- **Reading data from a CSV file**

(Comma separated value (CSV)) or a file that separates values with a semicolon (;) or a comma (;), as in the example below:

Example :

```
n = c(2, 3, 5)
s = c("red", "green", "white")
b = c(TRUE, FALSE, TRUE)
df = data.frame(n, s, b)       #       df هو اطار بيانات
```

To download this file, you can use the read.table() function after specifying the interval, or use the read.csv() function of type csv in both ways:

df=read.table(" <FileName>. csv" , header=FALSE, Seq=";")

df=read.csv(" <FileName>. csv" , header=FALSE)

csv: Contents of a CSV file محتويات لملف

data=read.csv(" <d:/mydir/myfile. csv" ,header=TRUE ,Seq=";")

The header indicates whether the first line of data in the file contains labels or not. If header=TRUE, it means that the first row contains data for variable names. The **seq** option specifies how to separate data units, and for the following formats:

- Commas : sep=" , " الفارزة
- Tab : seq=" /t " Tab اسم المفتاح
- Space : seq " " الفراغ

  data=read.csv("e:/nn.csv",header = TRUE)

- **Reading data from txt text files**

This is the second most common method for reading data from txt text files. Since fields are separated by spaces or periods, the R program provides the read.table() function, which allows reading text files and gives complete control over how the data is read. For example, we can be told that the first line represents the column

names and that the fields are separated by spaces using the Tab key, not Spacebars. txt contents:

Example :

```
x=c(1,2,3,4,5)
 y=c(6,7,8,9,10)
 z=(a,b,c,d,f)
 z=c(a,b,c,d,f)
 z=c("a","b","c","d","f")
 ss=data.frame(x,y,z)
 ss
```

	x	y	z
1	1	6	a
2	2	7	b
3	3	8	c
4	4	9	d
5	5	10	f

```
 df=read.table("ss.txt",header = TRUE)
```

We observe from using this function that the data from a file becomes a data.frame object, and that File.name is not always the filename, but could also be the webpage containing the data. The header specifies whether column names are defined in the data file

- **Importing Excel files into R**

To import Excel files into R, we first need to prepare the workspace by installing the necessary packages, using the following code:

Install package("اسم الحزمة")

After downloading the package, only the following can be activated in the workspace:

Library(اسم الحزمة)

- **Importing data from the internet**

Reading from the internet was also completed using the `read.table` function. The f ollowing is an example:

Example :

```
data <- read.table("https://people.sc.fsu.edu/~jburkardt/data/csv/airtravel.csv",
+                    header = TRUE, sep = ",")
 head(data)
```

	Month	X1958	X1959	X1960
1	JAN	340	360	417
2	FEB	318	342	391
3	MAR	362	406	419
4	APR	348	396	461
5	MAY	363	420	472
6	JUN	435	472	535

- **The graph in R**

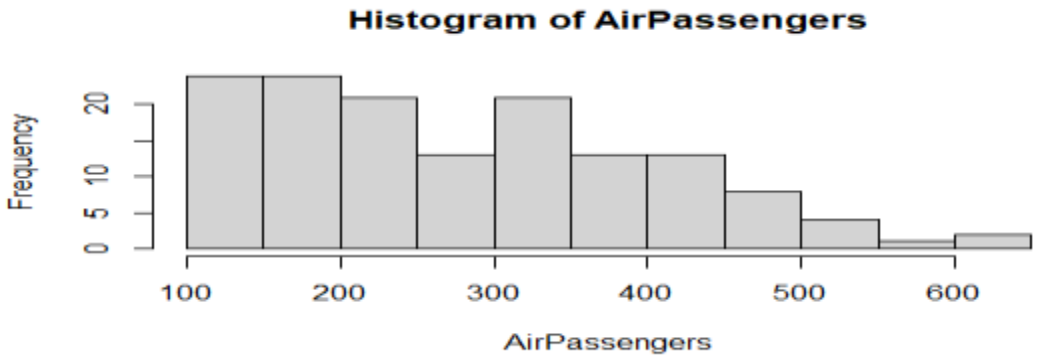
A graph is a visual representation of the distribution and identification of data. A graph consists of the x-axis, the y-axis, and multiple points of different heights. The y-axis shows how often the values on the x-axis occur in the data.

1- hit() Function

A graph of categorical data can be created using the `hit()` function in the R programming language. This function takes a vector of values which are then plotted graphically. The name of the data set is placed between parentheses in this function, as in the following example:

Example :

```
Hist (AirPassengers)    # ركاب الطائرة
```

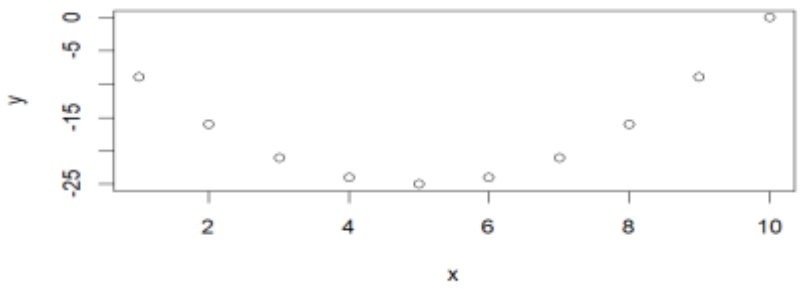


2- The plot() function:

This is the most frequently used function in R. It is a general function because it has many methods that are named according to the object type, as in the following example:

Example :

```
> plot(c(1,2),c(3,5))
> x=seq(1,10)
> y=x**2-10*x
> plot(x,y)
```



3-Subplots

Sometimes we need to place two or more graphs in one place, i.e., divide the drawing space. The R programming language has many graphical parameters that control how the graphs are displayed. The par() function displays a long list of parameters.

```
 par()

$xlog
[1] FALSE
$ylog
[1] FALSE
$adj
[1] 0.5
$ann
[1] TRUE
$ask
```

The graph parameter **mfrow()** can be used to specify the number of subplots we need. c(m,n) divides the results page into a number of graphs, as in the following example:

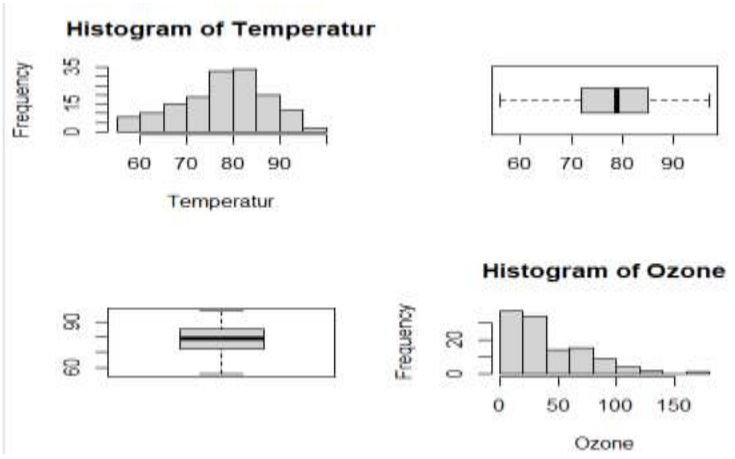
Example :

```
 par(mfrow=c(2,2))
 par(mfrow=c(2,2))  # 2×2 تعيين منطقة الرسم
 max.temp=c(41,44,43,35,40,22)
```

```
max.temp=c(41,44,43,35,40,22,33)
names(max.temp)=c("Sun","Mon","Tue","Wen","Thu","Fri","Sat")
max.temp
```

```
Sun Mon Tue Wen Thu Fri Sat
 41  44  43  35  40  22  33
```

```
barplot( max.temp,main="Barplot")
pie( max.temp,main="Piechart",radius = 1)
```



Example :

```
Temperatur=airquality$Temp
 Ozone=airquality$Ozone
 par(mfcol=c(2,3))
 hist(Temperatur)
 boxplot(Temperatur)
 boxplot(Temperatur,horizontal = TRUE)
>hist(Ozone,horizontal=TRUE)
```

4- Circular diagrams

Circular diagrams display a vector of numbers by dividing a circular disk into segments where the angle (and therefore the area) is proportional to each number.

For example, electronic grades assigned to a category may arise in the ratios, which are plotted using the following instruction:

Example :

```
> groupsize=c(10,30,18,20,20)
> tables=c("A","b","c","d","F")
> pie(groupsize,tables,col = c("grey40","White","grey","black","grey90"))
> pie(groupsize,tables)
```