**EILCO** **2023-2024**
**Cycle Ingénieur 1ère année**

**TP1 : Introduction à R**

# 1 Introduction

**Le logiciel R** R est un logiciel de statistique distribué gratuitement par le CRAN (Comprehensive R Archive Network) à l'adresse suivante http ://cran.r-project.org/. Le logiciel R est devenu un outil incontournable de statistique et de visualisation de données, aussi bien dans le monde universitaire que dans le monde de l'entreprise. L'un des avantages de R est la richesse de packages développés par les utilisateurs et développeurs qu'on peut installer pour augmenter ses capacités dans des domaines très variés de la statistique.

**L'environnement RStudio** RStudio est un environnement de développement gratuit, libre et multiplateforme pour R. La fenêtre de RStudio se divise en quatre sous-fenêtres, dans le sens des aiguilles d'une montre :

1. un éditeur de texte pour les scripts :
2. l'espace de travail ou d'historique des commandes ;
3. le navigateur de fichiers, graphiques, packages, documentations ;
4. la console R.

On peut taper directement les codes dans la console (sous-fenêtre 4), mais le mieux est de les taper dans un script et d'appuyer sur l'onglet *run* (sous-fenêtre 1). Pour mettre fin à un programme, appuyer sur le balai (sous-fenêtre 4). La sous-fenêtre 2 permet de visualiser l'historique et l'environnement (données actuelles). Pour supprimer les données de l'environnement, taper dans la console `rm(list=ls())` ; et pour supprimer l'historique, appuyer sur le balai. Pour utiliser un package, il faut d'abord l'installer, puis le charger (sous-fenêtre 3). Pour consulter l'aide, aller dans la sous-fenêtre 3 ou/et rechercher des informations directement sur Google.

La barre *session*, suivi de l'onglet *set working directory*, permet de choisir le dossier dans lequel travailler. La commande `getwd()` permet de visualiser dans quel dossier on travaille.

# 2 Bases du langage

## 2.1 Commandes

Taper les expressions suivantes :

```
cos(pi)
x<-3+4; x
x==7
x!=7
2*6; 12/3; sqrt(16)
```

## 2.2 Modes, longueurs et classes de données

Dans R, tout est un objet. Le mode spécifie ce qu'un objet peut contenir. Les modes principaux sont :

- `numeric` : nombres réels ;
- `character` : chaînes de caractères ;
- `logical` : valeurs logiques vrai/faux ;
- `list` : liste, collection d'objets ;
- `function` : fonction.

Les objets de mode numeric, character et logical, sont des objets simples qui peuvent contenir des données d'un seul type. Au contraire, les objets de mode list sont des objets spéciaux qui peuvent contenir d'autres objets. On accède au mode de l'objet avec la fonction `mode()`.

Taper la commande `mode()` pour les expressions suivantes :

```
vecteur<-c(18,17,19)
uneliste<-list(noms=c("Jean", "Mathilde", "Paul"), age=vecteur)
is.numeric(pi)
```

La commande `length()` permet de déteminer la longueur d'un objet. Utiliser cette commande pour les quantités suivantes :

```
v<-c(1,2,3)
uneliste<-list(noms=c("Jean", "Mathilde", "Paul"), age=vecteur)
"abc"
c("a","b","c")
```

Un objet spécial est la valeur manquante `NA`. Par défaut, son mode est `logical`, cependant `NA` n'est ni `TRUE` ni `FALSE`. Pour tester si une valeur est manquante, utiliser la fonction `is.na()`. Utiliser cette commande pour les quantités suivantes :

```
is.na(NA)
is.na(mean(c(1,4,NA)))
```

## 2.3 Vecteurs, matrices, tableaux

**Vecteurs** En R l'unité de base dans les calculs est le vecteur. Taper les expressions suivantes (le `c()` signifie *concaténation*) :

```
v<-c(1,7,5);
v[2]
v[c(1,3)]
v[-3]
v>4
v[v>4]
v[1]<-9; v
names(v)<-c("a", "b", "c"); v
v["c"]
1:10
2+1:10
letters[1:10]
seq(from=2,to=20,by=3)
rep(1,5)
rep(NA,4)
numeric(5)
```

**Matrices** Une matrice est un vecteur avec un attribut `dim` de longueur deux. Tous les éléments d'une matrice ont donc le même mode. Taper les expressions suivantes :

```
M<-matrix(c(1,3,1,4,7,8),2,3); M
M[1,3]
M[3]
M[,3]
M[2,]
M[-1,]
cbind(M,c(11,2))
N<-rbind(M,c(2,5,4))
2+N
2*N
N∧2
sqrt(N)
N*N
N%*%N
1/N
det(N)
solve(N)
t(N)
```

```
length(N)
dim(N)
diag(4)
matrix(0,10,8)
rowSums(N)
colSums(N)
cumprod(N)
cumsum(N[,2])
apply(N,2,sum)
apply(N,1,prod)
apply(N,1,min)
nrow(N); ncol(N)
```

Les matrices sont remplies par colonnes. Si on veut remplir une matrice par lignes, utiliser la commande `byrow`. Par exemple :

```
matrix(2:7, 2, 3)
matrix(2:7, 2, 3, byrow=TRUE)
```

**Tableaux** La commande `array` permet de générer un ensemble de tableaux. Elle généralise la notion de matrices (une matrice est un cas particulier de tableau de dimension 2). Par exemple :

```
a<-array(c(1,5,3,2,6,9,4,1,2,1,1,5,7,9,0,1,3,7,9,-2,2,3,1),dim=c(2,3,4))
```

Le résultat obtenu est un tableau de taille $2 \times 3 \times 4$, c'est-à-dire un ensemble de 4 matrices à 2 lignes et 3 colonnes. Les fonctions valables pour les matrices sont encore valables pour les tableaux; tester par exemple `colSums()`, `apply()`.

## 2.4 Listes

Les listes sont des vecteurs spéciaux qui peuvent stocker des éléments de n'importe quelle mode (y compris d'autres listes). Comme tout autre vecteur, une liste est indicée par l'opérateur `[ ]`. Taper les expressions suivantes :

```
L<-list(A="Lyon", B=8, C="Calais", D=TRUE)
L[1]
L[[1]]
L$A; L$A==L[1]
L[[1]]<-c("ville", "rhone");L
```

## 2.5 Data frames

Des cas particuliers de listes sont les *data frames*. Un data frame est représenté sous forme d'un tableau à deux dimensions dont les colonnes sont ses éléments. Typiquement, dans un data frame les colonnes représentent les variables ; et les lignes les observations. Contrairement aux matrices, les éléments d'un data frame peuvent avoir des modes différents. Taper les expressions suivantes :

```
df<-data.frame(NAME=c("chien", "chat", "tigre"), lieu=c("France", "Allemagne",
"Autriche"), autrevaleur=c(5,6,9), logique=c(TRUE, TRUE, FALSE))
df[3,4]; df[,3]; df[c(1,3), ]
df[df$autrevaleur>5,]
```

## 2.6 Fonctions définies par l'utilisateur

Voici un exemple de fonction :

```
phi<-function(x,y=10)
{x-y}
```

Par défaut, la valeur de $y$ dans l'exemple ci-dessus est égale à 10. Taper les expressions suivantes :

```
phi(3)
phi(1:10)
sapply(1:10,phi)
```

Pour évaluer la fonction sur une suite de valeurs (en l'occurrence de 1 à 10), il est préférable d'utiliser la commande `sapply` plutôt que d'évaluer la fonction en un vecteur. Le risque, en effet, est que `R` ne puisse pas évaluer la fonction en tous les points du vecteur. C'est, par exemple, le cas si on a une boucle en *if* :

```
g<-function(x)
{ if (x<=5) 2*x
else 0 }
```

L'expression `sapply(1:10,g)` permet d'évaluer $g$ en 1,2,..., 10 ; mais `g(1:10)` affichera un message d'erreur. En effet, `R` ne saura pas comment interpréter la condition `if (x<=5)` car cela sous-entend qu'on prend un nombre réel et non un vecteur.

## 2.7 Structures de contrôle

Les boucles sont à éviter car `R` présente de nombreuses commandes qui permettent à la fois de faire de la programmation plus facilement, mais aussi de diminuer le temps de calculs. Les boucles sont de la forme :

- `if (expression) {...}`
- `for (expression) {...}`

— `while (expression) {...}`

On peut faire également une boucle avec la commande `repeat` pour répéter une expression. Cette dernière devra comporter un test d'arrêt qui utilisera la commande `break`. Une boucle repeat est toujours exécutée au moins une fois. Taper le code suivant :

```
y<-1:5
for (i in 1:5){
y[i] <- exp(i) }
y
```

Voici un autre exemple :

```
x <- numeric(0)
i <- 1
repeat {
if (i %% 10 != 0)
x<-c(x,i)
if (100 < (i <- i + 1))
break
}
x
```

## 2.8 Importation de données

Pour importer des données, on utilise la commande `read.table("")`. Le résultat obtenu est un data frame. `R` possède, par ailleurs, un grand nombre de données. Selon les packages chargés, ces ensembles de données sont plus ou moins nombreux. La fonction `data()` permet d'afficher la liste des ensembles de données. Si l'on veut travailler, par exemple, sur les *datasets* d'"AirPassengers", il suffit de taper `AirPassengers`.

# 3 Exercices

**Exercice 1** Faire les exercices suivants dans le livre de V. Goulet :

1. 2.2;
2. 2.3;
3. 3.1;
4. 3.2.

**Exercice 2** On considère la matrice

$$B = \begin{pmatrix} 1 & 4 & 7 \\ 8 & -1 & 9 \\ 0 & 1 & 5 \end{pmatrix}.$$

1. A l'aide des fonctions `det()` et `eigen()`, calculer le déterminant de $B$, les valeurs propres et les vecteurs propres. Calculer l'inverse de B.
2. Calculer $B^5$.

**Exercice 3** Faire les exercices suivants dans le livre de V. Goulet :

1. 4.2 ;
2. 4.4 ;
3. 5.1 ;
4. 5.4.

**Exercice 4** Ecrire une fonction `racine()` qui calcule la racine carrée d'un nombre réel positif $a > 0$ en implémentant la méthode de Newton. La fonction prendra en argument $a$, un point d'initialisation positif $x_1$ et le nombre d'itérations $n$. On rappelle que la méthode de Newton pour trouver les zéros de la fonction $f$ consiste à construire une suite $(x_n)$ où $x_{n+1}$ est donnée par l'intersection de la tangente à $f$ en $x_n$ et l'axe des abscisses :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

**Exercice 5** Rappelons que la suite de Fibonacci $(F_n)$ est définie par $F_1 = 0$, $F_2 = 1$ et $F_n = F_{n-1} + F_{n-2}$ pour tout $n \geq 3$.

1. En utilisant une boucle `for`, calculer les 12 premiers termes de la suite de Fibonacci.
2. Changer les deux premiers termes de la suite en 2 et 2 et calculer les 12 premiers termes de la nouvelle suite.
3. En utilisant une boucle `while`, lister tous les termes de la suite de Fibonacci inférieurs à 300.
4. Ecrire une fonction calculant les $n$ pemiers termes de la suite de Fibonacci.
5. Calculer les 30 premiers termes de la suite de terme général $x_n = \frac{F_{n+1}}{F_n}$, $n \geq 2$, où $(F_n)$ désigne la suite de Fibonacci.
6. La suite $(x_n)$ est-elle convergente ? Calculer $\frac{1+\sqrt{5}}{2}$. Conclure.

**Références** Pour le cours :

- V. Goulet, *Introduction à la programmation en R*, https://cran.r-project.org/doc/contrib/Goulet_introduction_programmation_R.pdf ;
- E. Paradis, *R pour les débutants*, https://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf ;
- A. Philippe, *Notes de cours sur le logiciel R*, https://www.math.sciences.univ-nantes.fr/~philippe/download/Anne-Philippe-cours-R-par4.pdf.

Pour la partie TP, on s'est appuyé sur les TPs de F. Corset (Univ. Grenobles), A. Illig (Univ. Versailles St-Quentin), L. Smoch (Univ. Littoral) et V. Perduca (Univ. Paris).