TP n^o 0 : Premiers pas avec R

L'objectif de ce TP est de vous familiariser avec les objets et les commandes du logiciels R. R est un logiciel libre et collaboratif de statistique qui s'enrichit au fur et à mesure de l'apport gracieux de chercheurs du monde entier. Il est devenu un outil incontournable de statistique et de visualisation de données, tant dans le monde universitaire que dans le monde de l'entreprise.

I. R est une calculatrice scientifique

R permet de faire les opérations de calcul élémentaire. Essayez les commandes suivantes :

```
4*5-2^4
1+14/4
```

R permet de faire des calculs plus élaborés. Il utilise pour cela des fonctions. Plusieurs fonctions prédéfinies sont disponibles. Que font les fonctions suivantes ?

```
sqrt(5)
abs(-4)
log(1)
cos(pi)
exp(1)
round(pi,2)
```

II. R est un langage de programmation

I. Création de variables

```
x<-2.5
x
y<-2*x
y
y<-1+log(y)^2
y
Y<-floor(y)       # partie entière de x
Y
```

Dans la première ligne, On a créé une variable de nom x contenant la valeur 2.5. La flèche, obtenu en tapant $<$ et $-$ est appelée opérateur d'affectation. Cet opérateur dit que la variable x doit contenir la valeur 2.5. Si cette variable n'existe pas, elle est créée. Si elle existe, son contenu est remplacé par la nouvelle valeur. Pour rappeler le contenu d'une variable, il suffit de taper son nom (ligne 2).

II. Création d'un vecteur

Il y a différentes manières de créer des vecteurs dans R. Étudiez les lignes de commande suivantes

```
vec<-c(-1,5,2,0.5,-5)
vec
vec1<-c(3,10,vec)
vec1
length(vec1)
rep(c(3,5),2)
1:20
5:1
seq(from=1, to=16, by=3)
seq(from=1, to=2, length=10)
```

III. Opérations sur les vecteurs

Créer deux vecteurs x et y quelconques et de même taille. Que font les commandes suivantes ?

```
2*x+1
exp(y)
x+y
x*y
sum(x)
cumsum(x)
which(y<0)
which.min(y)
sort(y)
```

IV. Accès aux éléments d'un vecteur

On peut accéder aux éléments d'un vecteur grâce à l'opérateur "[]".

```
x<-c(3,-1,5,7,0,3,2,-9)
y<-c("rouge","noir","vert")            # vecteur de chaines de charactères
x[1:3]
x[-1]
x[c(2,4)]
x[x>0 & x<=5]
x[1]<-1.25
x
z=y[y!="rouge"]
z
```

V. Notion de dataframe

Tout comme les vecteurs, les dataframes sont des objets, mais plus élaborés et spécialement désignés pour le stockage de données. Il s'agit plus ou moins d'un tableau à 2 dimensions (une matrice), mais avec en plus des noms de colonnes, des noms de lignes, etc. Très généralement, et ce sera toujours le cas en ce qui nous concerne, les lignes d'un dataframe seront des individus (dans l'exemple ci-dessous ce sont des sujets) et les colonnes des variables (dans l'exemple ci-dessous ce sont le poids, la taille et l'Indice de Masse Corporelle des sujets).

```
poids= c(65,82,45,63,70)
taille= c(1.75,1.78,1.52,1.57,1.80)
IMC= poids/(taille^2)                  # IMC est l'indice de masse corporelle
data=cbind(poids,taille,IMC)
# cbind permet de coller des colonnes les unes à la suite des autres
# et rbind permet de coller des lignes les unes à la suite des autres
```

De la même manière que pour les vecteurs, les crochets permettent d'aller chercher des éléments d'une matrice ou d'un tableau. Il y a alors 2 paramètres à définir, le premier pour les lignes et le second pour les colonnes à sélectionner (les 2 séparés par une virgule). Si rien n'est préciser pour le premier paramètre, cela signifie que l'on prend toutes les lignes. Si rien n'est préciser pour le deuxième paramètre, cela signifie que l'on prend toutes les colonnes.

```
data[3,2]       # taille du troisième sujet
data[ ,1]       # poids de tous les sujets
dim(data)
nrow(data)
ncol(data)
row.names(data)
col.names(data)
```

VI. Conditions et boucles

- *while* : une boucle while est utilisée pour répéter une instruction jusqu'à ce qu'une condition spécifique soit remplie.

```
while (condition) instruction
```

```
i <- 1
while (i < 6) {
print(i)
i = i+1
}
```

- *for* : une boucle for permet de répéter une instruction un nombre de fois défini a priori.

```
for (variable in vecteur) instruction
```

```
factorielle <-1
for (i in 1:10)
factorielle <- factorielle*i   # on calcule 10! (factorielle de 10)
```

- *If* & *else* : les instructions if et else fournissent un moyen d'exécuter du code si une condition est respectée ou non.

```
# Première forme (pas de code si condition == FALSE)
if (condition) instruction
# Seconde forme
if (condition) instruction si vrai else instruction si faux
```

```
x <- c(2,5,3,9,8,11,6)
count <- 0
for (val in x) {
if(val %% 2 == 0)  count = count+1
}
print(count)

if(count > 3) x <- prod(x) else x <- sum(x)
```

VII. Programmer une fonction

Avec R, on peut programmer notre propre fonction.

```
nom_de_fonction <- function(arguments) {
    instructions
}
```

```
# fonction calculant P(X=k) où X est une v.a qui suit une loi B(n,p)
proba.binomiale <- function(n,p,k) {
combinaison <- factorial(n)/( factorial(k)*factorial(n-k) )
proba <- combinaison * p^k * (1-p)^(n-k)
return(proba)
}

proba.binomiale(10,0.75,8)    #P(X=8) où X suit B(10,0.75)
```

III. R est un langage pour les statistiques

I. Fonctions statistiques

R possède plusieurs fonctions statistiques. Que font les fonctions suivantes ?

```
min(x)
max(x)
mean(x)
sd(x)
quantile(x,0.25)
median(x)
var(x)
summary(x)
```

II. Représentations graphiques

La fonction plot
La fonction plot est la fonction générique de graphe.

```
x=seq(-2,2,by=0.1)
y=exp(x)
plot(x, y)
```

Il est possible de paramétrer l'affichage obtenu (couleurs, etc). Voici un bref aperçu des options de la fonction plot, vous pouvez regarder l'aide (en tapant ?plot) pour plus de détails. Nous aurons de toute façon l'occasion de revenir sur l'utilisation de cette fonction.

- **type** (chaîne de caractères) "p" pour points, "l" pour lignes, "b" pour les deux (both) ;
- **main** (chaîne de caractères) un titre pour le graphique ;
- **sub** (chaîne de caractères) un sous-titre pour le graphique ;
- **xlab** (chaîne de caractères) un titre pour l'axe des abscisses ;
- **ylab** (chaîne de caractères) un titre pour l'axe des ordonnées ;
- **col** (vecteur de chaînes de caractères) la ou les couleur(s) à utiliser ;
- etc ...

```
plot(x, y, pch = 1, type="l", col = "red", main = "Courbe de la fct exp",
xlab = "abscisse", ylab = "ordonnée")
```