R_notebook

Class 1 Introduction to R programming

What is R?

The common misconception is that R is a programming language but in fact it is much more than that. Think of R as an environment for statistical computing and graphics, which brings together a number of features to provide powerful functionality.

The R environment combines:

- effective handling of big data
- collection of integrated tools
- graphical facilities
- simple and effective programming language

Why use R?

R is a powerful, extensible environment. It has a wide range of statistics and general data analysis and visualization capabilities.

- Data handling, wrangling, and storage
- Wide array of statistical methods and graphical techniques available
- Easy to install on any platform and use (and it's free!)
- Open source with a large and growing community of peers

What is RStudio?

RStudio is freely available open-source Integrated Development Environment (IDE). RStudio provides an environment with many features to make using R easier and is a great alternative to working on R in the terminal.

- Graphical user interface, not just a command prompt
- Great learning tool
- Free for academic use
- Platform agnostic
- Open source

Creating a new project directory in RStudio

Let's create a new project directory for our "Introduction to R" lesson today.

1. Open RStudio
2. Go to the `File` menu and select `New Project`.
3. In the `New Project` window, choose `New Directory`. Then, choose `New Project`. Name your new directory `Intro-to-R` and then "Create the project as subdirectory of:" the Desktop (or location of your choice).
4. Click on `Create Project`.
5. After your project is completed, if the project does not automatically open in RStudio, then go to the `File` menu, select `Open Project`, and choose `Intro-to-R.Rproj`.

6. When RStudio opens, you will see three panels in the window.
7. Go to the `File` menu and select `New File`, and select `R Script`. The RStudio interface should now look like the screenshot below.

RStudio Interface

The RStudio interface has four main panels:

1. **Script editor**: where you can type out commands and save to file. You can also submit the commands to run in the console.
2. **Console**: where you can type commands and see output. *The console is all you would see if you ran R in the command line without RStudio.*
3. **Environment/History**: environment shows all active objects and history keeps track of all commands run in console
4. **Files/Plots/Packages/Help**

Assignment operator

To do useful and interesting things in R, we need to assign *values* to *variables* using the assignment operator, `<-`. For example, we can use the assignment operator to assign the value of `3` to `x` by executing:

```
x <- 3
```

The assignment operator (`<-`) assigns **values on the right** to **variables on the left**.

In RStudio, typing `Alt + -` (push `Alt` at the same time as the `-` key, on Mac type `option + -`) will write `<-` in a single keystroke.

Variables

A variable is a symbolic name for (or reference to) information. Variables in computer programming are analogous to "buckets", where information can be maintained and referenced. On the outside of the bucket is a name. When referring to the bucket, we use the name of the bucket, not the data stored in the bucket.

In the example above, we created a variable or a 'bucket' called `x`. Inside we put a value, `3`.

Let's create another variable called `y` and give it a value of 5.

```
y <- 5
```

When assigning a value to an variable, R does not print anything to the console. You can force to print the value by using parentheses or by typing the variable name.

```
y
```

You can also view information on the variable by looking in your `Environment` window in the upper right-hand corner of the RStudio interface.

Interacting with R

There are **two main ways** of interacting with R in RStudio: using the **console** or by using **script editor** (plain text files that contain your code).

Script editor

Best practice is to enter the commands in the **script editor**, and save the script. You are encouraged to comment liberally to describe the commands you are running using `#`. This way, you have a complete record of what you did, you can easily show others how you did it and you can do it again later on if needed.

The Rstudio script editor allows you to 'send' the current line or the currently highlighted text to the R console by clicking on the `Run` button in the upper-right hand corner of the script editor. Alternatively, you can run by simply pressing the `Ctrl` and `Enter` keys at the same time as a shortcut.

Now let's try entering commands to the **script editor** and using the comments character `#` to add descriptions and highlighting the text to run:

```
2+3
# use variable name  data set
x <- 2
y <- 5
# apply operation
x+y
```

Console window

You should see the command run in the console and output the result.

Output

```
> 2+3
[1] 5
> # use variable name fie data set
> x <- 2
> y <- 5
> # apply operation
> x+y
[1] 7
```

Packages

install packages from CRAN

install.packages("packagename")

load package to use in current R session

library(packagename)

Class 2 Import data from excel in R

Data Types

Variables can contain values of specific types within R. The six **data types** that R uses include:

- `"numeric"` for any numerical value
- `"character"` for text values, denoted by using quotes ("") around value
- `"integer"` for integer numbers (e.g., `2L`, the `L` indicates to R that it's an integer)
- `"logical"` for `TRUE` and `FALSE` (the Boolean data type)
- `"complex"` to represent complex numbers with real and imaginary parts (e.g., `1+4i`) and that's all we're going to say about them
- `"raw"` that we won't discuss further

Data Structures

We know that variables are like buckets, and so far we have seen that bucket filled with a single value. Even when `number` was created, the result of the mathematical operation was a single value. **Variables can store more than just a single value, they can store a multitude of different data structures.** These include, but are not limited to, vectors (`c`), factors (`factor`), matrices (`matrix`), data frames (`data.frame`) and lists (`list`).

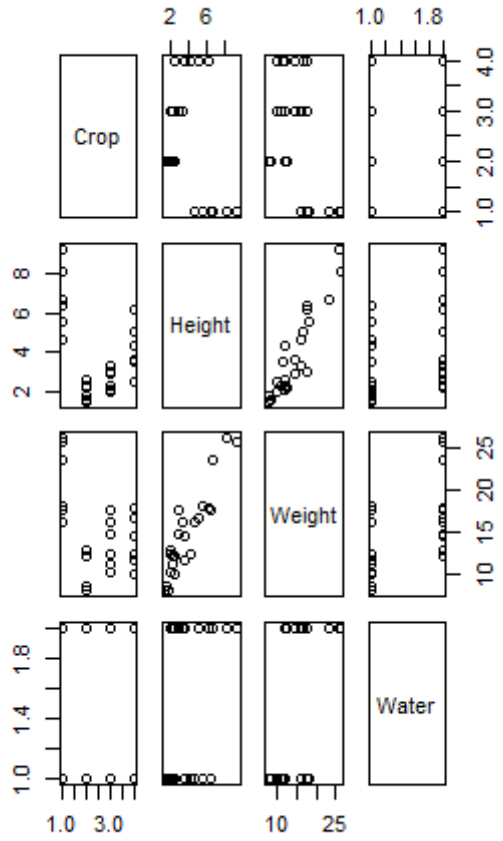
```
# Load libraries
library(readxl)

# Load data
data <- read_excel("D:/R/test1/test1/data.xlsx",
                   col_types = c("text", "numeric", "numeric",
                                 "text"))
# View data
View(data)

# what are the structure of the data set
str(data)
```

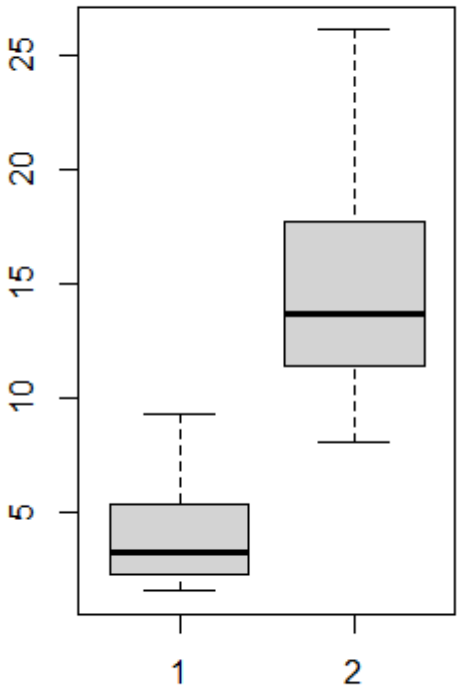
```
head(data) # First 6 rows

tail(data) # Last 6 rows

# Plot our data
plot(data)

# box plot
boxplot(data$Height, data$Weight)

boxplot(data$Crop, data$Weight) # Error because data type for both variable is
different.

# For different type of variable box plot
boxplot(data$Height ~  data$Crop)
```

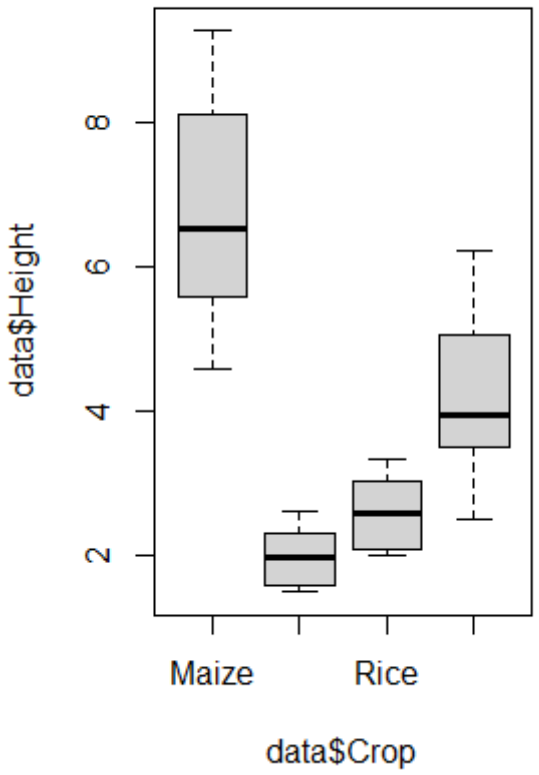
Output

```
> # Load libraries
> library(readxl)
> # Load data
> data <- read_excel("D:/R/test1/test1/data.xlsx",
+                    col_types = c("text", "numeric", "numeric",
+                                  "text"))
> # View data
> View(data)
> # what are the structure of the data set
> str(data)
tibble [24 x 4] (S3: tbl_df/tbl/data.frame)
 $ Crop  : chr [1:24] "Wheat" "Wheat" "Wheat" "Maize" ...
 $ Height: num [1:24] 2.5 3.5 4.3 4.6 6.4 5.6 2.3 2 2.1 1.5 ...
 $ Weight: num [1:24] 10 11.5 12.3 16.2 17.6 ...
 $ Water : chr [1:24] "No" "No" "No" "No" ...
> head(data) # First 6 rows
# A tibble: 6 x 4
  Crop  Height Weight Water
  <chr>  <dbl>  <dbl> <chr>
1 Wheat    2.5   10   No
2 Wheat    3.5   11.5 No
3 Wheat    4.3   12.4 No
4 Maize    4.6   16.2 No
5 Maize    6.4   17.6 No
6 Maize    5.6   18   No
> tail(data) # Last 6 rows
# A tibble: 6 x 4
  Crop  Height Weight Water
  <chr>  <dbl>  <dbl> <chr>
1 Rice    3.34   16.2 yes
```

```
2 Rice    2.9    14.7 yes
3 Rice    3.04   17.7 yes
4 Potao   2.17   12.9 yes
5 Potao   2.32   12.2 yes
6 Potao   2.61   12.0 yes
> # Plot our data
> plot(data)
> # box plot
> boxplot(data$Height, data$Weight)
> boxplot(data$Crop, data$Weight) # Error because data type for both variable is
different.
Error in x[floor(d)] + x[ceiling(d)] :
  non-numeric argument to binary operator
> # For different type of variable box plot
> boxplot(data$Height ~  data$Crop)
```


Crop
Height
Weight
Water


1
2


data$Height
Maize
Rice
data$Crop

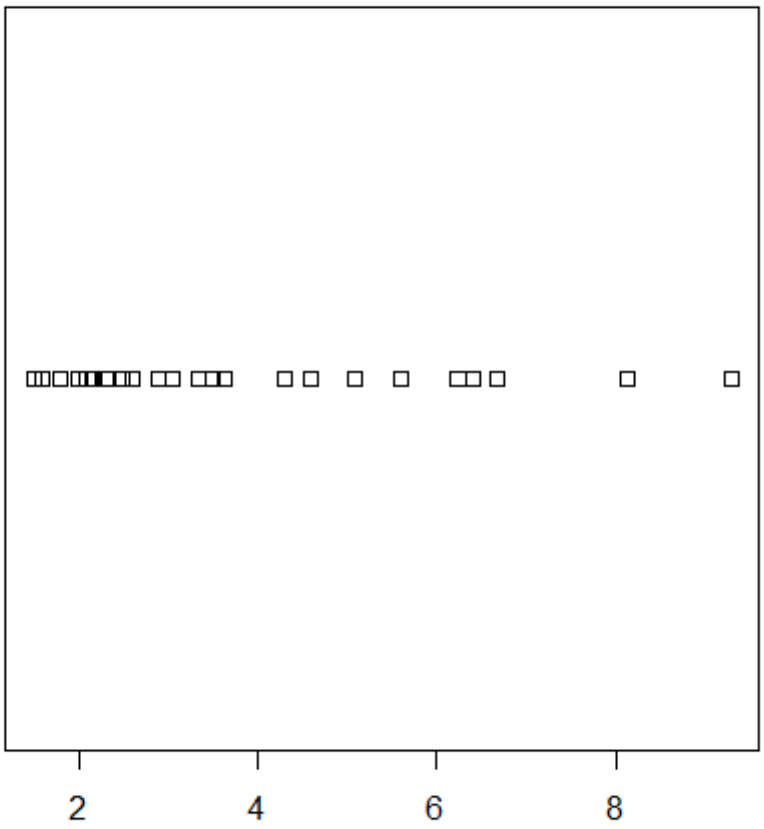
Class 3 Type of Plots in R

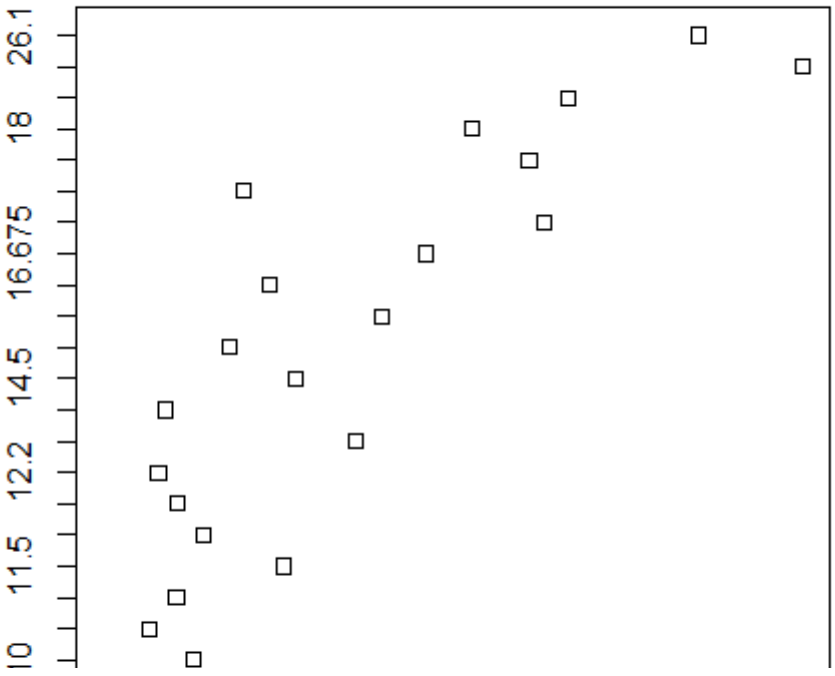
```
# Load libraries
library(readxl)

# Load data
data <- read_excel("D:/R/test1/test1/data.xlsx",
                   col_types = c("text", "numeric", "numeric",
                                 "text"))
# View data
View(data)

# Strip Chart
stripchart(data$Height)

# Between two variable of same type
stripchart(data$Height~data$Weight)

# Histogram
hist(data$Height)
hist(data$Weight) # histogram Also gives you the frequency of the data set

# Plot (Also known as scatter plot)
plot(data$Weight, data$Height)

# QQnorm-plot
qqnorm(data$Height)

# Bar plot
barplot(data$Height)


# Mosaic plot
mosaicplot(data$Crop~data$Height)
```

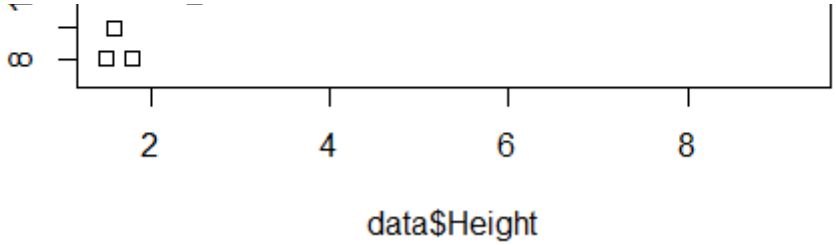
output

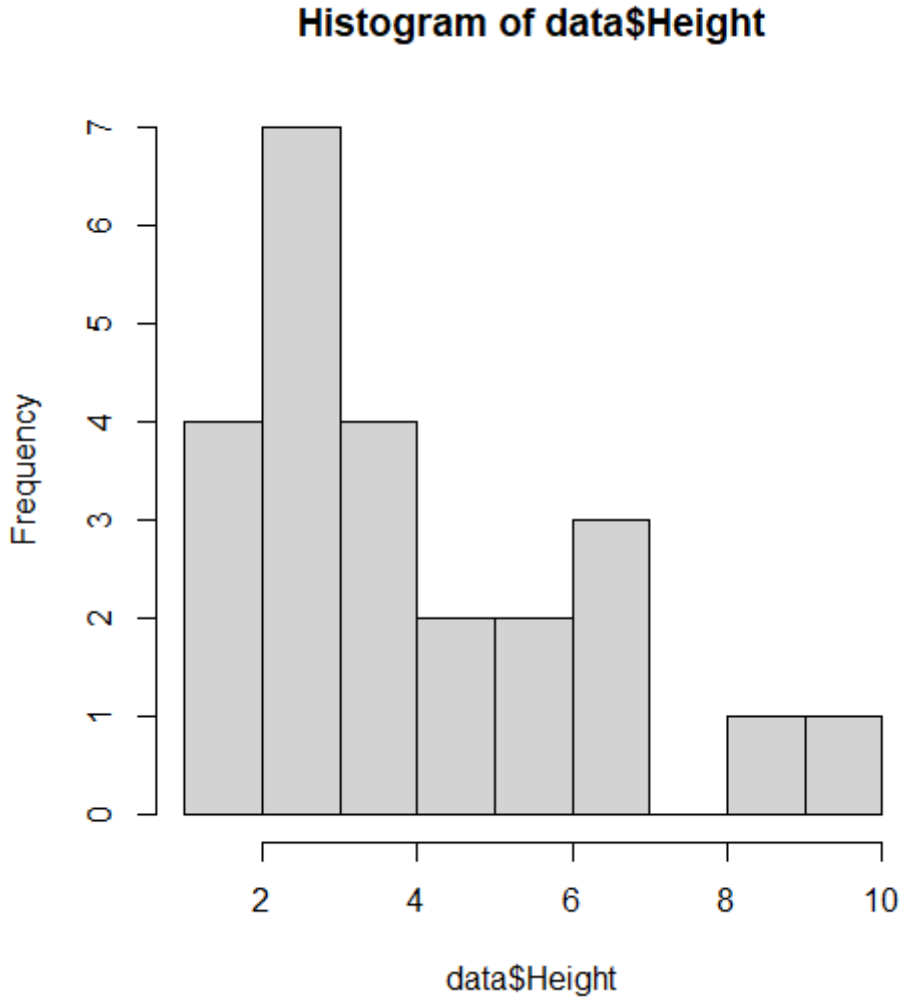
```
> # Load libraries
> library(readxl)
>
> # Load data
> data <- read_excel("D:/R/test1/test1/data.xlsx",
+                    col_types = c("text", "numeric", "numeric",
+                                  "text"))
```

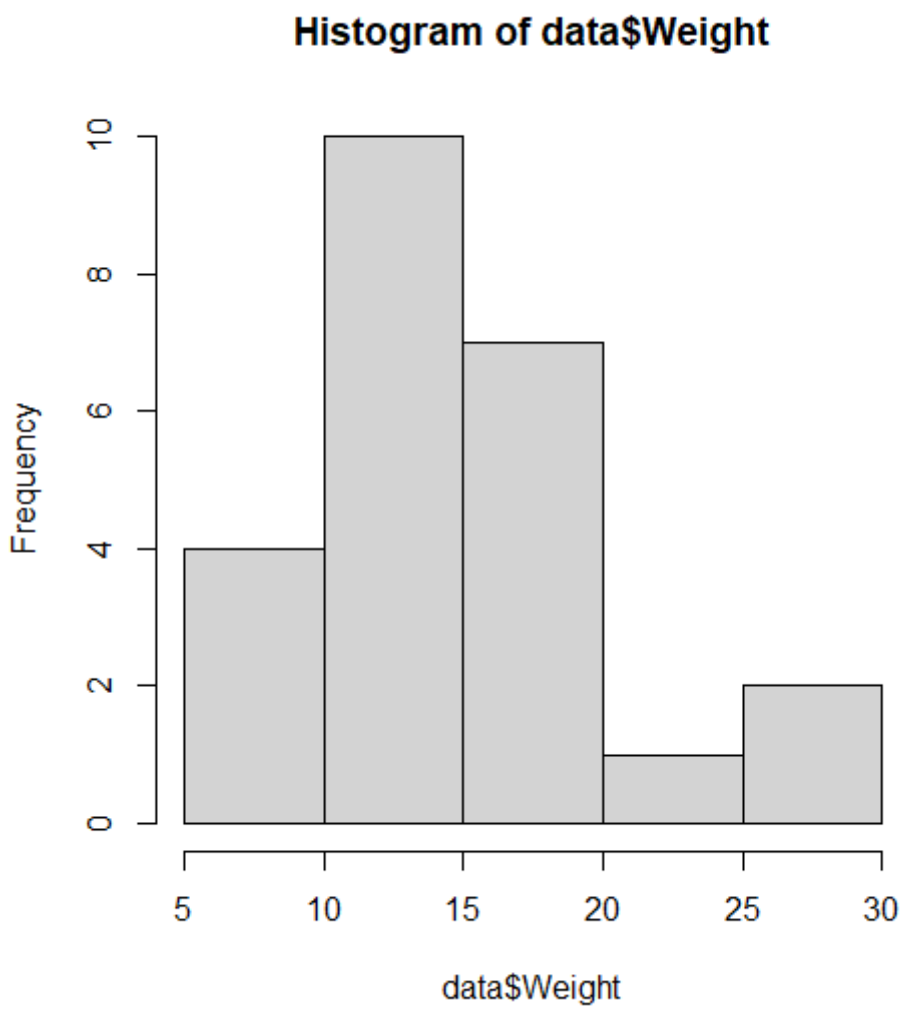
```
> # View data
> View(data)
>
> # Strip Chart
> stripchart(data$Height)
>
> # Between two variable of same type
> stripchart(data$Height~data$Weight)
>
> # Histogram
> hist(data$Height)
> hist(data$Weight) # histogram Also gives you the frequency of the data set
>
> # Plot (Also known as scatter plot)
> plot(data$Weight, data$Height)
>
> # QQnorm-plot
> qqnorm(data$Height)
>
> # Bar plot
> barplot(data$Height)
>
>
> # Mosaic plot
> mosaicplot(data$Crop~data$Height)
```

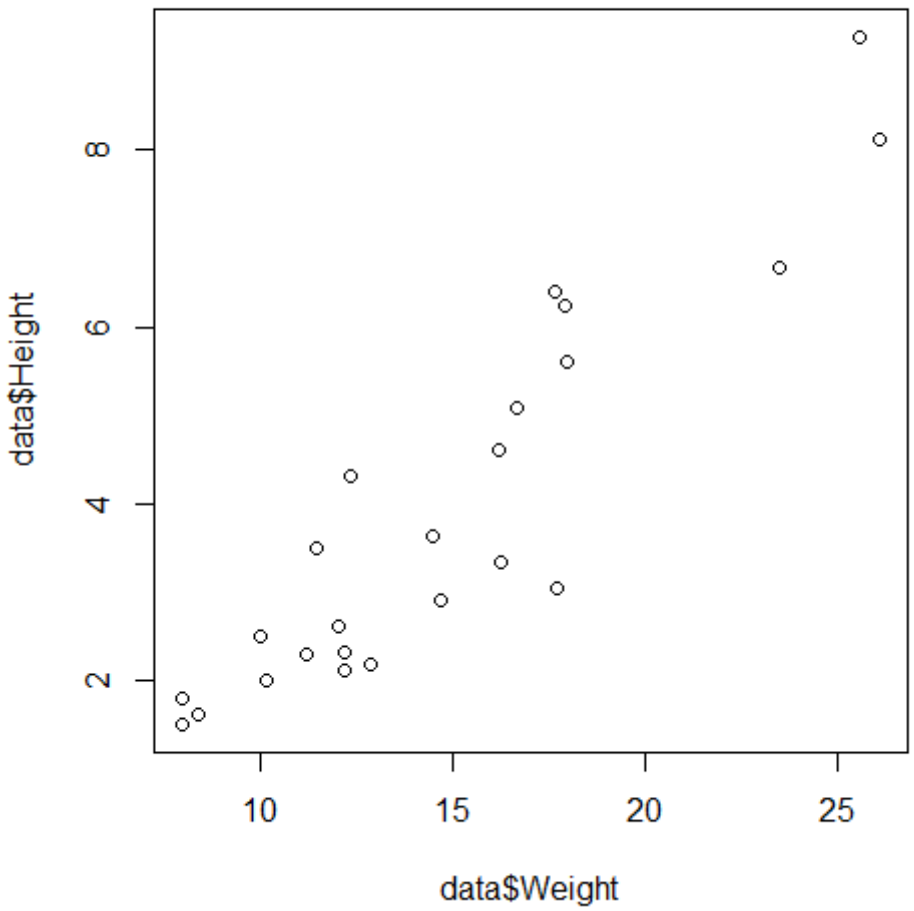

2
4
6
8

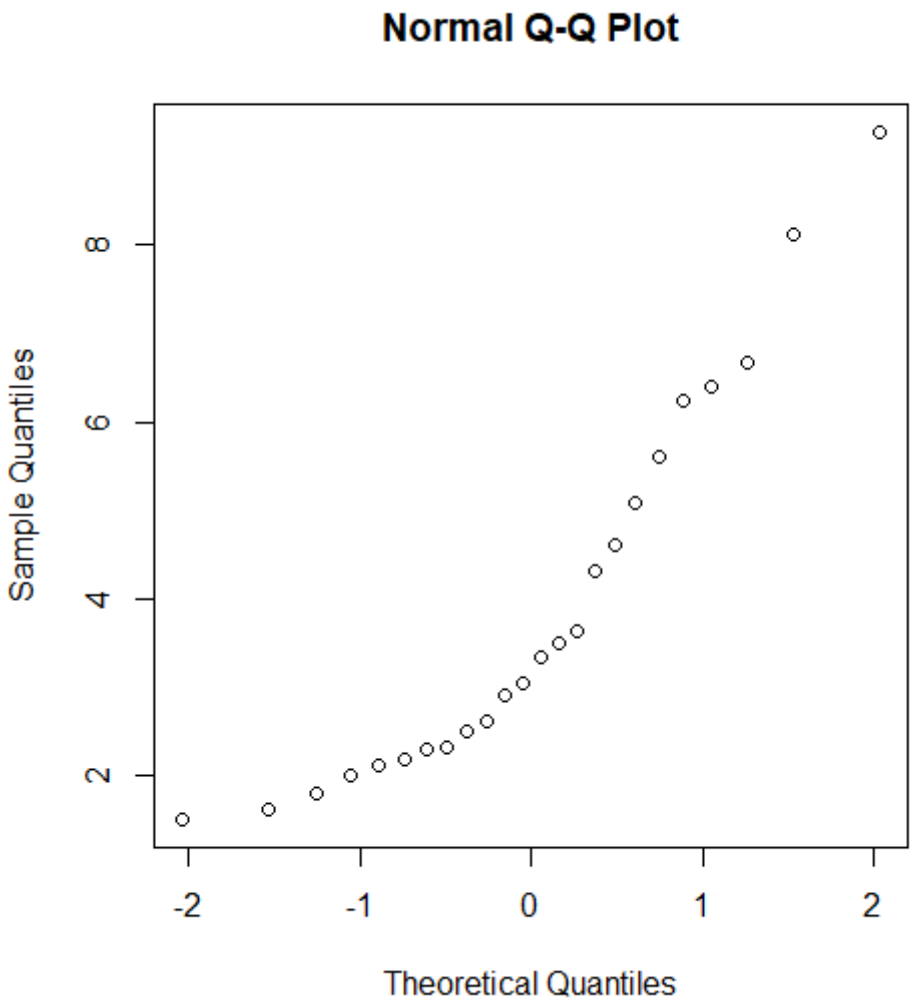

10
11.5
12.2
14.5
16.675
18
26.1

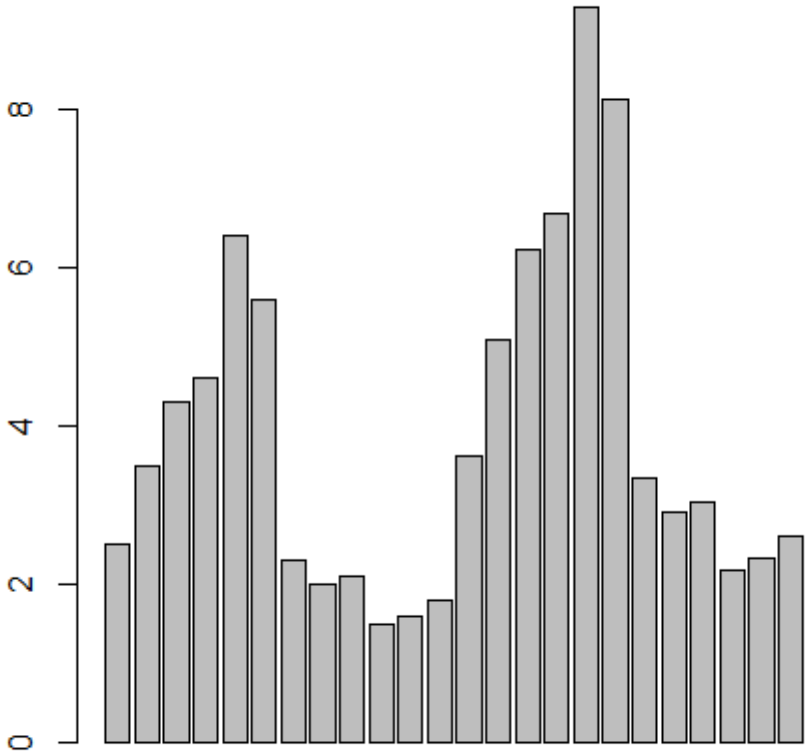

8
2
4
6
8
data$Height

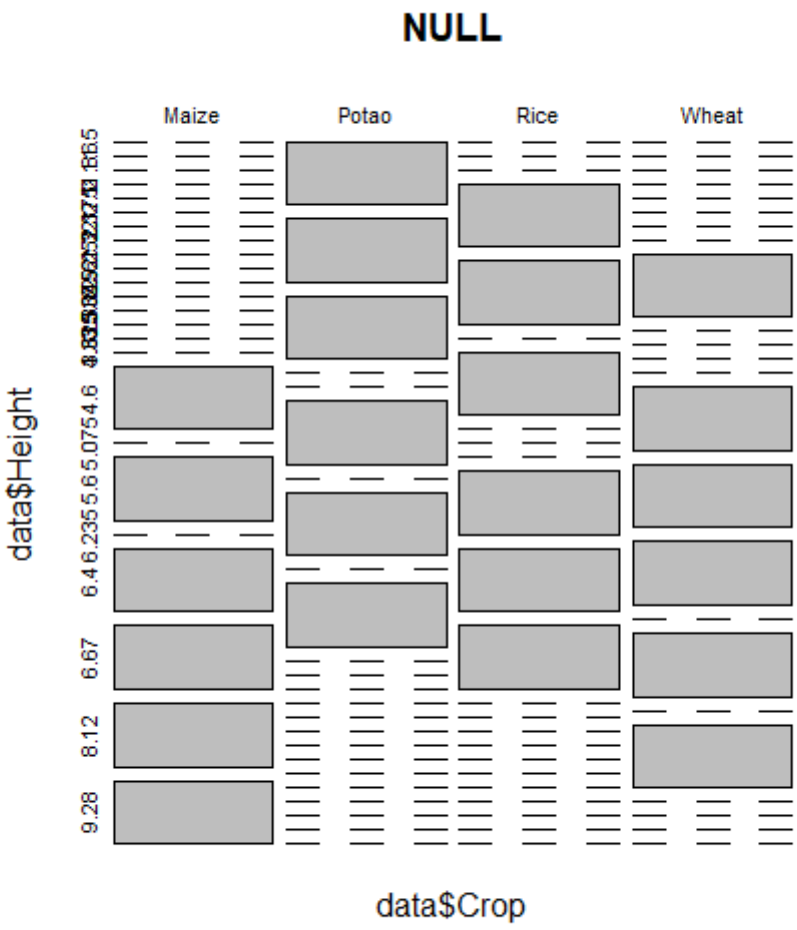

Histogram of data$Height
Frequency
0
1
2
3
4
5
6
7
2
4
6
8
10
data$Height

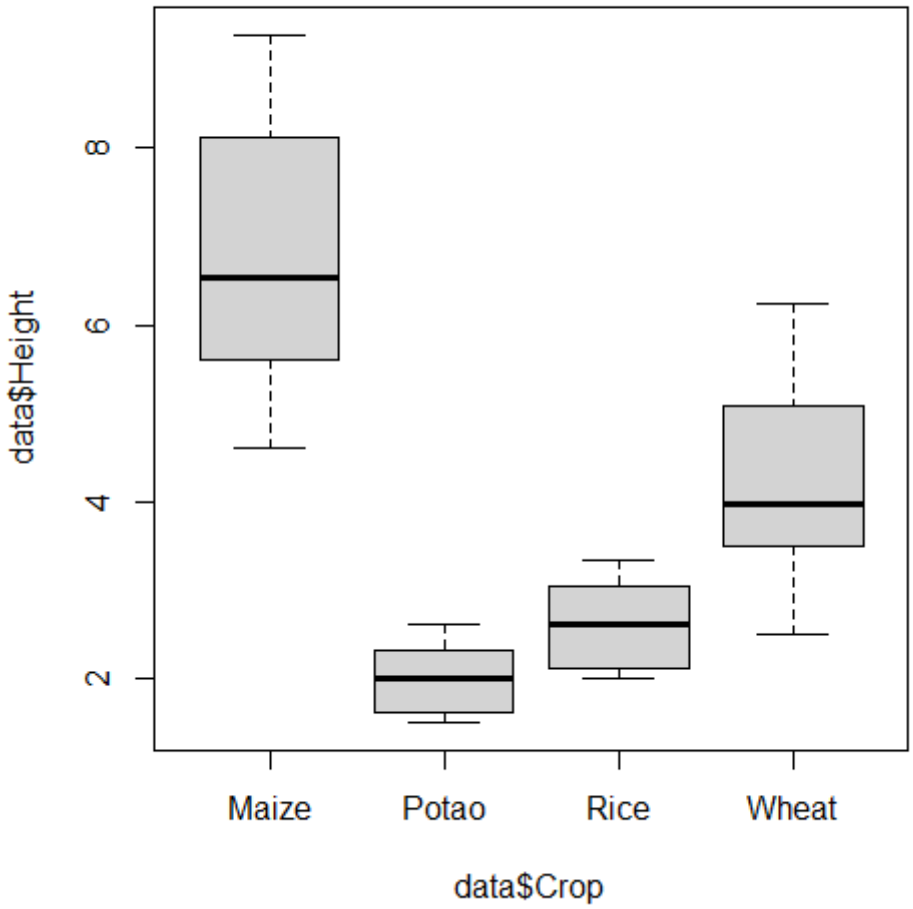

Histogram of data$Weight
Frequency
0
2
4
6
8
10
5
10
15
20
25
30
data$Weight


data$Height
2
4
6
8
10
15
20
25
data$Weight


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
2
4
6
8
-2
-1
0
1
2


0
2
4
6
8


NULL
Maize
Potao
Rice
Wheat
data$Height
9.28
8.12
6.67
6.4
6.2
5.6
5.07
4.6
data$Crop



Class 4 Boxplots (basics) in R

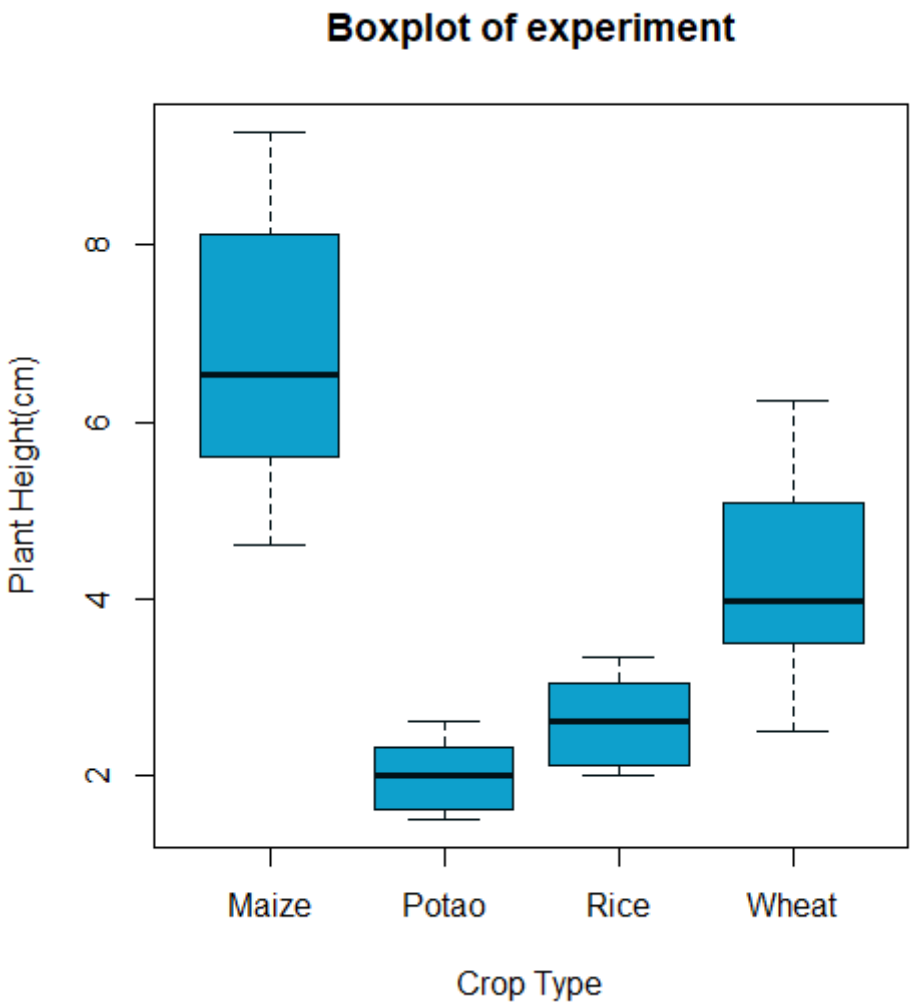
```
# Load libraries
library(readxl)

# Load data
data <- read_excel("D:/R/test1/test1/data.xlsx",
                   col_types = c("text", "numeric", "numeric",
                                 "text"))
# View data
View(data)

# In previous classes we just plot the data without any other format, in this class
we will see how to class, gives title and so on.

# Box Plot
boxplot(data$Height~data$Crop)
# Labeling
```

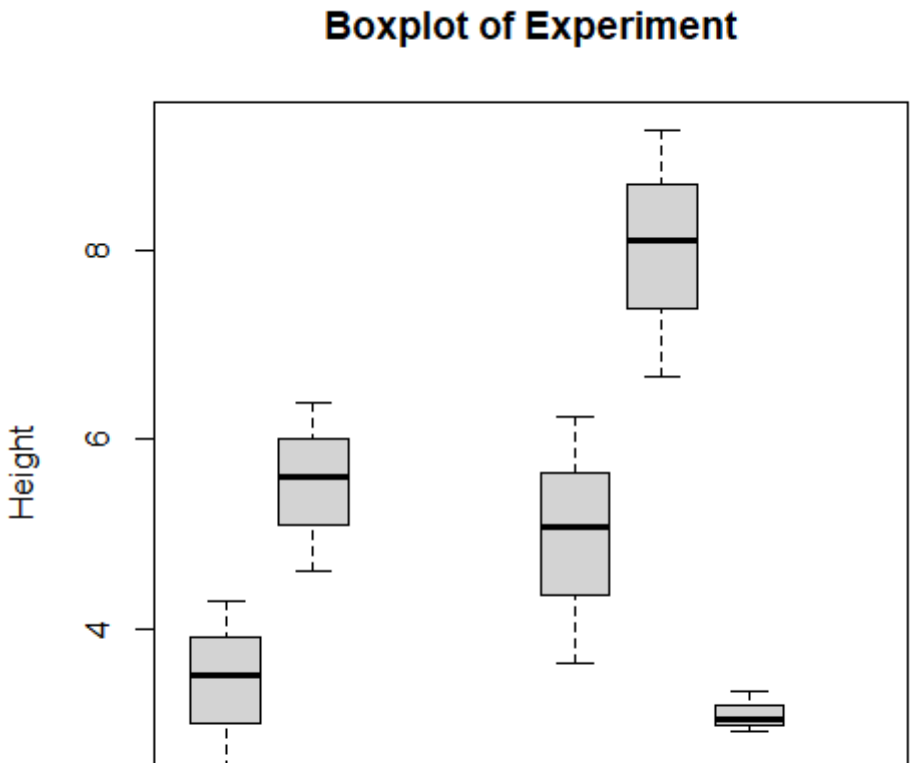
```
boxplot(Height~Crop, data = data)

# Main means title of the plot
boxplot(Height~Crop, data = data, main="Boxplot of experiment")

# Labeling x and y-axis
boxplot(Height~Crop, data = data, main="Boxplot of experiment",
        xlab = "Crop Type", ylab = "Plant Height(cm)")

# Coloring
# Inside Coloring
boxplot(Height~Crop, data = data, main="Boxplot of experiment",
        xlab = "Crop Type", ylab = "Plant Height(cm)",
        col= "gray")

# Broder coloring
boxplot(Height~Crop, data = data, main="Boxplot of experiment",
        xlab = "Crop Type", ylab = "Plant Height(cm)",
        col= "gray", border= "red")

# We can also use the color which we like.
boxplot(Height~Crop, data = data, main="Boxplot of experiment",
        xlab = "Crop Type", ylab = "Plant Height(cm)",
        col= "#0ea0cc", border= "#021217")

# Grouping of treatment

# Box Plot
boxplot(data$Height~data$Crop)

# if you want to add new column in the graph use * OR + sign
boxplot(data$Height~data$Crop*data$Water)

# if we select the name of crops according to your own desire by simple manipulation
data$Crop <- factor(data$Crop, levels = c("Wheat", "Maize", "Rice", "Potato"))

# Again Box plot
boxplot(data$Height~data$Crop*data$Water,
        main = "Boxplot of Experiment",
        xlab = "Crop&Water",
        ylab = "Height")
```

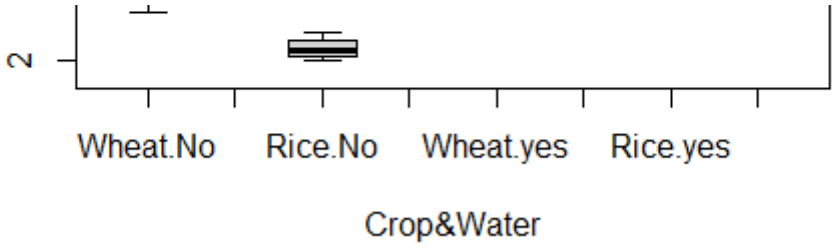
output

```
> # Load libraries
> library(readxl)
> # Load data
```

```
> data <- read_excel("D:/R/test1/test1/data.xlsx",
+                    col_types = c("text", "numeric", "numeric",
+                                  "text"))
> # View data
> View(data)
> # Box Plot
> boxplot(data$Height~data$Crop)
> boxplot(Height~Crop, data = data)
> # Main means title of the plot
> boxplot(Height~Crop, data = data, main="Boxplot of experiment")
> # Labeling x and y-axis
> boxplot(Height~Crop, data = data, main="Boxplot of experiment",
+         xlab = "Crop Type", ylab = "Plant Height(cm)")
> # Coloring
> # Inside Coloring
> boxplot(Height~Crop, data = data, main="Boxplot of experiment",
+         xlab = "Crop Type", ylab = "Plant Height(cm)",
+         col= "gray")
> # Broder coloring
> boxplot(Height~Crop, data = data, main="Boxplot of experiment",
+         xlab = "Crop Type", ylab = "Plant Height(cm)",
+         col= "gray", border= "red")
> # We can also use the color which we like.
> boxplot(Height~Crop, data = data, main="Boxplot of experiment",
+         xlab = "Crop Type", ylab = "Plant Height(cm)",
+         col= "#0ea0cc", border= "#021217")
> # Box Plot
> boxplot(data$Height~data$Crop)
> boxplot(data$Height~data$Crop*data$Water)
> # if we select the name of crops according to your own desire by simple
manipulation
> data$Crop <- factor(data$Crop, levels = c("Wheat", "Maize", "Rice", "Potato"))
> # Again Box plot
> boxplot(data$Height~data$Crop*data$Water,
+         main = "Boxplot of Experiment",
+         xlab = "Crop&Water",
+         ylab = "Height")
```


Boxplot of experiment
Plant Height(cm)
2
4
6
8
Maize
Potao
Rice
Wheat
Crop Type


Boxplot of Experiment
Height
4
6
8



class 5 ANOVA and Tukey Test in R

```
# Load libraries
library(readxl)

# Load data
data <- read_excel("D:/R/test1/test1/data.xlsx",
                   col_types = c("text", "numeric", "numeric",
                                 "text"))
# View data
View(data)

# Descriptive statistics
mean(data$Height)
mean(data$Weight)
median(data$Height)
median(data$Weight)

min(data$Height)
max(data$Height)

# Range
range(data$Height)

# Quatiles
quantile(data$Height, 0.25)
quantile(data$Height, 0.75)

sd(data$Height)
var(data$Height)

# find the descriptive statistics for both variables
lapply(data[, 2:3], mean)
lapply(data[, 2:3], sd)
lapply(data[, 2:3], var)

# find the descriptive statistics
summary(data)
```

```
# ANOVA (analysis of variance )
# Check the difference between the height by crop wise
aov(data$Height~data$Crop)
a1 <- aov(data$Height~data$Crop)
# to check the significance of anova or not
summary(a1)

# Which one id differ
TukeyHSD(a1)

# Group avova
a2<- aov(data$Height~data$Crop*data$Water)
summary(a2)
TukeyHSD(a2)
# Lettering automatic then use package(Agricolae)
```

output

```
> # Load libraries
> library(readxl)
>
> # Load data
> data <- read_excel("D:/R/test1/test1/data.xlsx",
+                    col_types = c("text", "numeric", "numeric",
+                                  "text"))
> # View data
> View(data)
>
> # Descriptive statistics
> mean(data$Height)
[1] 3.899583
> mean(data$Weight)
[1] 14.73915
> median(data$Height)
[1] 3.19
> median(data$Weight)
[1] 13.7025
>
> min(data$Height)
[1] 1.5
> max(data$Height)
[1] 9.28
>
> # Range
> range(data$Height)
[1] 1.50 9.28
>
> # Quatiles
```

```
> quantile(data$Height, 0.25)
    25%
2.26875
> quantile(data$Height, 0.75)
    75%
5.20625
>
> sd(data$Height)
[1] 2.159917
> var(data$Height)
[1] 4.665241
>
> # find the descriptive statistics for both variables
> lapply(data[, 2:3], mean)
$Height
[1] 3.899583

$Weight
[1] 14.73915

> lapply(data[, 2:3], sd)
$Height
[1] 2.159917

$Weight
[1] 5.091549

> lapply(data[, 2:3], var)
$Height
[1] 4.665241

$Weight
[1] 25.92388

>
> # find the descriptive statistics
> summary(data)
     Crop               Height          Weight         Water
 Length:24          Min.   :1.500   Min.   : 8.00   Length:24
 Class :character   1st Qu.:2.269   1st Qu.:11.43   Class :character
 Mode  :character   Median :3.190   Median :13.70   Mode  :character
                    Mean   :3.900   Mean   :14.74
                    3rd Qu.:5.206   3rd Qu.:17.66
                    Max.   :9.280   Max.   :26.10
>
> # ANOVA (analysis of variance )
> # Check the difference between the height by crop wise
> aov(data$Height~data$Crop)
Call:
   aov(formula = data$Height ~ data$Crop)
```

```
Terms:
                data$Crop Residuals
Sum of Squares   81.84409  25.45646
Deg. of Freedom         3        20

Residual standard error: 1.128195
Estimated effects may be unbalanced
> a1 <- aov(data$Height~data$Crop)
> # to check the significance of anova or not
> summary(a1)
            Df Sum Sq Mean Sq F value   Pr(>F)
data$Crop    3  81.84  27.281   21.43 1.85e-06 ***
Residuals   20  25.46   1.273
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
> # Which one id differ
> TukeyHSD(a1)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = data$Height ~ data$Crop)

$`data$Crop`
               diff        lwr        upr     p adj
Potao-Maize -4.7775 -6.6006246 -2.9543754 0.0000024
Rice-Maize  -4.1650 -5.9881246 -2.3418754 0.0000171
Wheat-Maize -2.5725 -4.3956246 -0.7493754 0.0040555
Rice-Potao   0.6125 -1.2106246  2.4356246 0.7837703
Wheat-Potao  2.2050  0.3818754  4.0281246 0.0143462
Wheat-Rice   1.5925 -0.2306246  3.4156246 0.1005338

>
> # Group avova
> a2<- aov(data$Height~data$Crop*data$Water)
> summary(a2)
                     Df Sum Sq Mean Sq F value   Pr(>F)
data$Crop             3  81.84  27.281  42.040 8.17e-08 ***
data$Water            1  12.31  12.312  18.973  0.00049 ***
data$Crop:data$Water  3   2.76   0.920   1.418  0.27410
Residuals            16  10.38   0.649
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> TukeyHSD(a2)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = data$Height ~ data$Crop * data$Water)
```

```
$`data$Crop`
               diff        lwr       upr     p adj
Potao-Maize -4.7775 -6.1081455 -3.446854 0.0000001
Rice-Maize  -4.1650 -5.4956455 -2.834354 0.0000007
Wheat-Maize -2.5725 -3.9031455 -1.241854 0.0002413
Rice-Potao   0.6125 -0.7181455  1.943146 0.5659505
Wheat-Potao  2.2050  0.8743545  3.535646 0.0011425
Wheat-Rice   1.5925  0.2618545  2.923146 0.0164815

$`data$Water`
         diff       lwr      upr     p adj
yes-No 1.4325 0.7353232 2.129677 0.0004903

$`data$Crop:data$Water`
                       diff        lwr        upr     p adj
Potao:No-Maize:No   -3.900 -6.1772068 -1.6227932 0.0004404
Rice:No-Maize:No    -3.400 -5.6772068 -1.1227932 0.0018460
Wheat:No-Maize:No   -2.100 -4.3772068  0.1772068 0.0821314
Maize:yes-Maize:No   2.490  0.2127932  4.7672068 0.0270516
Potao:yes-Maize:No  -3.165 -5.4422068 -0.8877932 0.0036796
Rice:yes-Maize:No   -2.440 -4.7172068 -0.1627932 0.0312939
Wheat:yes-Maize:No  -0.555 -2.8322068  1.7222068 0.9872978
Rice:No-Potao:No     0.500 -1.7772068  2.7772068 0.9930677
Wheat:No-Potao:No    1.800 -0.4772068  4.0772068 0.1807540
Maize:yes-Potao:No   6.390  4.1127932  8.6672068 0.0000009
Potao:yes-Potao:No   0.735 -1.5422068  3.0122068 0.9434362
Rice:yes-Potao:No    1.460 -0.8172068  3.7372068 0.3900775
Wheat:yes-Potao:No   3.345  1.0677932  5.6222068 0.0021678
Wheat:No-Rice:No     1.300 -0.9772068  3.5772068 0.5243761
Maize:yes-Rice:No    5.890  3.6127932  8.1672068 0.0000028
Potao:yes-Rice:No    0.235 -2.0422068  2.5122068 0.9999454
Rice:yes-Rice:No     0.960 -1.3172068  3.2372068 0.8169814
Wheat:yes-Rice:No    2.845  0.5677932  5.1222068 0.0094931
Maize:yes-Wheat:No   4.590  2.3127932  6.8672068 0.0000675
Potao:yes-Wheat:No  -1.065 -3.3422068  1.2122068 0.7335043
Rice:yes-Wheat:No   -0.340 -2.6172068  1.9372068 0.9993702
Wheat:yes-Wheat:No   1.545 -0.7322068  3.8222068 0.3270648
Potao:yes-Maize:yes -5.655 -7.9322068 -3.3777932 0.0000048
Rice:yes-Maize:yes  -4.930 -7.2072068 -2.6527932 0.0000281
Wheat:yes-Maize:yes -3.045 -5.3222068 -0.7677932 0.0052465
Rice:yes-Potao:yes   0.725 -1.5522068  3.0022068 0.9471214
Wheat:yes-Potao:yes  2.610  0.3327932  4.8872068 0.0190246
Wheat:yes-Rice:yes   1.885 -0.3922068  4.1622068 0.1457497

> # Lettering automatic then use package(Agricolae)
```

class 6 ANOVA and multiple means comparison

```r
# Load libraries
library(readxl)

# Load data
x <- read_excel("D:/R/test1/test1/data.xlsx",
                  col_types = c("text", "numeric", "numeric",
                                "text"))
# View data
View(data)

# Box plot
boxplot(x$Height~x$Crop)
boxplot(Height~Crop, data = x)

# multiple comparison of means

# 1- Tu-key-HSD test (firstly we install packages)
install.packages("agricolae")
library(agricolae)

# If you want to check the history of this packages use "help"command to see the
whole document regarding this pacakge.
help("agricolae-package")
?`agricolae-package`

# Make model
model <- aov(Height~Crop, data = x)

# Applying test
out <- HSD.test(model, "Crop", group = TRUE, console = TRUE, main = "Tukey test")

# plot
plot(out)

#2- LSD Test

# Make model
model <- aov(Height~Crop, data = x)

# Applying test
out <- LSD.test(model, "Crop", group = TRUE, console = TRUE, main = "LSD test")

# plot
plot(out)

#3- Duncan test

# Make model
model <- aov(Height~Crop, data = x)
```

```
# Applying test
out <- duncan.test(model, "Crop", group = TRUE, console = TRUE, main = "Duncan
test")

# plot
plot(out)


# Grouping
boxplot(Height~Crop*Water, data = x, las=2, xlab = "")
model1 <- aov(Height~Crop*Water, data = x)

# Applying test
out1 <- HSD.test(model1, c("Crop", "Water"), group = TRUE, console = TRUE, main =
"Tukey test")

# plot
plot(out1, horiz = TRUE, las=2)
plot(out1, las=2)


# String is a character or categorical variable.

# save in super quality graph.
jpeg(filename = "Tukey test.tiff",
     width = 6, height = 4, units = "in", res = 300)
boxplot(Height~Crop*Water, data = x, las=2, xlab = "")
dev.off()

# letter graph saving
jpeg(filename = "Tukey testplot.tiff",
     width = 6, height = 4, units = "in", res = 300)
plot(out1, horiz = TRUE, las=2)
dev.off()
```

output

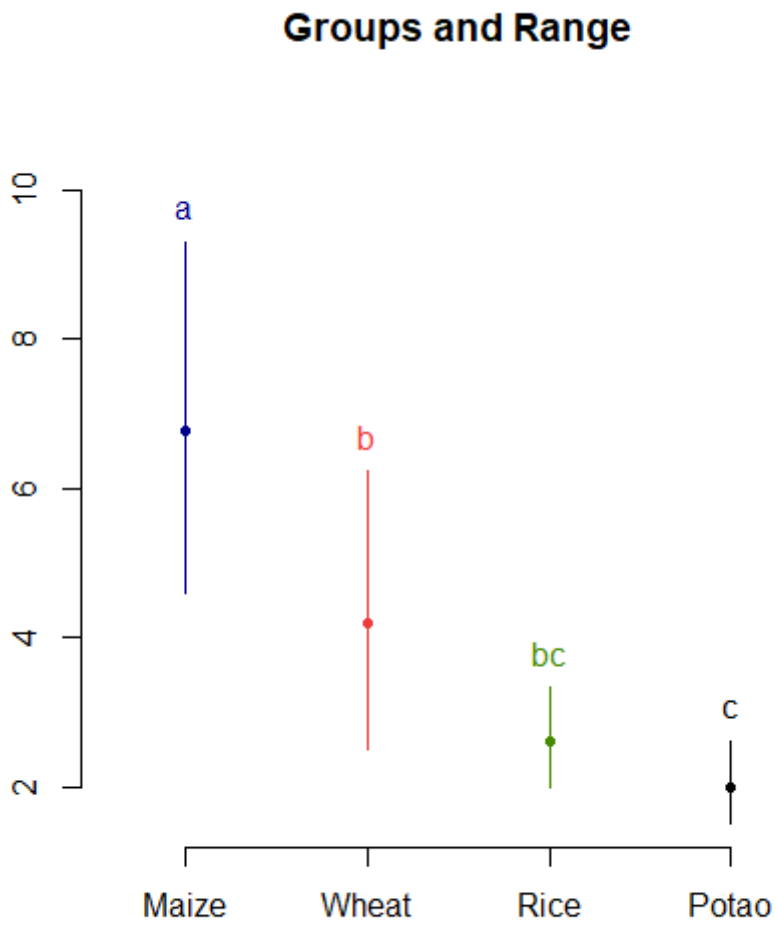
```
> # Load libraries
> library(readxl)
>
> # Load data
> x <- read_excel("D:/R/test1/test1/data.xlsx",
+                   col_types = c("text", "numeric", "numeric",
+                                 "text"))
> # View data
> View(data)
>
```

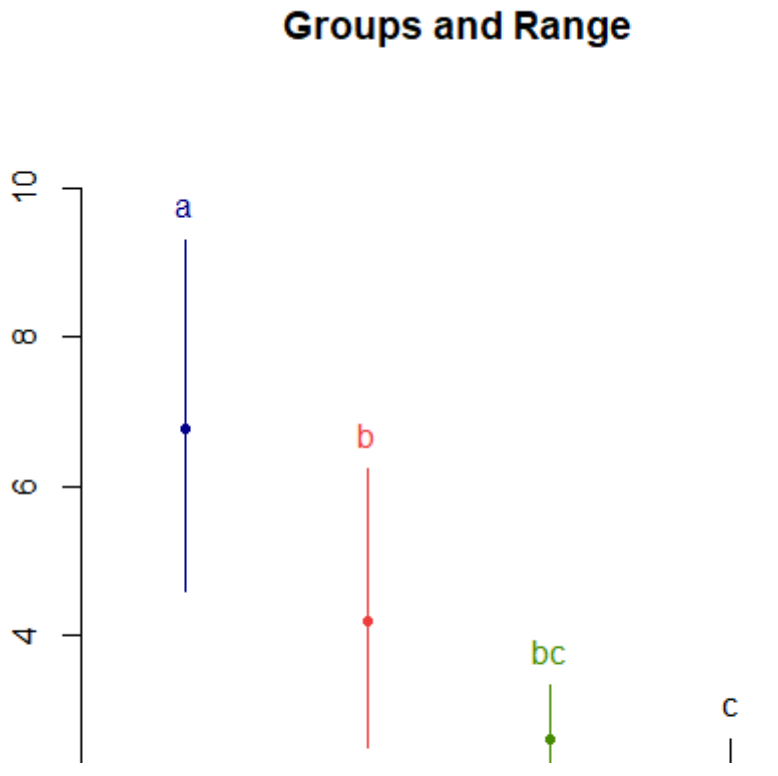
```
> # Box plot
> boxplot(x$Height~x$Crop)
> boxplot(Height~Crop, data = x)
> # Make model
> model <- aov(Height~Crop, data = x)
> # Applying test
> out <- HSD.test(model, "Crop", group = TRUE, console = TRUE, main = "Tukey test")

Study: Tukey test

HSD Test for Height

Mean Square Error:  1.272823

Crop,  means

        Height       std r Min   Max
Maize 6.778333 1.6939235 6 4.6 9.280
Potao 2.000833 0.4370631 6 1.5 2.610
Rice  2.613333 0.5526632 6 2.0 3.335
Wheat 4.205833 1.3135654 6 2.5 6.235

Alpha: 0.05 ; DF Error: 20
Critical Value of Studentized Range: 3.958293

Minimun Significant Difference: 1.823125

Treatments with the same letter are not significantly different.

        Height groups
Maize 6.778333      a
Wheat 4.205833      b
Rice  2.613333     bc
Potao 2.000833      c
> # plot
> plot(out)
> # Make model
> model <- aov(Height~Crop, data = x)
> # Applying test
> out <- LSD.test(model, "Crop", group = TRUE, console = TRUE, main = "LSD test")

Study: LSD test

LSD t Test for Height

Mean Square Error:  1.272823

Crop,  means and individual ( 95 %) CI

        Height       std r      LCL      UCL Min   Max
```

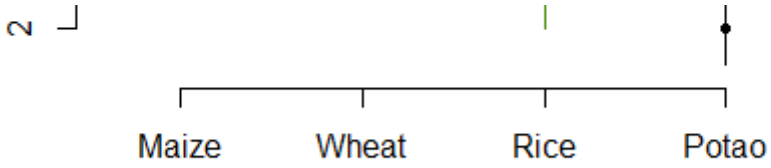
```
Maize 6.778333 1.6939235 6 5.817573 7.739094 4.6 9.280
Potao 2.000833 0.4370631 6 1.040073 2.961594 1.5 2.610
Rice  2.613333 0.5526632 6 1.652573 3.574094 2.0 3.335
Wheat 4.205833 1.3135654 6 3.245073 5.166594 2.5 6.235

Alpha: 0.05 ; DF Error: 20
Critical Value of t: 2.085963

least Significant Difference: 1.35872

Treatments with the same letter are not significantly different.

        Height groups
Maize 6.778333      a
Wheat 4.205833      b
Rice  2.613333      c
Potao 2.000833      c
> # plot
> plot(out)
> # Make model
> model <- aov(Height~Crop, data = x)
> # Applying test
> out <- duncan.test(model, "Crop", group = TRUE, console = TRUE, main = "Duncan
test")

Study: Duncan test

Duncan's new multiple range test
for Height

Mean Square Error:  1.272823

Crop,  means

        Height       std r Min   Max
Maize 6.778333 1.6939235 6 4.6 9.280
Potao 2.000833 0.4370631 6 1.5 2.610
Rice  2.613333 0.5526632 6 2.0 3.335
Wheat 4.205833 1.3135654 6 2.5 6.235

Alpha: 0.05 ; DF Error: 20

Critical Range
       2        3        4
1.358720 1.426200 1.469084

Means with the same letter are not significantly different.

        Height groups
Maize 6.778333      a
```

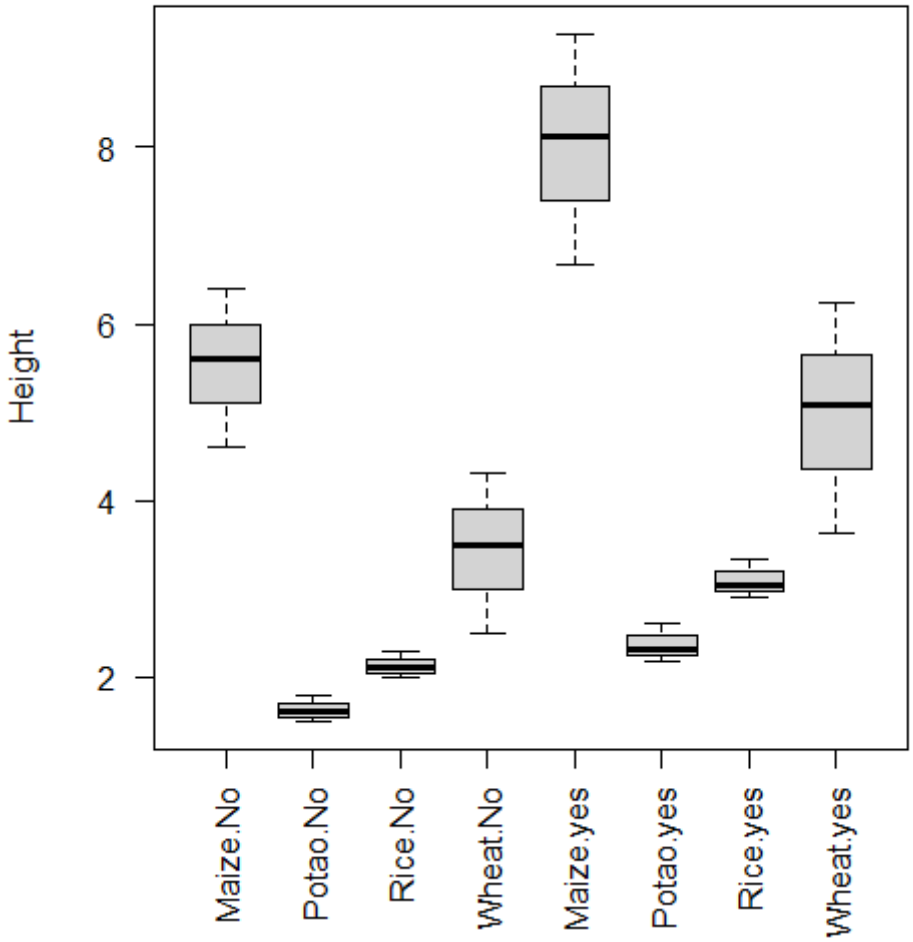
```
Wheat 4.205833      b
Rice  2.613333      c
Potao 2.000833      c
> # plot
> plot(out)
> boxplot(Height~Crop*Water, data = x, las=2, xlab = "")
> model1 <- aov(Height~Crop*Water, data = x)
> # Applying test
> out1 <- HSD.test(model1, c("Crop", "Water"), group = TRUE, console = TRUE, main =
"Tukey test")

Study: Tukey test

HSD Test for Height

Mean Square Error:  0.6489396

Crop:Water,  means

            Height       std r   Min   Max
Maize:No  5.533333 0.9018500 3 4.600 6.400
Maize:yes 8.023333 1.3076824 3 6.670 9.280
Potao:No  1.633333 0.1527525 3 1.500 1.800
Potao:yes 2.368333 0.2214912 3 2.175 2.610
Rice:No   2.133333 0.1527525 3 2.000 2.300
Rice:yes  3.093333 0.2214912 3 2.900 3.335
Wheat:No  3.433333 0.9018500 3 2.500 4.300
Wheat:yes 4.978333 1.3076824 3 3.625 6.235

Alpha: 0.05 ; DF Error: 16
Critical Value of Studentized Range: 4.89622

Minimun Significant Difference: 2.277207

Treatments with the same letter are not significantly different.

            Height groups
Maize:yes 8.023333      a
Maize:No  5.533333      b
Wheat:yes 4.978333     bc
Wheat:No  3.433333    bcd
Rice:yes  3.093333     cd
Potao:yes 2.368333      d
Rice:No   2.133333      d
Potao:No  1.633333      d
> # plot
> plot(out1, horiz = TRUE, las=2)
> plot(out1, las=2)
> jpeg(filename = "Tukey test.tiff",
+      width = 6, height = 4, units = "in", res = 300)
```

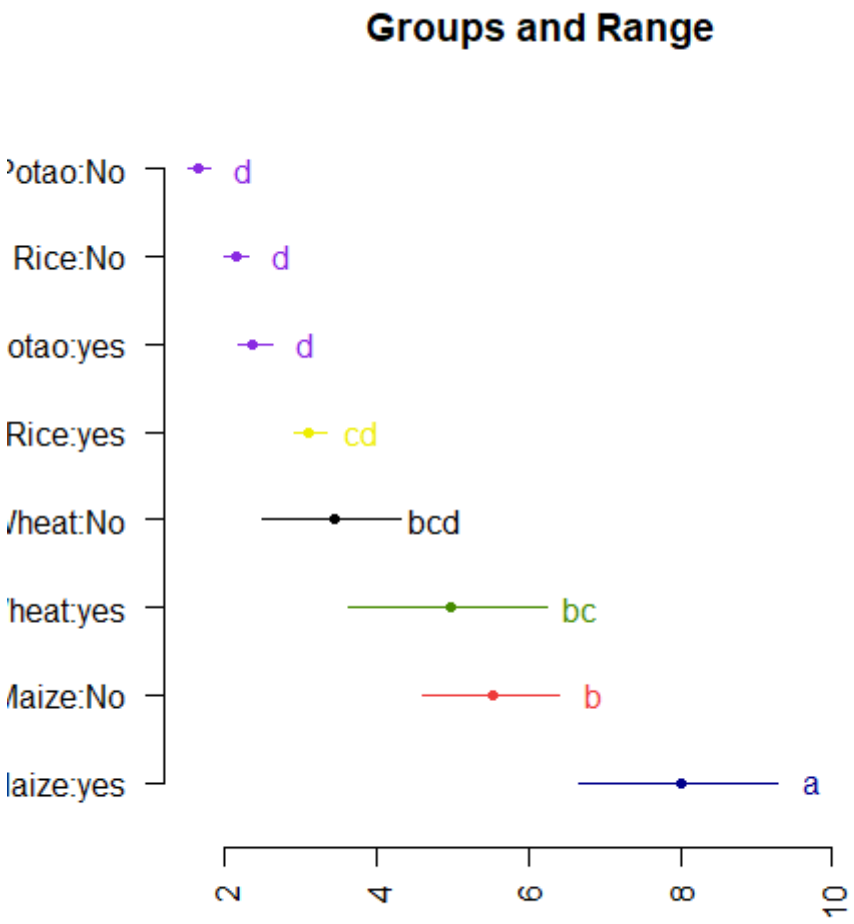
```
> boxplot(Height~Crop*Water, data = x, las=2, xlab = "")
> dev.off()
RStudioGD
        2
> jpeg(filename = "Tukey testplot.tiff",
+      width = 6, height = 4, units = "in", res = 300)
> plot(out1, horiz = TRUE, las=2)
> dev.off()
RStudioGD
        2
```

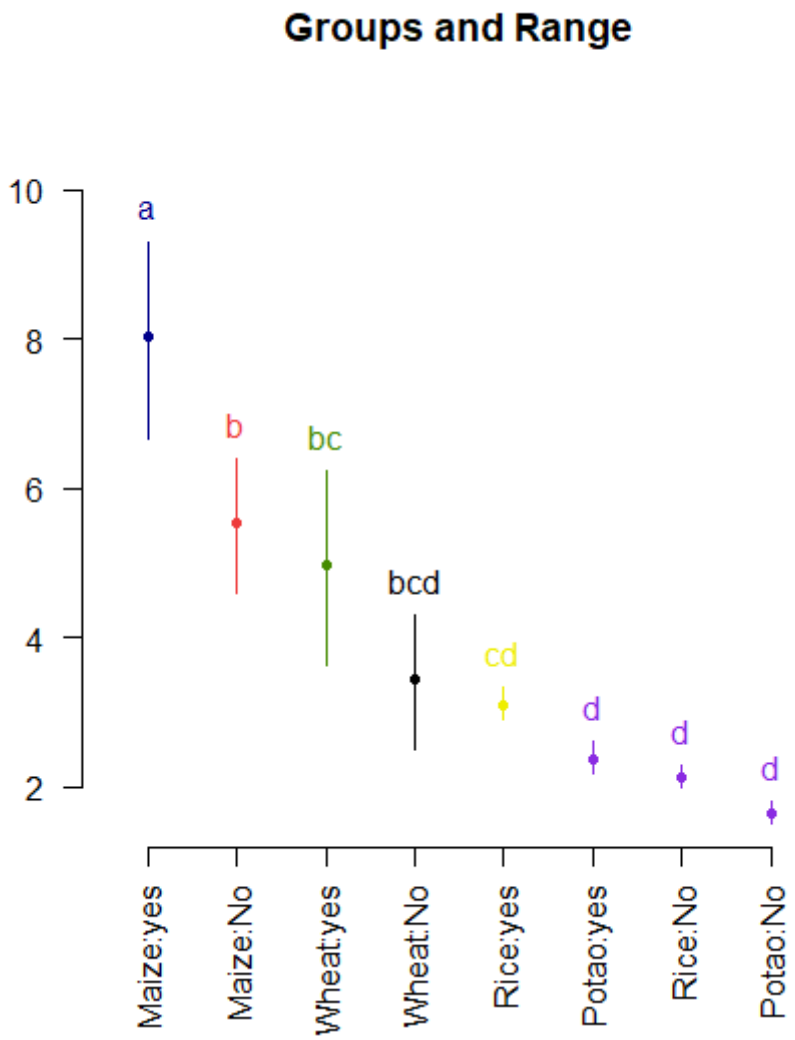

Groups and Range
10
8
6
4
2
a
b
bc
c
Maize
Wheat
Rice
Potao


Groups and Range
10
8
6
4
a
b
bc
c


2
Maize
Wheat
Rice
Potao


Height
2
4
6
8
Maize.No
Potao.No
Rice.No
Wheat.No
Maize.yes
Potao.yes
Rice.yes
Wheat.yes


Groups and Range
otao:No
Rice:No
otao:yes
Rice:yes
heat:No
heat:yes
laize:No
laize:yes
d
d
d
cd
bcd
bc
b
a
2
4
6
8
10



class 7 Data Visulization with ggplot2 package in R

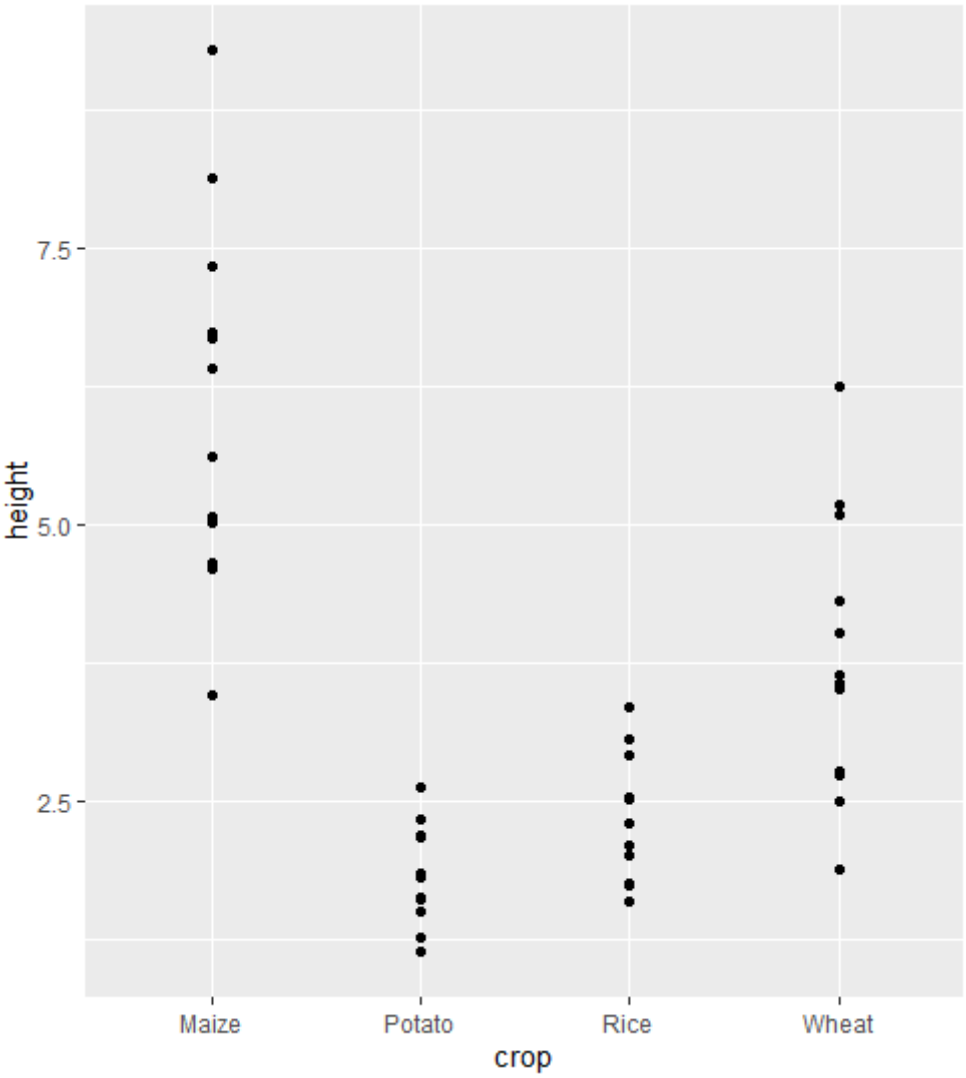
```
# Data visualization
install.packages("ggplot2")
library(ggplot2)

# ggplot (function)
# Data
# Mapping(x-axis, y-axis)
# Geometry(name of graph, like scatter, box, line etc)

library(readxl)
x <- read_excel("D:/R/test1/test1/ggdata.xlsx",
                     col_types = c("text", "numeric", "text",
                                   "text"))
View(ggdata)

# ggplot
```

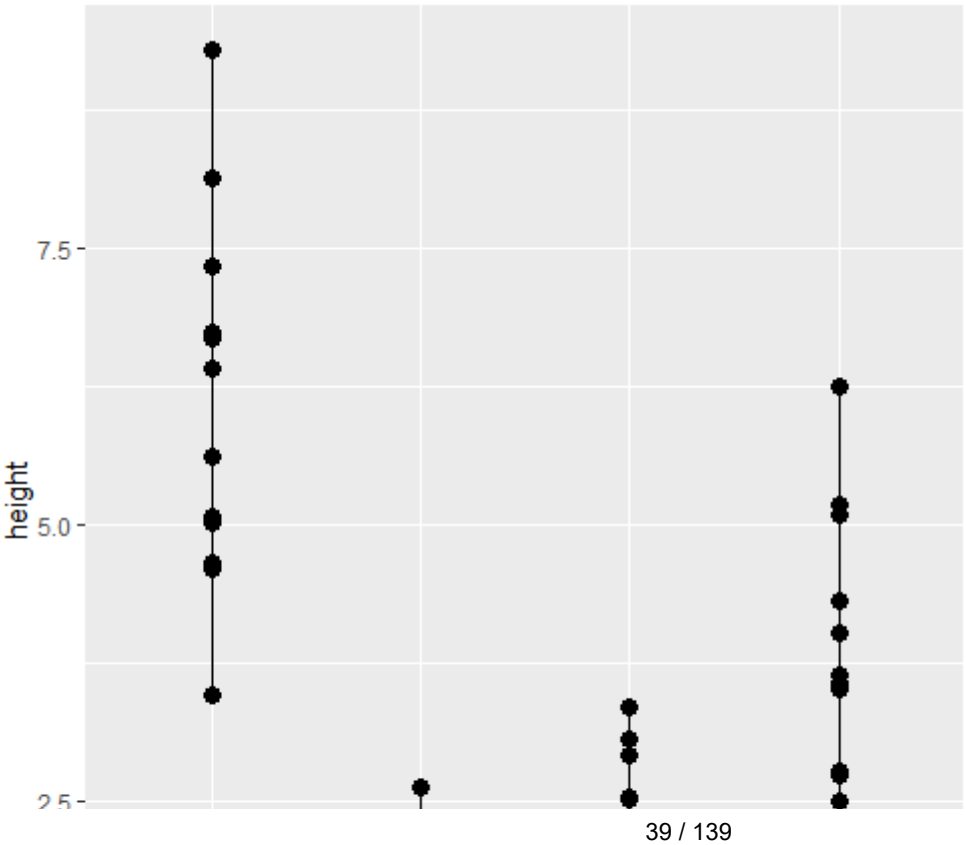
```
ggplot(data = x, mapping = aes(x=crop, y=height))+ geom_point()

# in ggplot it knowns what is meant by data, aes etc.
ggplot(x, aes(crop, height))+
  geom_point(size=3)+
  geom_line()

# box plot
ggplot(x, aes(crop, height))+
  geom_boxplot()+
  geom_point(size=3, colour="red", alpha= 0.8)

# other variable
ggplot(x, aes(crop, height, color= water))+
  geom_boxplot()

# inside filling
ggplot(x, aes(crop, height, fill= water))+
  geom_boxplot()

# Divide according to fert.type
ggplot(x, aes(crop, height, fill= water))+
  geom_boxplot()+
  facet_wrap(~fert.type)+
  labs(x="Crop Type", y="Plant height (cm)",
      title = "Plant Growth")+
  theme_bw()+
  ggsave("ggplot.tiff", units = "in", width = 8, height = 6, dpi = 300, compression
= "lzw")

# variation
ggplot(x, aes(crop, height, fill= water))+
  geom_boxplot()+
  facet_wrap(~fert.type)+
  labs(x="Crop Type", y="Plant height (cm)",
       title = "Plant Growth")+
  theme_bw()+
  coord_flip()+
  ggsave("ggplot.tiff", units = "in", width = 8, height = 6, dpi = 300, compression
= "lzw")
```

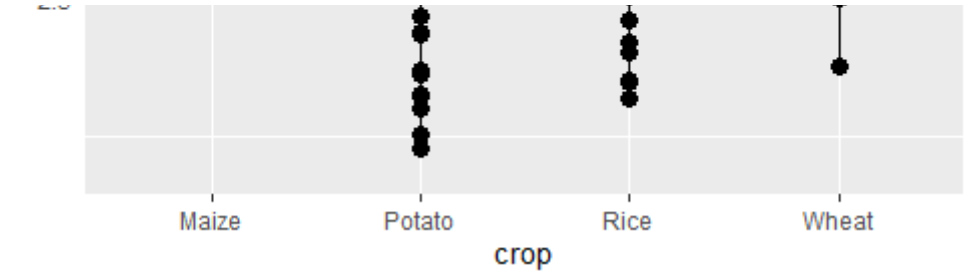
output

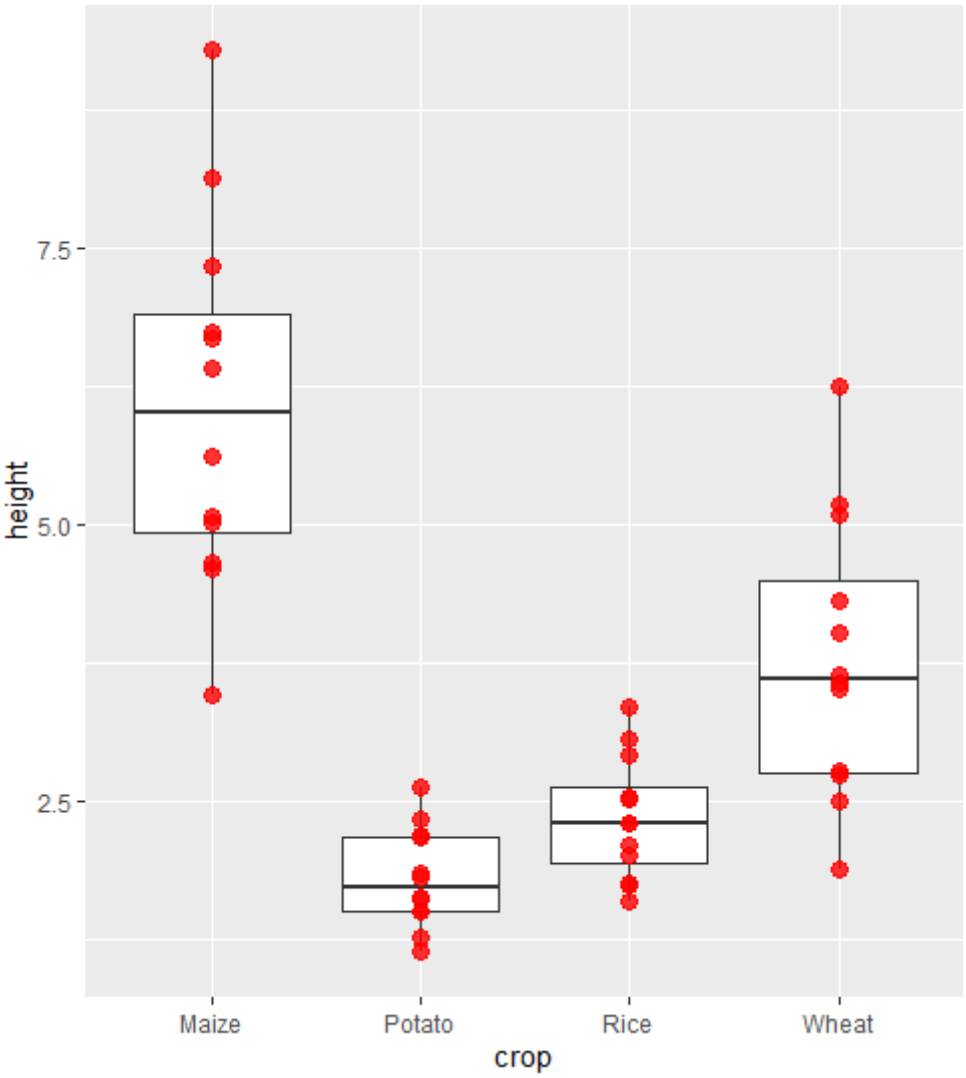
```
> library(ggplot2)
> library(readxl)
> x <- read_excel("D:/R/test1/test1/ggdata.xlsx",
+                      col_types = c("text", "numeric", "text",
+                                    "text"))
```

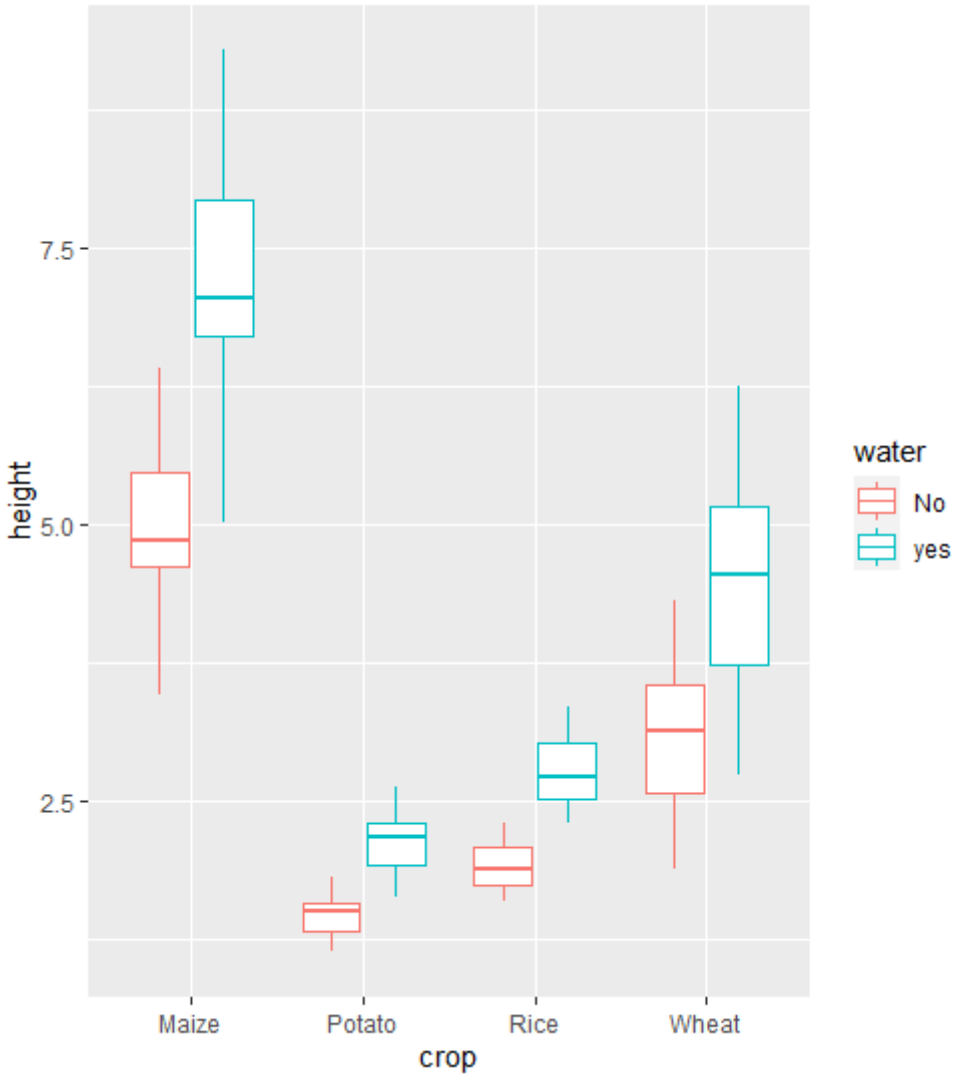
```
> View(ggdata)
> # ggplot
> ggplot(data = x, mapping = aes(x=crop, y=height))+ geom_point()
> ggplot(x, aes(crop, height))+
+   geom_point(size=3)+
+   geom_line()
> ggplot(x, aes(crop, height))+
+   geom_boxplot()+
+   geom_point(size=3, colour="red", alpha= 0.8)
> ggplot(x, aes(crop, height, color= water))+
+   geom_boxplot()
> ggplot(x, aes(crop, height, fill= water))+
+   geom_boxplot()
> ggplot(x, aes(crop, height, fill= water))+
+   geom_boxplot()+
+   facet_wrap(~fert.type)+
+   labs(x="Crop Type", y="Plant height (cm)",
+       title = "Plant Growth")+
+   theme_bw()+
+   ggsave("ggplot.tiff", units = "in", width = 8, height = 6, dpi = 300,
compression = "lzw")
> ggplot(x, aes(crop, height, fill= water))+
+   geom_boxplot()+
+   facet_wrap(~fert.type)+
+   labs(x="Crop Type", y="Plant height (cm)",
+        title = "Plant Growth")+
+   theme_bw()+
+   coord_flip()+
+   ggsave("ggplot.tiff", units = "in", width = 8, height = 6, dpi = 300,
compression = "lzw")
```

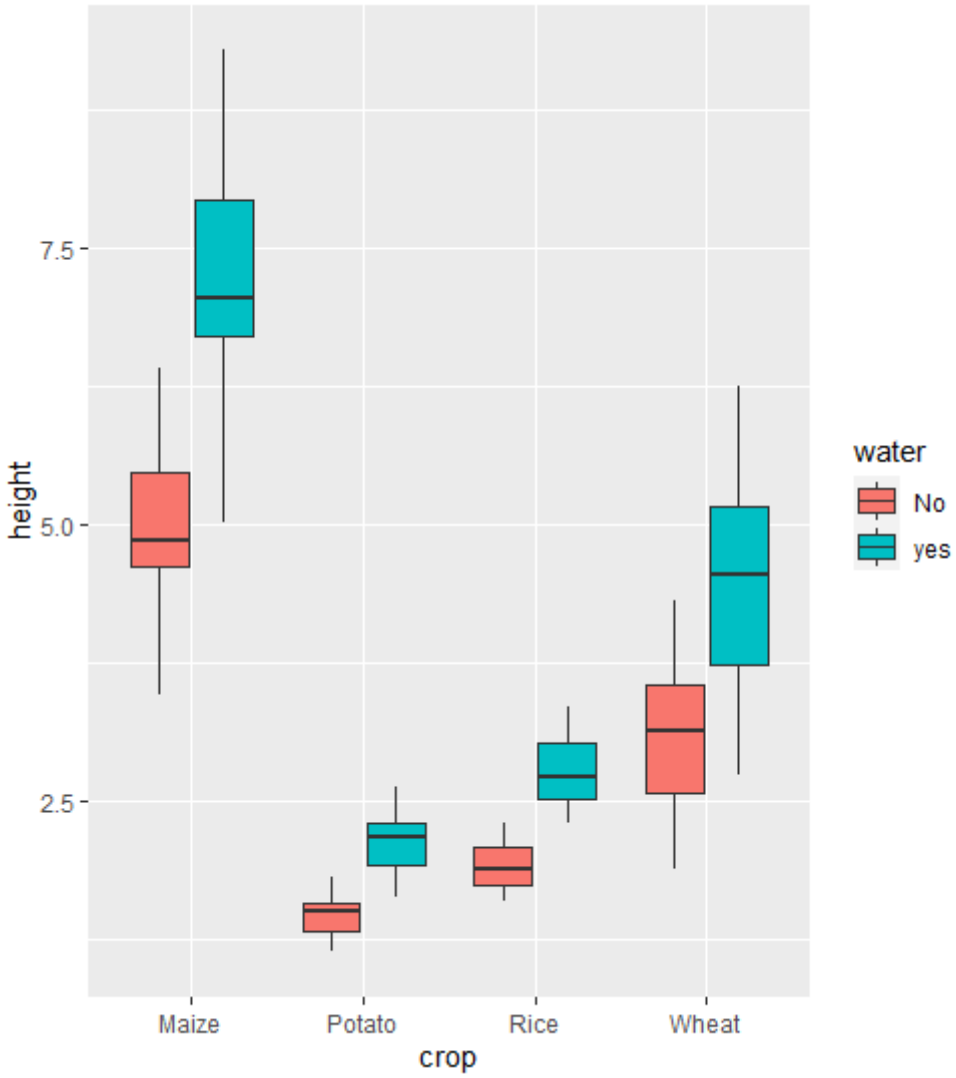

height
7.5
5.0
2.5
Maize
Potato
Rice
Wheat
crop


height
7.5
5.0
2.5


Maize
Potato
Rice
Wheat
crop


7.5
5.0
2.5
height
Maize
Potato
Rice
Wheat
crop


height
7.5
5.0
2.5
Maize
Potato
Rice
Wheat
crop
water
No
yes



class 8 Built-in Datasets in R

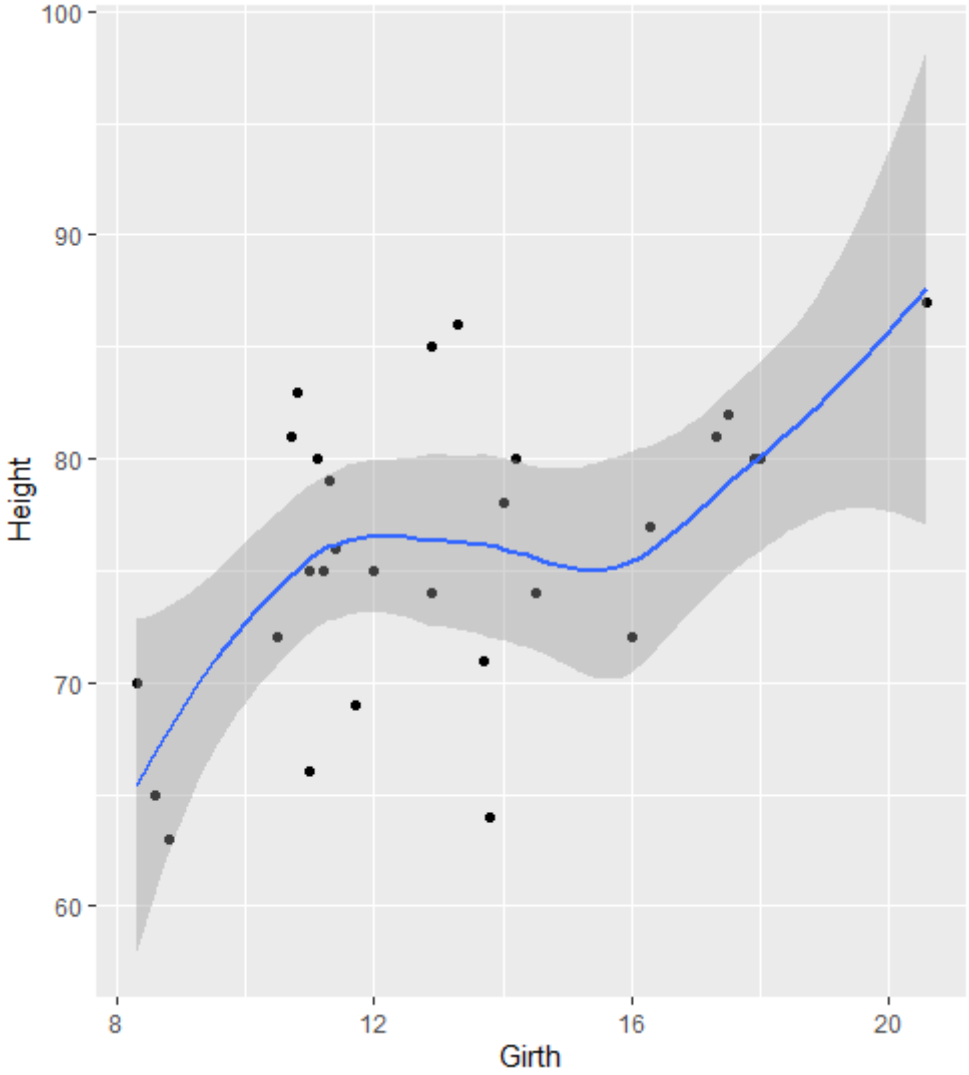
```
# bulitin datasets in r
data()

# co2 dataset
View(CO2)

#other datasets
View(PlantGrowth)
View(iris)
View(mtcars)
View(trees)
View(npk)

# check data set
head(CO2)
names(CO2)
```

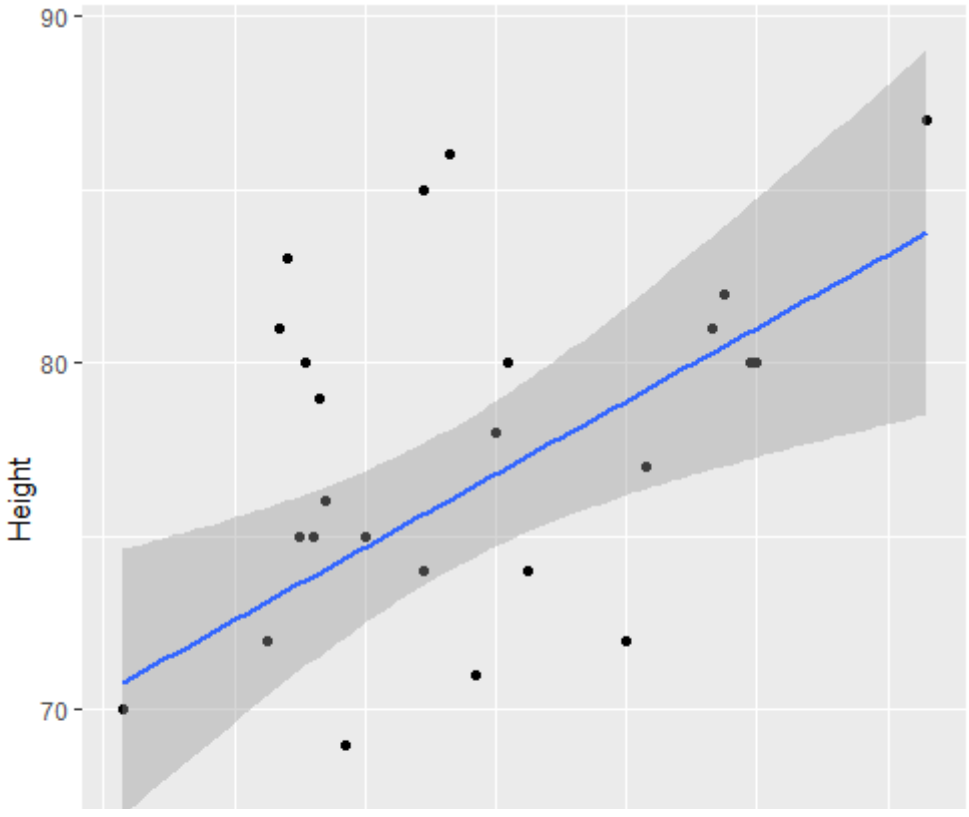
```
nrow(CO2)
ncol(CO2)

# Save the bulitin data set
install.packages("writexl")

library(writexl)
write_xlsx(CO2, path = "D:\\R\\R-practice\\CO2.xlsx")

# gg plot 2
library(ggplot2)

ggplot(trees,aes(Girth, Height))+
  geom_point()+
  geom_smooth()

# linear trend

ggplot(trees,aes(Girth, Height))+
  geom_point()+
  geom_smooth(method = "lm")
```

output

```
> # bulitin datasets in r
> data()
> # co2 dataset
> View(CO2)
> View(PlantGrowth)
> View(iris)
> View(mtcars)
> View(trees)
> View(npk)
> head(CO2)
Grouped Data: uptake ~ conc | Plant
  Plant   Type  Treatment conc uptake
1   Qn1 Quebec nonchilled   95   16.0
2   Qn1 Quebec nonchilled  175   30.4
3   Qn1 Quebec nonchilled  250   34.8
4   Qn1 Quebec nonchilled  350   37.2
5   Qn1 Quebec nonchilled  500   35.3
6   Qn1 Quebec nonchilled  675   39.2
> names(CO2)
[1] "Plant"     "Type"      "Treatment" "conc"      "uptake"
> nrow(CO2)
[1] 84
> ncol(CO2
```

```
+ library(writexl)
Error: unexpected symbol in:
"ncol(CO2
library"
> library(writexl)
> write_xlsx(CO2, path = "D:\\R\\R-practice\\CO2.xlsx")
> # gg plot 2
> library(ggplot2)
> ggplot(trees,aes(Girth, Height))+
+   geom_point()+
+   geom_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
> ggplot(trees,aes(Girth, Height))+
+   geom_point()+
+   geom_smooth(method = "lm")
`geom_smooth()` using formula 'y ~ x'
```


100
90
80
70
60
Height
8
12
16
20
Girth


90
80
70
Height

Class 9 Hash-tags (# tags) in R

```
# bulitin datasets in r

# How to use bulit in data sets in r
data() # All data sets in r

# co2 dataset
View(CO2) # to view data set in Console

#other dataset
View(PlantGrowth)
View(iris)
View(mtcars)
View(trees)
View(npk) # NPK data for factorial design

# check data set
head(CO2) # to show first 6 rows of data
names(CO2) # to show the names of column
nrow(CO2)# to look for the total number of rows (sample size)
ncol(CO2) # to look for the number of columns ( number of variables)

# to Save the bulit in data set in xlsx
install.packages("writexl")

library(writexl)
write_xlsx(CO2, path = "D:\\R\\R-practice\\CO2.xlsx")

# gg plot 2
library(ggplot2)

ggplot(trees,aes(Girth, Height))+
  geom_point()+
  geom_smooth() # Scatter plot

# linear trend
library(ggplot2)
```

```
ggplot(trees,aes(Girth, Height))+
  geom_point()+
  geom_smooth(method = "lm")
```

Class 10 Vectors/Arrays in R

```
# Vector in r

# Definition of vector
# " Vector are the one column of your excel sheet"

v1 <- c(3, 4, 6, 8, 9, 13)
v2 <- c(12, 13, 13, 2, 12, 9)

print(v1)
print(v2)

# Conbine both vectors
v3 <- c(v1, v2)
v1+v2
v1*v2
v1+v3

 # vectors for strings
s1 <- c("1", "love", "R", "Ammar")
print(s1)

s2 <- c(v1, s1)
```

output

```
> # Vector in r
>
> # Definition of vector
> # " Vector are the one column of your excel sheet"
>
> v1 <- c(3, 4, 6, 8, 9, 13)
> v2 <- c(12, 13, 13, 2, 12, 9)
>
> print(v1)
[1]  3  4  6  8  9 13
> print(v2)
[1] 12 13 13  2 12  9
>
```

```
> # Conbine both vectors
> v3 <- c(v1, v2)
> v1+v2
[1] 15 17 19 10 21 22
> v1*v2
[1]  36  52  78  16 108 117
> v1+v3
 [1]  6  8 12 16 18 26 15 17 19 10 21 22
>
>  # vectors for strings
> s1 <- c("1", "love", "R", "Ammar")
> print(s1)
[1] "1"     "love"  "R"     "Ammar"
>
> s2 <- c(v1, s1)
```

Class 11 Sequence and Repeats in R

```
# sequence and repeats

seq(from= 0, to= 100)
seq(from=1, to=121)
seq(1,121)

# differnec in number sequency
seq(1, 100, by=5)
seq(0, 100, by=10)
seq1 <- seq(2, 33, by=1.3)
seq1

# rep

rep("hello", 3)
rep("hello", times= 3)
rep(123, 100)
rep("Homework", 100)
# repeat each number or letter according to the requriment
rep(1:10, each=3)
rep(1:10, each=3, times=2)

rep("I Love R with Ammar", 100)
```

output

```
> # sequence and repeats
>
> seq(from= 0, to= 100)
  [1]   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39
 [41]  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79
 [81]  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
99 100
> seq(from=1, to=121)
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 [41]  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59
60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
 [81]  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
[121] 121
> seq(1,121)
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 [41]  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59
60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
 [81]  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
[121] 121
>
> # differnec in number sequency
> seq(1, 100, by=5)
 [1]  1  6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96
> seq(0, 100, by=10)
 [1]   0  10  20  30  40  50  60  70  80  90 100
> seq1 <- seq(2, 33, by=1.3)
> seq1
 [1]  2.0  3.3  4.6  5.9  7.2  8.5  9.8 11.1 12.4 13.7 15.0 16.3 17.6 18.9 20.2 21.5
22.8 24.1 25.4 26.7 28.0 29.3 30.6 31.9
>
> # rep
>
> rep("hello", 3)
[1] "hello" "hello" "hello"
> rep("hello", times= 3)
[1] "hello" "hello" "hello"
> rep(123, 100)
  [1] 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
 [41] 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
 [81] 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123 123
123
```

```
> rep("Homework", 100)
  [1] "Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
"Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
 [15] "Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
"Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
 [29] "Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
"Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
 [43] "Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
"Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
 [57] "Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
"Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
 [71] "Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
"Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
 [85] "Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
"Homework" "Homework" "Homework" "Homework" "Homework" "Homework" "Homework"
 [99] "Homework" "Homework"
> # repeat each number or letter according to the requriment
> rep(1:10, each=3)
 [1]  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9  9
9 10 10 10
> rep(1:10, each=3, times=2)
 [1]  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9  9
9 10 10 10  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7  8  8  8
[55]  9  9  9 10 10 10
>
> rep("I Love R with Ammar", 100)
  [1] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
  [8] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [15] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [22] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [29] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [36] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [43] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [50] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [57] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [64] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [71] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [78] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
```

```
 [85] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [92] "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R
with Ammar" "I Love R with Ammar" "I Love R with Ammar" "I Love R with Ammar"
 [99] "I Love R with Ammar" "I Love R with Ammar"
```

Class 12 Scatter plots in R with ggplot2

```
# Scatter plot in r
 library(ggplot2)

# Create between two numeric variables
data("cars")
View(cars)

ggplot(data=cars, aes(x=speed, y=dist))+geom_point()+
  geom_smooth(method = "lm", se=T, leve=0.95)


# Scatter plot with multiple lines
data("Orange")
View(Orange)

ggplot(Orange, aes(age, circumference))+
  geom_point()

# Colour the vises of tress
ggplot(Orange, aes(age, circumference, color=Tree))+
  geom_point(size=4)

# change the size and shapes of the points
ggplot(Orange, aes(age, circumference, color=Tree))+
  geom_point(size=4, shape= 19)

# Sperate by shapes
ggplot(Orange, aes(age, circumference, color=Tree))+
  geom_point(size=4, aes(shape=Tree))

# use previous
ggplot(Orange, aes(age, circumference, color=Tree))+
  geom_point(size=4, shape=19)+
  geom_line(linetype=1, size=1)

# Change the line type
ggplot(Orange, aes(age, circumference, color=Tree))+
  geom_point(size=4, shape=19)+
```

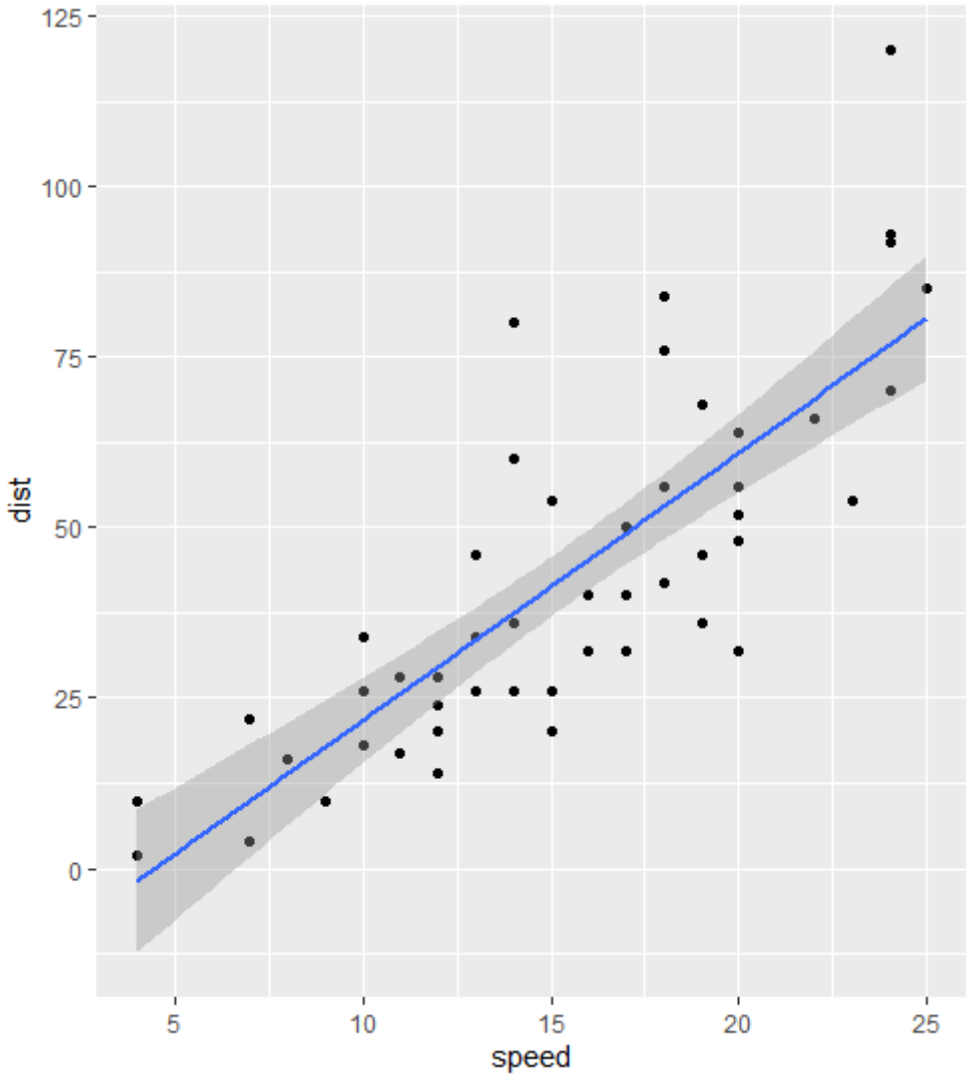
```
  geom_line(linetype=2, size=1)

# using aes
ggplot(Orange, aes(age, circumference, color=Tree))+
  geom_point(size=4, shape=19)+
  geom_line(aes(linetype=Tree), size=1)+
  labs(x="age", y=" circumference", title = "Scatter plot")



# Bubble plots
install.packages("viridis")
library(viridis)
data("quakes")
View(quakes)
nrow(quakes)

# large data set so we take sample
q_sample <- quakes[seq(from=1, to=1000, by=10),]
?viridis

# Checking rows
nrow(q_sample)

# Creating graph
ggplot(data = q_sample, aes(x=lat, y=long))+
  geom_point()

# Changes
ggplot(data = q_sample, aes(x=lat, y=long))+
  geom_point(aes(size=mag, color=mag))+
  guides(size=F)+
  scale_color_viridis_b(option = "B")+
  scale_size_continuous(range = c(1,9))+
  labs(x="Latitude", y=" Longitude", title = "Bubble plot")+
  ggsave("bubbleplot.pdf")


# jjtter graph
data("diamonds")
View(diamonds)
nrow(diamonds)

# Taking sample
d_sample <- diamonds[seq(from=1, to= 49000, by=10), ]
nrow(d_sample)

# Creating jjtter grapg

# ggplot(d_sample, aes(cut, price))+geom_point()
```

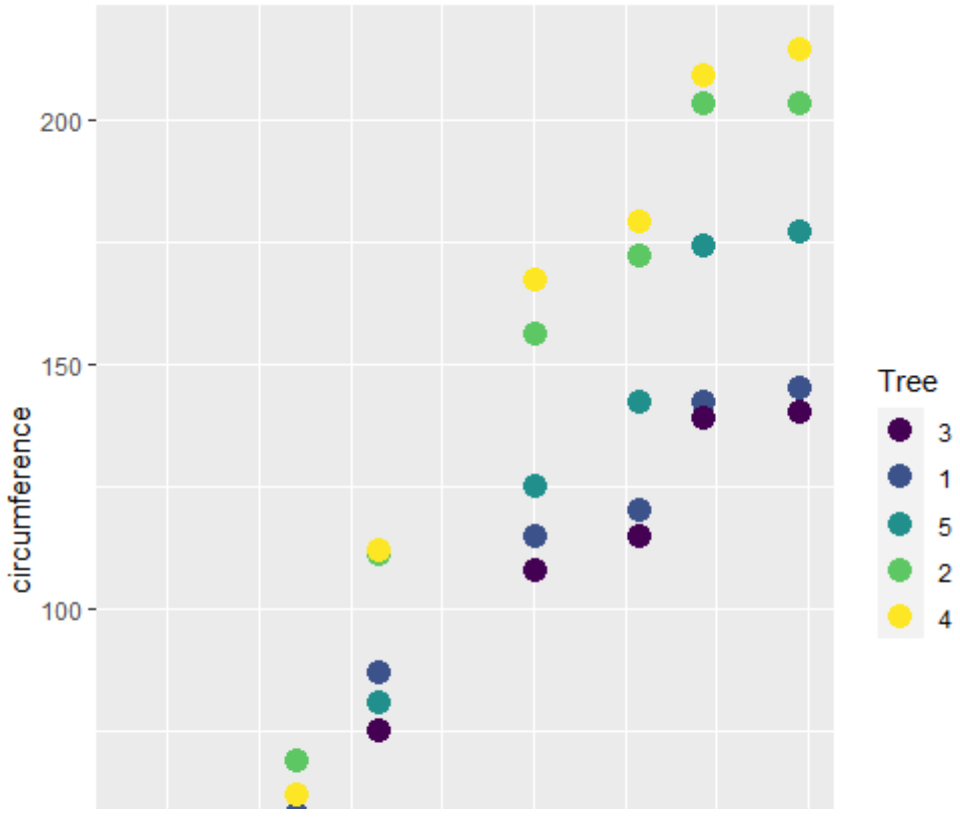
```
ggplot(d_sample, aes(cut, price, color= cut))+geom_jitter()
```

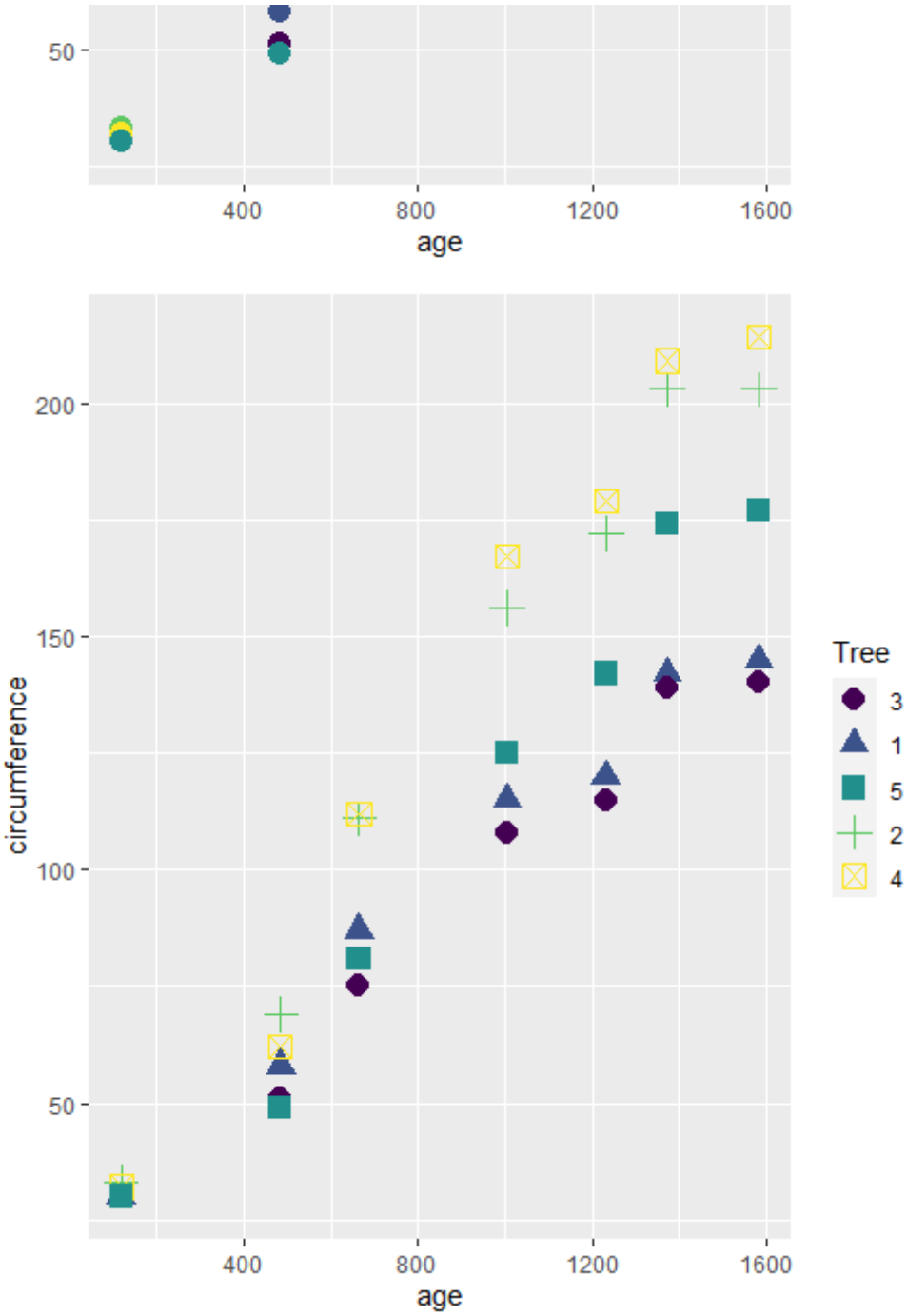
output

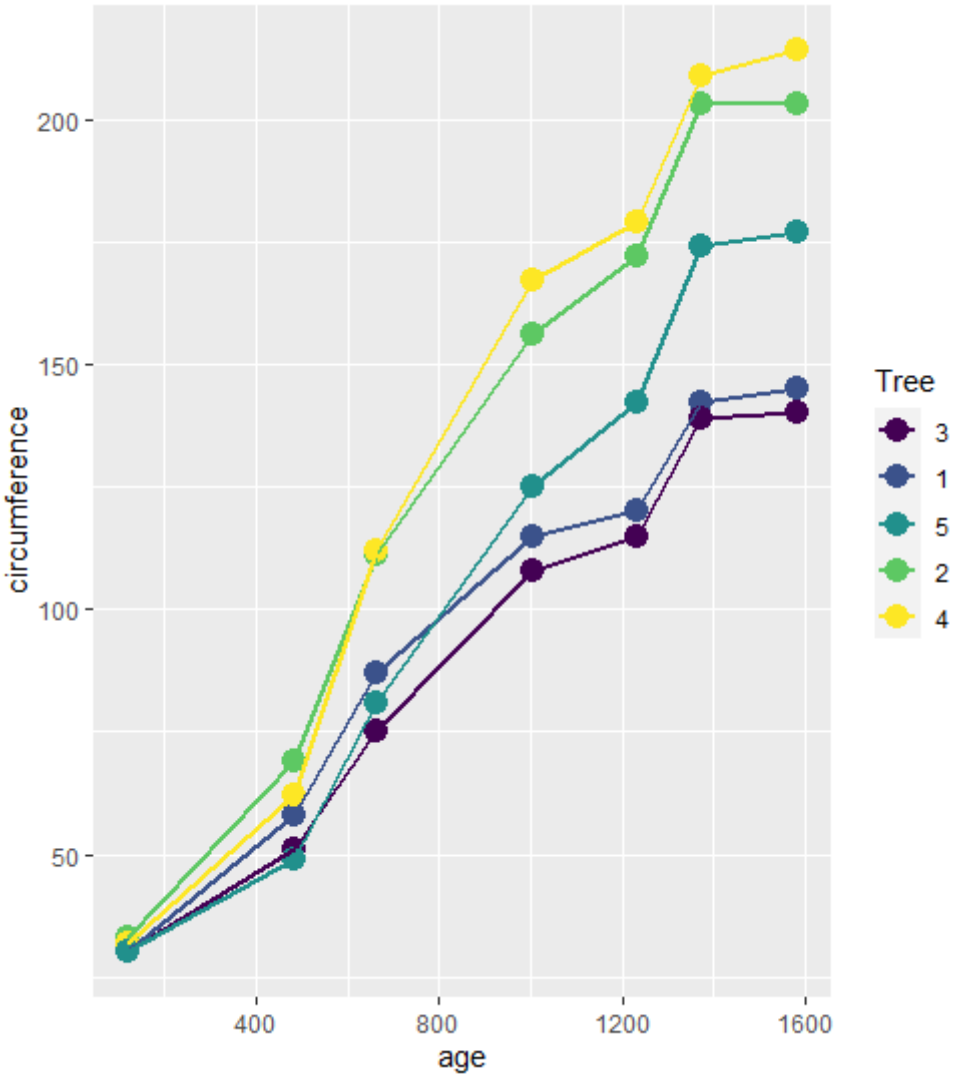
```
> # Scatter plot in r
>  library(ggplot2)
> data("cars")
> View(cars)
> ggplot(data=cars, aes(x=speed, y=dist))+geom_point()+
+   geom_smooth(method = "lm", se=T, leve=0.95)
`geom_smooth()` using formula 'y ~ x'
Warning message:
Ignoring unknown parameters: leve
> data("Orange")
> View(Orange
+ ggplot(Orange, aes(age, circumference))+
Error: unexpected symbol in:
"View(Orange
ggplot"
>   geom_point()
geom_point: na.rm = FALSE
stat_identity: na.rm = FALSE
position_identity
> ggplot(Orange, aes(age, circumference, color=Tree))+
+   geom_point(size=4)
> ggplot(Orange, aes(age, circumference, color=Tree))+
+   geom_point(size=4, shape= 19)
> ggplot(Orange, aes(age, circumference, color=Tree))+
+   geom_point(size=4, aes(shape=Tree))
Warning message:
Using shapes for an ordinal variable is not advised
> ggplot(Orange, aes(age, circumference, color=Tree))+
+   geom_point(size=4, shape=19)+
+   geom_line(linetype=1, size=1)
> ggplot(Orange, aes(age, circumference, color=Tree))+
+   geom_point(size=4, shape=19)+
+   geom_line(linetype=2, size=1)
> ggplot(Orange, aes(age, circumference, color=Tree))+
+   geom_point(size=4, shape=19)+
+   geom_line(aes(linetype=Tree), size=1)+
+   labs(x="age", y=" circumference", title = "Scatter plot")
> library(viridis)
> data("quakes")
> View(quakes)
> nrow(quakes)
[1] 1000
```

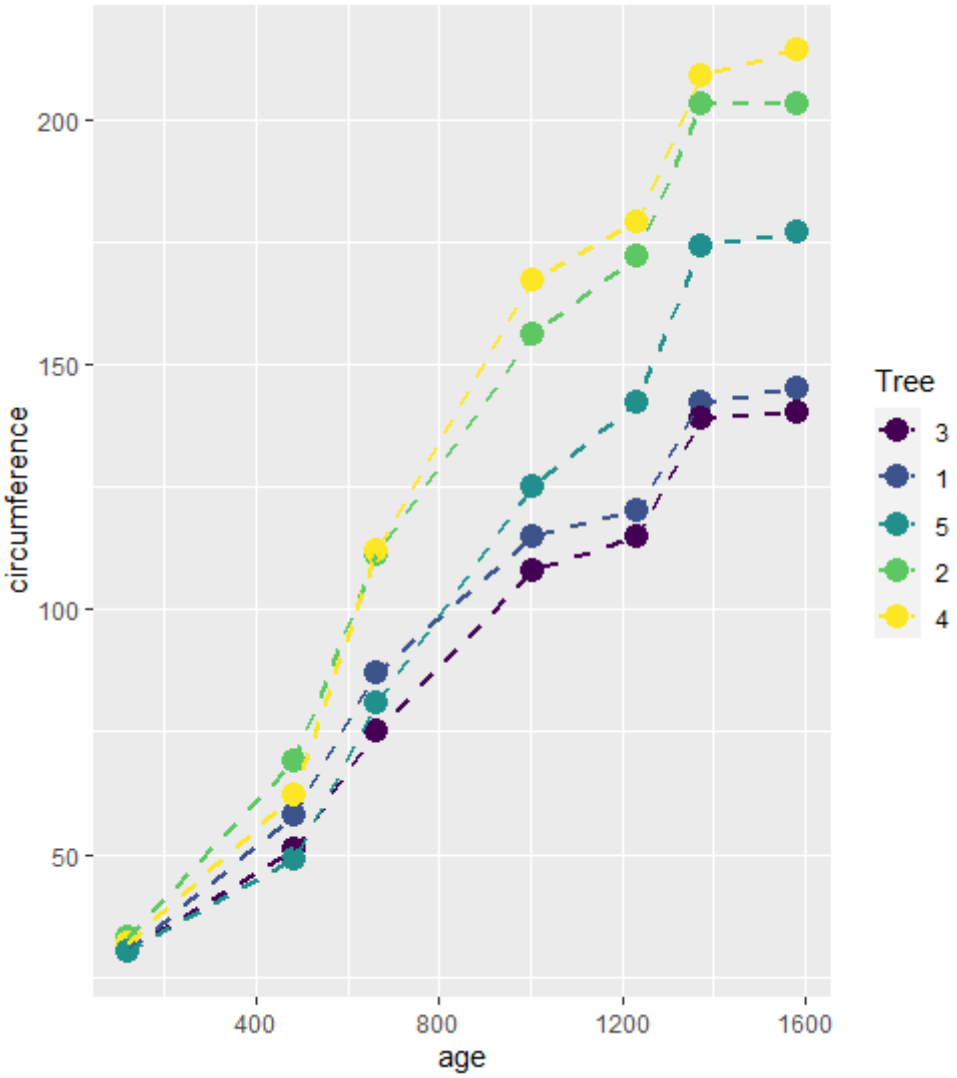
```
> # large data set so we take sample
> q_sample <- quakes[seq(from=1, to=1000, by=10),]
> # Checking rows
> nrow(q_sample)
[1] 100
> # Creating graph
> ggplot(data = q_sample, aes(x=lat, y=long))+
+   geom_point()
> ggplot(data = q_sample, aes(x=lat, y=long))+
+   geom_point(aes(size=mag, color=mag))+
+   guides(size=F)+
+   scale_color_viridis_b(option = "B")+
+   scale_size_continuous(range = c(1,9))+
+   labs(x="Latitude", y=" Longitude", title = "Bubble plot")
Warning message:
`guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> = "none")`
instead.
> data("diamonds")
> View(diamonds)
> nrow(diamonds)
[1] 53940
> # Taking sample
> d_sample <- diamonds[seq(from=1, to= 49000, by=10), ]
> nrow(d_sample)
[1] 4900
> ggplot(d_sample, aes(cut, price, color= cut))+geom_jitter()
```

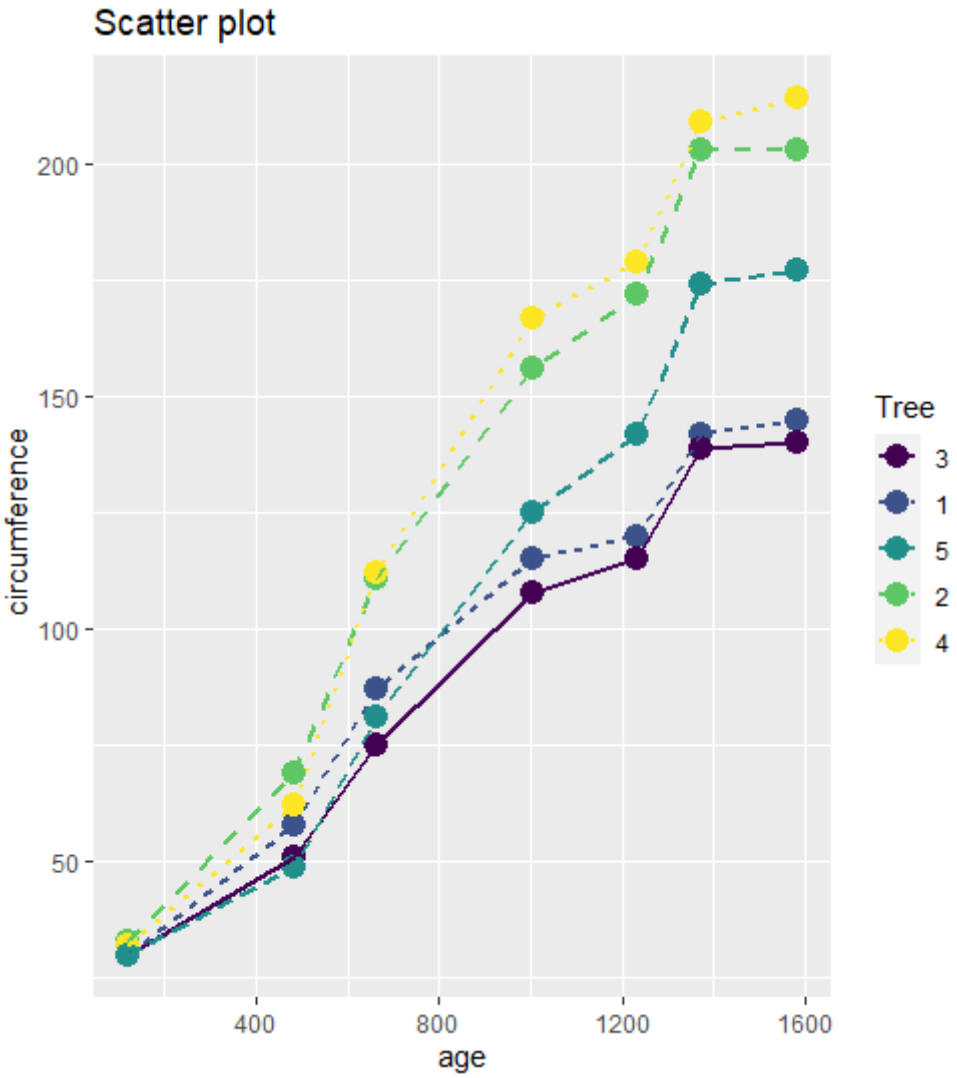

125
100
75
50
25
0
dist
5
10
15
20
25
speed

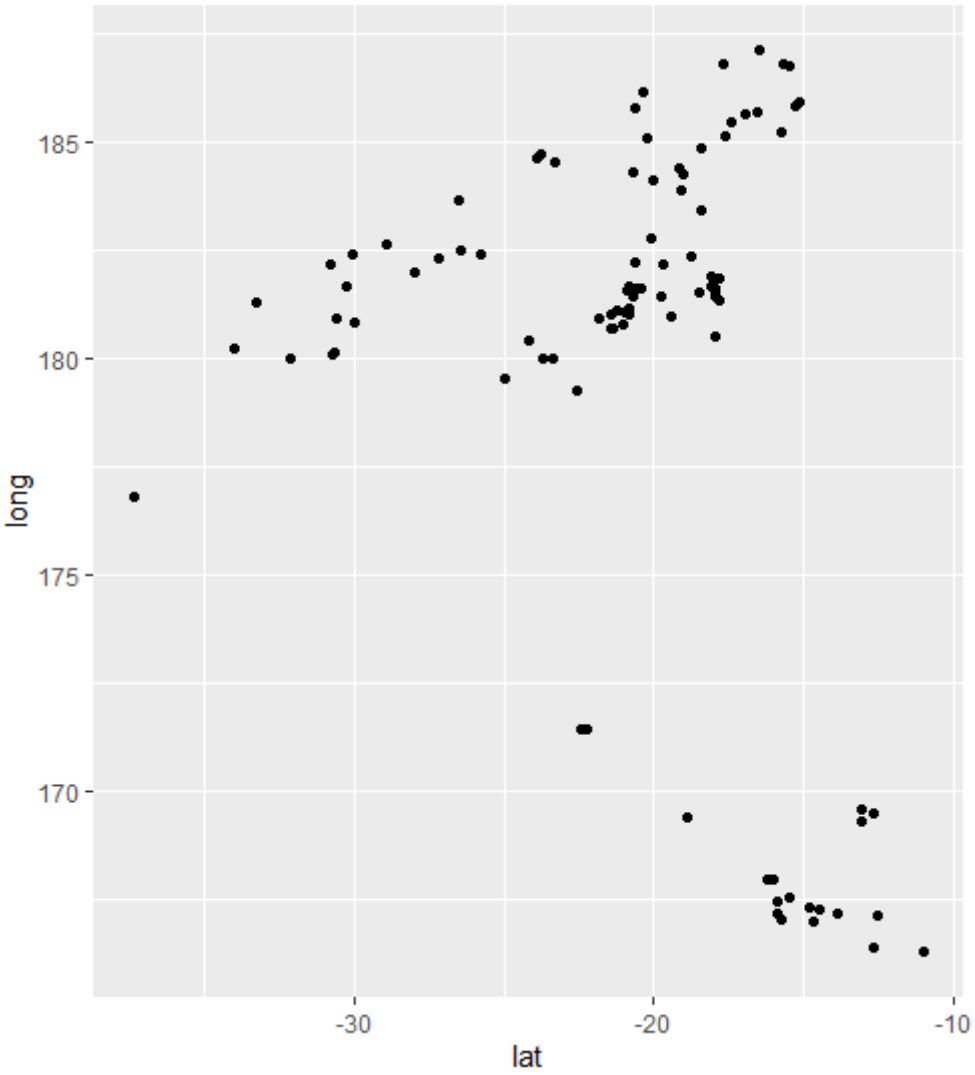

200
150
100
circumference
Tree
3
1
5
2
4

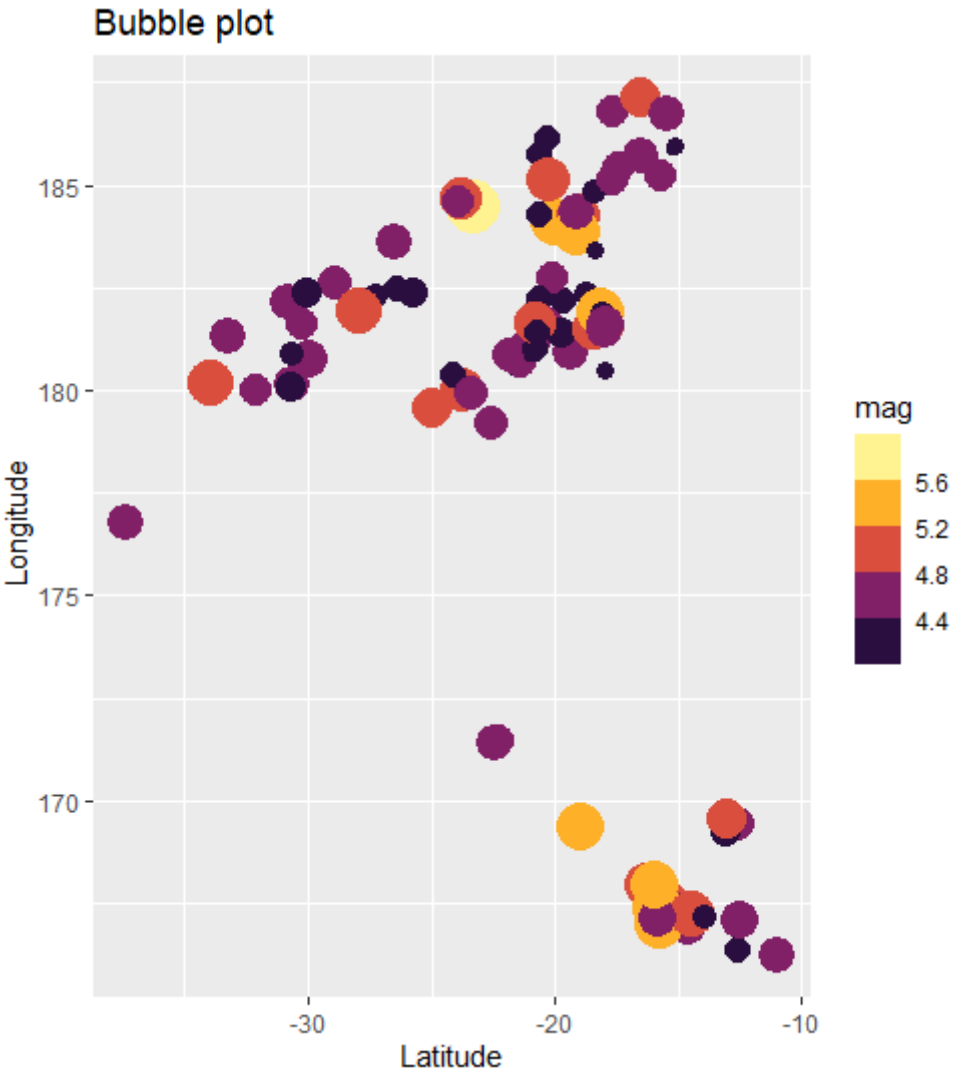

50
400
800
1200
1600
age
200
150
100
50
circumference
400
800
1200
1600
age
Tree
3
1
5
2
4

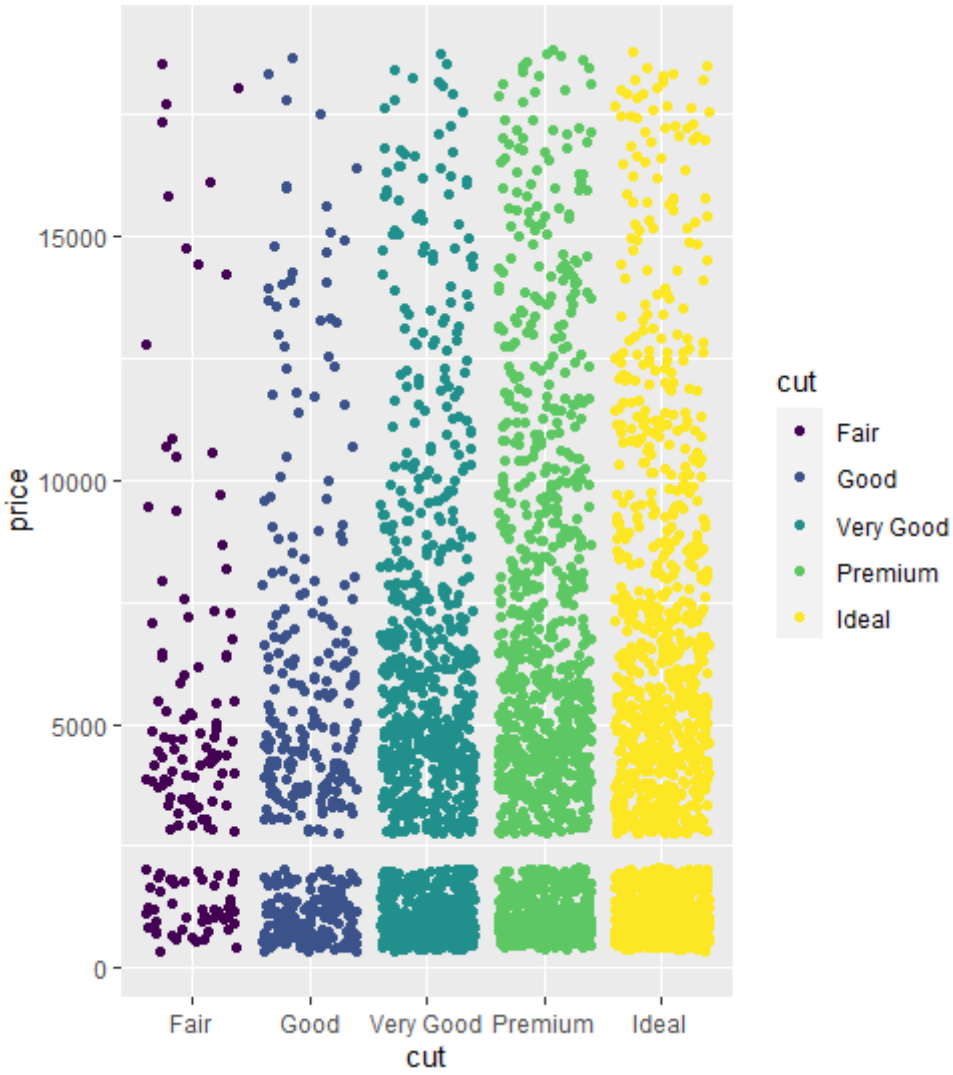

200
150
100
50
circumference
400
800
1200
1600
age
Tree
3
1
5
2
4


200
150
100
50
circumference
400
800
1200
1600
age
Tree
3
1
5
2
4


Scatter plot
circumference
200
150
100
50
400
800
1200
1600
age
Tree
3
1
5
2
4


185
180
175
170
long
-30
-20
-10
lat


Bubble plot
185
180
175
170
Longitude
-30
-20
-10
Latitude
mag
5.6
5.2
4.8
4.4



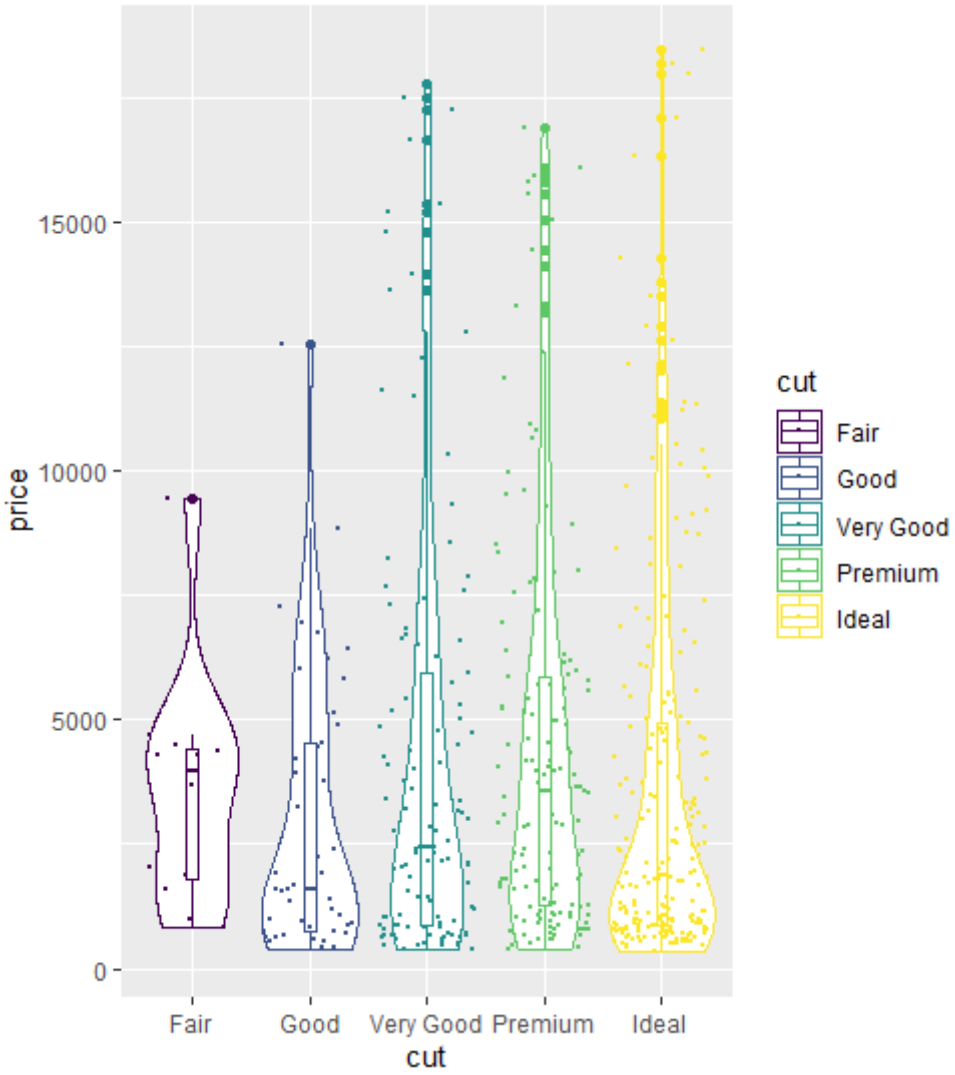
Class 13 Violin plots in R with ggplot2

```
# Violin plot

# geom_violin()
library(ggplot2)

# Load data set
data("diamonds")
View(diamonds)
nrow(diamonds)

# Taking sample
d_sample <- diamonds[seq(from=1, to= 53000, by=100), ]
nrow(d_sample)

# plot
p <- ggplot(data = d_sample, aes(x= cut, y= price, color=cut))+
```

```
  geom_violin()

p + geom_boxplot(width=0.1)+geom_jitter(size=0.5)
p+ggsave("box and violin plot.pdf")
```

output

```
> # Violin plot
> # geom_violin()
> library(ggplot2)
> data("diamonds")
> View(diamonds)
> nrow(diamonds)
[1] 53940
> # Taking sample
> d_sample <- diamonds[seq(from=1, to= 53000, by=100), ]
> nrow(d_sample)
[1] 530
> # plot
> p <- ggplot(data = d_sample, aes(x= cut, y= price, color=cut))+
+   geom_violin()
> p + geom_boxplot(width=0.1)+geom_jitter(size=0.5)
> p+ggsave("box and violin plot.pdf")
```



class 14 Principal Component Analysis in R

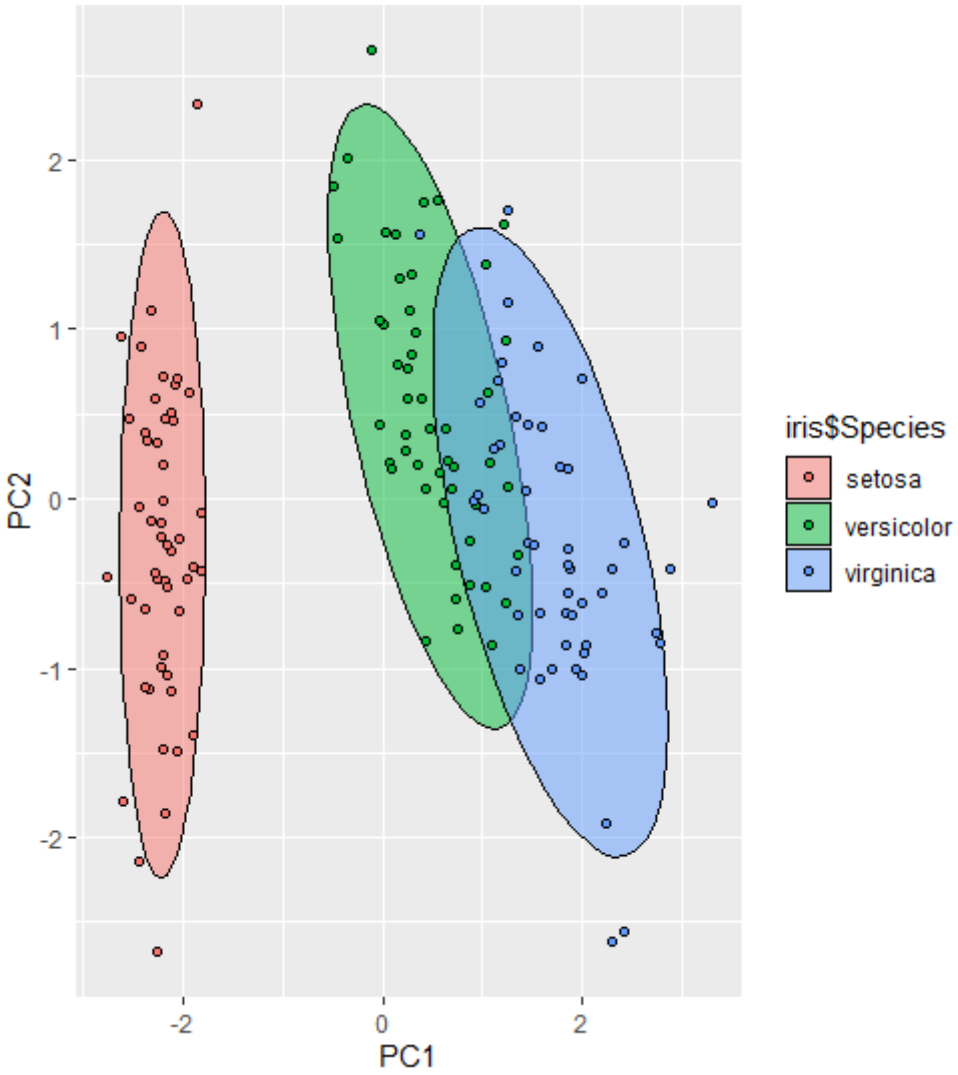
```
# Principal component analysis
# how perform in R

data("iris")
View(iris)

#PCA
x <- prcomp(iris[, -5], center = TRUE, scale. = TRUE)
print(x)
summary(x)

# ggplot2
library(ggplot2)

iris <- cbind(iris, x$x)
```

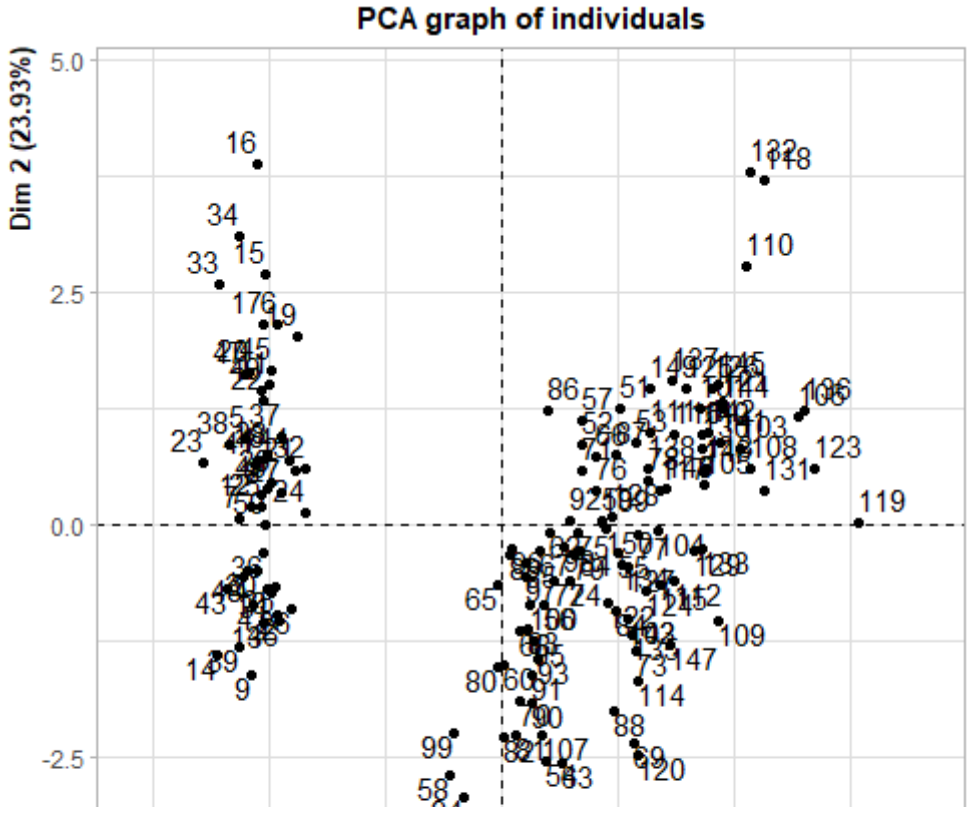
```
ggplot(iris, aes(PC1, PC2, col=iris$Species, fill= iris$Species))+
  stat_ellipse(geom = "polygon", col= "black", alpha=0.5)+
  geom_point(shape=21, col="black")

# PCA PLOT
install.packages("factoextra")
install.packages("FactoMineR")
library(factoextra)
library(FactoMineR)

# PCA Table
iris.pca <- PCA(iris[, -5], graph = TRUE, scale.unit = TRUE)

# Scree plot
fviz_eig(iris.pca, addlabels = TRUE, ylim=c(0,70))

# PCA plot
fviz_pca_var(iris.pca, col.var = "cos2",
             gradient.col= c("#FFCC00", "#CC9933", "#660033", "#330033"),
             repel = TRUE)+
  labs(title = "PCA of Prarmeter", x= "PC1(49%)", y= "PC2(23.9%)",
       colour= "cos2")+
  ggsave("PCA.png",  units = "in", width=6.5, height=5.5)
```

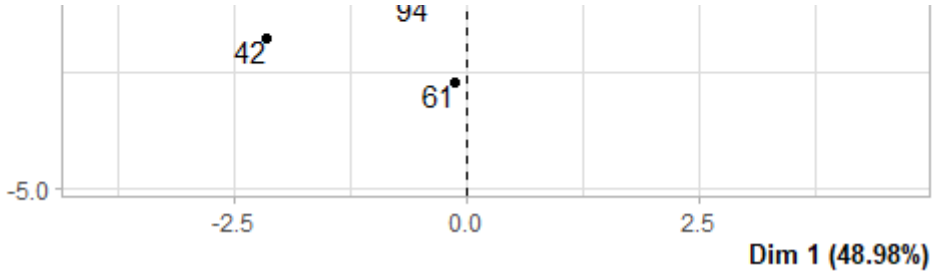
output

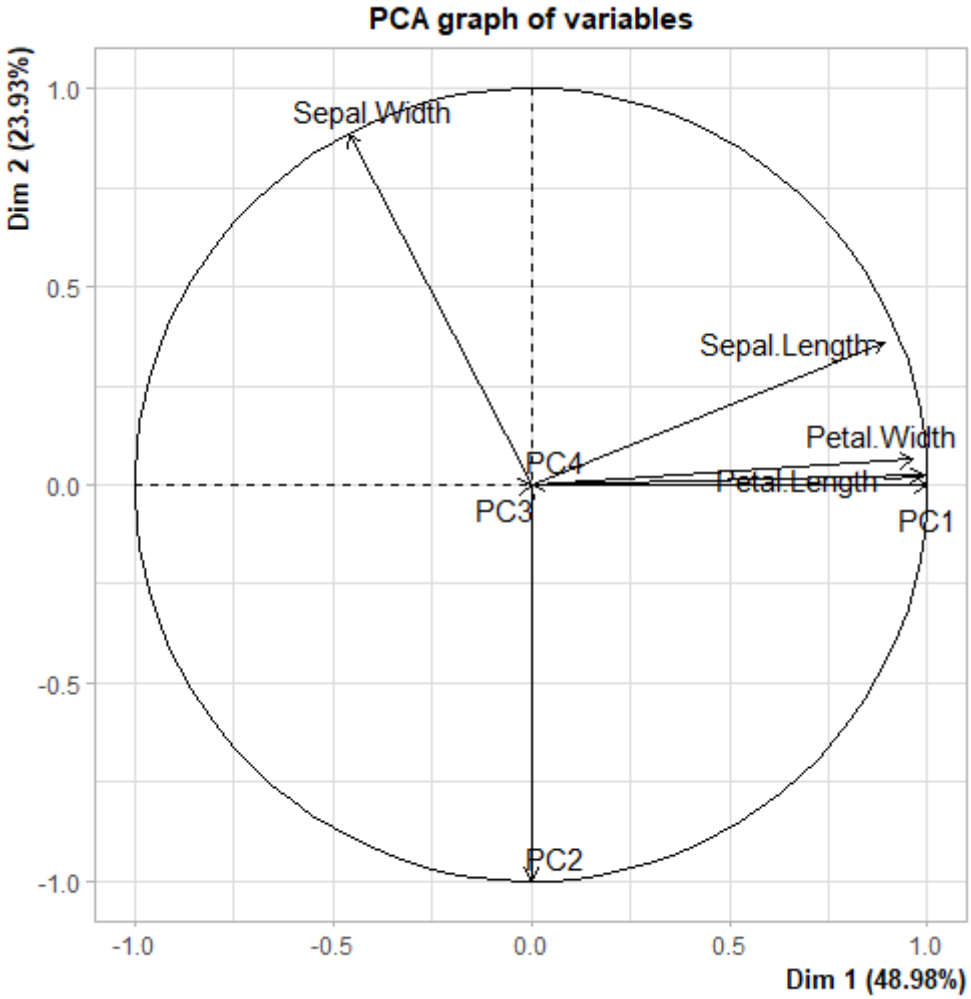
```
> data("iris")
> View(iris)
> x <- prcomp(iris[, -5], center = TRUE, scale. = TRUE)
> print(x)
Standard deviations (1, .., p=4):
[1] 1.7083611 0.9560494 0.3830886 0.1439265

Rotation (n x k) = (4 x 4):
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
> summary(x)
Importance of components:
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
> iris <- cbind(iris, x$x)
> ggplot(iris, aes(PC1, PC2, col=iris$Species, fill= iris$Species))+
```

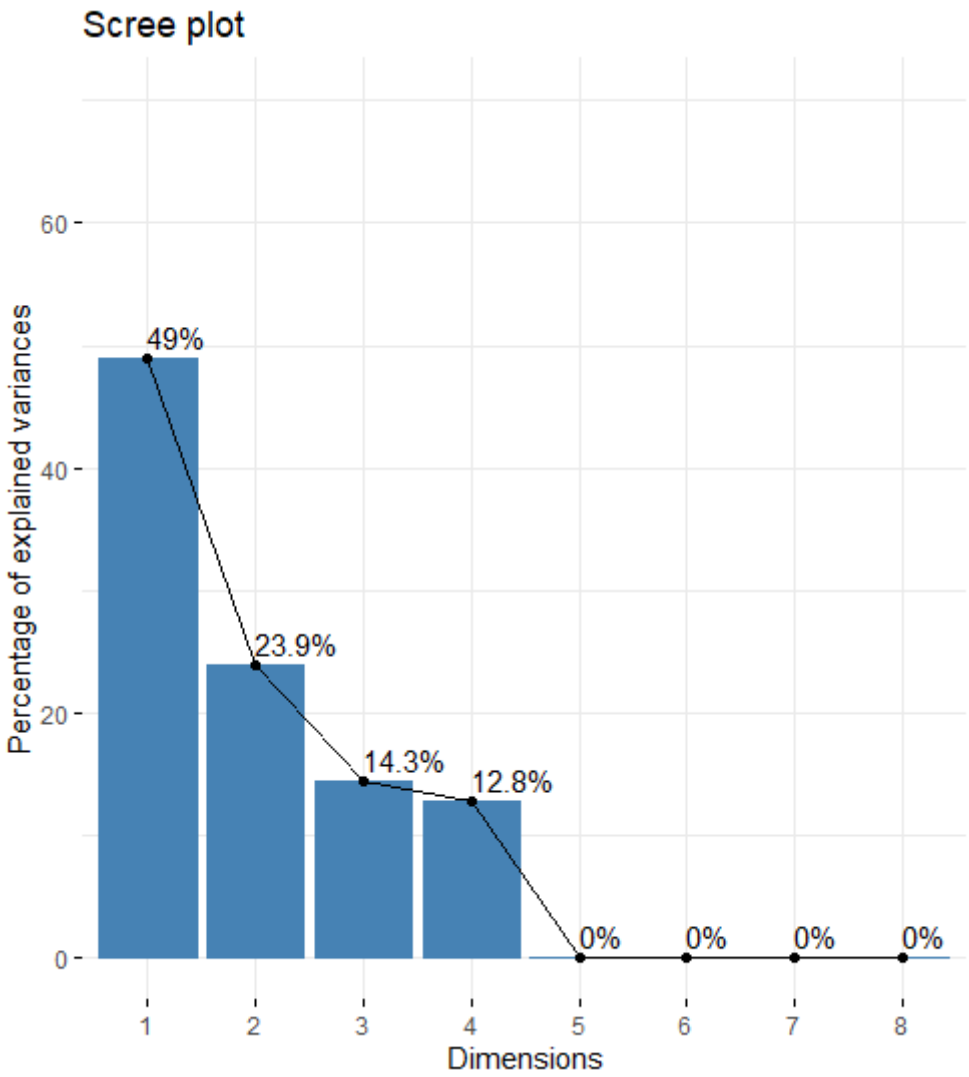
```
+   stat_ellipse(geom = "polygon", col= "black", alpha=0.5)+
+   geom_point(shape=21, col="black")
Warning messages:
1: Use of `iris$Species` is discouraged. Use `Species` instead.
2: Use of `iris$Species` is discouraged. Use `Species` instead.
> library(factoextra)
> library(FactoMineR)
> # PCA Table
> iris.pca <- PCA(iris[, -5], graph = TRUE, scale.unit = TRUE)
> # Scree plot
> fviz_eig(iris.pca, addlabels = TRUE, ylim=c(0,70))
> # PCA plot
> fviz_pca_var(iris.pca, col.var = "cos2",
+              gradient.col= c("#FFCC00", "#CC9933", "#660033", "#330033"),
+              repel = TRUE)+
+   labs(title = "PCA of Prarmeter", x= "PC1(49%)", y= "PC2(23.9%)",
+        colour= "cos2")+
+   ggsave("PCA.png",  units = "in", width=6.5, height=5.5)
```


PC2
PC1
iris$Species
setosa
versicolor
virginica


PCA graph of individuals
Dim 2 (23.93%)


94
42
61
-5.0
-2.5
0.0
2.5
Dim 1 (48.98%)


PCA graph of variables
Dim 2 (23.93%)
1.0
0.5
0.0
-0.5
-1.0
Sepal.Width
Sepal.Length
Petal.Width
Petal.Length
PC4
PC3
PC1
PC2
-1.0
-0.5
0.0
0.5
1.0
Dim 1 (48.98%)



Class 15 Heatmaps in R

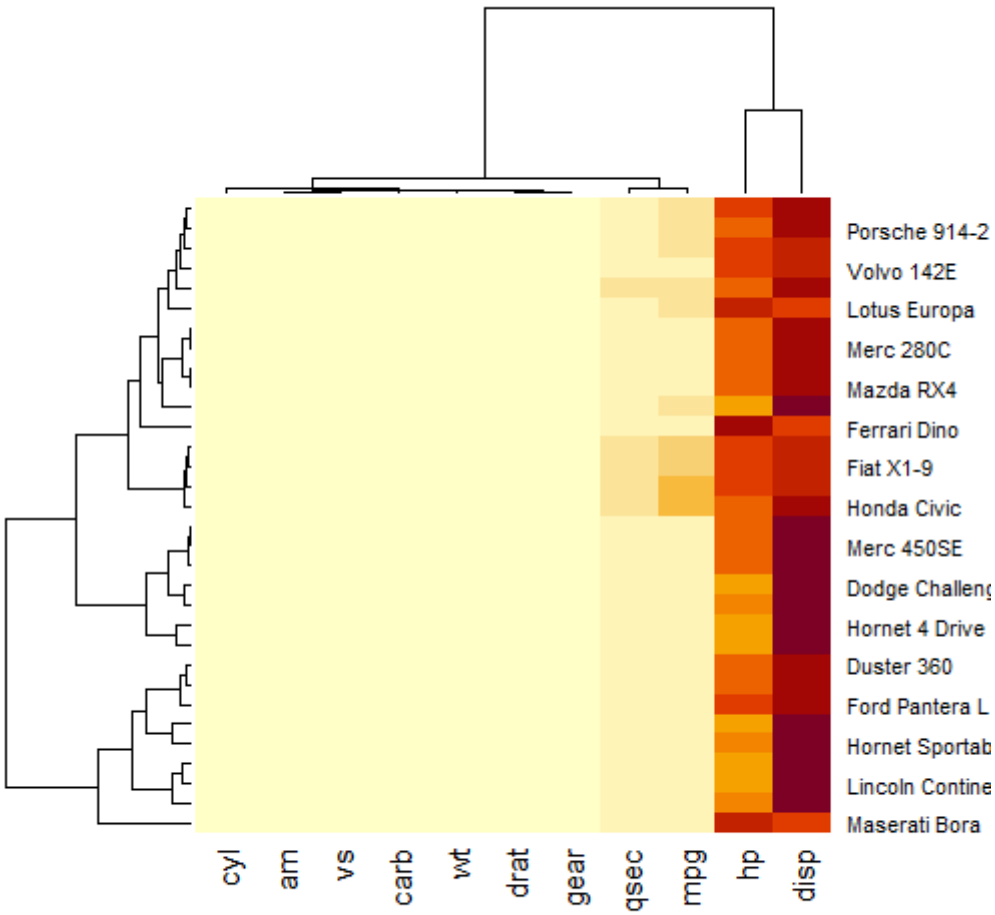
```
# 1- Heat map
View(mtcars)
x <- mtcars
heatmap(mtcars)

# Data should be in numeric matrix
x <- as.matrix(mtcars)
heatmap(x)

# Blance units of data
heatmap(x, scale = "column")

# gplots
library(gplots)

heatmap.2(x, scale = "column", col = bluered(100))
```

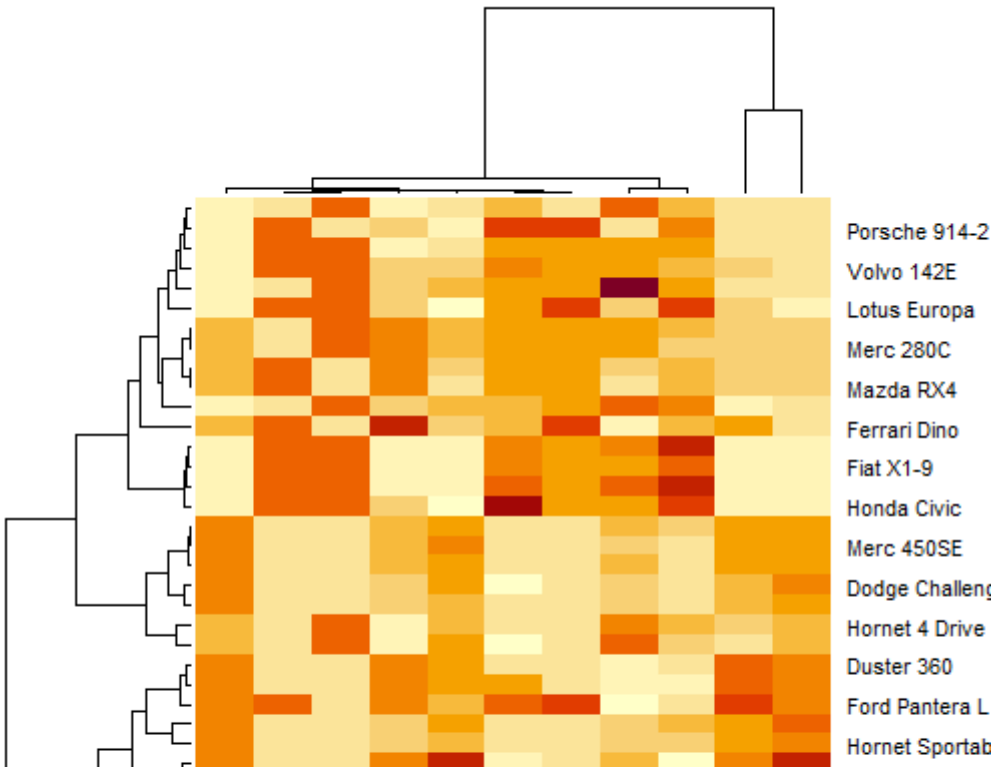
```
# Remove tracing
heatmap.2(x, scale = "column", col = bluered(100),
          trace = "none")

# Help
?heatmap.2()

# 3- pheatmap
# install.packages("pheatmap")
library(pheatmap)

pheatmap(x, scale = "column")

# cut after 4 rows
pheatmap(x, scale = "column", cutree_rows = 4)

# cut in cloumns
pheatmap(x, scale = "column", cutree_cols = 2)

#4- method
library(ggplot2)

# load data
y <- iris

# for reshape dta intall package
install.packages("reshape")
library(reshape)
y1 <- melt(iris)

ggplot(y1, aes(y1$Species, y1$variable, fill= y1$value))+
  geom_tile()+
  scale_fill_gradient(low = "red", high = "green")
```

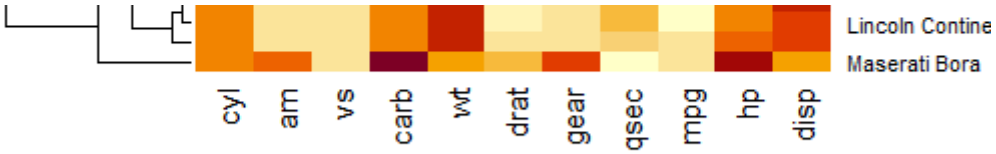
output

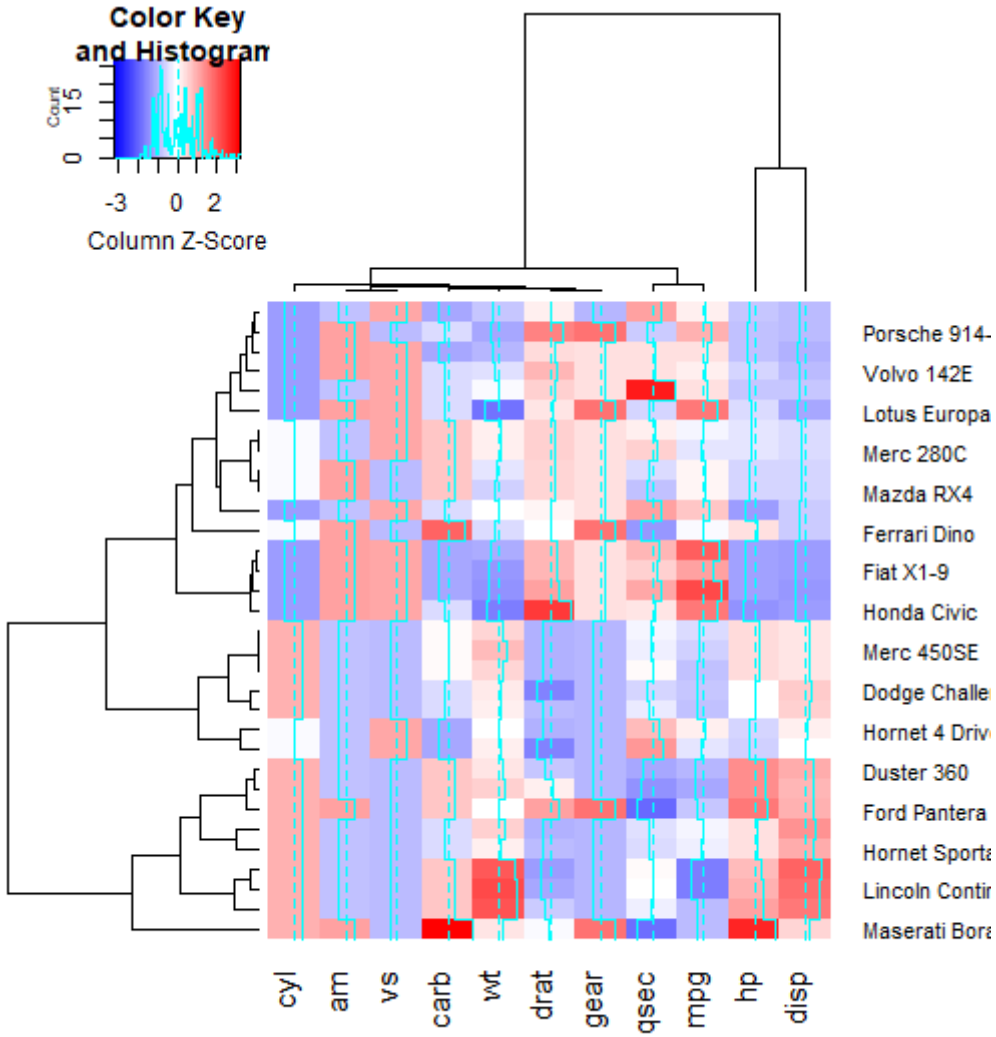
```
> View(mtcars)
> x <- mtcars
> heatmap(mtcars)
Error in heatmap(mtcars) : 'x' must be a numeric matrix
> # Data should be in numeric matrix
> x <- as.matrix(mtcars)
> heatmap(x)
> # Blance units of data
> heatmap(x, scale = "column")
> # gplots
```

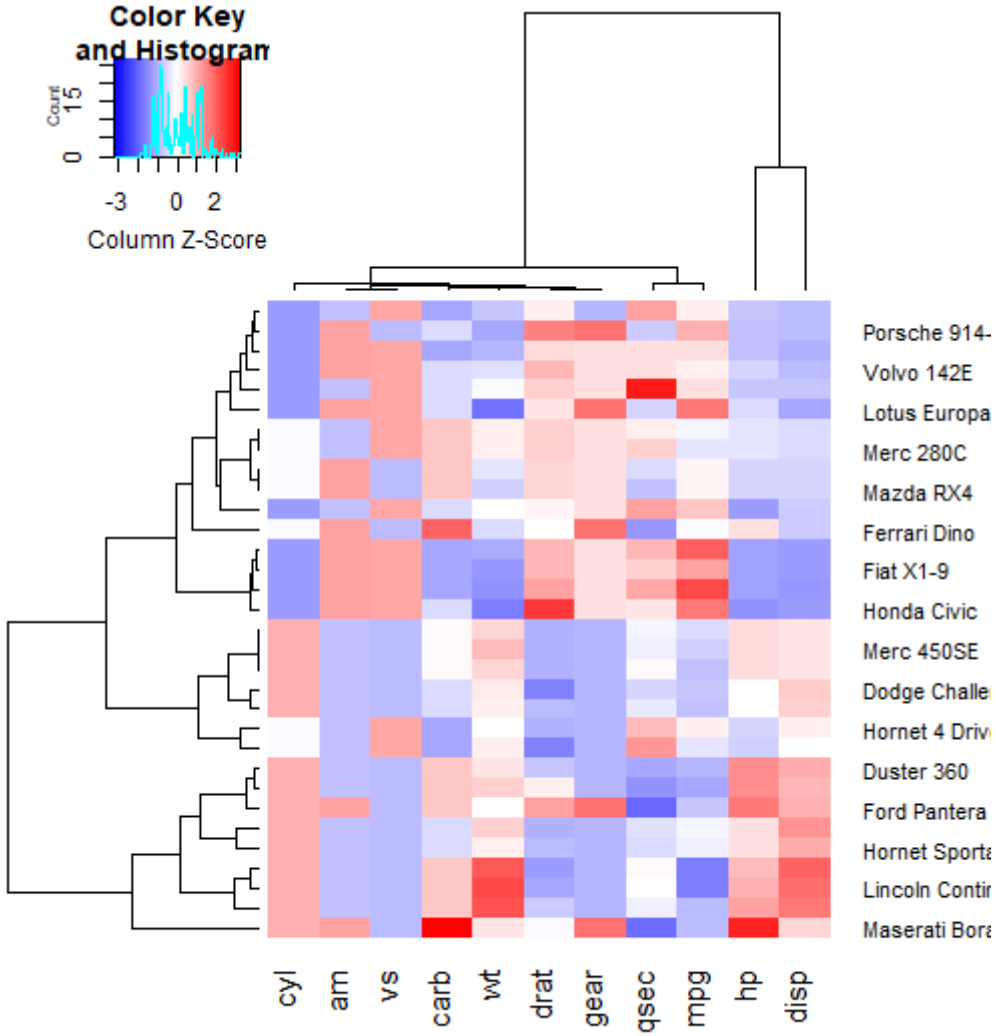
```
> library(gplots)
> heatmap.2(x, scale = "column", col = bluered(100))
> # Remove tracing
> heatmap.2(x, scale = "column", col = bluered(100),
+           trace = "none")
> # 3- pheatmap
> # install.packages("pheatmap")
> library(pheatmap)
> pheatmap(x, scale = "column")
> # cut after 4 rows
> pheatmap(x, scale = "column", cutree_rows = 4)
> # cut in cloumns
> pheatmap(x, scale = "column", cutree_cols = 2)
> #4- method
> library(ggplot2)
> # load data
> y <- iris
> library(reshape)
> y1 <- melt(iris)
Using Species as id variables
> ggplot(y1, aes(y1$Species, y1$variable, fill= y1$value))+
+   geom_tile()+
+   scale_fill_gradient(low = "red", high = "green")
Warning messages:
1: Use of `y1$Species` is discouraged. Use `Species` instead.
2: Use of `y1$variable` is discouraged. Use `variable` instead.
3: Use of `y1$value` is discouraged. Use `value` instead.
```

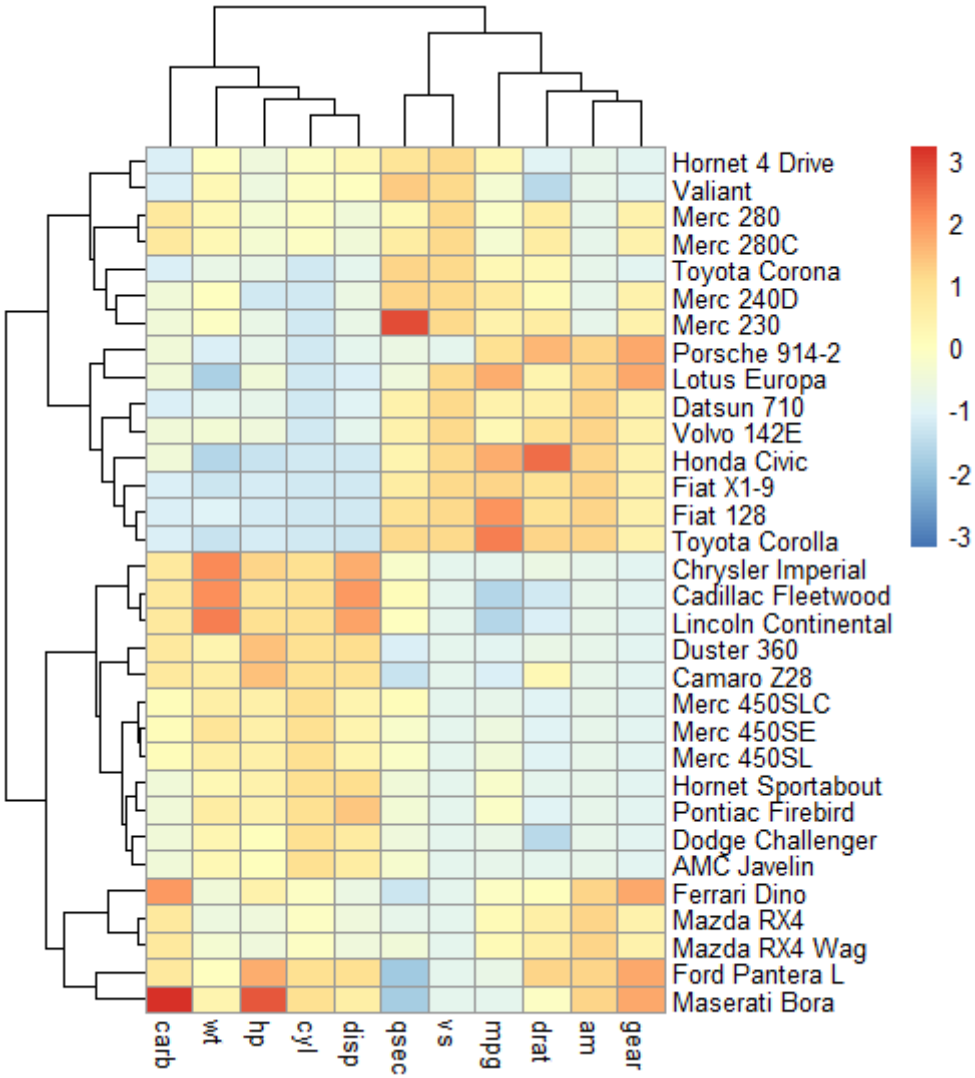

Porsche 914-2
Volvo 142E
Lotus Europa
Merc 280C
Mazda RX4
Ferrari Dino
Fiat X1-9
Honda Civic
Merc 450SE
Dodge Challeng
Hornet 4 Drive
Duster 360
Ford Pantera L
Hornet Sportab
Lincoln Contine
Maserati Bora
cyl
am
vs
carb
wt
drat
gear
qsec
mpg
hp
disp

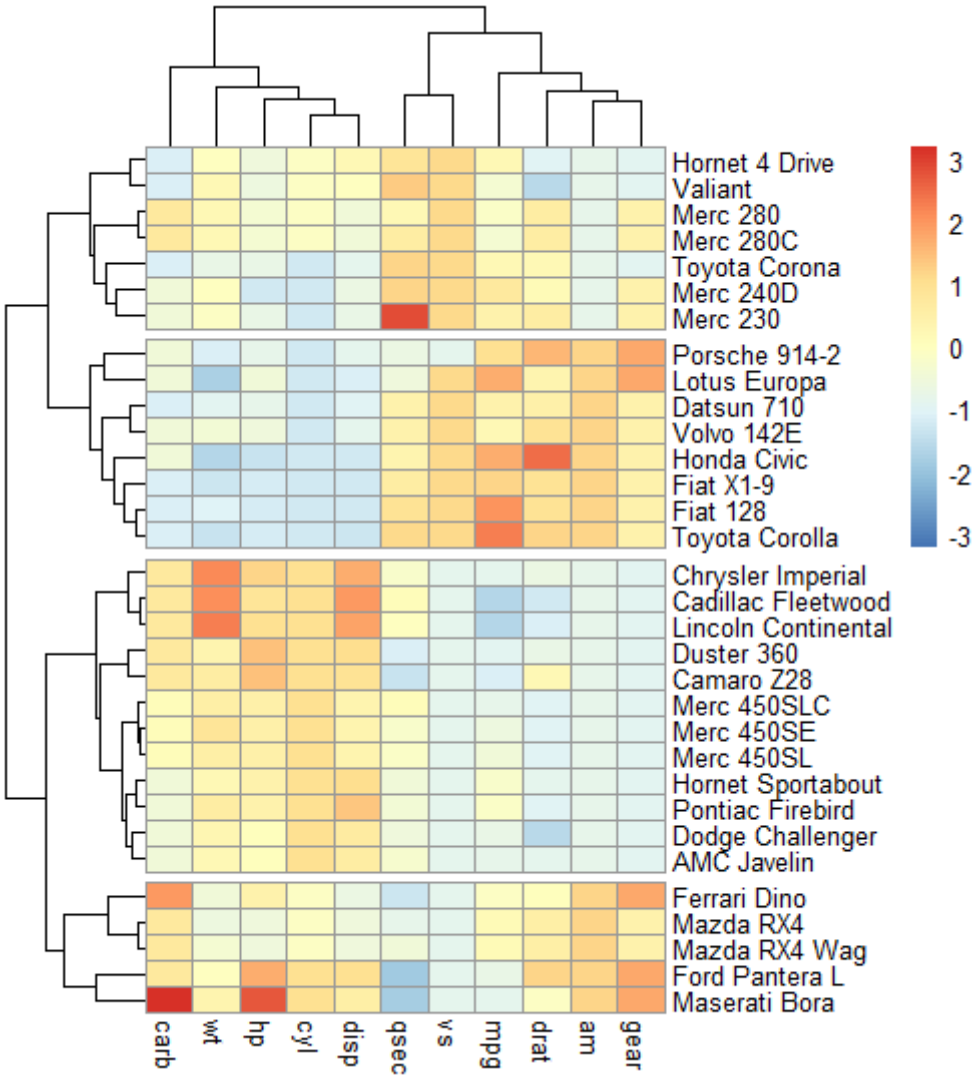

Porsche 914-2
Volvo 142E
Lotus Europa
Merc 280C
Mazda RX4
Ferrari Dino
Fiat X1-9
Honda Civic
Merc 450SE
Dodge Challeng
Hornet 4 Drive
Duster 360
Ford Pantera L
Hornet Sportab

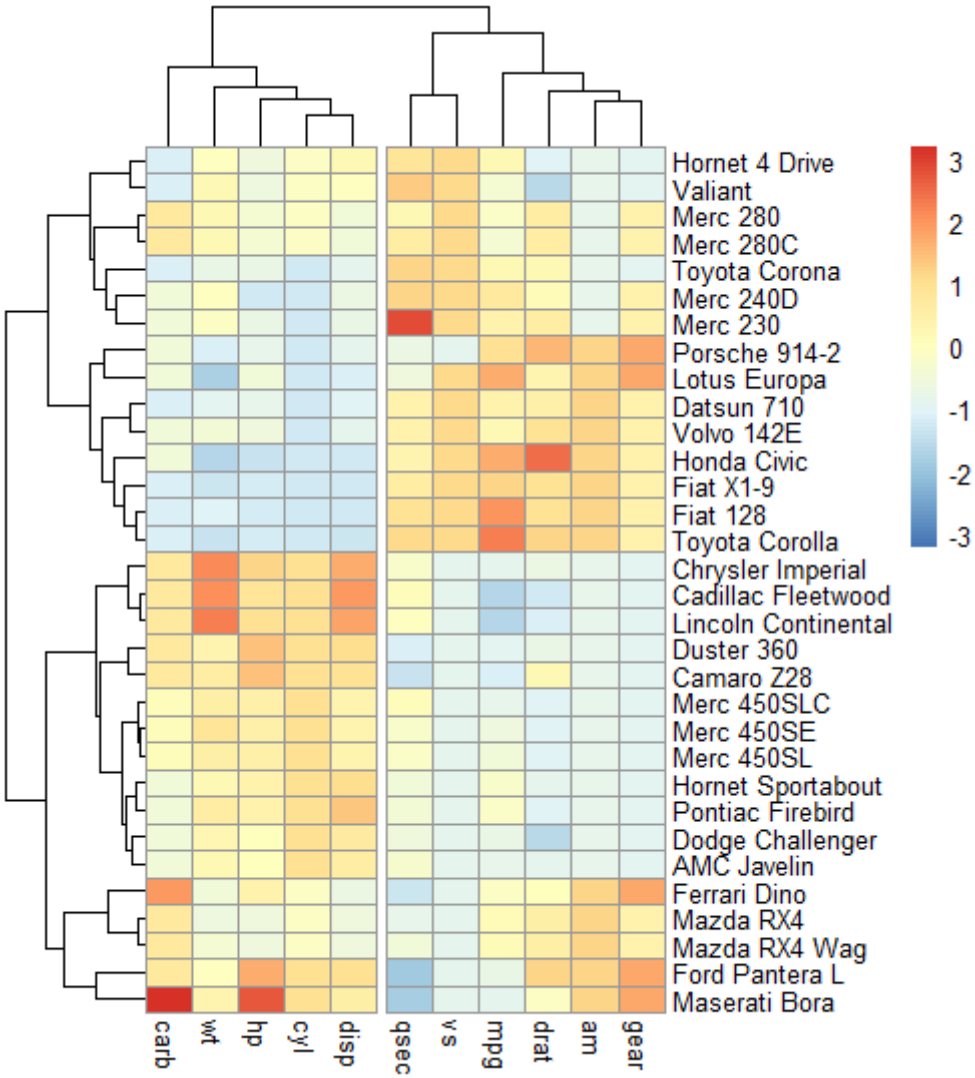

Lincoln Contine
Maserati Bora
cyl
am
vs
carb
wt
drat
gear
qsec
mpg
hp
disp

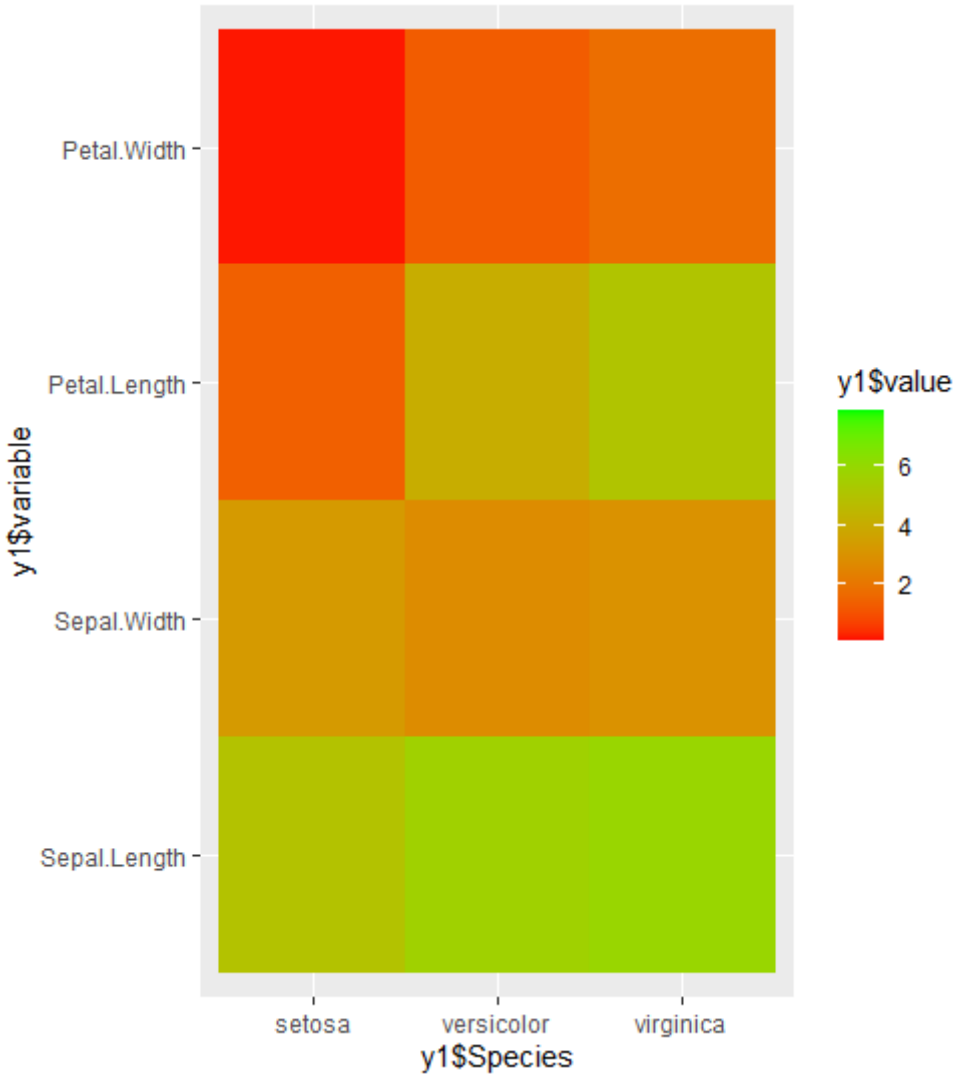

Color Key
and Histogram
Count
0
15
-3
0
2
Column Z-Score
Porsche 914-
Volvo 142E
Lotus Europa
Merc 280C
Mazda RX4
Ferrari Dino
Fiat X1-9
Honda Civic
Merc 450SE
Dodge Challe
Hornet 4 Driv
Duster 360
Ford Pantera
Hornet Sporta
Lincoln Contir
Maserati Bora
cyl
am
vs
carb
wt
drat
gear
qsec
mpg
hp
disp


Color Key
and Histogram
Count
15
0
-3
0
2
Column Z-Score
Porsche 914-
Volvo 142E
Lotus Europa
Merc 280C
Mazda RX4
Ferrari Dino
Fiat X1-9
Honda Civic
Merc 450SE
Dodge Challe
Hornet 4 Driv
Duster 360
Ford Pantera
Hornet Sporta
Lincoln Conti
Maserati Bora
cyl
am
vs
carb
wt
drat
gear
qsec
mpg
hp
disp


Hornet 4 Drive
Valiant
Merc 280
Merc 280C
Toyota Corona
Merc 240D
Merc 230
Porsche 914-2
Lotus Europa
Datsun 710
Volvo 142E
Honda Civic
Fiat X1-9
Fiat 128
Toyota Corolla
Chrysler Imperial
Cadillac Fleetwood
Lincoln Continental
Duster 360
Camaro Z28
Merc 450SLC
Merc 450SE
Merc 450SL
Hornet Sportabout
Pontiac Firebird
Dodge Challenger
AMC Javelin
Ferrari Dino
Mazda RX4
Mazda RX4 Wag
Ford Pantera L
Maserati Bora
carb
wt
hp
cyl
disp
qsec
vs
mpg
drat
am
gear
3
2
1
0
-1
-2
-3


Hornet 4 Drive
Valiant
Merc 280
Merc 280C
Toyota Corona
Merc 240D
Merc 230
Porsche 914-2
Lotus Europa
Datsun 710
Volvo 142E
Honda Civic
Fiat X1-9
Fiat 128
Toyota Corolla
Chrysler Imperial
Cadillac Fleetwood
Lincoln Continental
Duster 360
Camaro Z28
Merc 450SLC
Merc 450SE
Merc 450SL
Hornet Sportabout
Pontiac Firebird
Dodge Challenger
AMC Javelin
Ferrari Dino
Mazda RX4
Mazda RX4 Wag
Ford Pantera L
Maserati Bora
carb
wt
hp
cyl
disp
qsec
vs
mpg
drat
am
gear
3
2
1
0
-1
-2
-3


Hornet 4 Drive
Valiant
Merc 280
Merc 280C
Toyota Corona
Merc 240D
Merc 230
Porsche 914-2
Lotus Europa
Datsun 710
Volvo 142E
Honda Civic
Fiat X1-9
Fiat 128
Toyota Corolla
Chrysler Imperial
Cadillac Fleetwood
Lincoln Continental
Duster 360
Camaro Z28
Merc 450SLC
Merc 450SE
Merc 450SL
Hornet Sportabout
Pontiac Firebird
Dodge Challenger
AMC Javelin
Ferrari Dino
Mazda RX4
Mazda RX4 Wag
Ford Pantera L
Maserati Bora
carb
wt
hp
cyl
disp
qsec
vs
mpg
drat
am
gear
3
2
1
0
-1
-2
-3



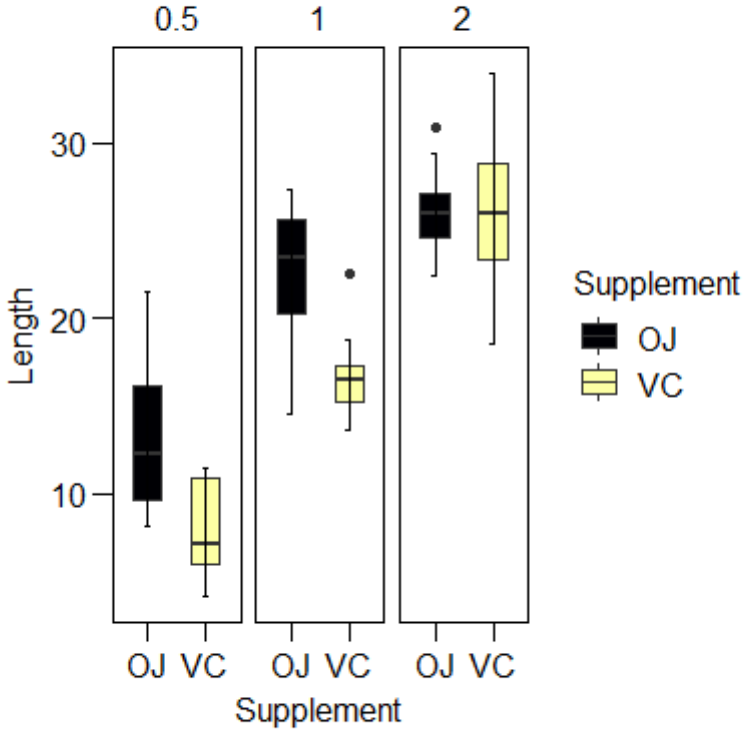
Class 16 Adding p_values in plots

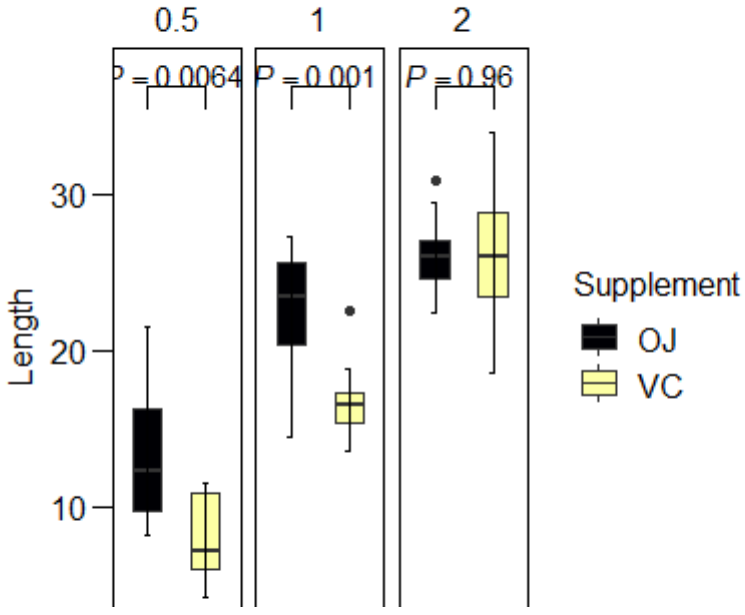
```
# install packages
install.packages("ggplot2")
install.packages("ggpval")

# load packages
library(ggpval)
library(ggplot2)
library(ggthemes)

# Draw plot
data("ToothGrowth")

p <- ggplot(ToothGrowth)+
  aes(x= supp, y= len, fill= supp)+
  geom_boxplot(shape = "circle", width=0.5)+
  stat_boxplot(geom = "errorbar", width=0.1)+
```

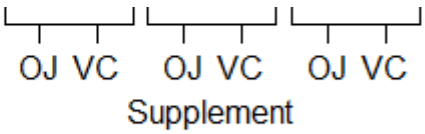
```
  scale_fill_viridis_d(option = "inferno", direction = 1)+
  labs(x= "Supplement", y= "Length", fill= "Supplement")+
  ggthemes::theme_par()+
  facet_wrap(vars(dose));p

# Add p_value
add_pval(p, pairs = list(c(1,2)),
         test = "t.test")

# for more information
help("add_pval
```

output

```
> # load packages
> library(ggpval)
> library(ggplot2)
> library(ggthemes)
> # Draw plot
> data("ToothGrowth")
> p <- ggplot(ToothGrowth)+
+   aes(x= supp, y= len, fill= supp)+
+   geom_boxplot(shape = "circle", width=0.5)+
+   stat_boxplot(geom = "errorbar", width=0.1)+
+   scale_fill_viridis_d(option = "inferno", direction = 1)+
+   labs(x= "Supplement", y= "Length", fill= "Supplement")+
+   ggthemes::theme_par()+
+   facet_wrap(vars(dose));p
> # Add p_value
> add_pval(p, pairs = list(c(1,2)),
+          test = "t.test")
```


0.5
1
2
30
20
10
Length
OJ VC
OJ VC
OJ VC
Supplement
Supplement
OJ
VC


0.5
1
2
P = 0.0064
P = 0.001
P = 0.96
30
20
10
Length
Supplement
OJ
VC



Class 17 18-important Resources to Learn R

1- Rstudio Website: https://www.rstudio.com/resources/books/
2- https://intro2r.com/
3- https://bookdown.org/ndphillips/YaRrr/
4- https://rcompanion.org/handbook/D_03....
5- https://rafalab.github.io/dsbook/
6- https://r4ds.had.co.nz/
7- https://adv-r.hadley.nz/
8- https://www.danielnettle.org.uk/wp-co...
9- https://stackoverflow.com/questions/1...
10- https://ourcodingclub.github.io/tutor...
11- https://ourcodingclub.github.io/cours...
12- https://www.tidytuesday.com/
13- https://twitter.com/search?q=%23rstat...
14- https://twitter.com/hashtag/TidyTuesd...
15- https://github.com/AammarTufail/R_wit...
16- http://www.sthda.com/english/wiki/ggp...
17- https://www.tidyverse.org/
18- https://bookdown.org/yihui/rmarkdown/

Class 18 Barplot with one-way ANOVA and TukeyHSD test lettering

```
# publication ready graph
install.packages("ggthemes")
install.packages("multcompView")
install.packages("dplyr")

# load libraries
library(ggplot2)
library(ggthemes)
library(multcompView)
library(dplyr)
```

```
## 1- load or import your data
data("chickwts")
tibble(chickwts)

# 2- Calculate the "means" of you of treatment group and the SD
#to show the error bars as functions
# feed our treatment in this example

mean_data <- group_by(chickwts, feed) %>%
  summarise(weight_mean= mean(weight), sd = sd(weight))%>%
  arrange(desc(weight_mean))#to arrange in descending order
tibble(mean_data)

# 3- This step involves performing analysis of variance "ANOVA"
# using the built in function ***aov()*** .

anova <- aov(weight~feed, data = chickwts)
summary(anova)

# 4- If the ANOVA is significantly different then,
# we will draw the multiple mean comparison
tukey <- TukeyHSD(anova)
tukey

# 5- Draw the Multiple comparison letters using "multcomp"
# R packages are as follows:
group_letter <- multcompLetters4(anova, tukey)
group_letter

# Extracting group letters
group_letter <- as.data.frame.list(group_letter$feed)
group_letter

# Adding to the mean data
mean_data$group_letters <- group_letter$Letters
tibble(mean_data)

# 6- Draw the publication ready Barplot* in ggplot2
p <- ggplot(mean_data, aes(x = feed, y = weight_mean))+
  # Bar plot
   geom_bar(stat = "identity", aes(fill = feed), show.legend = FALSE, width = 0.6)+
  geom_errorbar(# this argument is putting SD as error bars
    aes(ymin = weight_mean-sd, ymax = weight_mean+sd), width = 0.1)+
  # Adding letters
  geom_text(aes(label = group_letters, y = weight_mean+sd), vjust = -0.4)+
  scale_fill_brewer(palette = "BrBG", direction = 1)+ #theme setting
  labs(# This will add labels
    x = "Feed Type",
    y = "Chicken Weight(g)",
```

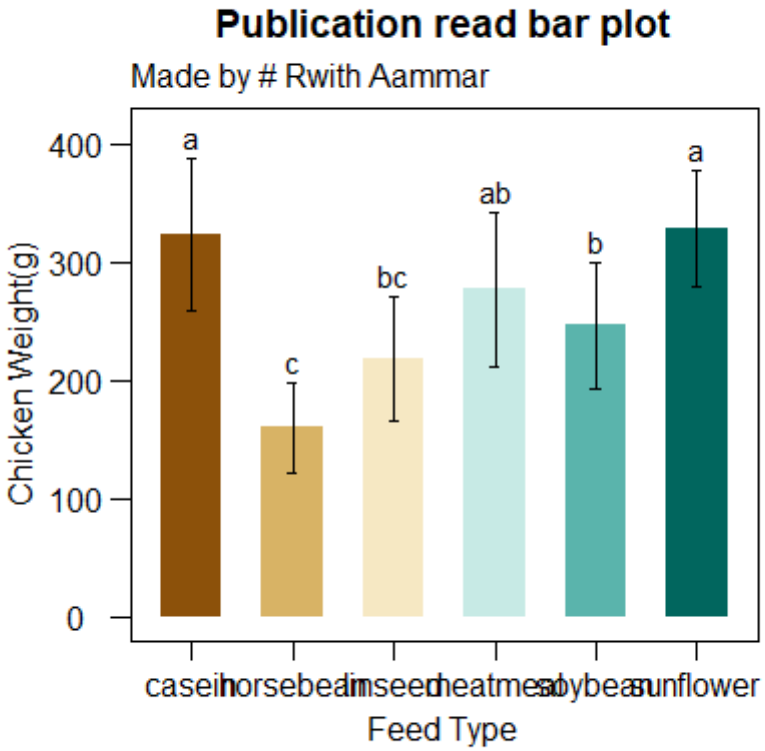
```
    title = "Publication read bar plot",
    subtitle = "Made by # Rwith Aammar",
    fill = "Feed Type")+
  ylim(0,410)+ # Change your y axis limits based on the letters
  ggthemes::theme_par();p

# 7- Saving up-to 4k bar plot in R

tiff("Bar plot.tiff", units = "in", width = 10, height = 6,
     res = 300, compression = "lzw")
p
dev.off()
```

output

```
> library(ggplot2)
> library(ggthemes)
> library(multcompView)
> library(dplyr)
> ## 1- load or import your data
> data("chickwts")
> tibble(chickwts)
# A tibble: 71 x 2
   weight feed
    <dbl> <fct>
 1    179 horsebean
 2    160 horsebean
 3    136 horsebean
 4    227 horsebean
 5    217 horsebean
 6    168 horsebean
 7    108 horsebean
 8    124 horsebean
 9    143 horsebean
10    140 horsebean
# ... with 61 more rows
> mean_data <- group_by(chickwts, feed) %>%
+   summarise(weight_mean= mean(weight), sd = sd(weight))%>%
+   arrange(desc(weight_mean))#to arrange in descending order
> mean_data <- group_by(chickwts, feed) %>%
+   summarise(weight_mean= mean(weight), sd = sd(weight))%>%
+   arrange(desc(weight_mean))#to arrange in descending order
> mean_data <- group_by(chickwts, feed) %>%
+   summarise(weight_mean= mean(weight), sd = sd(weight))%>%
+   arrange(desc(weight_mean))#to arrange in descending order
> tibble(mean_data)
# A tibble: 6 x 3
  feed      weight_mean    sd
```

```
  <fct>           <dbl> <dbl>
1 sunflower        329.  48.8
2 casein           324.  64.4
3 meatmeal         277.  64.9
4 soybean          246.  54.1
5 linseed          219.  52.2
6 horsebean        160.  38.6
> anova <- aov(weight~feed, data = chickwts)
> summary(anova)
            Df Sum Sq Mean Sq F value   Pr(>F)
feed         5 231129   46226   15.37 5.94e-10 ***
Residuals   65 195556    3009
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> # 4- If the ANOVA is significantly different then,
> # we will draw the multiple mean comparison
> tukey <- TukeyHSD(anova)
> tukey
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = weight ~ feed, data = chickwts)

$feed
                           diff         lwr       upr     p adj
horsebean-casein    -163.383333 -232.346876 -94.41979 0.0000000
linseed-casein      -104.833333 -170.587491 -39.07918 0.0002100
meatmeal-casein      -46.674242 -113.906207  20.55772 0.3324584
soybean-casein       -77.154762 -140.517054 -13.79247 0.0083653
sunflower-casein       5.333333  -60.420825  71.08749 0.9998902
linseed-horsebean     58.550000  -10.413543 127.51354 0.1413329
meatmeal-horsebean   116.709091   46.335105 187.08308 0.0001062
soybean-horsebean     86.228571   19.541684 152.91546 0.0042167
sunflower-horsebean  168.716667   99.753124 237.68021 0.0000000
meatmeal-linseed      58.159091   -9.072873 125.39106 0.1276965
soybean-linseed       27.678571  -35.683721  91.04086 0.7932853
sunflower-linseed    110.166667   44.412509 175.92082 0.0000884
soybean-meatmeal     -30.480519  -95.375109  34.41407 0.7391356
sunflower-meatmeal    52.007576  -15.224388 119.23954 0.2206962
sunflower-soybean     82.488095   19.125803 145.85039 0.0038845

> # 5- Draw the Multiple comparison letters using "multcomp"
> # R packages are as follows:
> group_letter <- multcompLetters4(anova, tukey)
> group_letter
$feed
sunflower    casein  meatmeal   soybean   linseed horsebean
      "a"       "a"      "ab"       "b"      "bc"       "c"

> # Extracting group letters
```

```
> group_letter <- as.data.frame.list(group_letter$feed)
> group_letter
          Letters monospacedLetters LetterMatrix.a LetterMatrix.b LetterMatrix.c
sunflower       a               a             TRUE          FALSE          FALSE
casein          a               a             TRUE          FALSE          FALSE
meatmeal       ab              ab             TRUE           TRUE          FALSE
soybean         b               b            FALSE           TRUE          FALSE
linseed        bc              bc            FALSE           TRUE           TRUE
horsebean       c               c            FALSE          FALSE           TRUE
> # Adding to the mean data
> mean_data$group_letters <- group_letter$Letters
> tibble(mean_data)
# A tibble: 6 x 4
  feed      weight_mean    sd group_letters
  <fct>           <dbl> <dbl> <chr>
1 sunflower        329.  48.8 a
2 casein           324.  64.4 a
3 meatmeal         277.  64.9 ab
4 soybean          246.  54.1 b
5 linseed          219.  52.2 bc
6 horsebean        160.  38.6 c
> # 6- Draw the publication ready Barplot* in ggplot2
> p <- ggplot(mean_data, aes(x = feed, y = weight_mean))+
+   # Bar plot
+    geom_bar(stat = "identity", aes(fill = feed), show.legend = FALSE, width =
0.6)+
+   geom_errorbar(# this argument is putting SD as error bars
+     aes(ymin = weight_mean-sd, ymax = weight_mean+sd), width = 0.1)+
+   # Adding letters
+   geom_text(aes(label = group_letters, y = weight_mean+sd), vjust = -0.4)+
+   scale_fill_brewer(palette = "BrBG", direction = 1)+ #theme setting
+   labs(# This will add labels
+     x = "Feed Type",
+     y = "Chicken Weight(g)",
+     title = "Publication read bar plot",
+     subtitle = "Made by # Rwith Aammar",
+     fill = "Feed Type")+
+   ylim(0,410)+ # Change your y axis limits based on the letters
+   ggthemes::theme_par();p
> tiff("Bar plot.tiff", units = "in", width = 10, height = 6,
+      res = 300, compression = "lzw")
> p
> dev.off()
RStudioGD
        2
```



class 20 Correlation in R

```
# Correlation
## Insatll packages
install.packages("psych")
install.packages("corrplot")
install.packages("RColorBrewer")

# read packages
library(psych)
library(corrplot)
library(RColorBrewer)

# Psych packages
data("iris")
View(iris)

# Naming
```

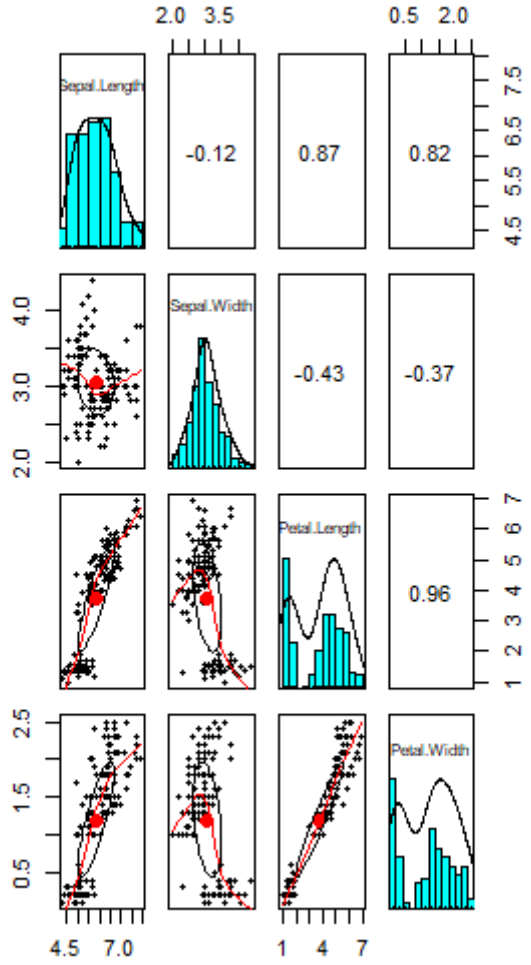
```
x <- corr.test(iris[,-5])

# for values or correlation matrix
x

# plot
pairs.panels(iris[-5])

# Corr plot package
# we are using "iris data again"
data("iris")

# Correlation
m <- cor(iris[,-5])
# plot
corrplot(m)
# 2nd corr plot

corrplot(m, type = "upper")
corrplot(m, type = "lower")
# commands to plot
corrplot(m, type = "upper", order = "hclust", method = "pie")
# different methods
corrplot(m, type = "upper", order = "hclust", method = "square")
# color
?RColorBrewer()
corrplot(m, type = "upper", order = "hclust", method = "pie",
         col = brewer.pal(n = 8, name = "RdYlBu"))

#  mixed Corr plot
corrplot.mixed(m)
# Make upper and lower in differently
corrplot.mixed(m, lower.col = "black", number.cex=0.7)
# one more differnt plot

corrplot.mixed(m, lower = "number", upper = "pie", t1.col="red")
# Change method of corr
?cor

m1 <- cor(iris[,-5], method = "spearman")
# plot
corrplot(m1)
# 2nd corr plot

corrplot(m1, type = "upper")
corrplot(m1, type = "lower")
# commands to plot
corrplot(m1, type = "upper", order = "hclust", method = "pie")
# different methods
corrplot(m1, type = "upper", order = "hclust", method = "square")
```

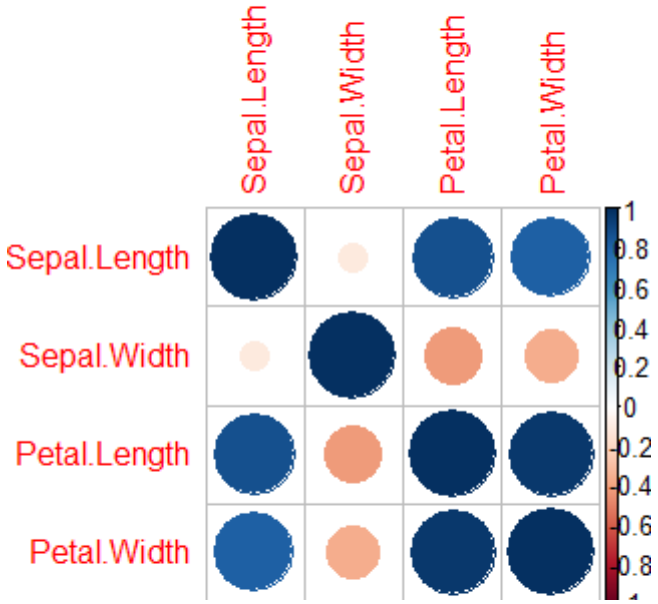
```
# color
?RColorBrewer()
corrplot(m1, type = "upper", order = "hclust", method = "pie",
         col = brewer.pal(n = 8, name = "RdYlBu"))
```

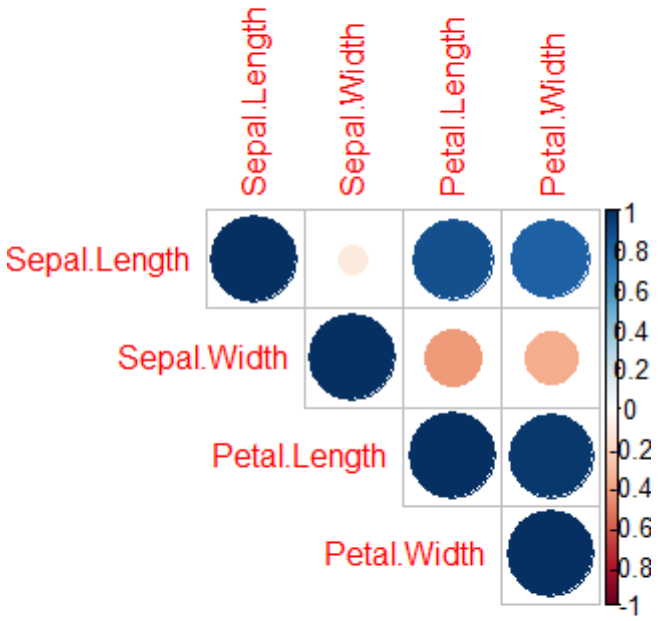
output

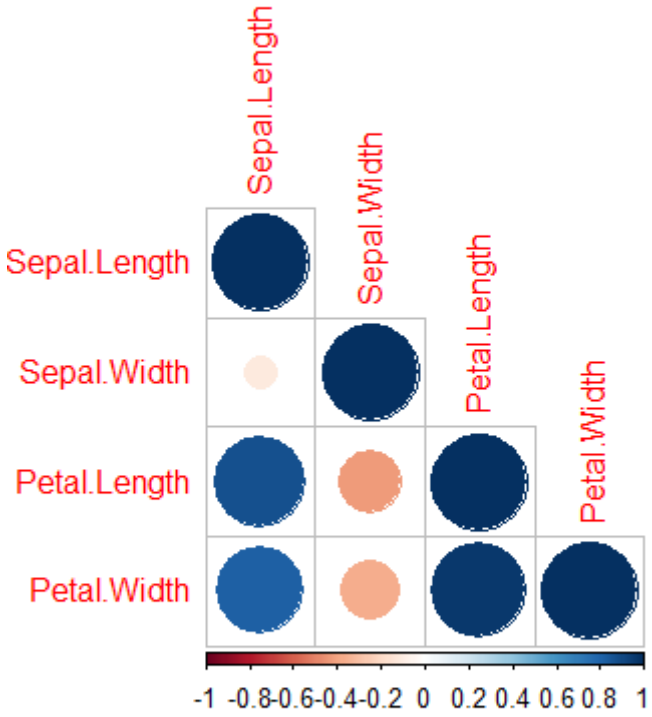
```
> # read packages
> library(psych)
> library(corrplot)
> library(RColorBrewer)
> # Psych packages
> data("iris")
> View(iris)
> # Naming
> x <- corr.test(iris[,-5])
> # for values or correlation matrix
> x
Call:corr.test(x = iris[, -5])
Correlation matrix
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length         1.00       -0.12         0.87        0.82
Sepal.Width         -0.12        1.00        -0.43       -0.37
Petal.Length         0.87       -0.43         1.00        0.96
Petal.Width          0.82       -0.37         0.96        1.00
Sample Size
[1] 150
Probability values (Entries above the diagonal are adjusted for multiple tests.)
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length         0.00        0.15            0           0
Sepal.Width          0.15        0.00            0           0
Petal.Length         0.00        0.00            0           0
Petal.Width          0.00        0.00            0           0

 To see confidence intervals of the correlations, print with the short=FALSE option
> # plot
> pairs.panels(iris[-5])
> # Corr plot package
> # we are using "iris data again"
> data("iris")
> # Correlation
> m <- cor(iris[,-5])
> # plot
> corrplot(m)
> corrplot(m, type = "upper")
> corrplot(m, type = "lower")
> # commands to plot
> corrplot(m, type = "upper", order = "hclust", method = "pie")
```

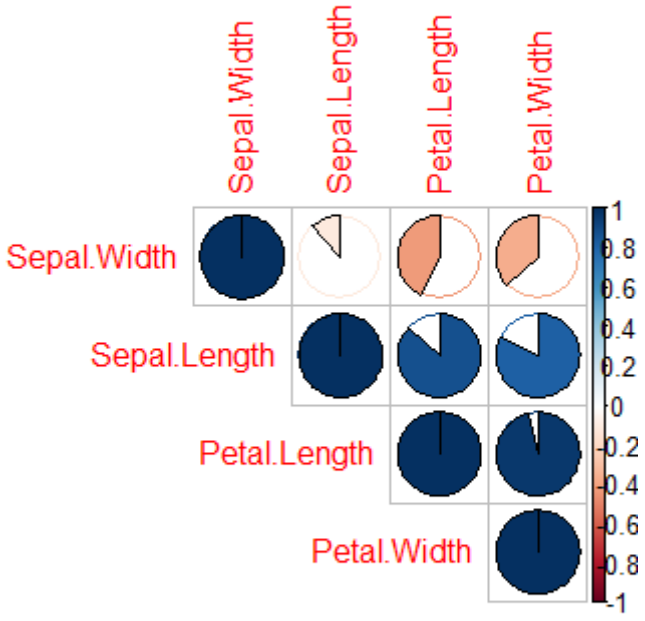
```
> # different methods
> corrplot(m, type = "upper", order = "hclust", method = "square")
> corrplot(m, type = "upper", order = "hclust", method = "pie",
+          col = brewer.pal(n = 8, name = "RdYlBu"))
> #  mixed Corr plot
> corrplot.mixed(m)
> # Make upper and lower in differently
> corrplot.mixed(m, lower.col = "black", number.cex=0.7)
> corrplot.mixed(m, lower = "number", upper = "pie", t1.col="red")
Warning messages:
1: In text.default(pos.ylabel[, 1] + 0.5, pos.ylabel[, 2], newcolnames[1:min(n,  :
  "t1.col" is not a graphical parameter
2: In title(title, ...) : "t1.col" is not a graphical parameter
3: In title(title, ...) : "t1.col" is not a graphical parameter
> m1 <- cor(iris[,-5], method = "spearman")
Warning messages:
1: In doTryCatch(return(expr), name, parentenv, handler) :
  "t1.col" is not a graphical parameter
2: In doTryCatch(return(expr), name, parentenv, handler) :
  "t1.col" is not a graphical parameter
3: In doTryCatch(return(expr), name, parentenv, handler) :
  "t1.col" is not a graphical parameter
4: In doTryCatch(return(expr), name, parentenv, handler) :
  "t1.col" is not a graphical parameter
5: In doTryCatch(return(expr), name, parentenv, handler) :
  "t1.col" is not a graphical parameter
6: In doTryCatch(return(expr), name, parentenv, handler) :
  "t1.col" is not a graphical parameter
> # plot
> corrplot(m1)
> corrplot(m1, type = "upper")
> corrplot(m1, type = "lower")
> # commands to plot
> corrplot(m1, type = "upper", order = "hclust", method = "pie")
> # different methods
> corrplot(m1, type = "upper", order = "hclust", method = "square")
> corrplot(m1, type = "upper", order = "hclust", method = "pie",
+          col = brewer.pal(n = 8, name = "RdYlBu"))
```

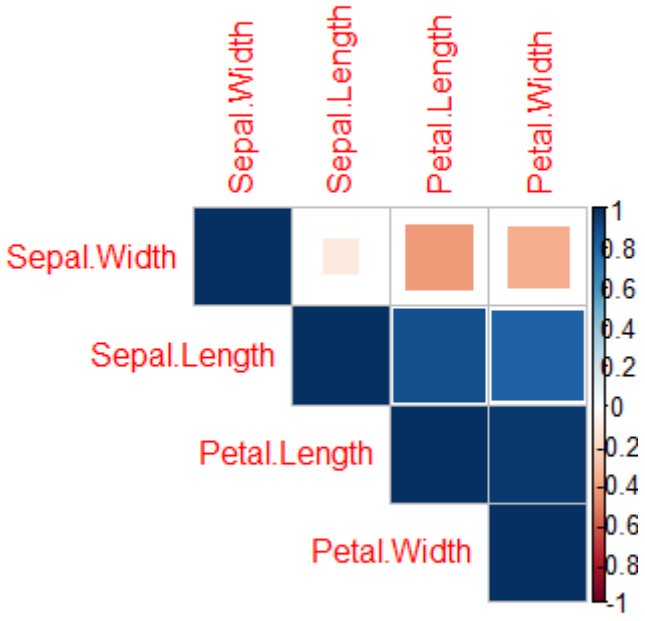

Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
-0.12
0.87
0.82
-0.43
-0.37
0.96

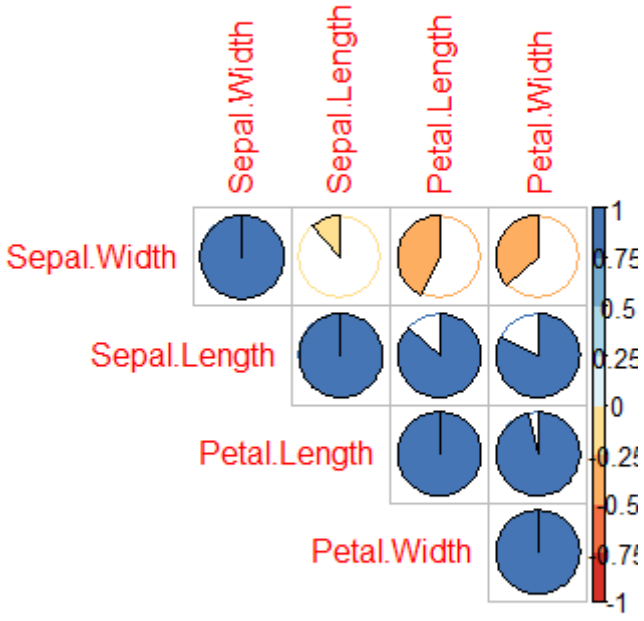

Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8

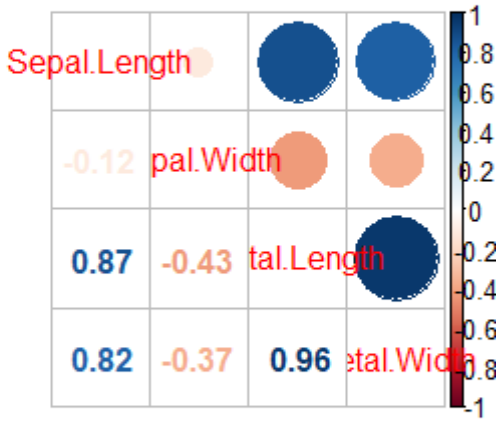

Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

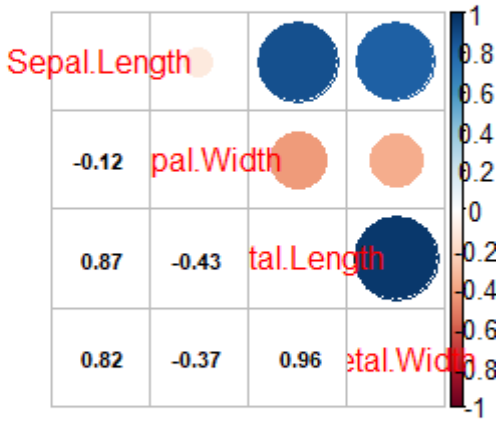

Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1

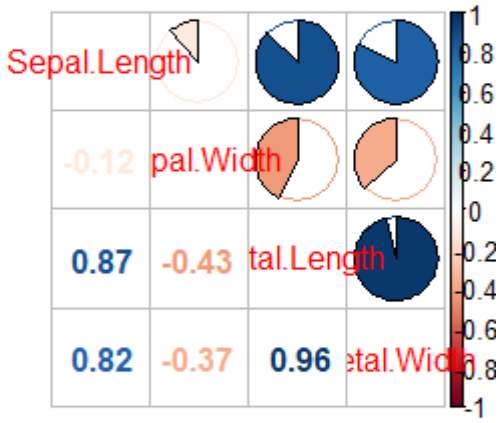

Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

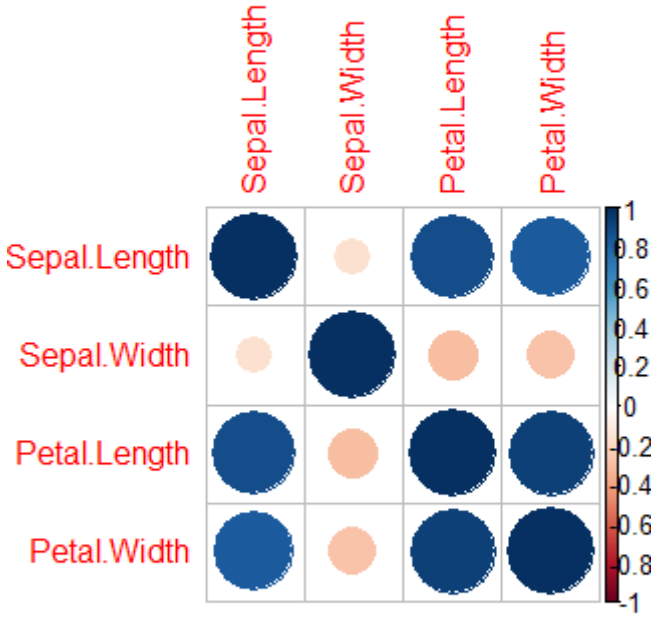

Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

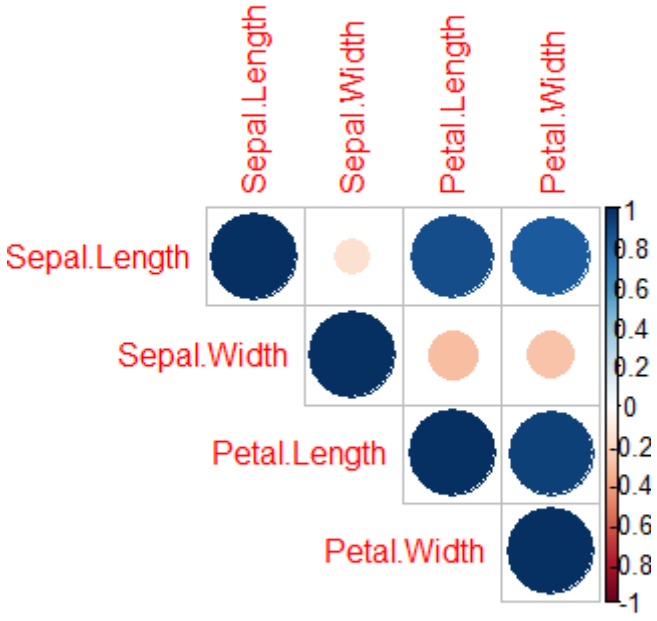

Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1

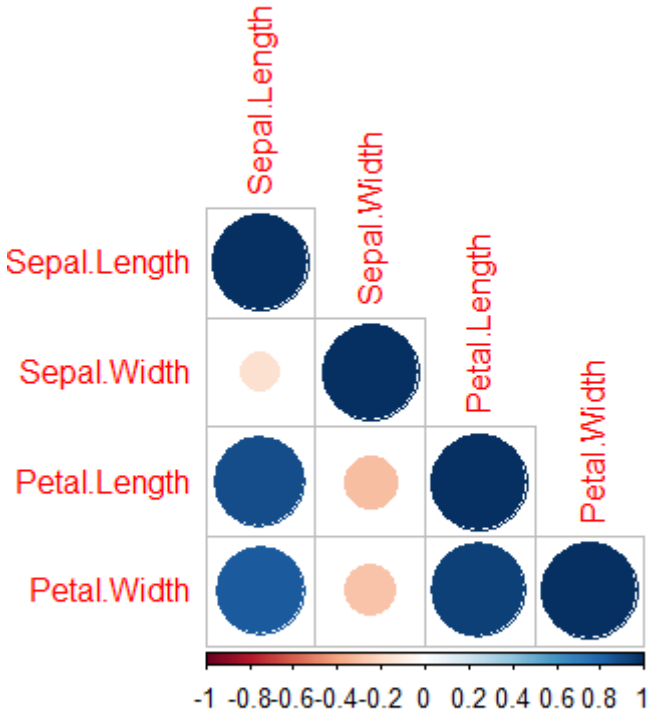

Sepal.Length
-0.12
pal.Width
0.87
-0.43
tal.Length
0.82
-0.37
0.96
etal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

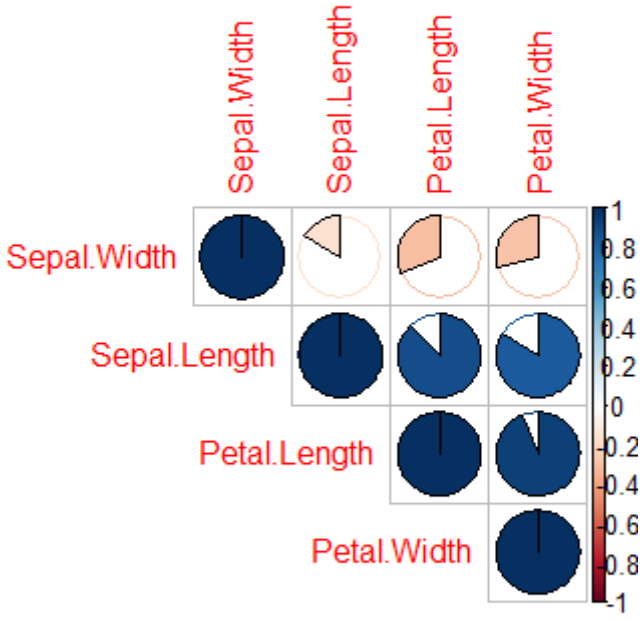

Sepal.Length
-0.12
pal.Width
0.87
-0.43
tal.Length
0.82
-0.37
0.96
etal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

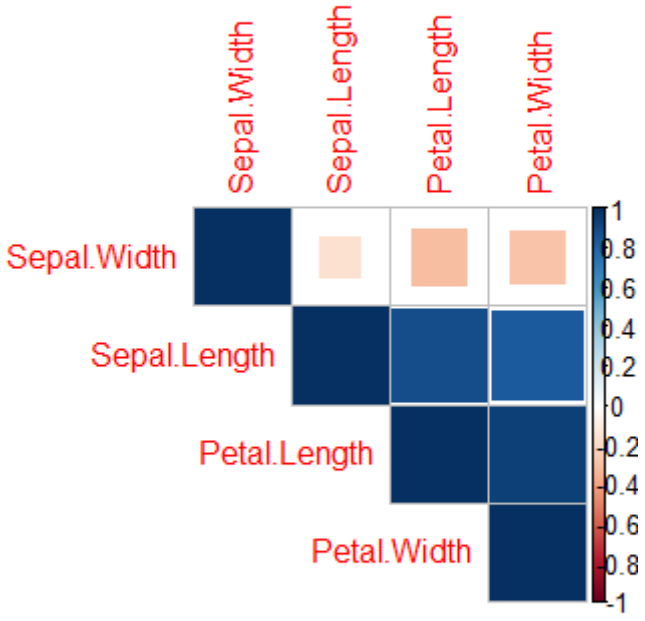

Sepal.Length
-0.12
pal.Width
0.87
-0.43
tal.Length
0.82
-0.37
0.96
etal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

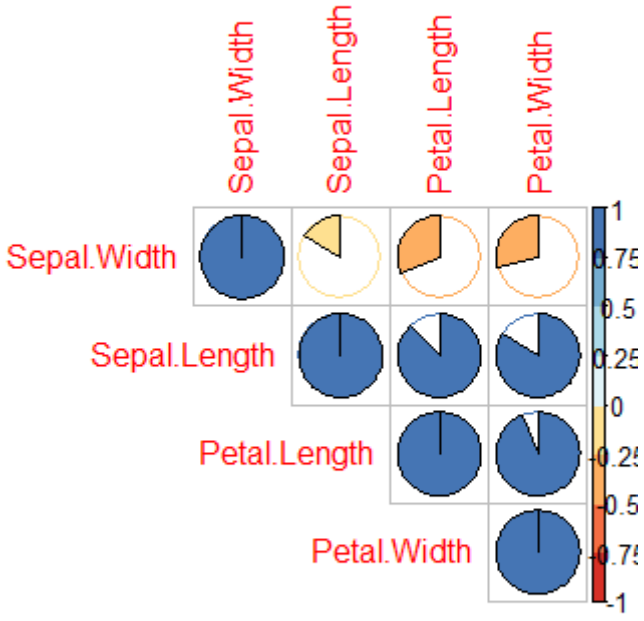

Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1


Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
Sepal.Width
Sepal.Length
Petal.Length
Petal.Width
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1



Data Wrangling

```
#Recommended book: Introduction to R, data analysis and prediction algorithm with R

# Why data wrangling is so important?
# 1. if your data is not set then no matter how many visualization graph you made
# graph structure will be change, std will be so big you can't calculate difference.
# 2. if you are using using your data as it is mean without transformation the
statistic
# analysis will be not so good.
# 3. most impotently you will not know your data structure.

# what is tibble?
# basically its a kind of table in stat we called it tibble, tuple etc. You will
# data structure or DF in console window of R.

# Data wrangling and data transformation
```

```
install.packages("tidyverse")
install.packages("tidyr")
library(tidyverse)
library(tidyr)

#bultin datasets in R
data()

#Data types
##numeric
num <- 2.2
class(num) #class function use to see data type

##character
chr <- "RwithAammar"
class(chr)

##Logical
logi <- TRUE
class(logi)

#how to confirm a class/type of variable:
num
is.numeric(num)
is.character(num)

chr
is.character(chr)

logi
is.logical(logi)

#how to transform a data type
num1 <- "3.2"
class (num1)

num1 <- as.numeric(num1) #use this to convert data type
class(num1)

num2 <- c(1,2, 3,4,5) # vector is who increase continuously
num2
class(num2)

num2 <- as.logical(num2)
class (num2)

num2 <- c(1,2, 3,4,5,0) # when you use logical operator only value of 0 will be
false
num2
class(num2)
```

```
num2 <- as.logical(num2)
class (num2)
num2

num2 <- as.character(num2)
class(num2)
print(num2) #or num2 only are same things

#data types are more important than data itself....
#data structure is more important than data significance....

###step-1- set your home directory (Ctrl+sift+H)
###Step-2- Import your data in R (readx1, readcsv)
###step-3- Data Structure and Types
###step-4- Data Transformation and tidy tips and tricks
###step-5-

##step-3- Data Structure and Types

###Data types
###numeric
num <- 2.2
class(num) #class function use to see data type

###character
chr <- "RwithAammar"
class(chr)

###Logical
logi <- TRUE
class(logi)

###how to confirm a class/type of variable:
num
is.numeric(num)
is.character(num)

chr
is.character(chr)

logi
is.logical(logi)

###how to transform a data type
num1 <- "3.2"
class (num1)

num1 <- as.numeric(num1) #use this to convert data type
class(num1)
```

```
num2 <- c(1,2, 3,4,5) # vector is who increase continuously
num2
class(num2)

num2 <- as.logical(num2)
class (num2)

num2 <- c(1,2, 3,4,5,0) # when you use logical operator only value of 0 will be
false
num2
class(num2)

num2 <- as.logical(num2)
class (num2)
num2

num2 <- as.character(num2)
class(num2)
print(num2) #or num2 only are same things

#Data struture
data()
data("diamonds")
x <- diamonds
head (x)
str(x)
view(x)
class(x$carat)
class(x$price) # class int because its whole number
class(x$x)

# difference between tibble and data frame
# lets take example of diamond data if we melt down "carat, cut, color, clarity"
# into one column and name it as diamond characteristics then we created a data
frame.
# in the form of table it we can call it tibble

class(x["carat"])
as.data.frame (x) # not use moslty to see data
as_tibble(x) # tibble adjust its column according to how widely your console window
is open'
glimpse(x) # this function give you overview or bird eye view of your data


###step-4- Data Transformation and tidy tips and tricks

#Data Transformation
data()
data("diamonds")
```

```
x <- diamonds
head (x)
str(x)
view(x)
class(x$carat)
class(x$price) # class int because its whole number
class(x$x)

# difference between tibble and data frame
# lets take example of diamond data if we melt down "carat, cut, color, clarity"
# into one column and name it as diamond characteristics then we created a data
frame.
# in the form of table it we can call it tibble

class(x["carat"])
as.data.frame (x) # not use moslty to see data
as_tibble(x) # tibble adjust its column according to how widely your console window
is open'
glimpse(x) # this function give you overview or bird eye view of your data

#convert into right data type?
x$carat <- as.numeric(x$carat, x$depth)
class(x$table)
glimpse(x)
str(x)

#vector
vecl <- c(1,2,3,4,6,9,0,-1,0.5)
class (vecl)
#Factor
vec2  <- c("blue", "red", "purple", 'green') # can be use a in color command to give
color to different variable on x axis sequence wise
class(vec2)
vec2_fac <- as.factor(vec2)
class(vec2_fac )

#tidy means to collect data and make it usable for statistics in R
x <- CO2
glimpse(x)
as_tibble(x)
as.data.frame (x)
x1 <- chickwts
as_tibble(x1)

# #manipulating data
install.packages ("dslabs")
library (dslabs)
x <- diamonds

#add a column
```

```
as_tibble(x)
x <- mutate(x, a = x+y)
as_tibble(x)
x2<- mutate (x, sd=  sd(y+z))
as_tibble(x2)


# #subsetting
as_tibble(x)
x_al <- filter(x, a == 7.93)
as_tibble(x_al)

X_a2 <- filter(x, cut=="Ideal")#subsetting a character variable
as_tibble(X_a2)

#subsetting using select function
as_tibble(x)
x1 <- select(x, cut, color, x)
as_tibble(x1)

#task for attendees
x <- murders
as_tibble(x)
#data= murders
#criterial= pop > 5^6
#region = south
#Plots for both send to whatsapp inbo

x2 <- filter(x, region=="West", population>5000000)
as_tibble(x2)

plot(x2)


#Less code for more output
install.packages("magrittr") # package installations are only needed the first time
you use it
install.packages("dplyr")    # alternative installation of the %>%
library(magrittr) # needs to be run every time you start R and want to use %>%
library(dplyr)    # alternatively, this also loads %>%
#dplyr pipe X- ctrl+shift+m
# object
## murders
# # filter
## select
## mutate

x <- murders %>%
  filter(region=="West") %>%
  select(region, population) %>%
```

```
  mutate(population_2=population+2)
x
```

output

```
> library(tidyverse)
> library(tidyr)
> #Data types
> ##numeric
> num <- 2.2
> class(num) #class function use to see data type
[1] "numeric"
> ##character
> chr <- "RwithAammar"
> class(chr)
[1] "character"
> ##Logical
> logi <- TRUE
> class(logi)
[1] "logical"
> #how to confirm a class/type of variable:
> num
[1] 2.2
> is.numeric(num)
[1] TRUE
> is.character(num)
[1] FALSE
> chr
[1] "RwithAammar"
> is.character(chr)
[1] TRUE
> logi
[1] TRUE
> is.logical(logi)
[1] TRUE
> #how to transform a data type
> num1 <- "3.2"
> class (num1)
[1] "character"
> num1 <- as.numeric(num1) #use this to convert data type
> class(num1)
[1] "numeric"
> num2 <- c(1,2, 3,4,5) # vector is who increase continuously
> num2
[1] 1 2 3 4 5
> class(num2)
[1] "numeric"
> num2 <- as.logical(num2)
```

```
> class (num2)
[1] "logical"
> num2 <- c(1,2, 3,4,5,0) # when you use logical operator only value of 0 will be
false
> num2
[1] 1 2 3 4 5 0
> class(num2)
[1] "numeric"
> num2 <- as.logical(num2)
> class (num2)
[1] "logical"
> num2
[1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
> num2 <- as.character(num2)
> class(num2)
[1] "character"
> print(num2) #or num2 only are same things
[1] "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "FALSE"
> ###Data types
> ###numeric
> num <- 2.2
> class(num) #class function use to see data type
[1] "numeric"
> ###character
> chr <- "RwithAammar"
> class(chr)
[1] "character"
> ###Logical
> logi <- TRUE
> class(logi)
[1] "logical"
> ###how to confirm a class/type of variable:
> num
[1] 2.2
> is.numeric(num)
[1] TRUE
> is.character(num)
[1] FALSE
> chr
[1] "RwithAammar"
> is.character(chr)
[1] TRUE
> logi
[1] TRUE
> is.logical(logi)
[1] TRUE
> ###how to transform a data type
> num1 <- "3.2"
> class (num1)
[1] "character"
```

```
> num1 <- as.numeric(num1) #use this to convert data type
> class(num1)
[1] "numeric"
> num2 <- c(1,2, 3,4,5) # vector is who increase continuously
> num2
[1] 1 2 3 4 5
> class(num2)
[1] "numeric"
> num2 <- as.logical(num2)
> class (num2)
[1] "logical"
> num2 <- c(1,2, 3,4,5,0) # when you use logical operator only value of 0 will be
false
> num2
[1] 1 2 3 4 5 0
> class(num2)
[1] "numeric"
> num2 <- as.logical(num2)
> class (num2)
[1] "logical"
> num2
[1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
> num2 <- as.character(num2)
> class(num2)
[1] "character"
> print(num2) #or num2 only are same things
[1] "TRUE"  "TRUE"  "TRUE"  "TRUE"  "TRUE"  "FALSE"
> #Data struture
> data()
> data("diamonds")
> x <- diamonds
> head (x)
# A tibble: 6 x 10
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
> str(x)
tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
 $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
 $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
```

```
 $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
> view(x)
> class(x$carat)
[1] "numeric"
> class(x$price) # class int because its whole number
[1] "integer"
> class(x$x)
[1] "numeric"
> class(x["carat"])
[1] "tbl_df"     "tbl"        "data.frame"
> as.data.frame (x) # not use moslty to see data
    carat       cut color clarity depth table price    x    y    z
1    0.23     Ideal     E     SI2  61.5  55.0   326 3.95 3.98 2.43
2    0.21   Premium     E     SI1  59.8  61.0   326 3.89 3.84 2.31
3    0.23      Good     E     VS1  56.9  65.0   327 4.05 4.07 2.31
4    0.29   Premium     I     VS2  62.4  58.0   334 4.20 4.23 2.63
5    0.31      Good     J     SI2  63.3  58.0   335 4.34 4.35 2.75
6    0.24 Very Good     J    VVS2  62.8  57.0   336 3.94 3.96 2.48
7    0.24 Very Good     I    VVS1  62.3  57.0   336 3.95 3.98 2.47
8    0.26 Very Good     H     SI1  61.9  55.0   337 4.07 4.11 2.53
9    0.22      Fair     E     VS2  65.1  61.0   337 3.87 3.78 2.49
10   0.23 Very Good     H     VS1  59.4  61.0   338 4.00 4.05 2.39
11   0.30      Good     J     SI1  64.0  55.0   339 4.25 4.28 2.73
12   0.23     Ideal     J     VS1  62.8  56.0   340 3.93 3.90 2.46
13   0.22   Premium     F     SI1  60.4  61.0   342 3.88 3.84 2.33
14   0.31     Ideal     J     SI2  62.2  54.0   344 4.35 4.37 2.71
15   0.20   Premium     E     SI2  60.2  62.0   345 3.79 3.75 2.27
16   0.32   Premium     E      I1  60.9  58.0   345 4.38 4.42 2.68
17   0.30     Ideal     I     SI2  62.0  54.0   348 4.31 4.34 2.68
18   0.30      Good     J     SI1  63.4  54.0   351 4.23 4.29 2.70
19   0.30      Good     J     SI1  63.8  56.0   351 4.23 4.26 2.71
20   0.30 Very Good     J     SI1  62.7  59.0   351 4.21 4.27 2.66
21   0.30      Good     I     SI2  63.3  56.0   351 4.26 4.30 2.71
22   0.23 Very Good     E     VS2  63.8  55.0   352 3.85 3.92 2.48
23   0.23 Very Good     H     VS1  61.0  57.0   353 3.94 3.96 2.41
24   0.31 Very Good     J     SI1  59.4  62.0   353 4.39 4.43 2.62
25   0.31 Very Good     J     SI1  58.1  62.0   353 4.44 4.47 2.59
26   0.23 Very Good     G    VVS2  60.4  58.0   354 3.97 4.01 2.41
27   0.24   Premium     I     VS1  62.5  57.0   355 3.97 3.94 2.47
28   0.30 Very Good     J     VS2  62.2  57.0   357 4.28 4.30 2.67
29   0.23 Very Good     D     VS2  60.5  61.0   357 3.96 3.97 2.40
30   0.23 Very Good     F     VS1  60.9  57.0   357 3.96 3.99 2.42
31   0.23 Very Good     F     VS1  60.0  57.0   402 4.00 4.03 2.41
32   0.23 Very Good     F     VS1  59.8  57.0   402 4.04 4.06 2.42
33   0.23 Very Good     E     VS1  60.7  59.0   402 3.97 4.01 2.42
34   0.23 Very Good     E     VS1  59.5  58.0   402 4.01 4.06 2.40
35   0.23 Very Good     D     VS1  61.9  58.0   402 3.92 3.96 2.44
36   0.23      Good     F     VS1  58.2  59.0   402 4.06 4.08 2.37
```

```
37   0.23      Good     E     VS1  64.1  59.0   402 3.83 3.85 2.46
38   0.31      Good     H     SI1  64.0  54.0   402 4.29 4.31 2.75
39   0.26 Very Good     D     VS2  60.8  59.0   403 4.13 4.16 2.52
40   0.33     Ideal     I     SI2  61.8  55.0   403 4.49 4.51 2.78
41   0.33     Ideal     I     SI2  61.2  56.0   403 4.49 4.50 2.75
42   0.33     Ideal     J     SI1  61.1  56.0   403 4.49 4.55 2.76
43   0.26      Good     D     VS2  65.2  56.0   403 3.99 4.02 2.61
44   0.26      Good     D     VS1  58.4  63.0   403 4.19 4.24 2.46
45   0.32      Good     H     SI2  63.1  56.0   403 4.34 4.37 2.75
46   0.29   Premium     F     SI1  62.4  58.0   403 4.24 4.26 2.65
47   0.32 Very Good     H     SI2  61.8  55.0   403 4.35 4.42 2.71
48   0.32      Good     H     SI2  63.8  56.0   403 4.36 4.38 2.79
49   0.25 Very Good     E     VS2  63.3  60.0   404 4.00 4.03 2.54
50   0.29 Very Good     H     SI2  60.7  60.0   404 4.33 4.37 2.64
51   0.24 Very Good     F     SI1  60.9  61.0   404 4.02 4.03 2.45
52   0.23     Ideal     G     VS1  61.9  54.0   404 3.93 3.95 2.44
53   0.32     Ideal     I     SI1  60.9  55.0   404 4.45 4.48 2.72
54   0.22   Premium     E     VS2  61.6  58.0   404 3.93 3.89 2.41
55   0.22   Premium     D     VS2  59.3  62.0   404 3.91 3.88 2.31
56   0.30     Ideal     I     SI2  61.0  59.0   405 4.30 4.33 2.63
57   0.30   Premium     J     SI2  59.3  61.0   405 4.43 4.38 2.61
58   0.30 Very Good     I     SI1  62.6  57.0   405 4.25 4.28 2.67
59   0.30 Very Good     I     SI1  63.0  57.0   405 4.28 4.32 2.71
60   0.30      Good     I     SI1  63.2  55.0   405 4.25 4.29 2.70
61   0.35     Ideal     I     VS1  60.9  57.0   552 4.54 4.59 2.78
62   0.30   Premium     D     SI1  62.6  59.0   552 4.23 4.27 2.66
63   0.30     Ideal     D     SI1  62.5  57.0   552 4.29 4.32 2.69
64   0.30     Ideal     D     SI1  62.1  56.0   552 4.30 4.33 2.68
65   0.42   Premium     I     SI2  61.5  59.0   552 4.78 4.84 2.96
66   0.28     Ideal     G    VVS2  61.4  56.0   553 4.19 4.22 2.58
67   0.32     Ideal     I    VVS1  62.0  55.3   553 4.39 4.42 2.73
68   0.31 Very Good     G     SI1  63.3  57.0   553 4.33 4.30 2.73
69   0.31   Premium     G     SI1  61.8  58.0   553 4.35 4.32 2.68
70   0.24   Premium     E    VVS1  60.7  58.0   553 4.01 4.03 2.44
71   0.24 Very Good     D    VVS1  61.5  60.0   553 3.97 4.00 2.45
72   0.30 Very Good     H     SI1  63.1  56.0   554 4.29 4.27 2.70
73   0.30   Premium     H     SI1  62.9  59.0   554 4.28 4.24 2.68
74   0.30   Premium     H     SI1  62.5  57.0   554 4.29 4.25 2.67
75   0.30      Good     H     SI1  63.7  57.0   554 4.28 4.26 2.72
76   0.26 Very Good     F    VVS2  59.2  60.0   554 4.19 4.22 2.49
77   0.26 Very Good     E    VVS2  59.9  58.0   554 4.15 4.23 2.51
78   0.26 Very Good     D    VVS2  62.4  54.0   554 4.08 4.13 2.56
79   0.26 Very Good     D    VVS2  62.8  60.0   554 4.01 4.05 2.53
80   0.26 Very Good     E    VVS1  62.6  59.0   554 4.06 4.09 2.55
81   0.26 Very Good     E    VVS1  63.4  59.0   554 4.00 4.04 2.55
82   0.26 Very Good     D    VVS1  62.1  60.0   554 4.03 4.12 2.53
83   0.26     Ideal     E    VVS2  62.9  58.0   554 4.02 4.06 2.54
84   0.38     Ideal     I     SI2  61.6  56.0   554 4.65 4.67 2.87
85   0.26      Good     E    VVS1  57.9  60.0   554 4.22 4.25 2.45
86   0.24   Premium     G    VVS1  62.3  59.0   554 3.95 3.92 2.45
```

```
87   0.24   Premium     H    VVS1  61.2  58.0   554 4.01 3.96 2.44
88   0.24   Premium     H    VVS1  60.8  59.0   554 4.02 4.00 2.44
89   0.24   Premium     H    VVS2  60.7  58.0   554 4.07 4.04 2.46
90   0.32   Premium     I     SI1  62.9  58.0   554 4.35 4.33 2.73
91   0.70     Ideal     E     SI1  62.5  57.0  2757 5.70 5.72 3.57
92   0.86      Fair     E     SI2  55.1  69.0  2757 6.45 6.33 3.52
93   0.70     Ideal     G     VS2  61.6  56.0  2757 5.70 5.67 3.50
94   0.71 Very Good     E     VS2  62.4  57.0  2759 5.68 5.73 3.56
95   0.78 Very Good     G     SI2  63.8  56.0  2759 5.81 5.85 3.72
96   0.70      Good     E     VS2  57.5  58.0  2759 5.85 5.90 3.38
97   0.70      Good     F     VS1  59.4  62.0  2759 5.71 5.76 3.40
98   0.96      Fair     F     SI2  66.3  62.0  2759 6.27 5.95 4.07
99   0.73 Very Good     E     SI1  61.6  59.0  2760 5.77 5.78 3.56
100  0.80   Premium     H     SI1  61.5  58.0  2760 5.97 5.93 3.66
 [ reached 'max' / getOption("max.print") -- omitted 53840 rows ]
> as_tibble(x) # tibble adjust its column according to how widely your console
window is open'
# A tibble: 53,940 x 10
   carat cut       color clarity depth table price     x     y     z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
# ... with 53,930 more rows
> glimpse(x) # this function give you overview or bird eye view of your data
Rows: 53,940
Columns: 10
$ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.30,~
$ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Very G~
$ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I, E,~
$ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, SI1~
$ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64.0,~
$ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58, 5~
$ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 342, ~
$ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.25,~
$ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.28,~
$ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.73,~
> #Data Transformation
> data()
> data("diamonds")
> x <- diamonds
> head (x)
# A tibble: 6 x 10
```

```
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
> str(x)
tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
 $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
 $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
> view(x)
> class(x$carat)
[1] "numeric"
> class(x$price) # class int because its whole number
[1] "integer"
> class(x$x)
[1] "numeric"
> class(x["carat"])
[1] "tbl_df"     "tbl"        "data.frame"
> as.data.frame (x) # not use moslty to see data
    carat       cut color clarity depth table price    x    y    z
1    0.23     Ideal     E     SI2  61.5  55.0   326 3.95 3.98 2.43
2    0.21   Premium     E     SI1  59.8  61.0   326 3.89 3.84 2.31
3    0.23      Good     E     VS1  56.9  65.0   327 4.05 4.07 2.31
4    0.29   Premium     I     VS2  62.4  58.0   334 4.20 4.23 2.63
5    0.31      Good     J     SI2  63.3  58.0   335 4.34 4.35 2.75
6    0.24 Very Good     J    VVS2  62.8  57.0   336 3.94 3.96 2.48
7    0.24 Very Good     I    VVS1  62.3  57.0   336 3.95 3.98 2.47
8    0.26 Very Good     H     SI1  61.9  55.0   337 4.07 4.11 2.53
9    0.22      Fair     E     VS2  65.1  61.0   337 3.87 3.78 2.49
10   0.23 Very Good     H     VS1  59.4  61.0   338 4.00 4.05 2.39
11   0.30      Good     J     SI1  64.0  55.0   339 4.25 4.28 2.73
12   0.23     Ideal     J     VS1  62.8  56.0   340 3.93 3.90 2.46
13   0.22   Premium     F     SI1  60.4  61.0   342 3.88 3.84 2.33
14   0.31     Ideal     J     SI2  62.2  54.0   344 4.35 4.37 2.71
15   0.20   Premium     E     SI2  60.2  62.0   345 3.79 3.75 2.27
16   0.32   Premium     E      I1  60.9  58.0   345 4.38 4.42 2.68
17   0.30     Ideal     I     SI2  62.0  54.0   348 4.31 4.34 2.68
18   0.30      Good     J     SI1  63.4  54.0   351 4.23 4.29 2.70
19   0.30      Good     J     SI1  63.8  56.0   351 4.23 4.26 2.71
```

```
20   0.30 Very Good     J     SI1  62.7  59.0   351 4.21 4.27 2.66
21   0.30      Good     I     SI2  63.3  56.0   351 4.26 4.30 2.71
22   0.23 Very Good     E     VS2  63.8  55.0   352 3.85 3.92 2.48
23   0.23 Very Good     H     VS1  61.0  57.0   353 3.94 3.96 2.41
24   0.31 Very Good     J     SI1  59.4  62.0   353 4.39 4.43 2.62
25   0.31 Very Good     J     SI1  58.1  62.0   353 4.44 4.47 2.59
26   0.23 Very Good     G    VVS2  60.4  58.0   354 3.97 4.01 2.41
27   0.24   Premium     I     VS1  62.5  57.0   355 3.97 3.94 2.47
28   0.30 Very Good     J     VS2  62.2  57.0   357 4.28 4.30 2.67
29   0.23 Very Good     D     VS2  60.5  61.0   357 3.96 3.97 2.40
30   0.23 Very Good     F     VS1  60.9  57.0   357 3.96 3.99 2.42
31   0.23 Very Good     F     VS1  60.0  57.0   402 4.00 4.03 2.41
32   0.23 Very Good     F     VS1  59.8  57.0   402 4.04 4.06 2.42
33   0.23 Very Good     E     VS1  60.7  59.0   402 3.97 4.01 2.42
34   0.23 Very Good     E     VS1  59.5  58.0   402 4.01 4.06 2.40
35   0.23 Very Good     D     VS1  61.9  58.0   402 3.92 3.96 2.44
36   0.23      Good     F     VS1  58.2  59.0   402 4.06 4.08 2.37
37   0.23      Good     E     VS1  64.1  59.0   402 3.83 3.85 2.46
38   0.31      Good     H     SI1  64.0  54.0   402 4.29 4.31 2.75
39   0.26 Very Good     D     VS2  60.8  59.0   403 4.13 4.16 2.52
40   0.33     Ideal     I     SI2  61.8  55.0   403 4.49 4.51 2.78
41   0.33     Ideal     I     SI2  61.2  56.0   403 4.49 4.50 2.75
42   0.33     Ideal     J     SI1  61.1  56.0   403 4.49 4.55 2.76
43   0.26      Good     D     VS2  65.2  56.0   403 3.99 4.02 2.61
44   0.26      Good     D     VS1  58.4  63.0   403 4.19 4.24 2.46
45   0.32      Good     H     SI2  63.1  56.0   403 4.34 4.37 2.75
46   0.29   Premium     F     SI1  62.4  58.0   403 4.24 4.26 2.65
47   0.32 Very Good     H     SI2  61.8  55.0   403 4.35 4.42 2.71
48   0.32      Good     H     SI2  63.8  56.0   403 4.36 4.38 2.79
49   0.25 Very Good     E     VS2  63.3  60.0   404 4.00 4.03 2.54
50   0.29 Very Good     H     SI2  60.7  60.0   404 4.33 4.37 2.64
51   0.24 Very Good     F     SI1  60.9  61.0   404 4.02 4.03 2.45
52   0.23     Ideal     G     VS1  61.9  54.0   404 3.93 3.95 2.44
53   0.32     Ideal     I     SI1  60.9  55.0   404 4.45 4.48 2.72
54   0.22   Premium     E     VS2  61.6  58.0   404 3.93 3.89 2.41
55   0.22   Premium     D     VS2  59.3  62.0   404 3.91 3.88 2.31
56   0.30     Ideal     I     SI2  61.0  59.0   405 4.30 4.33 2.63
57   0.30   Premium     J     SI2  59.3  61.0   405 4.43 4.38 2.61
58   0.30 Very Good     I     SI1  62.6  57.0   405 4.25 4.28 2.67
59   0.30 Very Good     I     SI1  63.0  57.0   405 4.28 4.32 2.71
60   0.30      Good     I     SI1  63.2  55.0   405 4.25 4.29 2.70
61   0.35     Ideal     I     VS1  60.9  57.0   552 4.54 4.59 2.78
62   0.30   Premium     D     SI1  62.6  59.0   552 4.23 4.27 2.66
63   0.30     Ideal     D     SI1  62.5  57.0   552 4.29 4.32 2.69
64   0.30     Ideal     D     SI1  62.1  56.0   552 4.30 4.33 2.68
65   0.42   Premium     I     SI2  61.5  59.0   552 4.78 4.84 2.96
66   0.28     Ideal     G    VVS2  61.4  56.0   553 4.19 4.22 2.58
67   0.32     Ideal     I    VVS1  62.0  55.3   553 4.39 4.42 2.73
68   0.31 Very Good     G     SI1  63.3  57.0   553 4.33 4.30 2.73
69   0.31   Premium     G     SI1  61.8  58.0   553 4.35 4.32 2.68
```

```
70   0.24   Premium     E    VVS1  60.7  58.0   553 4.01 4.03 2.44
71   0.24 Very Good     D    VVS1  61.5  60.0   553 3.97 4.00 2.45
72   0.30 Very Good     H     SI1  63.1  56.0   554 4.29 4.27 2.70
73   0.30   Premium     H     SI1  62.9  59.0   554 4.28 4.24 2.68
74   0.30   Premium     H     SI1  62.5  57.0   554 4.29 4.25 2.67
75   0.30      Good     H     SI1  63.7  57.0   554 4.28 4.26 2.72
76   0.26 Very Good     F    VVS2  59.2  60.0   554 4.19 4.22 2.49
77   0.26 Very Good     E    VVS2  59.9  58.0   554 4.15 4.23 2.51
78   0.26 Very Good     D    VVS2  62.4  54.0   554 4.08 4.13 2.56
79   0.26 Very Good     D    VVS2  62.8  60.0   554 4.01 4.05 2.53
80   0.26 Very Good     E    VVS1  62.6  59.0   554 4.06 4.09 2.55
81   0.26 Very Good     E    VVS1  63.4  59.0   554 4.00 4.04 2.55
82   0.26 Very Good     D    VVS1  62.1  60.0   554 4.03 4.12 2.53
83   0.26     Ideal     E    VVS2  62.9  58.0   554 4.02 4.06 2.54
84   0.38     Ideal     I     SI2  61.6  56.0   554 4.65 4.67 2.87
85   0.26      Good     E    VVS1  57.9  60.0   554 4.22 4.25 2.45
86   0.24   Premium     G    VVS1  62.3  59.0   554 3.95 3.92 2.45
87   0.24   Premium     H    VVS1  61.2  58.0   554 4.01 3.96 2.44
88   0.24   Premium     H    VVS1  60.8  59.0   554 4.02 4.00 2.44
89   0.24   Premium     H    VVS2  60.7  58.0   554 4.07 4.04 2.46
90   0.32   Premium     I     SI1  62.9  58.0   554 4.35 4.33 2.73
91   0.70     Ideal     E     SI1  62.5  57.0  2757 5.70 5.72 3.57
92   0.86      Fair     E     SI2  55.1  69.0  2757 6.45 6.33 3.52
93   0.70     Ideal     G     VS2  61.6  56.0  2757 5.70 5.67 3.50
94   0.71 Very Good     E     VS2  62.4  57.0  2759 5.68 5.73 3.56
95   0.78 Very Good     G     SI2  63.8  56.0  2759 5.81 5.85 3.72
96   0.70      Good     E     VS2  57.5  58.0  2759 5.85 5.90 3.38
97   0.70      Good     F     VS1  59.4  62.0  2759 5.71 5.76 3.40
98   0.96      Fair     F     SI2  66.3  62.0  2759 6.27 5.95 4.07
99   0.73 Very Good     E     SI1  61.6  59.0  2760 5.77 5.78 3.56
100  0.80   Premium     H     SI1  61.5  58.0  2760 5.97 5.93 3.66
 [ reached 'max' / getOption("max.print") -- omitted 53840 rows ]
> as_tibble(x) # tibble adjust its column according to how widely your console
window is open'
# A tibble: 53,940 x 10
   carat cut       color clarity depth table price     x     y     z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
# ... with 53,930 more rows
> glimpse(x) # this function give you overview or bird eye view of your data
Rows: 53,940
```

```
Columns: 10
$ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.30,~
$ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Very G~
$ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I, E,~
$ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, SI1~
$ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64.0,~
$ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58, 5~
$ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 342, ~
$ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.25,~
$ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.28,~
$ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.73,~
> #convert into right data type?
> x$carat <- as.numeric(x$carat, x$depth)
> class(x$table)
[1] "numeric"
> glimpse(x)
Rows: 53,940
Columns: 10
$ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.30,~
$ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Very G~
$ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I, E,~
$ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, SI1~
$ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64.0,~
$ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58, 5~
$ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 342, ~
$ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.25,~
$ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.28,~
$ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.73,~
> str(x)
tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
 $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
 $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
 $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
 $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
 $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
 $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
 $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
 $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
 $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
> #vector
> vecl <- c(1,2,3,4,6,9,0,-1,0.5)
> class (vecl)
[1] "numeric"
> #Factor
> vec2  <- c("blue", "red", "purple", 'green') # can be use a in color command to
give color to different variable on x axis sequence wise
> class(vec2)
[1] "character"
> vec2_fac <- as.factor(vec2)
```

```
> class(vec2_fac )
[1] "factor"
> #tidy means to collect data and make it usable for statistics in R
> x <- CO2
> glimpse(x)
Rows: 84
Columns: 5
$ Plant     <ord> Qn1, Qn1, Qn1, Qn1, Qn1, Qn1, Qn1, Qn2, Qn2, Qn2, Qn2, Qn2, Qn2~
$ Type      <fct> Quebec, Quebec, Quebec, Quebec, Quebec, Quebec, Quebec, Quebec,~
$ Treatment <fct> nonchilled, nonchilled, nonchilled, nonchilled, nonchilled, non~
$ conc      <dbl> 95, 175, 250, 350, 500, 675, 1000, 95, 175, 250, 350, 500, 675,~
$ uptake    <dbl> 16.0, 30.4, 34.8, 37.2, 35.3, 39.2, 39.7, 13.6, 27.3, 37.1, 41.~
> as_tibble(x)
# A tibble: 84 x 5
   Plant Type   Treatment   conc uptake
   <ord> <fct>  <fct>      <dbl>  <dbl>
 1 Qn1   Quebec nonchilled    95   16
 2 Qn1   Quebec nonchilled   175   30.4
 3 Qn1   Quebec nonchilled   250   34.8
 4 Qn1   Quebec nonchilled   350   37.2
 5 Qn1   Quebec nonchilled   500   35.3
 6 Qn1   Quebec nonchilled   675   39.2
 7 Qn1   Quebec nonchilled  1000   39.7
 8 Qn2   Quebec nonchilled    95   13.6
 9 Qn2   Quebec nonchilled   175   27.3
10 Qn2   Quebec nonchilled   250   37.1
# ... with 74 more rows
> as.data.frame (x)
   Plant        Type  Treatment conc uptake
1    Qn1      Quebec nonchilled   95   16.0
2    Qn1      Quebec nonchilled  175   30.4
3    Qn1      Quebec nonchilled  250   34.8
4    Qn1      Quebec nonchilled  350   37.2
5    Qn1      Quebec nonchilled  500   35.3
6    Qn1      Quebec nonchilled  675   39.2
7    Qn1      Quebec nonchilled 1000   39.7
8    Qn2      Quebec nonchilled   95   13.6
9    Qn2      Quebec nonchilled  175   27.3
10   Qn2      Quebec nonchilled  250   37.1
11   Qn2      Quebec nonchilled  350   41.8
12   Qn2      Quebec nonchilled  500   40.6
13   Qn2      Quebec nonchilled  675   41.4
14   Qn2      Quebec nonchilled 1000   44.3
15   Qn3      Quebec nonchilled   95   16.2
16   Qn3      Quebec nonchilled  175   32.4
17   Qn3      Quebec nonchilled  250   40.3
18   Qn3      Quebec nonchilled  350   42.1
19   Qn3      Quebec nonchilled  500   42.9
20   Qn3      Quebec nonchilled  675   43.9
21   Qn3      Quebec nonchilled 1000   45.5
```

```
22   Qc1      Quebec    chilled   95   14.2
23   Qc1      Quebec    chilled  175   24.1
24   Qc1      Quebec    chilled  250   30.3
25   Qc1      Quebec    chilled  350   34.6
26   Qc1      Quebec    chilled  500   32.5
27   Qc1      Quebec    chilled  675   35.4
28   Qc1      Quebec    chilled 1000   38.7
29   Qc2      Quebec    chilled   95    9.3
30   Qc2      Quebec    chilled  175   27.3
31   Qc2      Quebec    chilled  250   35.0
32   Qc2      Quebec    chilled  350   38.8
33   Qc2      Quebec    chilled  500   38.6
34   Qc2      Quebec    chilled  675   37.5
35   Qc2      Quebec    chilled 1000   42.4
36   Qc3      Quebec    chilled   95   15.1
37   Qc3      Quebec    chilled  175   21.0
38   Qc3      Quebec    chilled  250   38.1
39   Qc3      Quebec    chilled  350   34.0
40   Qc3      Quebec    chilled  500   38.9
41   Qc3      Quebec    chilled  675   39.6
42   Qc3      Quebec    chilled 1000   41.4
43   Mn1 Mississippi nonchilled   95   10.6
44   Mn1 Mississippi nonchilled  175   19.2
45   Mn1 Mississippi nonchilled  250   26.2
46   Mn1 Mississippi nonchilled  350   30.0
47   Mn1 Mississippi nonchilled  500   30.9
48   Mn1 Mississippi nonchilled  675   32.4
49   Mn1 Mississippi nonchilled 1000   35.5
50   Mn2 Mississippi nonchilled   95   12.0
51   Mn2 Mississippi nonchilled  175   22.0
52   Mn2 Mississippi nonchilled  250   30.6
53   Mn2 Mississippi nonchilled  350   31.8
54   Mn2 Mississippi nonchilled  500   32.4
55   Mn2 Mississippi nonchilled  675   31.1
56   Mn2 Mississippi nonchilled 1000   31.5
57   Mn3 Mississippi nonchilled   95   11.3
58   Mn3 Mississippi nonchilled  175   19.4
59   Mn3 Mississippi nonchilled  250   25.8
60   Mn3 Mississippi nonchilled  350   27.9
61   Mn3 Mississippi nonchilled  500   28.5
62   Mn3 Mississippi nonchilled  675   28.1
63   Mn3 Mississippi nonchilled 1000   27.8
64   Mc1 Mississippi    chilled   95   10.5
65   Mc1 Mississippi    chilled  175   14.9
66   Mc1 Mississippi    chilled  250   18.1
67   Mc1 Mississippi    chilled  350   18.9
68   Mc1 Mississippi    chilled  500   19.5
69   Mc1 Mississippi    chilled  675   22.2
70   Mc1 Mississippi    chilled 1000   21.9
71   Mc2 Mississippi    chilled   95    7.7
```

```
72   Mc2 Mississippi    chilled  175   11.4
73   Mc2 Mississippi    chilled  250   12.3
74   Mc2 Mississippi    chilled  350   13.0
75   Mc2 Mississippi    chilled  500   12.5
76   Mc2 Mississippi    chilled  675   13.7
77   Mc2 Mississippi    chilled 1000   14.4
78   Mc3 Mississippi    chilled   95   10.6
79   Mc3 Mississippi    chilled  175   18.0
80   Mc3 Mississippi    chilled  250   17.9
81   Mc3 Mississippi    chilled  350   17.9
82   Mc3 Mississippi    chilled  500   17.9
83   Mc3 Mississippi    chilled  675   18.9
84   Mc3 Mississippi    chilled 1000   19.9
> x1 <- chickwts
> as_tibble(x1)
# A tibble: 71 x 2
   weight feed
    <dbl> <fct>
 1    179 horsebean
 2    160 horsebean
 3    136 horsebean
 4    227 horsebean
 5    217 horsebean
 6    168 horsebean
 7    108 horsebean
 8    124 horsebean
 9    143 horsebean
10    140 horsebean
# ... with 61 more rows
> library (dslabs)
> x <- diamonds
> #add a column
> as_tibble(x)
# A tibble: 53,940 x 10
   carat cut       color clarity depth table price     x     y     z
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
# ... with 53,930 more rows
> x <- mutate(x, a = x+y)
> as_tibble(x)
# A tibble: 53,940 x 11
```

```
   carat cut       color clarity depth table price     x     y     z     a
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43  7.93
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31  7.73
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31  8.12
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63  8.43
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75  8.69
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48  7.9
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47  7.93
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53  8.18
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49  7.65
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39  8.05
# ... with 53,930 more rows
> x2<- mutate (x, sd=  sd(y+z))
> as_tibble(x2)
# A tibble: 53,940 x 12
   carat cut       color clarity depth table price     x     y     z     a    sd
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43  7.93  1.83
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31  7.73  1.83
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31  8.12  1.83
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63  8.43  1.83
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75  8.69  1.83
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48  7.9   1.83
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47  7.93  1.83
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53  8.18  1.83
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49  7.65  1.83
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39  8.05  1.83
# ... with 53,930 more rows
> # #subsetting
> as_tibble(x)
# A tibble: 53,940 x 11
   carat cut       color clarity depth table price     x     y     z     a
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43  7.93
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31  7.73
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31  8.12
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63  8.43
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75  8.69
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48  7.9
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47  7.93
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53  8.18
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49  7.65
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39  8.05
# ... with 53,930 more rows
> x_al <- filter(x, a == 7.93)
> as_tibble(x_al)
# A tibble: 28 x 11
   carat cut       color clarity depth table price     x     y     z     a
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl>
```

```
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43  7.93
 2  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47  7.93
 3  0.23 Very Good D     VS2      60.5    61   357  3.96  3.97  2.4   7.93
 4  0.23 Premium   F     VVS2     61.3    59   571  3.99  3.94  2.43  7.93
 5  0.23 Very Good E     VVS1     62.4    54   583  3.95  3.98  2.47  7.93
 6  0.23 Ideal     E     VS1      61.5    54   586  3.96  3.97  2.44  7.93
 7  0.23 Premium   I     VVS1     60.5    61   414  3.98  3.95  2.4   7.93
 8  0.24 Very Good F     VS1      62      56   417  3.94  3.99  2.46  7.93
 9  0.24 Very Good E     VS2      63.1    56   419  3.95  3.98  2.5   7.93
10  0.23 Premium   E     VVS1     60.8    56   640  3.99  3.94  2.41  7.93
# ... with 18 more rows
> X_a2 <- filter(x, cut=="Ideal")#subsetting a character variable
> as_tibble(X_a2)
# A tibble: 21,551 x 11
   carat cut   color clarity depth table price     x     y     z     a
   <dbl> <ord> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl>
 1  0.23 Ideal E     SI2      61.5    55   326  3.95  3.98  2.43  7.93
 2  0.23 Ideal J     VS1      62.8    56   340  3.93  3.9   2.46  7.83
 3  0.31 Ideal J     SI2      62.2    54   344  4.35  4.37  2.71  8.72
 4  0.3  Ideal I     SI2      62      54   348  4.31  4.34  2.68  8.65
 5  0.33 Ideal I     SI2      61.8    55   403  4.49  4.51  2.78  9
 6  0.33 Ideal I     SI2      61.2    56   403  4.49  4.5   2.75  8.99
 7  0.33 Ideal J     SI1      61.1    56   403  4.49  4.55  2.76  9.04
 8  0.23 Ideal G     VS1      61.9    54   404  3.93  3.95  2.44  7.88
 9  0.32 Ideal I     SI1      60.9    55   404  4.45  4.48  2.72  8.93
10  0.3  Ideal I     SI2      61      59   405  4.3   4.33  2.63  8.63
# ... with 21,541 more rows
> #subsetting using select function
> as_tibble(x)
# A tibble: 53,940 x 11
   carat cut       color clarity depth table price     x     y     z     a
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43  7.93
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31  7.73
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31  8.12
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63  8.43
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75  8.69
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48  7.9
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47  7.93
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53  8.18
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49  7.65
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39  8.05
# ... with 53,930 more rows
> x1 <- select(x, cut, color, x)
> as_tibble(x1)
# A tibble: 53,940 x 3
   cut       color     x
   <ord>     <ord> <dbl>
 1 Ideal     E      3.95
 2 Premium   E      3.89
```

```
 3 Good      E      4.05
 4 Premium   I      4.2
 5 Good      J      4.34
 6 Very Good J      3.94
 7 Very Good I      3.95
 8 Very Good H      4.07
 9 Fair      E      3.87
10 Very Good H      4
# ... with 53,930 more rows
> #task for attendees
> x <- murders
> as_tibble(x)
# A tibble: 51 x 5
   state                abb   region    population total
   <chr>                <chr> <fct>          <dbl> <dbl>
 1 Alabama              AL    South        4779736   135
 2 Alaska               AK    West          710231    19
 3 Arizona              AZ    West         6392017   232
 4 Arkansas             AR    South        2915918    93
 5 California           CA    West        37253956  1257
 6 Colorado             CO    West         5029196    65
 7 Connecticut          CT    Northeast    3574097    97
 8 Delaware             DE    South         897934    38
 9 District of Columbia DC    South         601723    99
10 Florida              FL    South       19687653   669
# ... with 41 more rows
> x2 <- filter(x, region=="West", population>5000000)
> as_tibble(x2)
# A tibble: 4 x 5
  state      abb   region population total
  <chr>      <chr> <fct>       <dbl> <dbl>
1 Arizona    AZ    West      6392017   232
2 California CA    West     37253956  1257
3 Colorado   CO    West      5029196    65
4 Washington WA    West      6724540    93
> plot(x2)
> #Less code for more output
> library(magrittr) # needs to be run every time you start R and want to use %>%
> library(dplyr)    # alternatively, this also loads %>%
> x <- murders %>%
+   filter(region=="West") %>%
+   select(region, population) %>%
+   mutate(population_2=population+2)
> view(x)
> x
   region population population_2
1    West     710231       710233
2    West    6392017      6392019
3    West   37253956     37253958
4    West    5029196      5029198
```

```
5    West    1360301      1360303
6    West    1567582      1567584
7    West     989415       989417
8    West    2700551      2700553
9    West    2059179      2059181
10   West    3831074      3831076
11   West    2763885      2763887
12   West    6724540      6724542
13   West     563626       563628
```

```
library(tidyverse)
library(tidyr)

#Summarise Function
x <- mtcars
view(x)
as_tibble(x)
class(x$mpg)
help("summarise")
summarise(x, avg = mean (mpg))
count (x, mpg)
x %>% summarise(avg= mean(hp), count=n())

#group by
mtcars %>%
  group_by (cyl) %>%
  summarise(avg = mean (mpg))

starwars %>%
  rowwise() %>%
  mutate(film_count = length(films))


#Manipulate Data
x1 <- filter(mtcars, mpg > 20)
view(x1)
data (mtcars)
x2 <- distinct (mtcars, gear)
x2 <- slice (mtcars, 21:25)
x2 <- slice_sample(mtcars, n = 10, replace = TRUE)
x2 <- slice_min(mtcars, mpg, prop = 1) #proportion in percet of data

x2 <- slice_head (mtcars, n = 10)
head (mtcars)
#order data
x1 <- arrange (mtcars, mpg) #By default ascending
x2 <- arrange (mtcars, desc (mpg)) #descending
head (x1)
```

```
head (x2)
#add rows
data(cars)
#cars <- add_row(cars, speed = 20, dist = 5)
cars <- add_row (cars, speed = 20)
cars[is.na (cars)] = 0 #to convert na va lues into zero

#manipulating variables
x <- pull (mtcars, wt)
x1 <- select (mtcars, mpg, cyl)
x1 <- select (mtcars, mpg:drat)
x1 <- relocate (mtcars, mpg, cyl, .after = last_col())


summarise (mtcars, mean(mpg)) #for only one variable
x1 <- summarise (mtcars, across (everything(), mean)) #across variables

mtcars %>%
  rowwise() %>%
  mutate(zzz = (mpg+cyl+disp)/3) #we have to recheck the across functions?

x1 <- mutate (mtcars, gpm = 1/ mpg)
x2 <- transmute (mtcars, gpm = 1 / mpg)
x1 <- rename (x1, R_with_Aammar = gpm)
```

output

```
> library(tidyverse)
> library(tidyr)
> #Summarise Function
> x <- mtcars
> view(x)
> as_tibble(x)
# A tibble: 32 x 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21       6  160    110  3.9   2.62  16.5     0     1     4     4
 2  21       6  160    110  3.9   2.88  17.0     0     1     4     4
 3  22.8     4  108     93  3.85  2.32  18.6     1     1     4     1
 4  21.4     6  258    110  3.08  3.22  19.4     1     0     3     1
 5  18.7     8  360    175  3.15  3.44  17.0     0     0     3     2
 6  18.1     6  225    105  2.76  3.46  20.2     1     0     3     1
 7  14.3     8  360    245  3.21  3.57  15.8     0     0     3     4
 8  24.4     4  147.    62  3.69  3.19  20       1     0     4     2
 9  22.8     4  141.    95  3.92  3.15  22.9     1     0     4     2
10  19.2     6  168.   123  3.92  3.44  18.3     1     0     4     4
# ... with 22 more rows
> class(x$mpg)
```

```
[1] "numeric"
> help("summarise")
> summarise(x, avg = mean (mpg))
       avg
1 20.09062
> count (x, mpg)
    mpg n
1  10.4 2
2  13.3 1
3  14.3 1
4  14.7 1
5  15.0 1
6  15.2 2
7  15.5 1
8  15.8 1
9  16.4 1
10 17.3 1
11 17.8 1
12 18.1 1
13 18.7 1
14 19.2 2
15 19.7 1
16 21.0 2
17 21.4 2
18 21.5 1
19 22.8 2
20 24.4 1
21 26.0 1
22 27.3 1
23 30.4 2
24 32.4 1
25 33.9 1
> x %>% summarise(avg= mean(hp), count=n())
       avg count
1 146.6875    32
> #group by
> mtcars %>%
+   group_by (cyl) %>%
+   summarise(avg = mean (mpg))
# A tibble: 3 x 2
    cyl   avg
  <dbl> <dbl>
1     4  26.7
2     6  19.7
3     8  15.1
> starwars %>%
+   rowwise() %>%
+   mutate(film_count = length(films))
# A tibble: 87 x 15
# Rowwise:
```

```
   name               height  mass hair_color    skin_color  eye_color birth_year
sex    gender    homeworld species films     vehicles  starships film_count
   <chr>               <int> <dbl> <chr>         <chr>       <chr>          <dbl>
<chr>  <chr>     <chr>     <chr>   <list>    <list>    <list>         <int>
 1 Luke Skywalker        172    77 blond         fair        blue            19
male   masculine Tatooine  Human   <chr [5]> <chr [2]> <chr [2]>          5
 2 C-3PO                 167    75 NA            gold        yellow         112
none   masculine Tatooine  Droid   <chr [6]> <chr [0]> <chr [0]>          6
 3 R2-D2                  96    32 NA            white, blue red             33
none   masculine Naboo     Droid   <chr [7]> <chr [0]> <chr [0]>          7
 4 Darth Vader           202   136 none          white       yellow          41.9
male   masculine Tatooine  Human   <chr [4]> <chr [0]> <chr [1]>          4
 5 Leia Organa           150    49 brown         light       brown           19
female feminine  Alderaan  Human   <chr [5]> <chr [1]> <chr [0]>          5
 6 Owen Lars             178   120 brown, grey   light       blue            52
male   masculine Tatooine  Human   <chr [3]> <chr [0]> <chr [0]>          3
 7 Beru Whitesun lars    165    75 brown         light       blue            47
female feminine  Tatooine  Human   <chr [3]> <chr [0]> <chr [0]>          3
 8 R5-D4                  97    32 NA            white, red  red             NA
none   masculine Tatooine  Droid   <chr [1]> <chr [0]> <chr [0]>          1
 9 Biggs Darklighter     183    84 black         light       brown           24
male   masculine Tatooine  Human   <chr [1]> <chr [0]> <chr [1]>          1
10 Obi-Wan Kenobi        182    77 auburn, white fair        blue-gray       57
male   masculine Stewjon   Human   <chr [6]> <chr [1]> <chr [5]>          6
# ... with 77 more rows
> #Manipulate Data
> x1 <- filter(mtcars, mpg > 20)
> view(x1)
> data (mtcars)
> x2 <- distinct (mtcars, gear)
> x2 <- slice (mtcars, 21:25)
> x2 <- slice_sample(mtcars, n = 10, replace = TRUE)
> x2 <- slice_min(mtcars, mpg, prop = 1) #proportion in percet of data
> x2 <- slice_head (mtcars, n = 10)
> head (mtcars)
                   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
> #order data
> x1 <- arrange (mtcars, mpg) #By default ascending
> x2 <- arrange (mtcars, desc (mpg)) #descending
> head (x1)
                     mpg cyl disp  hp drat    wt  qsec vs am gear carb
Cadillac Fleetwood  10.4   8  472 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8  460 215 3.00 5.424 17.82  0  0    3    4
Camaro Z28          13.3   8  350 245 3.73 3.840 15.41  0  0    3    4
```

```
Duster 360          14.3   8  360 245 3.21 3.570 15.84  0  0    3    4
Chrysler Imperial   14.7   8  440 230 3.23 5.345 17.42  0  0    3    4
Maserati Bora       15.0   8  301 335 3.54 3.570 14.60  0  1    5    8
> head (x2)
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
> #add rows
> data(cars)
> #cars <- add_row(cars, speed = 20, dist = 5)
> cars <- add_row (cars, speed = 20)
> cars[is.na (cars)] = 0 #to convert na va lues into zero
> #manipulating variables
> x <- pull (mtcars, wt)
> x1 <- select (mtcars, mpg, cyl)
> x1 <- select (mtcars, mpg:drat)
> x1 <- relocate (mtcars, mpg, cyl, .after = last_col())
> summarise (mtcars, mean(mpg)) #for only one variable
  mean(mpg)
1  20.09062
> x1 <- summarise (mtcars, across (everything(), mean)) #across variables
> mtcars %>%
+   rowwise() %>%
+   mutate(zzz = (mpg+cyl+disp)/3) #we have to recheck the across functions?
# A tibble: 32 x 12
# Rowwise:
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb   zzz
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21       6  160    110  3.9   2.62  16.5     0     1     4     4  62.3
 2  21       6  160    110  3.9   2.88  17.0     0     1     4     4  62.3
 3  22.8     4  108     93  3.85  2.32  18.6     1     1     4     1  44.9
 4  21.4     6  258    110  3.08  3.22  19.4     1     0     3     1  95.1
 5  18.7     8  360    175  3.15  3.44  17.0     0     0     3     2 129.
 6  18.1     6  225    105  2.76  3.46  20.2     1     0     3     1  83.0
 7  14.3     8  360    245  3.21  3.57  15.8     0     0     3     4 127.
 8  24.4     4  147.    62  3.69  3.19  20       1     0     4     2  58.4
 9  22.8     4  141.    95  3.92  3.15  22.9     1     0     4     2  55.9
10  19.2     6  168.   123  3.92  3.44  18.3     1     0     4     4  64.3
# ... with 22 more rows
> x1 <- mutate (mtcars, gpm = 1/ mpg)
> x2 <- transmute (mtcars, gpm = 1 / mpg)
> x1 <- rename (x1, R_with_Aammar = gpm)
```

Animated graphs

```
#Animation?


#Rapid display of Sequence of images
#. Illusion of movement
#. Optical Illusion due to static eye
#Motion picture or Video
#. 2D or 3D animation

# Packages needs to be installed
#install.packages("ggplot2")
#install.packages('gganimate')
#install.packages ("gifski")
#install.packeges("av")

#load libaraies
library(ggplot2)
library(gifski)
library(av)
library(gapminder)
library(gganimate)
theme_set(theme_bw())

# dataset

head(gapminder)

# Static plot

p <- ggplot(
  gapminder,
  aes(x = gdpPercap, y=lifeExp, size = pop, colour = country)
) +
  geom_point(show.legend = FALSE, alpha = 0.7) +
  scale_color_viridis_d() +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
  labs(x = "GDP per capita", y = "Life expectancy")
p

# Transition through distinct states in time

p + transition_time(year) +
  labs(title = "Year: {frame_time}")

# Create facets by continent

p + facet_wrap(~continent) +
  transition_time(year) +
```

```
  labs(title = "Year: {frame_time}")

# Let the view follow the data in each frame
p + transition_time(year) +
  labs(title = "Year: {frame_time}") +
  view_follow(fixed_y = TRUE)

# Show preceding frames with gradual falloff

p + transition_time(year) +
  labs(title = "Year: {frame_time}") +
  shadow_wake(wake_length = 0.1, alpha = FALSE)

# Show the original data as background marks

p + transition_time(year) +
  labs(title = "Year: {frame_time}") +
  shadow_mark(alpha = 0.3, size = 0.5)

#Reveal data along a given dimension
# Static plot

p <- ggplot(
  airquality,
  aes(Day, Temp, group = Month, color = factor(Month))
) +
  geom_line() +
  scale_color_viridis_d() +
  labs(x = "Day of Month", y = "Temperature") +
  theme(legend.position = "top")
p

## Reveal by day (x-axis)
p + transition_reveal(Day)

# Reveal by day (x-axis)

p +
  geom_point() +
  transition_reveal(Day)

# Points can be kept by giving them a unique group

p +
  geom_point(aes(group = seq_along(Day))) +
  transition_reveal(Day)

# Transition between several distinct stages of the data
library(dplyr)
mean.temp <- airquality %>%
```

```
  group_by(Month) %>%
  summarise(Temp = mean(Temp))
mean.temp

# Create a bar plot of mean temperature
p <- ggplot(mean.temp, aes(Month, Temp, fill = Temp)) +
  geom_col() +
  scale_fill_distiller(palette = "Reds", direction = 1) +
  theme_minimal() +
  theme(
    panel.grid = element_blank(),
    panel.grid.major.y = element_line(color = "white"),
    panel.ontop = TRUE
  )
p

# transition_states()
p + transition_states(Month, wrap = FALSE) +
  shadow_mark()

#enter_grow() + enter_fade()

p + transition_states(Month, wrap = FALSE) +
  shadow_mark() +
  enter_grow() +
  enter_fade()
anim_save("barplot.gif")
```

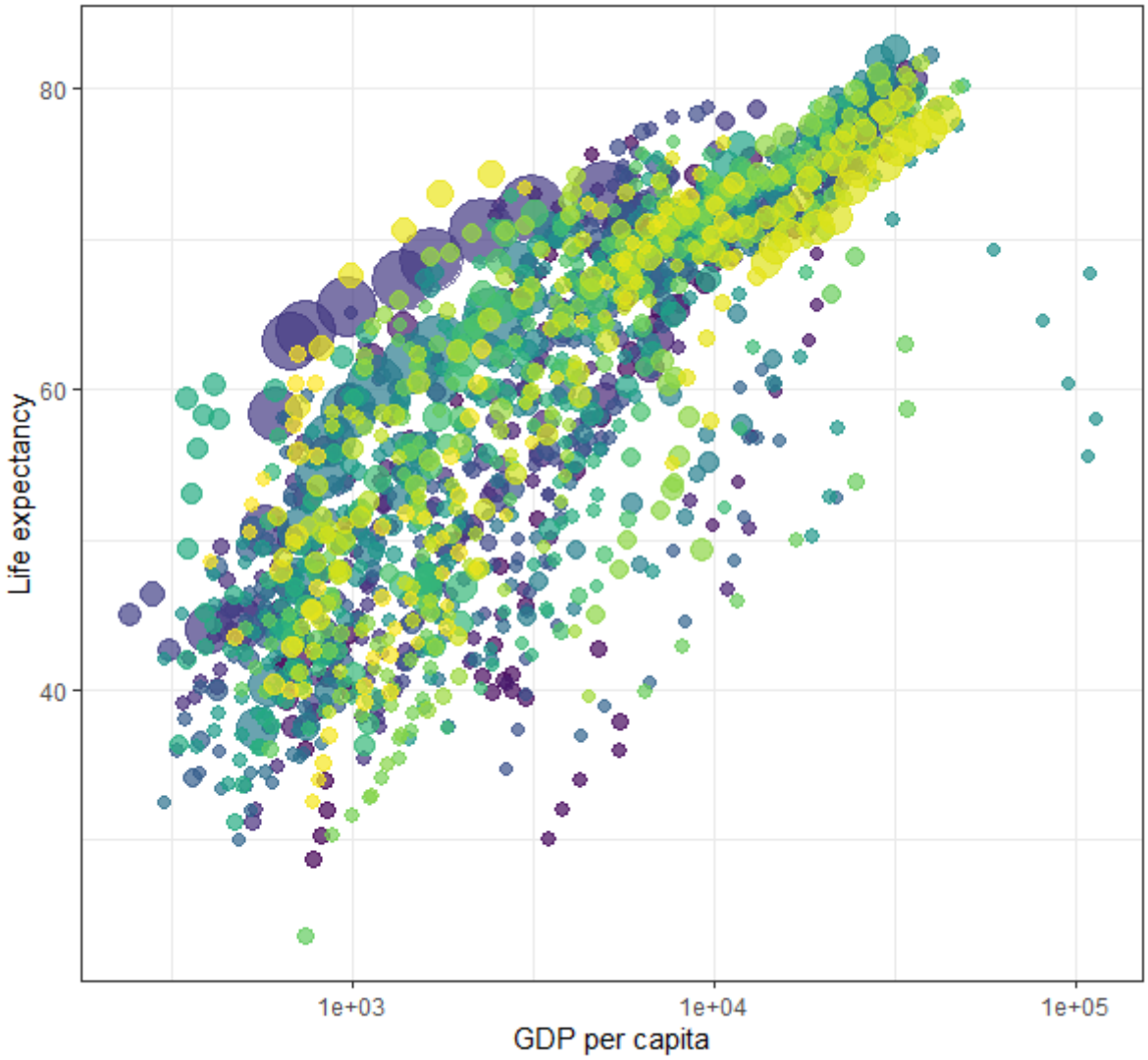
output

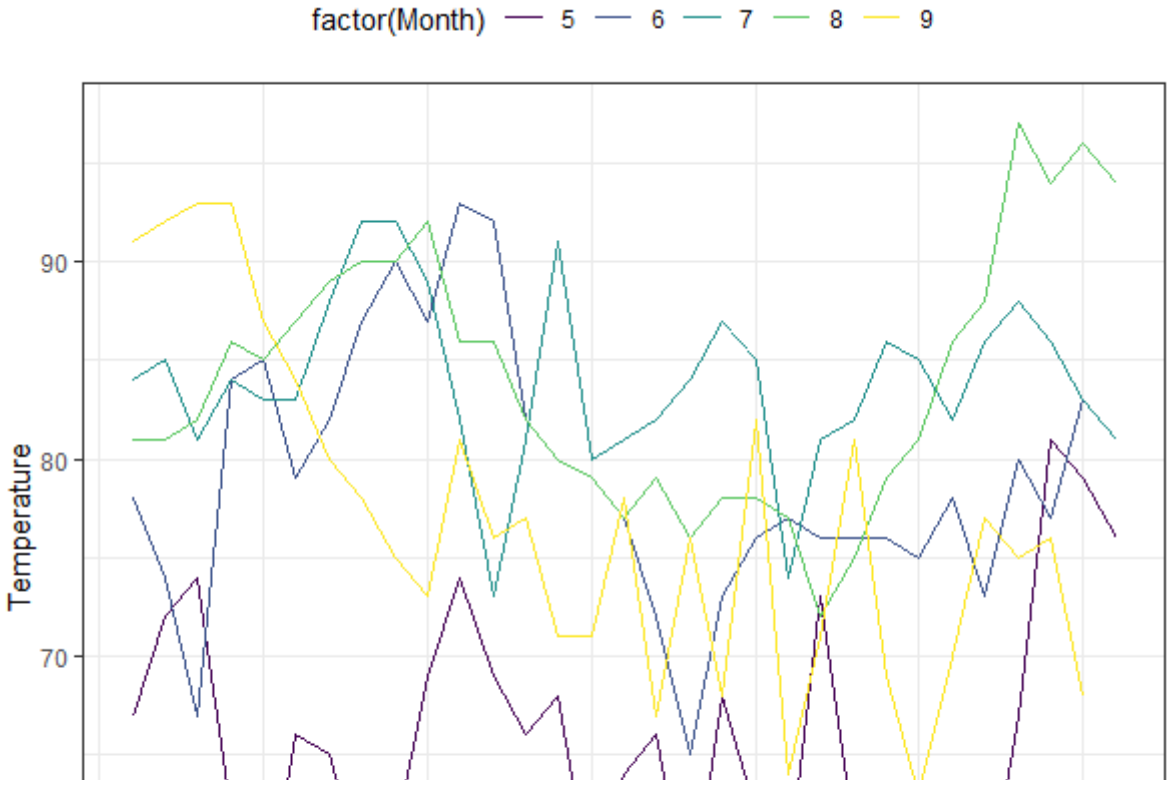
```
> #load libaraies
> library(ggplot2)
> library(gifski)
> library(av)
> library(gapminder)
> library(gganimate)
> theme_set(theme_bw())
> head(gapminder)
# A tibble: 6 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
4 Afghanistan Asia       1967    34.0 11537966      836.
5 Afghanistan Asia       1972    36.1 13079460      740.
6 Afghanistan Asia       1977    38.4 14880372      786.
> p <- ggplot(
```

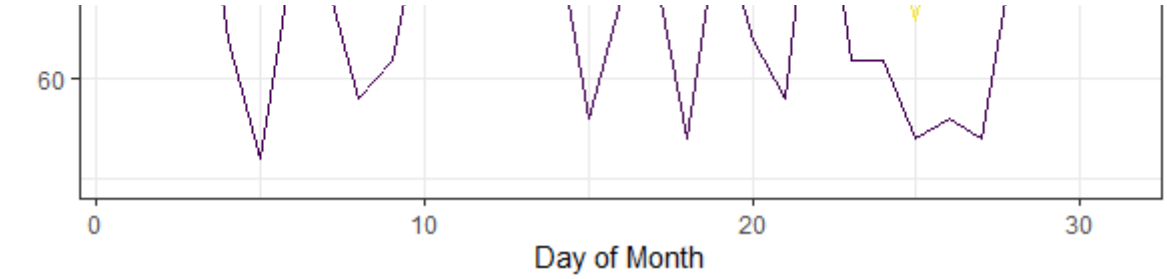
```
+   gapminder,
+   aes(x = gdpPercap, y=lifeExp, size = pop, colour = country)
+ ) +
+   geom_point(show.legend = FALSE, alpha = 0.7) +
+   scale_color_viridis_d() +
+   scale_size(range = c(2, 12)) +
+   scale_x_log10() +
+   labs(x = "GDP per capita", y = "Life expectancy")
> p
> p + transition_time(year) +
+   labs(title = "Year: {frame_time}")

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> p + facet_wrap(~continent) +
+   transition_time(year) +
+   labs(title = "Year: {frame_time}")

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> # Let the view follow the data in each frame
> p + transition_time(year) +
+   labs(title = "Year: {frame_time}") +
+   view_follow(fixed_y = TRUE)

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> p + transition_time(year) +
+   labs(title = "Year: {frame_time}") +
+   shadow_wake(wake_length = 0.1, alpha = FALSE)

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> p + transition_time(year) +
+   labs(title = "Year: {frame_time}") +
+   shadow_mark(alpha = 0.3, size = 0.5)

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> p <- ggplot(
+   airquality,
+   aes(Day, Temp, group = Month, color = factor(Month))
+ ) +
+   geom_line() +
+   scale_color_viridis_d() +
+   labs(x = "Day of Month", y = "Temperature") +
+   theme(legend.position = "top")
> p
> ## Reveal by day (x-axis)
> p + transition_reveal(Day)
```

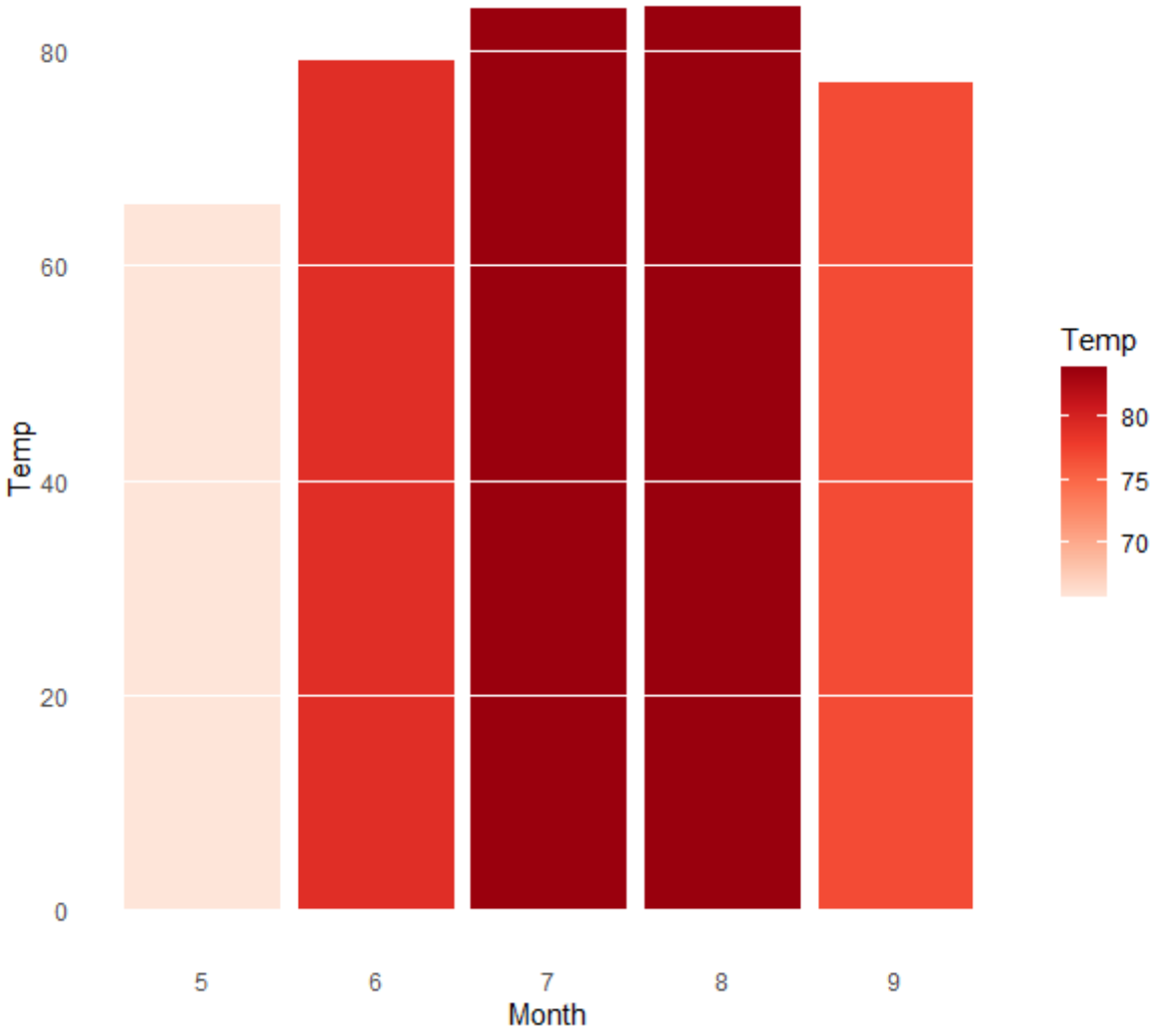
```
geom_path: Each group consists of only one observation. Do you need to adjust the
group aesthetic?
geom_path: Each group consists of only one observation. Do you need to adjust the
group aesthetic?

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> p +
+   geom_point() +
+   transition_reveal(Day)
geom_path: Each group consists of only one observation. Do you need to adjust the
group aesthetic?
geom_path: Each group consists of only one observation. Do you need to adjust the
group aesthetic?

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> p +
+   geom_point(aes(group = seq_along(Day))) +
+   transition_reveal(Day)
geom_path: Each group consists of only one observation. Do you need to adjust the
group aesthetic?
geom_path: Each group consists of only one observation. Do you need to adjust the
group aesthetic?

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> # Transition between several distinct stages of the data
> library(dplyr)
> mean.temp <- airquality %>%
+   group_by(Month) %>%
+   summarise(Temp = mean(Temp))
> mean.temp
# A tibble: 5 x 2
  Month  Temp
  <int> <dbl>
1     5  65.5
2     6  79.1
3     7  83.9
4     8  84.0
5     9  76.9
> # Create a bar plot of mean temperature
> p <- ggplot(mean.temp, aes(Month, Temp, fill = Temp)) +
+   geom_col() +
+   scale_fill_distiller(palette = "Reds", direction = 1) +
+   theme_minimal() +
+   theme(
+     panel.grid = element_blank(),
+     panel.grid.major.y = element_line(color = "white"),
+     panel.ontop = TRUE
```

```
+   )
> p
> # transition_states()
> p + transition_states(Month, wrap = FALSE) +
+   shadow_mark()

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> p + transition_states(Month, wrap = FALSE) +
+   shadow_mark() +
+   enter_grow() +
+   enter_fade()

Inserting image 100 at 9.90s (100%)...
Encoding to gif... done!
> anim_save("barplot.gif")
```


80
60
40
Life expectancy
1e+03
1e+04
1e+05
GDP per capita


factor(Month) 5 6 7 8 9
90
80
70
Temperature


60
0
10
20
30
Day of Month


80
60
40
20
0
Temp
5
6
7
8
9
Month
Temp
80
75
70