# *Introduction to R*

## *ICCWPMBB 2023*

Dahrii Paul

15-02-2023

## *1.Introduction*

Acquiring, analyzing and presenting data are fundamental components of any research endeavour. This module describes various data structures and ways to present data using R. R is an open-source, free software. It offers a wide range of libraries and packages for data analysis, visualization, and statistical modelling. They are simple and easy to learn, read & write.

### *There are multiple books you can refer to:*

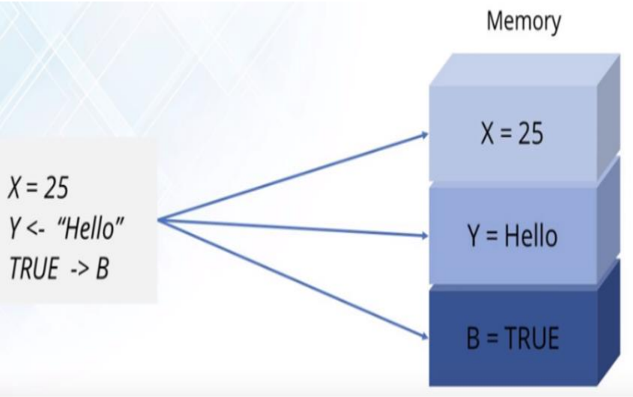
- **R-Manuals cran**
- **R in Action by *Rob Kabacoff***
- **R Graphics by *Paul Murrell***
- **R Graphics Cookbook *by Winston Chang***
- **ggplot2 *by Hadley Wickham***
- **Lattice Multivariate Data Visualization with R *by***
- **Interactive and Dynamic Graphics for Data Analysis *by Cook and Swayne***

## *2. Installation*

- Download R: https://cran.r-project.org/bin/windows/base/
- Download Rstudio: https://www.rstudio.com/
- R cloud: https://rstudio.cloud/

## 3. Variables in R

- Variables are reserved memory location to store values.
- i.e When you create a variable you reserved some space in memory



## 4. Operators in R

### a). Assignment Operators

| *Operators* | | *Example* |
|---|---|---|
| Right assignment | = | x = 5 |
| | <- | x <- 5 |
| | <<- | x <<- 5 |
| Left assignment | -> | 3 -> y |
| | ->> | 3 ->> y |

*Assignment Operators*

```
x = 5
x <- 5
x <<- 5
x <- x + 3
3 -> y
3 ->> y
```

### b). Arithmetic Operators

| Operators | | Example | |
|---|---|---|---|
| | | input | return |
| Addition | + | 2 + 3 | 5 |
| Subtraction | - | 5 - 2 | 3 |
| Multiplication | * | 3 * 4 | 12 |
| Division | / | 6/2 | 3 |
| Exponentiation | ^ | 2^3 | 8 |
| Integer division | %/% | 7 %/% 2 | 3 |
| Modulus | %% | 7 %% 2 | 1 |

*Arithmetic Operators*

```r
a <- 2
b <- 3
c <- a + b # c will return 5
d <- a - b # d will return -1
e <- a * b # e will return 6
f <- a / b # f will return 0.66667
g <- a^b  # g will return 8
h <- 7 %/% 2 # h will return 3
i <- 7 %% 2 # i will return 1
```

### c). Relational Operators

| Operators | | Example | |
|---|---|---|---|
| | | input | return |
| less than | < | 2<3 | TRUE |
| greater than | > | 2>3 | FALSE |
| greater than or equal to | >= | 2>=3 | FALSE |
| less than or equal to | <= | 2<=3 | TRUE |
| equal to | == | 2==2 | TRUE |
| not equal to | != | 2!=2 | TRUE |

### d). Logical Operators

| Operators | |
|---|---|
| And | & |
| Or | \| |
| Not | ! |

- ***And Operator '&'***

```
a <- 5
b <- 10
ifelse(a > 3 & b < 15,
       "Both conditions are true", "Either condition is false")

## [1] "Both conditions are true"
```

- ***Or Operator '|'***

```
a <- 5
b <- 10
ifelse(a < 3 | b > 15,
       "Either conditions are true", "Both condition are false")

## [1] "Both condition are false"
```

- ***Not Operator '!'***

```
a <- 5
b <- 10
ifelse(!(a > 3 & b < 15),"Either condition is false",

                         "Both conditions are true")

## [1] "Both conditions are true"
```

### e). Special Operators

| Operators | | Example |
|---|---|---|
| Help | ? | ?vector |
| Sequence | : | x=1:3 |
| Matching | %in% | x=1:3;y=2;y%in%x |
| List subset | $ | dataframe$column |

## *5. Basic Data Structures*

### a) Vectors

- A vector is a sequence of data elements of the same basic type
- It can contain elements of different data types, such as numeric, character, or logical values

- ***Create a numeric vector***

```
v1<-c(1,2,3,4,5,6)
v1

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

- **Create a numeric vector using *c(range)***

```
v2<-c(5:11)
v2

## [1]  5  6  7  8  9 10 11
```

- ***Create a string vector***

```
v3 <- c("A","A","G","T","C","G")
v3

## [1] "A" "A" "G" "T" "C" "G"
```

- ***Create mix vector type***

```
v_mix <- c("new",1,2,3,"four")
v_mix

## [1] "new"  "1"    "2"    "3"    "four"
```

- ***Create an integer vector***

```
v4<-c(8L,16L,64L,128L)
v4

## [1]   8  16  64 128
```

### b) Factors

- Factor is a data structure which are used to categorize the data and store it as levels
- Can store both integers and strings

```
v5 <- as.factor(v3)
v5

## [1] A A G T C G
## Levels: A C G T
```

### c) Array

- A multi-dimensional data structure that can store data in more than two dimensions
- Arrays hold multidimensional rectangular data
- "***Rectangular***" means that each row is the same length, and likewise for each column and other dimensions
- Arrays can store only values having similar kinds of data, i.e. variables / elements having similar data type

- ***Create an Array 1-D***

```
array_1<-array(c(v1))
array_1

## [1] 1 2 3 4 5 6

class(array_1)

## [1] "array"
```

- ***Create an Array 2-D***

```
array_2<-array(1:12,c(4,3))
array_2

##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

- ***Create multiple-D array***

```
array_multi<- array(1:24,c(3,4,2))
array_multi

## , , 1
## 
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
## 
## , , 2
## 
##      [,1] [,2] [,3] [,4]
## [1,]   13   16   19   22
## [2,]   14   17   20   23
## [3,]   15   18   21   24
```

### d) Matrices

- They are 2-dimensional data structures arranged in a rectangular layout
- Can have only homogeneous element type

```
length(v1)

## [1] 6

#Copy the vector
mat1 <- v1
dim(mat1) <- c(3,2)
mat1

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

class(mat1)

## [1] "matrix" "array"

dim(mat1)

## [1] 3 2

mat2 <- cbind(v1,v2)

mat2

##      v1 v2
## [1,]  1  5
## [2,]  2  6
## [3,]  3  7
## [4,]  4  8
## [5,]  5  9
## [6,]  6 10
## [7,]  1 11

mat3 <- rbind(v1,v2)

mat3

##    [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## v1    1    2    3    4    5    6    1
## v2    5    6    7    8    9   10   11
```

- ***Create a matrix using ‘matrix’ function***

```
mat4 <-  matrix(c(v1, v2), nrow = 6, ncol = 2)

## Warning in matrix(c(v1, v2), nrow = 6, ncol = 2): data length [13]
is not a
## sub-multiple or multiple of the number of rows [6]

mat4

##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
## [5,]    5    9
## [6,]    6   10

# Create a matrix – by range
mat5 <- matrix(c(1:5), nrow = 4, ncol = 4)

mat5

##      [,1] [,2] [,3] [,4]
## [1,]    1    5    4    3
## [2,]    2    1    5    4
## [3,]    3    2    1    5
## [4,]    4    3    2    1

mat5 <- matrix(c(1:5), nrow = 4, byrow = TRUE)

mat5

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

## e) Lists

- Objects which contain elements of different types such as strings, numbers, vectors or another list inside under one name

```
ls1 <- list(v1,v2,v3,v4,array_1,array_2,array_multi,mat1,mat2,mat3,mat5)
ls1[[3]]

## [1] "A" "A" "G" "T" "C" "G"

ls1[[6]][2,2]

## [1] 6
```

## f) Data Frame

- A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column
- ***Characteristics of a data frame***
  - The column names should be non-empty
  - The row names should be unique
  - The data stored in a data frame can be of numeric, factor or character type
  - Each column should contain same number of data items

```
dim(mat4);length(v3)

## [1] 6 2

## [1] 6

df1 <-data.frame(mat4,v3)
df1

##   X1 X2 v3
## 1  1  5  A
## 2  2  6  A
## 3  3  7  G
## 4  4  8  T
## 5  5  9  C
## 6  6 10  G

colnames(df1)[1:3] <- c("var1","var2","DNA")
colnames(df1)

## [1] "var1" "var2" "DNA"

names(df1)[1] <- "col1"
colnames(df1)

## [1] "col1" "var2" "DNA"
```

# 6. Data Wrangling

```
#install.packages("MASS")
library(MASS)
data(package = "MASS")
```

*Load the data*

```
data(cats)
head(cats)

##   Sex Bwt Hwt
## 1   F 2.0 7.0
## 2   F 2.0 7.4
## 3   F 2.0 9.5
## 4   F 2.1 7.2
## 5   F 2.1 7.3
## 6   F 2.1 7.6

tail(cats)

##     Sex Bwt  Hwt
## 139   M 3.6 15.0
## 140   M 3.7 11.0
## 141   M 3.8 14.8
## 142   M 3.8 16.8
## 143   M 3.9 14.4
## 144   M 3.9 20.5

dim(cats)

## [1] 144   3

str(cats)

## 'data.frame':    144 obs. of  3 variables:
##  $ Sex: Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Bwt: num  2 2 2 2.1 2.1 2.1 2.1 2.1 2.1 2.1 ...
##  $ Hwt: num  7 7.4 9.5 7.2 7.3 7.6 8.1 8.2 8.3 8.5 ...

summary(cats)

##  Sex         Bwt             Hwt
##  F:47   Min.   :2.000   Min.   : 6.30
##  M:97   1st Qu.:2.300   1st Qu.: 8.95
##         Median :2.700   Median :10.10
##         Mean   :2.724   Mean   :10.63
##         3rd Qu.:3.025   3rd Qu.:12.12
##         Max.   :3.900   Max.   :20.50
```

- ***Select subset***

```
cats[,1]

cats$Sex

cats$Sex[1]

## [1] F
## Levels: F M

males <- subset(cats, cats$Sex == "M")
females <- subset(cats, cats$Sex == "F")
summary(males)

##  Sex          Bwt            Hwt
##  F: 0   Min.   :2.0   Min.   : 6.50
##  M:97   1st Qu.:2.5   1st Qu.: 9.40
##         Median :2.9   Median :11.40
##         Mean   :2.9   Mean   :11.32
##         3rd Qu.:3.2   3rd Qu.:12.80
##         Max.   :3.9   Max.   :20.50

summary(females)

##  Sex          Bwt             Hwt
##  F:47   Min.   :2.00   Min.   : 6.300
##  M: 0   1st Qu.:2.15   1st Qu.: 8.350
##         Median :2.30   Median : 9.100
##         Mean   :2.36   Mean   : 9.202
##         3rd Qu.:2.50   3rd Qu.:10.100
##         Max.   :3.00   Max.   :13.000

sd(males$Hwt)

## [1] 2.542288

cats1 <-cats
cats1$Sex <- as.character(cats1$Sex)
str(cats1)

## 'data.frame':    144 obs. of  3 variables:
##  $ Sex: chr  "F" "F" "F" "F" ...
##  $ Bwt: num  2 2 2 2.1 2.1 2.1 2.1 2.1 2.1 2.1 ...
##  $ Hwt: num  7 7.4 9.5 7.2 7.3 7.6 8.1 8.2 8.3 8.5 ...

cats1$Sex[cats1$Sex == "F"] = 1
cats1$Sex[cats1$Sex == "M"] = 2
table(cats$Sex)

##
##  F  M
## 47 97
```

## Join and Merge

- ***Data Frame one as df1***

```
df1 <- data.frame(id = c(1,2,3,4), name = c("potein1", "potein2",
                                            "potein3","potein4"))
df1

##   id    name
## 1  1 potein1
## 2  2 potein2
## 3  3 potein3
## 4  4 potein4
```

- ***Data Frame one as df2***

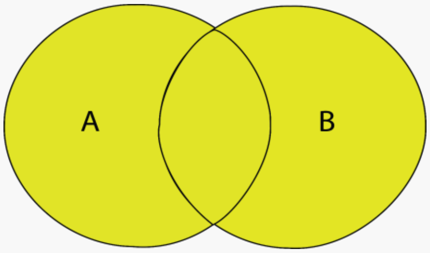
```
df2 <- data.frame(id = c(2,3,4,5), a.site = c(25, 30, 35, 40))
df2

##   id a.site
## 1  2     25
## 2  3     30
## 3  4     35
## 4  5     40
```

- **Function *merge()***

```
merged_df <- merge(df1, df2, by = "id", all.x = TRUE)
merged_df

##   id    name a.site
## 1  1 potein1     NA
## 2  2 potein2     25
## 3  3 potein3     30
## 4  4 potein4     35

merged_df <- merge(df1, df2, by = "id", all.x = FALSE)
merged_df

##   id    name a.site
## 1  2 potein2     25
## 2  3 potein3     30
## 3  4 potein4     35
```

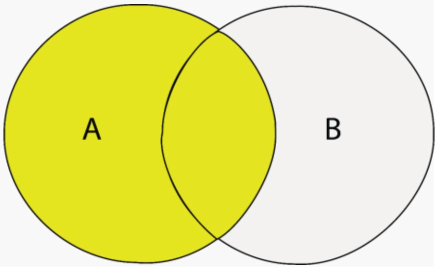
- **Function ‘dplyr’ package**
  **Function *full_join()***

```
library(dplyr)

full_df <- full_join(df1, df2, by = "id")
full_df

##   id    name a.site
## 1  1 potein1     NA
## 2  2 potein2     25
## 3  3 potein3     30
## 4  4 potein4     35
## 5  5    <NA>     40
```



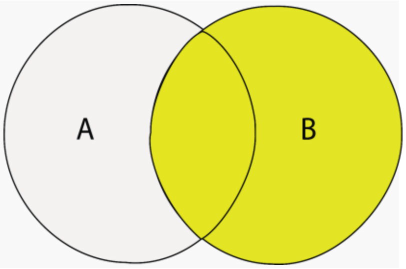
- **Function *left_join()***

```
left_df <- left_join(df1, df2, by = "id")
left_df

##   id    name a.site
## 1  1 potein1     NA
## 2  2 potein2     25
## 3  3 potein3     30
## 4  4 potein4     35
```



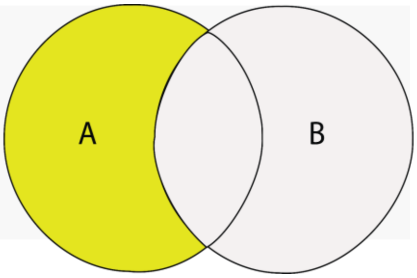
- **Function *right_join()***

```
right_df <- right_join(df1, df2, by = "id")
right_df

##   id    name a.site
## 1  2 potein2     25
## 2  3 potein3     30
## 3  4 potein4     35
## 4  5    <NA>     40
```



- **Function *semi_join()*** *i.e. either for df1 or df2*

```
semi_df <- semi_join(df1,df2, by = "id")
semi_df

##   id    name
## 1  2 potein2
## 2  3 potein3
## 3  4 potein4
```



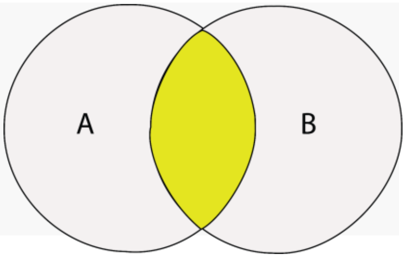
- **Function *anti_join()***

```
anti_df <- anti_join(df1, df2, by = "id")
anti_df

##   id    name
## 1  1 potein1
```

- **Function *inner_join()***

```
inner_df <- inner_join(df1, df2, by = "id")
inner_df
```

```
##   id    name a.site
## 1  2 potein2     25
## 2  3 potein3     30
## 3  4 potein4     35
```



UCI Machine Learning Repository “**Census Income**” link

Data link github

```r
#install.packages("dplyr")
library(dplyr)
df <- read.csv("https://raw.githubusercontent.com/Dahrii-Paul/R_Basic/d1f0be2d9bc12bfd1df3093723db9c40f8865a78/adult.csv")
head(df,2)
dim(df)
## [1] 2839   15
```

### a) Function *‘filter()’*

- The *filter()* function is used to subset a data frame, retaining all rows that satisfy your conditions i.e based on a logical condition
- It is part of the *‘dplyr’* package

```r
colnames(df)
df$native.country <- as.factor(df$native.country)
levels(df$native.country)
filter(df, native.country %in% "Scotland")
filter(df,native.country %in% c("Scotland","Honduras"))
filter(df,native.country %in% c("Scotland","Honduras"), hours.per.week > 50 )
```

### b) Function *‘select()’*

- The *select()* function in R is used to pick specific variables or features of a DataFrame
- It is part of the *‘dplyr’* package
- *select(data, column1, column2, ...)*
- *select(data, -column1, -column2, ...)*

```r
dplyr::select(df, age, income)
dplyr::select(df, -age, -income)
```

### c) Pipe operator %>%

- Pipe operator *%>%* is a special operator commonly used in *'dplyr package'*, which allow multiple sequence of operations *(function/argument)* on a data frame
- **syntax**

  ***data %>% function1() %>% function2() %>% function3() %>% argument***

```
df %>%
  filter(native.country %in% c("Scotland","Honduras"), sex == "Male",
hours.per.week > 50) %>%
  select(age, native.country, sex, hours.per.week)

##   age native.country  sex hours.per.week
## 1  29       Scotland Male             55
## 2  49       Scotland Male             60
```

### Summary

```
df %>%
  select(-workclass, -education, -occupation, -marital.status, -
relationship,-race,-sex, -native.country, -income) %>%
  summarise_all(list(mn=mean, stdev=sd))

##     age_mn fnlwgt_mn education.num_mn capital.gain_mn capital.loss_mn
## 1 38.37654  215582.9         9.223318        746.3596        220.6904
##   hours.per.week_mn age_stdev fnlwgt_stdev education.num_stdev
## 1          40.84185  12.94998     110356.2            3.748357
##   capital.gain_stdev capital.loss_stdev hours.per.week_stdev
## 1           6037.088           682.4993              11.6732
```

### Group Level

```
df %>%
  select(age, race, sex, hours.per.week) %>%
  group_by(race)%>%
  summarise(sampSz=n(), Avg =mean(hours.per.week), stDev =
sd(hours.per.week))

## # A tibble: 5 × 4
##   race               sampSz   Avg stDev
##   <chr>               <int> <dbl> <dbl>
## 1 Amer-Indian-Eskimo     15  38.2  9.15
## 2 Asian-Pac-Islander    622  40.9 12.1
## 3 Black                 198  38.2  9.48
## 4 Other                 118  40.9 12.2
## 5 White                1886  41.1 11.7
```

### Sub-setting data population sample size

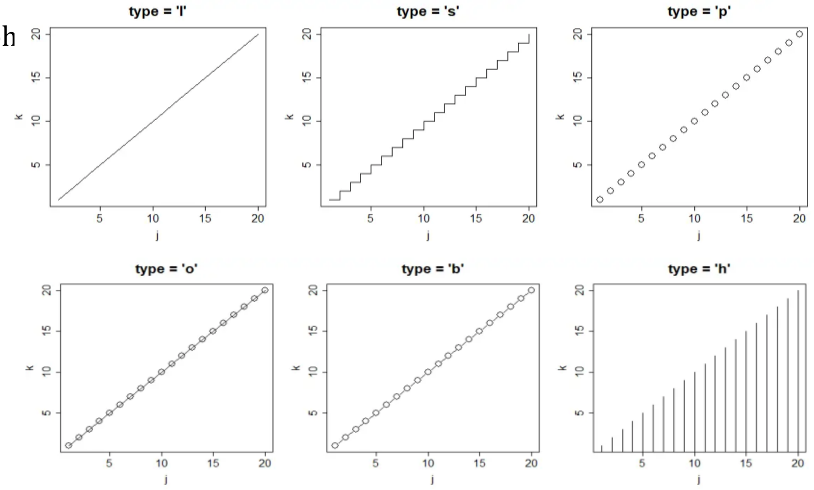
```
df2 <-df %>%
  select(age, native.country, sex, hours.per.week) %>%
  group_by(native.country)%>%
  mutate(samplSz=n())%>%
  filter(samplSz >50) %>%
  ungroup()
df2

## # A tibble: 2,353 × 5
##      age native.country sex    hours.per.week samplSz
##    <int> <fct>          <chr>           <int>   <int>
##  1    82 United-States  Female             18     181
##  2    54 United-States  Female             40     181
##  3    41 United-States  Female             40     181
##  4    34 United-States  Female             45     181
##  5    38 United-States  Male               40     181
##  6    74 United-States  Female             20     181
##  7    68 United-States  Female             40     181
##  8    45 United-States  Female             35     181
##  9    38 United-States  Male               45     181
## 10    52 United-States  Female             20     181
## # … with 2,343 more rows
```

## *7. Data Visualization*

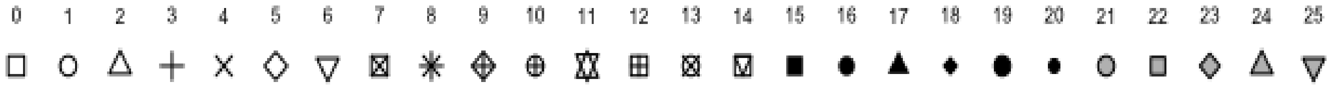
### *Features in plot*

1. ***type***: determines the type of the output graph

| | |
|---|---|
| *type = "p"* | for points |
| *type = "l"* | for line |
| *type = "b"* | for both |
| *type = "c"* | for  the line part alone of "b" |
| *type = "h"* | for histogram like vertical lines |
| *type = "s"* | for stair steps |
| *type = "n"* | for no plotting |



2. ***pch***: *pch* argument is for the plotting character



*3.* ***col****:* Stands for colors of the plot. There is a choice of 667 colors.

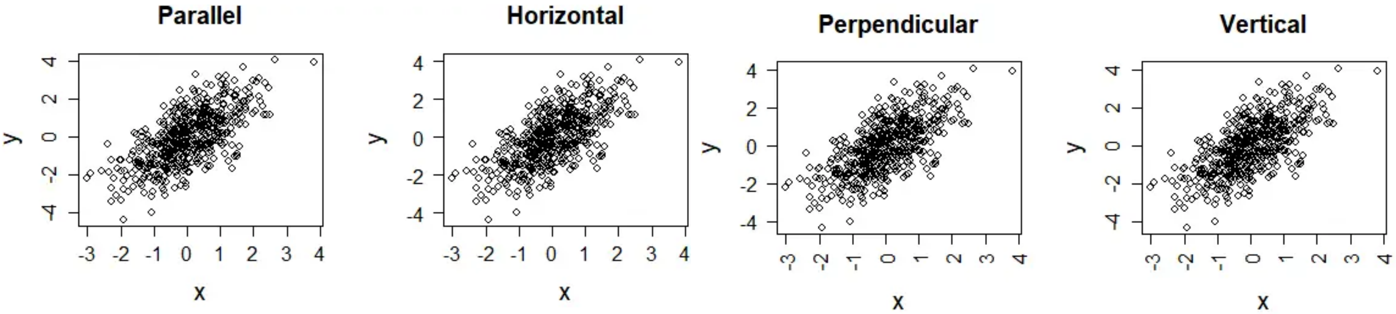
4. ***las***: It's a label style.

*las = 0* (the default) mark labels are parallel to the axis

*las = 1* labels are perpendicular to the axis

*las = 2* labels are parallel to the axis, but rotated by 90 degrees

*las = 3* mark labels are perpendicular to the axis, and rotated by 90 degrees



5. ***xlim***: It is the limits of the values of x used for plotting. Example `xlim = c(0, 10)` the lower limit of the x-axis to 0 and the upper limit to 10.

6. ***ylim***: It is the limits of the values of y used for plotting. Example `ylim = c(0, 10)` the lower limit of the y-axis to 0 and the upper limit to 10.

7. ***main***: Facilitates a main title for the graph. The title can be colored. Example `col.main =”red”`

8. ***sub***: Facilitates a subtitle title for the graph

9. ***cex***: Determine the size of the plotting character

*cex = 0* (the default)

*cex = 1.5* 50% larger

*cex = 0.5* 50% smaller

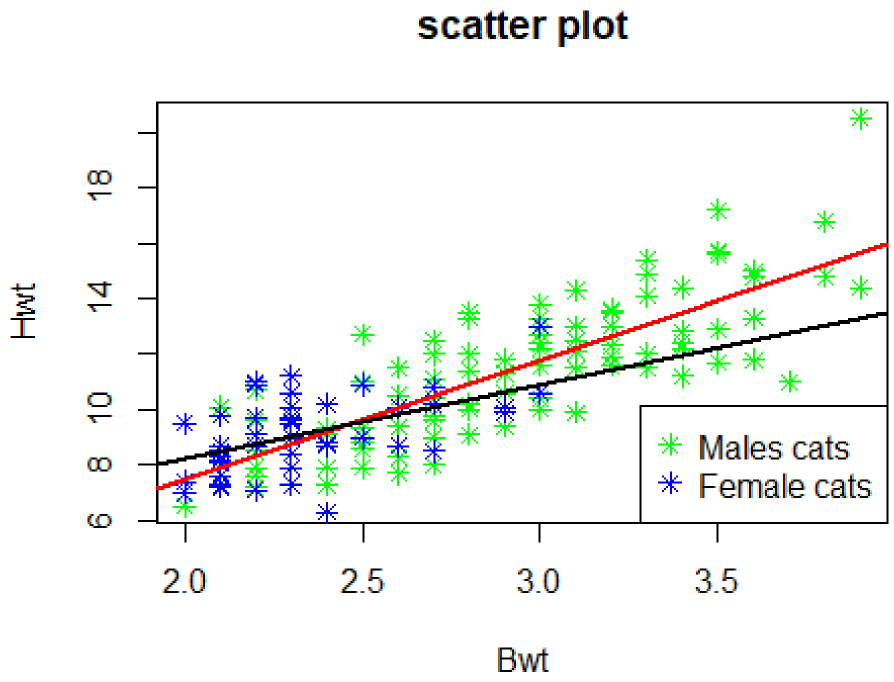
10. ***cex.lab***: Determine the size of the text labels on the axes.

*11.* ***cex.axis****:* Determine the size of the numbers on the tick marks.

*12.* ***par()****:* Facilitates accommodation of several graph in a single frame
*Example:* ***par(mfrow = c(2,3), oma= c(2,0,4,0))***
***oma****: outer margins used to specify the size of the outer margins of a plot. It has four-element vector that specifies the margin sizes in the following order: bottom, left, top, and right.*

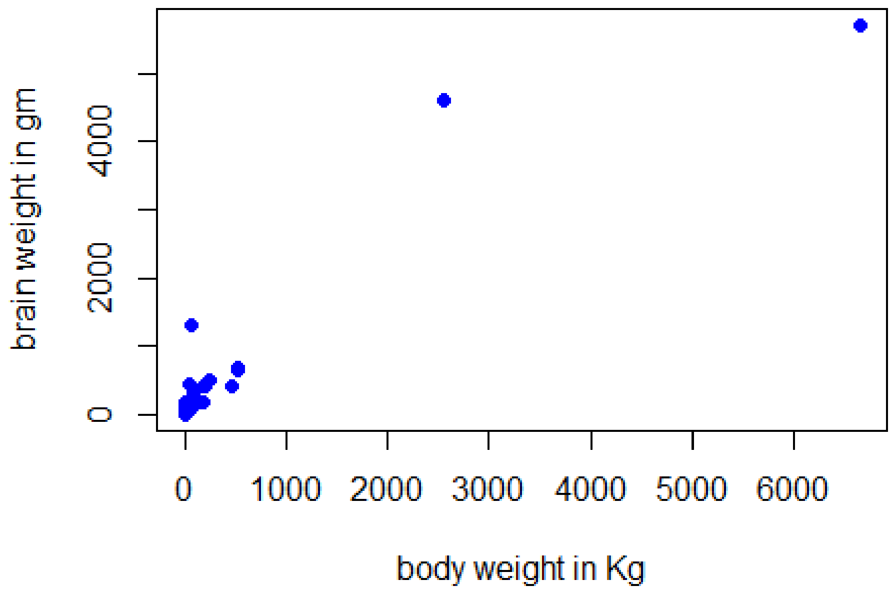
## plot

```
library(MASS)

data(cats)
males <- subset(cats, cats$Sex == "M")
females <- subset(cats, cats$Sex == "F")


plot(males$Bwt,males$Hwt,
     pch = 8,
     xlab = "Bwt", ylab = "Hwt",
     col = "green", main = "scatter plot", las =0)
points(females$Bwt,females$Hwt,
       pch = 8,
       xlab = "Bwt", ylab = "Hwt",
       col = "blue", main = "scatter plot", las =0)


malesReg <- lm(Hwt ~ Bwt ,data = males)
abline(malesReg, col = "red" , lwd = 2)
femaleReg <- lm(Hwt ~ Bwt,data = females)
abline(femaleReg, col = "black",lwd =2)
legend("bottomright",legend = c("Males cats","Female cats"),
       pch = c(8,8), col = c("green","blue"))
```

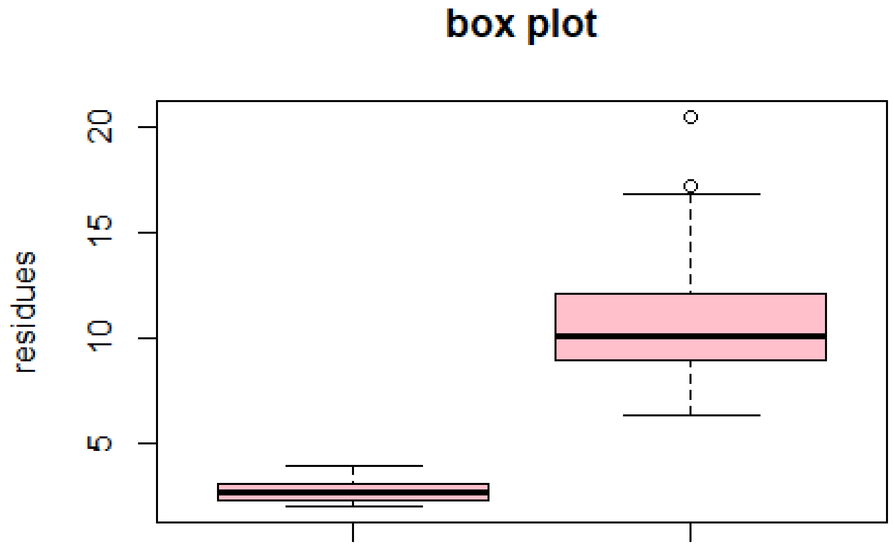


***Identify point using name***

```
data(mammals)
plot(mammals$body,mammals$brain ,
     pch =  16,
     col = "blue",
     las = 0,
     xlab = "body weight in Kg",ylab = "brain weight in gm")

identify(mammals$body,mammals$brain, labels = rownames(mammals))
```



```
boxplot(cats$Bwt,cats$Hwt, col = "pink", ylab = "residues", main = "box
plot")
```


box plot
residues