❖ Lists

A list is a vector with different types of elements, as shown in the following figure:

Example :

```
> x=1:10  # متجه
> y=matrix(1:12,nrow = 3)# matrix
> z="bell"  # character variable
> my.list=list(x,y,z)  #creat the list
> my.list   #   view the list
[[1]]
 [1]  1  2  3  4  5  6  7  8  9 10

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

To name the items in the list, we use the symbol $, as shown below:

Example :

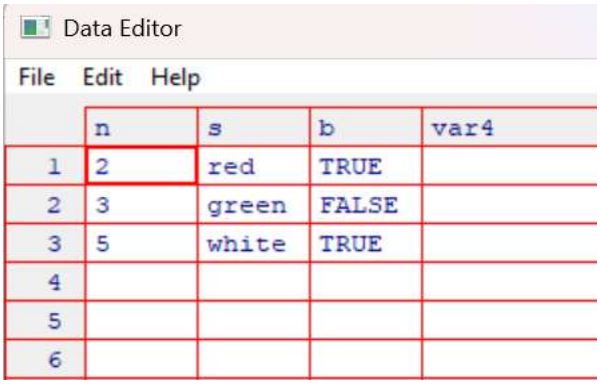
```
> names(my.list)=c("my.vector","my.matrix","my.name")
> $my.vector
> my.list$my.vector
 [1]  1  2  3  4  5  6  7  8  9 10
> my.list$my.matrix
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> my.list$my.name
[1] "bell"
```

- **Data Frames**

A data frame is used to store data tables, it is more general than an array but the col umns can be different between numbers, letters, operators, vectors, etc., as shown i n 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
> fix(df)
```



	n	s	b	var4
1	2	red	TRUE	
2	3	green	FALSE	
3	5	white	TRUE	
4				
5				
6				

We notice that the variable df in the above example contains three vectors (n, s, b) wrapped in the variable df, and the fix function is used to arrange the data frame in the form of a table representing the vectors (n, s, b) and the column headings.

The following function can be substituted to name the variables:

```
names(df)<-c("number","color","passed")   #  اسماء المتغيرات
 df
```

	Number	color	passed
1	2	red	TRUE
2	3	green	FALSE
3	5	white	TRUE

Example :

```
# Create a table containing gene names and the percentage of change
results<- data.frame(geneName = c("gene1","gene2","gene3"),
+                     expression= c(1,4,0.3)  )
```

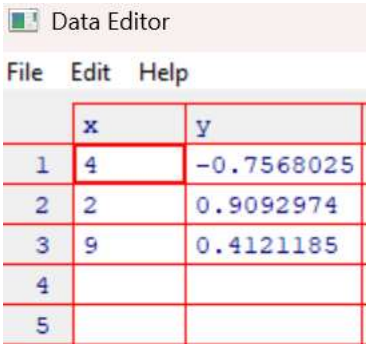
```
 results
  geneName expression
1    gene1        1.0
2    gene2        4.0
3    gene3        0.3
```

If we want to get only the gene names, we write the variable name +$+column name:

```
> results$geneName
[1] "gene1" "gene2" "gene3"
```

Example :

```
> x=c(4,2,9)
> y=sin(x)
> ss=data.frame(x,y)
> ss
     x         y
1    4 -0.7568025
2    2  0.9092974
3    9  0.4121185
> fix(ss)
```



****Save and recall data**

Loading data into R may present some limitations. This requires a specific function for each type to easily and quickly import data into R.

To import data into R, we need the data stored in a file on your computer for example, Excel, SPSS, or any other file type). It must also be available online or available from other sources.

The first step to importing data into R is to prepare the R Workspace. There may still be an environment full of data and values. The ls() command displays the names of all variables, arrays, and functions created in R. You can also delete all objects from a particular environment using rm(list=ls()) as shown below:

```
> ls()
[1] "b"  "df" "n"  "s"
> rm(list = ls())
> ls()
character(0)
```

To save data in R, each object can be stored and retrieved from a file with the Save and Load commands, as in the following example:

Example :

```
X=1:8
save(X,file = "X.Rdata") # للخزن
rm(X)    #للخزن
X
load("X.Rdata") # للتحميل
X
[1] 1 2 3 4 5 6 7 8
```

To import data from the local disk:

```
getwd()  #  معرفة مجلد العمل الحا لي
[1] "C:/Users/User/Documents"
```

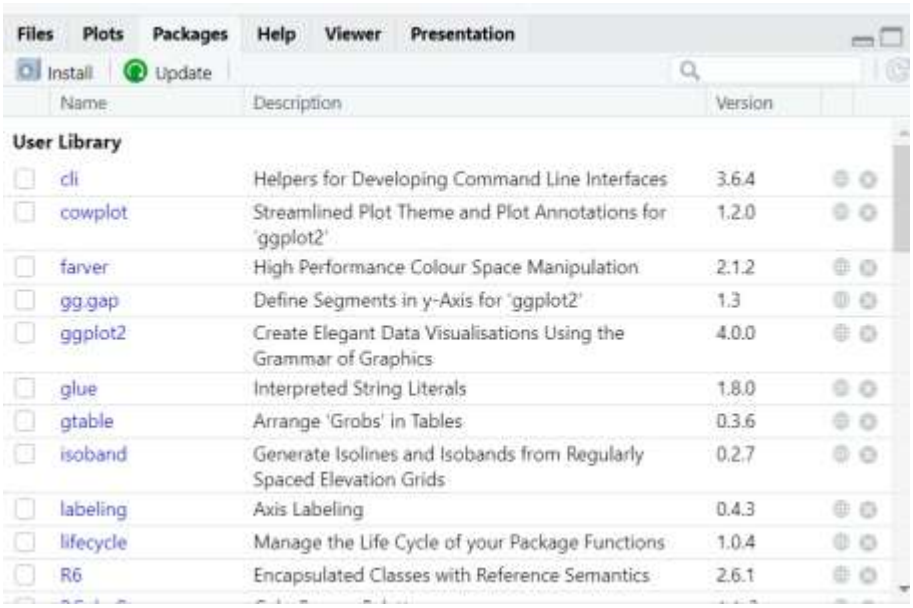
This means that any file you save or retrieve without specifying a path will be auto matically processed within this folder.

You can also change the folder through the RStudio interface without writing code. Select **Session → Set Working Directory → Choose Directory**, then select t he desired folder.

- **Documentation**

The Library() command function helps in performing specific tasks such as s tatistical analysis, reading Excel files, and the effects are varied. The Library() command is also used to benefit from memory after it is installed, so that you ca n use its binary directly (R session), an example of this is :

```
library(class)
library(class)     # تحميل الحزمة
install.packages("ggplot2")     #  تثبيت الحزمة مرة واحدة
ggplot2(mpg, aes(displ, hwy, color = class)) + geom_point#()  استخدام دالة منها
```



Example :

```
data("CO2")#    co2    لتحميل واستخدام البيانات كما في المثال
head(CO2)#  لتحميل البيانات لست صفوف في المقدمة
```

	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

tail(CO2)# co2 لتحميل واستخدام البيانات في النهاية

	Plant	Type	Treatment	conc	uptake
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

tail(CO2,10)# co2 لتحميل واستخدام البيانات لعشر صفوف في النهاية

	Plant	Type	Treatment	conc	uptake
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

Once the data is imported, the values in any of the table columns can be accessed u sing the formula CO2$conc, where conc refers to the column name.

The attach() function is used for variable names when working with data.

```
attach(CO2)
names(CO2)
  [1] "Plant"     "Type"      "Treatment" "conc"      "uptake"
CO2.copy=(CO2)
```