

# Statistics and R

**Workshop / Practical Works**

**Authors: Yahia Lebbah**

**Language: French**

## Table des matières

1	Basic syntax .....	4
1.1	Variables .....	6
1.2	Vecteurs.....	7
1.3	Listes.....	8
1.4	Matrice .....	8
1.5	Data Frame .....	10
1.6	Instructions de flux de contrôle .....	11
1.6.1	L'instruction conditionnelle.....	11
1.6.2	Boucle .....	12
1.7	Fonctions .....	14
1.8	Mise en pratique .....	15
2	Statistique descriptive .....	16
2.1	Cas d'étude : sources gratuites de données.....	16
2.2	Lire et écrire des fichiers de données.....	16
2.2.1	Lire des fichiers CSV.....	16
2.2.2	Ecrire dans un fichier CSV.....	17
2.2.3	Lire un fichier Excel.....	17
2.2.4	Ecrire un fichier Excel .....	18
2.3	Traitements basiques des données.....	18
2.3.1	Sélectionner les données .....	18
2.3.2	Trier .....	19
2.3.3	Filtrer .....	19
2.3.4	Supprimer les valeurs manquantes .....	20
2.3.5	Supprimer les doublons.....	20
2.4	Les fonctions summary() et str() .....	20
2.5	Statistiques de base.....	21
2.5.1	Mode .....	21
2.5.2	Médiane.....	22
2.5.3	Moyenne .....	22
2.5.4	Quartiles .....	22

2.5.5	Distribution normale et test de normalité .....	23
3	Visualisation .....	25
3.1	Bar-Chart et Histogram .....	25
3.2	Line Chart et Pie Chart.....	28
3.3	Scatterplot et Boxplot .....	31
3.4	Marice Scatterplot.....	33
3.5	Utiliser ggplot2 .....	34
3.5.1	Géométrie dans ggplot2.....	36
3.5.2	Etiquettes .....	37
3.5.3	Thèmes .....	37
3.5.4	Charts.....	38
3.5.5	Scatterplot .....	40
3.5.6	Line chart.....	41
3.5.7	Boxplot.....	42
3.5.8	Charts interactifs avec Plotly et ggplot2.....	42
4	Inférence statistique.....	44
4.1	De quoi s'agit-il ? .....	44
4.2	apply(), lapply(), sapply() .....	44
4.3	Echantillonnage .....	45
4.3.1	Echantillonnage aléatoire.....	45
4.3.2	Echantillonnage stratifié.....	45
4.3.3	Echantillonnage par partitionnement .....	45
4.4	Comment échantillonner ?.....	45
4.5	Corrélations .....	48
4.6	Covariance .....	48
4.7	Test d'hypothèses et P-Value .....	49
4.8	T-Test.....	49
4.8.1	T-Test sur la différence de deux variables.....	50
4.8.2	Formes de T-Test .....	51
4.9	Test de Chi-Square.....	51
4.9.1	Fit Test .....	51
4.9.2	Test de contingence .....	52
4.10	ANOVA.....	53
4.10.1	One-Way ANOVA.....	53
4.10.2	Two-Way ANOVA.....	55
4.10.3	MANOVA .....	55
4.11	Test non-paramétrique .....	56
4.11.1	Wilcoxon.....	56

4.11.2	Test de Wilcoxon-Mann-Whitney .....	57
4.11.3	Formule Wilcoxon .....	58
4.11.4	Kruskal-Walis .....	58
5	Modèles statistiques .....	59
5.1	Régressions.....	59
5.1.1	Régression linéaire .....	59
5.1.2	Régression multiples .....	60
6	Cas d'études .....	62
6.1	Cas : test d'hypothèse sur la moyenne .....	62
6.2	Cas : test d'hypothèse sur des données gaussiennes .....	62
6.3	Cas : test d'hypothèse sur deux échantillons appariés .....	63
6.4	Cas : test d'hypothèse sur deux échantillons appariés .....	63
6.5	Cas : .....	64
6.6	Cas .....	64
6.7	Cas .....	64
6.8	Cas : quartiles .....	64
6.9	Cas : Test d'indépendance.....	65
6.10	Cas : Test d'indépendance.....	65
6.11	Cas : analyse de plusieurs moyennes .....	66
6.12	Cas : analyse de plusieurs moyennes sur deux catégories.....	67
6.13	Cas : Wilcoxon .....	68

## 1 Basic syntaxe

```
> 58+7  
[1] 65
```

```
> 1/3  
[1] 0.3333333
```

Pour calculer l'écart type d'un ensemble de données :

```
> sd(c(1, 2, 3, 4, 5, 6))  
[1] 1.870829
```

La moyenne :

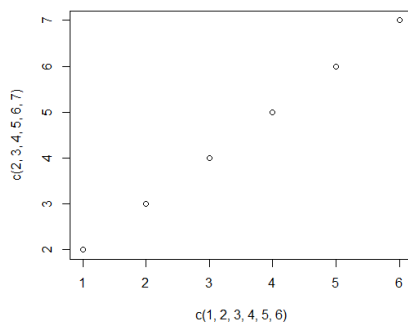
```
> mean(c(1, 2, 3, 4, 5, 6))  
mean(c(1, 2, 3, 4, 5, 6))
```

Le Min :

```
> min(c(1, 2, 3, 4, 5, 6))  
[1] 1
```

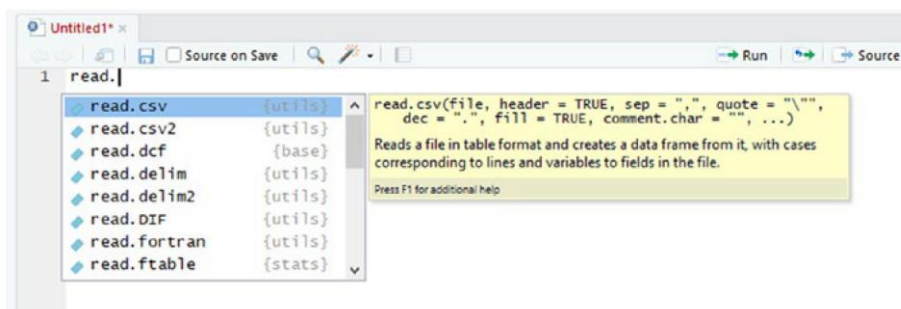
Pour dessiner une courbe en 2D point par point :

```
> plot(c(1, 2, 3, 4, 5, 6), c(2, 3, 4, 5, 6, 7))
```

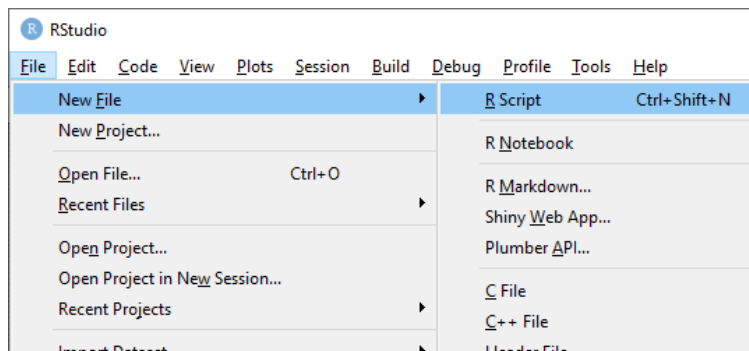


En somme, R, avec sa syntaxe basique, permet :

- Un prototypage rapide pour tester et mettre à l'épreuve rapidement vos traitements de données
- Une bonne performance de vos traitements sur des données
- Editeur intelligent pour vous accompagner dans l'écriture de votre code R, notamment via la mise en couleurs des différentes parties de votre code, ainsi que des suggestions de complétion de votre code R.



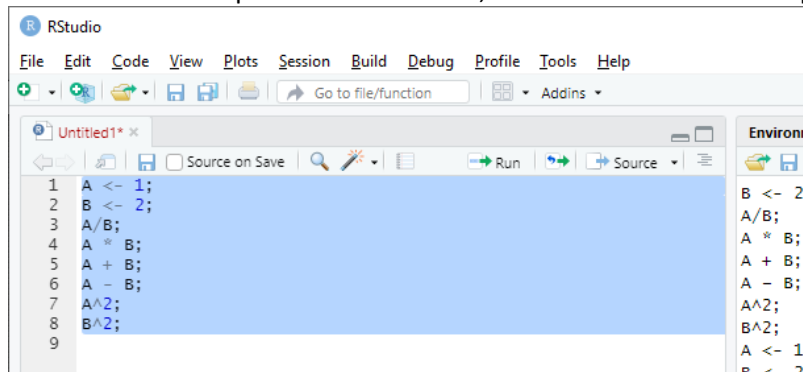
Pour créer un nouveau script, il suffit d'invoquer le menu comme suit :



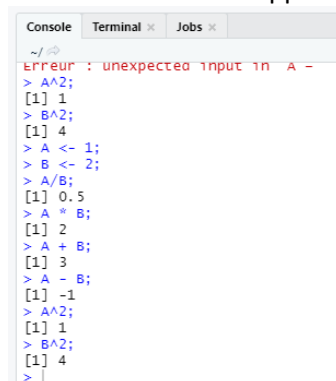
Puis reprendre les commandes R :

```
A <- 1;
B <- 2;
A/B;
A * B;
A + B;
A - B;
A^2;
B^2;
```

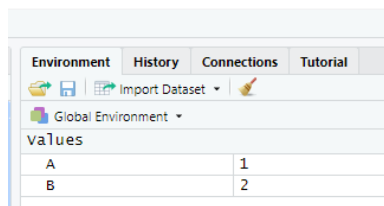
Pour exécuter une partie du code saisi, il suffit de la sélectionner puis cliquer sur Run :



L'exécution du code apparaît en bas de la fenêtre :



Dans le volet « Environnement », vous accéder à l'état des variables de votre programme :



Les commentaires sont introduits en commençant par le symbole # :

```
#Create variable A with value 1
A <- 1;
#Create variable B with value 2
B <- 2;
#Calculate A divide B
A/B;
#Calculate A times B
A * B;
#Calculate A plus B
A + B;
#Calculate A subtract B
A - B;
#Calculate A to power of 2
A^2;
#Calculate B to power of 2
B^2;
```

## 1.1 Variables

Les variables sont déclarées directement sans se soucier de leurs types. Par exemple :

```
A <- 1
B <- 2
```

déclare deux variables A et B, en leur affectant des valeurs respectives 1 et 2.

R donne accès à plusieurs types de variables notamment : logiques, numériques et chaînes de caractères :

Data Types	Values
Logical	TRUE
	FALSE
Numeric	12.3
	2.55
	1.0
Character	"a"
	"abc"
	"this is a bat"

Voir le lien [http://www.tutorialspoint.com/r/r\\_data\\_types.htm](http://www.tutorialspoint.com/r/r_data_types.htm)

Vous pouvez afficher le type de vos variables :

```
A <- "ABC";
print(class(A));
```

```
A <- 123;  
print(class(A));
```

```
A <- TRUE;  
print(class(A));
```

Soit :

```
A <- 123;  
B <- "aaa";  
A + B;
```

Vous renvoie :

```
Error in A + B : argument non numérique pour un opérateur binaire
```

Vous pouvez vérifier le bon typage de vos variables :

```
A <- 123;  
print(is.numeric(A));  
print(is.character(A));
```

Vous pouvez aussi convertir vos variables :

```
A <- 12;  
B <- "56";  
B <- as.numeric(B);  
A + B;
```

## 1.2 Vecteurs

Les vecteurs sont créés via l'opérateur c :

```
A <- c(1, 2, 3, 4, 5, 6);  
print(A);
```

Vous pouvez vérifier le type de votre vecteur via les deux opérateurs :

```
typeof(A);  
class(A);
```

La longueur d'un vecteur :

```
length(A);
```

La création d'un vecteur de taille 8 :

```
A <- 1:8;  
print(A);
```

L'accès à un élément du vecteur

```
A[2];
```

On peut extraire des éléments d'un vecteur. Par exemple le retrait des éléments de rang 2 et 5 :

```
A <- 1:8;  
print(A);  
A[c(2, 5)];
```

Pour retirer tous les éléments sauf le deuxième :

```
A <- 1:8;  
print(A);  
A[-2];
```

Pour retirer les éléments suivant votre choix élément par élément :

```
A <- 1:8;
print(A);
A[c(FALSE, TRUE, FALSE, TRUE, FALSE, TRUE, FALSE, TRUE)];
```

Pour retirer les éléments suivant une condition :

```
A <- 1:8;
print(A);
A[A > 5];
```

Vous pouvez changer tout élément de votre tableau via l'opérateur d'affectation <- :

```
A <- 1:8;
print(A);
A[3] <- 9;
print(A);
```

### 1.3 Listes

Les listes ont le même statut que les vecteurs, à l'exception qu'une liste peut abriter des valeurs de types différents :

```
A <- list("a", "b", 1, 2);
print(A);
```

Pour récupérer le type de chaque élément de votre liste :

```
A <- list("a", "b", 1, 2);
print(A);
str(A);
```

Pour supprimer un élément, il suffit de lui affecter la valeur nulle :

```
A <- list("a", "b", 1, 2);
A[[2]] <- NULL;
print(A);
str(A);
```

### 1.4 Matrice

La matrice est de dimension 2, alors que les vecteurs sont de dimension 1. Les matrices servent à abriter uniquement des valeurs numériques. Pour manipuler des matrices avec des types différents, utiliser le type Data-Frame.

La syntaxe de création d'une matrice :

```
variable <- matrix(vector, nrow=n, ncol=i)
```

Voici différentes syntaxes pour créer des matrices :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3);
print(A);
```

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));
print(A);
```

Le type :



```
class(A);
```

Les attributs d'une matrice :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,  
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));  
print(A);  
attributes(A);
```

On peut obtenir les noms des colonnes et des lignes comme suit :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,  
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));  
print(A);  
colnames(A);  
rownames(A);
```

On peut créer une matrice par colonnes ou par lignes :

```
B <- cbind(c(1, 2, 3), c(4, 5, 6));  
print(B);
```

```
C <- rbind(c(1, 2, 3), c(4, 5, 6));  
print(C);
```

Pour sélectionner la première ligne :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,  
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));  
print(A)  
A[1,];
```

... la première colonne :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,  
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));  
print(A)  
A[,1];
```

Pour sélectionner toutes les lignes sauf la dernière :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,  
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));  
print(A)  
A[-3,];
```

Pour sélectionner une cellule particulière :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,  
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));  
print(A)  
A[2, 2];
```

Pour utiliser des valeurs booléennes pour sélectionner les lignes :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,  
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));  
print(A)  
A[c(TRUE, FALSE, FALSE),];
```

Pour ajouter une ligne, utiliser la fonction `rbind()` :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));
print(A)
B <- rbind(A, c(1, 2, 3));
print(B);
```

... une colonne, `cbind()` :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));
print(A)
C <- cbind(A, c(1, 2, 3));
print(C);
```

Pour transposer une matrice :

```
A <- matrix(c(1, 2, 3, 4, 6, 7, 8, 9, 0), nrow=3, ncol=3,
            dimnames=list(c("X", "Y", "Z"), c("A", "S", "D")));
print(A)
A <- t(A);
print(A);
```

## 1.5 Data Frame

Un Data Frame est une sorte de liste en deux dimensions.

Il est créé via la syntaxe :

```
variable = data.frame(colName1 = c(..., ..., ...), colName2 = c(..., ..., ...), ...)
```

Exemple :

```
A <- data.frame(
  emp_id=c(1, 2, 3),
  names=c("Ali", "Safi", "Chafi"),
  salary=c(111.1, 222.2, 333.3));
print(A);
```

Les dimensions d'un data frame :

```
A <- data.frame(
  emp_id=c(1, 2, 3),
  names=c("Ali", "Safi", "Chafi"),
  salary=c(111.1, 222.2, 333.3));
ncol(A);
nrow(A)
```

La structure d'un data frame :

```
A <- data.frame(
  emp_id=c(1, 2, 3),
  names=c("Ali", "Safi", "Chafi"),
  salary=c(111.1, 222.2, 333.3));
str(A);
```

Vous pouvez lire un data frame à partir d'un fichier CSV via la fonction `read.csv()` :

```
myData<- read.csv(file="D:/data.csv", header=TRUE, sep=",");
ncol(myData);
```

Pour récupérer des données de différentes façons :

```
A <- data.frame(
  emp_id=c(1, 2, 3),
  names=c("Ali", "Safi", "Chafi"),
  salary=c(111.1, 222.2, 333.3));
A["names"];
A$names;
A[[2]];
```

Pour modifier une cellule :

```
A <- data.frame(
  emp_id=c(1, 2, 3),
  names=c("Ali", "Safi", "Chafi"),
  salary=c(111.1, 222.2, 333.3));
print(A);
A[2, 2] <- "Ali";
print(A);
```

En ajoute une colonne avec rbind() :

```
A <- data.frame(
  emp_id=c(1, 2, 3),
  names=c("Ali", "Safi", "Chafi"),
  salary=c(111.1, 222.2, 333.3));
print(A);
B <- rbind(A, list(4, "Omar", 444.4));
print(B);
```

... une colonne avec cbind() :

```
A <- data.frame(
  emp_id=c(1, 2, 3),
  names=c("Ali", "Safi", "Chafi"),
  salary=c(111.1, 222.2, 333.3));
print(A);
B <- cbind(A, state=c("AL", "OR", "CO"));
print(B);
```

Pour supprimer une colonne :

```
A <- data.frame(
  emp_id=c(1, 2, 3),
  names=c("Ali", "Safi", "Chafi"),
  salary=c(111.1, 222.2, 333.3));
print(A);
A$salary<- NULL;
print(A);
```

## 1.6 Instructions de flux de contrôle

### 1.6.1 L'instruction conditionnelle

Pour exprimer le choix entre deux traitements suivant une condition, on fait appel à l'instruction :

```
if (Boolean expression) {
  #Codes to execute if Boolean expression is true
}
```

```
else {  
  #code to execute if Boolean expression is false }
```

Les conditions utilisables sont :

Boolean Operator	Definition
==	Equal to
>=	Greater than or equal to
<=	Lesser than or equal to
>	Greater than
<	Lesser than
!=	Not equal to

Soit l'exemple :

```
A <- c(1, 2)  
B <- sum(A) #1 + 2  
if (B >= 5) {  
  print("B is greater or equal to 5")  
} else if (B >= 3) {  
  print("B is more than or equal to 3")  
} else {  
  print("B is less than 3")  
}
```

qui affichera certainement

```
[1] "B is more than or equal to 3"
```

### 1.6.2 Boucle

La boucle sert à exprimer les traitements répétitifs. Ci-dessous les différentes formes pour l'exprimer.

Forme 1 :

```
for (value in vector) {  
  #statements  
}
```

Forme 2 :

```
While (Boolean Expression) {  
  #Code to run or repeat until Boolean Expression is false  
}
```

Forme 3 :

```
repeat {  
  #code to repeat  
}
```

Exemple :

```
A <- c(1:5); #create a vector with values 1, 2, 3, 4, 5  
for(e in A) { #for each element and value in vector A  
  print(e); #print the element and value  
}
```

Exemple :

```
i<- 1;
while(i<= 10) {
  print(i);
  i<- i+1;
}
```

Exemple :

```
A <- c(1:10);
for(e in A) {
  if(e == 5) {
    break;
  }
  print(e);
}
```

## 1.7 Fonctions

Pour nommer un traitement opérant sur des arguments donnés, on l'exprime dans une fonction comme suit :

```
function_name<- function(arg1, arg2, ...) {  
  # Codes fragments  
  function_name = #value to return  
}
```

Example:

```
A <- c(1:5);  
productVect <- function(a) {  
  res <- 1;  
  for(e in a) {  
    res <- res * e;  
  }  
  productVect = res;  
}  
print(productVect(A));
```

## 1.8 Mise en pratique

Petit programme pour concevoir une calculatrice ...

```
add <- function(a, b) {
  res <- a + b;
  return(res);
}
subtract <- function(a, b) {
  res <- a - b;
  return(res);
}
product <- function(a, b) {
  res <- a * b;
  return(res);
}
division <- function(a, b) {
  res <- a / b;
  return(res);
}
print("Select your option: ");
print("1. Add");
print("2. Subtract");
print("3. Product");
print("4. Division");
opt <- as.integer(readline(prompt = "> "));
firstNum<- as.integer(readline(prompt="Enter first number: "));
secondNum<- as.integer(readline(prompt="Enter second number:
"));
res <- 0;
if(opt == 1) {
  res <- add(firstNum, secondNum);
} else if(opt == 2) {
  res <- subtract(firstNum, secondNum);
} else if(opt == 3) {
  res <- product(firstNum, secondNum);
} else if(opt == 4) {
  res <- division(firstNum, secondNum);
} else {
  print("Error. ");
}
print(res);
```

## 2 Statistique descriptive

### 2.1 Cas d'étude : sources gratuites de données

<https://www.kaggle.com/>

<https://r-dir.com/reference/datasets.html>

<https://quickstats.nass.usda.gov/>

<https://cloud.csiss.gmu.edu/Crop-CASMA/>

<https://registry.opendata.aws/tag/agriculture/>

### 2.2 Lire et écrire des fichiers de données

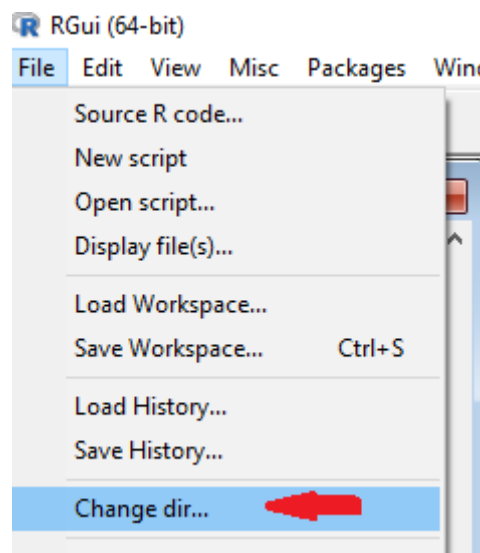
Dans un premier temps, il est nécessaire de pointer sur le répertoire de travail. Pour afficher le répertoire courant de travail, on peut faire appel à :

```
print(getwd());
```

Pour changer de répertoire de travail :

```
#set the current workspace location  
setwd("C:/Ateliers CTS");
```

On peut aussi passer par le menu en configurant l'option « Change dir.. »



#### 2.2.1 Lire des fichiers CSV

Soit un fichier CSV contenant les données suivantes sauvegardées dans un fichier "data.csv" :

```
x,      x2,   x3,   y  
2.21,   3.5,  6.5,  3.3  
1,      2,    3,    4  
5,      6,    7,    8
```

Vous pouvez lire le fichier CSV de données dans la console R à l'aide de read.csv ()

```
data <- read.csv(file="data.csv", header=TRUE, sep=",");
```

Vous pouvez afficher les données du fichier .CSV en utilisant

```
View(data);
```



	x	x2	x3	y
1	2.21624472	4.774511945	-4.87198610	0
2	-0.18104835	4.100479091	6.97727175	1
3	1.69712196	2.328894837	3.92445970	0
4	1.65499099	2.462167830	0.74972168	0
5	1.06797834	1.053091767	3.35380788	1
6	0.67543296	1.918655276	1.56826805	1
7	0.19982505	3.063870668	4.48912276	1
8	0.91662531	1.953422065	3.29408509	1
9	1.30843083	1.322800550	0.39717775	1
10	-0.12830745	3.929044754	3.63913066	1
11	1.39507566	0.270269185	0.26466993	0
12	2.21668825	1.611527264	3.69688978	0
13	2.64020481	4.240357413	3.74926815	0
14	-0.60394410	1.130285226	5.49485937	0
15	0.49529219	0.961328129	2.72051420	1
16	1.91349092	2.348240992	2.21675086	0
17	1.33149648	4.189856264	7.09660419	1
18	1.42607352	2.374329561	5.94312583	0
19	2.93044162	1.913885092	2.27876092	0

### 2.2.2 Ecrire dans un fichier CSV

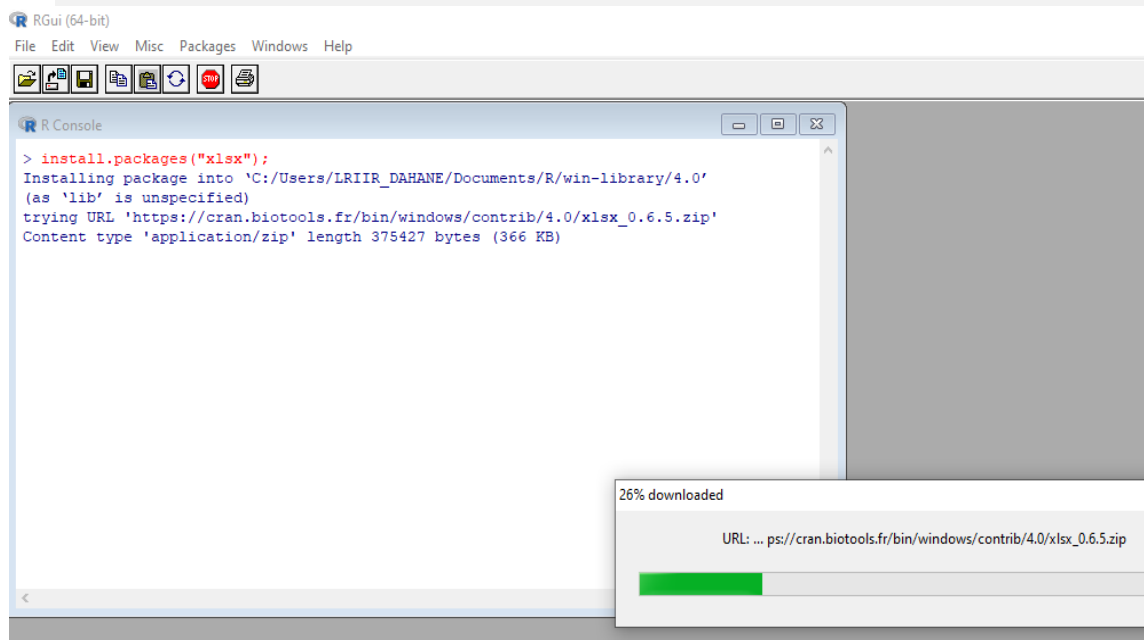
Pour écrire un fichier CSV, vous pouvez utiliser la fonction `write.csv()`

```
write.csv(data, file="data2.csv", quote=TRUE, na="na", row.names=FALSE);
```

### 2.2.3 Lire un fichier Excel

L'ensemble de données peut également être au format Excel ou au format .xlsx. Pour lire un fichier Excel, vous devez utiliser le package `xlsx`. Le package `xlsx` nécessite un runtime Java, vous devez donc l'installer sur votre ordinateur...

```
install.packages("xlsx");
```

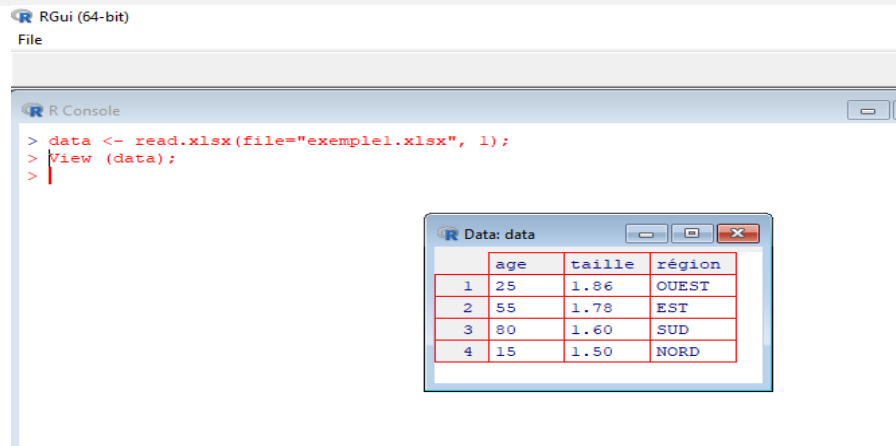


La commande suivante permet d'activer le package

```
require("xlsx");
```

Pour lire un fichier Excel on utilise la commande suivante :

```
data <- read.xlsx(file="exemple1.xlsx", 1);
```



## 2.2.4 Ecrire un fichier Excel

Pour écrire dans un fichier Excel on utilise la commande suivante :

```
write.xlsx(data, file="data2.xlsx", sheetName="sheet1", col.names=TRUE,  
row.names=FALSE);
```

...

## 2.3 Traitements basiques des données

### 2.3.1 Sélectionner les données

Vous pouvez sélectionner quelques colonnes des données à l'aide d'un vecteur :

```
age,      taille,   région  
25,      1.86,     OUEST  
55,      1.78,     EST  
80,      1.60,     SUD  
15,      1.50,     NORD
```

```
data;
```

```
> data;  
  age taille région  
1  25  1.86  OUEST  
2  55  1.78   EST  
3  80  1.60   SUD  
4  15  1.50  NORD
```

```
data[, c("age", "taille")];
```

```
> data[, c("age", "taille")];
```

	age	taille
1	25	1.86
2	55	1.78
3	80	1.60
4	15	1.50

### 2.3.2 Trier

Vous pouvez trier les données en Age dans l'ordre croissant en utilisant :

```
> data;
```

	age	taille	région
1	25	1.86	OUEST
2	55	1.78	EST
3	80	1.60	SUD
4	15	1.50	NORD

```
data[order(data$age),];
```

```
> data[order(data$age),];
```

	age	taille	région
4	15	1.50	NORD
1	25	1.86	OUEST
2	55	1.78	EST
3	80	1.60	SUD

Vous pouvez trier les données en age par ordre décroissant en utilisant

```
data[order(data$age, decreasing=TRUE),];
```

```
> data[order(data$age, decreasing=TRUE),];
```

	age	taille	région
3	80	1.60	SUD
2	55	1.78	EST
1	25	1.86	OUEST
4	15	1.50	NORD

Vous pouvez trier les données selon plusieurs variables :

```
data[order(data$age, data$taille),];
```

```
> data[order(data$age, data$taille),];
```

	age	taille	région
4	15	1.50	NORD
1	25	1.86	OUEST
2	55	1.78	EST
3	80	1.60	SUD

### 2.3.3 Filtrer

Vous pouvez filtrer les données à l'aide d'expressions et d'instructions booléennes :

```
data[data$age > 25,];
```

```
> data[data$age > 25, ];
  age taille région
2  55   1.78   EST
3  80   1.60   SUD
```

Vous pouvez également filtrer les données avec des expressions plus complexes :

```
data[data$age > 25 & data$taille < 1.7, ];
> data[data$age > 25 & data$taille < 1.7, ];
  age taille région
3  80   1.6   SUD
```

### 2.3.4 Supprimer les valeurs manquantes

Vous pouvez supprimer des lignes avec des valeurs NA dans n'importe quelle variable :

```
na.omit(data);
> data <- read.xlsx (file="exemple3.xlsx", 1);
> data;
  age taille région
1  25   1.86 OUEST
2  55   1.78   EST
3  80   1.60   SUD
4  15   1.50  NORD
5  NA   1.90   EST
> na.omit(data);
  age taille région
1  25   1.86 OUEST
2  55   1.78   EST
3  80   1.60   SUD
4  15   1.50  NORD
```

### 2.3.5 Supprimer les doublons

Vous pouvez supprimer les doublons en fonction de la variable age en utilisant :

```
> data <- read.xlsx (file="exemple4.xlsx", 1);
> data;
  age taille région
1  25   1.86 OUEST
2  55   1.78   EST
3  80   1.60   SUD
4  15   1.50  NORD
5  19   1.90   EST
6  80   1.60   SUD
data[!duplicated(data$age), ];
> data[!duplicated(data$age), ];
  age taille région
1  25   1.86 OUEST
2  55   1.78   EST
3  80   1.60   SUD
4  15   1.50  NORD
5  19   1.90   EST
```

## 2.4 Les fonctions summary() et str()

Les fonctions summary() et str() sont les moyens les plus rapides d'obtenir des statistiques descriptives des données. La fonction summary() donne les statistiques descriptives de base des données. La fonction str() donne la structure des variables.

Vous pouvez obtenir les statistiques descriptives de base en utilisant la fonction summary() :

```
> summary(data);
```

INFARCT	CO	TABAC	AGE	POIDS	TAILLE	ATCD
Min. :0.0000	Min. :0.0000	Min. :0.0000	Min. : 15.00	Min. : 33.00	Min. :138.0	Length:449
1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.: 33.00	1st Qu.: 52.00	1st Qu.:160.0	Class :character
Median :0.0000	Median :0.0000	Median :1.0000	Median : 44.00	Median : 64.00	Median :166.0	Mode :character
Mean :0.3318	Mean :0.4454	Mean :0.7416	Mean : 45.62	Mean : 66.09	Mean :165.2	
3rd Qu.:1.0000	3rd Qu.:1.0000	3rd Qu.:1.0000	3rd Qu.: 56.00	3rd Qu.: 78.00	3rd Qu.:171.0	
Max. :1.0000	Max. :1.0000	Max. :2.0000	Max. :100.00	Max. :128.00	Max. :184.0	

```
HTA
Min. :0.0000
1st Qu.:0.0000
Median :0.0000
Mean :0.3541
3rd Qu.:1.0000
Max. :1.0000
```

Vous pouvez obtenir la structure des données en utilisant la fonction str() :

```
> str(data)
'data.frame': 449 obs. of 8 variables:
 $ INFARCT: num 0 0 0 0 0 0 0 0 0 0 ...
 $ CO : num 0 0 0 0 0 0 0 0 0 0 ...
 $ TABAC : num 0 0 0 0 0 0 0 0 0 0 ...
 $ AGE : num 47 17 35 82 50 31 60 30 44 38 ...
 $ POIDS : num 48 52 53 78 52 47 60 75 68 52 ...
```

## 2.5 Statistiques de base

Soient les données d'un fichier csv :

x,	x2,	x3,	y
2.21624472,	4.774511945,	-4.87198610,	0
-0.18104835,	4.100479091,	6.97727175,	1
1.69712196,	2.328894837,	3.92445970,	0
1.65499099,	2.462167830,	0.74972168,	0
1.06797834,	1.053091767,	3.35380788,	1
0.67543296,	1.918655276,	1.56826805,	1
0.19982505,	3.063870668,	4.48912276,	1
0.91662531,	1.953422065,	3.29408509,	1
1.30843083,	1.322800550,	0.39717775,	1

Charger les données :

```
data <- read.csv(file="data.csv", header=TRUE, sep=",");
data
```

### 2.5.1 Mode

Le mode est une valeur dans les données qui a la fréquence la plus élevée et est utile lorsque les différences ne sont pas numériques et se produisent rarement.

Pour obtenir le mode dans R, vous commencez avec des données :

```
> A <- c(1, 2, 3, 4, 5, 5, 5, 6, 7, 8);
```

Pour obtenir le mode dans un vecteur, vous créez une table de fréquences :

Vous voulez obtenir la fréquence la plus élevée, vous utilisez donc les éléments suivants pour obtenir le mode :

```
names(y)[which(y==max(y))];
> A <- c(1, 2, 3, 4, 5, 5, 5, 6, 7, 8);
> y <- table(A);
> y;
A
1 2 3 4 5 6 7 8
1 1 1 1 3 1 1 1
> names(y)[which(y==max(y))];
[1] "5"
```

Obtenons le mode pour un ensemble de données. Tout d'abord, vous avez les données :

```

> data <- read.xlsx (file="exemple1.xlsx", 1);
> data;
  age taille région
1  25   1.86  OUEST
2  55   1.78   EST
3  80   1.60   SUD
4  15   1.50  NORD
> y <- table(data$age);
> print(y);

15 25 55 80
 1  1  1  1

```

Pour obtenir le mode dans l'ensemble de données, vous devez obtenir les valeurs de la fréquence la plus élevée. Puisque toutes les valeurs sont de fréquence 1, votre mode est :

```

> data <- read.xlsx (file="exemple1.xlsx", 1);
> data;
  age taille région
1  25   1.86  OUEST
2  55   1.78   EST
3  80   1.60   SUD
4  15   1.50  NORD
> y <- table(data$age);
> print(y);

15 25 55 80
 1  1  1  1
> names(y) [which(y==max(y))];
[1] "15" "25" "55" "80"

```

## 2.5.2 Médiane

Soit la donnée :

```
A <- c(1, 2, 3, 4, 5, 5, 5, 6, 7, 8);
```

La médiane est calculée :

```
median(A);
```

## 2.5.3 Moyenne

```
mean(A);
```

## 2.5.4 Quartiles

Intervalle des valeurs :

```
range(A);
```

Largeur de l'intervalle des valeurs :

```
res <- range(A);
diff(res);
```

min et max des valeurs :

```
min(A);
max(A);
```

Intervalle interquartile :

```
IQR(A);
```

Les quantiles :

```
quantile(A);
```

Quantiles à 25 et 75 :

```
quantile(A, 0.25);
```

Variance :

```
var(A)
```

Ecart type :

```
sqrt(variance);
```

### 2.5.5 Distribution normale et test de normalité

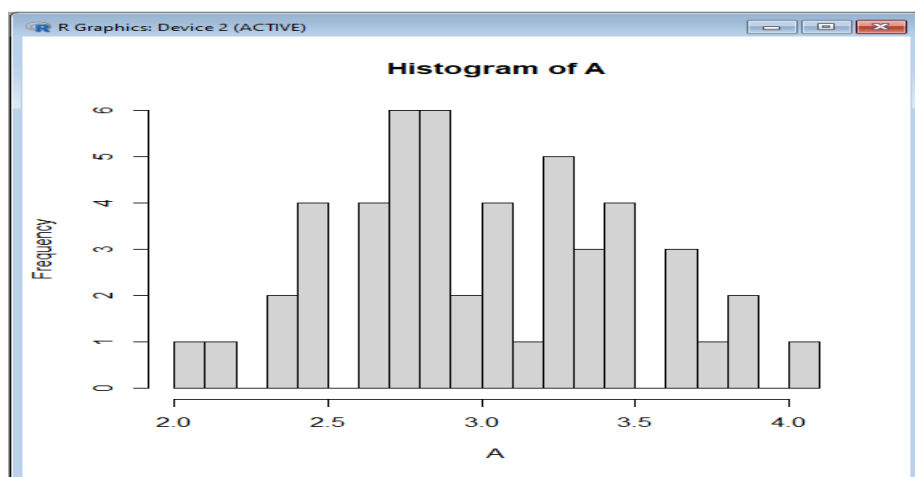
La distribution normale est l'une des théories les plus importantes car presque tous les tests statistiques nécessitent que les données soient distribuées normalement. Il décrit à quoi ressemblent les données lorsqu'elles sont tracées. La distribution normale est aussi appelée courbe en cloche

Dans R, pour générer des nombres aléatoires à partir de la distribution normale, vous utilisez la fonction `rnorm()` :

```
set.seed(123);  
A <- rnorm(50, 3, 0.5);  
hist(A, breaks=15)
```

```
> set.seed(123);  
> A <- rnorm(50, 3, 0.5);  
> hist(A, breaks=15);
```

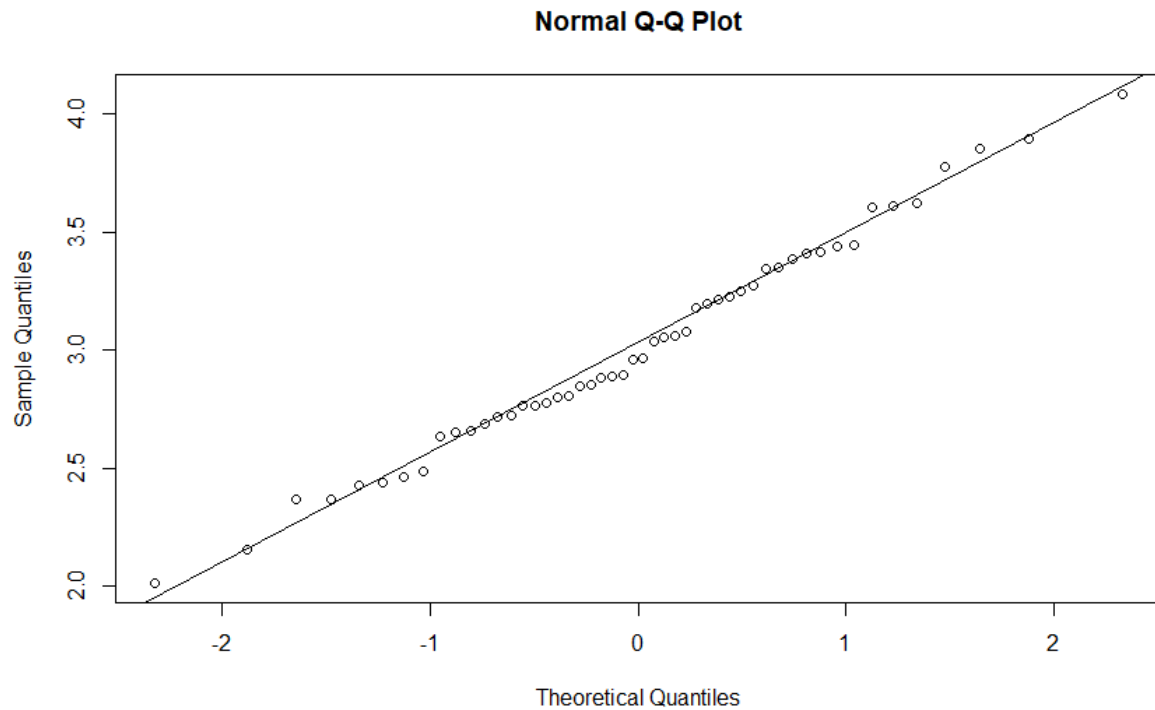
3 est la moyenne et 0,5 est l'écart type. Dans les fonctions ci-dessus, vous avez généré 50 valeurs aléatoires à partir de la distribution normale



Pour voir si les données sont normalement distribuées, vous pouvez utiliser le `qqnorm()` et les fonctions `qqline()` :

```
qqnorm(A);  
qqline(A);
```

Si les points ne s'écartent pas de la ligne, les données sont normalement distribuées :



Vous pouvez également utiliser un test de Shapiro pour tester si les données sont normalement distribuées :

```
shapiro.test(A);
```

Si la valeur de  $p$  est supérieure à 0,05, vous pouvez conclure que les données ne s'écartent pas de la distribution normale.



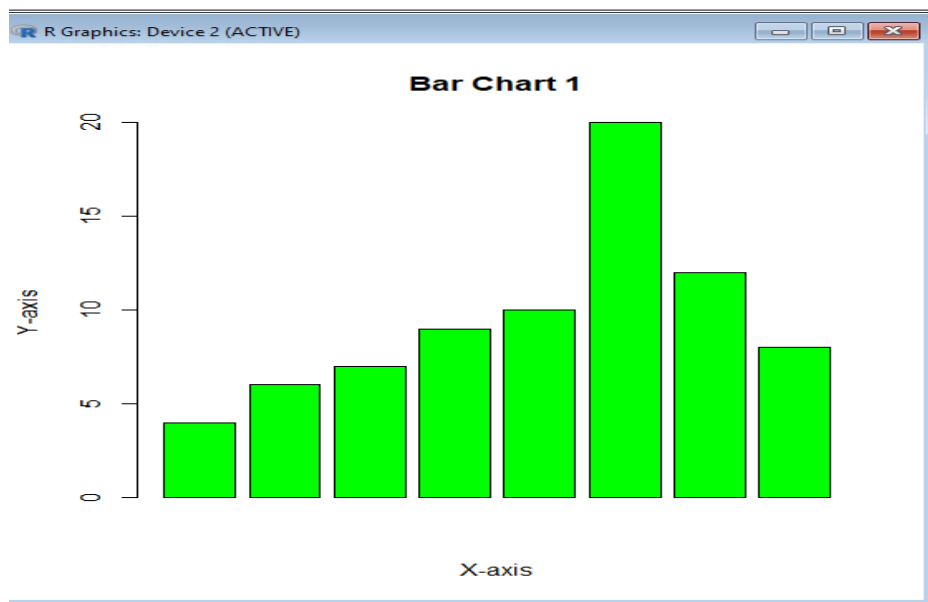
## 3 Visualisation

### 3.1 Bar-Chart et Histogram

La programmation R nous permet de tracer des graphiques à barres et des histogrammes. Un graphique à barres représente les données à l'aide de barres, les valeurs y étant la valeur de la variable. La programmation R utilise la fonction `barplot()` pour créer des graphiques à barres, et R peut dessiner des graphiques à barres horizontaux et verticaux. Un histogramme, en revanche, représente les fréquences des valeurs au sein d'une variable et les dessine en barres.

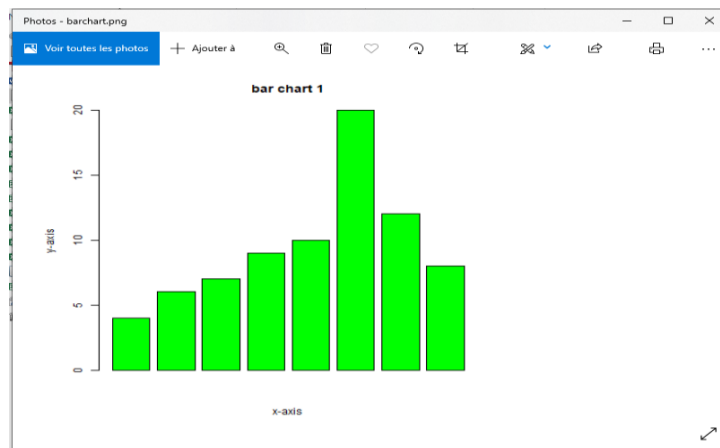
```
data <- c(4, 6, 7, 9, 10, 20, 12, 8);  
barplot(data, xlab="X-axis", ylab="Y-axis", main="Bar chart 1", col="green");  
> data <- c(4, 6, 7, 9, 10, 20, 12, 8);  
> barplot(data, xlab="X-axis", ylab="Y-axis", main="Bar Chart 1",  
+ col="green");
```

data est les données à tracer, xlab est le nom de l'axe des x, ylab est le nom de l'axe des y, main est le titre principal et col est la couleur du graphique



Pour exporter le graphique à barres dans un fichier image, on utilise la commande suivante :

```
> png(file="C:/Ateliers CTS/barchart.png");  
> barplot(data, xlab="x-axis", ylab="y-axis", main="bar chart 1", col="green");  
> dev.off();  
png  
2
```



Soit la table d'exemple de véhicules :

```
mtcars;
```

Pour tracer un graphique à barres empilées, vous pouvez créer l'ensemble de données suivant :

```
data(mtcars);
```

```
data <- table(mtcars$gear, mtcars$carb);
```

```
data;
```

```
> data(mtcars);
```

```
> data <- table(mtcars$gear, mtcars$carb);
```

```
> data;
```

```
1 2 3 4 6 8
3 3 4 3 5 0 0
4 4 4 0 4 0 0
5 0 2 0 1 1 1
```

```
> png(file="C:/Ateliers CTS/barchart1.png");
```

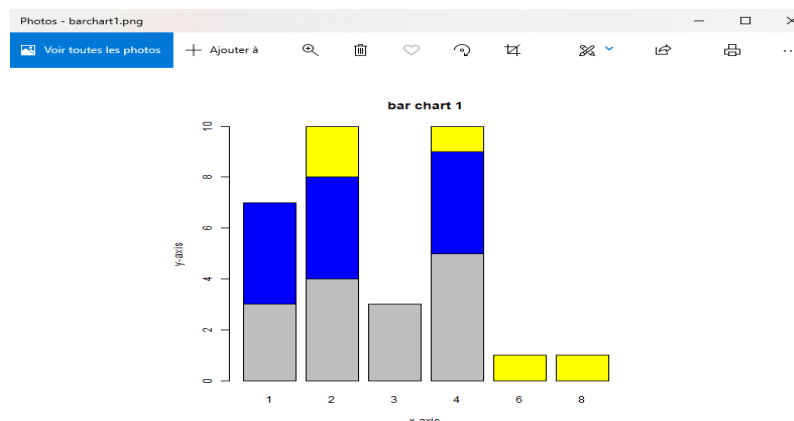
```
png(file="carschart.png");
```

```
barplot(data, xlab="x-axis", ylab="y-axis", main="bar chart 1", col=c("grey", "blue", "yellow"));
dev.off();
```

```
> barplot(data, xlab="x-axis", ylab="y-axis", main="bar chart 1", col=c("grey", "blue", "yellow"));
> dev.off();
```

```
png
```

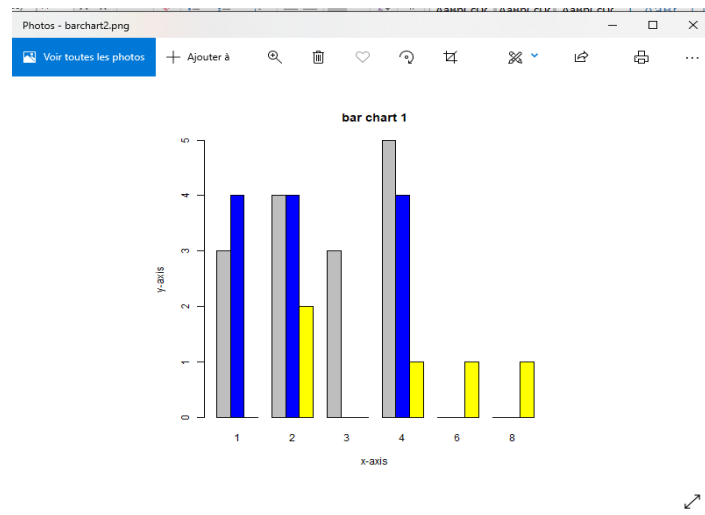
```
2
```



Pour tracer un graphique à barres groupées, vous pouvez utiliser beside=TRUE:

```
png(file="carschart.png");
barplot(data, xlab="x-axis", ylab="y-axis", main="bar chart 1", col=c("grey", "blue", "yellow"),
beside=TRUE);
dev.off();
```

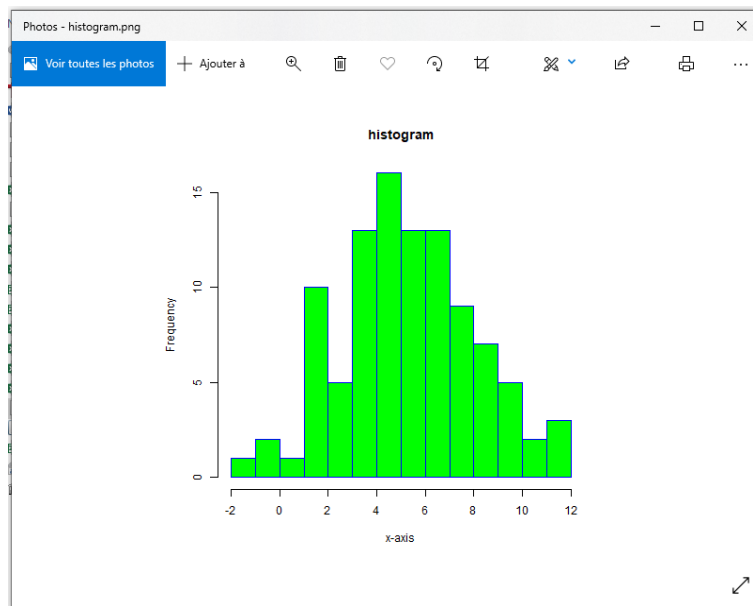
```
> png(file="C:/Ateliers CTS/barchart2.png");
> barplot(data, xlab="x-axis", ylab="y-axis", main="bar chart 1",col=c("grey", "blue", "yellow"), beside=TRUE);
> dev.off();
png
2
```



Pour tracer un histogramme, vous pouvez utiliser la fonction hist() :

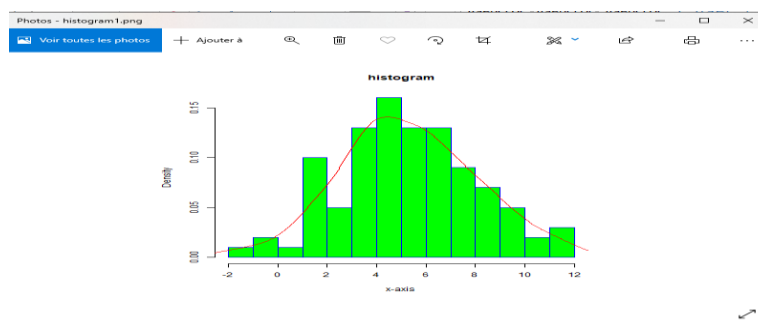
```
set.seed(123);
data1 <- rnorm(100, mean=5, sd=3);
png(file="histogram.png");
hist(data1, main="histogram", xlab="x-axis", col="green",
border="blue", breaks=10);
dev.off();
```

```
> set.seed(123);
> data1 <- rnorm(100, mean=5, sd=3);
> png(file="C:/Ateliers CTS/histogram.png");
> hist(data1, main="histogram", xlab="x-axis", col="green",border="blue", breaks=10);
> dev.off();
png
2
.
```



Pour tracer un histogramme avec une ligne de densité, vous pouvez changer `freq=FALSE` de sorte que l'histogramme soit tracé en fonction de la probabilité, et vous pouvez utiliser la fonction `lines()` pour ajouter la ligne de densité :

```
set.seed(123);
data1 <- rnorm(100, mean=5, sd=3);
hist(data1, main="histogram", xlab="x-axis", col="green",
      border="blue", breaks=10, freq=FALSE);
lines(density(data1), col="red");
> png(file="C:/Ateliers CTS/histogram1.png");
> hist(data1, main="histogram", xlab="x-axis", col="green",border="blue", breaks=10, freq=FALSE);
> lines(density(data1), col="red");
> dev.off();
png
2
.
```



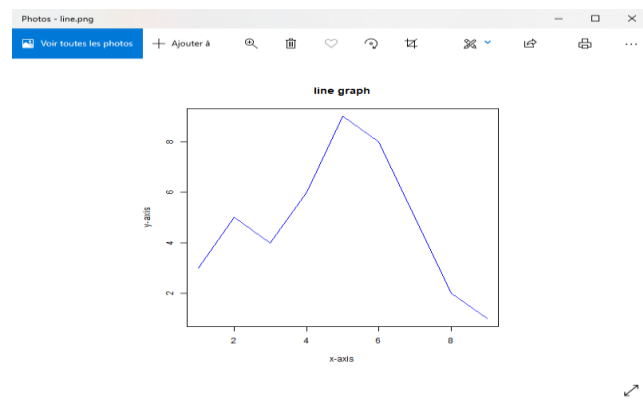
### 3.2 Line Chart et Pie Chart

Un graphique en courbes est un graphique qui a tous les points reliés entre eux en traçant des lignes entre eux. Un graphique en courbes est très utile pour la tendance analyse et analyse de séries chronologiques. Un camembert, d'autre part, est la représentation des données sous forme de tranches d'un cercle avec différentes couleurs.

Vous pouvez tracer un graphique linéaire en utilisant la fonction `plot()` :

```
x <- c(1, 2, 3, 4, 5, 6, 8, 9);
y <- c(3, 5, 4, 6, 9, 8, 2, 1);
plot(x, y, type="l", xlab="x-axis", ylab="y-axis", main="line graph", col="blue");
```

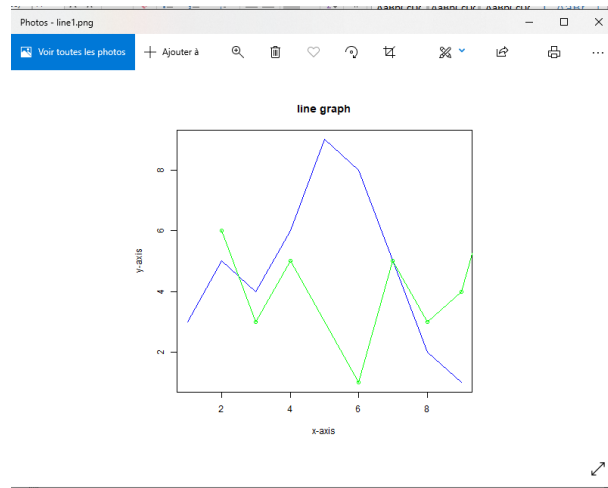
```
> x <- c(1, 2, 3, 4, 5, 6, 8, 9);
> y <- c(3, 5, 4, 6, 9, 8, 2, 1);
> png(file="C:/Ateliers CTS/line.png");
> plot(x, y, type="l", xlab="x-axis", ylab="y-axis", main="line graph", col="blue");
> dev.off();
png
2
```



Vous utilisez type="l" lorsque vous souhaitez tracer un graphique en courbes et type="p" lorsque vous souhaitez tracer un graphique en points ou en nuage de points. xlab est le nom de l'axe x, ylab est le nom de l'axe y, main est le titre principal et col est la couleur du graphique.

Pour tracer un graphique à plusieurs lignes, vous pouvez ajouter la fonction lines() :

```
x <- c(1, 2, 3, 4, 5, 6, 8, 9);
y <- c(3, 5, 4, 6, 9, 8, 2, 1);
x.1 <- c(2, 3, 4, 6, 7, 8, 9, 10);
y.1 <- c(6, 3, 5, 1, 5, 3, 4, 8);
plot(x, y, type="l", xlab="x-axis", ylab="y-axis", main="line graph", col="blue");
lines(x.1, y.1, type="o", col="green");
> x <- c(1, 2, 3, 4, 5, 6, 8, 9);
> y <- c(3, 5, 4, 6, 9, 8, 2, 1);
> x.1 <- c(2, 3, 4, 6, 7, 8, 9, 10);
> y.1 <- c(6, 3, 5, 1, 5, 3, 4, 8);
> png(file="C:/Ateliers CTS/line1.png");
> plot(x, y, type="l", xlab="x-axis", ylab="y-axis", main="line graph", col="blue");
> lines(x.1, y.1, type="o", col="green");
> dev.off();
png
2
```

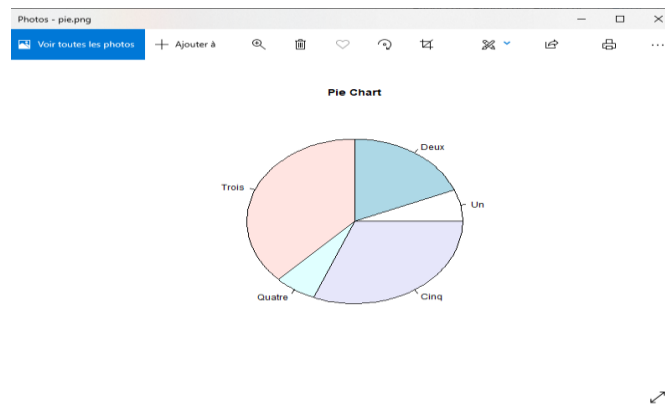


Pour tracer un camembert, vous pouvez utiliser la fonction `pie()` :

```
x <- c(10, 30, 60, 10, 50);
labels <- c("one", "two", "three", "four", "five");
pie(x, labels, main="Pie Chart");

> x <- c(10, 30, 60, 10, 50);
> labels <- c("Un", "Deux", "Trois", "Quatre", "Cinq");
> png(file="C:/Ateliers CTS/pie.png");
> pie(x, labels, main="Pie Chart");
> dev.off();

png
2
```



Pour tracer un camembert 3D, vous devez installer la bibliothèque `plotrix` :

```
> install.packages("plotrix");
Installation du package dans 'C:/Users/Miloud Dahane/Documents/R/win-library/4.1'
(car 'lib' n'est pas spécifié)
--- SVP sélectionnez un miroir CRAN pour cette session ---
essai de l'URL 'https://mirror.ibcp.fr/pub/CRAN/bin/windows/contrib/4.1/plotrix_3.8-1.zip'
Content type 'application/zip' length 1138090 bytes (1.1 MB)
downloaded 1.1 MB

package 'plotrix' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\Miloud Dahane\AppData\Local\Temp\RtmpmIv8Oj\downloaded_packages
```

Vous pouvez utiliser la fonction `require()` ou la fonction `library()` pour appeler la bibliothèque `plotrix` :

Vous pouvez utiliser la fonction `pie3D()` pour tracer le camembert 3D

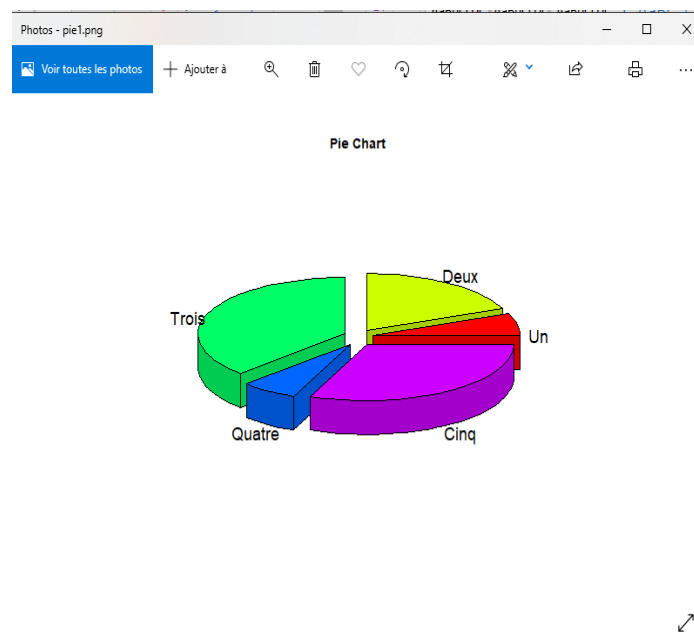
```
library(plotrix);
x <- c(10, 30, 60, 10, 50);
labels <- c("one", "two", "three", "four", "five");
pie3D(x, labels=labels, explode=0.1, main="Pie Chart");
```

ou

```
library(plotrix);
x <- c(10, 30, 60, 10, 50);
labels <- c("one", "two", "three", "four", "five");
png(file="pie.png");
pie3D(x, labels=labels, explode=0.1, main="Pie Chart");
dev.off();
```

```
> library(plotrix);
> png(file="C:/Ateliers CTS/pie1.png");
> pie3D(x, labels=labels, explode=0.1, main="Pie Chart");
> dev.off();
```

png  
2



### 3.3 Scatterplot et Boxplot

Un nuage de points est un graphique qui représente des données à l'aide de points dans le plan cartésien. Chaque point est la valeur de deux variables. Une boîte à moustaches montre les statistiques des données.

Vous pouvez tracer un nuage de points à l'aide de la fonction plot() :

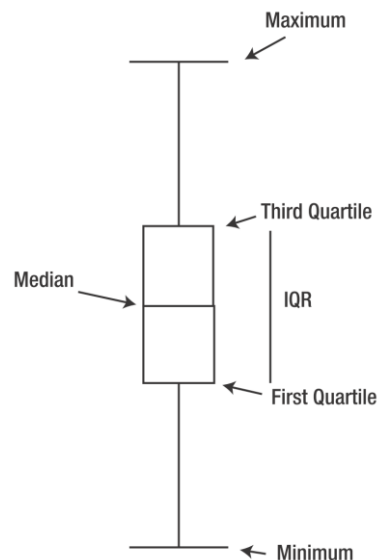
```
x <- c(1, 2, 3, 4, 5, 6, 8, 9);
y <- c(3, 5, 4, 6, 9, 8, 2, 1);
plot(x, y, xlab="x-axis", ylab="y-axis", main="scatterplot");

> x <- c(1, 2, 3, 4, 5, 6, 8, 9);
> y <- c(3, 5, 4, 6, 9, 8, 2, 1);
> png(file="C:/Ateliers CTS/scatter.png");
> plot(x, y, xlab="x-axis", ylab="y-axis", main="scatterplot");
> dev.off();
```

png  
2



Une boîte à moustaches représente la façon dont les données d'un ensemble de données sont distribuées, représentant le minimum, le maximum, la médiane, le premier quartile et le troisième quartile.



Pour créer le boxplot , vous pouvez utiliser la fonction `boxplot()` :

```
set.seed(12);
var1 <- rnorm(100, mean=3, sd=3);
var2 <- rnorm(100, mean=2, sd=2);
var3 <- rnorm(100, mean=1, sd=3);
data <- data.frame(var1, var2, var3);
boxplot(data, main="boxplot", notch=FALSE, varwidth=TRUE, col=c("green", "purple", "blue"));
```

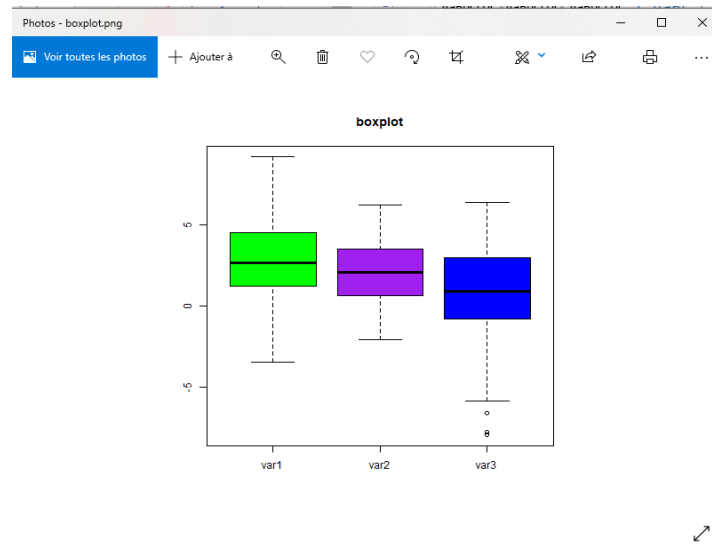
```
> set.seed(12);
> var1 <- rnorm(100, mean=3, sd=3);
> var2 <- rnorm(100, mean=2, sd=2);
> var3 <- rnorm(100, mean=1, sd=3);
> data <- data.frame(var1, var2, var3);
> png(file="C:/Ateliers CTS/boxplot.png");
> boxplot(data, main="boxplot", notch=FALSE, varwidth=TRUE, col=c("green", "purple", "blue"));
> dev.off();
```

png  
2



data est les données, main est le titre principal, notch est une valeur logique pour indiquer comment les médianes des différents groupes correspondent les unes aux autres, varwidth est une valeur logique pour indiquer s'il faut dessiner la largeur de la boîte proportionnellement à la taille de l'échantillon, et col est la couleur de la boîte à moustaches.

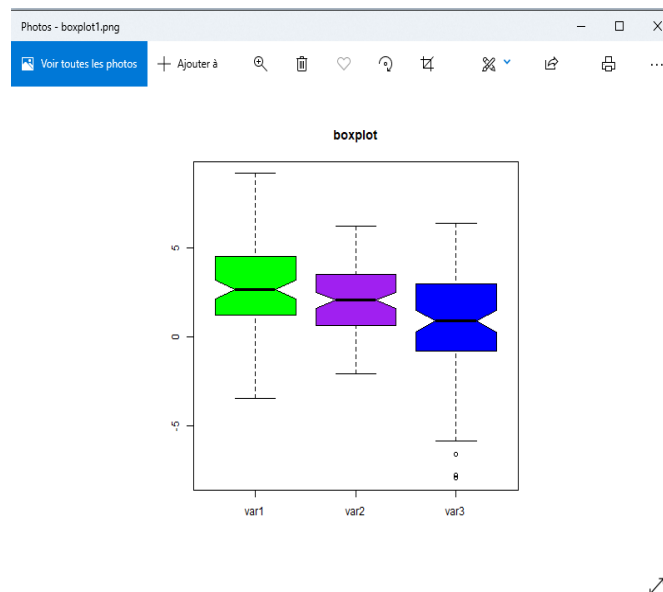
Vous pouvez dessiner une boîte à moustaches avec une encoche en définissant notch=TRUE



```
boxplot(data, main="boxplot", notch=true, varwidth=TRUE, col=c("green", "purple", "blue"));
```

```
> png(file="C:/Ateliers CTS/boxplot1.png");
> boxplot(data, main="boxplot", notch=TRUE, varwidth=TRUE, col=c("green", "purple", "blue"));
> dev.off();
```

png  
2



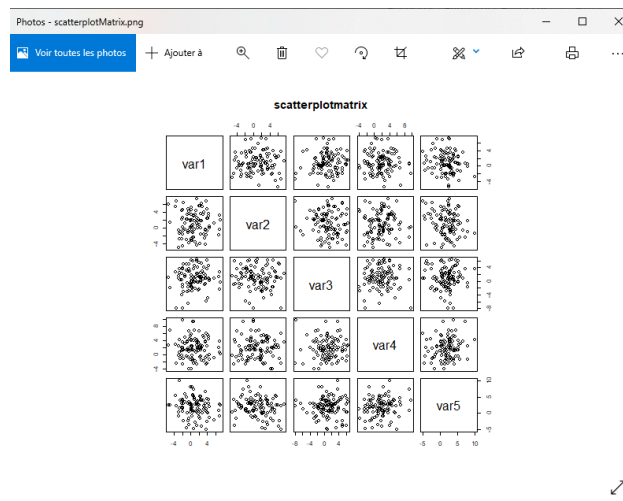
### 3.4 Marice Scatterplot

Une matrice de nuage de points est utilisée pour trouver la corrélation entre une variable et d'autres variables, et vous l'utilisez généralement pour sélectionner les variables importantes, ce qui est également connu sous le nom de sélection de variables.

Pour tracer la matrice de nuage de points, vous pouvez utiliser la fonction `pairs()` :

```
set.seed(12);
var1 <- rnorm(100, mean=1, sd=3);
var2 <- rnorm(100, mean=1, sd=3);
var3 <- rnorm(100, mean=1, sd=3);
var4 <- rnorm(100, mean=2, sd=3);
var5 <- rnorm(100, mean=2, sd=3);
data <- data.frame(var1, var2, var3, var4, var5);
pairs(~var1+var2+var3+var4+var5, data=data, main="scatterplot matrix");
```

```
> setwd("C:/Ateliers CTS");
> set.seed(12);
> var1 <- rnorm(100, mean=1, sd=3);
> var2 <- rnorm(100, mean=1, sd=3);
> var3 <- rnorm(100, mean=1, sd=3);
> var4 <- rnorm(100, mean=2, sd=3);
> var5 <- rnorm(100, mean=2, sd=3);
> data <- data.frame(var1, var2, var3, var4, var5);
> png(file="C:/Ateliers CTS/scatterplotMatrix.png");
> pairs(~var1+var2+var3+var4+var5, data=data, main="scatterplotmatrix");
> dev.off();
```



### 3.5 Utiliser ggplot2

ggplot2 est un package créé par Hadley Wickham qui offre un langage graphique puissant pour créer des graphiques avancés. ggplot2 est très populaire et célèbre dans la communauté R, et il nous permet de créer des graphiques qui représentent des données univariées, multivariées et catégorielles de manière simple. La fonctionnalité intégrée de R offre le tracé de graphiques, mais ggplot nous permet de tracer des graphiques plus avancés en utilisant la grammaire des graphiques.

Pour utiliser ggplot2, vous devez installer le package :

```
> install.packages("ggplot2");
Installation du package dans 'C:/Users/Miloud Dahane/Documents/R/win-library/4.1'
(car 'lib' n'est pas spécifié)
--- SVP sélectionnez un miroir CRAN pour cette session ---
installation des dépendances 'colorspace', 'cli', 'crayon', 'utf8', 'farver', 'labeling', 'lifecycle', 'munsell', 'R6', 'RColorBrewer', 'Rdpack', 'rlang', 'rmarkdown', 'rstatix', 'tidyr', 'tibble', 'tidyselect', 'vctrs', 'withr'

essai de l'URL 'https://mirror.ibcp.fr/pub/CRAN/bin/windows/contrib/4.1/colorspace_2.0-2.zip'
Content type 'application/zip' length 2644645 bytes (2.5 MB)
downloaded 2.5 MB

essai de l'URL 'https://mirror.ibcp.fr/pub/CRAN/bin/windows/contrib/4.1/cli_3.0.1.zip'
Content type 'application/zip' length 758127 bytes (740 KB)
downloaded 740 KB

essai de l'URL 'https://mirror.ibcp.fr/pub/CRAN/bin/windows/contrib/4.1/crayon_1.4.1.zip'
Content type 'application/zip' length 142532 bytes (139 KB)
downloaded 139 KB
```

Pour utiliser ggplot2, vous devez appeler la bibliothèque à l'aide des fonctions `library()` ou `require()`. Vous devez également indiquer à ggplot quel ensemble de données utiliser, et vous pouvez utiliser la fonction `ggplot()` :

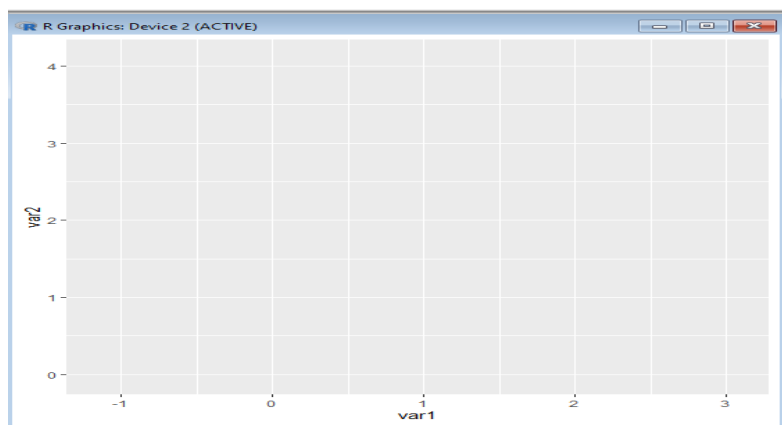
```
> library(ggplot2);
Use suppressPackageStartupMessages() to eliminate package startup messages
> set.seed(12);
> var1 <- rnorm(100, mean=1, sd=1);
> var2 <- rnorm(100, mean=2, sd=1);
> var3 <- rnorm(100, mean=1, sd=2);
> data <- data.frame(var1, var2, var3);
> ggplot(data);
```

```
library(ggplot2);
set.seed(12);
var1 <- rnorm(100, mean=1, sd=1);
var2 <- rnorm(100, mean=2, sd=1);
var3 <- rnorm(100, mean=1, sd=2);
data <- data.frame(var1, var2, var3);
ggplot(data);
```

`ggplot()` ne peut prendre qu'une variable de trame de données.

Vous pouvez utiliser l'esthétique dans ggplot2 via la fonction `aes()` :

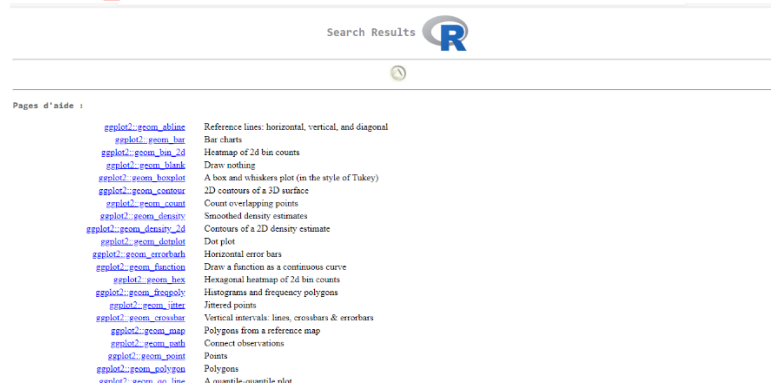
```
> ggplot(data, aes(x=var1, y=var2));
```



### 3.5.1 Géométrie dans ggplot2

Les objets géométriques sont les tracés ou les graphiques que vous souhaitez placer dans le graphique. Vous pouvez utiliser `geom_point()` pour créer un nuage de points, `geom_line()` pour créer un graphique linéaire et `geom_boxplot()` pour créer une boîte à moustaches dans le graphique. Vous pouvez voir les objets géométriques disponibles en utilisant :

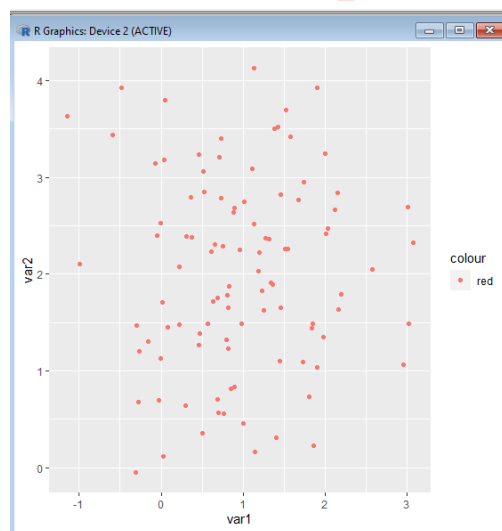
```
> help.search("geom_", package="ggplot2");
```



Dans ggplot2, `geom` est également les couches du graphique. Vous pouvez ajouter un objet `geom` après l'autre, tout comme ajouter un calque après un autre calque.

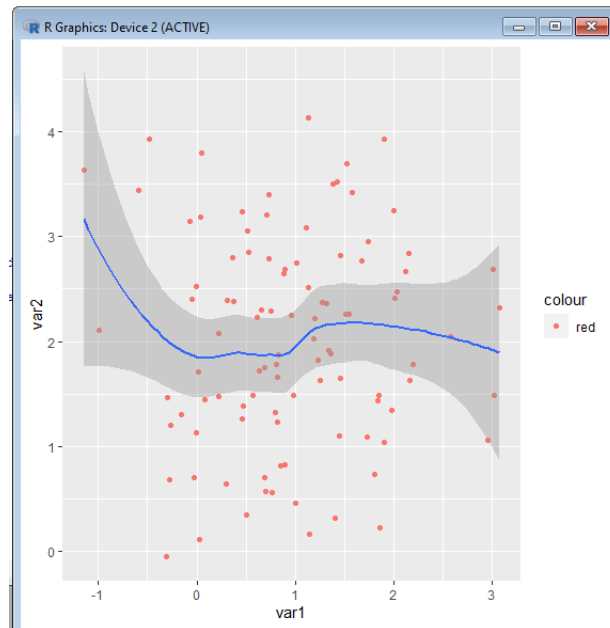
Vous pouvez ajouter un nuage de points à l'aide de la fonction `geom_point()` :

```
ggplot(data, aes(x=var1, y=var2)) + geom_point(aes(color="red"));  
> ggplot(data, aes(x=var1, y=var2)) + geom_point(aes(color="red"));
```



You can add in a smoother that includes a line and a ribbon to the scatter plot using another layer:

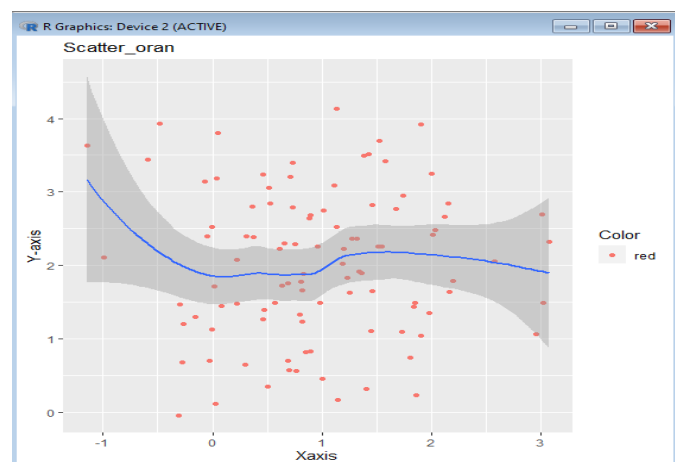
```
ggplot(data, aes(x=var1, y=var2)) + geom_point(aes(color="red")) + geom_smooth();  
> ggplot(data, aes(x=var1, y=var2)) + geom_point(aes(color="red")) + geom_smooth() using method = 'loess' and formula 'y ~ x'
```



### 3.5.2 Etiquettes

Vous avez tracé les graphiques dans les graphes, alors ajoutons maintenant le titre principal et les titres des axes x et y. Vous pouvez le faire en utilisant la couche `labs()` pour spécifier les étiquettes. Pour ajouter le titre de l'axe des x, le titre de l'axe des y et le titre principal, vous pouvez utiliser la fonction `labs()` :

```
ggplot(data, aes(x=var1, y=var2)) + geom_point(aes(color="red")) + geom_smooth() +  
  labs(title="Scatter", x = "Xaxis", y = "Y-axis", color="Color");  
> ggplot(data, aes(x=var1, y=var2)) + geom_point(aes(color="red")) + geom_smooth() + labs(title="Scatter_oran", x = "Xaxis", y = "Y-axis", color="Color"); geo$
```



### 3.5.3 Thèmes

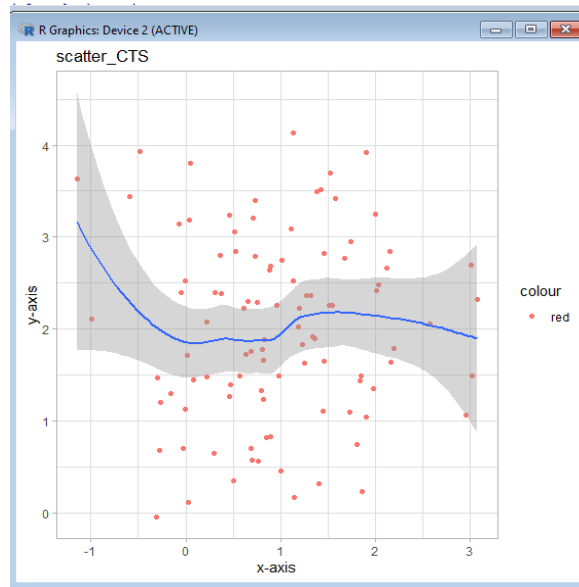
Si vous souhaitez modifier la taille et le style des étiquettes et des légendes, la fonction `theme()` peut vous aider. La fonction `theme()` dans `ggplot` gère les éléments :

- Étiquettes d'axe
- Arrière-plan du tracé

- Arrière-plan à facettes
- Apparence de la légende

Exemple avec l'utilisation `theme_light()` :

```
ggplot(data, aes(x=var1, y=var2)) + geom_point(aes(color="red")) + geom_smooth() +
  labs(title="scatter", x="x-axis", y="y-axis") + theme_light();
> ggplot(data, aes(x=var1, y=var2)) + geom_point(aes(color="red")) + geom_smooth() + labs(title="scatter_CTS", x="x-axis", y="y-axis") + theme_light(); 'geom_s$
```



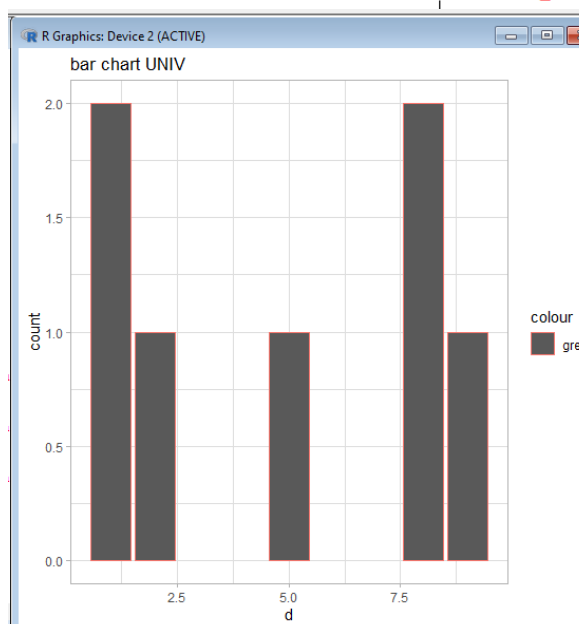
### 3.5.4 Charts

Après avoir appris les bases de `ggplot` et la grammaire des graphiques, les sections suivantes couvrent quelques graphiques courants qui peuvent être tracés avec `ggplot`.

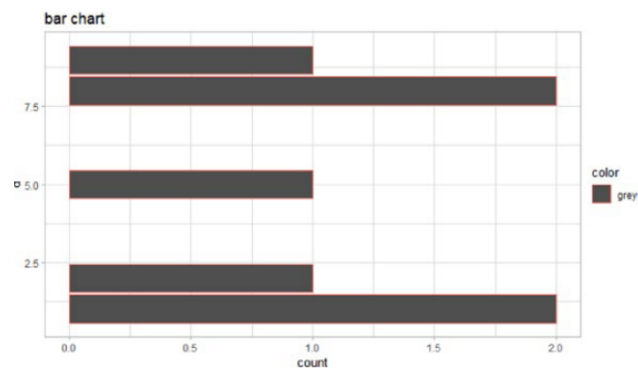
#### 3.5.4.1 Graphique à barres

Un graphique à barres est utilisé lorsque vous souhaitez comparer les éléments entre les groupes :

```
d <- c(1, 5, 8, 9, 8, 2, 1);
df <- data.frame(d);
ggplot(df) + geom_bar(aes(color="grey", x=d)) + labs(title="bar chart") + theme_light();
ggplot(df) + geom_bar(aes(color="grey", x=d)) + labs(title="bar chart UNIV") + theme_light();
```



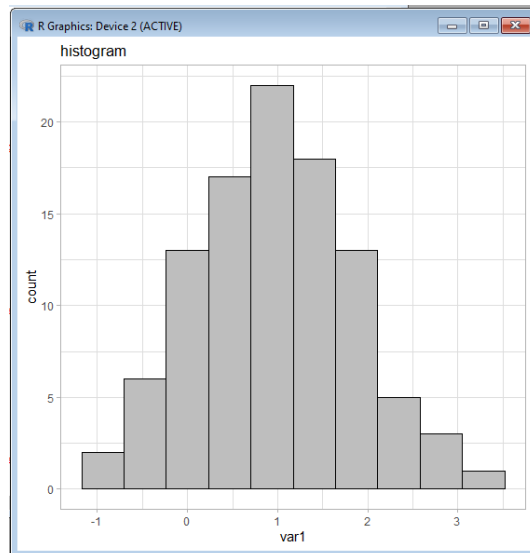
```
ggplot(df) + geom_bar(aes(color="grey", x=d)) + coord_flip() + labs(title="bar chart") +
theme_light();
```



### 3.5.4.2 Histogramme

Un histogramme vous permet de voir si les données sont normalement distribuées.

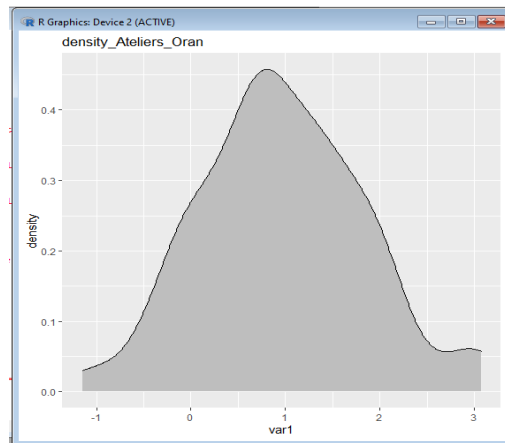
```
set.seed(12);
var1 <- rnorm(100, mean=1, sd=1);
var2 <- rnorm(100, mean=2, sd=1);
var3 <- rnorm(100, mean=1, sd=2);
data <- data.frame(var1, var2, var3);
ggplot(data, aes(x=var1)) + geom_histogram(bins=10, color="black", fill="grey") +
labs(title="histogram") + theme_light();
> ggplot(data, aes(x=var1)) + geom_histogram(bins=10,color="black", fill="grey") + labs(title="histogram") +theme_light();
```



### 3.5.4.3 Diagramme de densité

Un graphique de densité peut également vous montrer si les données sont normalement distribuées

```
ggplot(data, aes(x=var1)) + geom_density(fill="grey") + labs(title="density");
> ggplot(data, aes(x=var1)) + geom_density(fill="grey") +labs(title="density_Ateliers_Oran");
```

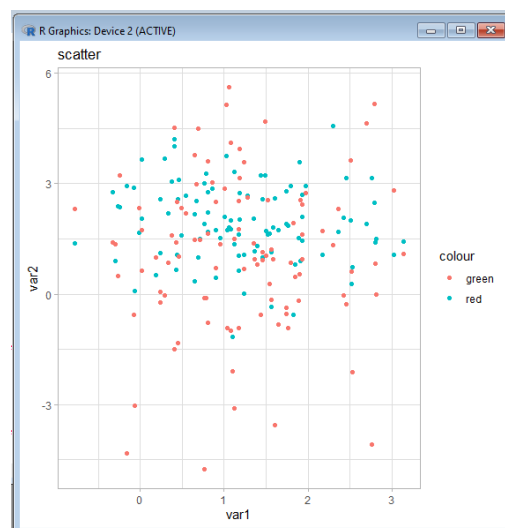


### 3.5.5 Scatterplot

Un nuage de points montre les relations entre deux variables.

```
> ggplot(data) + geom_point(aes(color="red", x=var1, y=var2)) + geom_point(aes(color="green", x=var1, y=var3)) + labs(title="scatter") + theme_light();
```

```
ggplot(data) +  
  geom_point(aes(color="red", x=var1, y=var2)) +  
  geom_point(aes(color="green", x=var1, y=var3)) +  
  labs(title="scatter") +  
  theme_light();
```



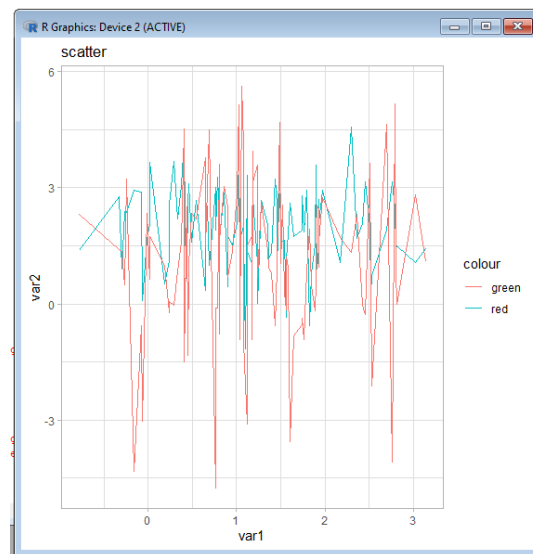


### 3.5.6 Line chart

Un graphique linéaire montre également la relation entre deux variables et peut également être utilisé pour l'analyse des tendances.

```
ggplot(data) +  
  geom_line(aes(color="red", x=var1, y=var2)) +  
  geom_line(aes(color="green", x=var1, y=var3)) +  
  labs(title="scatter") +  
  theme_light();
```

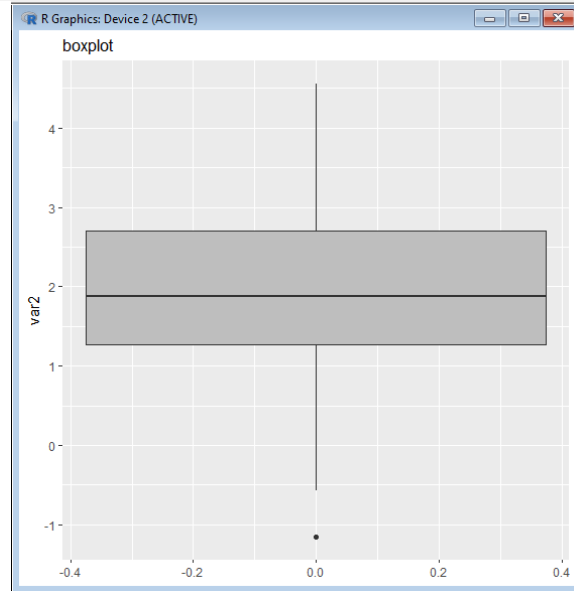
```
> ggplot(data) + geom_line(aes(color="red", x=var1, y=var2))+ geom_line(aes(color="green", x=var1, y=var3)) +labs(title="scatter") + theme_light();
```



### 3.5.7 Boxplot

Une boîte à moustaches montre les statistiques des données.

```
> ggplot(data, aes(y=var2)) + geom_boxplot(fill="grey") + labs(title="boxplot");  
ggplot(data, aes(y=var2)) + geom_boxplot(fill="grey") +  
labs(title="boxplot");
```



### 3.5.8 Charts interactifs avec Plotly et ggplot2

Plotly JS vous permet de créer des graphiques interactifs de qualité publication. Vous pouvez créer un graphique Plotly en utilisant ggplot. Pour utiliser Plotly ou pour créer un graphique Plotly, vous devez télécharger le package plotly comme suit :

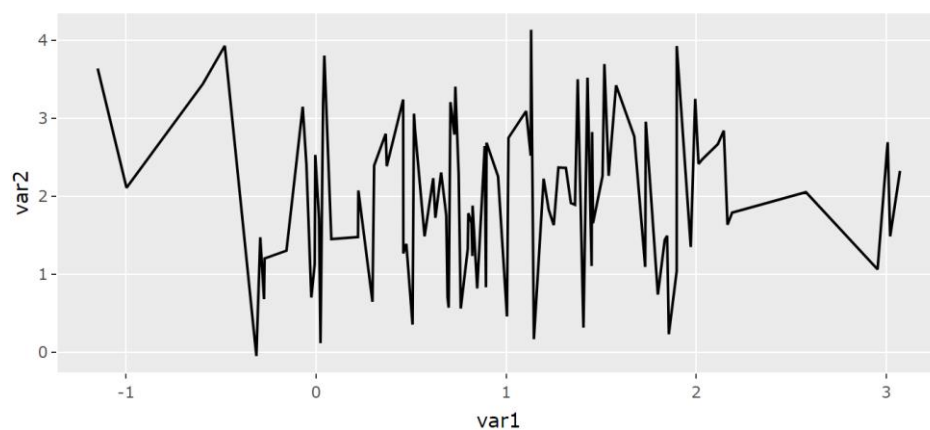
```
> install.packages("plotly");  
Installation du package dans 'C:/Users/Miloud Dahane/Documents/R/win-library/4.1'  
(car 'lib' n'est pas spécifié)  
--- SVP sélectionnez un miroir CRAN pour cette session ---  
installation des dépendances 'sys', 'askpass', 'curl', 'mime', 'openssl', 'fastmap', 'yaml', 'tidyselect', 'cpp11', 'generics', 'Rcpp', 'later', 'httr', 'jsonl'  
  
essai de l'URL 'https://mirror.ibcp.fr/pub/CRAN/bin/windows/contrib/4.1/sys_3.4.zip'  
Content type 'application/zip' length 59837 bytes (58 KB)  
downloaded 58 KB  
  
essai de l'URL 'https://mirror.ibcp.fr/pub/CRAN/bin/windows/contrib/4.1/askpass_1.1.zip'  
Content type 'application/zip' length 243661 bytes (237 KB)  
downloaded 237 KB  
  
essai de l'URL 'https://mirror.ibcp.fr/pub/CRAN/bin/windows/contrib/4.1/curl_4.3.2.zip'
```

Pour créer un graphique Plotly , vous pouvez utiliser la fonction ggplotly() :

```
> set.seed(12);  
> var1 <- rnorm(100, mean=1, sd=1);  
> var2 <- rnorm(100, mean=2, sd=1);  
> var3 <- rnorm(100, mean=1, sd=2);  
> data <- data.frame(var1, var2, var3);  
> gg <- ggplot(data) + geom_line(aes(x=var1, y=var2));  
> g <- ggplotly(gg);  
> g;
```

```
library(plotly);  
set.seed(12);  
var1 <- rnorm(100, mean=1, sd=1);  
var2 <- rnorm(100, mean=2, sd=1);  
var3 <- rnorm(100, mean=1, sd=2);  
data <- data.frame(var1, var2, var3);
```

```
gg <- ggplot(data) + geom_line(aes(x=var1, y=var2));  
g <- ggplotly(gg);  
g;
```



## 4 Inférence statistique

### 4.1 De quoi s'agit-il ?

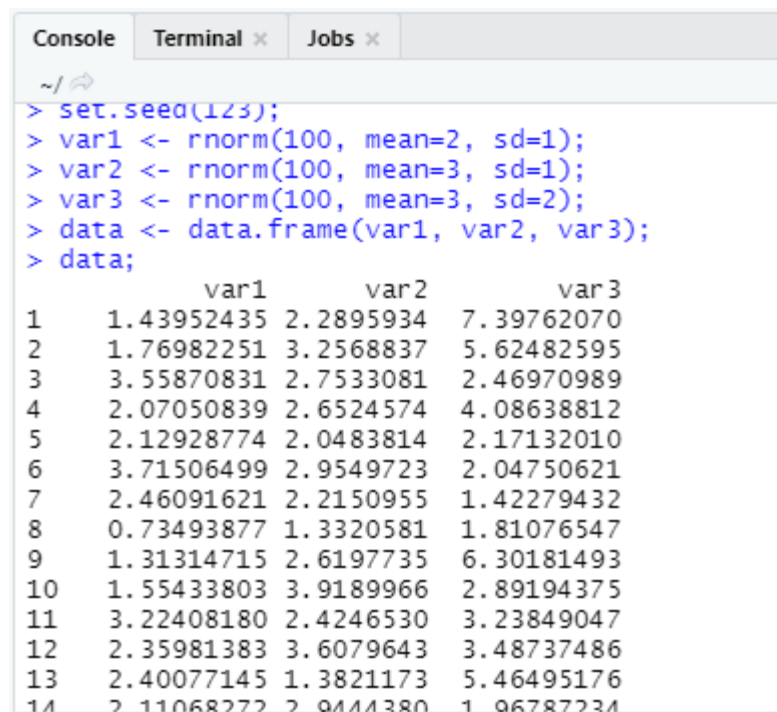
Les statistiques descriptives permettent de dresser des indicateurs synthétiques sur des données en utilisant les tendances centrales et de dispersion. L'inférence statistique établit des inférences de propriétés sur des paramètres des échantillons d'une population donnée. En estimant des paramètres, nous caractérisons des paramètres d'une population donnée. En test d'hypothèses, toute question est posée sous forme d'une hypothèse. Par exemple, on peut s'intéresser à l'existence d'une différence de notes en mathématiques entre deux groupes d'étudiants ? La question posée initialement est de la forme *existe-t-il* ? La réponse est de la forme *il existe*, ou *il n'existe pas*. La question initialement posée est dite hypothèse nulle,  $H_0$ , et l'hypothèse alternative  $H_a$ . L'hypothèse nulle peut être formulée  $\mu_1 = \mu_2$ , et l'alternative  $\mu_1 \neq \mu_2$ , où  $\mu_1$  est la moyenne du premier groupe et  $\mu_2$  est la moyenne du deuxième groupe. Nous ferons appel à la valeur p (p-value) pour accepter ou rejeter l'hypothèse nulle. En général, on rejette l'hypothèse nulle à partir d'une valeur p inférieure à 0.05.

Les méthodes de régression permettent de dresser une relation entre les variables dépendantes et les indépendantes d'une population donnée, et étudier les différentes relations possibles.

### 4.2 `apply()`, `lapply()`, `sapply()`

Commençons tout d'abord par se donner une population tirée au sort avec 3 caractéristiques (variables).

```
set.seed(123);
var1 <- rnorm(100, mean=2, sd=1);
var2 <- rnorm(100, mean=3, sd=1);
var3 <- rnorm(100, mean=3, sd=2);
data <- data.frame(var1, var2, var3);
data;
```



The screenshot shows a R console window with the following content:

```
> set.seed(123);
> var1 <- rnorm(100, mean=2, sd=1);
> var2 <- rnorm(100, mean=3, sd=1);
> var3 <- rnorm(100, mean=3, sd=2);
> data <- data.frame(var1, var2, var3);
> data;
```

	var1	var2	var3
1	1.43952435	2.2895934	7.39762070
2	1.76982251	3.2568837	5.62482595
3	3.55870831	2.7533081	2.46970989
4	2.07050839	2.6524574	4.08638812
5	2.12928774	2.0483814	2.17132010
6	3.71506499	2.9549723	2.04750621
7	2.46091621	2.2150955	1.42279432
8	0.73493877	1.3320581	1.81076547
9	1.31314715	2.6197735	6.30181493
10	1.55433803	3.9189966	2.89194375
11	3.22408180	2.4246530	3.23849047
12	2.35981383	3.6079643	3.48737486
13	2.40077145	1.3821173	5.46495176
14	2.11068272	2.9444380	1.96787234

On peut faire appel à la fonction `apply` pour calculer la moyenne des individus ou des variables. Soit sur les individus :

```
apply(data, 1, mean);
```

```

Console Terminal x Jobs x
~/
98 3.53261063 1.7487286 0.29819463
99 1.76429964 2.3888341 3.04196717
100 0.97357910 1.8145199 5.49982914
> apply(data, 1, mean);
[1] 3.708913 3.550511 2.927242 2.936451 2.116330 2.905848 2.032935
[8] 1.292587 3.411579 2.788426 2.962408 3.151718 3.082613 2.340998
[15] 1.992851 4.479820 2.573733 1.315515 1.793035 1.311216 1.967292
[22] 2.690174 2.901045 2.810064 2.830503 1.926952 2.554660 2.265633
[29] 2.556436 2.383774 4.593868 2.658598 3.121812 2.326194 1.873185
[36] 2.395315 2.242476 3.172000 3.417250 1.537521 2.143278 2.175163
[43] 2.718194 2.126560 2.416108 3.383570 1.977803 1.833860 3.183512
[50] 2.533507 2.763284 2.538914 2.533833 2.847073 3.617265 3.019648
[57] 2.058612 2.157804 2.057810 1.501852 2.706705 1.877070 2.166087

```

Sinon aussi sur les variables :

```
apply(data, 2, mean);
```

```

Console Terminal x Jobs x
~/
[85] 2.593174 2.800894 3.549599 3.933870 2.663340 2.995215 3.848239
[92] 3.444024 3.541280 1.773932 4.018272 3.176785 4.840582 1.859845
[99] 2.398367 2.762643
> apply(data, 2, mean);
      var1      var2      var3
2.090406 2.892453 3.240930
> |

```

On peut calculer la moyenne sur une variable donnée :

```
lapply(data$var1, mean);
```

ou encore

```
sapply(data$var1, mean);
```

### 4.3 Echantillonnage

C'est l'opération qui consiste à sélectionner un sous-ensemble d'éléments à partir de la masse complète de données d'une façon aléatoire.

#### 4.3.1 Echantillonnage aléatoire

Tout élément doit avoir la même probabilité d'être sélectionnée.

#### 4.3.2 Echantillonnage stratifié

Cet échantillonnage consiste à diviser la population en groupe dont les éléments sont tirés au sort aléatoirement, chaque élément par rapport à son groupe.

#### 4.3.3 Echantillonnage par partitionnement

Par rapport à l'échantillonnage stratifié, les groupes aussi sont tirés au sort.

### 4.4 Comment échantillonner ?

On peut via la fonction sample :

```
sample(data$var1, 5, replace=TRUE);
```

```
Console Terminal x Jobs x
~/
> set.seed(123);
> var1 <- rnorm(100, mean=2, sd=1);
> var2 <- rnorm(100, mean=3, sd=1);
> var3 <- rnorm(100, mean=3, sd=2);
> data <- data.frame(var1, var2, var3);
> sample(data$var1, 5, replace=TRUE);
[1] 2.215942 1.497677 1.861109 1.313147 3.224082
> |
```

Pour utiliser l'échantillonnage stratifié, il y a besoin de télécharger et charger la librairie dplyr :

```
install.packages("dplyr");
library(dplyr);
```

Nous travaillerons maintenant sur le dataset iris :

```
data(iris);
summary(iris);
```

```
Console Terminal x Jobs x
~/
[1] 2.215942 1.497677 1.861109 1.313147 3.224082
> data(iris);
> summary(iris);
  Sepal.Length   Sepal.width   Petal.Length   Petal.width
Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
Median :5.800   Median :3.000   Median :4.350   Median :1.300
Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
   Species
setosa   :50
versicolor:50
virginica :50
> |
```

Soit la requête pour effectuer un échantillonnage stratifié :

```
iris_sample <- iris %>%
  group_by(Species) %>%
  sample_n(13)
iris_sample;
```

```

Console Terminal x Jobs x
~/
> iris_sample <- iris %>%
+   group_by(species) %>%
+   sample_n(13)
> iris_sample;
# A tibble: 39 x 5
# Groups:   species [3]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
      <dbl>      <dbl>      <dbl>      <dbl>    <fct>
1         4.4         3.2         1.3         0.2 setosa
2         4.7         3.2         1.6         0.2 setosa
3         4.9         3.1         1.5         0.1 setosa
4         5.5         4.2         1.4         0.2 setosa
5         4.9         3.6         1.4         0.1 setosa
6         5.1         3.8         1.5         0.3 setosa
7         4.3         3.0         1.1         0.1 setosa
8         5.4         3.9         1.7         0.4 setosa
9         5.1         3.5         1.4         0.3 setosa
10        4.9         3.0         1.4         0.2 setosa
# ... with 29 more rows
> |

```

View(iris\_sample);

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	4.4	3.2	1.3	0.2	setosa
2	4.7	3.2	1.6	0.2	setosa
3	4.9	3.1	1.5	0.1	setosa
4	5.5	4.2	1.4	0.2	setosa
5	4.9	3.6	1.4	0.1	setosa
6	5.1	3.8	1.5	0.3	setosa
7	4.3	3.0	1.1	0.1	setosa
8	5.4	3.9	1.7	0.4	setosa
9	5.1	3.5	1.4	0.3	setosa
10	4.9	3.0	1.4	0.2	setosa
11	5.0	3.4	1.5	0.2	setosa

...

30	6.5	3.0	5.8	2.2	virginica
31	6.5	3.2	5.1	2.0	virginica
32	6.2	3.4	5.4	2.3	virginica
33	5.8	2.7	5.1	1.9	virginica
34	7.2	3.6	6.1	2.5	virginica
35	4.9	2.5	4.5	1.7	virginica
36	6.4	3.1	5.5	1.8	virginica
37	7.7	3.0	6.1	2.3	virginica
38	7.7	2.8	6.7	2.0	virginica
39	6.3	2.8	5.1	1.5	virginica

## 4.5 Corrélations

La corrélation mesure de combien les deux variables sont linéairement dépendantes. Elle permet notamment de réduire le nombre de variables.

Soit la formule de calcul de corrélation entre deux variables :

$$r_{x,y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

Reprenons les données :

```
set.seed(123);
var1 <- rnorm(100, mean=2, sd=1);
var2 <- rnorm(100, mean=3, sd=1);
var3 <- rnorm(100, mean=3, sd=2);
data <- data.frame(var1, var2, var3);
```

Pour calculer la corrélation, il suffit d'invoquer :

```
cor(data$var1, data$var2);
```

```

Console Terminal x Jobs x
~/
> data <- data.frame(var1, var2, var3);
> cor(data$var1, data$var2);
[1] -0.04953215
> |
```

## 4.6 Covariance

Elle mesure la variabilité entre deux variables données.

Elle est donnée par la formule :

$$cov(X, Y) = \sum_{i=1}^N \frac{(x_i - \bar{x})(y_i - \bar{y})}{N}$$

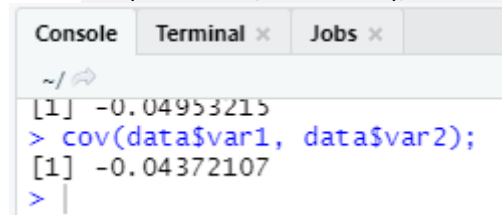
Reprenons encore les mêmes données :

```
set.seed(123);
var1 <- rnorm(100, mean=2, sd=1);
var2 <- rnorm(100, mean=3, sd=1);
var3 <- rnorm(100, mean=3, sd=2);
data <- data.frame(var1, var2, var3);
```



Pour calculer la corrélation, il suffit d'invoquer :

```
cov(data$var1, data$var2);
```



```
Console Terminal x Jobs x
~/
[1] -0.04953215
> cov(data$var1, data$var2);
[1] -0.04372107
> |
```

#### 4.7 Test d'hypothèses et P-Value

On peut formuler l'hypothèse nulle et l'hypothèse alternative comme suit :

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu_1 \neq \mu_2$$

où  $\mu_1$  est la moyenne du premier groupe et  $\mu_2$  est la moyenne du deuxième groupe.

On peut calculer la valeur p (p value) sur les données relativement à leurs deux hypothèses. On peut faire appel au t-test sur des variables continues, chi-square sur des données catégorielles. Sur des données plus complexes, on peut faire appel à ANOVA. Si les données ne suivent pas la loi normale, on peut faire appel aux méthodes de test non-paramétriques, notamment Wilcoxon.

#### 4.8 T-Test

Il permet de comparer des moyennes.

Soit par exemple les données :

```
set.seed(123);
var1 <- rnorm(100, mean=2, sd=1);
var2 <- rnorm(100, mean=3, sd=1);
var3 <- rnorm(100, mean=3, sd=2);
data <- data.frame(var1, var2, var3);
data;
```

Evaluons le test d'hypothèse suivant :

$$H_0 : \mu_1 = 0.6$$

$$H_a : \mu_1 \neq 0.6$$

où  $\mu_1$  est la moyenne de la variable `data$var1`. On voudrait donc savoir si la moyenne 0.6 est statistiquement proche de la moyenne de `data$var1`. Pour ce, il suffit d'invoquer :

```
t.test(data$var1, mu=0.6);
```

```
Console Terminal x Jobs x
~/
99 1.70429904 2.3888341 3.04190717
100 0.97357910 1.8145199 5.49982914
> t.test(data$var1, mu=0.6);

One Sample t-test

data: data$var1
t = 16.328, df = 99, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0.6
95 percent confidence interval:
 1.909283 2.271528
sample estimates:
mean of x
 2.090406

> |
```

Comme la valeur p est égale à  $2.2e-16$ , alors on peut établir statistiquement que la moyenne 0.6 n'est pas proche de celle de la variable en question.

On peut aussi comparer la moyenne de deux variables comme suit :

```
t.test(data$var1, data$var2, var.equal=TRUE, paired=FALSE);
```

```
Console Terminal x Jobs x
~/
2.090406

> t.test(data$var1, data$var2, var.equal=TRUE, paired=FALSE);

Two Sample t-test

data: data$var1 and data$var2
t = -6.0315, df = 198, p-value = 7.843e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.0642808 -0.5398138
sample estimates:
mean of x mean of y
 2.090406  2.892453

> |
```

(paired test : Test sur la différence égale à zéro, donc sur la différence des deux variables).  
D'où le fait statistique que les deux moyennes sont différentes, dès lors que la valeur p est  $7.843e-09$ .

#### 4.8.1 T-Test sur la différence de deux variables

On utilisera le « paired test » : Test sur la différence égale à zéro, donc sur la différence des deux variables.

```
t.test(data$var1, data$var2, paired=TRUE);
```

```
Console Terminal x Jobs x
~/
2.030400 2.032433

> t.test(data$var1, data$var2, paired=TRUE);

Paired t-test

data: data$var1 and data$var2
t = -5.8876, df = 99, p-value = 5.379e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.0723482 -0.5317464
sample estimates:
mean of the differences
      -0.8020473

> |
```

La différence des deux moyennes n'est pas nulle.

#### 4.8.2 Formes de T-Test

```
# Comparaison d'une moyenne observée
# à une moyenne théorique mu
t.test(x,mu=0)
# Test de student non-apparié
# Comparaison des moyennes de deux groupes (x et y)
t.test(x,y)
# Test de student apparié
t.test(x,y,paired=TRUE)
t.test(x,y,alternative=c("two.sided","less","greater"))

t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)
# S3 method for formula
t.test(formula, data, subset, na.action, ...)
```

#### 4.9 Test de Chi-Square

Le Chi-Square est dédié aux variables catégorielles.

##### 4.9.1 Fit Test

Soit les données :

```
data <- c(B=200, c=300, D=400);
```

Soit l'appel de Chi-Square :

```
chisq.test(data);
```

```

Console Terminal x Jobs x
~/
> data <- c(B=200, c=300, D=400);
> chisq.test(data);

      Chi-squared test for given probabilities

data:  data
X-squared = 66.667, df = 2, p-value = 3.338e-15

> |

```

#### 4.9.2 Test de contingence

Soient les données :

```

var1 <- c("Male", "Female", "Male", "Female", "Male", "Female", "Male", "Female", "Male",
"Female");
var2 <- c("chocolate", "strawberry", "strawberry", "strawberry", "chocolate", "chocolate",
"chocolate", "strawberry", "strawberry", "strawberry");
data <- data.frame(var1, var2);
data;

```

On peut calculer la table des fréquences :

```

data.table <- table(data$var1, data$var2);
data.table;

```

```

Console Terminal x Jobs x
~/
9   Male strawberry
10  Female strawberry
> data.table <- table(data$var1, data$var2);
> data.table;

      chocolate strawberry
Female         1          4
Male           3          2

> |

```

On peut faire un test d'indépendance entre les deux variables via Chi-Square.

- $H_0$  : les deux variables sont indépendantes
- $H_a$  : les deux variables sont non indépendantes

On procédera comme suit :

```

chisq.test(data.table);

```

```

Console Terminal x Jobs x
~/
chocolate strawberry
Female      1      4
Male        3      2
> chisq.test(data.table);

Pearson's Chi-squared test with Yates' continuity correction

data: data.table
X-squared = 0.41667, df = 1, p-value = 0.5186

warning message:
In chisq.test(data.table) :
  l'approximation du Chi-2 est peut-être incorrecte
> |

```

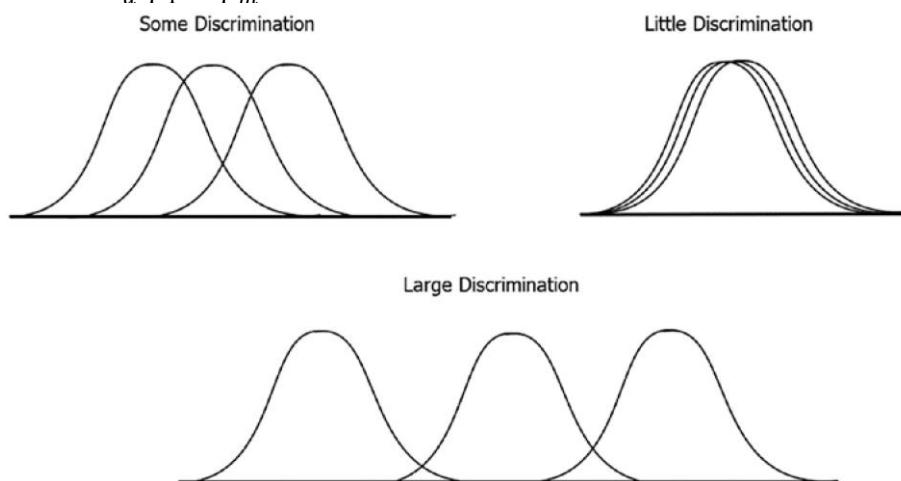
d'où les deux variables sont indépendantes.

#### 4.10 ANOVA

Elle permet de comparer plusieurs moyennes à la fois.

En utilisant la terminologie du test d'hypothèses :

- $H_0: \mu_1 = \dots = \mu_L$
- $H_a: \mu_1 \neq \mu_m$



##### 4.10.1 One-Way ANOVA

Si nous disposons d'une seule variable indépendante.

Soit les données :

```

set.seed(123);
var1 <- rnorm(12, mean=2, sd=1);
var2 <- c("B", "B", "B", "B", "C", "C", "C", "C", "C", "D", "D", "B");
data <- data.frame(var1, var2);
data;

```

```

Console Terminal x Jobs x
~/
> set.seed(123);
> var1 <- rnorm(12, mean=2, sd=1);
> var2 <- c("B", "B", "B", "B", "C", "C", "C", "C", "C", "D", "D", "B");
> data <- data.frame(var1, var2);
> data;
      var1 var2
1  1.4395244  B
2  1.7698225  B
3  3.5587083  B
4  2.0705084  B
5  2.1292877  C
6  3.7150650  C
7  2.4609162  C
8  0.7349388  C
9  1.3131471  C
10 1.5543380  D
11 3.2240818  D
12 2.3598138  B
> |

```

```

fit <- aov(data$var1 ~ data$var2, data=data);
fit;

```

```

Console Terminal x Jobs x
~/
> fit <- aov(data$var1 ~ data$var2, data=data);
> fit;
call:
aov(formula = data$var1 ~ data$var2, data = data)

Terms:
              data$var2 Residuals
Sum of Squares    0.162695   9.255706
Deg. of Freedom         2         9

Residual standard error: 1.014106
Estimated effects may be unbalanced
> |

```

```
summary(fit);
```

```

56 summary(fit);|
56:14 (Top Level)
Console Terminal x Jobs x
~/
> summary(fit);
              Df Sum Sq Mean Sq F value Pr(>F)
data$var2      2  0.163   0.0813   0.079  0.925
Residuals     9  9.256   1.0284
> |

```

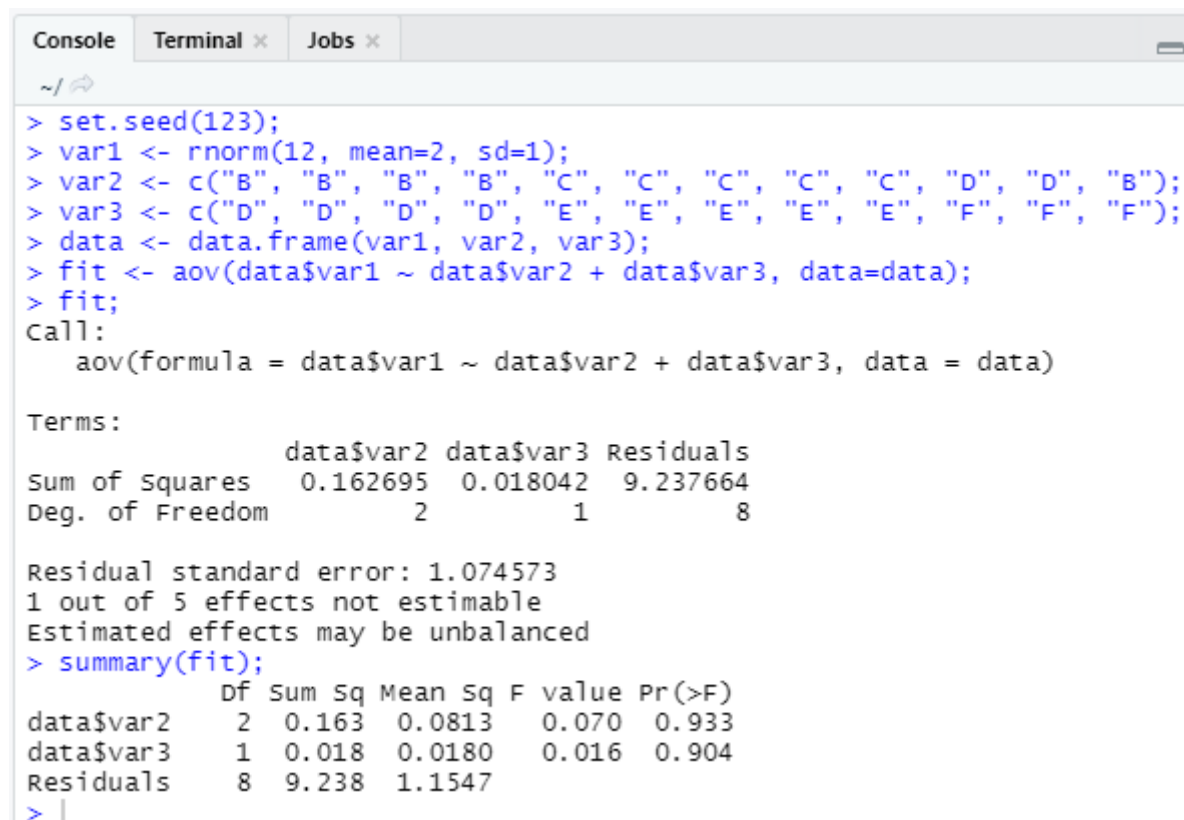
La valeur p est plus grande que 0.05, d'où l'échec à rejeter l'hypothèse nulle. Les deux variables ont bien statistiquement la même moyenne.

#### 4.10.2 Two-Way ANOVA

Si nous disposons de deux variables indépendantes.

Soient les données :

```
set.seed(123);
var1 <- rnorm(12, mean=2, sd=1);
var2 <- c("B", "B", "B", "B", "C", "C", "C", "C", "C", "D", "D", "B");
var3 <- c("D", "D", "D", "D", "E", "E", "E", "E", "E", "F", "F", "F");
data <- data.frame(var1, var2, var3);
fit <- aov(data$var1 ~ data$var2 + data$var3, data=data);
fit;
summary(fit);
```



```
> set.seed(123);
> var1 <- rnorm(12, mean=2, sd=1);
> var2 <- c("B", "B", "B", "B", "C", "C", "C", "C", "C", "D", "D", "B");
> var3 <- c("D", "D", "D", "D", "E", "E", "E", "E", "E", "F", "F", "F");
> data <- data.frame(var1, var2, var3);
> fit <- aov(data$var1 ~ data$var2 + data$var3, data=data);
> fit;
Call:
aov(formula = data$var1 ~ data$var2 + data$var3, data = data)

Terms:
            data$var2 data$var3 Residuals
Sum of Squares    0.162695    0.018042    9.237664
Deg. of Freedom         2         1         8

Residual standard error: 1.074573
1 out of 5 effects not estimable
Estimated effects may be unbalanced
> summary(fit);
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
data\$var2	2	0.163	0.0813	0.070	0.933
data\$var3	1	0.018	0.0180	0.016	0.904
Residuals	8	9.238	1.1547		

```
> |
```

La moyenne de var1 et la moyenne de var2 ont des valeurs de p de 0.933, ce qui est supérieur à 0,05. Ainsi, vous ne parvenez pas à rejeter l'hypothèse nulle selon laquelle la moyenne var1 est la même que var2. L'hypothèse nulle est vraie à l'intervalle de confiance de 95 %. La moyenne var1 et la moyenne var3 ont des valeurs de p de 0.904, ce qui est supérieur à 0,05. Par conséquent, vous ne rejetez pas l'hypothèse nulle selon laquelle la moyenne var1 est identique à la moyenne var3. L'hypothèse nulle est vraie à 95 % de confiance intervalle.

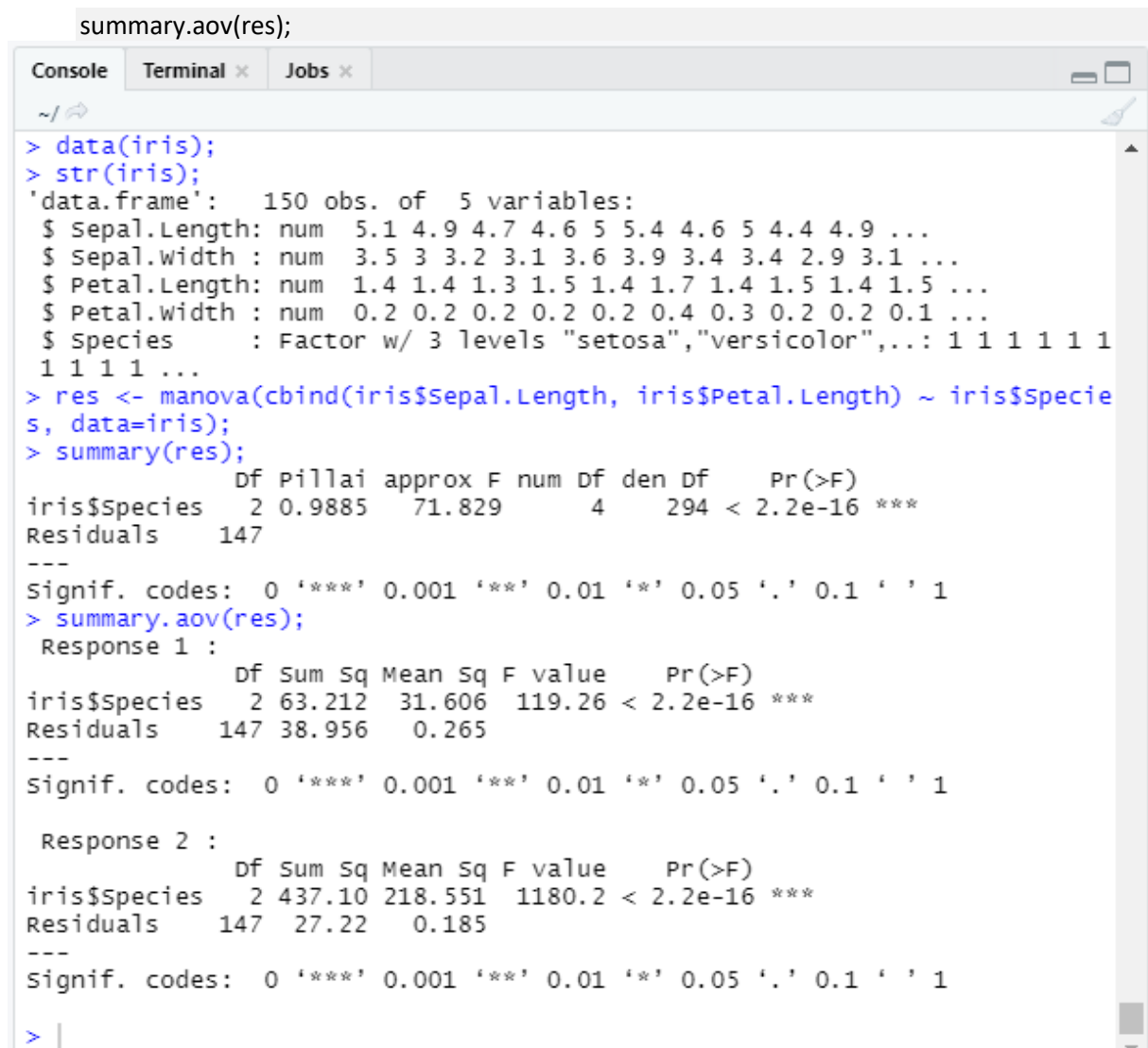
#### 4.10.3 MANOVA

C'est un test pour raisonner sur plusieurs variables à fois.

Soient les données de Iris :

```
data(iris);
str(iris);
res <- manova(cbind(iris$Sepal.Length, iris$Petal.Length) ~ iris$Species, data=iris);
summary(res);
```

```
summary.aov(res);
```



```
> data(iris);
> str(iris);
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1
1 1 1 ...
> res <- manova(cbind(iris$Sepal.Length, iris$Petal.Length) ~ iris$Species, data=iris);
> summary(res);
              Df Pillai approx F num Df den Df    Pr(>F)
iris$Species   2 0.9885    71.829      4    294 < 2.2e-16 ***
Residuals    147
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> summary.aov(res);
Response 1 :
              Df Sum Sq Mean Sq F value    Pr(>F)
iris$Species   2 63.212   31.606  119.26 < 2.2e-16 ***
Residuals    147 38.956    0.265
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Response 2 :
              Df Sum Sq Mean Sq F value    Pr(>F)
iris$Species   2 437.10  218.551  1180.2 < 2.2e-16 ***
Residuals    147  27.22    0.185
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> |
```

Permettant de répondre au test d'hypothèse :

- $H_0: \mu_{Sepal.Length} = \mu_{Petal.Length} = \mu_{Species}$
- $H_a: \mu_{Sepal.Length} \neq \mu_{Petal.Length} \neq \mu_{Species}$

La valeur p est de 2,2e-16, soit moins de 0,05. Par conséquent, vous rejetez l'hypothèse nulle : les moyennes sont bien différentes.

## 4.11 Test non-paramétrique

Ce sont des techniques pour établir un test d'hypothèse sans supposer une quelconque distribution sur les données.

### 4.11.1 Wilcoxon

```
install.packages("random");
```

```
library(random);
var1 <- randomNumbers(n=100, min=1, max=1000, col=1);
var2 <- randomNumbers(n=100, min=1, max=1000, col=1);
var3 <- randomNumbers(n=100, min=1, max=1000, col=1);
data <- data.frame(var1[,1], var2[,1], var3[,1]);
data;
```



```
wilcox.test(data[,1], mu=0, alternatives="two.sided");
```

```
74 install.packages( random );
75 library(random);
76 var1 <- randomNumbers(n=100, min=1, max=1000, col=1);
77 var2 <- randomNumbers(n=100, min=1, max=1000, col=1);
78 var3 <- randomNumbers(n=100, min=1, max=1000, col=1);
79 data <- data.frame(var1[,1], var2[,1], var3[,1]);
80 data;
81 wilcox.test(data[,1], mu=0, alternatives="two.sided");
82
83
```

81:55 (Top Level) R Script

Console Terminal Jobs

```
~ /
99      799      102      504
100     204      781      977
> wilcox.test(data[,1], mu=0, alternatives="two.sided");

      wilcoxon signed rank test with continuity correction

data: data[, 1]
V = 5050, p-value < 2.2e-16
alternative hypothesis: true location is not equal to 0
> |
```

La valeur médiane est bien différente de zéro.

#### 4.11.2 Test de Wilcoxon-Mann-Whitney

Soient les données et la manip :

```
var1 <- randomNumbers(n=100, min=1, max=1000, col=1);
var2 <- randomNumbers(n=100, min=1, max=1000, col=1);
var3 <- randomNumbers(n=100, min=1, max=1000, col=1);
data <- data.frame(var1[,1], var2[,1], var3[,1]);
data;
wilcox.test(data[,1], data[,2], correct=FALSE);
```

```

83 var1 <- randomNumbers(n=100, min=1, max=1000, col=1);
84 var2 <- randomNumbers(n=100, min=1, max=1000, col=1);
85 var3 <- randomNumbers(n=100, min=1, max=1000, col=1);
86 data <- data.frame(var1[,1], var2[,1], var3[,1]);
87 data;
88 wilcox.test(data[,1], data[,2], correct=FALSE);
89
90

```

83:1 (Top Level) R Script

Console Terminal x Jobs x

```

~/
99      30      84      72
100     40      37      423
      223     116     507
> wilcox.test(data[,1], data[,2], correct=FALSE);

      wilcoxon rank sum test

data: data[, 1] and data[, 2]
W = 5113.5, p-value = 0.7815
alternative hypothesis: true location shift is not equal to 0
> |

```

Pas de différence entre les deux médiane (plutôt égale).

#### 4.11.3 Formule Wilcoxon

```

# S3 method for default
wilcox.test(x, y = NULL,
            alternative = c("two.sided", "less", "greater"),
            mu = 0, paired = FALSE, exact = NULL, correct = TRUE,
            conf.int = FALSE, conf.level = 0.95, ...)
# S3 method for formula
wilcox.test(formula, data, subset, na.action, ...)

```

#### 4.11.4 Kruskal-Wallis

Permet de réaliser le test :

$$H_0 : \mu_0 = \mu_1 = \mu_2 = \dots = \mu_k$$

$$H_a : \mu_0 \neq \mu_k$$

où  $\mu$  est la médiane.

```

data("airquality");
str(airquality);
summary(airquality);
kruskal.test(airquality$Ozone ~ airquality$Month);

```

```
90 data("airquality");
91 str(airquality);
92 summary(airquality);
93 kruskal.test(airquality$Ozone ~ airquality$Month);
94
95
```

93:51 (Top Level) R Script

Console Terminal x Jobs x

~/airquality

Max. :9.000 Max. :31.0

```
> kruskal.test(airquality$Ozone ~ airquality$Month);

kruskal-wallis rank sum test

data: airquality$Ozone by airquality$Month
kruskal-wallis chi-squared = 29.267, df = 4, p-value =
6.901e-06

> |
```

La valeur p est de 6,901e-06, soit moins de 0,05. Par conséquent, vous rejetez la hypothèse nulle. Il existe des différences significatives dans la médiane pour la première variable et la deuxième variable. L'hypothèse alternative est vraie à l'intervalle de confiance de 95 %.

## 5 Modèles statistiques

### 5.1 Régressions

#### 5.1.1 Régression linéaire

```
set.seed(123);
x <- rnorm(100, mean=1, sd=1);
y <- rnorm(100, mean=2, sd=2);
data <- data.frame(x, y);
mod <- lm(data$y ~ data$x, data=data);
mod;
summary(mod);
```

```
95 set.seed(123);
96 x <- rnorm(100, mean=1, sd=1);
97 y <- rnorm(100, mean=2, sd=2);
98 data <- data.frame(x, y);
99 mod <- lm(data$y ~ data$x, data=data);
100 mod;
101 summary(mod);
102 |
103
```

102:1 (Top Level) R Script

Console Terminal x Jobs x

~/

Estimate Std. Error t value Pr(>|t|)

(Intercept)	1.8993	0.3033	6.261	1.01e-08 ***
data\$x	-0.1049	0.2138	-0.491	0.625

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

Residual standard error: 1.941 on 98 degrees of freedom  
Multiple R-squared: 0.002453, Adjusted R-squared: -0.007726  
F-statistic: 0.241 on 1 and 98 DF, p-value: 0.6246

> |

Lorsque la valeur de p (intercept) est inférieure à 0,05, le modèle est significatif pour y.

La signification est indiquée par le nombre de \*. Le x a une valeur p de 0,625, qui est supérieure à 0,05, donc il n'y a pas de signification avec la variable y. L'hypothèse nulle est vraie à l'intervalle de confiance à 95 %.

### 5.1.2 Régression multiples

```
set.seed(123);
x <- rnorm(100, mean=1, sd=1);
x2 <- rnorm(100, mean=2, sd=5);
y <- rnorm(100, mean=2, sd=2);
data <- data.frame(x, x2, y);
mod <- lm(data$y ~ data$x + data$x2, data=data);
mod;
```

```

103 set.seed(123);
104 x <- rnorm(100, mean=1, sd=1);
105 x2 <- rnorm(100, mean=2, sd=5);
106 y <- rnorm(100, mean=2, sd=2);
107 data <- data.frame(x, x2, y);
108 mod <- lm(data$y ~ data$x + data$x2, data=data);
109 mod;
110 summary(mod);
111
112

```

110:14 (Top Level) ↕

R Script ↕

Console

Terminal x

Jobs x

~/

```

> x <- rnorm(100, mean=1, sd=1);
> x2 <- rnorm(100, mean=2, sd=5);
> y <- rnorm(100, mean=2, sd=2);
> data <- data.frame(x, x2, y);
> mod <- lm(data$y ~ data$x + data$x2, data=data);
> mod;

```

```

Call:
lm(formula = data$y ~ data$x + data$x2, data = data)

```

Coefficients:

```

(Intercept)      data$x      data$x2
  2.517425    -0.266343    0.009525

```

```
> summary(mod);
```

```

Call:
lm(formula = data$y ~ data$x + data$x2, data = data)

```

Residuals:

```

      Min       1Q   Median       3Q      Max
-3.7460 -1.3215 -0.2489  1.2427  4.1597

```

Coefficients:

```

              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.517425   0.305233   8.248 7.97e-13 ***
data$x       -0.266343   0.209739  -1.270   0.207
data$x2       0.009525   0.039598   0.241   0.810
---

```

```

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

```

```

Residual standard error: 1.903 on 97 degrees of freedom
Multiple R-squared:  0.01727,    Adjusted R-squared:  -0.00299
F-statistic: 0.8524 on 2 and 97 DF,  p-value: 0.4295

```

```
> |
```

$Y = -0.266343x + 0.009525x_2 + 2.517425$

## 6 Cas d'études

### 6.1 Cas : test d'hypothèse sur la moyenne

(st-l-inf-tests.pdf)

Des relevés effectués pendant de nombreuses années ont permis d'établir que le niveau naturel des pluies dans une région X en millimètres par an suit une loi normale  $N(600, 100^2)$ . Des entrepreneurs, surnommés "faiseurs de pluie", prétendaient pouvoir augmenter de 50mm le niveau moyen de pluie, ceci par insémination des nuages au moyen d'iodure d'argent. Leur procédé fut mis à l'essai entre 1951 et 1959 et on releva les hauteurs de pluies suivantes :

Année	1951	1952	1953	1954	1955	1956	1957	1958	1959
mm	510	614	780	512	501	534	603	788	650

Que pouvait-on en conclure ?

#### Solution :

Voici les hypothèses sur la comparaison des deux expériences (habituelle et nouvelle) :

$$H_0 : \mu_1 = 610.22$$

$$H_a : \mu_1 \neq 610.22$$

où  $\mu_1$  est la moyenne de la distribution habituelle d'une moyenne 600 et d'écart type 10000.

On veut vérifier si les deux moyennes concordent (l'habituelle et la nouvelle).

```
pl=c(510, 614, 780, 512, 501, 534, 603, 788, 650);  
set.seed(123);  
var1 <- rnorm(600, mean=600, sd=100);  
t.test(var1, mu=mean(pl), alternative="two.sided");
```

Résultat : p-value = 0.04198

Si on prend  $\alpha = 0.01$ . D'où  $H_0$  n'est pas rejeté. Statistiquement  $\mu_1 = 610.22$  et donc le procédé n'a pas amélioré la pluviométrie.

### 6.2 Cas : test d'hypothèse sur des données gaussiennes

(M10.pdf)

Les plantations d'eucalyptus est une base de l'économie régionale de plusieurs pays. Un chercheur en agronomie veut vérifier si un nouveau clone d'eucalyptus (le no 51n78) est assez productif pour que son exploitation soit rentable. L'exploitation est rentable si le rendement moyen est de plus de  $2\text{m}^3$  de bois en moyenne par arbre après 10 ans. Pour vérifier si le clone proposé est rentable, 57 arbres ont été plantés et les résultats après 10 ans sont les suivants :  $\bar{x} = 2.2\text{m}^3$  et  $s = 1\text{m}^3$ . Des données biologiques montrent que le rendement d'un arbre en  $\text{m}^3$  est une mesure dont la distribution est très près d'une loi normale. On veut vérifier l'hypothèse selon laquelle le clone d'eucalyptus no 51n78 est rentable ce qui se traduit par les hypothèses statistiques (2)

$$H_0 : \mu = 2$$

$$H_1 : \mu > 2$$

où  $\mu$  est l'espérance de la v.a. qui donne la production de bois en  $\text{m}^3$  du clone d'eucalyptus no 51n78.

Pour mener à bien ce test, on simule un échantillon taille 57 d'une variable suivant la loi normale de moyenne 2 et de variance celle de l'échantillon donné.

#### Solution :

```
set.seed(123);  
var1 <- rnorm(57, mean=2, sd=1*sqrt(57));  
t.test(var1, mu=2, alternative="greater");
```

p-value = 0.335

D'où  $H_0$  n'est pas rejeté. D'où l'eucalyptus no 51n78 n'est pas rentable.

### 6.3 Cas : test d'hypothèse sur deux échantillons appariés

(M10.pdf)

Une chercheuse en biologie veut déterminer l'effet de la congélation à -80°C sur le nombre de bactéries dans une culture. L'hypothèse étant que la congélation à une telle température devrait détruire une certaine proportion des bactéries. Pour faire l'expérimentation, la chercheuse observe 30 cultures pour déterminer le nombre de bactéries présentes dans la culture puis chaque culture est réfrigérée à -80°C pendant 1 semaine pour ensuite être décongelée. Le nombre de bactéries est alors comptabilisé dans chaque culture décongelée. Si on observe les données standardisées ( $\times 10^2$ ) suivantes :

Culture	1	2	3	4	5	6	7	8	9	10
Avant	57	45	70	62	47	55	50	61	45	71
Après	55	50	60	59	50	49	43	55	49	66

Culture	11	12	13	14	15	16	17	18	19	20
Avant	45	66	69	54	47	53	58	67	59	47
Après	40	59	70	49	53	45	50	63	55	50

Culture	21	22	23	24	25	26	27	28	29	30
Avant	57	64	73	48	55	59	62	61	51	54
Après	53	56	63	45	59	51	56	62	48	51

peut-on déterminer si la congélation a détruit des bactéries ?

N.B. Les hypothèses statistiques sont

$$H_0 : \mu_d = 0$$

$$H_1 : \mu_d > 0$$

**Solution :**

av = c(57, 45, 70, 62, 47, 55, 50, 61, 45, 71, 45, 66, 69, 54, 47, 53, 58, 67, 59, 47, 57, 64, 73, 48, 55, 59, 62, 61, 51, 54);

ap = c(55, 50, 60, 59, 50, 49, 43, 55, 49, 66, 40, 59, 70, 49, 53, 45, 50, 63, 55, 50, 53, 56, 63, 45, 59, 51, 56, 62, 48, 51);

t.test(av, ap, alternative = "greater", paired = TRUE);

p-value = 0.0002796

Donc  $H_0$  est rejeté, et donc la congélation est efficace.

### 6.4 Cas : test d'hypothèse sur deux échantillons appariés

(M10.pdf)

On pense que les automobilistes sont plus soucieux de l'usure des pneus d'hiver que des pneus d'été.

Sur un échantillon de 15 automobilistes on remarque les usures (en %) pour les pneus d'été :

45, 48, 38, 46, 37, 39, 29, 46, 47, 43, 31, 43, 37, 27, 49

L'usure des pneus d'hiver a été mesurée chez les mêmes automobilistes :

41, 37, 44, 28, 28, 32, 40, 41, 31, 30, 30, 49, 35, 35, 37

Peut-on dire que les automobilistes ont des pneus d'hiver moins usés au niveau 10% en considérant que la mesure de l'usure est une v.a. approximativement normale ?

N.B. Les hypothèses statistiques sont

$$H_0 : \mu_d = 0$$

$$H_1 : \mu_d < 0$$

**Solution :**

av1 = c(45, 48, 38, 46, 37, 39, 29, 46, 47, 43, 31, 43, 37, 27, 49);

ap1 = c(41, 37, 44, 28, 28, 32, 40, 41, 31, 30, 30, 49, 35, 35, 37);

t.test(av1, ap1, alternative = "greater", paired = TRUE);

p-value = 0.03838

Si on prend  $\alpha = 0.05$ . D'où  $H_0$  est rejeté, et donc il y a usure.

### 6.5 Cas :

(CoursMasterIDjelfa.pdf)

On veut savoir si le temps écoulé depuis la vaccination contre la petite vérole a ou non une influence sur le degré de gravité de la maladie lorsqu'elle apparaît. Les patients sont divisés en trois catégories selon la gravité de leur maladie : légère (L), moyenne (M), ou grave (G) et en trois autres quant à la durée écoulée depuis la vaccination : moins de 10 ans (A), entre 10 et 25 ans (B), plus de 25 ans (C). Les résultats d'une observation portant sur  $n = 1574$  malades sont les suivants :

Degré de gravité Y de la maladie	Durée X écoulée depuis la vaccination			Total
	A	B	C	
G	1	42	230	273
M	6	114	347	467
L	23	301	510	834
Total	30	457	1087	1574

Mener à bien un test où :

- $H_0$  : la durée écoulée depuis la vaccination et le degré de gravité de la maladie sont indépendantes",
- $H_1$  : la durée écoulée depuis la vaccination et le degré de gravité de la maladie sont liées".

C'est dans ce genre de situations, que le test d'indépendance de Khi-Deux peut intervenir.

### 6.6 Cas

(An Introduction to Biostatistic-Waveland Press, Inc. (2015).pdf)

### 6.7 Cas

(Applications of Hypothesis Testing for Environmental Science-Elsevier (2020).pdf)

### 6.8 Cas : quartiles

(ExBiostatistiquesous.pdf)

Soit la donnée :

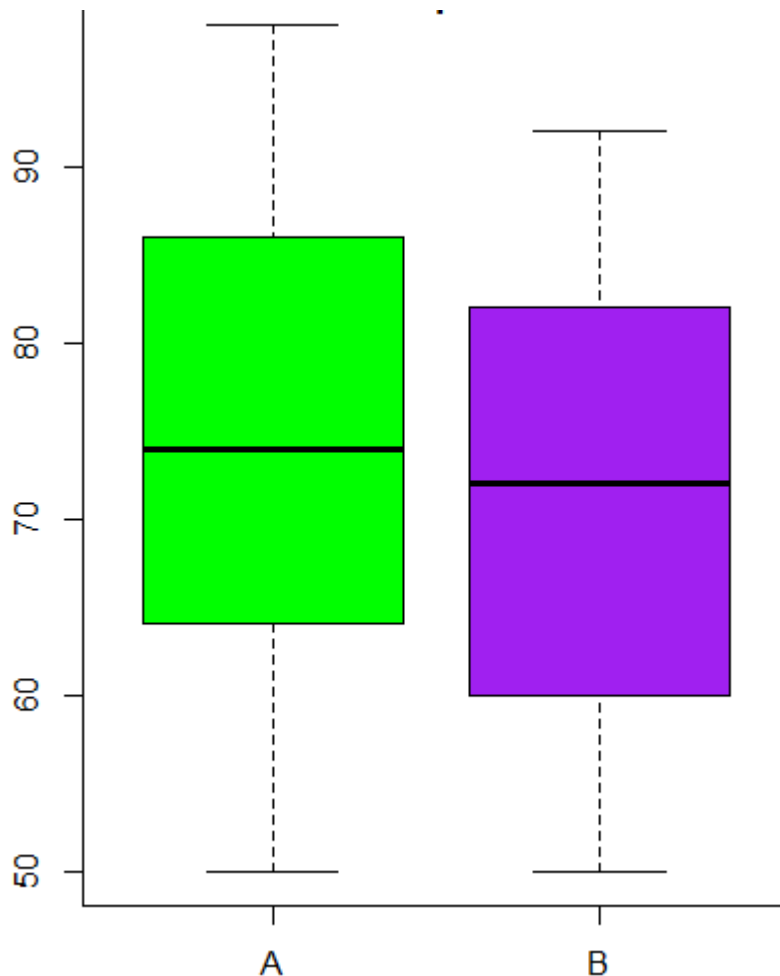
	Poids					
	Moyenne	Maximun	Minimun	Médiane	25 centile	75 centile
Femme	74.4	98	50	74	64	86
Homme	71.33	92	50	72	60	82

Tracer les boîtes à moustaches (quartiles) associées à ces données ?

**Solution :**

```
A <- c(50, 64, 74, 86, 98);
B <- c(50, 60, 72, 82, 92);
data <- data.frame(A, B);
boxplot(data, main="boxplot", notch=FALSE, varwidth=TRUE, col=c("green", "purple"));
```





### 6.9 Cas : Test d'indépendance

Dans l'enquête sur l'ensemble de données intégré, la colonne Fumée enregistre l'habitude de fumer des élèves, tandis que la colonne Exer enregistre leur niveau d'exercice physique (sport). Les valeurs autorisées dans Smoke sont "Heavy", "Regul" (régulièrement), "Occas" (occasionnellement) et "Jamais". Quant à Exer, ce sont "Freq" (fréquemment), "Some" et "None".

Nous pouvons comparer l'habitude de fumer des étudiants avec le niveau d'exercice avec la fonction de table dans R. Le résultat est appelé le tableau de contingence des deux variables.

```
library(MASS) # load the MASS package
tbl = table(survey$Smoke, survey$Exer)
tbl
```

Testez l'hypothèse si l'habitude de fumer des élèves est indépendante de leur niveau d'exercice au seuil de signification de 0,05.

**Solution :**

```
chisq.test(tbl)
```

Comme la valeur de p 0,4828 est supérieure au niveau de signification de 0,05, nous ne rejetons pas l'hypothèse nulle selon laquelle l'habitude de fumer est indépendante du niveau d'exercice des étudiants.

### 6.10 Cas : Test d'indépendance

(ExBiostatistiquesous.pdf)

Une étude a été réalisée sur le cancer de la gorge. Pour cela, une population de 1000 personnes a été interrogée. Les résultats obtenus sont données dans le tableau suivant :

	Atteint du cancer de la gorge	Non atteint du cancer de la gorge
Fumeur	344	258
Non-fumeur	160	238

- Préciser le type des deux caractères étudiés ?
- Peut-on rejeter au risque 5% l'hypothèse d'indépendance des deux caractères ?

**Solution :**

```
tab <- matrix(c(344, 258, 160, 238), ncol=2, byrow=TRUE) ;
colnames(tab) <- c('OuiCa', 'NonCa') ;
rownames(tab) <- c('Fum', 'NonFum') ;
tab <- as.table(tab) ;
chisq.test(tab);
```

p-value = 2.214e-07

En reprenant les hypothèses du test Chi-Square :

- $H_0$  : les deux variables sont indépendantes
- $H_a$  : les deux variables sont non indépendantes

Avec p-value = 2.214e-07, on rejette  $H_0$  d'où les deux variables sont dépendantes. On peut affirmer qu'il y a une relation entre « fumer » et « avoir le cancer de la gorge ».

### 6.11 Cas : analyse de plusieurs moyennes

(ExBiostatistiquesous.pdf)

Soit la hauteur des arbres plantés dans trois forêts :

Forêt 1	Forêt 2	Forêt 3
23,3	18,9	22,5
24,4	21,1	22,9
24,6	21,1	23,7
24,9	22,1	24,0
25,0	22,5	24,0
26,2	23,5	24,5

Etablir une analyse ANOVA à un facteur et interpréter ces résultats ?

**Solution :**

```
var = c("F1", "F1", "F1", "F1", "F1", "F1", "F2", "F2", "F2", "F2", "F2", "F2", "F3", "F3", "F3", "F3",
"F3", "F3");
haut <- c(23.3, 24.4, 24.6, 24.9, 25.0, 26.2, 18.9, 21.1, 21.1, 22.1, 22.5, 23.5, 22.5, 22.9, 23.7,
24.0, 24.0, 24.5);
data <- data.frame(haut, var);
fit <- aov(haut ~ var, data=data);
summary(fit);
```

$\Pr(>F) = p\text{-value} = 0.000765$

On rappelle les hypothèses de ANOVA :

- $H_0 : \mu_1 = \dots = \mu_L$
- $H_a : \mu_1 \neq \mu_m$

Puisque  $p\text{-value} = 0.000765$ , on peut rejeter  $H_0$ . Les moyennes sont bien différentes.

## 6.12 Cas : analyse de plusieurs moyennes sur deux catégories

(ExBiostatistiquesous.pdf)

Expérience : des secrétaires tapent un texte pendant 5 minutes sur différentes machines à écrire.

L'expérience est répétée le lendemain.

- Variable quantitative : nombre moyen de mots tapés en une minute.

Machines à écrire	Secrétaires				
	1	2	3	4	5
1	33	31	34	34	31
	36	31	36	33	31
2	32	37	39	33	35
	35	35	36	36	36
3	37	35	34	31	37
	39	35	37	35	39
4	29	31	33	31	33
	31	33	34	27	33

Y a-t-il un effet ?

**Solution :**

```
va1 = c("M1", "M1", "M1", "M1", "M1", "M1", "M1", "M1", "M1", "M1",
        "M2", "M2", "M2", "M2", "M2", "M2", "M2", "M2", "M2", "M2",
        "M3", "M3", "M3", "M3", "M3", "M3", "M3", "M3", "M3", "M3",
        "M4", "M4", "M4", "M4", "M4", "M4", "M4", "M4", "M4", "M4");
va2 = c("J1", "J1", "J1", "J1", "J1", "J2", "J2", "J2", "J2", "J2",
        "J1", "J1", "J1", "J1", "J1", "J2", "J2", "J2", "J2", "J2",
        "J1", "J1", "J1", "J1", "J1", "J2", "J2", "J2", "J2", "J2",
        "J1", "J1", "J1", "J1", "J1", "J2", "J2", "J2", "J2", "J2");
lettres = c(33, 31, 34, 34, 31,
            36, 31, 36, 33, 31,
            32, 37, 39, 33, 35,
            35, 35, 36, 36, 36,
            37, 35, 34, 31, 37,
            39, 35, 37, 35, 39,
            29, 31, 33, 31, 33,
            31, 33, 34, 27, 33);
data <- data.frame(lettres, va1, va2);
fit <- aov(lettres ~ va1 + va2 + va1:va2, data=data);
summary(fit);
```

On obtient le résultat :

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
va1	3	128.1	42.70	9.013	0.000179 ***
va2	1	8.1	8.10	1.71	0.200333
va1:va2	3	6.1	2.03	0.429	0.733465

Soit  $\alpha = 0.05$ .

Puisque p-value = 0.000179 sur va1, on peut affirmer que les machines à écrire ont des moyennes différentes.

Puisque p-value = 0.200333 sur va2, on peut affirmer que les 2 jours ont les mêmes moyennes.

Puisque p-value = 0.733465 sur va1 :va2, on peut affirmer que globalement les deux machines à écrire ont les mêmes moyennes entre les deux jours.

### 6.13 Cas : Wilcoxon

(slides\_tests\_non\_param.pdf)

TEST sur la différence de rendement de graines de maïs.

*Rendement du maïs pour différentes graines*

Ordinaires	Séchées
1903	2009
1935	1915
1910	2011
2496	2463
2108	2180
1961	1925
2060	2122
1444	1482
1612	1542
1316	1443
1511	1535

**Solution :**

```
ordi = c(1903, 1935, 1910, 2496, 2108, 1961, 2060, 1444, 1612, 1316, 1511);
sech = c(2009, 1915, 2011, 2463, 2180, 1925, 2122, 1482, 1542, 1443, 1535);
wilcox.test(ordi, sech, paired=T)
```

p-value = 0.123, on peut affirmer que les deux médianes sont statistiquement égales.

N.B. / Calcul de Wilcoxon :

```
diff = ordi - sech;
diffabs<-abs(diff)
r<-rank(diffabs)
si<-sign(diff)
rs<-r*si
T<-min(sum(rs[rs>0]),-sum(rs[rs<0]))
seuils<-c(qsignrank(0.025,11),qsignrank(0.025,11,lower.tail=F))
Pvaleur<-2*psignrank(T,11)
```