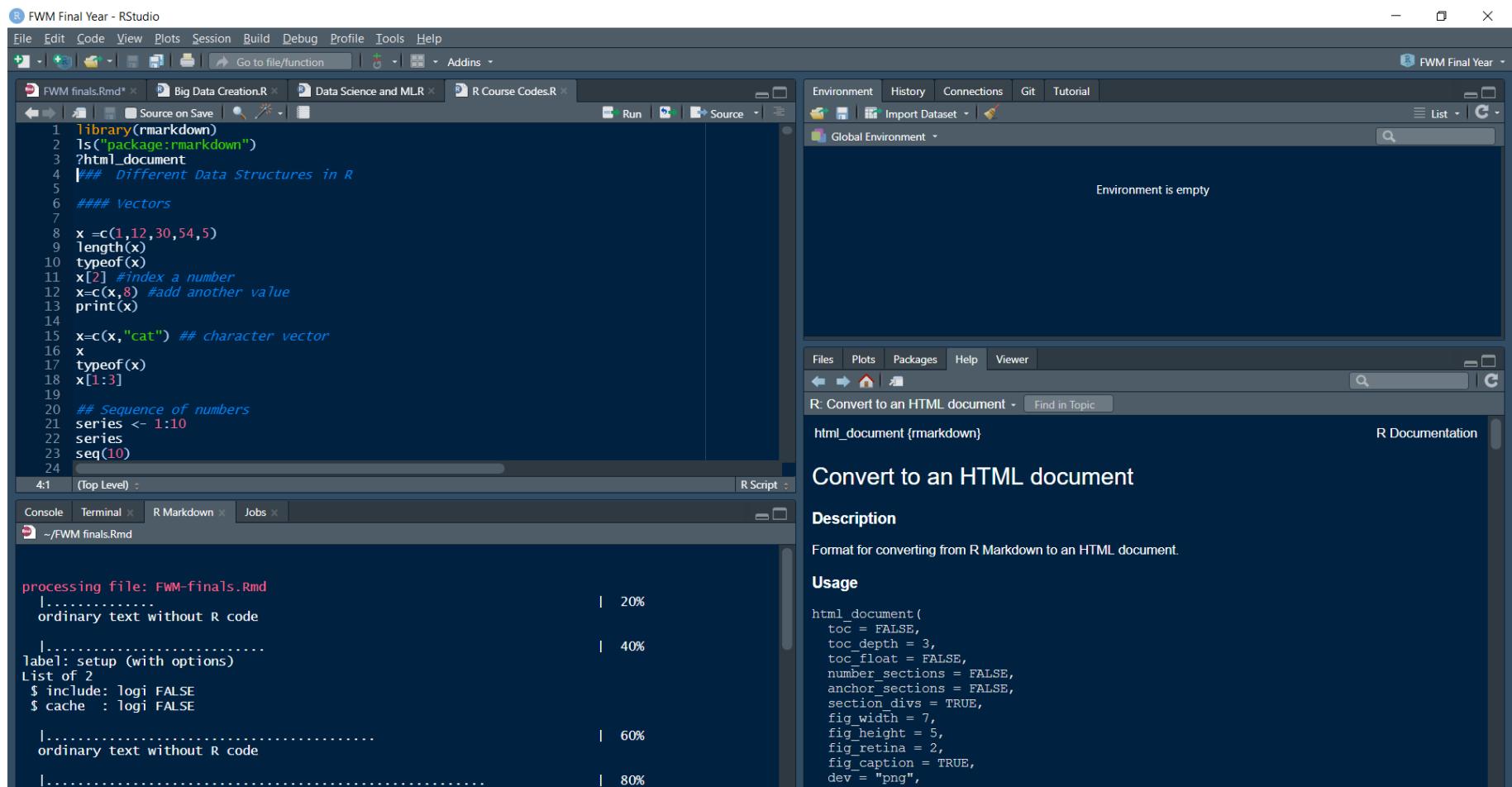


Figure 2: A Typical R Studio Environment

It is very important I talk on the concept of appearance as evident in the background colour I used in Figure 2 which is far different from the default white background colour on your system as a new R User. With some few tricks around the tools, this can be easily changed to your desired colour.

The environment is such a friendly one with a default three (3) panes when the software is first opened. The last pane, most times the editor pane where you write your codes and comments can be created using the file option on the menu bar. We will dwell more on this extensively when we get to session of data analysis with R by God's grace.

Installing R

R is freely available for all the major computing platforms such as macOS, Windows and LINUX.

Base R

- Visit the official R site at <https://cran.r-project.org> (<https://cran.r-project.org>)
- Select the operating system of your choice and then click on the download tab
- Run the installer and follow the instructions to successfully install the software.

After you successfully install the software, you should see something like what we have in the figure below;

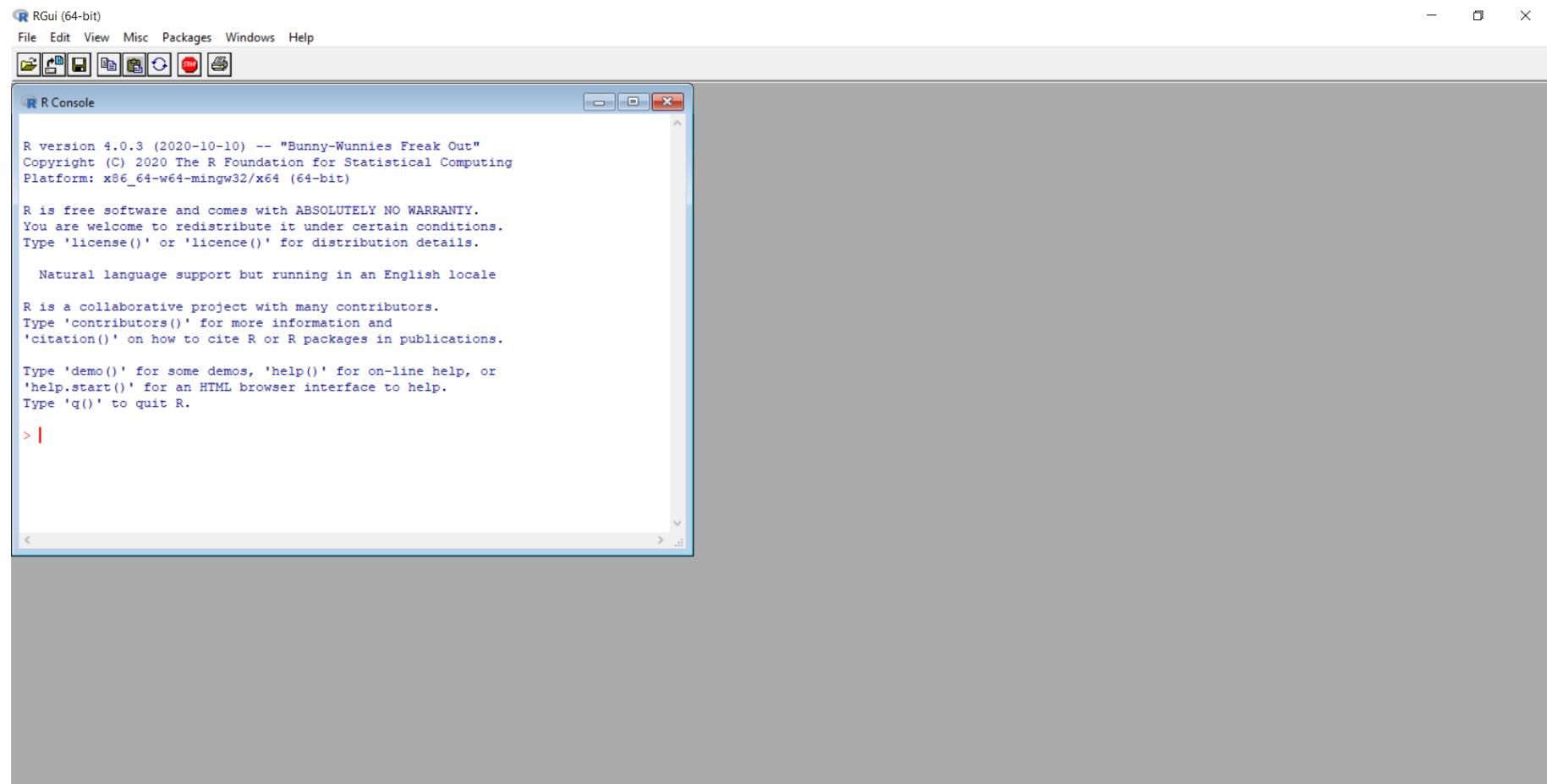


Figure 3: A Typical R Environment

NOTE: The base R is required for R studio to function effectively.

R Studio

- Visit the official R Studio website at <https://rstudio.com> (<https://rstudio.com>)
- Click the product option
- Select your server version or desktop version

- Select your OS and click on the download button
- Run the installer and follow the instruction

NOTE: If the base R is not first installed, you might be directed to a website to do this before the R Studio can work effectively.

Statistics, Data and Variable

- **Statistics**

Statistics is the practice of turning data into information to identify trends and understand features of populations. One of the most important keywords from the definition above is **Data and Information**.

This can be best seen with the concept of **Input** (raw) and **Output** (refined). The first thing every statistical analyst will always face in the course of performing data analysis is dealing with **raw data** - in other words, the records or observations that make up a sample. These data could be stored in a specialized **R object**.

- **Data**

At this section, we will look more into R Objects.

- **Data Types in R**



Data Types in R

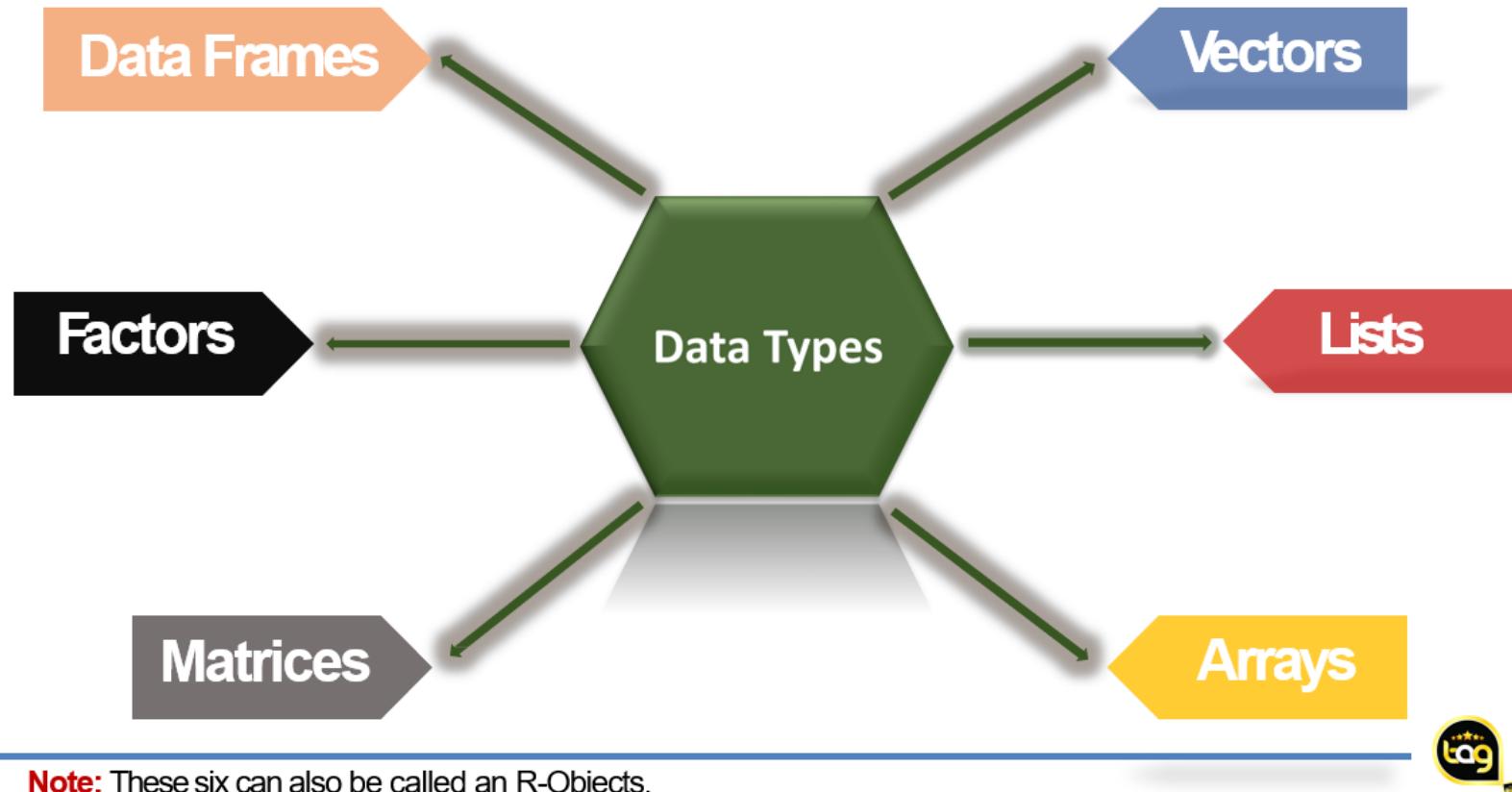


Figure 4: Data Types in R

- Vector

A vector is a sequence of data elements of the same basic type. When you want to create vector with more than one element, you should use `c()` function which means to **combine** the elements into a vector.

```
v <- c(1,2,3) # Hit the run button to the line of code or press Ctrl + Enter  
print(v)
```

```
[1] 1 2 3
```

```
class(v)
```

```
[1] "numeric"
```

```
ba <- c("FWM", "AQFM", "WMA")
print(ba)
```

```
[1] "FWM"   "AQFM"  "WMA"
```

```
class(ba)
```

```
[1] "character"
```

Basic Interpretation of information on the environment and console panes

```
## The concept of indexing
ba[2]
```

```
[1] "AQFM"
```

```
v[3]
```

```
[1] 3
```

```
## The > sign on the console means R is ready for a new command.  
## If it shows a + plus sign, it means the code is incomplete.
```

```
## The concept of replacement
```

```
ba[2] <- "EMT"
```

```
v[3] <- 5
```

```
print(ba)
```

```
[1] "FWM" "EMT" "WMA"
```

```
print(v)
```

```
[1] 1 2 5
```

```
## The concept of continuous replacement
```

```
v[1:2]
```

```
[1] 1 2
```

```
## Boolean or Logical Masking using comparison operators
```

```
v <- c(100,200,300,400)
```

```
v[v>200]
```

```
[1] 300 400
```

```
v[v>=200]
```

```
[1] 200 300 400
```

```
v[v<200]
```

```
[1] 100
```

```
v[v<=200]
```

```
[1] 100 200
```

```
my.true <- v>=300  
v[my.true] ## It means from v, index out the true values of the object my.true. This is more like filtering
```

```
[1] 300 400
```

```
## The concept of sorting  
basal_area <- c(30,25,12,37,40)  
sorted_ba <- sort(basal_area)  
print(sorted_ba)
```

```
[1] 12 25 30 37 40
```

```
desc_ba <- sort(basal_area, decreasing = TRUE )  
print(desc_ba)
```

```
[1] 40 37 30 25 12
```

```
## Adding another value  
basal_area <- c(basal_area, c(30,24,30))  
print(basal_area)
```

```
[1] 30 25 12 37 40 30 24 30
```

```
## Inbuilt function  
length(basal_area)
```

```
[1] 8
```

```
typeof(basal_area)
```

```
[1] "double"
```

```
## The concept of sequencing  
bl <- seq(5,7, by = 0.4)  
print(bl)
```

```
[1] 5.0 5.4 5.8 6.2 6.6 7.0
```

```
## Coercing a vector into different data types  
vt <- c(basal_area, "ba")  
print(vt)
```

```
[1] "30" "25" "12" "37" "40" "30" "24" "30" "ba"
```

```
length(vt)
```

```
[1] 9
```

```
class(vt)
```

```
[1] "character"
```

```
vt <- vt[-6]  
print(vt)
```

```
[1] "30" "25" "12" "37" "40" "24" "30" "ba"
```

```
class(vt)
```

```
[1] "character"
```

```
vt <- vt[-8]  
print(vt)
```

```
[1] "30" "25" "12" "37" "40" "24" "30"
```

```
length(vt)
```

```
[1] 7
```

```
class(vt)
```

```
[1] "character"
```

```
vt <- as.numeric(vt)  
length(vt)
```

```
[1] 7
```

```
class(vt)
```

```
[1] "numeric"
```

The simplest of the R-objects is the **vector object** and there are six data types of these atomic vectors, also termed as **six classes of vectors**. The other R-Objects are built upon the atomic vectors.

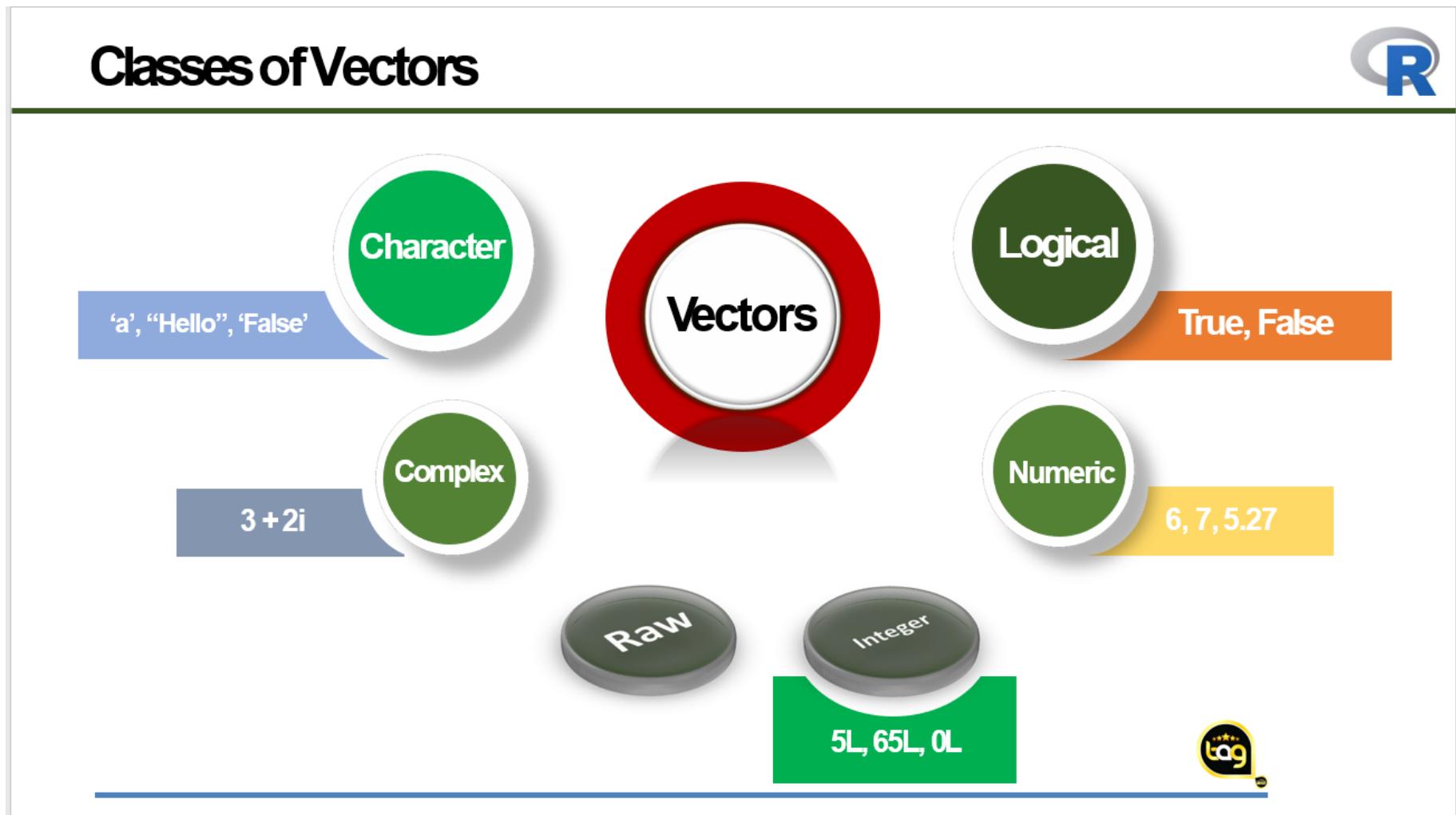


Figure 5: Six classes of vectors

- List

A list is an R-object which can contain many different types of elements inside it like vectors, string, numbers, functions and even another list inside it.

```
l <- list(2,"FWM",2+5i, FALSE, 5)
print(l)
```

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

```
[[2]]
[1] "FWM"
```

```
[[3]]
[1] 2+5i
```

```
[[4]]
[1] FALSE
```

```
[[5]]
[1] 5
```

```
class(l)
```

```
[1] "list"
```

```
l2 <- list(1,2,"FUNAAB")
class(l2)
```

```
[1] "list"
```

```
l3 <- c(l,l2)
print(l3)
```

```
[[1]]
```

```
[1] 2
```

```
[[2]]
```

```
[1] "FWM"
```

```
[[3]]
```

```
[1] 2+5i
```

```
[[4]]
```

```
[1] FALSE
```

```
[[5]]
```

```
[1] 5
```

```
[[6]]
```

```
[1] 1
```

```
[[7]]
```

```
[1] 2
```

```
[[8]]
```

```
[1] "FUNAAB"
```

```
class(13)
```

```
[1] "list"
```

```
13[4]
```

```
[[1]]
```

```
[1] FALSE
```

- **Arrays**

An arrays are the R Data Objects which can store data in more than two dimensions. It takes vectors as input and uses the value in the dim parameter to create an array.

```
a<-array(c(2,3,1),dim=c(2,3,3)) ## Row, column and frequency  
print(a)
```

```
, , 1  
  
[,1] [,2] [,3]  
[1,] 2 1 3  
[2,] 3 2 1  
  
, , 2  
  
[,1] [,2] [,3]  
[1,] 2 1 3  
[2,] 3 2 1  
  
, , 3  
  
[,1] [,2] [,3]  
[1,] 2 1 3  
[2,] 3 2 1
```

Let us modify the code a little as shown below;

```
a<-array(c(2,3,1),dim=c(2,3,3,2))  
class(a)
```

```
[1] "array"
```

```
print(a)
```

, , 1, 1

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 2, 1

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 3, 1

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 1, 2

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 2, 2

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 3, 2

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

Compare the above with this code below

```
a<-array(c(2,3,1),dim=c(2,3,6))
print(a)
```

, , 1

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 2

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 3

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 4

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 5

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

, , 6

[,1] [,2] [,3]
[1,] 2 1 3
[2,] 3 2 1

- **Matrices**

They are similar to an array but only store data in two dimensions. Let us take a look at some brief examples.

```
vtr<-c(1,2,3)
vtr1<-c(4,5,6)
mtr<-matrix(c(vtr,vtr1),3,3)
print(mtr)
```

```
[,1] [,2] [,3]
[1,]    1    4    1
[2,]    2    5    2
[3,]    3    6    3
```

```
volume <- c(12,15,10,11,13,12)
cv <- matrix(volume, 3,2)
class(cv)
```

```
[1] "matrix" "array"
```

```
print(cv)
```

```
[,1] [,2]
[1,]   12   11
[2,]   15   13
[3,]   10   12
```

cbind and the rbind function

As the names implies, cbind means column binding while rbind means row binding. This approach is used when combining multiple vectors together to form a matrix or an array of two or more dimensions.

```
Girth <- c(1,2,3,4,5)
Height <- c(6,7,8,9,10)
Volume <- Height/2

tree_data <- c(Girth, Height, Volume) ## vectorising
print(tree_data)
```

```
[1] 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 3.0 3.5 4.0 4.5 5.0
```

```
tree_data <- rbind(Girth, Height, Volume)
print(tree_data)
```

```
[,1] [,2] [,3] [,4] [,5]
Girth     1   2.0    3   4.0    5
Height    6   7.0    8   9.0   10
Volume    3   3.5    4   4.5    5
```

```
## Adding Column Names
locations <- c("loc1","loc2","loc3","loc4","loc5")
colnames(tree_data) <- locations
print(tree_data)
```

```
loc1 loc2 loc3 loc4 loc5
Girth     1   2.0    3   4.0    5
Height    6   7.0    8   9.0   10
Volume    3   3.5    4   4.5    5
```

```
## Chnaging row names
rownames(tree_data)[1] <- "Basal Area"
print(tree_data)
```

```
loc1 loc2 loc3 loc4 loc5
Basal Area    1  2.0    3  4.0    5
Height        6  7.0    8  9.0   10
Volume        3  3.5    4  4.5    5
```

```
rownames(tree_data)[3] <- "M Volume"
print(tree_data)
```

```
loc1 loc2 loc3 loc4 loc5
Basal Area    1  2.0    3  4.0    5
Height        6  7.0    8  9.0   10
M Volume      3  3.5    4  4.5    5
```

```
## Other useful inbuilt functions
colSums(tree_data)
```

```
loc1 loc2 loc3 loc4 loc5
10.0 12.5 15.0 17.5 20.0
```

```
rowSums(tree_data)
```

```
Basal Area      Height     M Volume
          15           40          20
```

```
colMeans(tree_data)
```

```
loc1      loc2      loc3      loc4      loc5
3.333333 4.166667 5.000000 5.833333 6.666667
```

```
rowMeans(tree_data)
```

```
Basal Area      Height      M Volume  
      3           8           4
```

```
nrow(tree_data)
```

```
[1] 3
```

```
ncol(tree_data)
```

```
[1] 5
```

```
## Indexing with matrices  
tree_data[2,3]
```

```
[1] 8
```

```
tree_data[1,] ## first role only
```

```
loc1 loc2 loc3 loc4 loc5  
    1    2    3    4    5
```

```
tree_data[1:2,] ## first two roles only
```

```
loc1 loc2 loc3 loc4 loc5  
Basal Area   1    2    3    4    5  
Height       6    7    8    9   10
```

```
tree_data[,1] ## first column only
```

Basal Area	Height	M Volume
1	6	3

```
tree_data[,1:2] ## first two columns only
```

	loc1	loc2
Basal Area	1	2.0
Height	6	7.0
M Volume	3	3.5

A matrix can easily be coerced to form a data frame.

- Factors

Factors are the data objects which are used to categorize the data and store it as levels. They are useful in **data analysis for statistical modelling**. Factors are created using the **factor()** function. The **nlevels** functions gives the count of levels. Let us work around some examples.

```
species <- c("gmelina", "tectona", "gmelina", "azadiracta", "mangifera")
factor(species)
```

```
[1] gmelina    tectona    gmelina    azadiracta mangifera
Levels: azadiracta gmelina mangifera tectona
```

```
nlevels(factor(species))
```

```
[1] 4
```

```
levels(factor(species))
```

```
[1] "azadiracta" "gmelina"    "mangifera"   "tectona"
```

```
### Ordinal categorical variable
```

```
temp <- c("cold","hot","med","hot","hot", "cold")
factor(temp)
```

```
[1] cold hot med hot hot cold
Levels: cold hot med
```

```
temp_R <- factor(temp, ordered = T, levels = c("cold","med","hot"))
temp_R
```

```
[1] cold hot med hot hot cold
Levels: cold < med < hot
```

```
class(temp_R)
```

```
[1] "ordered" "factor"
```

```
nlevels(temp_R)
```

```
[1] 3
```

```
summary(temp_R)
```

```
cold med hot
 2   1   3
```

- Data Frames

Data frames are tabular data objects created with the **data.frame()**. Unlike a matrix in data frame each column can contain different modes of data. It is one of the favourite way of creating tables in R.

```
SN <- c(1:5)
Gender <- c("Male","Male","Female","Male","Female")
Names <- c("Olakunle","Ademola","Tolu","Samson","Lizzy")
Age <- c(24,22,20,28,21)
Class2016 <- data.frame(SN,Gender,Names,Age)
print(Class2016)
```

	SN	Gender	Names	Age
1	1	Male	Olakunle	24
2	2	Male	Ademola	22
3	3	Female	Tolu	20
4	4	Male	Samson	28
5	5	Female	Lizzy	21

```
## Same way as above
tree <- data.frame(Girth=c(2,5,4), Height=c(10,7,11), Volume=c(0.34,.43,.56))
class(tree)
```

[1] "data.frame"

print(tree)

	Girth	Height	Volume
1	2	10	0.34
2	5	7	0.43
3	4	11	0.56

```
## Assessing columns of a data frame
tree$Girth ## The girth column alone
```

[1] 2 5 4

```
tree[1,2] ## first height observation
```

```
[1] 10
```

```
## Getting the names of columns in a data frame  
names(tree)
```

```
[1] "Girth" "Height" "Volume"
```

```
## structure of a data frame  
str(tree)
```

```
'data.frame': 3 obs. of 3 variables:  
 $ Girth : num 2 5 4  
 $ Height: num 10 7 11  
 $ Volume: num 0.34 0.43 0.56
```

```
## summary function on a data frame  
summary(tree)
```

Girth	Height	Volume
Min. :2.000	Min. : 7.000	Min. :0.3400
1st Qu.:3.000	1st Qu.: 8.500	1st Qu.:0.3850
Median :4.000	Median :10.000	Median :0.4300
Mean :3.667	Mean : 9.333	Mean :0.4433
3rd Qu.:4.500	3rd Qu.:10.500	3rd Qu.:0.4950
Max. :5.000	Max. :11.000	Max. :0.5600

```
## Free R datasets for practice
data()
data(package="datasets")
library(help="datasets")
data("ChickWeight")
str(ChickWeight)
```

```
Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame': 578 obs. of 4 variables:
 $ weight: num 42 51 59 64 76 93 106 125 149 171 ...
 $ Time   : num 0 2 4 6 8 10 12 14 16 18 ...
 $ Chick   : Ord.factor w/ 50 levels "18"<"16"<"15"<...: 15 15 15 15 15 15 15 15 15 15 ...
 $ Diet    : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "formula")=Class 'formula' language weight ~ Time | Chick
 .. . . . attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "outer")=Class 'formula' language ~Diet
 .. . . . attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "labels")=List of 2
 ..$ x: chr "Time"
 ..$ y: chr "Body weight"
 - attr(*, "units")=List of 2
 ..$ x: chr "(days)"
 ..$ y: chr "(gm)"
```

```
summary(ChickWeight)
```

weight	Time	Chick	Diet
Min. : 35.0	Min. : 0.00	13 : 12	1:220
1st Qu.: 63.0	1st Qu.: 4.00	9 : 12	2:120
Median :103.0	Median :10.00	20 : 12	3:120
Mean :121.8	Mean :10.72	10 : 12	4:118
3rd Qu.:163.8	3rd Qu.:16.00	17 : 12	
Max. :373.0	Max. :21.00	19 : 12	
		(Other):506	

```
head(ChickWeight)
```

	weight	Time	Chick	Diet
1	42	0	1	1
2	51	2	1	1
3	59	4	1	1
4	64	6	1	1
5	76	8	1	1
6	93	10	1	1

```
tail(ChickWeight)
```

	weight	Time	Chick	Diet
573	155	12	50	4
574	175	14	50	4
575	205	16	50	4
576	234	18	50	4
577	264	20	50	4
578	264	21	50	4

- Data Operators in R



Data Operators in R

1

Arithmetic Operators

2

Assignment Operators

3

Relational Operators

4

Logical Operators

5

Special Operators

- An **operator** is a symbol that tells the compiler to perform specific mathematical or logical manipulations.



Figure 6: Data Operators in R

- **Arithmetic Operator**

This operator gives us the privilege of using R as a basic calculator. Briefly, we will take some examples performing some basic calculations with this operator.

```
2+5
```

```
[1] 7
```

```
vec1 <- c(1,2,3)
vec2 <- c(4,5,6)

print(vec1 + vec2) ## Addition
```

```
[1] 5 7 9
```

```
print(vec1 - vec2) ## Subtraction
```

```
[1] -3 -3 -3
```

```
print(vec1 * vec2) ## Multiplication
```

```
[1] 4 10 18
```

```
print(vec2 / vec1) ## Division
```

```
[1] 4.0 2.5 2.0
```

```
print(vec2 ^ vec1) ## Power
```

```
[1] 4 25 216
```

```
print(vec2 ** vec1) ## Similarity with python power.
```

```
[1] 4 25 216
```

```
print(vec2 %% vec1) ## Modular division (Remainder)
```

```
[1] 0 1 0
```

```
print(vec2 %/% vec1) ## Flow division (Whole number)
```

```
[1] 4 2 2
```

- **Assignment Operator**

During the section on data types, in one of the examples, we have used the assignment operator to create a variable. Therefore, we will only do little of variable creation using the different types of assignment operators available in R.

```
vec1 <- c(1,2,3) ## Less than assignment operator  
vec1 <<- c(1,2,3)  
c(4,5,6) -> vec2 ## greater than assignment operator  
c(4,5,6) ->> vec2  
vec1 = c(1,2,3) ## equal to assignment operator
```

- **Relational Operators**

```
vec1 < vec2
```

```
[1] TRUE TRUE TRUE
```

```
vec1 > vec2
```

```
[1] FALSE FALSE FALSE
```

```
vec1 <= vec2
```

```
[1] TRUE TRUE TRUE
```

```
vec1 >= vec2
```

```
[1] FALSE FALSE FALSE
```

```
vec2 == vec2
```

```
[1] TRUE TRUE TRUE
```

```
vec1 != vec2
```

```
[1] TRUE TRUE TRUE
```

- **Logical Operators**

AND

```
2&3 ## prints TRUE only if both sides are TRUE
```

```
[1] TRUE
```

```
0&1
```

```
[1] FALSE
```

```
TRUE & TRUE
```

```
[1] TRUE
```

```
TRUE & FALSE
```

[1] FALSE

FALSE && FALSE

[1] FALSE

TRUE && TRUE

[1] TRUE

TRUE && FALSE

[1] FALSE

OR

2|3 ## prints TRUE as Long one of the sides is TRUE

[1] TRUE

0|1

[1] TRUE

TRUE | TRUE

[1] TRUE

TRUE | FALSE

[1] TRUE

FALSE || FALSE

[1] FALSE

TRUE || TRUE

[1] TRUE

TRUE || FALSE

[1] TRUE

Opposite

!0 ## It gives the opposite logical values

[1] TRUE

!6

[1] FALSE

!TRUE

```
[1] FALSE
```

```
!FALSE
```

```
[1] TRUE
```

- **Special Operators**

:

It creates the series of numbers in sequence for a vector.

```
v <- c(1:15)  
class(v)
```

```
[1] "integer"
```

%in%

This operator is used to identify if an element belongs to a vector.

```
vec2 %in% v
```

```
[1] TRUE TRUE TRUE
```

****%*%****

This operator is used to multiply a matrix with its transpose.

```
M = matrix( c(2,6,5,1,10,4),  
nrow=2,ncol=3,byrow = TRUE)
```

```
t = M %*% t(M)  
print(t)
```

```
[,1] [,2]
[1,] 65 82
[2,] 82 117
```

General View on Variable A variable is a characteristics of an individual in a population, the value of which can differ between entities within that population.

- **Types of Variable**

Numeric Variable: This is one whose observations are naturally recorded as numbers. They are of two types: Continuous and Discrete.

- **Continuous:** They can be recorded as any value in some interval, up to any number of decimal. e.g 15.12mm.
- **Discrete:** It takes only distinct numeric values – and if the range is restricted, then the numbers of possible values is finite. For example, it doesn't make sense to observe 15.245 heads if you are to observe the number of heads in a 20 flip of a coin.

Categorical Variable: Like some discrete variables, categorical variables takes only one of a finite number of possibilities. Unlike discrete variables, however, categorical observations are not always recorded as numeric values.

There are two types of categorical variables. They are:

- **Nominal:** Those that cannot be logically ranked. e.g sex. Sex has two categories. The order of these categories is irrelevant.

```
head(chickwts$feed)
```

```
[1] horsebean horsebean horsebean horsebean horsebean horsebean
Levels: casein horsebean linseed meatmeal soybean sunflower
```

- **Ordinal:** They are variables that can be logically ranked. e.g drug dosage with possible values of low, medium and high. The values can be ordered in either increasing or decreasing amount depending on the nature of the research.
- **Variables in R**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable, you reserve a space in the memory.

- **Finding Variables in R**

To know all the variables currently available in the workspace we use the **ls()** function. Also the **ls()** function can use patterns to match the variable names. Run the command **print(ls())**

- **Removing Variables in R**

print(ls(pattern="var")) this command will produce variables starting with the pattern "var".

Variables can be deleted by using the **rm()** function.

Data Analysis and Reporting with R

In this section, it will be more of practical on the procedure to deriving meaningful insight from your data with R.

Getting and Setting Working Directory

This is an important step we need to take before conducting any kind of analysis with R. What we will be doing here is knowing the exact folder on the system that our software is connected to. It is important we do this because our software will only be able to read in data from that folder and also write out to that same folder when necessary.

There are two ways to doing this. Either **manually** with the help of some inbuilt functions, or through **Session** column on the menu bar.

- **Manually**

```
getwd() ## To get working directory
```

```
[1] "C:/Users/HP Envy X360/OneDrive/Documents/FWM Final Year"
```

```
## setwd(dir= "C:/Users/HP Envy X360/OneDrive/Documents/FWM Final Year/NEW FOLDER")
## The forward slash is used to direct from one folder to the other.
```

Importing and Exporting of Data in R

- **.CSV**

```
dat <- read.csv("StudentsPerformance.csv", header = T)
head(dat[-1]) ## excluding the first column
```

	race.ethnicity	parental.level.of.education	lunch	
1	group B	bachelor's degree	standard	
2	group C	some college	standard	
3	group B	master's degree	standard	
4	group A	associate's degree	free/reduced	
5	group C	some college	standard	
6	group B	associate's degree	standard	
	test.preparation.course	math.score	reading.score	writing.score
1	none	72	72	74
2	completed	69	90	88
3	none	90	95	93
4	none	47	57	44
5	none	76	78	75
6	none	71	83	78

```
str(dat)
```

```
'data.frame': 1000 obs. of 8 variables:
 $ gender                  : chr "female" "female" "female" "male" ...
 $ race.ethnicity          : chr "group B" "group C" "group B" "group A" ...
 $ parental.level.of.education: chr "bachelor's degree" "some college" "master's degree" "associate's degree" ...
 $ lunch                    : chr "standard" "standard" "standard" "free/reduced" ...
 $ test.preparation.course : chr "none" "completed" "none" "none" ...
 $ math.score                : int 72 69 90 47 76 71 88 40 64 38 ...
 $ reading.score            : int 72 90 95 57 78 83 95 43 64 60 ...
 $ writing.score             : int 74 88 93 44 75 78 92 39 67 50 ...
```

- .txt

```
## dat1 <- read.table("table.txt", header = T)
```

- Excel Files

```
library(readxl)
## example <- read_excel("example.xls")
## head(example)
```

```
## excel_sheets("UnderG data.xlsx") ## To get the name of sheets
## each sheet is more like a data frame.

## read_excel("UnderG data.xlsx", sheet = "Sheet2")
```

NOTE: An easy way to import datasets is via the Global Environment or through the file pane.

- Using RCurl to read in csv data hosted online or on github

```
library(RCurl)
```

```
## dat2 <- read.csv(text = getURL("https://raw.githubusercontent.com/sciruela/Happiness-Salaries/ac87cac0e1b440826
eaa765e1b2c3e6bc264dad1/data.csv"))

## head(dat2)
```

Indexing and Subsetting Data Frames

```
data(iris)
str(iris)
```

```
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(iris)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width  
Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100  
1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300  
Median :5.800 Median :3.000 Median :4.350 Median :1.300  
Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199  
3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800  
Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500  
  
Species  
setosa :50  
versicolor:50  
virginica :50
```

```
summary(iris$Species)
```

```
setosa versicolor virginica  
50 50 50
```

```
head(iris)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
1 5.1 3.5 1.4 0.2 setosa  
2 4.9 3.0 1.4 0.2 setosa  
3 4.7 3.2 1.3 0.2 setosa  
4 4.6 3.1 1.5 0.2 setosa  
5 5.0 3.6 1.4 0.2 setosa  
6 5.4 3.9 1.7 0.4 setosa
```

```
head(iris,10)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

```
df3 <- iris[1:6,] ## To subset or isolate the first 6 rows.  
head(df3)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

```
df4 <- iris[,1:2] ## subset the first two columns  
head(df4)
```

	Sepal.Length	Sepal.Width
1	5.1	3.5
2	4.9	3.0
3	4.7	3.2
4	4.6	3.1
5	5.0	3.6
6	5.4	3.9

```
y <- iris[, c("Sepal.Length", "Sepal.Width")]
head(y)
```

	Sepal.Length	Sepal.Width
1	5.1	3.5
2	4.9	3.0
3	4.7	3.2
4	4.6	3.1
5	5.0	3.6
6	5.4	3.9

Indexing with relational operators

```
head(mtcars[mtcars$cyl==6 & mtcars$am==1,])
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Ferrari Dino	19.7	6	145	175	3.62	2.770	15.50	0	1	5	6

```
head(mtcars[mtcars$cyl==6 | mtcars$am==1,])
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4

```
head(mtcars[mtcars$mpg>20 & mtcars$cyl==6 & mtcars$gear==3, ])
```

```
mpg cyl disp hp drat    wt  qsec vs am gear carb
Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44  1  0    3    1
```

```
head(mtcars[mtcars$mpg>20 | mtcars$cyl==6 | mtcars$gear==3, ])
```

```
mpg cyl disp hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6 160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6 160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4 108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360 175 3.15 3.440 17.02  0  0    3    2
Valiant        18.1   6 225 105 2.76 3.460 20.22  1  0    3    1
```

```
head(mtcars[ (mtcars$mpg>20) & (mtcars$hp>100), c('mpg','cyl')])
```

```
mpg cyl
Mazda RX4      21.0   6
Mazda RX4 Wag  21.0   6
Hornet 4 Drive 21.4   6
Lotus Europa   30.4   4
Volvo 142E     21.4   4
```

```
vit <- c("Sepal.Length", "Petal.Length", "Species")
df5 <- iris[vit]
head(df5)
```

```
Sepal.Length Petal.Length Species
1          5.1         1.4  setosa
2          4.9         1.4  setosa
3          4.7         1.3  setosa
4          4.6         1.5  setosa
5          5.0         1.4  setosa
6          5.4         1.7  setosa
```

```
vit1 <- names(iris)%in% c("Species")
vit1
```

```
[1] FALSE FALSE FALSE FALSE  TRUE
```

```
df6 <- iris[!vit1]
head(df6)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5         1.4         0.2
2          4.9         3.0         1.4         0.2
3          4.7         3.2         1.3         0.2
4          4.6         3.1         1.5         0.2
5          5.0         3.6         1.4         0.2
6          5.4         3.9         1.7         0.4
```

Indexing with the subset function

```
dfset <- subset(iris, iris$Species=="setosa")
head(dfset)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
head(subset(mtcars, mpg>20 & cyl==6, c("mpg","cyl","disp")))
```

```
      mpg cyl disp
Mazda RX4     21.0   6 160
Mazda RX4 Wag 21.0   6 160
Hornet 4 Drive 21.4   6 258
```

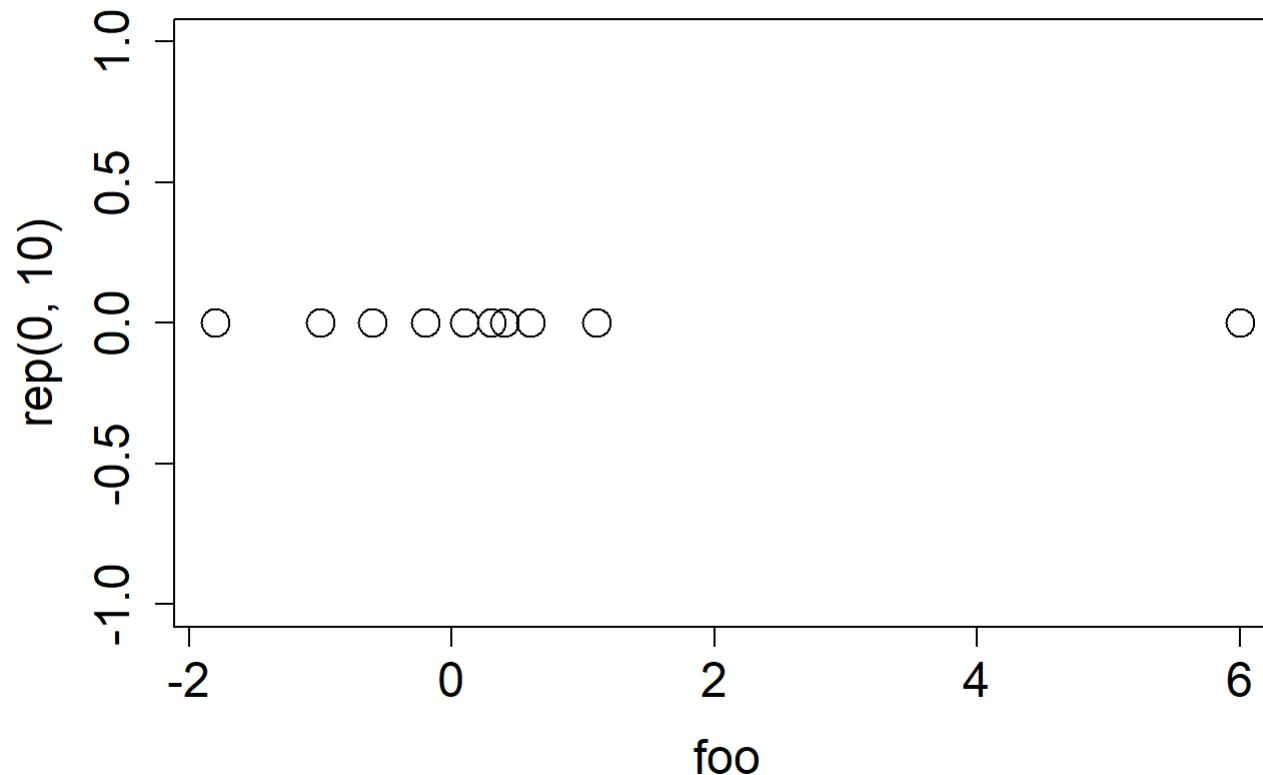
Exploratory Data Analysis (EDA)

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

- **Univariate** When discussing or analyzing data related to only one dimension, you are dealing with Univariate Data.

For example, the weight variable in the earlier example (**chickwts**) is univariate since each measurement can be expressed with one component – a single number.

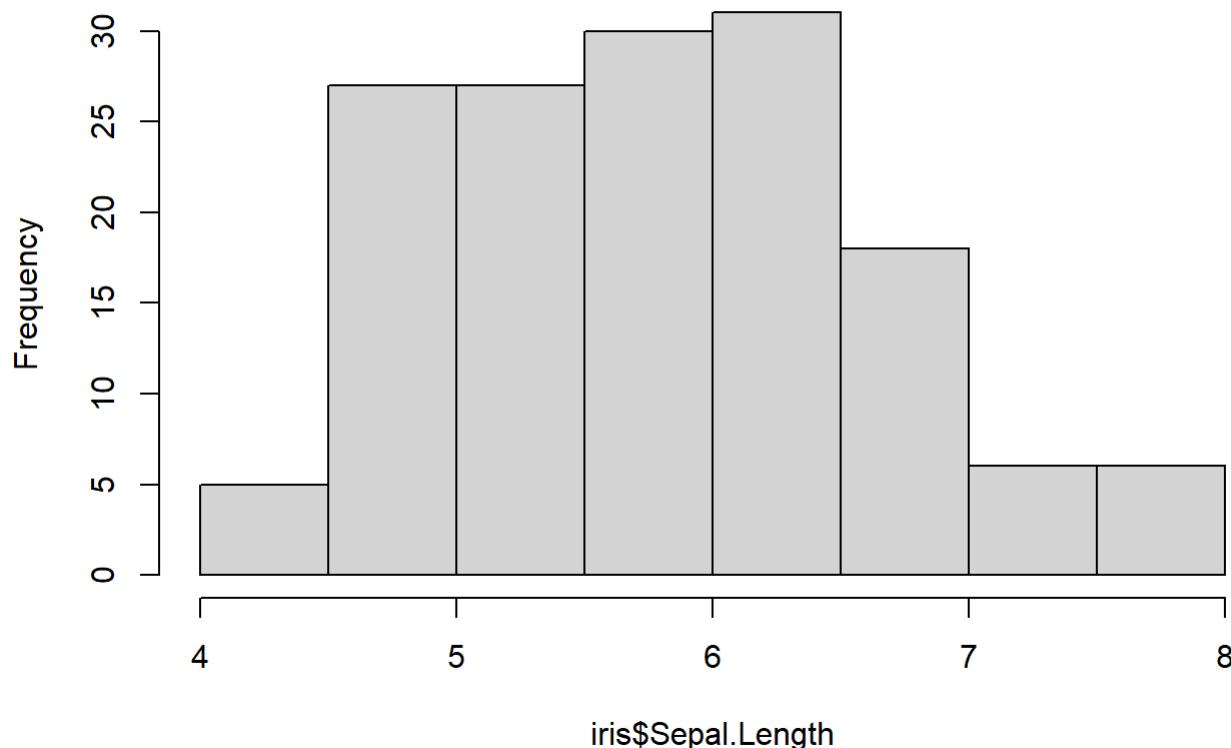
```
foo <- c(0.6,-0.6,0.1,-0.2,-1.0,0.4,0.3,-1.8,1.1,6.0)
plot(foo,rep(0,10),cex=2,cex.axis=1.5,cex.lab=1.5)
```



From the above EDA example with a univariate data, we were able to see an abnormal observation in the data popular called an **outlier**.

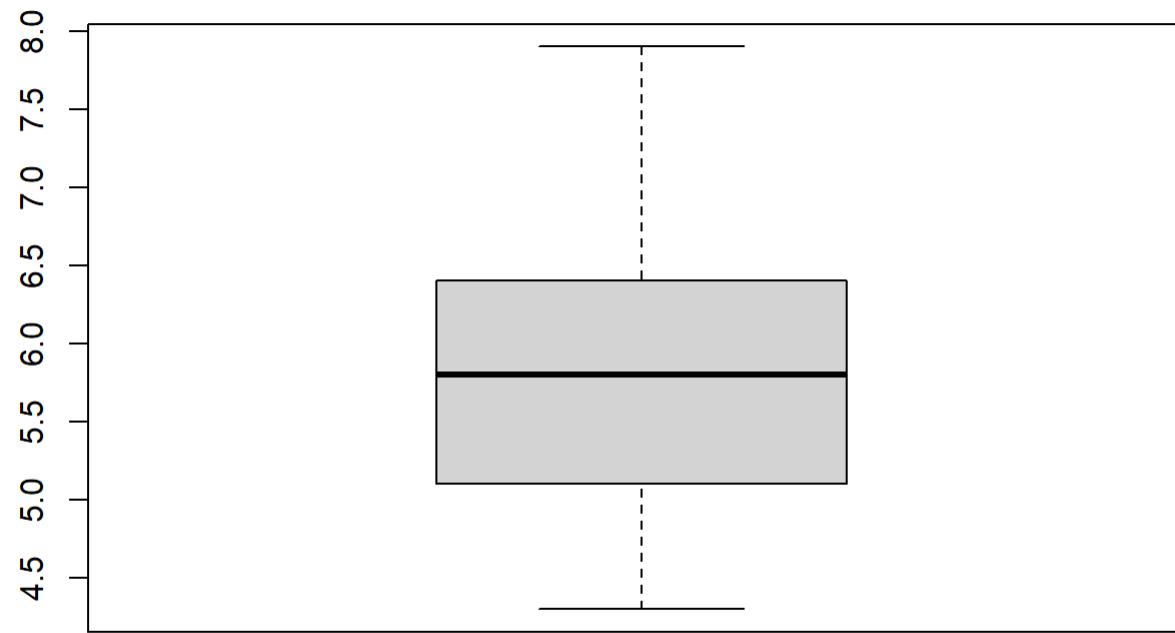
```
hist(iris$Sepal.Length)
```

Histogram of iris\$Sepal.Length



```
boxplot(iris$Sepal.Length, main="Summary of Iris", xlab="Sepal Length")
```

Summary of Iris



Sepal Length

```
summary(iris)
```

```
  Sepal.Length   Sepal.Width   Petal.Length   Petal.Width  
  Min.    :4.300  Min.    :2.000  Min.    :1.000  Min.    :0.100  
  1st Qu.:5.100  1st Qu.:2.800  1st Qu.:1.600  1st Qu.:0.300  
  Median  :5.800  Median  :3.000  Median  :4.350  Median  :1.300  
  Mean    :5.843  Mean    :3.057  Mean    :3.758  Mean    :1.199  
  3rd Qu.:6.400  3rd Qu.:3.300  3rd Qu.:5.100  3rd Qu.:1.800  
  Max.    :7.900  Max.    :4.400  Max.    :6.900  Max.    :2.500  
  
  Species  
  setosa    :50  
  versicolor:50  
  virginica :50
```

```
str(iris)
```

```
'data.frame': 150 obs. of 5 variables:  
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

Multivariate Data

When it is necessary to consider data with respect to variables that exists in more than one dimension (in other words, with more than one component or measurement associated with each observation), your data are considered Multivariate.

```
head(quakes)
```

```
lat long depth mag stations
1 -20.42 181.62 562 4.8      41
2 -20.62 181.03 650 4.2      15
3 -26.00 184.10 42 5.4      43
4 -17.97 181.66 626 4.1      19
5 -20.42 181.96 649 4.0      11
6 -19.68 184.31 195 4.0      12
```

```
summary(quakes)
```

```
lat           long          depth         mag
Min.   :-38.59  Min.   :165.7  Min.   : 40.0  Min.   :4.00
1st Qu.:-23.47 1st Qu.:179.6  1st Qu.: 99.0  1st Qu.:4.30
Median :-20.30  Median :181.4  Median :247.0  Median :4.60
Mean   :-20.64  Mean   :179.5  Mean   :311.4  Mean   :4.62
3rd Qu.:-17.64 3rd Qu.:183.2  3rd Qu.:543.0  3rd Qu.:4.90
Max.   :-10.72  Max.   :188.1  Max.   :680.0  Max.   :6.40

stations
Min.   : 10.00
1st Qu.: 18.00
Median : 27.00
Mean   : 33.42
3rd Qu.: 42.00
Max.   :132.00
```

```
str(quakes)
```

```
'data.frame': 1000 obs. of 5 variables:
 $ lat     : num -20.4 -20.6 -26 -18 -20.4 ...
 $ long    : num 182 181 184 182 182 ...
 $ depth   : int 562 650 42 626 649 195 82 194 211 622 ...
 $ mag     : num 4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
 $ stations: int 41 15 43 19 11 12 43 15 35 19 ...
```

Descriptive Statistics

Descriptive statistics is the term given to the analysis of data that helps **describe**, show or summarize data in a meaningful way such that, for example, patterns might emerge from the data.

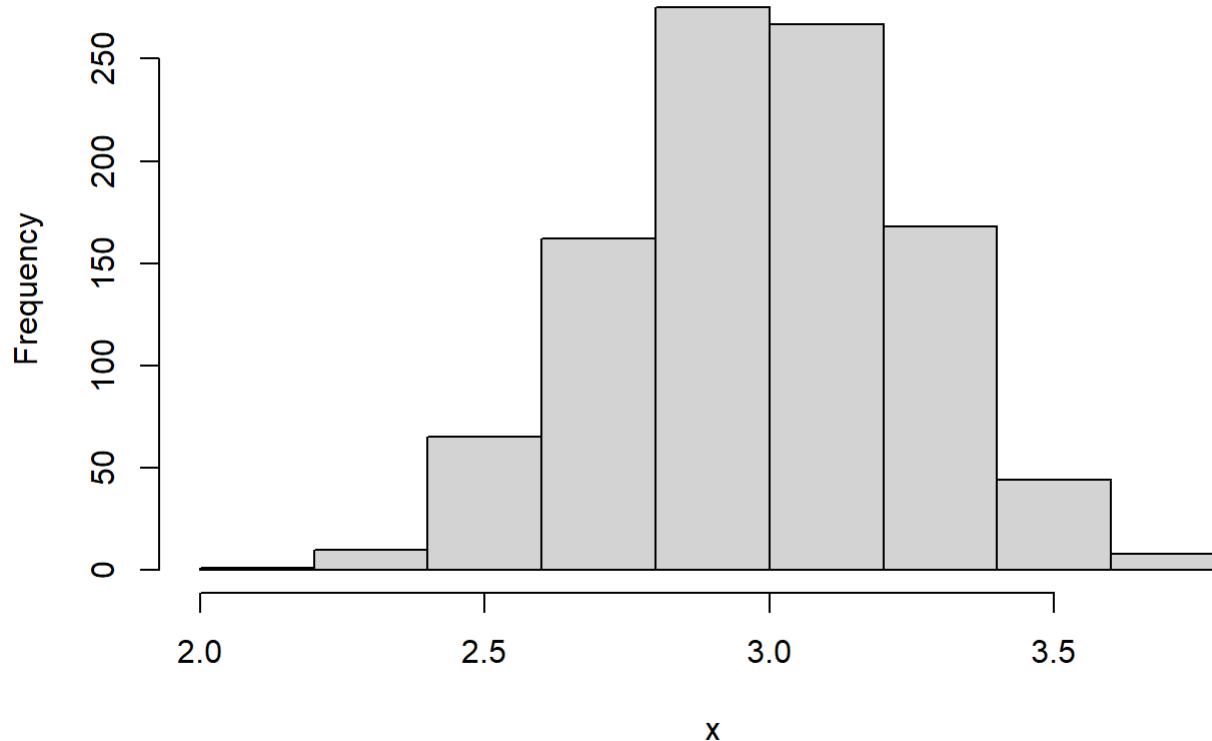
Descriptive statistics do not, however, allow us to make conclusions beyond the data we have analysed or reach conclusions regarding any hypotheses we might have made. They are simply a way to describe our data.

- **Measures of Central Tendency**

These are ways of describing the central position of a frequency distribution for a group of data.

```
x <- rnorm(1000,3,.25)
hist(x) ## distribution of values of continuous variables
```

Histogram of x



```
mean(x)
```

```
[1] 2.988755
```

```
median(x)
```

```
[1] 2.992323
```

```
y <- c(2,5,6,3,7,800,5,3,2,4)
mean(y) ## here, the mean is not a true representative of the salaries, so we report the median
```

```
[1] 83.7
```

```
median(y)
```

```
[1] 4.5
```

```
vab <- table(y) ## To have an idea of the modal value.
vab
```

```
y
 2   3   4   5   6   7 800
 2   2   1   2   1   1   1
```

```
mod <- vab[vab==max(vab)]
mod
```

```
y
2 3 5
2 2 2
```

```
min(y)
```

```
[1] 2
```

```
max(y)
```

```
[1] 800
```

```
range(y)
```

```
[1] 2 800
```

You can get a copy of my book on amazon titled; "Basic Statistics for Undergraduate Studies Using R" (<https://www.amazon.com/BASIC-STATISTICS-UNDERGRADUATE-STUDIES-USING/dp/B08C4FTJFT>) for more information on simple code to obtain the mode of your distributions.

```
library(moments)  
skewness(x)
```

```
[1] -0.1548827
```

```
## Skewness  
## Left Skewed: Mean < Median  
## Few smaller values reduces the mean  
  
## Right Skewed: Mean > Median  
## Few Larger values increases the mean
```

- **Measure of Spread**

These are ways of summarizing a group of data by describing how spread out the scores are.

```
#### MEASURE OF VARIATION  
std = sd(iris$Sepal.Length)  
std
```

```
[1] 0.8280661
```

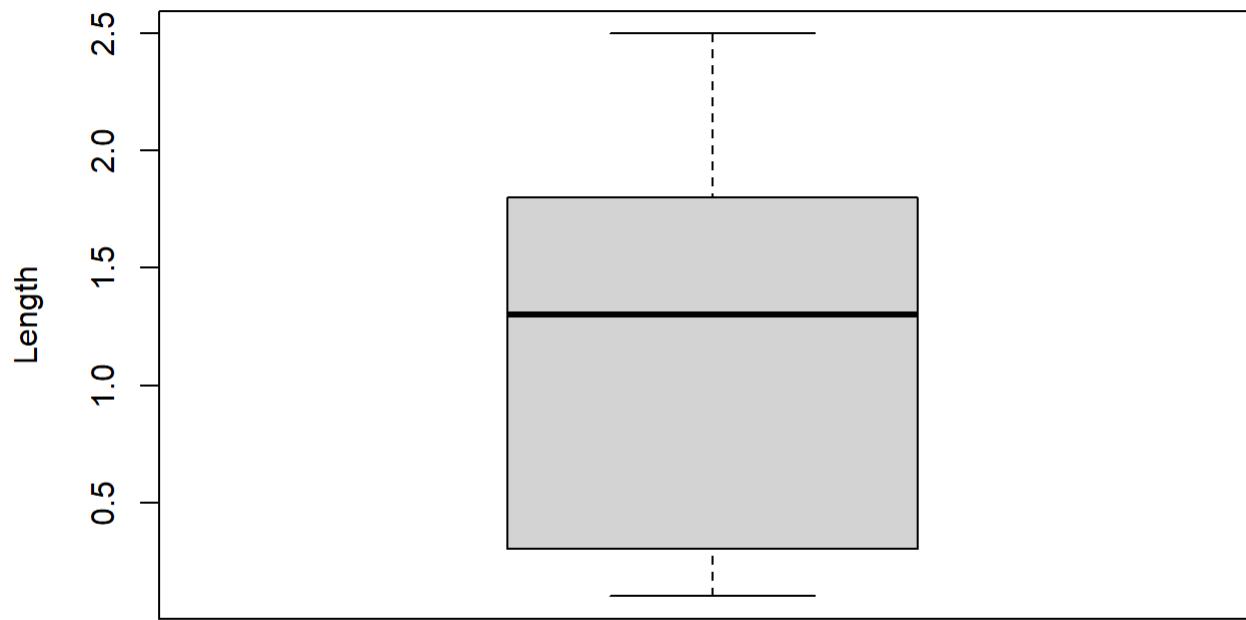
```
var(iris$Sepal.Length)
```

```
[1] 0.6856935
```

```
library(sciplot)
se(iris$Petal.Length)
```

```
[1] 0.144136
```

```
boxplot(iris$Petal.Width, Main = "Petal Length", ylab = "Length") ## 5 point number summary.
```



```
IQR(y)
```

```
[1] 2.75
```

Inferential Statistics

Inferential statistics use a random sample of data taken from a population to describe and make inferences about the population.

- Parameter and Statistics

We have seen that descriptive statistics provide information about our immediate group of data. Any group of data like this, which includes all the data you are interested in, is called a population. Descriptive statistics are applied to populations, and the properties of populations, like the mean or standard deviation, are called parameters as they represent the whole population (i.e., everybody you are interested in).

Often, however, you do not have access to the whole population you are interested in investigating, but only a limited number of data instead which is called the sample.

Properties of samples, such as the mean or standard deviation, are not called parameters, but statistics. Inferential statistics are techniques that allow us to use these samples to make generalizations about the populations from which the samples were drawn. It is, therefore, important that the sample accurately represents the population. The process of achieving this is called sampling.

The methods of inferential statistics are **(1) the estimation of parameter(s)** and **(2) testing of statistical hypotheses**.

- **Summary:**

The characteristics of that population are referred to as parameters. Estimates from a sample are known as statistics.

Parametric and Non-parametric test

Parametric tests are those that make assumptions about the parameters of the population distribution from which the sample is drawn but nonparametric tests don't require that your data follow the normal distribution. They're also known as distribution-free tests and can provide benefits in certain situations.

Non-parametric tests are valid for both non-Normally distributed data and Normally distributed data, so why not use them all the time? Parametric tests usually have more statistical power than their non-parametric equivalents. In other words, one is more likely to detect significant differences when they truly exist.

Parametric Test	Non- Parametric Equivalent
Paired t-test	Wilcoxon Rank Sum Test
Unpaired t-test	Mann-Whitney U Test
Pearson Correlation	Spearman Correlation
One-Way ANOVA	Kruskal Wallis Test



Figure 7: Parametric and non-parametric tests for comparing two or more groups

- **Hypothesis**

This is defined to be an assumption made with aim of calculating the probability of which the conclusion is incorrect. We have two types of it, they are; 1. Null hypothesis 2. Alternate hypothesis

The null hypothesis is the claim that is assumed to be true while the alternate hypothesis is what we are testing for against the null hypothesis.

NOTE

-When H_{alt} is defined with the less than statement, it is one-sided and can be said to be a lower-tailed test.

-When $H_{(alt)}$ is defined with the greater than statement, it is one-sided and can also be said to be an upper-tailed test.

-When $H_{(alt)}$ is defined with the not equal to sign, it is two-sided and can also be called a two-tailed test.

- **Hypothesis Testing**

After setting up a hypothesis, it is good for us to make some assumptions before carrying out the parametric test. These assumptions are listed below;

- We assume that the samples are taken at random and gives a perfect representation of the population without any element of bias.
- For two or more populations, we assume the equality of variance.
- We assume that the population(s) is normally distributed.

t-Test Application in R

- **One Sample t-test**

Example

Ten individual trees were randomly chosen from a population and their diameters at 1.3m above the ground level were measured in cm. The data is presented below;

25.46,31.51,17.04,30.10,38.40,31.19,37.56,41.70,21.01 and 41.83

In the light of the above data, could the mean diameter of the population be 28cm?

```
dat <- c(25.46,31.51,17.04,30.10,38.40,31.19,37.56,41.70,21.01,41.83)
dbh_test <- t.test(x=dat, mu=28, alternative = "less")
print(dbh_test)
```

One Sample t-test

```
data: dat
t = 1.3294, df = 9, p-value = 0.8918
alternative hypothesis: true mean is less than 28
95 percent confidence interval:
 -Inf 36.51652
sample estimates:
mean of x
31.58
```

```
names(dbh_test)
```

```
[1] "statistic"   "parameter"   "p.value"      "conf.int"     "estimate"
[6] "null.value"  "stderr"       "alternative"  "method"      "data.name"
```

```
attributes(dbh_test) ##same as above
```

```
$names
[1] "statistic"   "parameter"   "p.value"      "conf.int"     "estimate"
[6] "null.value"  "stderr"       "alternative"  "method"      "data.name"
```

```
$class
[1] "htest"
```

```
## To get the true C.I, we need to change the alternative argument to two sided.
dbh_test1 <- t.test(x=dat, mu=28, alternative = "two.sided")
names(dbh_test1)
```

```
[1] "statistic"   "parameter"   "p.value"      "conf.int"     "estimate"
[6] "null.value"  "stderr"       "alternative"  "method"      "data.name"
```

```
dbh_test1$conf.int
```

```
[1] 25.48808 37.67192
attr(,"conf.level")
[1] 0.95
```

- **Exercise**

Suppose a forester sells 16g of a tree seedling. A random sample of 9 seedlings were taken and weighed. 15.5, 16.2, 16.1, 15.8, 15.6, 16.0, 15.8, 15.9, 16.2

Is the average weight at least 16?

Answer

```
seedling <- c(15.5, 16.2, 16.1, 15.8, 15.6, 16.0, 15.8, 15.9, 16.2)
s_test <- t.test(x= seedling, mu= 16, alternative = "less")
print(s_test)
```

One Sample t-test

```
data: seedling
t = -1.2, df = 8, p-value = 0.1322
alternative hypothesis: true mean is less than 16
95 percent confidence interval:
 -Inf 16.05496
sample estimates:
mean of x
15.9
```

- **Unpaired/Independent Samples**

Unpooled Variance: Here, we cannot assume the equality of variances of the two groups.

Example

Suppose the forester in the exercise above selects another 6 random samples of seedling and weighed them. 14.7, 15.2, 16.6, 16.3, 15.4, 16.4

Can we conclude that there is a difference between the mean of the two samples?

```
seedling_samp <- c(14.7, 15.2, 16.6, 16.3, 15.4, 16.4)

testObj <- t.test(x=seedling_samp, y=seedling, alternative = "greater", conf.level = 0.95)
print(testObj)
```

Welch Two Sample t-test

```
data: seedling_samp and seedling
t = -0.40942, df = 5.7077, p-value = 0.6514
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
-0.7720041      Inf
sample estimates:
mean of x mean of y
15.76667  15.90000
```

Pooled: Here, we can assume equality of variance.

Example

Measurements on tree diameters from two different forest reserves were taken and recorded with the aim of assessing whether there is a difference between the mean diameter values obtained from the forest reserves. Perform a simple test confirming this.

shasha: 102,87,101,96,107,101,91,85,108,67,85,82 Ago-Owu: 73,81,111,109,143,95,92,120,93,89,119,79,90,126,62,92,77,106,105,111

```
shasha <- c(102,87,101,96,107,101,91,85,108,67,85,82)
ago_owu <- c(73,81,111,109,143,95,92,120,93,89,119,79,90,126,62,92,77,106,105,111)
testResult <- t.test(x=shasha, y=ago_owu, alternative = "two.sided", conf.level = 0.95, var.equal = T)
print(testResult)
```

Two Sample t-test

```
data: shasha and ago_owu
t = -0.93758, df = 30, p-value = 0.3559
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-19.016393 7.049727
sample estimates:
mean of x mean of y
92.66667 98.65000
```

- **Paired/Dependent Samples**

Paired data occur when the measurements forming two sets of observations are taken from the same individual.

Examples

A forester interested in the rate of growth of his gmelina arborea plantation takes a yearly measurement of the height of his trees and record them. The record for the first and second year are provide below:

height before: 52,66,89,87,89,72,66,65,49,62,70,52,75,63,65,61

height after: 51,66,71,73,70,68,60,51,40,57,65,53,64,56,60,59

```
height_before <- c(52,66,89,87,89,72,66,65,49,62,70,52,75,63,65,61)
height_after <- c(51,66,71,73,70,68,60,51,40,57,65,53,64,56,60,59)

length(height_before); length(height_after)
```

[1] 16

[1] 16

```
heightdiff <- height_after - height_before  
print(heightdiff)
```

```
[1] -1 0 -18 -14 -19 -4 -6 -14 -9 -5 -5 1 -11 -7 -5 -2
```

```
heightdiff_mean <- mean(heightdiff)  
heightdiff_sd <- sd(heightdiff)  
  
height_test <- heightdiff_mean / (heightdiff_sd/sqrt(16))  
print(height_test)
```

```
[1] -4.801146
```

```
p_val <- pt(height_test, df=15)  
print(p_val)
```

```
[1] 0.000116681
```

Doing this with a single line of code as usual.

```
testObject <- t.test(x=height_after, y=height_before, alternative = "less", conf.level = .95, paired = T)  
print(testObject)
```

Paired t-test

```
data: height_after and height_before
t = -4.8011, df = 15, p-value = 0.0001167
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
-Inf -4.721833
sample estimates:
mean of the differences
-7.4375
```

You can use the line of code below to get the confidence interval.

```
CI <- heighthdiff_mean + c(-1,1) * qt(0.975, df=15) * (heighthdiff_sd/sqrt(16))
print(CI)
```

```
[1] -10.739348 -4.135652
```

NOTE:

```
## It can also be used to examine the relationship between a numeric outcome variable y and a categorical explanatory variable with 2 levels. (yes,no)

## boxplot(x~y)
## t.test(y~x, mu=0, alt="two.sided", conf= 0.95, var.eq=F, paired=F)

## t.test(y~x)
## t.test(y[x=="no"], y[x=="yes"])
```

Equality of Variance Test

```
## First with a boxplot  
## boxplot(y~x) ensure both x and y are of the same length.  
  
## var(y); var(x)  
## var(y[x=="no"]); var(y[x=="yes"])  
  
## Levene's test  
  
## Null: Population variances are equal.  
## library(car)  
## LeveneTest(y~x)
```

For chi-square analysis with R, please get a copy of my book “Basic Statistics for Undergraduate Studies Using R” (<https://www.amazon.com/BASIC-STATISTICS-UNDERGRADUATE-STUDIES-USING/dp/B08C4FTJFT>).

- One-way ANOVA

```
head(chickwts)
```

```
weight      feed  
1    179 horsebean  
2    160 horsebean  
3    136 horsebean  
4    227 horsebean  
5    217 horsebean  
6    168 horsebean
```

```
levels(chickwts$feed)
```

```
[1] "casein"     "horsebean"   "linseed"     "meatmeal"    "soybean"    "sunflower"
```

```
library(dplyr)
chick <- slice(chickwts, c(1:36))
names(chick)
```

```
[1] "weight" "feed"
```

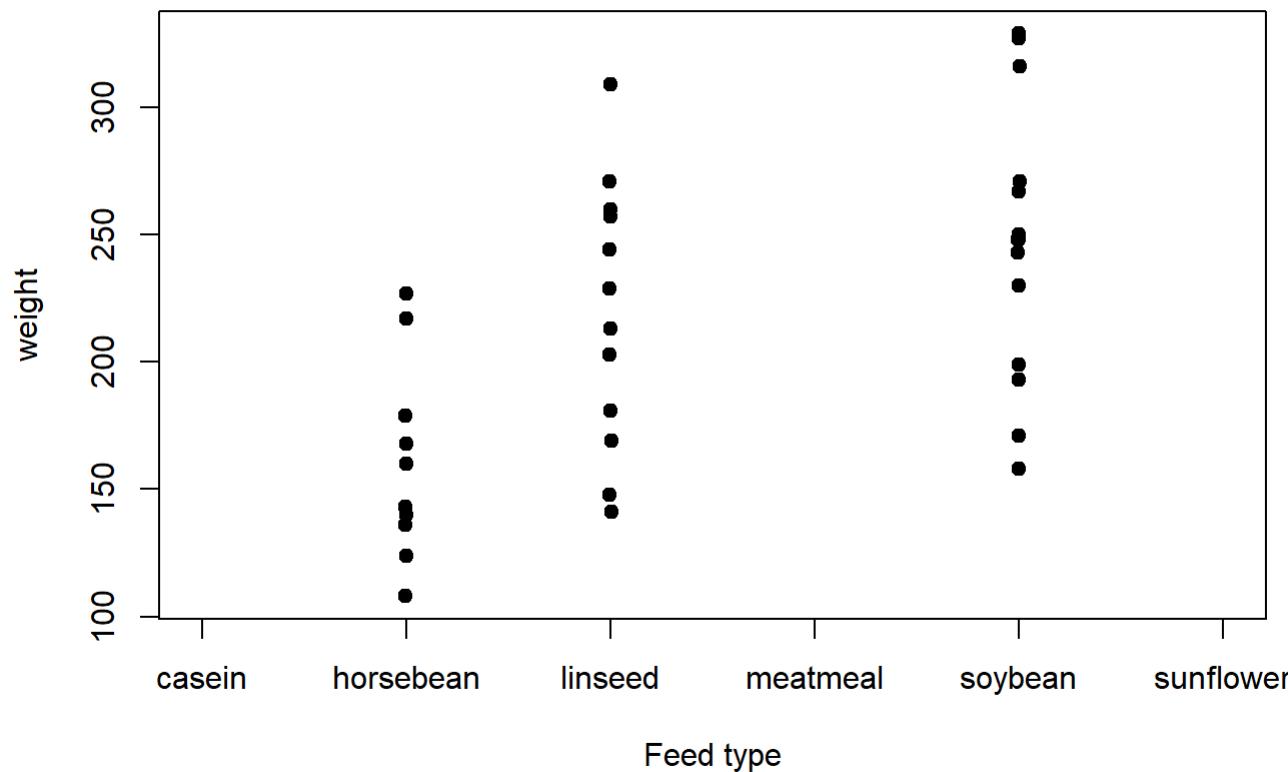
```
str(chick)
```

```
'data.frame': 36 obs. of 2 variables:
 $ weight: num 179 160 136 227 217 168 108 124 143 140 ...
 $ feed   : Factor w/ 6 levels "casein","horsebean",...: 2 2 2 2 2 2 2 2 2 2 ...
```

```
levels(chick$feed)
```

```
[1] "casein"    "horsebean"  "linseed"    "meatmeal"   "soybean"    "sunflower"
```

```
stripchart(weight~feed, vertical= T, pch=19, data = chick, xlab="Feed type", method= "jitter", jitter = .004)
```



```
result <- lm(weight~feed, data = chick)
anova(result)
```

Analysis of Variance Table

Response: weight

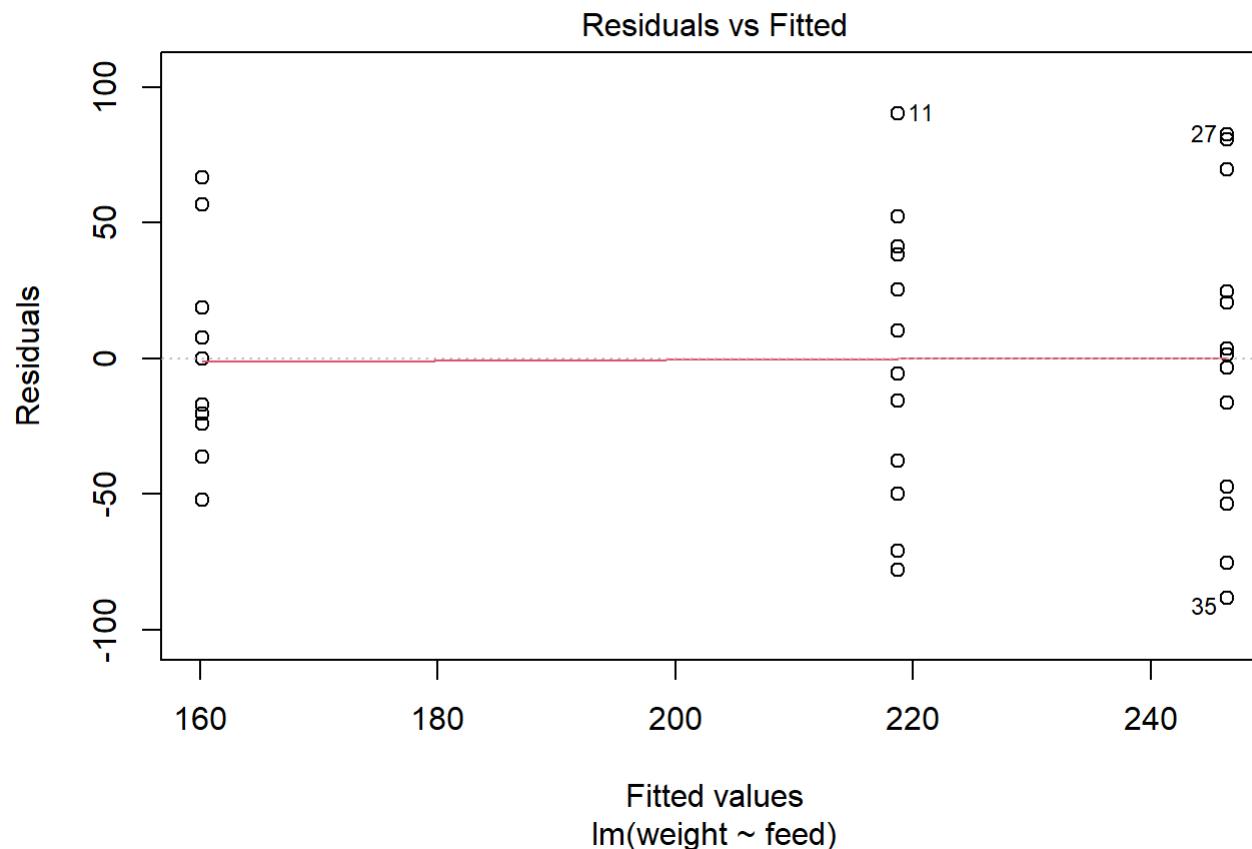
	Df	Sum Sq	Mean Sq	F value	Pr(>F)
--	----	--------	---------	---------	--------

feed 2 43917 21958.7 8.8879 0.0008169 ***

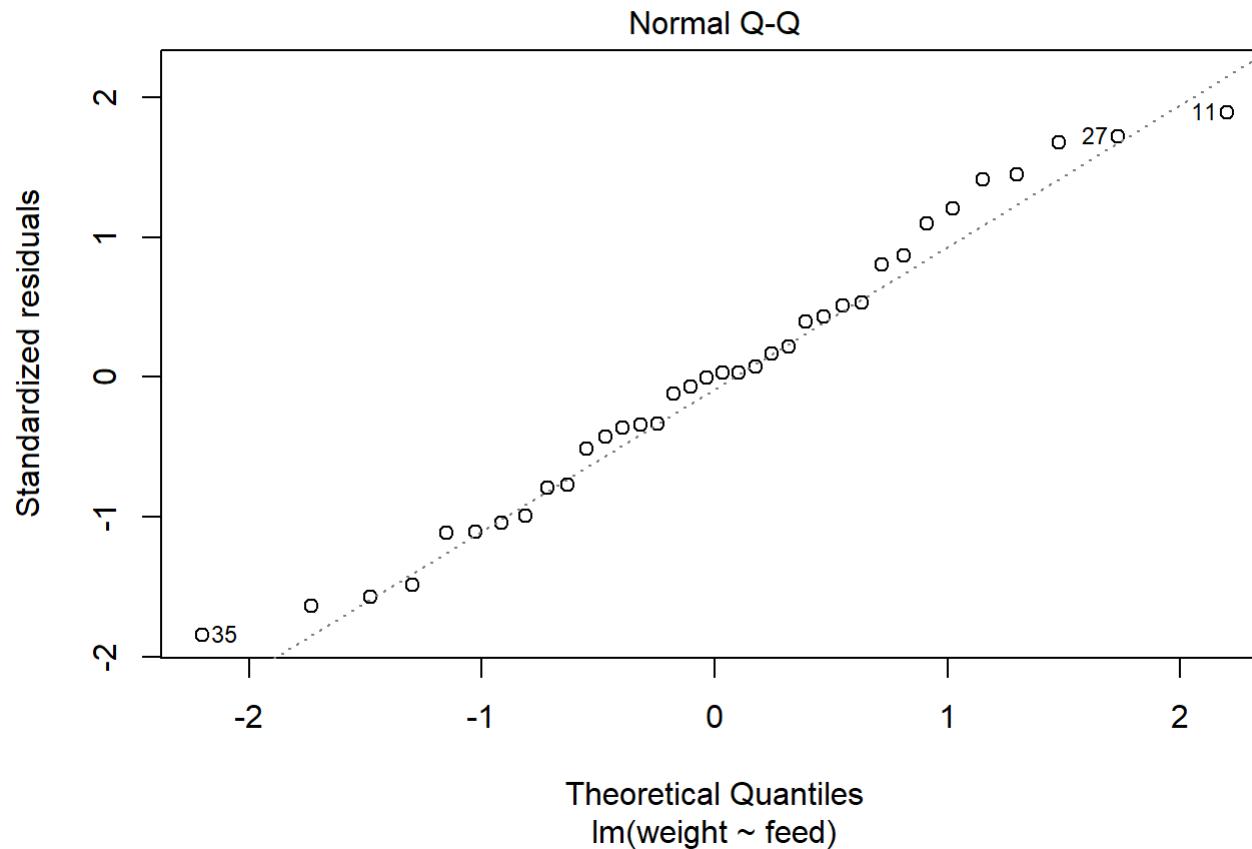
Residuals 33 81531 2470.6

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

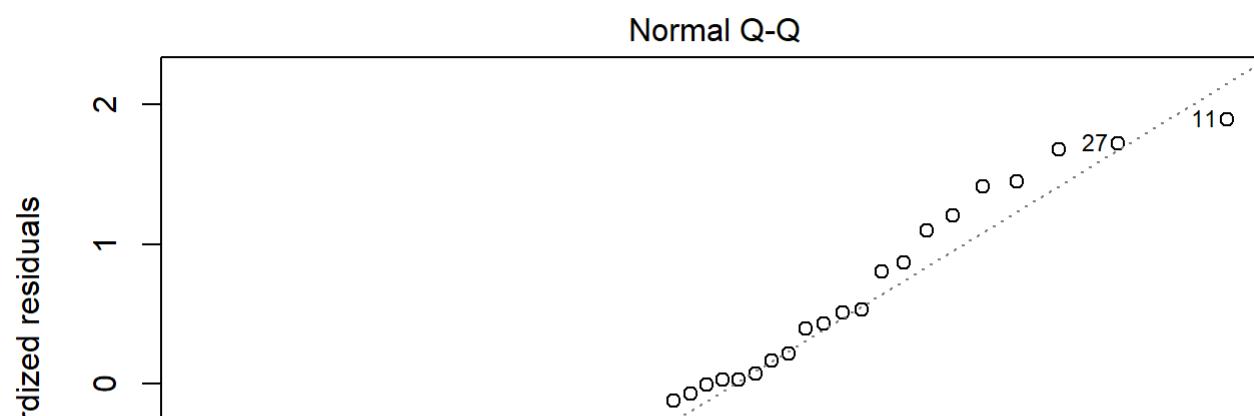
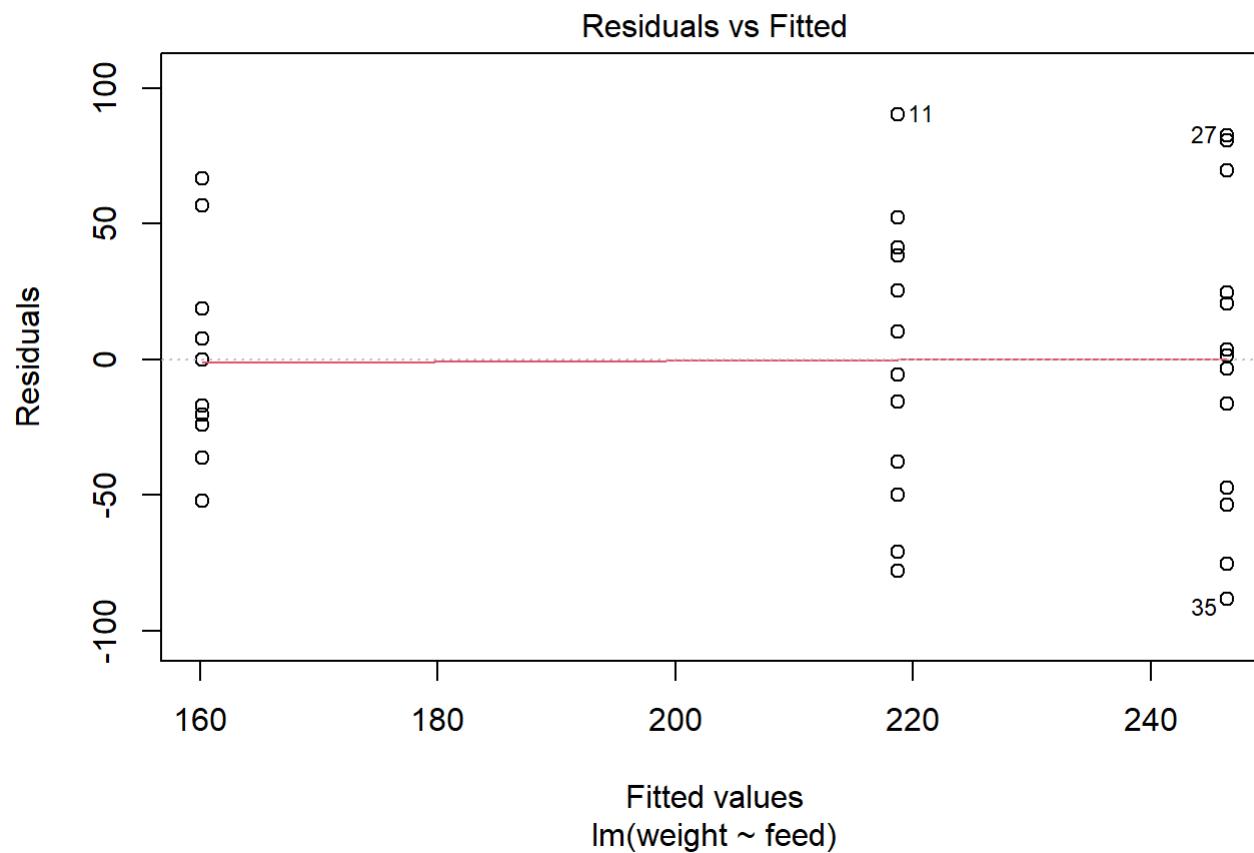
```
plot(result, which = 1)
```

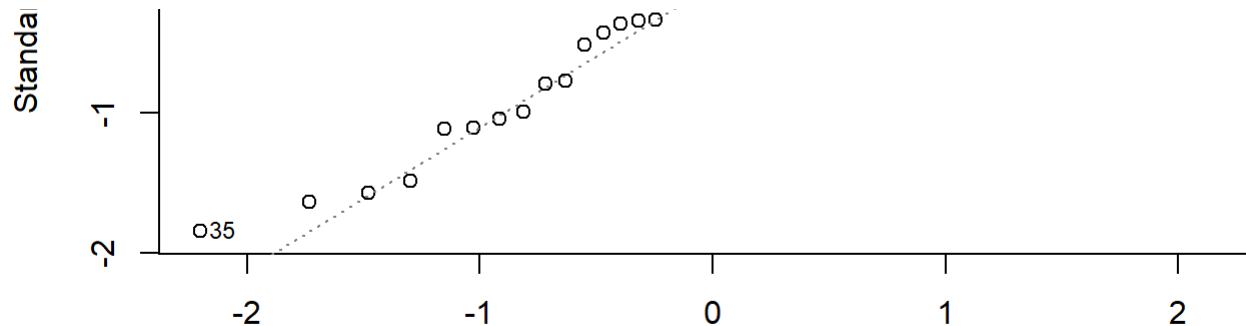


```
## Here we can see that the overall variances are relatively homogenous for all the fitted values.  
plot(result, which = 2) ## Normality assumptions
```

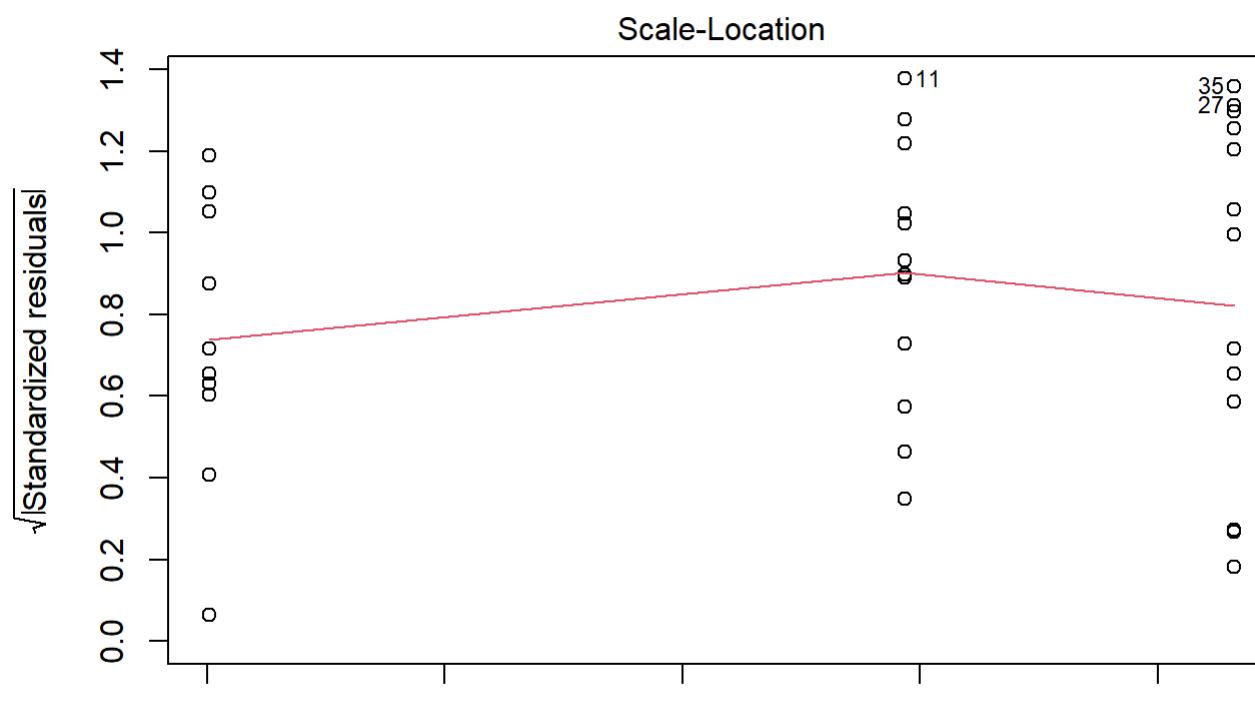


```
## Here we can see it follows the straight line except for a little skewness on top.  
plot(result)
```

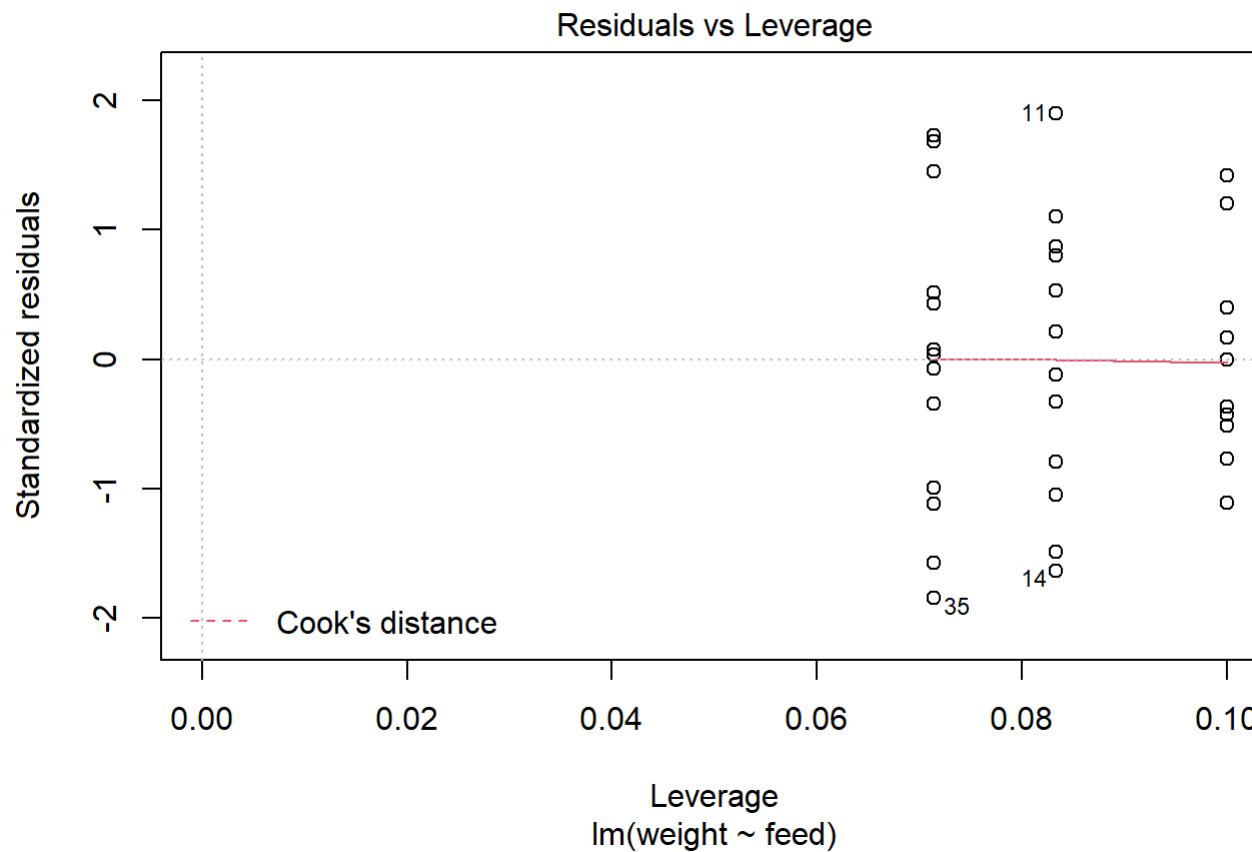





Theoretical Quantiles
 $lm(\text{weight} \sim \text{feed})$

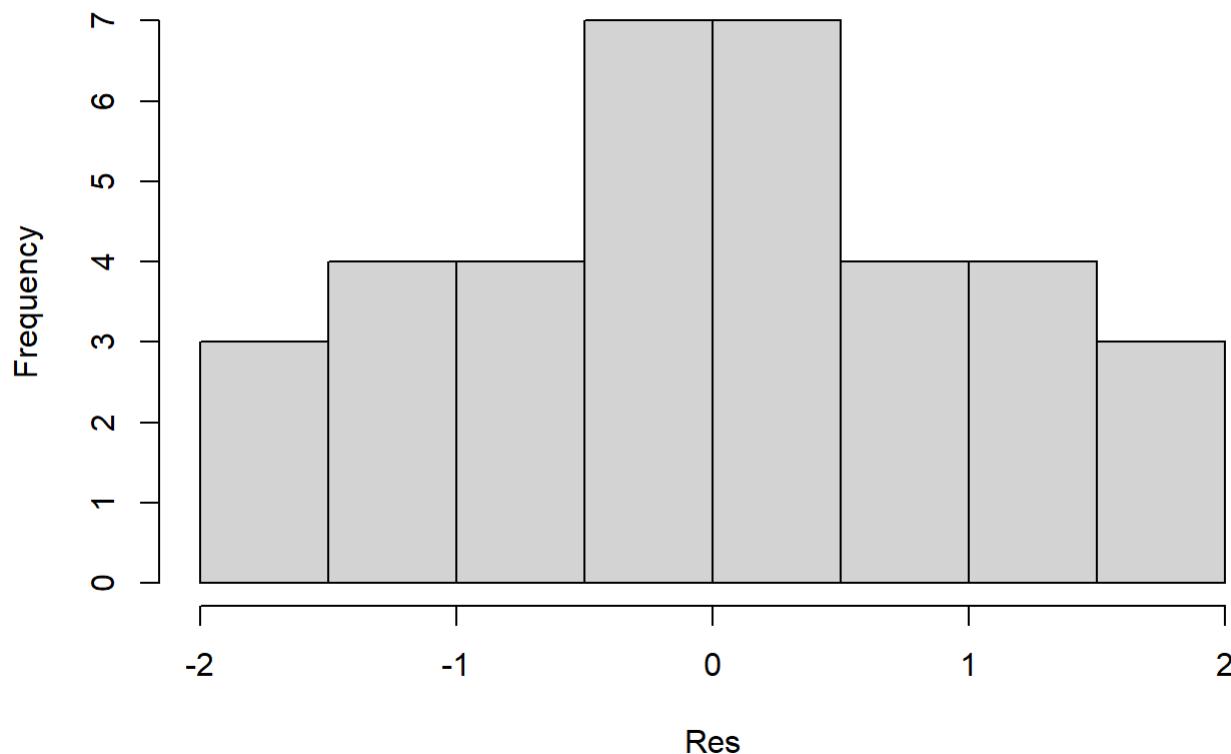


Fitted values
 $lm(\text{weight} \sim \text{feed})$



```
Res <- rstandard(result) ## for obtaining standardized residuals  
hist(Res)
```

Histogram of Res



```
TukeyHSD(aov(result))
```

```
Tukey multiple comparisons of means
95% family-wise confidence level
```

```
Fit: aov(formula = result)
```

```
$feed
```

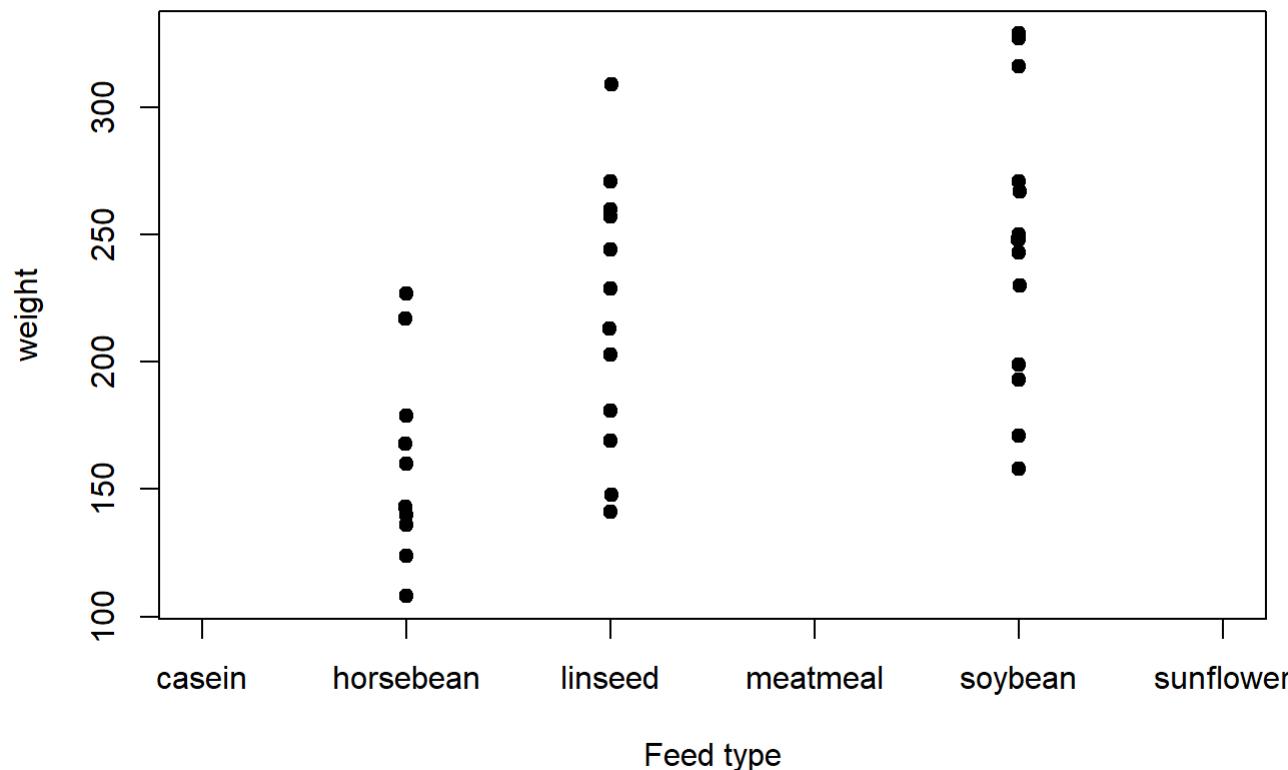
	diff	lwr	upr	p	adj
linseed-horsebean	58.55000	6.32673	110.77327	0.0252584	
soybean-horsebean	86.22857	35.72932	136.72782	0.0005601	
soybean-linseed	27.67857	-20.30310	75.66024	0.3447287	

```
## soybean and horsebean contributes most of the overall significance between the feed type.
```

```
## Let us visualize this.
```

```
stripchart(weight~feed, vertical= T, pch=19, data = chick, xlab="Feed type", method= "jitter", jitter = .004)
```

```
library(agricolae)
```



```
DMRT <- duncan.test(chick$weight, chick$feed, 33, 2470.6)
print(DMRT)
```

```
$statistics
  MSerror Df   Mean       CV
  2470.6  33 213.25 23.30838

$parameters
  test      name.t ntr alpha
Duncan chick$feed    3  0.05

$duncan
NULL

$means
  chick$weight      std  r Min Max     Q25     Q50     Q75
horsebean    160.2000 38.62584 10 108 227 137.00 151.5 176.25
linseed      218.7500 52.23570 12 141 309 178.00 221.0 257.75
soybean      246.4286 54.12907 14 158 329 206.75 248.0 270.00

$comparison
NULL

$groups
  chick$weight groups
soybean      246.4286      a
linseed      218.7500      a
horsebean    160.2000      b

attr("class")
[1] "group"
```

```
LSD <- LSD.test(chick$weight, chick$feed, 33, 2470.6)
print(LSD)
```

```
$statistics
  MSerror Df   Mean      CV
  2470.6 33 213.25 23.30838

$parameters
  test p.adjusted      name.t ntr alpha
  Fisher-LSD      none chick$feed  3  0.05

$means
  chick$weight      std  r      LCL      UCL Min Max    Q25    Q50
horsebean     160.2000 38.62584 10 128.2212 192.1788 108 227 137.00 151.5
linseed       218.7500 52.23570 12 189.5575 247.9425 141 309 178.00 221.0
soybean       246.4286 54.12907 14 219.4016 273.4556 158 329 206.75 248.0
  Q75
horsebean 176.25
linseed    257.75
soybean   270.00

$comparison
NULL

$groups
  chick$weight groups
soybean      246.4286     a
linseed       218.7500     a
horsebean    160.2000     b

attr(,"class")
[1] "group"
```

- **Two-way ANOVA**

This is just an extension of the one-way ANOVA that examines the influence of two different categorical independent variable on one continuous dependent variable.

- denotes interaction and : could be used as well. To add more than one predictor, the plus sign is used.

```
## interaction.plot(c1, c2, response, xLab, yLab, trace.Label=)
```

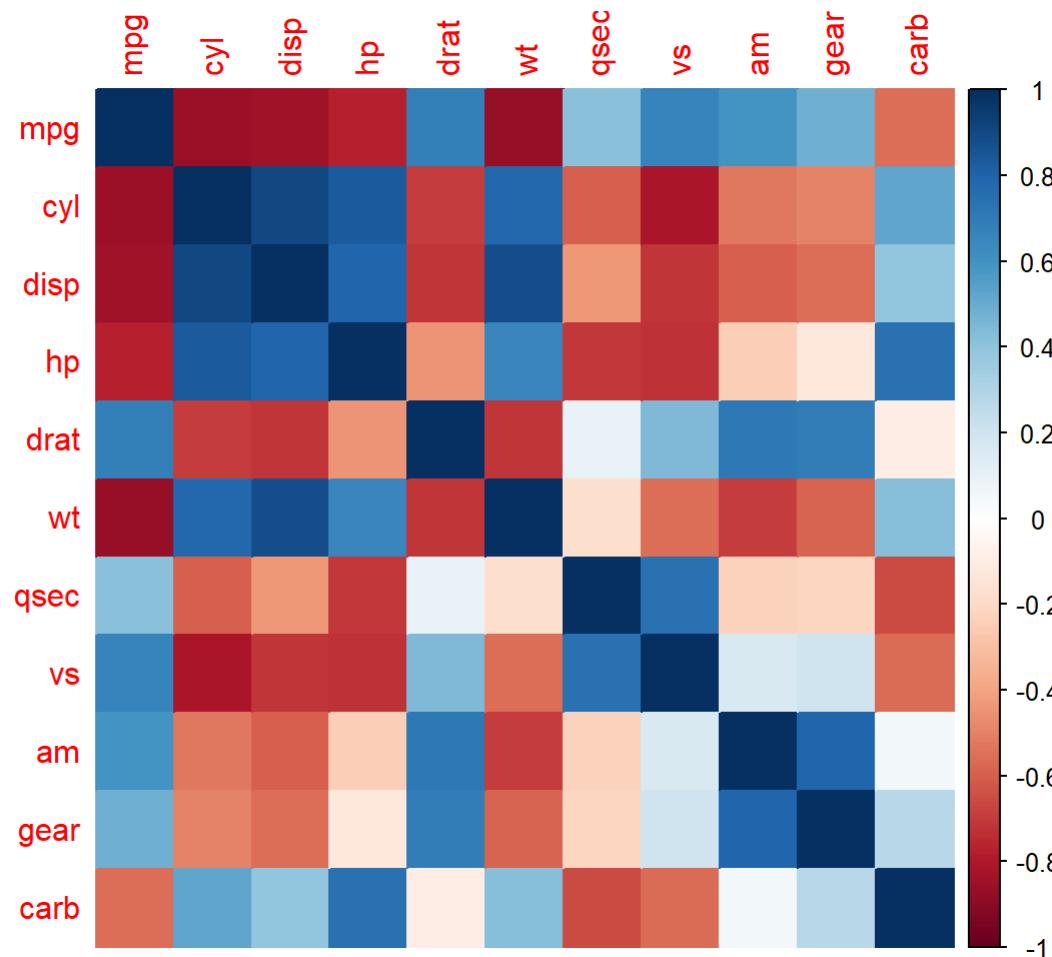
- Correlation and Regression in R

```
df = mtcars
num.cols <- sapply(df, is.numeric)
cor.data <- cor(df[, num.cols])
print(cor.data)
```

	mpg	cyl	disp	hp	drat	wt
mpg	1.0000000	-0.8521620	-0.8475514	-0.7761684	0.68117191	-0.8676594
cyl	-0.8521620	1.0000000	0.9020329	0.8324475	-0.69993811	0.7824958
disp	-0.8475514	0.9020329	1.0000000	0.7909486	-0.71021393	0.8879799
hp	-0.7761684	0.8324475	0.7909486	1.0000000	-0.44875912	0.6587479
drat	0.6811719	-0.6999381	-0.7102139	-0.4487591	1.0000000	-0.7124406
wt	-0.8676594	0.7824958	0.8879799	0.6587479	-0.71244065	1.0000000
qsec	0.4186840	-0.5912421	-0.4336979	-0.7082234	0.09120476	-0.1747159
vs	0.6640389	-0.8108118	-0.7104159	-0.7230967	0.44027846	-0.5549157
am	0.5998324	-0.5226070	-0.5912270	-0.2432043	0.71271113	-0.6924953
gear	0.4802848	-0.4926866	-0.5555692	-0.1257043	0.69961013	-0.5832870
carb	-0.5509251	0.5269883	0.3949769	0.7498125	-0.09078980	0.4276059
	qsec	vs	am	gear	carb	
mpg	0.41868403	0.6640389	0.59983243	0.4802848	-0.55092507	
cyl	-0.59124207	-0.8108118	-0.52260705	-0.4926866	0.52698829	
disp	-0.43369788	-0.7104159	-0.59122704	-0.5555692	0.39497686	
hp	-0.70822339	-0.7230967	-0.24320426	-0.1257043	0.74981247	
drat	0.09120476	0.4402785	0.71271113	0.6996101	-0.09078980	
wt	-0.17471588	-0.5549157	-0.69249526	-0.5832870	0.42760594	
qsec	1.0000000	0.7445354	-0.22986086	-0.2126822	-0.65624923	
vs	0.74453544	1.0000000	0.16834512	0.2060233	-0.56960714	
am	-0.22986086	0.1683451	1.00000000	0.7940588	0.05753435	
gear	-0.21268223	0.2060233	0.79405876	1.0000000	0.27407284	
carb	-0.65624923	-0.5696071	0.05753435	0.2740728	1.00000000	

```
## Library(DT)
## datatable(cor.data, extensions = "buttons", options = List(
##   dom = "Bfrtip",
##   buttons = c("copy", "csv", "excel", "pdf", "print")
## ))  
  
## Just for making a nice datatable with copy options.
```

```
library(corrgram)
library(corrplot)  
  
print(corrplot(cor.data, method = "color"))
```



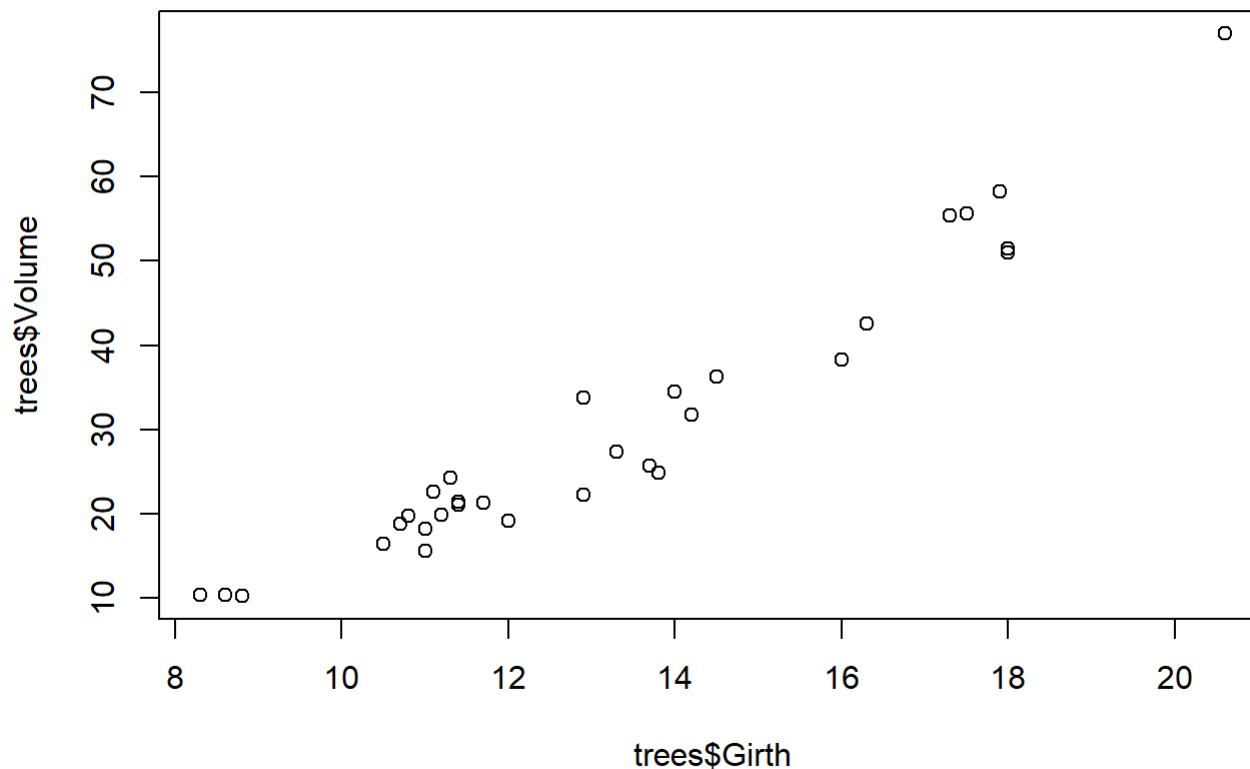
	mpg	cyl	disp	hp	drat	wt
mpg	1.0000000	-0.8521620	-0.8475514	-0.7761684	0.68117191	-0.8676594
cyl	-0.8521620	1.0000000	0.9020329	0.8324475	-0.69993811	0.7824958
disp	-0.8475514	0.9020329	1.0000000	0.7909486	-0.71021393	0.8879799
hp	-0.7761684	0.8324475	0.7909486	1.0000000	-0.44875912	0.6587479
drat	0.6811719	-0.6999381	-0.7102139	-0.4487591	1.0000000	-0.7124406
wt	-0.8676594	0.7824958	0.8879799	0.6587479	-0.71244065	1.0000000
qsec	0.4186840	-0.5912421	-0.4336979	-0.7082234	0.09120476	-0.1747159
vs	0.6640389	-0.8108118	-0.7104159	-0.7230967	0.44027846	-0.5549157
am	0.5998324	-0.5226070	-0.5912270	-0.2432043	0.71271113	-0.6924953
gear	0.4802848	-0.4926866	-0.5555692	-0.1257043	0.69961013	-0.5832870
carb	-0.5509251	0.5269883	0.3949769	0.7498125	-0.09078980	0.4276059
	qsec	vs	am	gear	carb	
mpg	0.41868403	0.6640389	0.59983243	0.4802848	-0.55092507	
cyl	-0.59124207	-0.8108118	-0.52260705	-0.4926866	0.52698829	
disp	-0.43369788	-0.7104159	-0.59122704	-0.5555692	0.39497686	
hp	-0.70822339	-0.7230967	-0.24320426	-0.1257043	0.74981247	
drat	0.09120476	0.4402785	0.71271113	0.6996101	-0.09078980	
wt	-0.17471588	-0.5549157	-0.69249526	-0.5832870	0.42760594	
qsec	1.0000000	0.7445354	-0.22986086	-0.2126822	-0.65624923	
vs	0.74453544	1.0000000	0.16834512	0.2060233	-0.56960714	
am	-0.22986086	0.1683451	1.0000000	0.7940588	0.05753435	
gear	-0.21268223	0.2060233	0.79405876	1.0000000	0.27407284	
carb	-0.65624923	-0.5696071	0.05753435	0.2740728	1.0000000	

Visualization Techniques with R

- The Plot Function

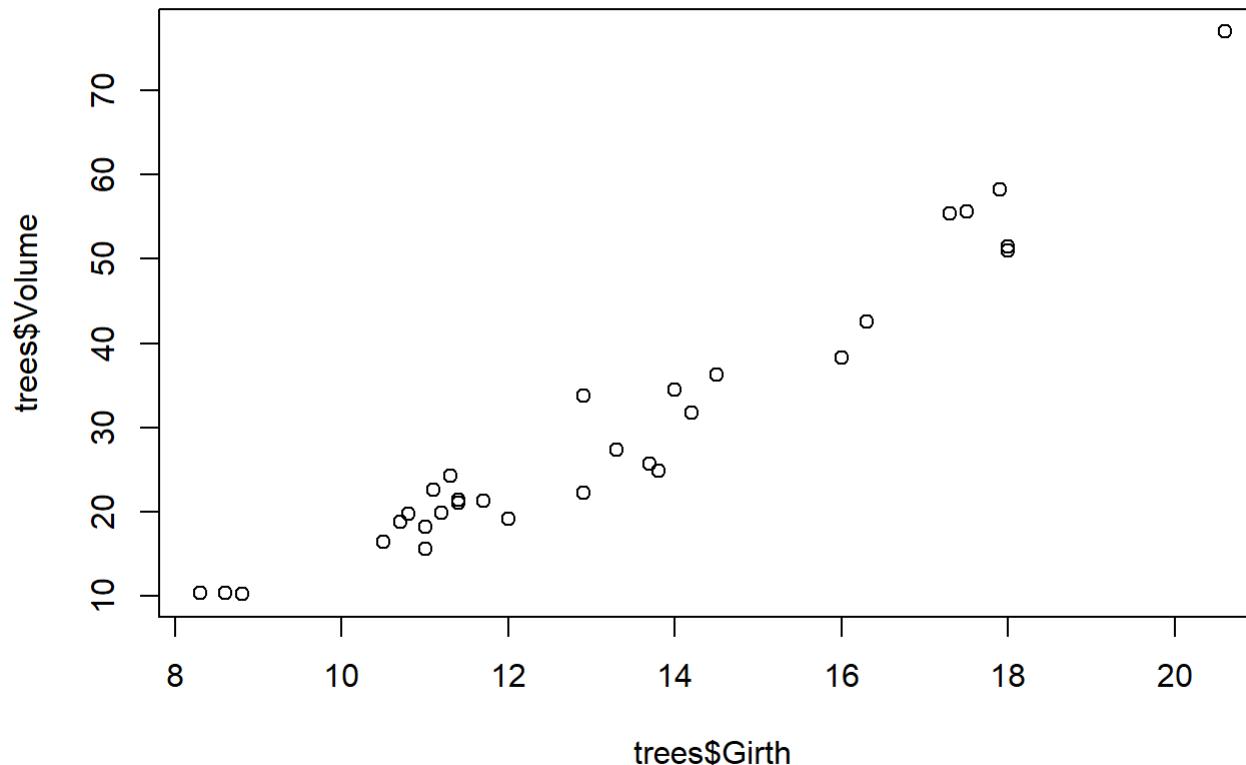
For visualizing simple scatterplots. You can make reference to my new book for more examples of other plot functions.

```
plot(trees$Girth,trees$Volume)
```



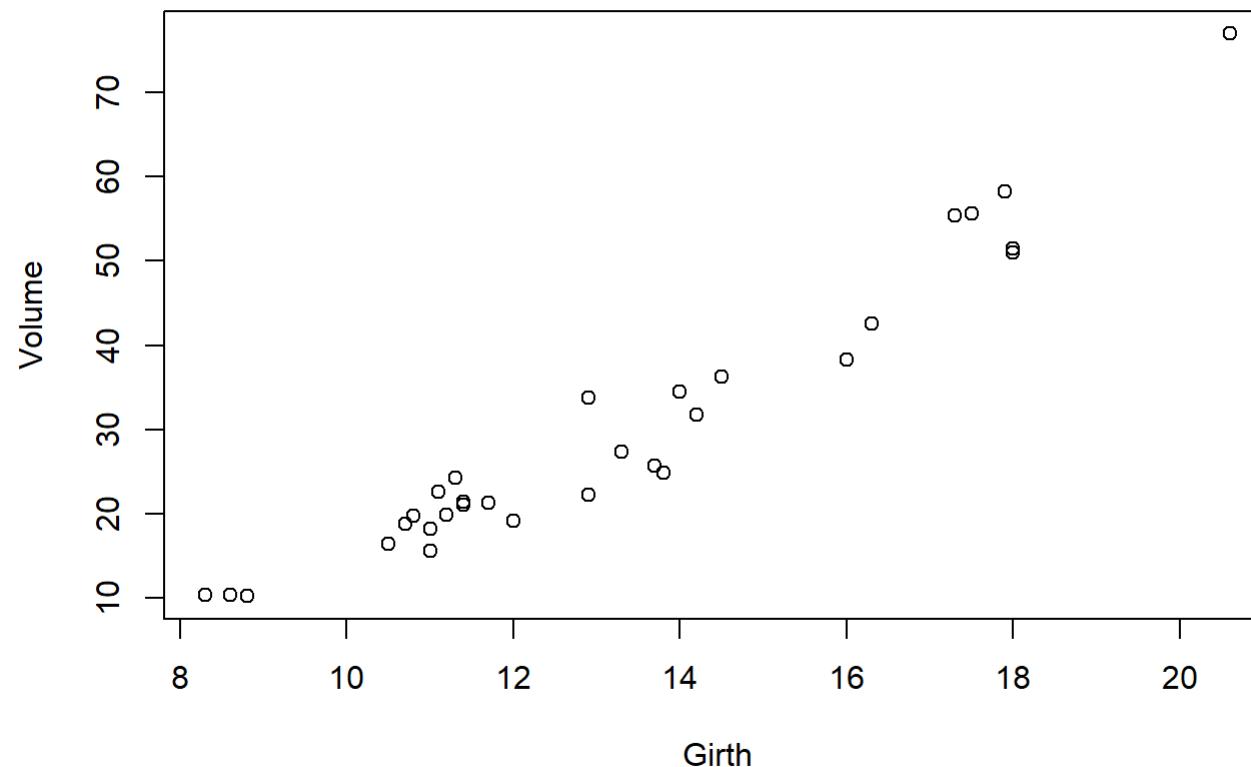
```
## To add title  
plot(trees$Girth,trees$Volume, main = "Scatterplot of Girth and Volume")
```

Scatterplot of Girth and Volume



```
## To add Labels
plot(trees$Girth,trees$Volume, main = "Scatterplot of Girth and Volume",
      xlab = "Girth", ylab = "Volume")
```

Scatterplot of Girth and Volume



- **ggplot2**

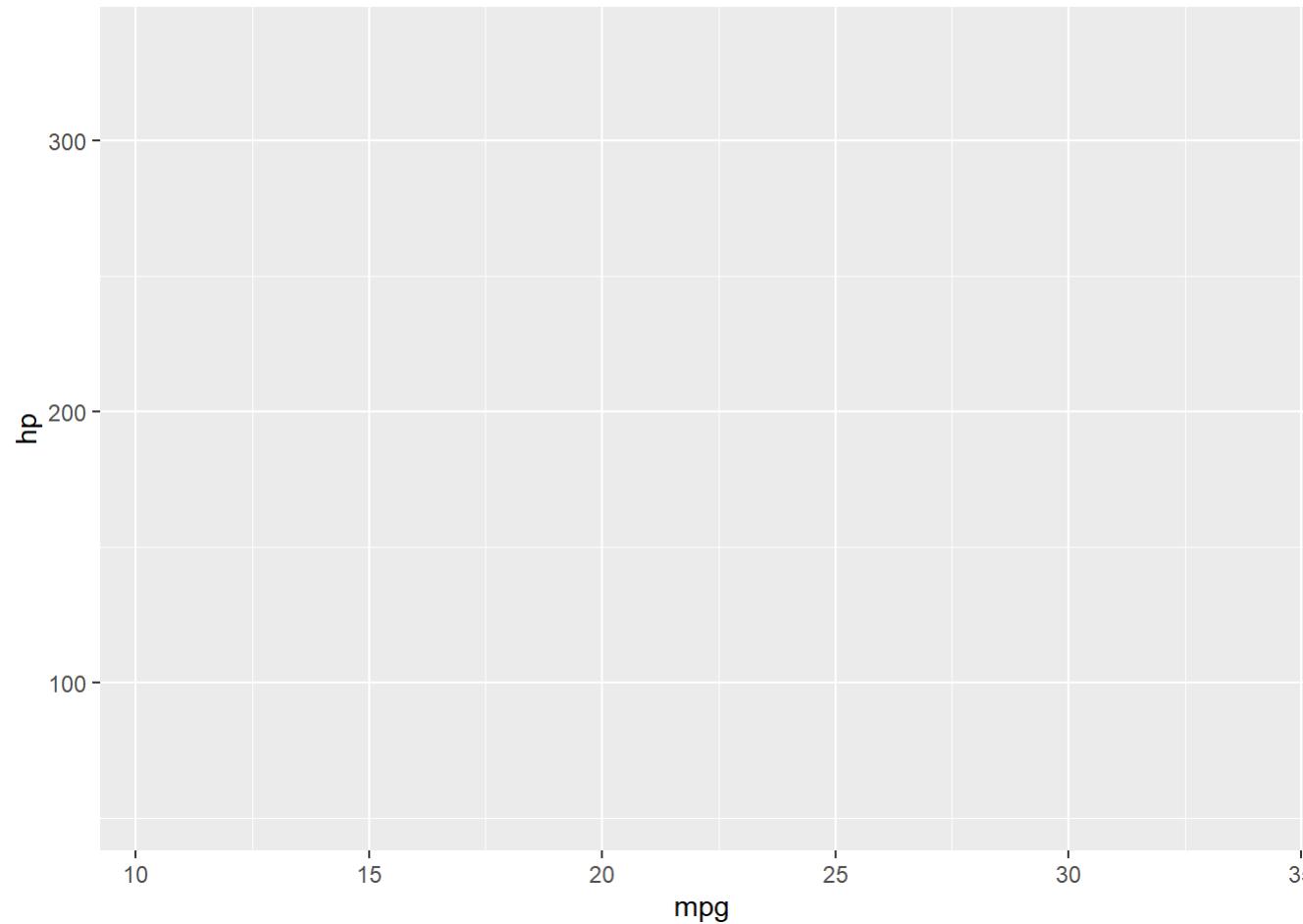
This is one of the most popular data visualization package for R created by Hadley Wickham. It follows a distinct pattern of graphics and it builds on the ideas of adding layers.

ggplot2 layers

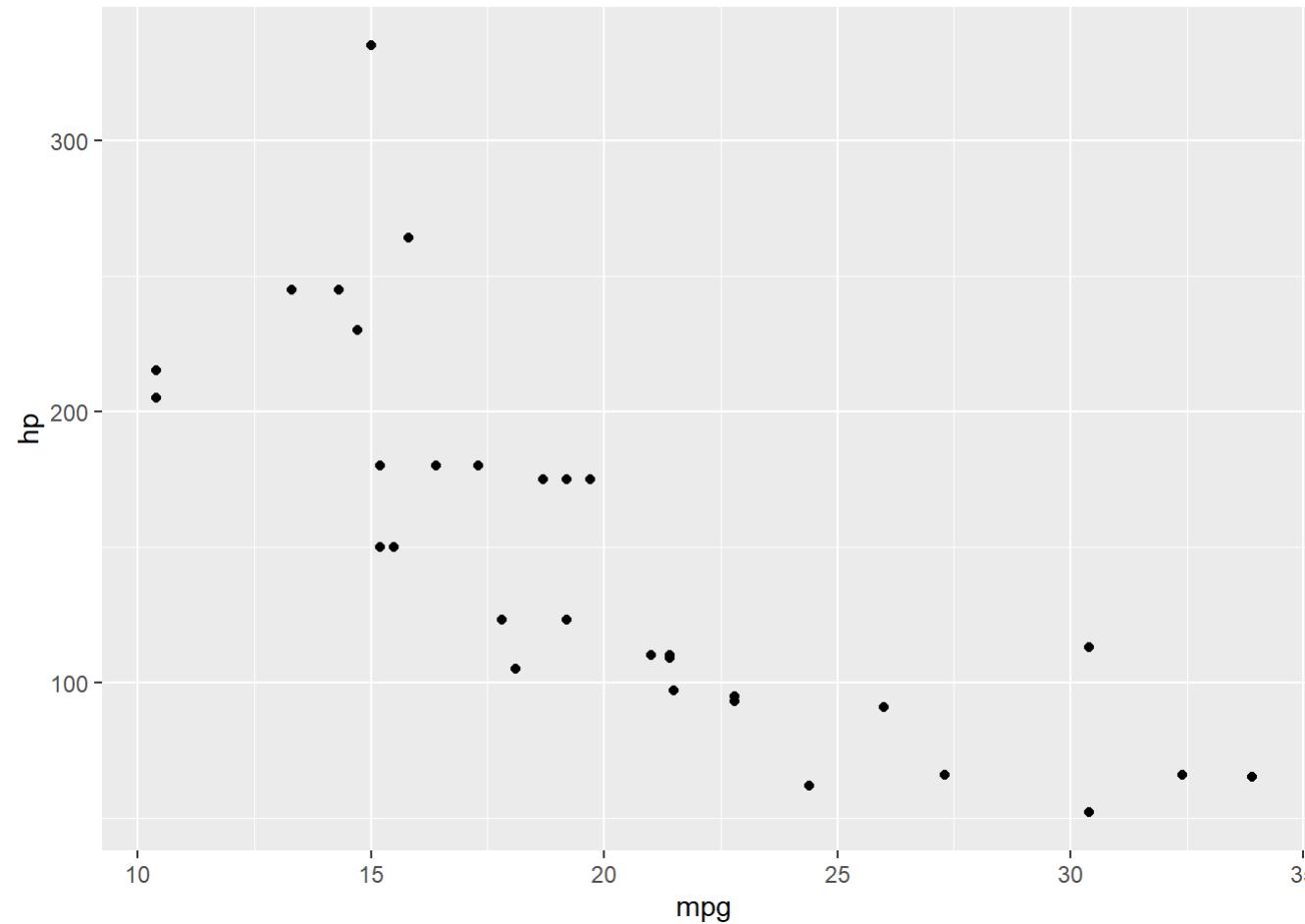
1. Data
2. Aesthetics
3. Geometry

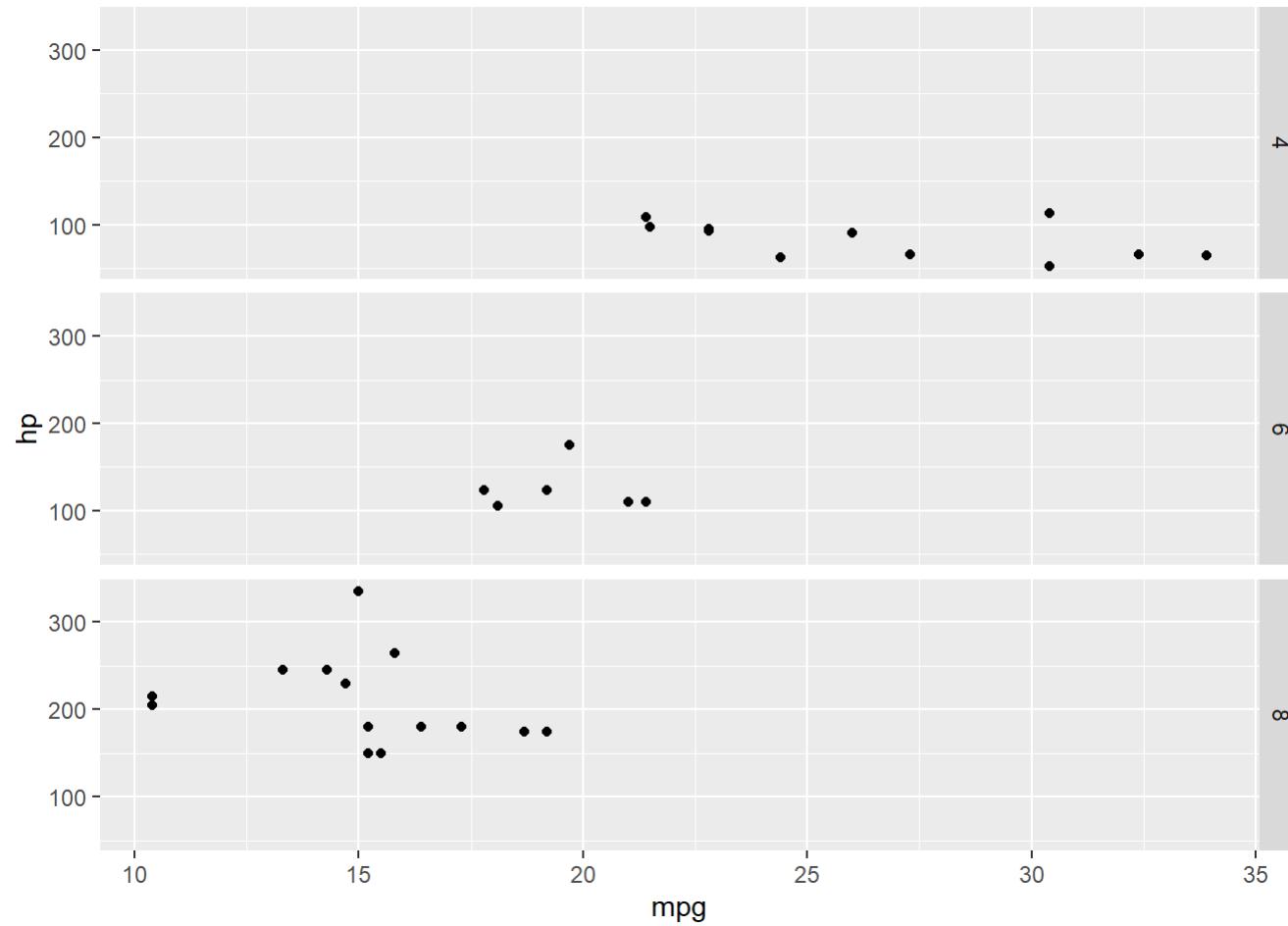
```
## Ensure ggplot2 package is installed on your software  
library(ggplot2)  
ggplot(data = mtcars) ## with this nothing will show up.
```

```
ggplot(data = mtcars, aes(x=mpg, y=hp)) ## data + aesthetics
```

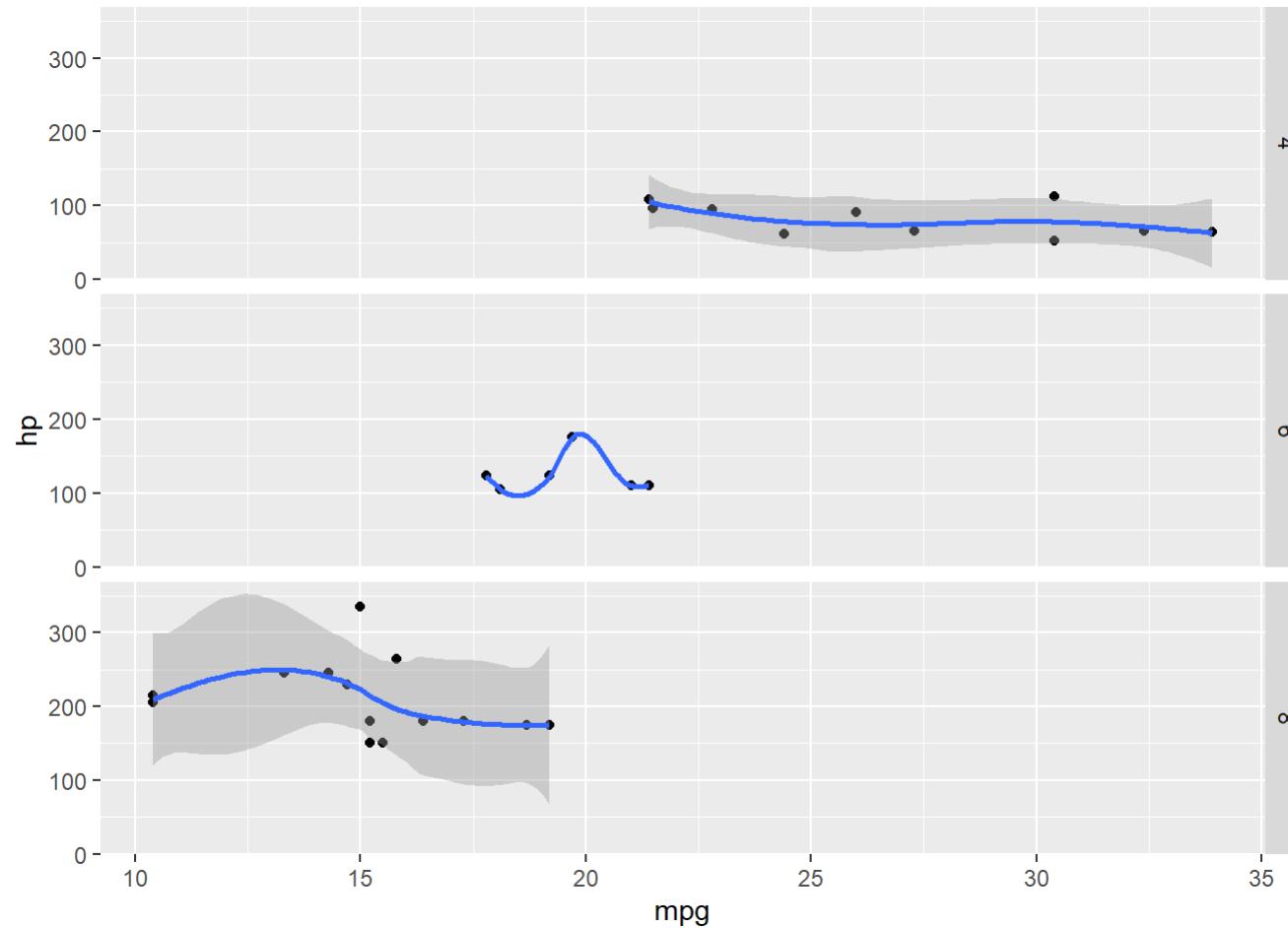


```
pl <- ggplot(data = mtcars, aes(x=mpg, y=hp))
pl + geom_point() ## point for scatterplot geometry.
```

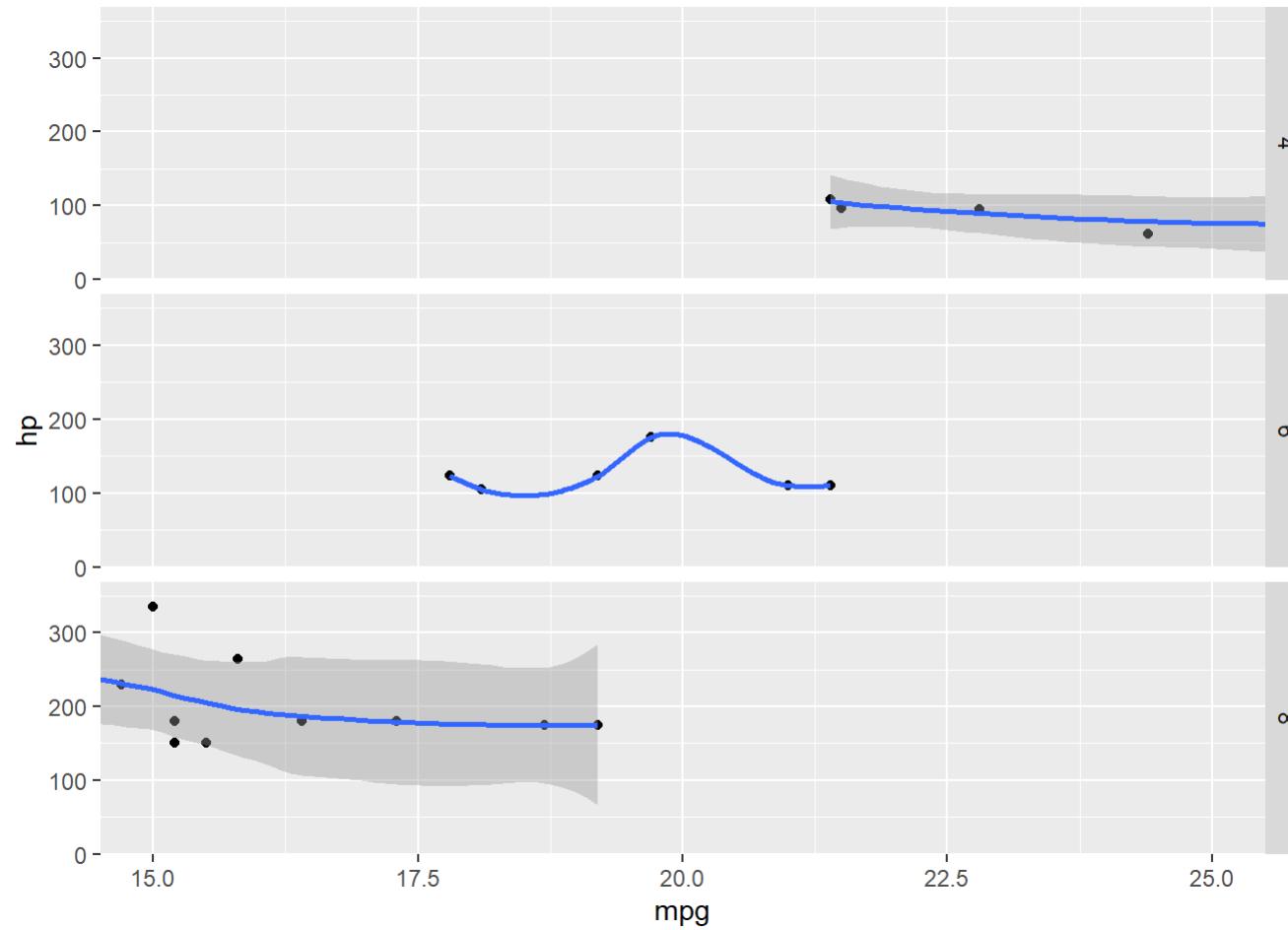




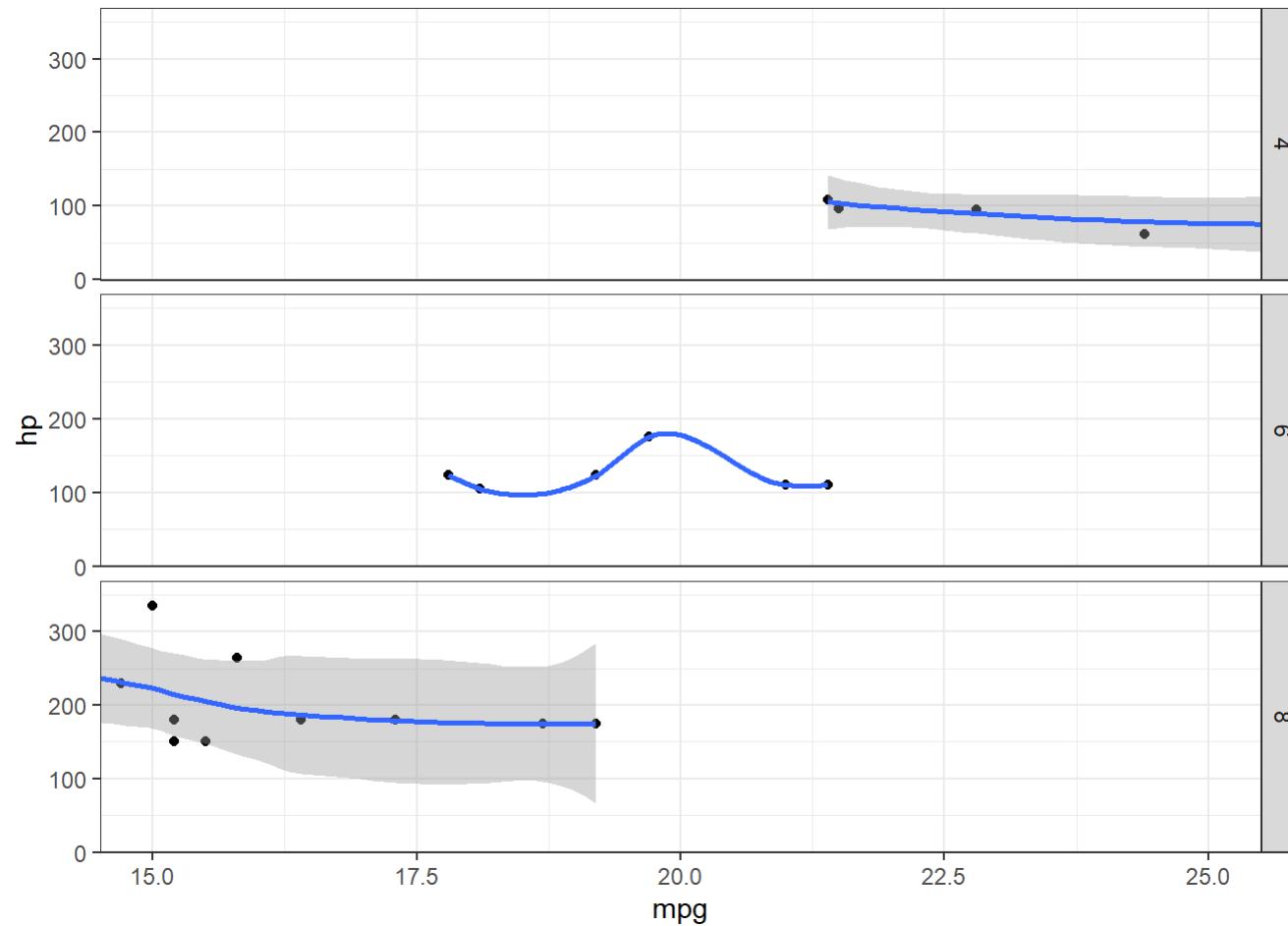
```
## Adding a smooth fit
pl + facet_grid(cyl~.) + stat_smooth() ## provides an error measure
```



```
## manipulating coordinates
pl2 <- pl + facet_grid(cyl~.)
pl2 + stat_smooth()
pl2 + coord_cartesian(xlim = c(15,25)) ## For bringing in and out of the axes
```

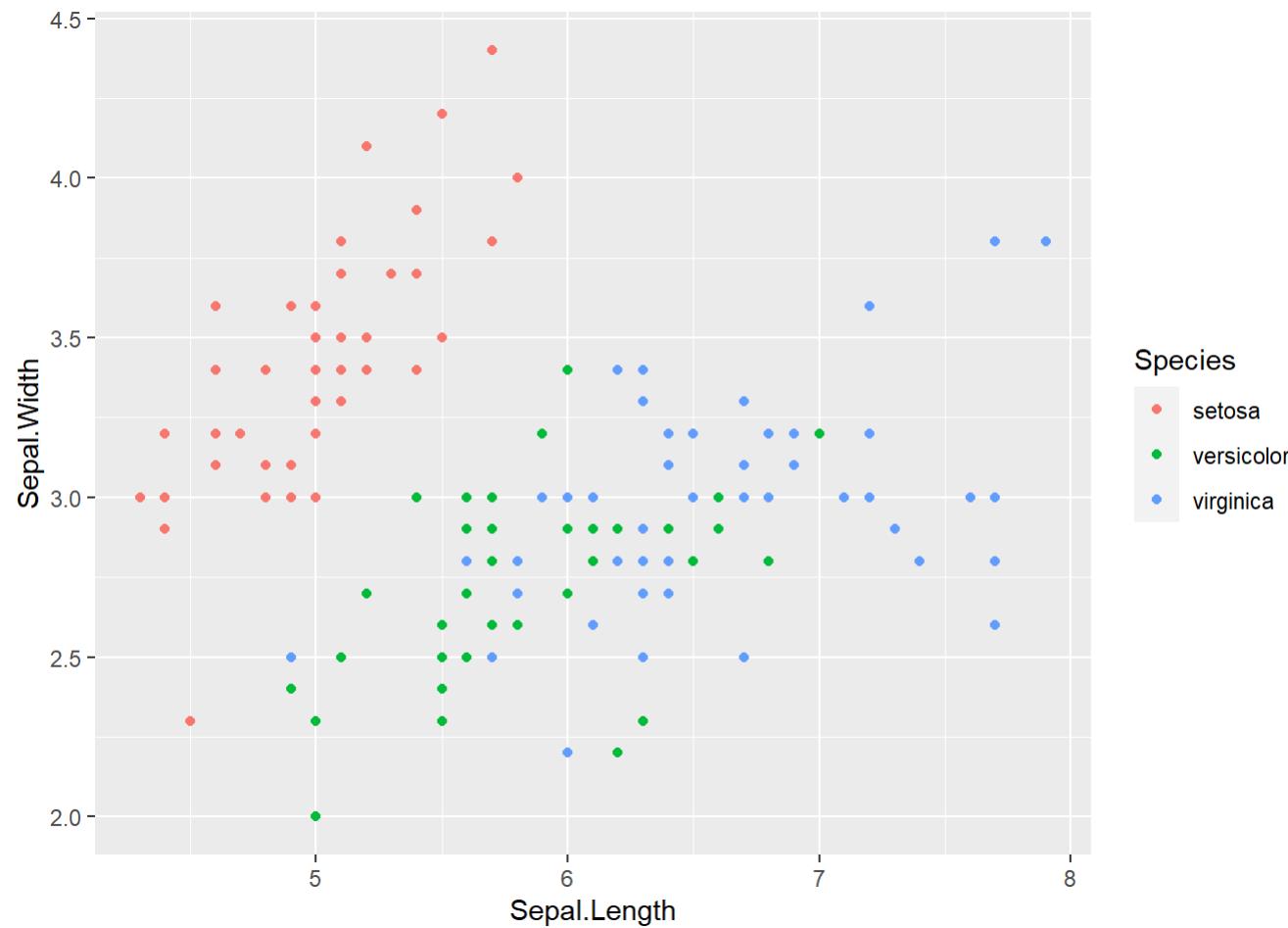


```
## Themes  
pl2 + coord_cartesian(xlim = c(15,25)) + theme_bw()
```



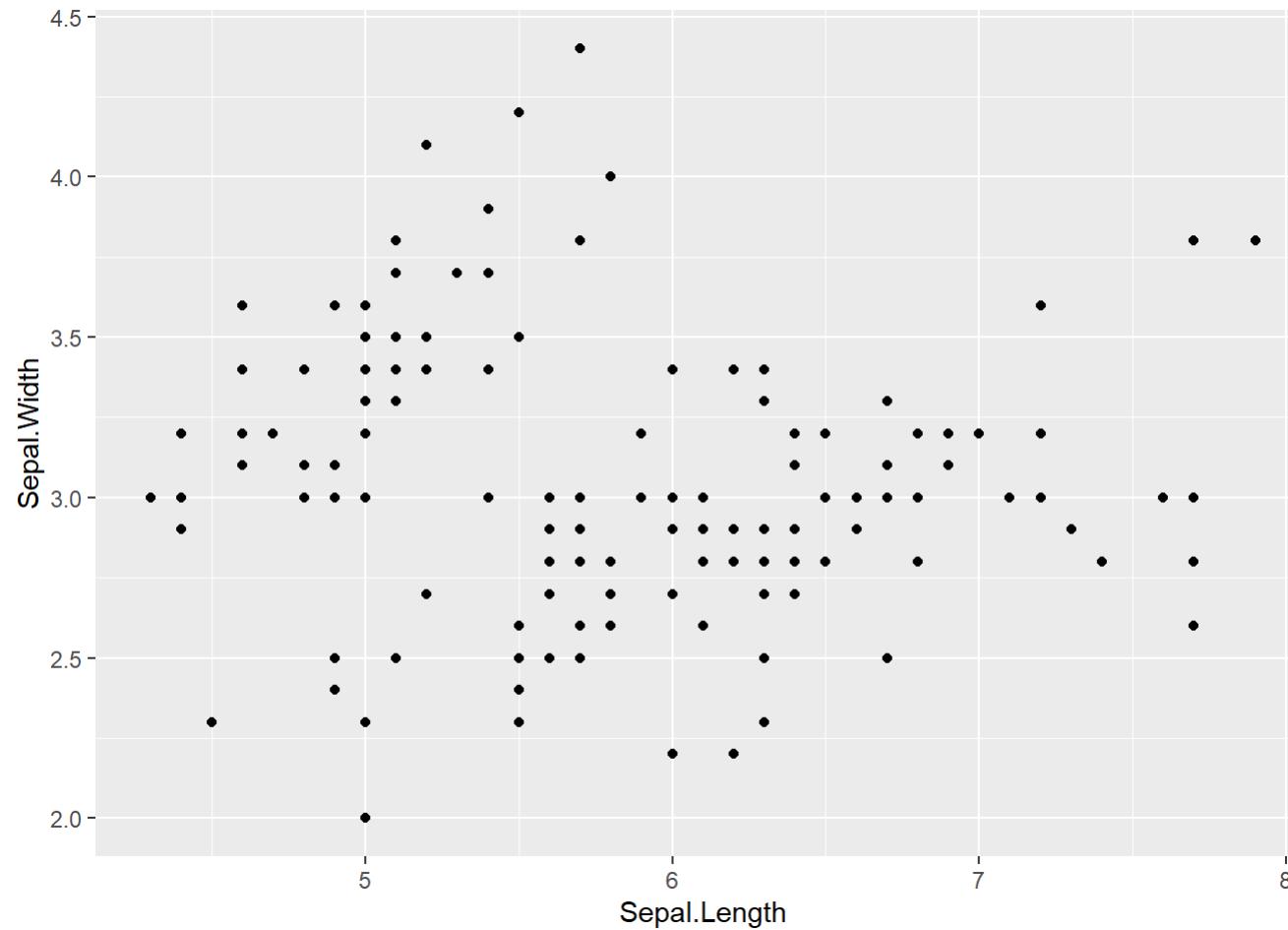
- **qplot**

```
qplot(Sepal.Length, Sepal.Width, data=iris, color = Species)
```

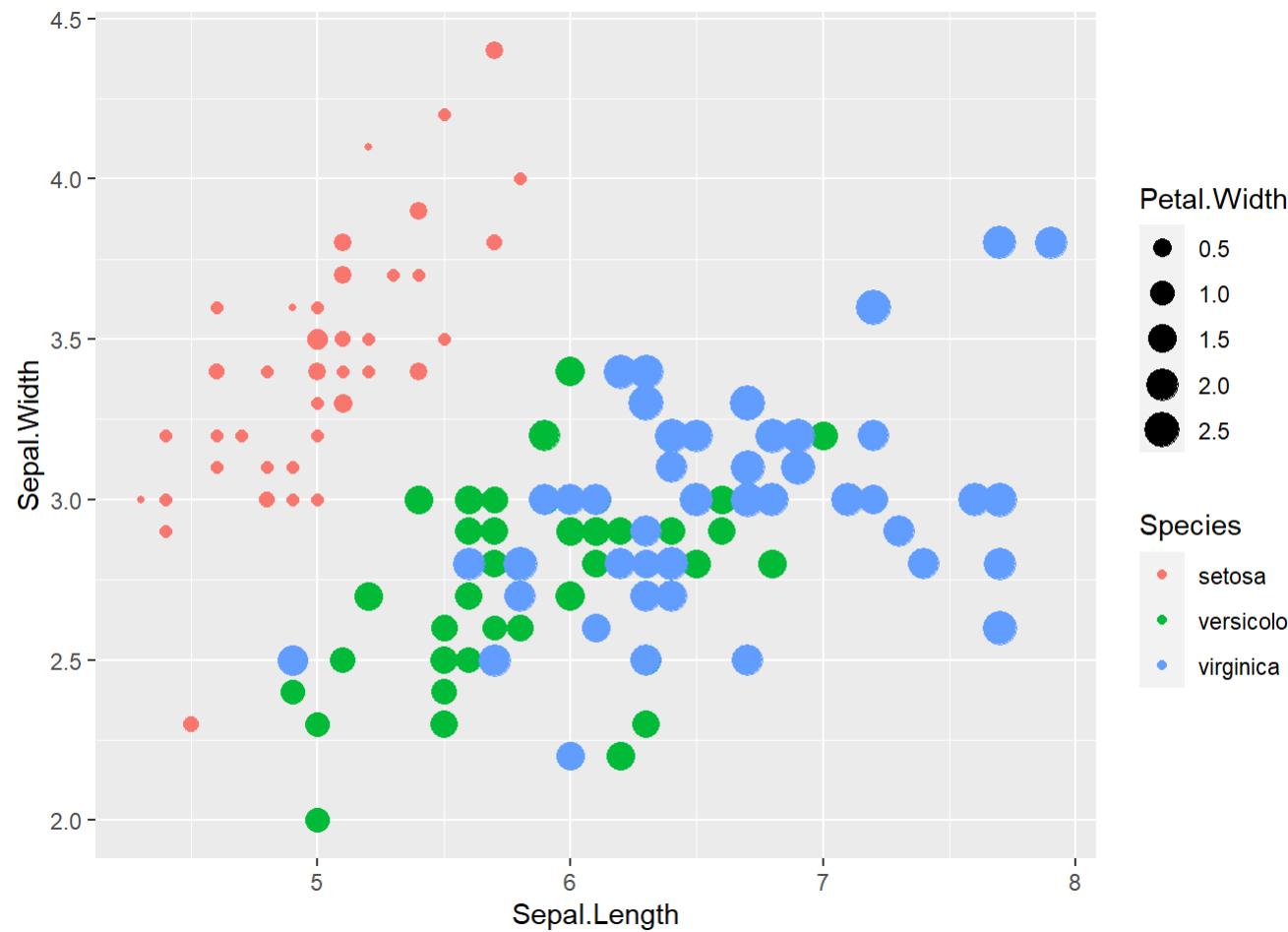


```
## Graph showing the relationship btw Length and width of 3 different species
```

```
qplot(Sepal.Length, Sepal.Width, data=iris)
```



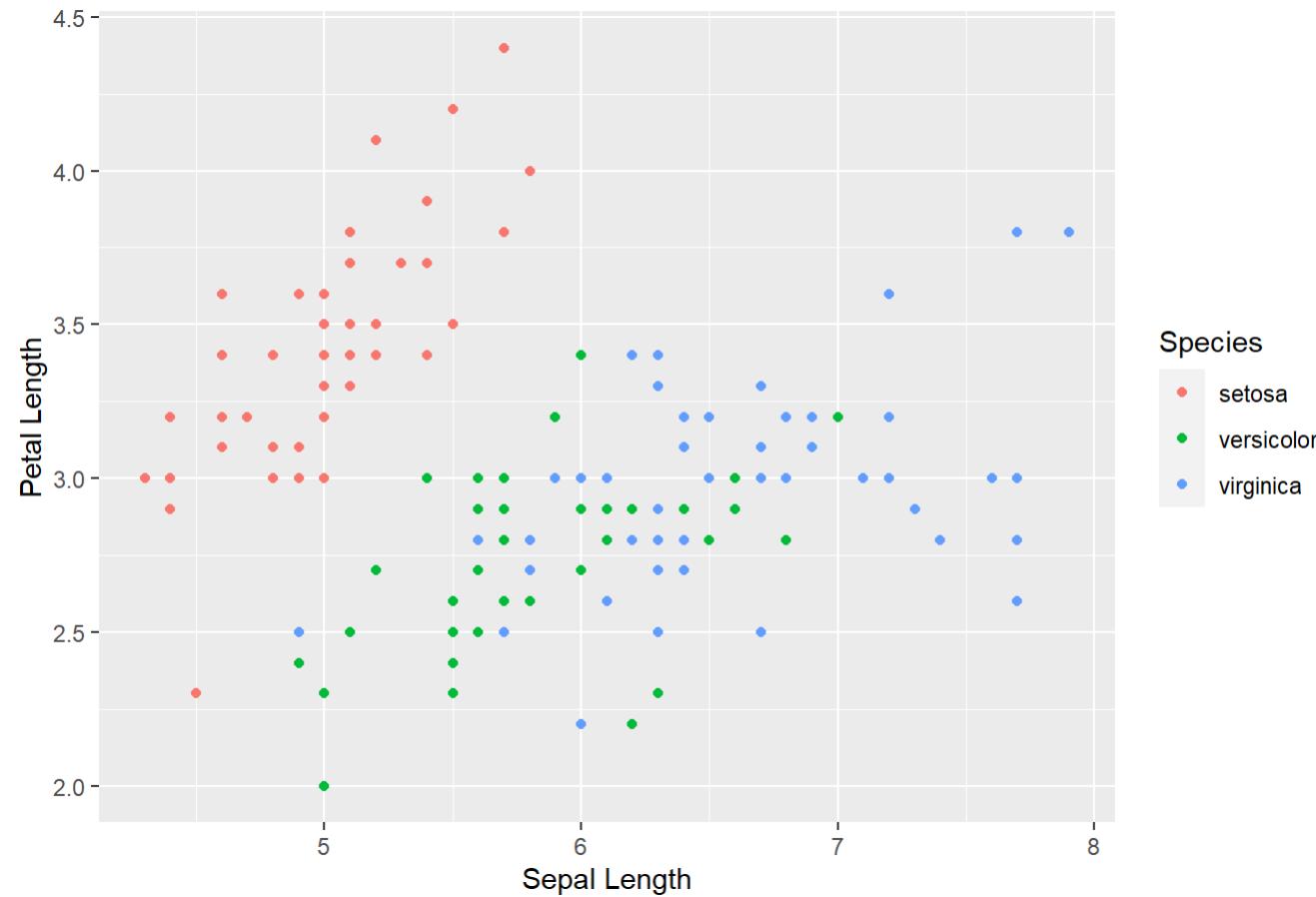
```
qplot(Sepal.Length, Sepal.Width, data=iris, color = Species, size = Petal.Width)
```



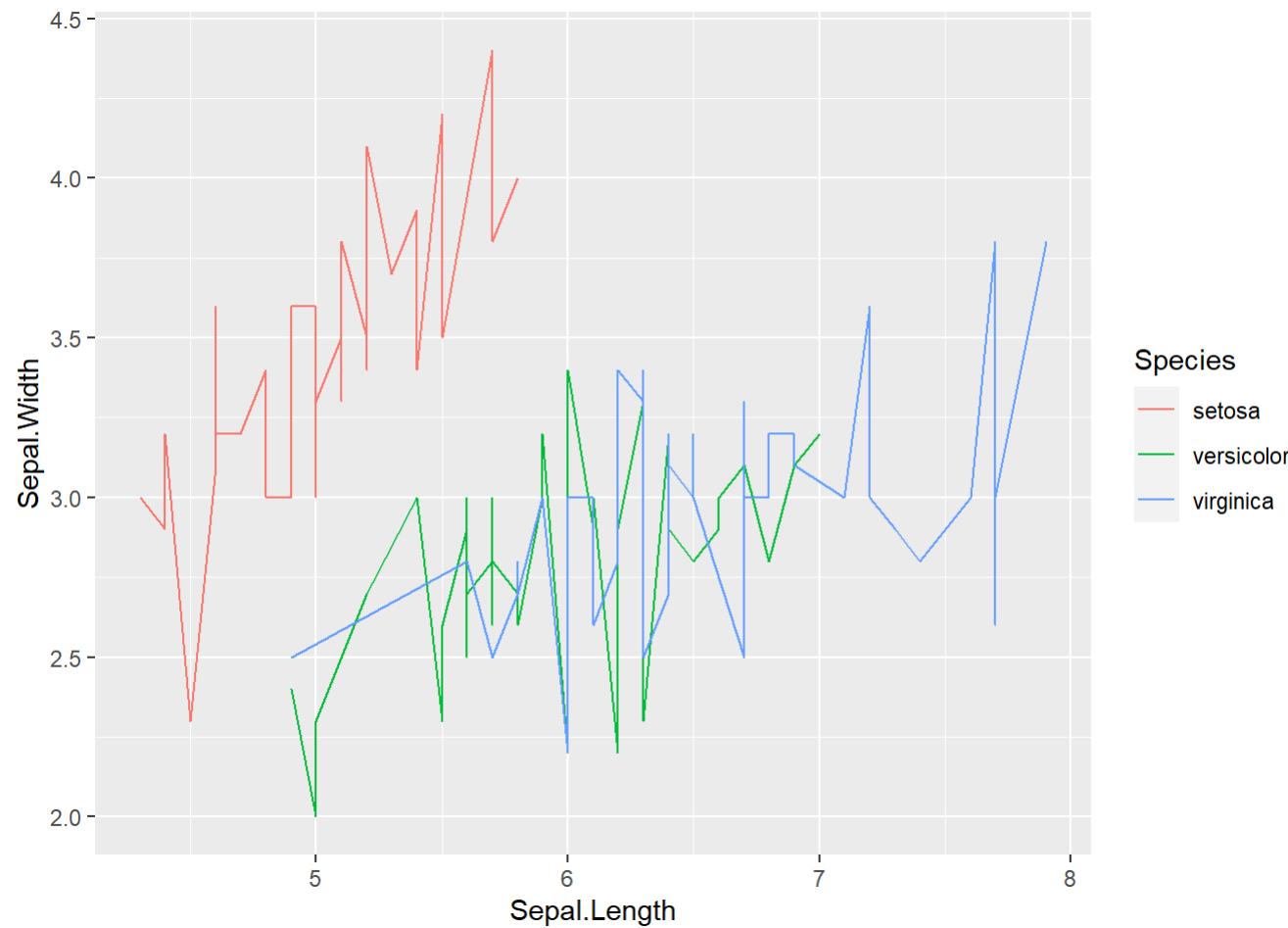
```
## The bases for size is Petal width. So, Setosa have the smallest.
```

```
##Adding labels to the plot
qplot(Sepal.Length, Sepal.Width, data=iris, color = Species,
      xlab = "Sepal Length", ylab = "Petal Length",
      main = "Sepal Vs Petal Length in Iris Dataset")
```

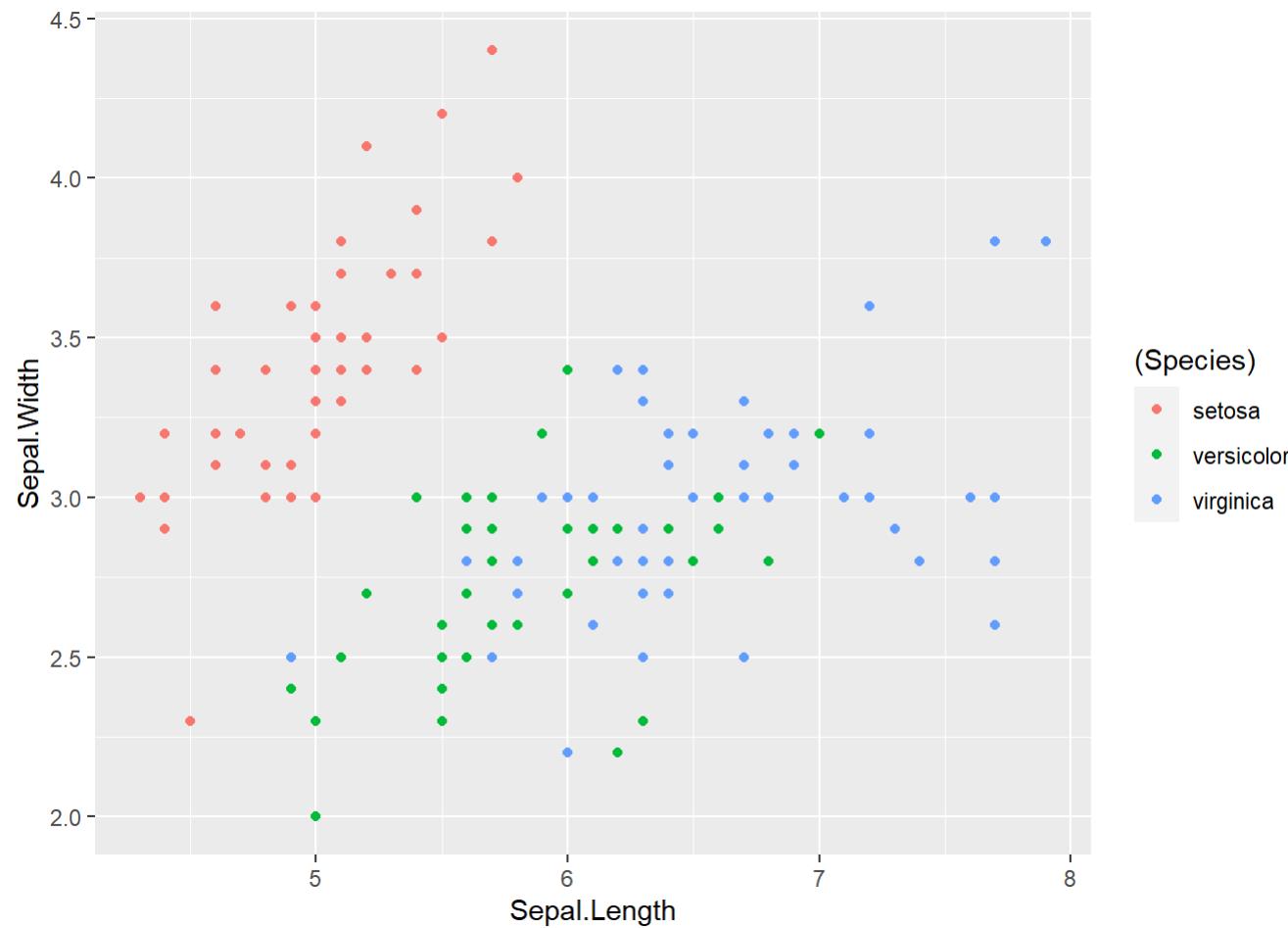
Sepal Vs Petal Length in Iris Dataset



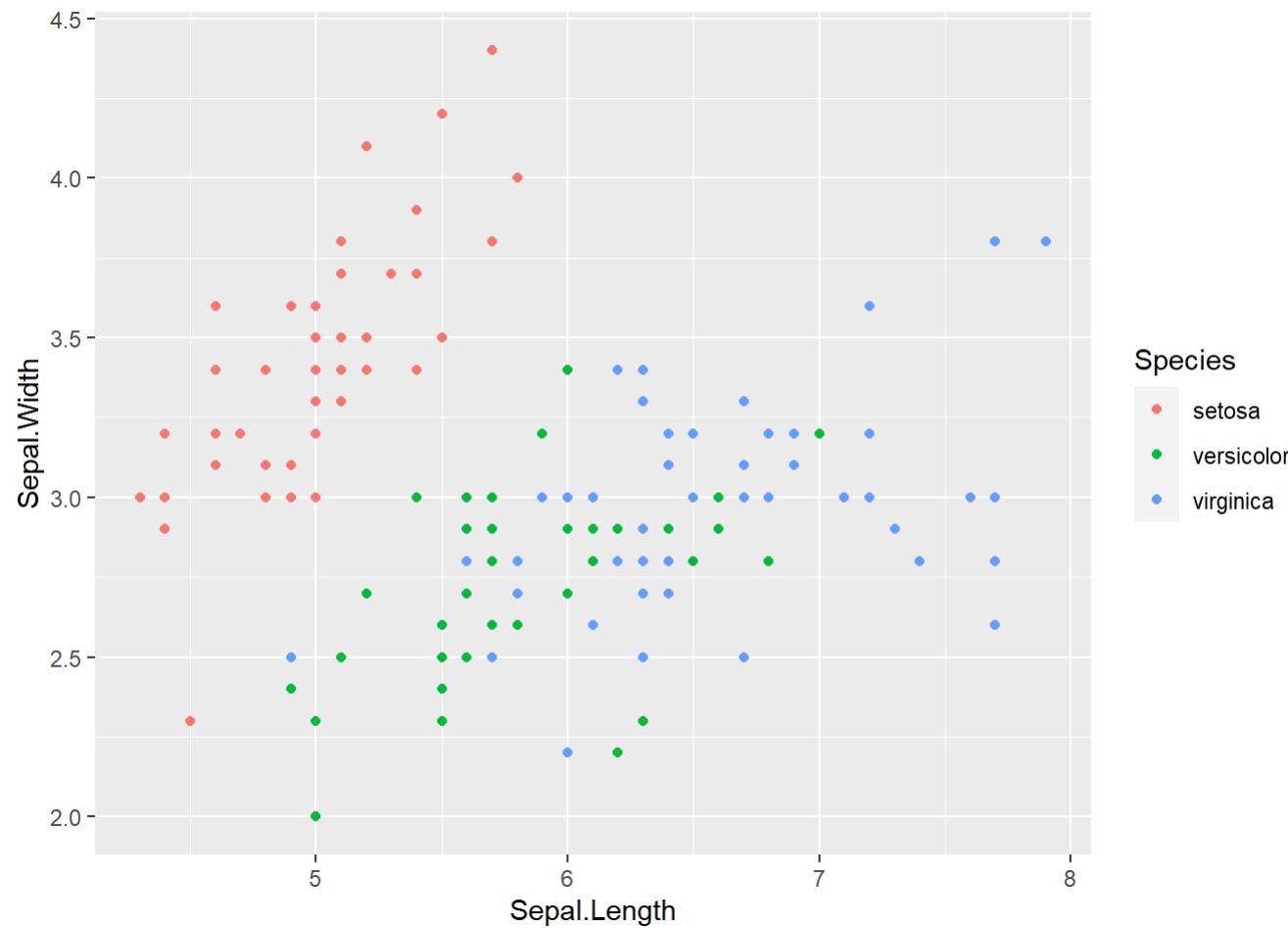
```
qplot(Sepal.Length, Sepal.Width, data=iris, geom = "line", color = Species)
```



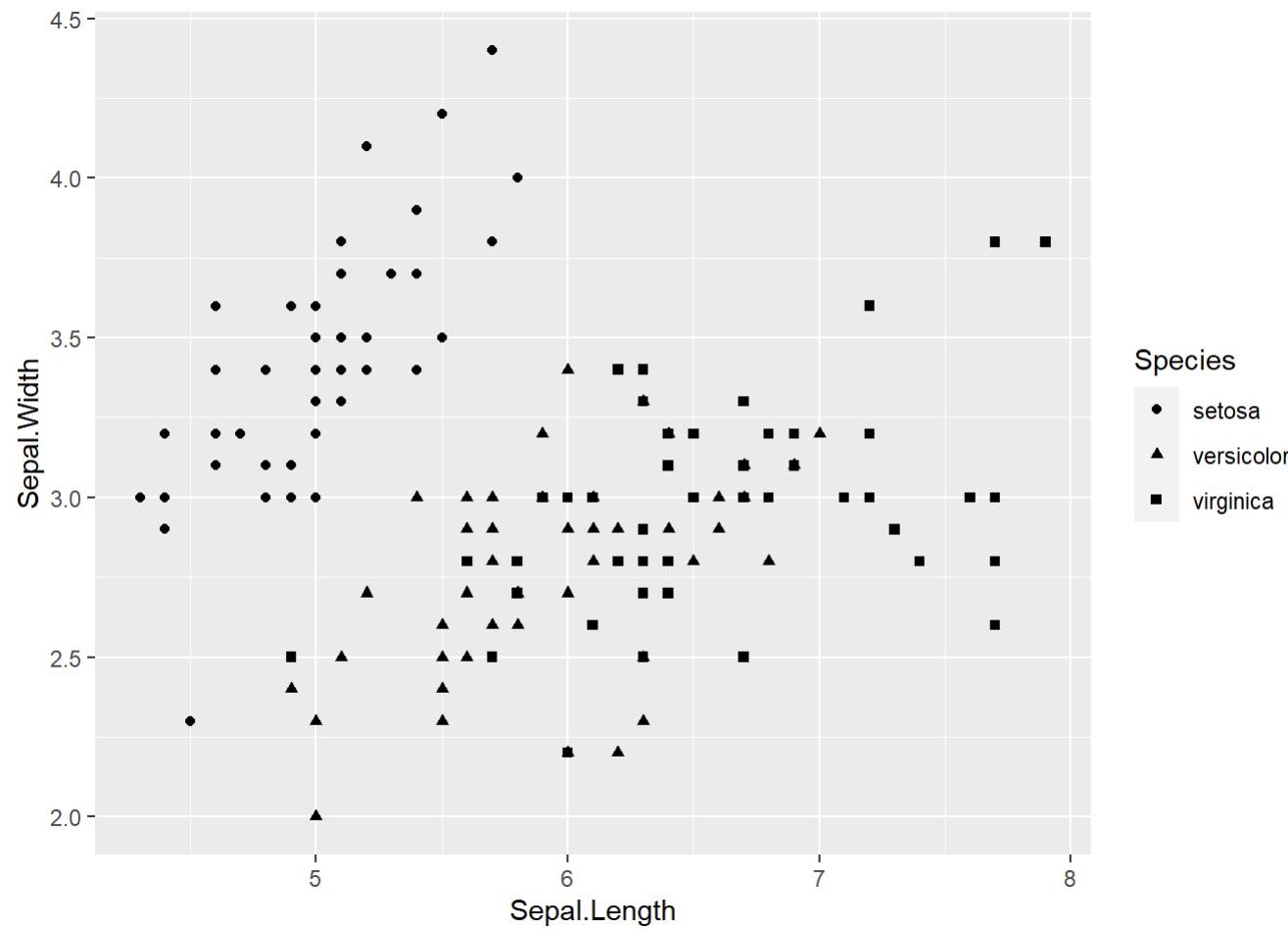
```
## Distinguish between species using color scheme  
ggplot(data = iris, aes(Sepal.Length, Sepal.Width)) + geom_point(aes(color = (Species)))
```



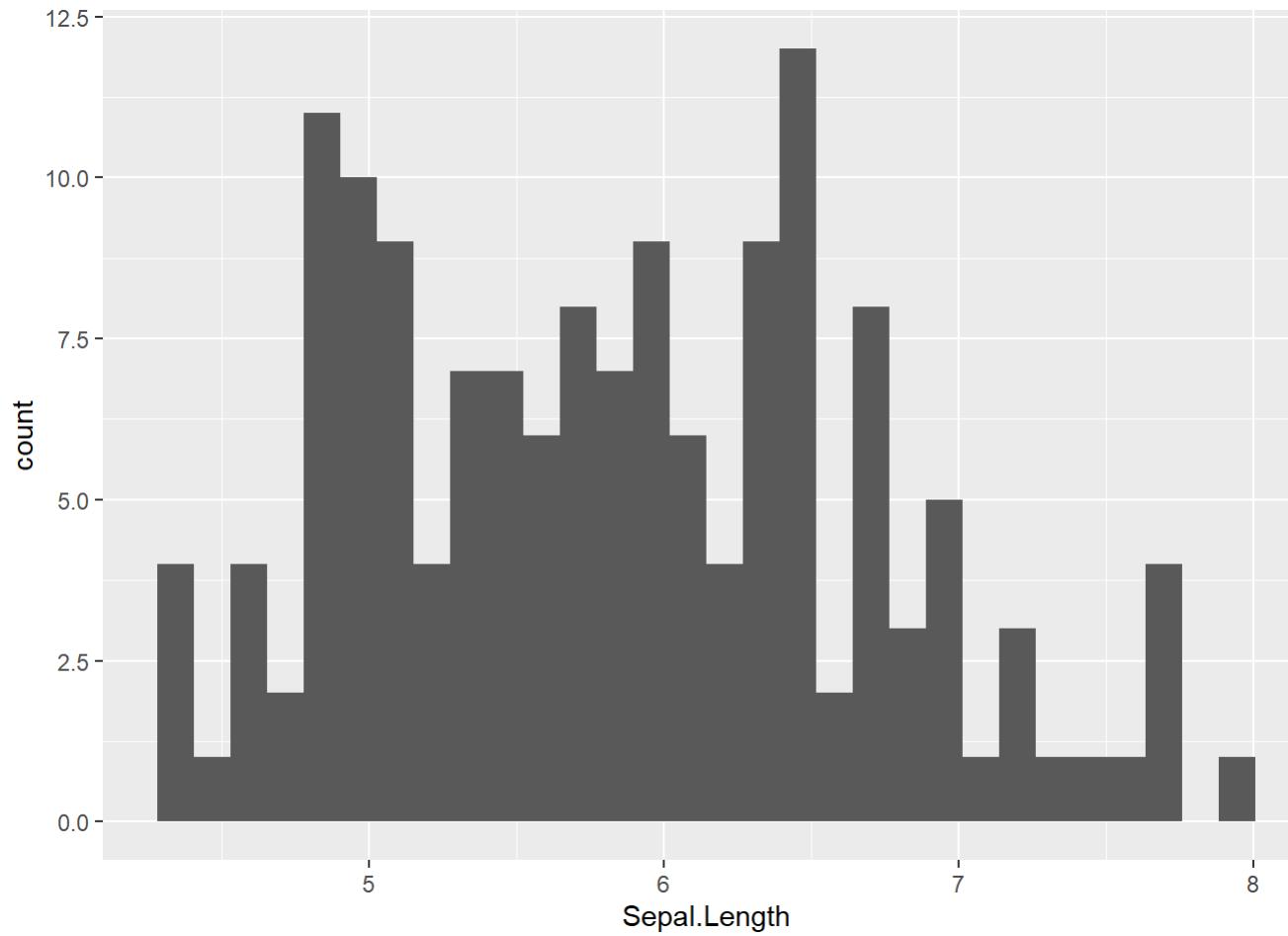
```
# OR  
ggplot(data = iris, aes(Sepal.Length, Sepal.Width, colour = Species)) + geom_point()
```



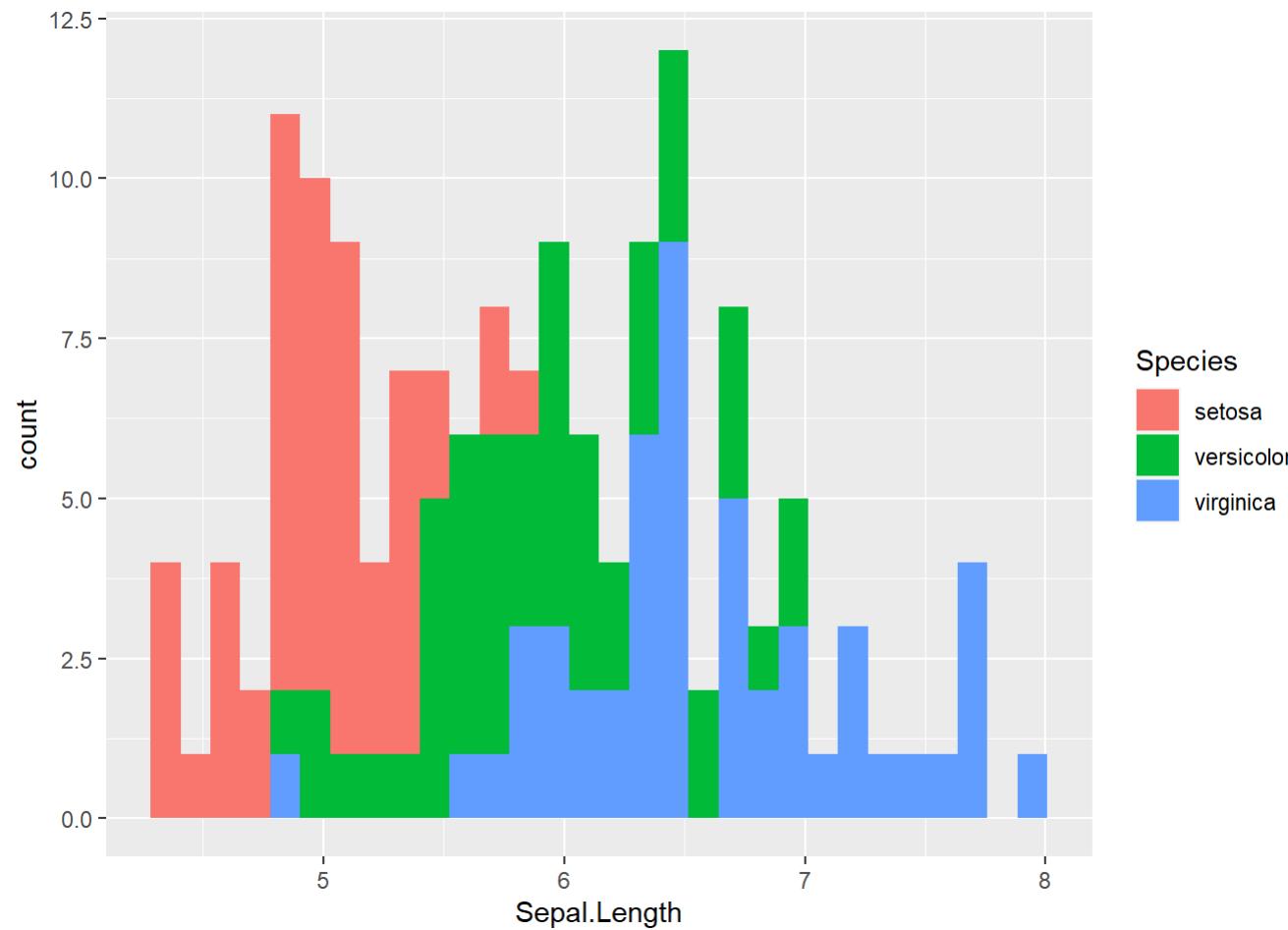
```
ggplot(data = iris, aes(Sepal.Length, Sepal.Width, shape = Species)) + geom_point()
```



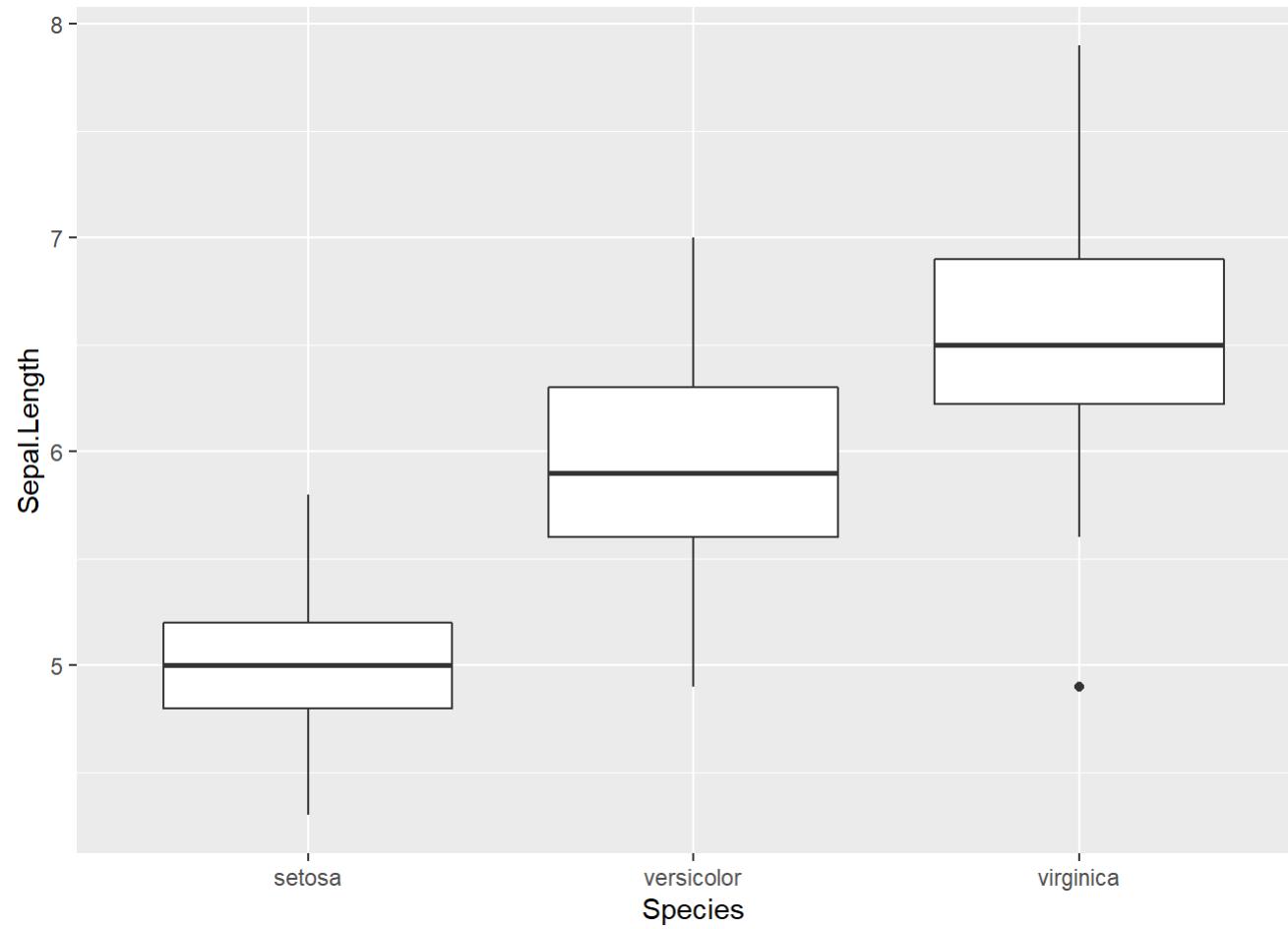
```
## Histogram  
ggplot(iris, aes(x= Sepal.Length)) +geom_histogram()
```



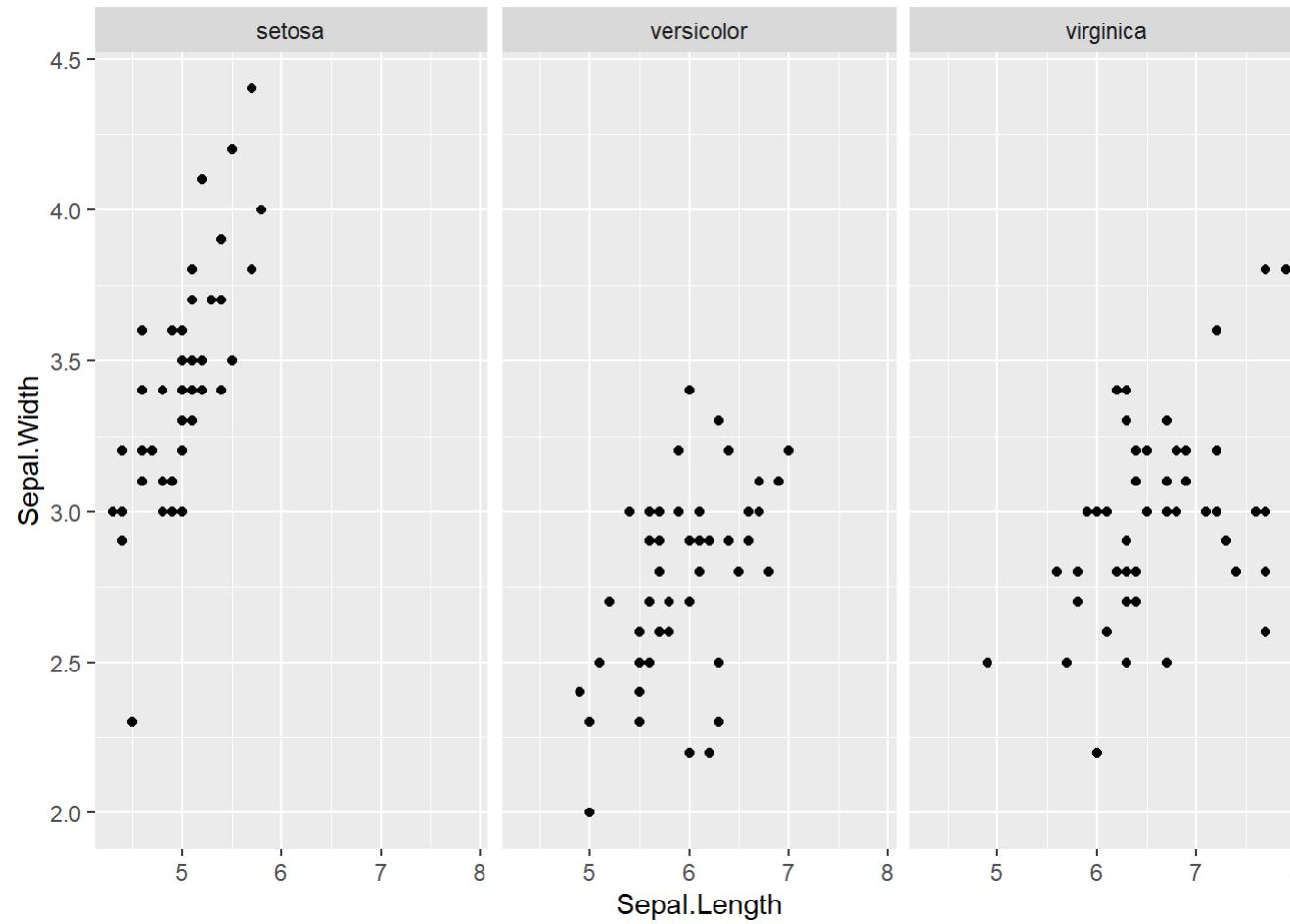
```
ggplot(iris, aes(x= Sepal.Length, fill = Species)) +geom_histogram()
```



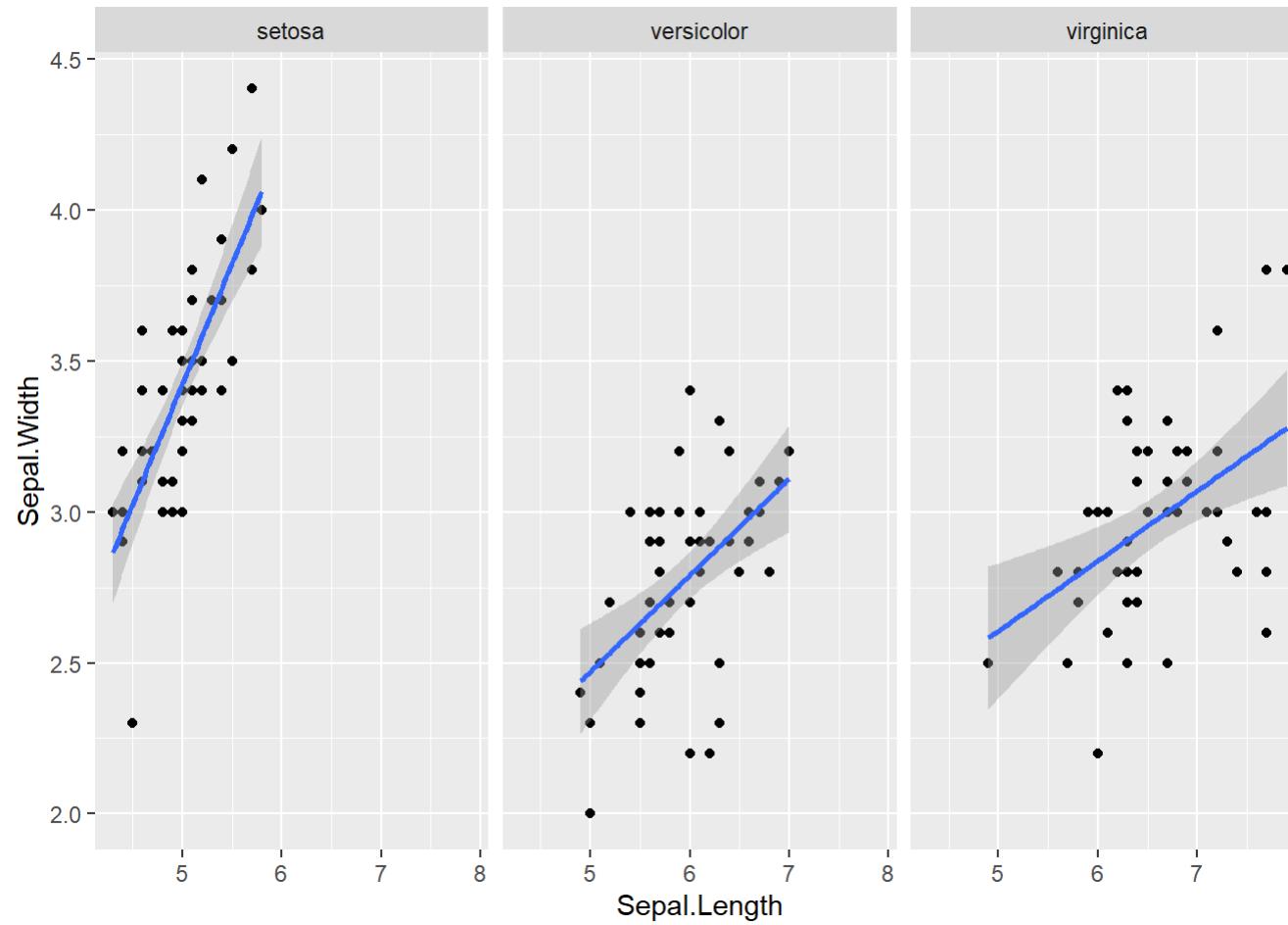
```
## Boxplot  
ggplot(iris, aes(x= Species, y= Sepal.Length)) + geom_boxplot()
```



```
## Visualizing relationship between variables for the 3 species  
ggplot(data = iris, aes(Sepal.Length, Sepal.Width)) + geom_point() + facet_grid(. ~ Species)
```



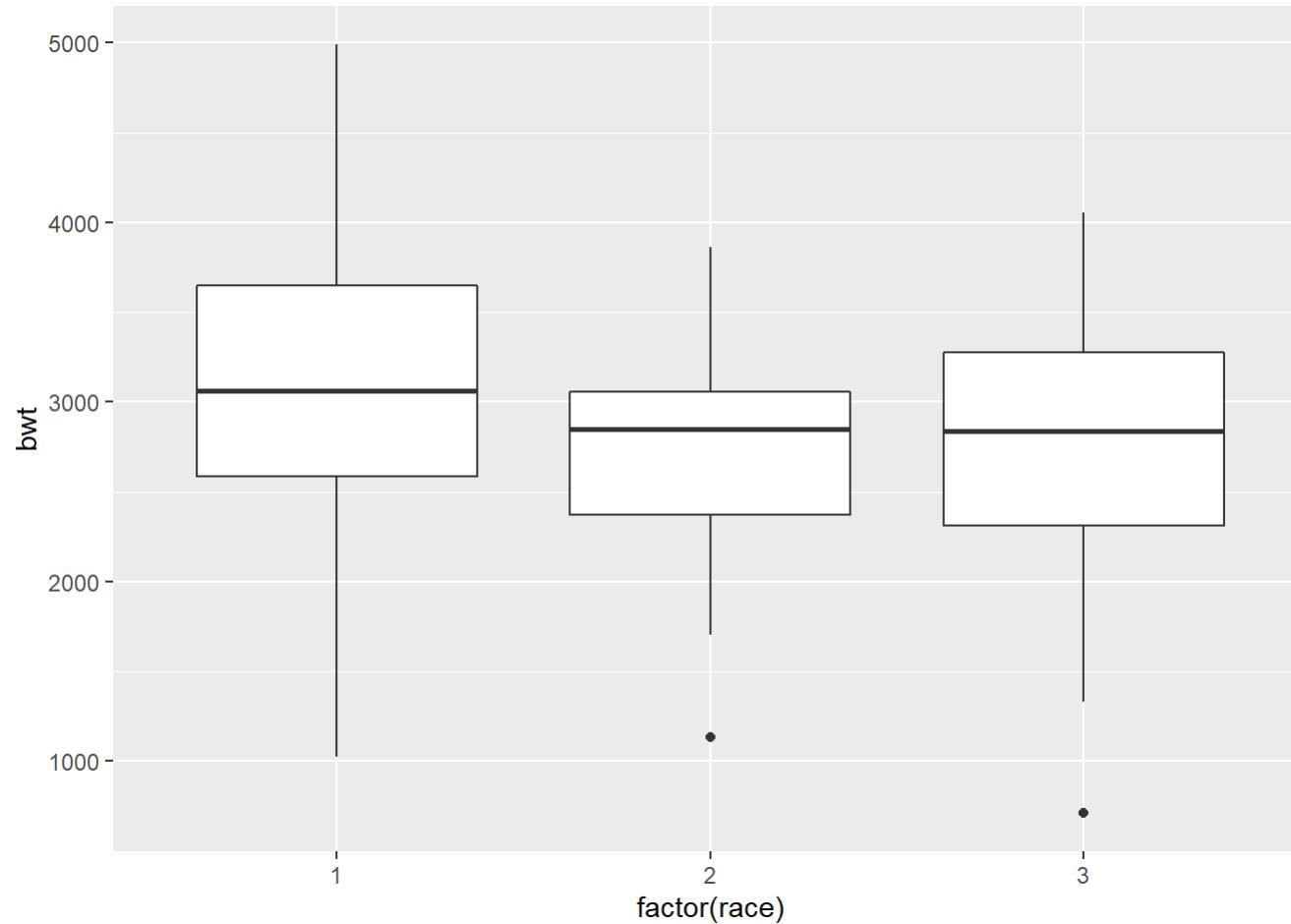
```
ggplot(data = iris, aes(Sepal.Length, Sepal.Width)) + geom_point() + facet_grid(. ~ Species) + geom_smooth(method = "lm")
```



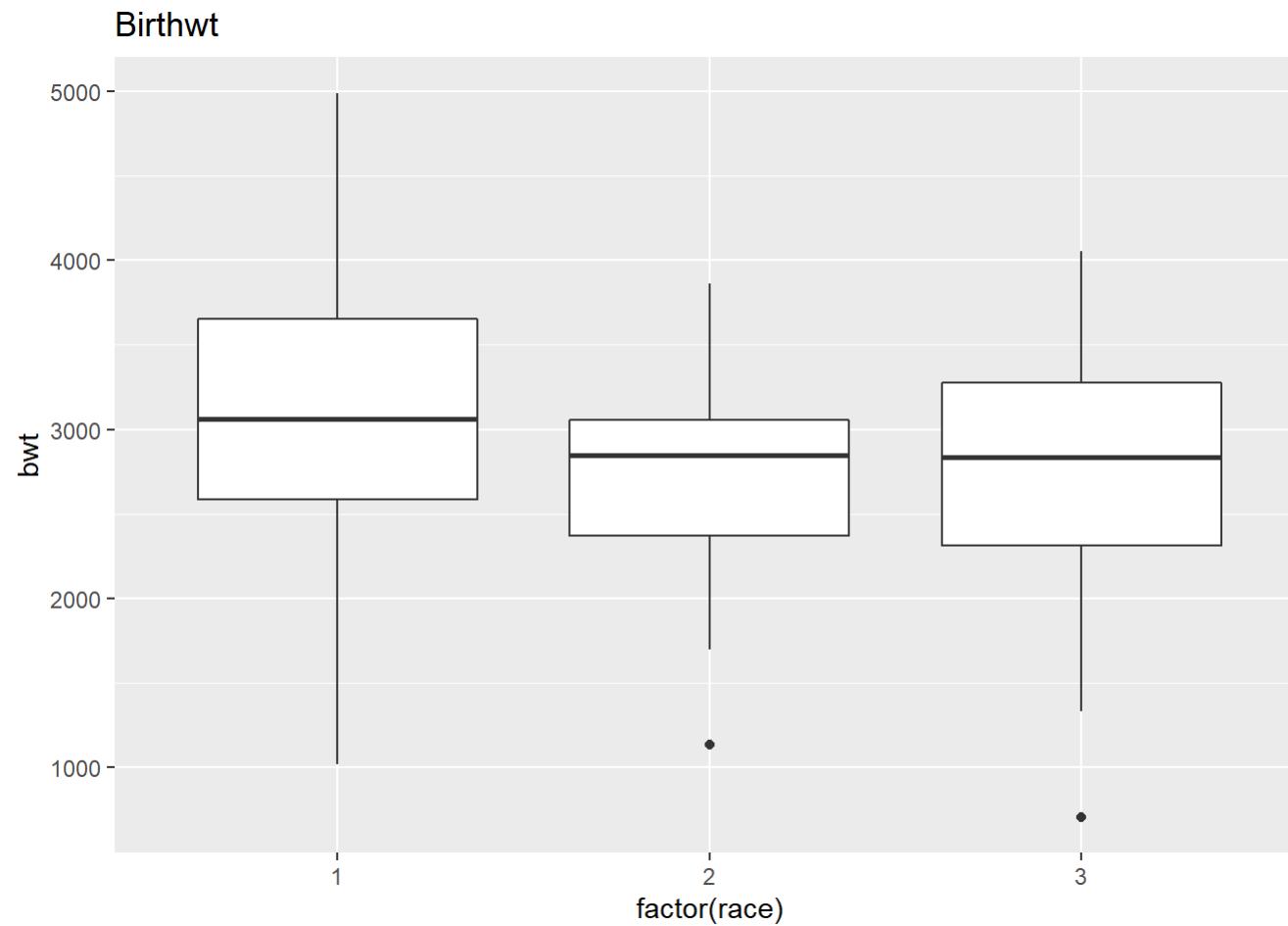
```
library(MASS)
head(birthwt)
```

	low	age	lwt	race	smoke	ptl	ht	ui	ftv	bwt
85	0	19	182	2	0	0	0	1	0	2523
86	0	33	155	3	0	0	0	0	3	2551
87	0	20	105	1	1	0	0	0	1	2557
88	0	21	108	1	1	0	0	1	2	2594
89	0	18	107	1	1	0	0	1	0	2600
91	0	21	124	3	0	0	0	0	0	2622

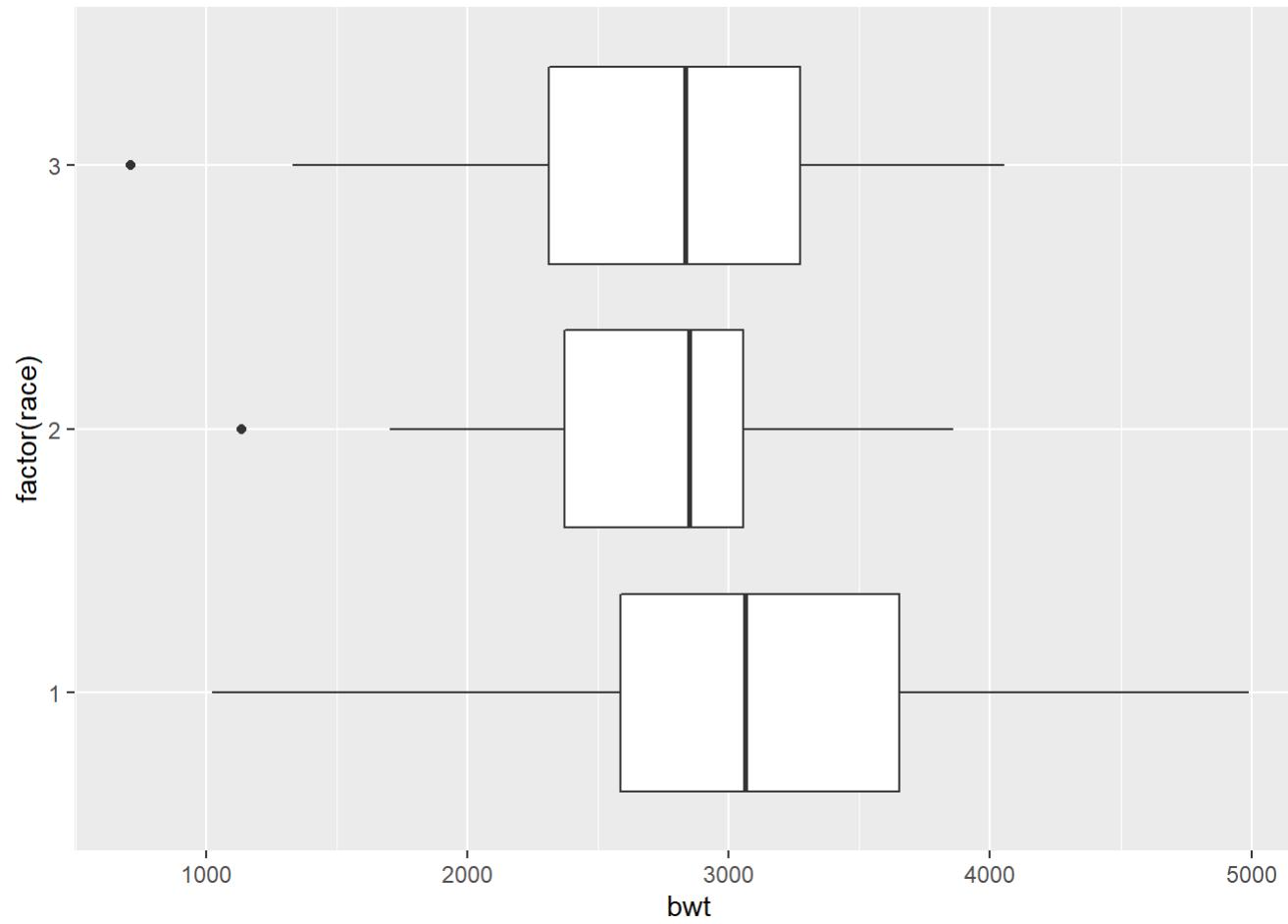
```
## compare numerical variables across all categories  
ggplot(birthwt, aes(x= factor(race), y= bwt)) +geom_boxplot()
```



```
## Do you notice the presence of an outlier?  
ggplot(birthwt, aes(x= factor(race), y= bwt)) +geom_boxplot() +ggtitle("Birthwt")
```



```
ggplot(birthwt, aes(x= factor(race), y= bwt)) +geom_boxplot() + coord_flip()
```



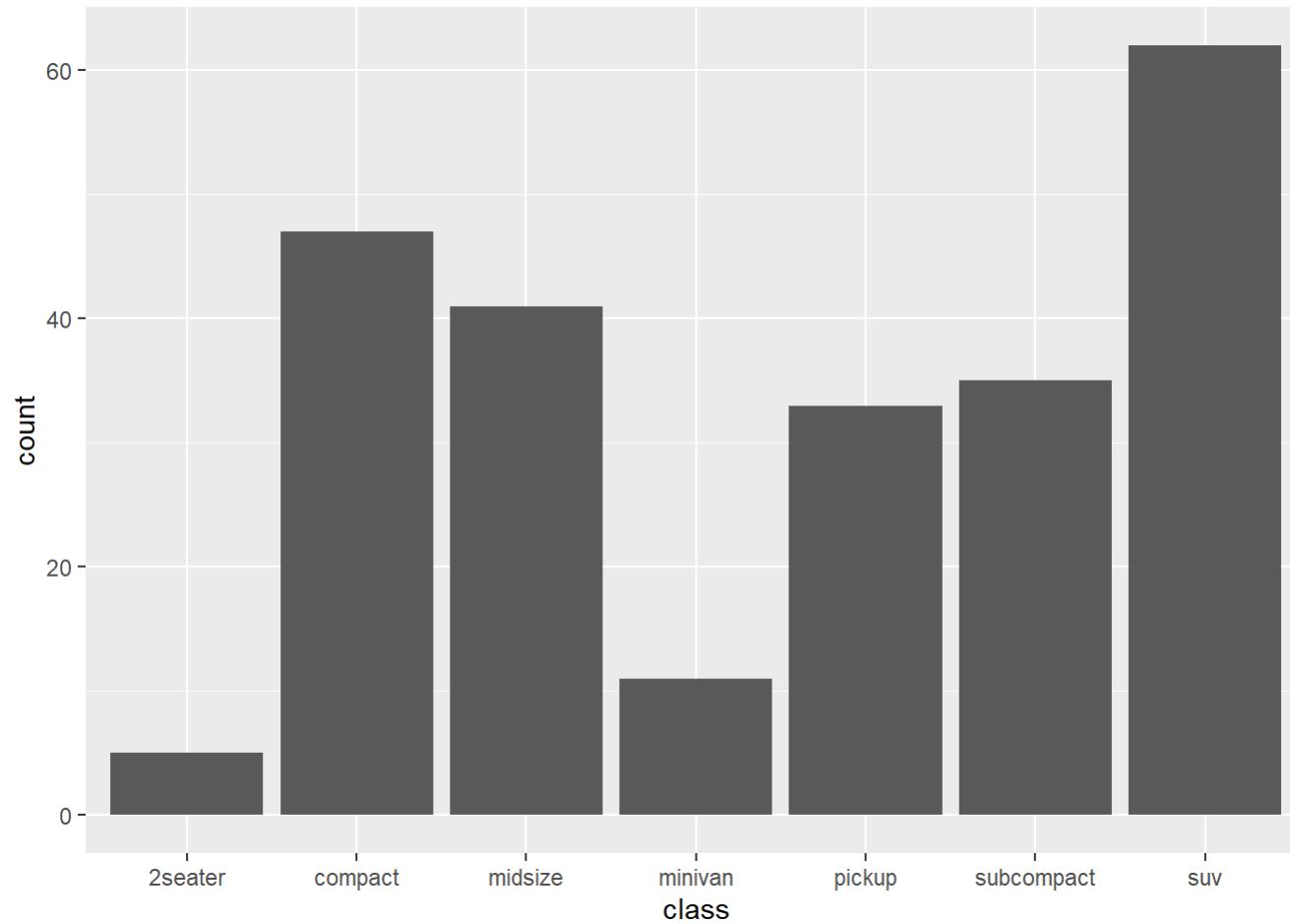
```
### BARPLOT
library(ggplot2)
df <- mpg
head(df)
```

```
# A tibble: 6 x 11
  manufacturer model displ year cyl trans     drv     cty   hwy fl class
  <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(l5) f      18    29 p    compa~
2 audi         a4      1.8  1999     4 manual(m5) f      21    29 p    compa~
3 audi         a4      2    2008     4 manual(m6) f      20    31 p    compa~
4 audi         a4      2    2008     4 auto(av)   f      21    30 p    compa~
5 audi         a4      2.8  1999     6 auto(l5)  f      16    26 p    compa~
6 audi         a4      2.8  1999     6 manual(m5) f      18    26 p    compa~
```

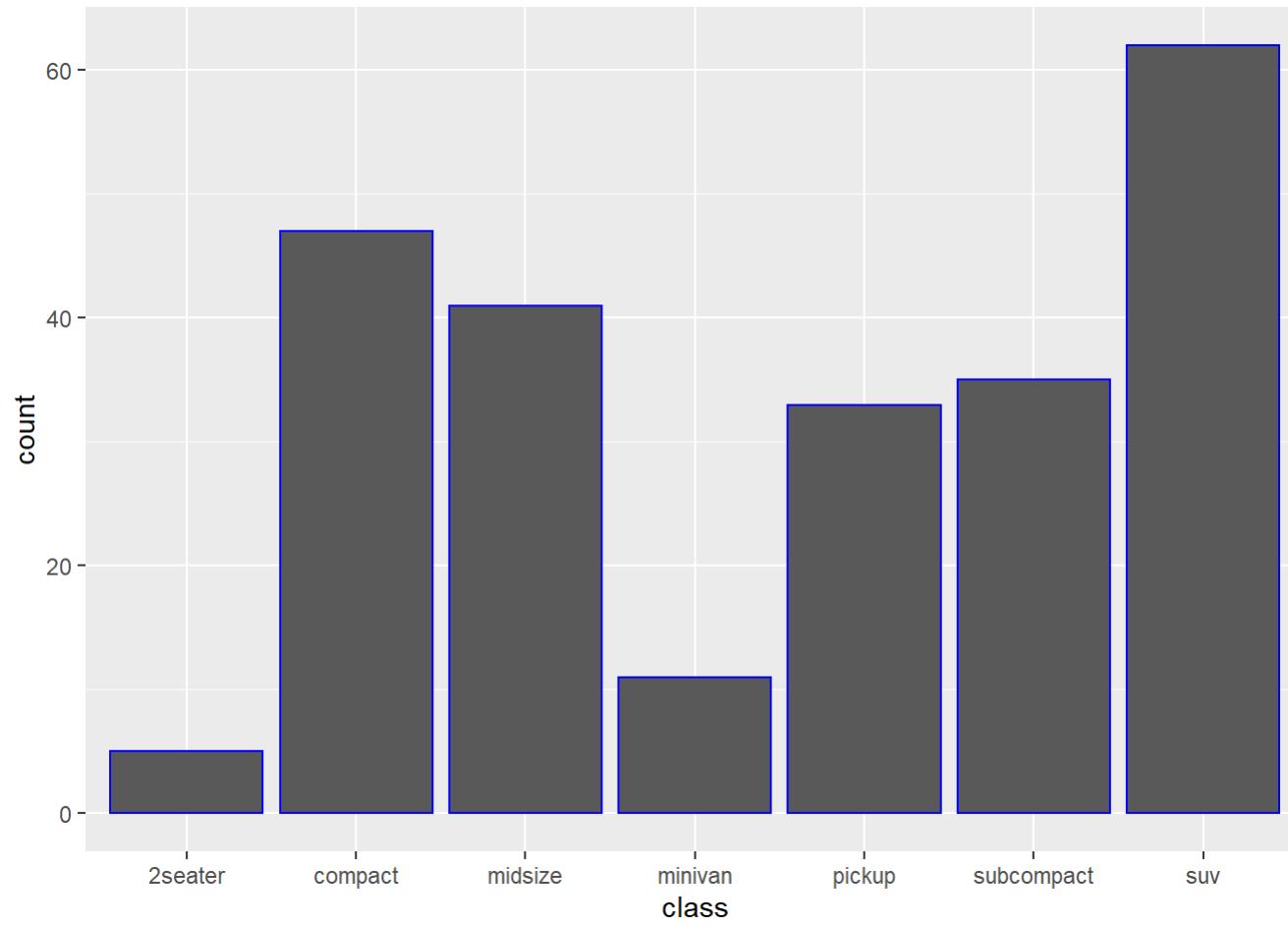
Scatterplot have continuous data on the x but barplot have a categorical data.

They both have counts on their y axis

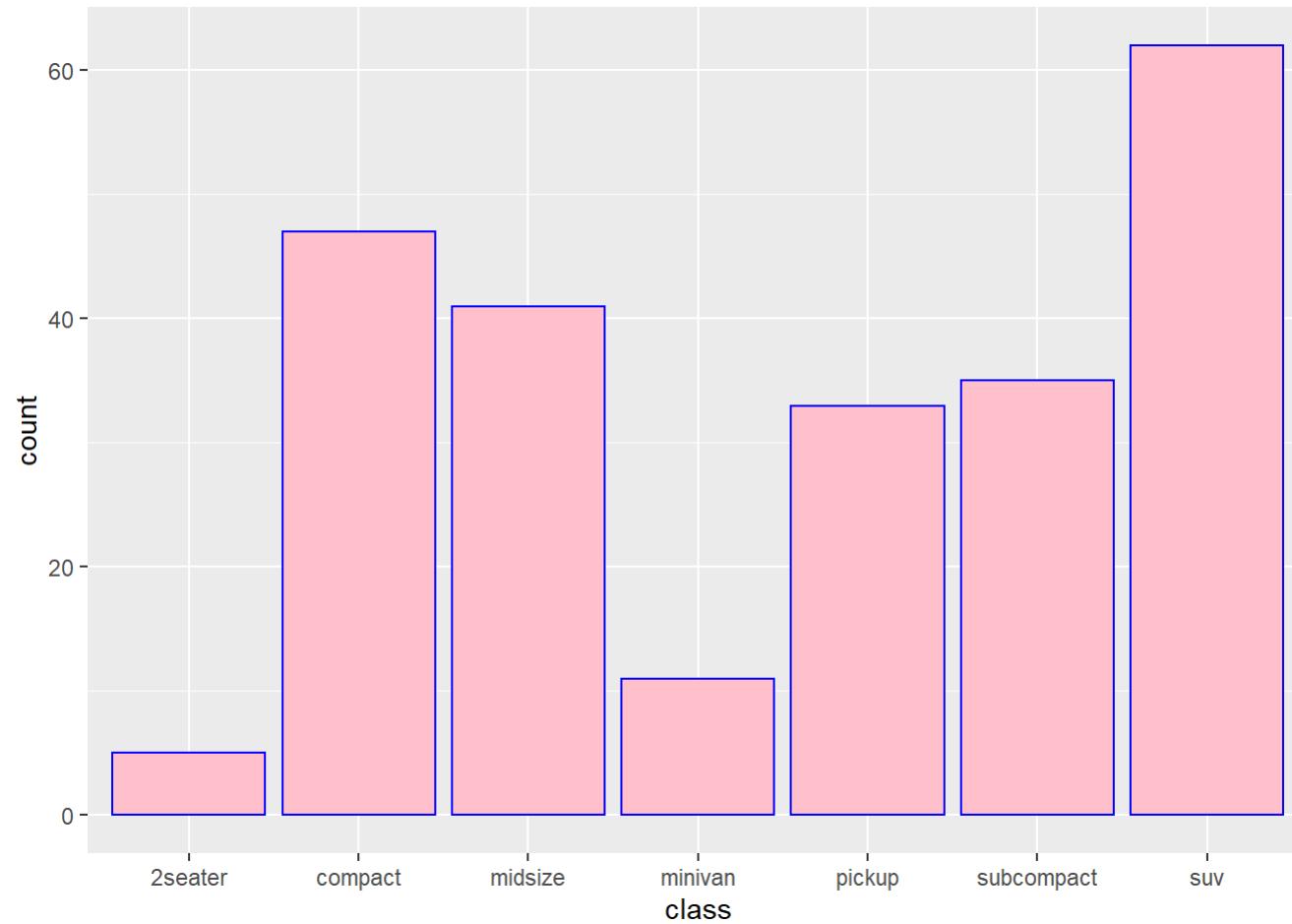
```
pl <- ggplot(df, aes(x=class))
print(pl + geom_bar())
```



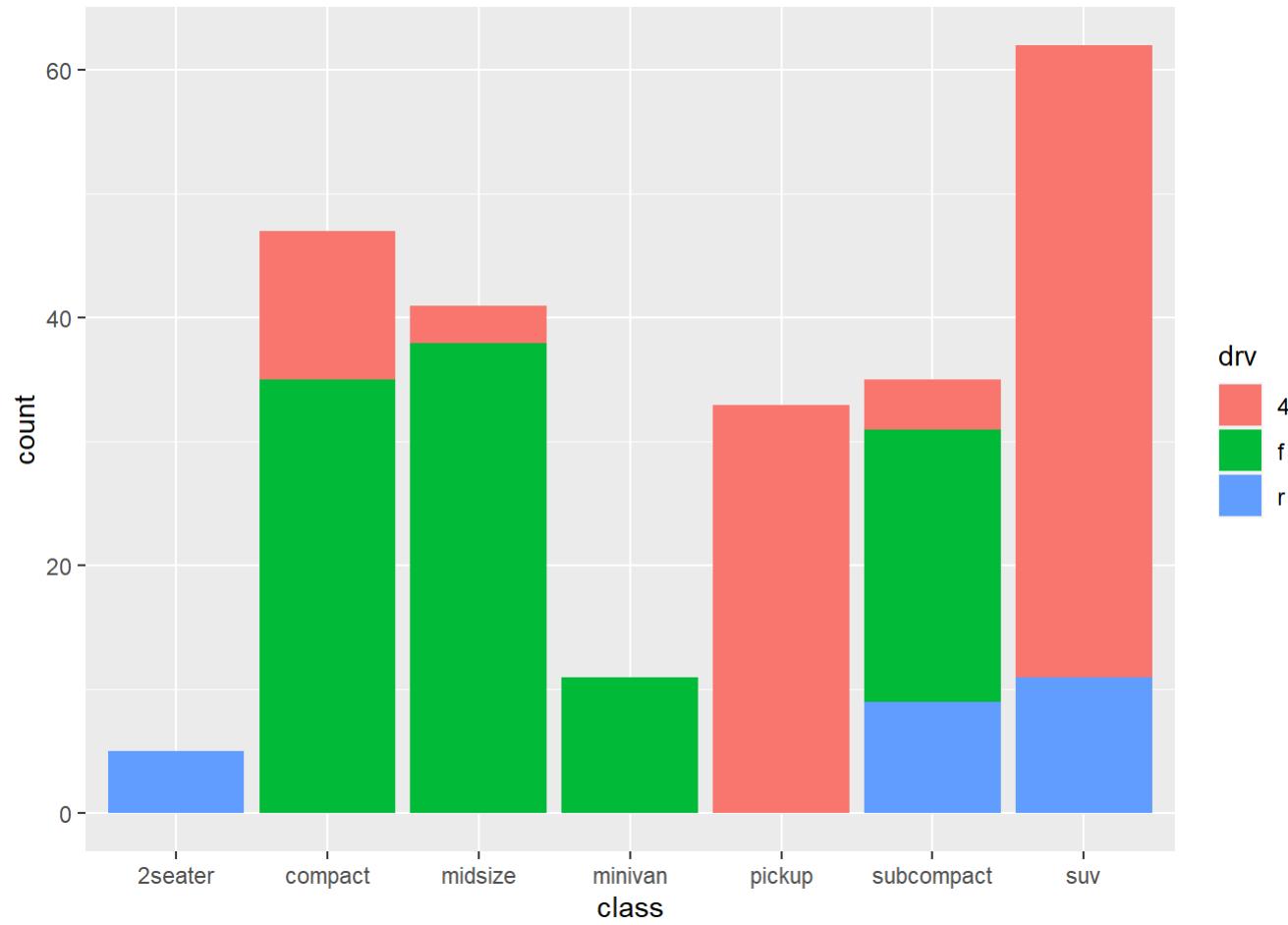
```
print(pl + geom_bar(color="blue")) ## for outline colour
```



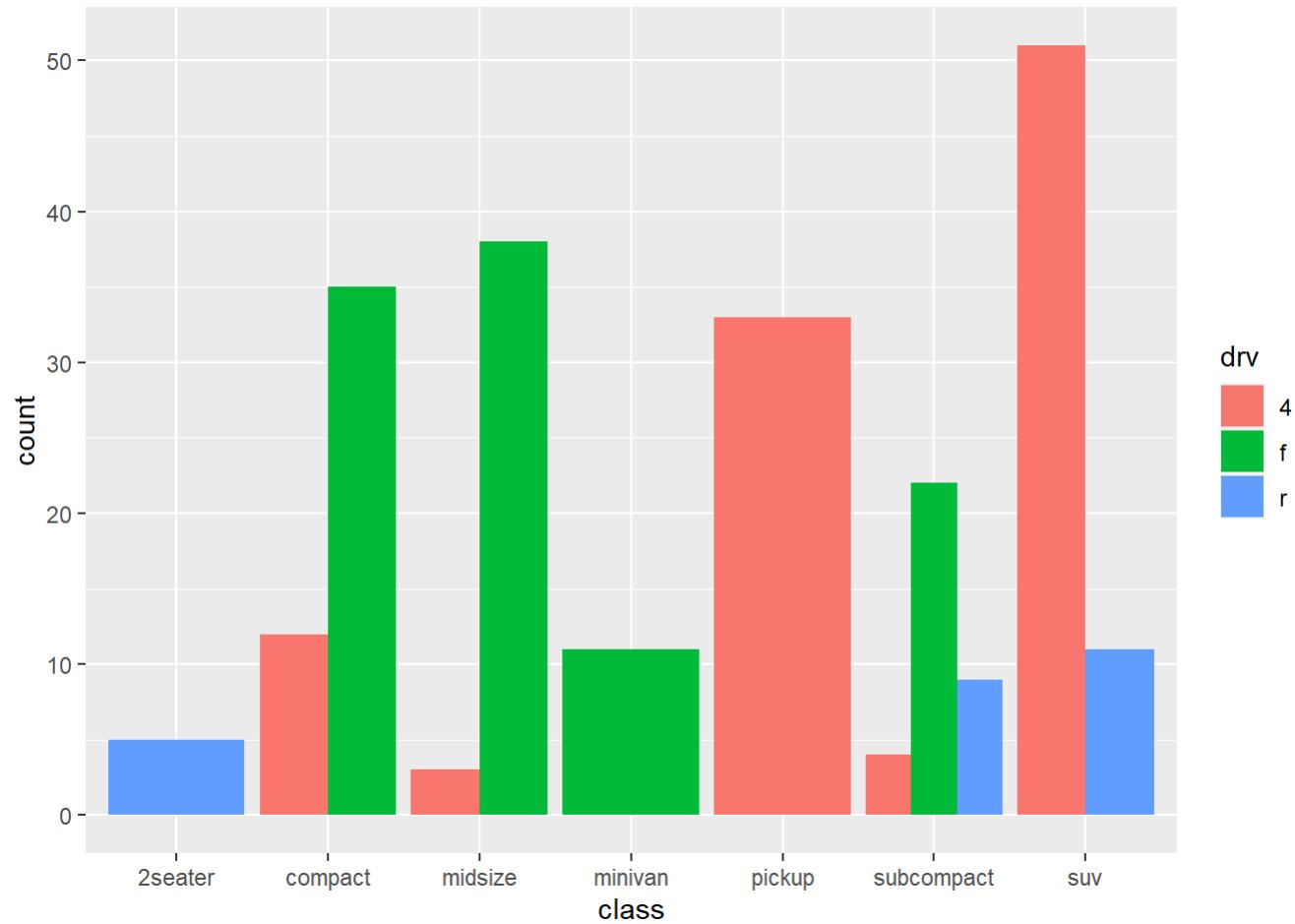
```
print(pl + geom_bar(color="blue", fill="pink"))
```



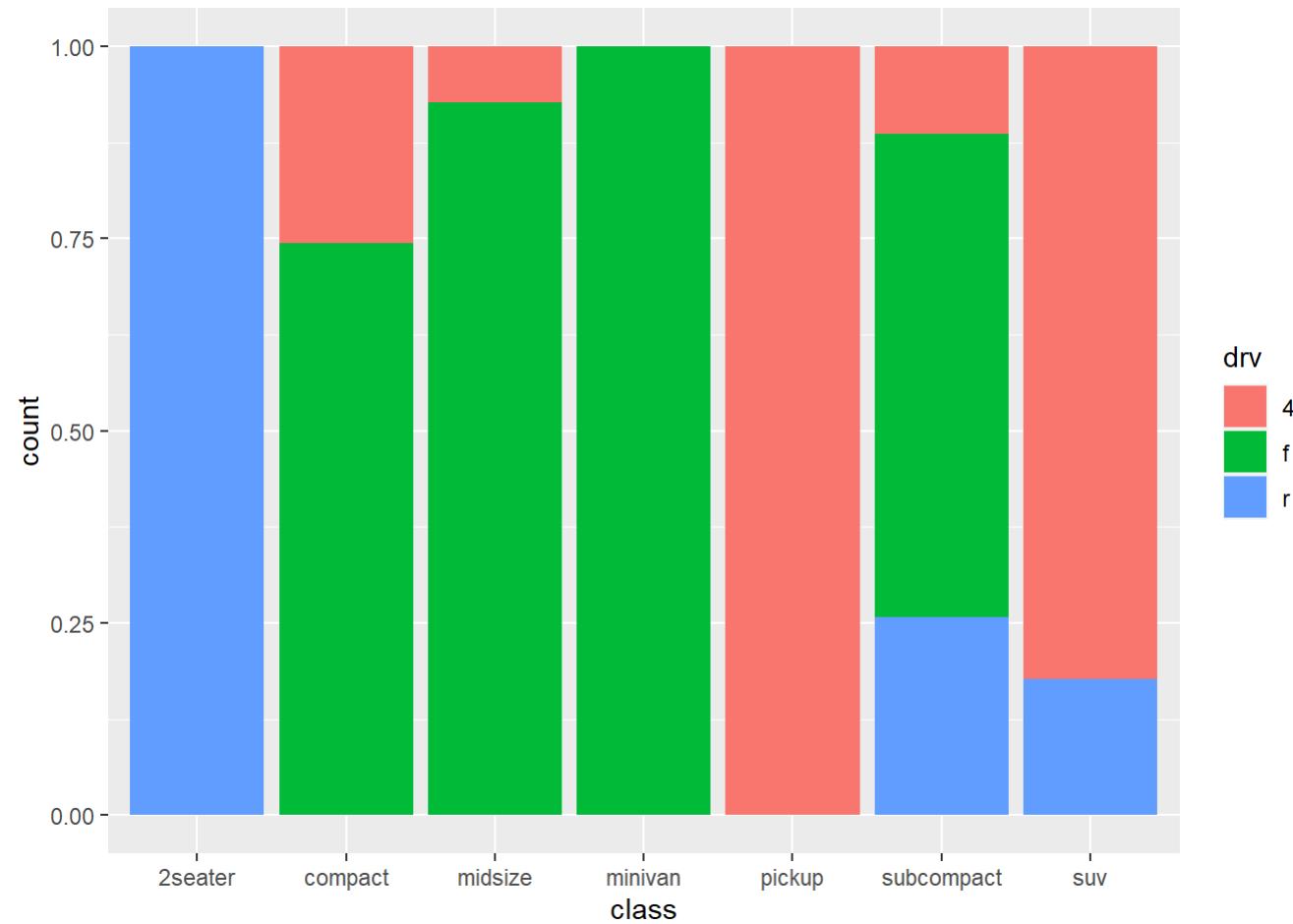
```
## We can also define our fill with another factor column in the data frame. To do this, we will introduce the aes  
(inside the geom)  
print(pl + geom_bar(aes(fill=drv))) ## This will automatically create a stacked bar for us.
```



```
print(pl + geom_bar(aes(fill=drv ), position = "dodge")) ## This helps in comparing
```



```
print(pl + geom_bar(aes(fill=drv ), position = "fill")) ## This shows the percentage instead of count
```



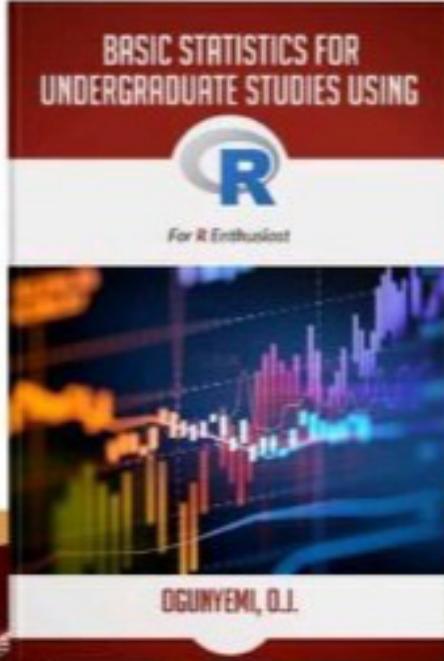
Conclusion

R is such an interesting programming language for statistical analysis, amazing visualization and reporting possibilities. To learn more, you need to read more and practice more.



Olakunle4impact

It's a great privilege.



The image shows the front cover and a portion of the back cover of a book titled "BASIC STATISTICS FOR UNDERGRADUATE STUDIES USING R". The front cover features a large blue "R" logo with the text "For R Enthusiast" below it. The background of the cover is a colorful, abstract chart with various data series. The author's name, "OGUNYEMI, O.I.", is printed at the bottom of the front cover. The back cover contains a brief introduction, the author's bio, and links to purchase the book.

BASIC STATISTICS FOR UNDERGRADUATE STUDIES USING R

OGUNYEMI, O.I.

This book is written for students who have basic programming skills in R. It is designed for students and researchers interested in learning how to use R very effectively for statistical analysis. It is intended to provide students with a very good understanding of the concepts of statistics and how to use R to implement them. It also provides students with the skills to use R effectively for statistical analysis.

Author's Bio:

Olakunle O.I. is a Lecturer in Statistics at the University of Lagos. He obtained his Bachelor's degree in Applied Mathematics from the University of Lagos and his Master's degree in Statistics from the University of Ibadan. He has also completed his Doctoral studies at the University of Lagos. He has published over 200 research publications, taught for organizations such as the Nigerian Statistical Institute, the Nigerian Institute of Economic Research, and the Central Bank of Nigeria, among others. He has also conducted several training programs for government officials and other professionals in the field of statistics.

Purchase Links:

www.amazon.com/dp/B08C4FTJFT 

Release date: 24th Sept. 2020

Enquires: Olakunle - 07063403997

₦2,500
(Soft Cover)

₦4,000
(Hard Cover)

 Free delivery within Lagos

Get your copy