# Le logiciel R

Enseignement à distance, 2020

# Table des matières

# Objectifs

Le TP logiciel spécialisé vise à :

- Familiariser avec un logiciel de traitement statistique (logiciel R)
- Définir quelques commandes prédéfinies du logiciel R
- Utiliser le logiciel R comme outil de calcul statistique

À l'issue de ce TP l'étudiant sera capable de :

- Programmer avec le logiciel R
- Utiliser le logiciel R pour le calcul statistique

# Introduction

Le logiciel R est un logiciel de statistique créé par Ross Ihaka & Robert Gentleman. Il est à la fois un langage informatique et un environnement de travail : les commandes sont exécutées grâce à des instructions codées dans un

langage relativement simple, les résultats sont affichés sous forme de texte et les graphiques sont visualisés directement dans une fenêtre qui leur est propre. C'est un clone du logiciel S-plus qui est fondé sur le langage de programmation

orienté objet S, développé par AT&T Bell Laboratories en 1988. Ce logiciel sert à manipuler des données, à tracer des graphiques et à faire des analyses statistiques sur ces données.

# Partie 1 : Initiation au logiciel R



L'objectif de ce court document est de vous initier au logiciel R et à la programmation dans cet environnement. L'idée est de vous exposer les bases de cet outil de travail et de vous habiliter à résoudre des problèmes de statistiques avec celui-ci.

## 1. Présentation du logiciel R

Initié dans les années 90 par Robert Gentleman et Ross Ihaka, auxquels sont venus depuis s'ajouter de nombreux chercheurs, le logiciel R constitue aujourd'hui un langage de programmation intégré d'analyse statistique. Le R est un environnement intégré de manipulation de données, de calcul et de préparation de graphiques. Toutefois, ce n'est pas seulement un environnement statistique, mais aussi un langage de programmation complet et autonome. Parmi ses caractéristiques particulièrement intéressantes,

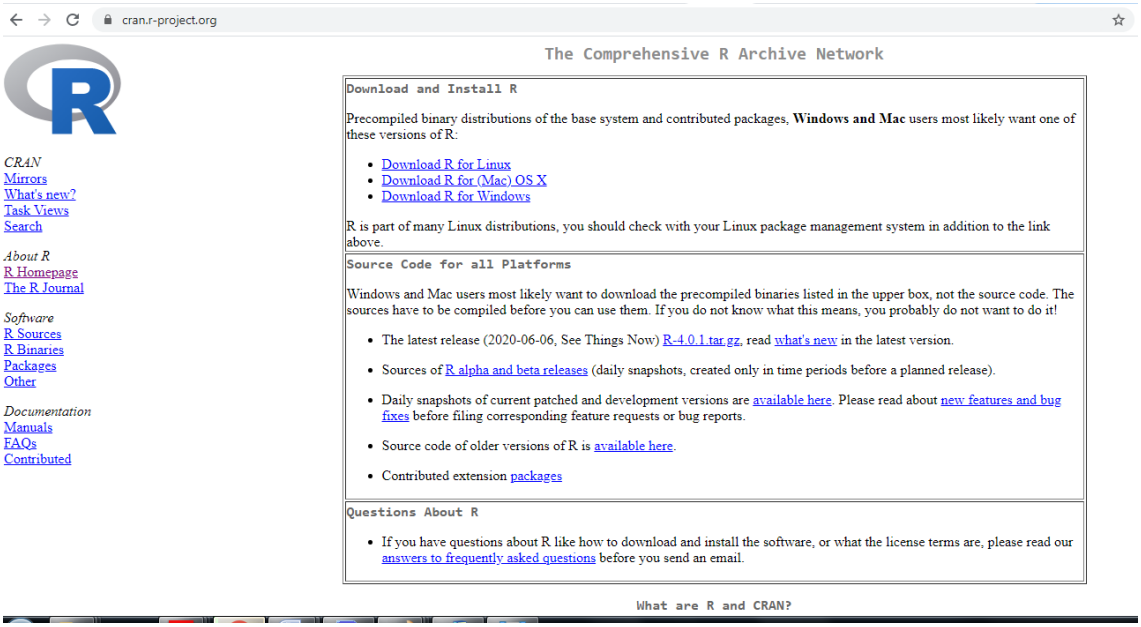
on note :

- langage basé sur la notion de vecteur, ce qui simplifie les calculs mathématiques et réduit considérablement le recours aux structures itératives
- (boucles for, while, etc.) ;
- pas de typage ni de déclaration obligatoire des variables ;
- programmes courts, en général quelques lignes de code seulement ;
- temps de développement très court.

# 2. Installation de R

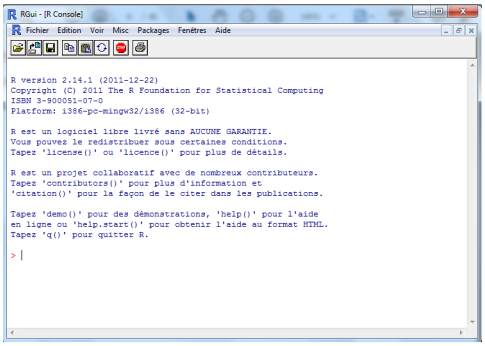
On télécharge gratuitement le R sur le site *https://cran.r-project.org/* et on

sélectionne l'OS voulu (Linux, windows, ...)




# 3. Démarrage de R

Pour lancer R, on clique sur l'icône R et la fenêtre de commandes apparaît. La version actuelle est 2.14.1, elle date de 2011




Le prompteur > indique que R est prêt recevoir les commandes . Pour quitter la session, il faut taper q()

# 4. Premières opérations

*Affectation*

```
> a <- 1.2
```

*a* est une variable, en interne c'est un vecteur de taille *1* et il peut contenir des données numériques.

Dans la terminologie R, *a* est un « objet »

*Lister le contenu de la mémoire*

```
> ls()
```

Affiche tous les objets situés dans la mémoire de l'environnement R

*Supprimer un objet*

```
> rm (a)
```

où *a* représente le nom de l'objet à supprimer

*Supprimer tous les objets en mémoire*

*>rm (list= ls())*

On liste le nom de tous les objets en mémoire et on les efface Attention, on supprime autant les données ( matrices) que les modèles (ex résultats de régression, etc

*Accès à l'aide d'une fonction*

```
> help(c)
```

Par ex. envoie l'aide pour la commande «c » qui sert à déclarer un vecteur

# 5. Les objets

Les éléments de base du langage R sont des objets qui peuvent être des données (vecteurs, matrices, séries chronologiques...), des fonctions, des graphiques...

## 5.1. Les vecteurs

Il s'agit de l'objet de base dans R. Un vecteur est une entité unique formée d'une collection ordonnée d'éléments de même nature. Dans R, les vecteurs peuvent être constitués d'éléments numériques, logiques ou alphanumériques. On peut créer manuellement un vecteur à l'aide de la commande

*c(elt1,elt2, ...)*. Les composantes elt1,elt2, ... du vecteur peuvent être numériques (réelles ou complexes), logiques (TRUE, FALSE, en majuscule) ou alphanumériques (chaînes de caractères).

### *Exemple*

vous allez maintenant taper les séquences suivantes au clavier (excepté le premier caractère >, invite de R signalant que la console est en attente d'une

instruction, ou encore le caractère + signalant que la console attend la suite d'une instruction).

- Création de l'objet *a* recevant un vecteur numérique de dimension 5 et de cordonnées 5, 5. 6, 1 , 4 ,-5.

```
> a=c(5,5.6,1,4,-5)
```

Notons que l'affectation par a<-c(5,5.6,1,4,-5) ne fonctionne plus sur les versions récentes, pas plus

que a_5 d'ailleurs.

- Pour afficher le vecteur *a*:

```
>a
```

- Affichage de la première coordonnée du vecteur *a*:

```
>a[1]
```

- Création du vecteur numérique *b* de dimension *3* et de coordonnées 5.6 , 1 ,4

```
>b=a[2:4]
```

- Création du vecteur numérique *d* de dimension *3* et de coordonnées 5,1,-5

```
> d=a[c(1,3,5)]
```

```
>d
```

- Multiplication par deux de chacune des coordonnées du vecteur a et affichage du résultat

```
>2*a
```

- Création du vecteur numérique *e* de dimension *3* et de coordonnées 3/5,3,-3/5

```
> e=3/d
```

```
> e
```

### 5.1.1. Commandes usuelles de R:

`>sum(a)` Calcul la somme des éléments du vecteur *a*

`>length(a)` Affichage de la dimension du vecteur *a*

`> a%*%b` Produit scalaire entre le vecteur ligne a et le vecteur colonne b

`>sqrt(2)` Racine carrée de 2

`>g=c(sqrt(2),log(10))` Création du vecteur numérique *g* de dimension *2* et de coordonnées $\sqrt{2}$, log(10)

`text=c("grand","petit")` Création du vecteur caractère text de dimension 2 de coordonnées grand,petit

## 5.2. Les matrices

Les matrices sont également les objets de base pour R. On peut effectuer sur les matrices de nombreuses manipulations de manière très simple.

Syntaxe :

```
>M = matrix(c(1:6),ncol=2)
```

```
>M
```

```
[,1] [,2]

[1,] 1 4

[2,] 2 5

[3,] 3 6
```

La fonction matrix crée une matrice à 2 colonnes en "empilant" les éléments (1, 2,...,6) en "colonnes".

Produit matriciel entre M1 et M2

```
>  b=M1%*%M2 ###le langage contrôle l'adéquation des dimensions
```

Affichage de dimension de la matrice M

```
>dim(M)
```

Sélection de l'élément [i,j] de la matrice M

```
>M[i,j]
```

Sélection de la nème colonne de M

```
>M[,n]
```

Sélection de la ième et la jème lignes de M

```
>M[c(i,j),]
```

Suppression de la ième ligne de M

```
>M[-i,]
```

Suppression de la ième et la jème colonnes de M

```
>M[,-c(i,j)]
```

### *Exemple*

On peut sélectionner (avec les conventions "usuelles") des éléments, des lignes ou des colonnes d'une matrice.

```
>A0 = matrix(c(1:6),ncol=2)

>A[1,2]

>[1] 4
```

La fonction dim renvoie le vecteur qui contient le nombre de lignes et de colonnes de la matrice.

```
>dim(A)
```

La transposée est obtenue avec t(A).

```
>t(A)
```

Il est également possible de créer directement des matrices diagonales.

```
>C = diag(c(1,2)) ### les éléments diagonaux sont 1, 2
```

### 5.2.1. Commandes usuelles de R:

Concaténation verticale des matrices M1 et M2

```
>rbind(M1,M2)
```

Concaténation horizontale des matrices M1 et M2

```
>cbind(M1,M2)
```

Calcul de la somme de M1 par colonne

```
>apply(M1,2,sum)
```

Calcul de la somme de M1 par ligne

```
>apply(M1,1,sum)
```

La fonction *solve* calcule l'inverse d'une matrice M

```
>solve(M)
```

La fonction *eigen* permet de diagonaliser une matrice carrée (symétrique ou non, valeurs propres peuvent être complexes).

```
>eigen(M)
```

Commande pour nommer les lignes

```
>rownames(M)
```

Commande pour nommer les colonnes

```
>colnames(M)
```

## 5.3. Les listes

Les listes sont des collections d'objets (vecteurs, matrices, ...) qui ne sont pas nécessairement du même type. Elles sont utilisées par de nombreuses fonctions pour retourner les résultats et permettent en particulier de stocker et manipuler simplement des objets de longueurs différentes dans une même structure.

La constitution d'une liste passe par la fonction *list(nom1=el1,nom2=el2,...),* l'utilisation des noms étant facultative. On peut accéder à chaque élément de la liste à l'aide de son index entre double crochets [[...]], ou par son nom précédé du signe $.

Syntaxe :

```
>li=list(nom1=, nom2=..)
```

*Exemple*

```
>li=list(num=1:5,y="couleur",a=TRUE)
> li
```

```
$num
[1] 1 2 3 4 5
$y
[1] "couleur"
$a
[1] TRUE
```

On peut accéder à chaque élément de la liste à l'aide de son index entre double crochets[[]]

```
>li[[1]]
[1] 1 2 3 4 5
```

Ou par son nom précédé du signe

```
> li$num
[1] 1 2 3 4 5
```

## 5.4. Les tableaux de données (data.frame)

Les data.frame se présentent sous la forme d'une matrice dont les colonnes peuvent être associées à des objets de différents modes (numérique, chaîne de caractère, ...). Ils constituent une classe particulière de listes ou chaque élément de la liste a la même longueur et est associé à une colonne.

Pour créer un tableau de données, on peut regrouper des variables de même longueur à l'aide de la commande data.frame(nom1=var1,nom2=var2,...). Ce format est bien adapté au stockage de données statistiques :

Un petit exemple

```
>x =1:4
> n = 10
>M = c(10, 35)
>y = 2:4
> data.frame(x, n)
x n
1 1 10
2 2 10
3 3 10
4 4 10
>data.frame(x, M)
```

```
>data.frame(x, y)
Error in data.frame(x, y) :
arguments imply differing number of rows: 4, 3
```

Si un facteur est inclus dans le tableau de données, il doit être de même longueur que le(s) vecteur(s).

### *Exemple*

- construire à partir des vecteurs grâce à la fonction un tableau de donnée

```
> db = data.frame(nom = c("Alan","Candice","Ali","Vicky","Zico","Nami","
larso"), age = c(19,20,16,21,22,25,20), genre = c("M","F","M","F","M","
F","F"),
regulier = c(FALSE,TRUE,FALSE,TRUE,FALSE,TRUE,TRUE))
> db # afficher notre data frame
nom age genre regulier
1 Alan 19 M FALSE
2 Candice 20 F TRUE
3 Ali 16 M FALSE
4 Vicky 21 F TRUE
5 Zico 22 M FALSE
6 Nami 25 F TRUE
7 larso 20 F TRUE
```

Les caractéristiques du data frame

```
> # les dimensions
> dim(db)
[1] 7 4
> # nombre de lignes ou observations
> nrow(db)
[1] 7
> # nombre de colonnes ou variables
> ncol(db)
[1] 4
```

- Renommer les observations avec un code alphabétique comme ci-dessous :
  R
  ```
  > rownames(db) = LETTERS[1:nrow(db)]
  ```

```
> # on voit bien que les chiffres représentant les noms des lignes
ont été remplacés par des lettres
> db
nom age genre regulier
A Alan 19 M FALSE
B Candice 20 F TRUE
C Ali 16 M FALSE
D Vicky 21 F TRUE
E Zico 22 M FALSE
F Nami 25 F TRUE
G larso 20 F TRUE
```

- Extraction d'éléments ou d ‘ensemble d'éléments d'un data frame

```
>db[1,]
nom age genre regulier
A Alan 19 M FALSE
>db["A",] # ou utiliser le nom
nom age genre regulier
A Alan 19 M FALSE
>db[2:5,] # ou utiliser les noms
nom age genre regulier
B Candice 20 F TRUE
C Ali 16 M FALSE
D Vicky 21 F TRUE
E Zico 22 M FALSE
>db[c("B","C","D","E"),]
nom age genre regulier
B Candice 20 F TRUE
C Ali 16 M FALSE
D Vicky 21 F TRUE
E Zico 22 M FALSE
>db[c(1,3,6),] # ou utiliser les noms
```

```
nom age genre regulier
A Alan 19 M FALSE
C Ali 16 M FALSE
F Nami 25 F TRUE
>db[c("A","C","F"),]
nom age genre regulier
A Alan 19 M FALSE
C Ali 16 M FALSE
F Nami 25 F TRUE
>## extraire une colonne
>db[2] # ou utiliser le nom
age
A 19
B 20
C 16
D 21
E 22
F 25
G 20
>db["age"]
age
A 19
B 20
C 16
D 21
E 22
F 25
G 20
>db$age
[1] 19 20 16 21 22 25 20
>## extraire un ensemble de colonnes ou variables
```

```
>db[2:4]
age genre regulier
A 19 M FALSE
B 20 F TRUE
C 16 M FALSE
D 21 F TRUE
E 22 M FALSE
F 25 F TRUE
G 20 F TRUE
>#  extraire un ensemble de colonnes ou variables discontinues
>db[c(1,3,4)]
nom genre regulier
A Alan M FALSE
B Candice F TRUE
C Ali M FALSE
D Vicky F TRUE
E Zico M FALSE
F Nami F TRUE
G larso F TRUE
>db[c("age","genre","regulier")]
age genre regulier
A 19 M FALSE
B 20 F TRUE
C 16 M FALSE
D 21 F TRUE
E 22 M FALSE
F 25 F TRUE
G 20 F TRUE
>#  extraire à la fois colonnes et lignes
>db[c(2,3,6),c(1,2,4)]
```

*Remarque*

- Pour l'accès aux colonnes (composants) dans les data.frame, on utilise la même syntaxe que les matrices et les listes.
- Il est possible de transformer une matrice en tableau de données en utilisant la commande *as. data.frame(mat).*

### 5.4.1. Commandes usuelles R

`>subset()` une commande qui prend en argument d'abord un data frame et ensuite un argument conditionnel, on peut faire des extractions très structurées et avancées.

Exemple

```
>#Extraire les observations ou individus ayant moins de 20 ans
>subset(db, age < 20 )##db[db$age < 20,]
nom age genre regulier
A Alan 19 M FALSE
C Ali 16 M FALSE
>#  Individus de sexe féminin et plus agées que la moyenne
>subset(db,age > mean(age)& genre == "F")# ou db[mean(db$age)& db$genre
== "F",]
nom age genre regulier
D Vicky 21 F TRUE
F Nami 25 F TRUE
```

`>order()` Ordonner un data frame

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

Exemple

nom age genre regulier

F Nami 25 F TRUE

E Zico 22 M FALSE

D Vicky 21 F TRUE

B Candice 20 F TRUE

G larso 20 F TRUE

A Alan 19 M FALSE

C Ali 16 M FALSE

# 6. Fonctions graphiques

Les graphiques sont également un point fort de R: Il offre de très nombreuses possibilités dont on peut avoir un premier aperçu avec la démonstration proposée par R. `> demo(graphics)`.

Une fonction de base est la commande *plot* qui permet de tracer des nuages de points.

La syntaxe générale est la suivante pour représenter le vecteur y en fonction de x

```
>y=c(12,10,7,13,26,16,4,12,13,14,16,8)
```

```
>x=c(5,7,9,9,4,8,7,5,5,10,12,13)
```

```
> plot(x,y,xlab="Légende abscisses",ylab="Légende ordonnées")
```

Les options de la commande plot sont très nombreuses, par exemple :

- Ajout d'un titre avec la commande *main*: (Exemple : `plot(x,y,main="Y en fonction de X")`)
- Choix des couleurs avec la commande *col* (Exemple : `plot(x,y,col="red")`)
- Choix de la taille des points avec la commande *cex* (Exemple : `plot(x,y,cex=2)`)
- Choix de la forme de points avec la commande pch (Exemple : `plot(x,y,pch=1)`)
- Possibilité de relier les points par des lignes avec la commande type (Exemple :`plot(x,y,type="l")`)
- Préciser les limites des axes (valeur minimale et maximale) avec xlim et ylim (Exemple :`plot(x,y,xlim=c(0,2)`

  Ajout à un graphique existant avec la commande *lines*(Exemple :

  ```
  >z=c(1:,12)
  >lines(x,z,col="green"))
  ```

La commande *plot*pour le cas de deux variables quantitatives :

Syntaxe :

>#Déclarer les deux vecteur x

Quelques autres fonctions graphiques utiles : Les diagrammes en barre *(barplot)* les histogrammes *hist* et les diagrammes en secteurs (commande pie) sont faciles à tracer avec R:

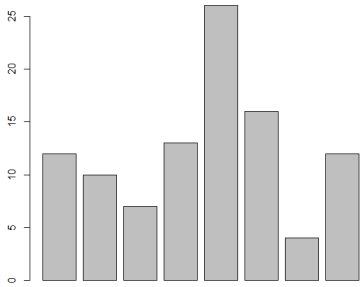
Exemple

Pour un vecteur data à 8 éléments, les commandes barplot(data) donnent 8 barres verticales

>data=c(12,10,7,13,26,16,4,12)

>barplot(data)

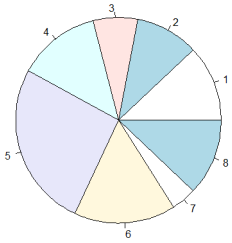
Cela renvoie :




Pour la commande *pie*, On fait :

```
> pie(data)
```

Cela renvoie :




# 7. Fonctions utiles

- `sum(x)` |Somme des éléments de x
- `prod(x)` Produit des éléments de x
- `max(x), min(x)` Maximum, minimum des éléments de x
- `which.max(x)` Retourne l'indice du maximum des éléments de x
- `which.min(x)` Retourne l'indice du minimum des éléments de x
- `mean(x)` Moyenne des éléments de x
- `median(x)` Médiane des éléments de x
- `var(x)` ou `cov(x)` Variance des éléments de x(calculée sur n-1) matrice des var et cov si x est une matrice
- `cor(x)` Matrice de corrélation si x est une matrice ou un data frame
- `sd(x)` Ecart-type des éléments de x
- `round(x,n)` Arrondit les éléments de x à n chiffres après la virgule
- `rev(x)` Inverse l'ordre des éléments de x
- `sort(x)` Trie les éléments de x dans l'ordre ascendant
- `scale(x)` Centre et réduit les données

# 8. Stratégies de travail

Dans la mesure où R se présente essentiellement sous forme d'une invite de commande, il existe deux grandes

stratégies de travail avec cet environnement statistique.

1- On entre des expressions à la ligne de commande pour les évaluer immédiatement.

```
>a=6+7

>a

[1] 13
```

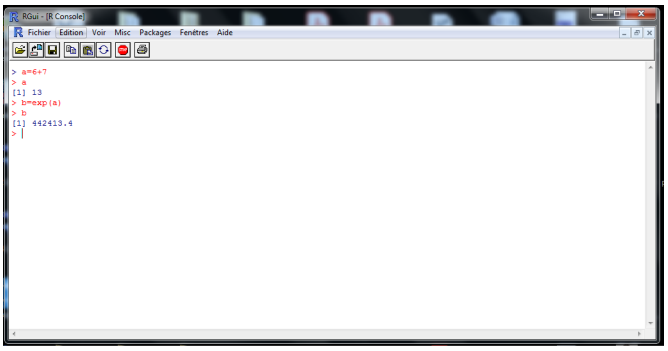
On peut également créer des objets contenant le résultat d'un calcul. Ces objets sont stockés en mémoire dans

l'espace de travail de R :

```
>b=exp(a)

>b

[1] 442413.4
```




*Espace de travail R*

Lorsque la session de travail est terminée, on sauvegarde une image de l'espace de travail sur le disque dur

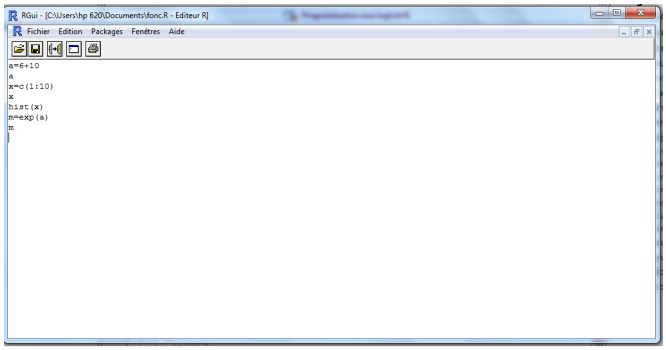
de l'ordinateur afin de pouvoir conserver les objets pour une future séance de travail :

```
> save.image()
```

Par défaut, l'image est sauvegardée dans un fichier nommé .RData dans le dossier de travail actif et cette image est automatiquement chargée en mémoire au prochain lancement de R.

2- La deuxième approche considère que ce qu'il importe de conserver d'une session de travail à l'autre n'est pas tant les objets que le code qui a servi à les créer. Ainsi, on sauvegardera dans ce que l'on nommera des fichiers de script nos expressions R et le code de nos fonctions personnelles. Par convention, on donne aux fichiers de script un nom se terminant avec l'extension *.R* (`nomdescript.R` ). Avec cette approche, les objets sont créés au besoin en exécutant le code des fichiers de script.

Comment ? Simplement en copiant le code du fichier de script et en le collant dans l'invite de commande de R en utilisant la commande `source("nomdescript.R)"`. La figure suivante montre l'espace de programmation du logiciel R




*Espace Editeur*

- Pour accéder à l'espace éditeur, dans la barre de menu :Fichier ->Nouveau Script
- Pour exécuter le script, dans l'environnement de travail. tapez la commande *source ("nomdescript.R")*,ou bien, dans la barre de menu : Edition->Exécuter tout