INTRODUCTION AU LOGICIEL R

Julien JACQUES
http://eric.univ-lyon2.fr/~jjacques/

L'objectif de ce document est de présenter une très courte introduction au logiciel R (via l'interface RStudio), de sorte que des étudiants découvrant R puissent en quelques heures se familiariser avec ce logiciel et être opérationnels par la suite pour réaliser des exercices de travaux pratiques accompagnant un cours de Statistique.

1 Introduction

Le logiciel R (disponible sur `http://www.r-project.org/`) est un logiciel de Statistique libre ayant un certain nombre d'atouts :

- il permet l'utilisation des méthodes statistiques classiques à l'aide de fonctions prédéfinies,
- il permet de créer ses propres programmes dans un langage de programmation assez simple d'utilisation (proche de Matlab),
- il permet d'utiliser des techniques statistiques innovantes et récentes à l'aide de package développés par les chercheurs et mis à disposition sur le site du CRAN (`http://cran.r-project.org/`).

Le logiciel R fonctionne initialement en ligne de commande, mais des interfaces permettent désormais une utilisation plus conviviale. Nous proposons ici de travailler avec l'interface RStudio, téléchargeable sur :

`http://www.rstudio.com/`

1.1 L'interface RStudio

L'interface RStudio (Figure 1) est composée de quatre fenêtres :

- Fenêtre d'édition (en haut à gauche) : dans cette fenêtre apparaissent les fichiers contenant les scripts R que l'utilisateur est en train de développer. Enregistrer le fichier avec une extension `.r` permet une coloration syntaxique adaptée au langage R. En entête de cette fenêtre, des icônes permettent de sauvegarder le fichier, d'exécuter un morceau de code sélectionné (icône `run`) ou l'intégralité du code contenu dans le fichier (icône `source`).
- Fenêtre de commande (en bas à gauche) : cette fenêtre contient une console dans laquelle les codes R sont saisis pour être exécutés.
- Fenêtre espace de travail / historique (en haut à droite) : contient les objets en mémoire, que l'on peut consulter en cliquant sur leur noms, ainsi que l'historique des commandes exécutées,
- Fenêtre explorateur / graphique / package / aide (en bas à droite) : l'explorateur permet de se déplacer dans l'arborescence des répertoires, la fenêtre graphique contient les graphiques tracés via R (il est possible de les exporter), la fenêtre package montre les packages installés et actuellement chargés et la fenêtre d'aide contient la documentation sur les fonctions et packages.

1.2 Le répertoire de travail

Le répertoire de travail est celui à partir duquel vous avez lancer l'interface RStudio. Il sera pratique de se placer dans un répertoire de travail bien défini, celui par exemple contenant le fichier `.r` dans lequel vous tapez vos scripts R. Pour ce faire, vous pouvez soit utiliser la commande `setwd` pour vous déplacer dans l'arborescence des répertoires, soit utiliser le menu de l'interface :

✓ Session
  ✓ Set Working Directory
    ✓ To Source File Location

Par la suite, lorsque vous serez amené à charger des jeux de données, si ceux-ci sont placés dans le répertoire courant dans lequel vous vous êtes placé, vous n'aurez pas à saisir le chemin complet de ce répertoire.

1.3 Les packages

Un grand nombre de fonctions, contenus dans différents packages, sont installés dans la version de base du logiciel R. Il est possible (et nous en aurons besoin dans les différents cours de Statistique) d'installer des packages supplémentaires. Pour cela, lorsque vous disposez d'une connexion internet, il suffit d'utiliser la commande suivante en indiquant le nom du package que l'on veut installer :

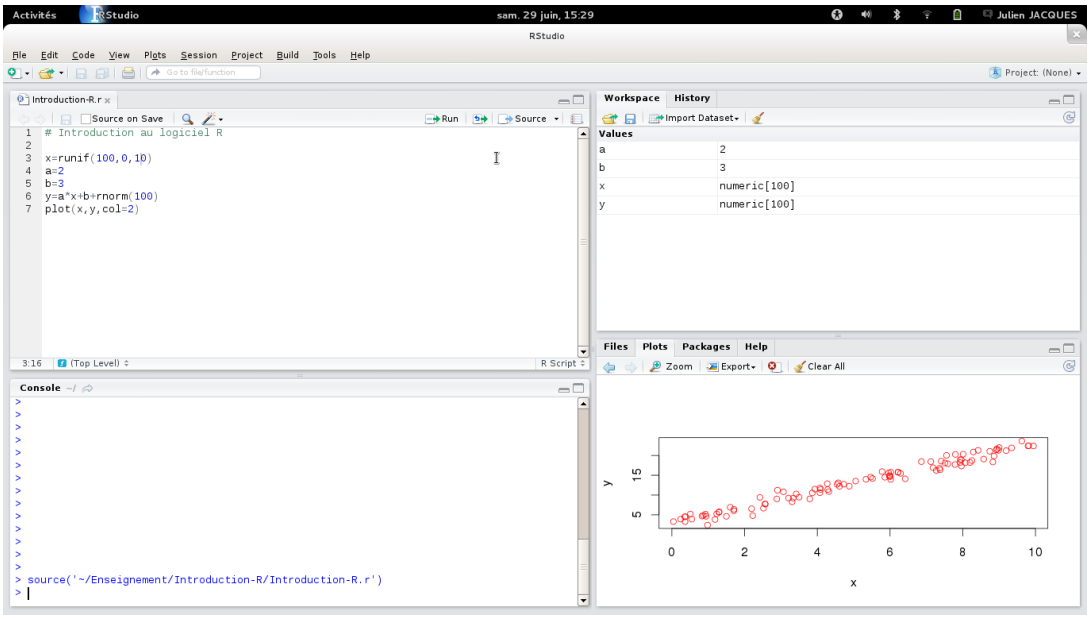

FIGURE 1 – Interface graphique RStudio

```
R> install.packages('funFEM')
```

RStudio vous proposera alors de choisir le serveur à utiliser pour télécharger le package et procédera ensuite à l'installation. Il faudra ensuite charger le package à l'aide de la commande `library()` :

```
R> library('funFEM')
```

L'installation n'est à réaliser qu'une seule fois, alors que le chargement du package doit être fait au lancement de chaque nouvelle session.

2 Premières commandes R

2.1 Calcul élémentaires

R peut être utilisé pour réaliser des opérations élémentaires :

```
R> ((1+sqrt(5))/2)^2
```

dont le résultat peut être stocké dans une variable

```
R> a = ((1+sqrt(5))/2)^2
```

gardée en mémoire (`a` apparaît alors dans la fenêtre espace de travail), et qui peut être ré-utilisée par la suite :

```
R> nombredor = sqrt(a)
```

Pour effacer les variables en mémoire dans la session R, il faut taper la commande suivante :

```
R> rm(list=ls())
```

2.2 Scalaires, vecteurs et matrices

La variable `a` définie ci-dessus est un scalaire. R gère également des vecteurs et matrices. Un des avantages de R est qu'un grand nombre d'opérations et de fonctions sont applicables directement sur des vecteurs (voir sur des matrices). Ainsi, le recours au boucle de type `for` peut être évitée, et doit généralement l'être pour des raisons de rapidité d'exécution.
Pour créer un vecteur, il est possible d'utiliser la fonction de concaténation `c` :

```
R> x = c(7,8,9)
R> y = 1:3
R> z = rep(x,y)
```

La commande `matrix` permet de créer une matrice

```
R> M = matrix(z,2,3)
```

ce qui peut également être fait en concaténant des vecteurs en ligne (`rbind`) ou en colonne (`cbind`) :

```
R> M = cbind(x,y)
R> M = rbind(x,y)
```

Les tableaux à plus de 2 dimensions, appelés `array` en R, sont également utilisables :

```
R> T = array(0,dim=c(2,3,4))
```

2.3 Les fonctions

R dispose d'un grand nombre de fonctions prédéfinies, utilisables en appelant la fonction par son nom suivi de ses arguments entre parenthèses :

```
R> mean(x)
R> rnorm(10)
R> rnorm(10,mean=1,sd=2)
```

Un memento des principales fonctions R est disponible Section 9. Il est également possible de créer ses propres fonctions (Section 8.3).

Astuce : lorsque vous commencez à taper le nom de la fonction, vous pouvez en appuyant sur la touche tabulation voir les différentes fonctions commençant par les lettres déjà saisies. Lorsque le nom de la fonction est totalement saisi, la tabulation permet de voir les arguments attendus par la fonction.

3 L'aide et la documentation

L'aide sur une fonction est accessible des deux façons suivantes :

```
R> help(rnorm)
R> ?rnorm
```

Astuce : un bon moyen pour trouver de l'aide et des exemples sur une fonction consiste simplement à taper le nom de la fonction sous Google.

4 Les scripts

Vous n'êtes pas obligé de taper toutes les commandes R dans la fenêtre de commande. Il est possible de créer des scripts R (dans la fenêtre d'édition), en les enregistrant avec une extension `.r` (`myscript.r` par exemple), et de les exécuter à l'aide de la commande source :

```
R> source("myscript.r")
```

Les icônes `source` et `run` permettent d'exécuter tout ou partie du script R affiché dans la fenêtre d'édition.

Astuce : il est également possible, sous Linux, de lancer l'exécution d'un script R sans ouvrir le logiciel R :

```
> nohup R CMD BATCH myscript.r resultfile &
```

L'ensemble des sorties seront alors redirigés dans le fichier `resultfile`.

5 Les structures de données

Nous avons déjà vu les notions de scalaire, vecteur, matrice et tableau à plus de deux dimensions. R gère également des listes, ainsi qu'une structure spécifique à R : le *data frame*

5.1 Listes

Une liste est une combinaison de structures de données de natures potentiellement différentes :

```
R> L=list(elt1=c(1,2,3),elt2=matrix(rnorm(9),3,3),elt3='tutu',
elt4=seq(1,4,by=0.5))
```

Les éléments de la liste sont alors accessible par un $, et les noms des éléments par la commande `names` :

```
R> L$elt4
R> names(L)
```

5.2 Data frames

Un data frame est une matrice dont les colonnes ont des noms. C'est la structure de données principalement utilisée en R pour manipuler des jeux de données de type individus / variables. L'avantage est alors de pouvoir appeler une colonne par le nom de la variable qui y est stockée, sans avoir à connaître son numéro :

```
R> df=data.frame(x = c(11,12,14), y = c(19,20,21), z = c(10,9,7))
R> mean(df$x)
```

6 Les graphiques

R permet de créer un grand nombre de graphiques, qui seront introduits au fur et à mesure des cours de Statistique. Nous présentons ici deux premières fonctions `plot` et `hist`.

La fonction `plot` permet de représenter un nuage de points (Figure 2) :

```
R> x=rnorm(20);y=2*x+1+rnorm(20,0,0.1)
R> plot(x,y,type='p',xlab='x',ylab='y',main='Nuage de
points',col=2)
```

La fonction `hist` permet de représenter un histogramme (Figure 2) :

```
R> hist(rnorm(1000),breaks=20,main='Histogramme')
```

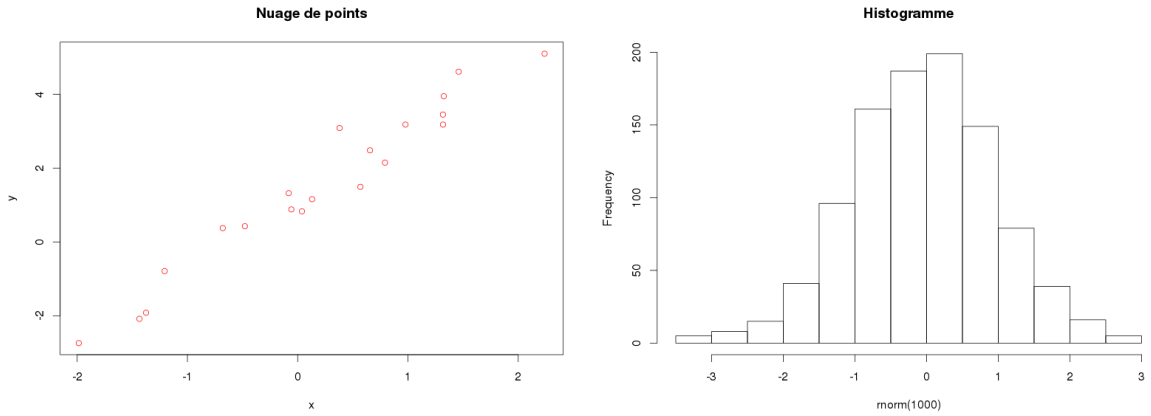

FIGURE 2 – Nuage de points (gauche) avec la fonction `plot` et histogramme avec la fonction `hist`.

7 Importer et exporter des fichiers de données

Il y a plusieurs façons d'importer et d'exporter des fichiers de données dans R. Les principales sont les fonctions `write.table` et `read.table` qui permettent respectivement d'exporter dans un fichier texte un data

frame et d'importer un fichier texte (de type individus en ligne et variables en colonnes) dans un data frame. Voici un exemple d'utilisation :

```
R> df=data.frame(x = c(11,12,14), y = c(19,20,21), z = c(10,9,7))
R> write.table(df,file='mydataframe.txt',row.names=FALSE)
R> newdf=read.table('mydataframe.txt',header=TRUE)
```

L'argument `row.names=FALSE` de `write.table` permet de ne pas sauvegarder de noms aux lignes. Par défaut l'option `col.names=TRUE` sauvegarde les noms des colonnes, qui sont ensuite ré-importées grâce à l'option `header=TRUE` de `read.table`.

8 Éléments de programmation

8.1 Condition if

La syntaxe pour la condition `if` est la suivante (la condition `else` pouvant être omise) :

```
R> w=2
R> if (w>3){
R> res=2
R> }else{
R> res=4
R> }
R> print(res)
```

8.2 Boucle for

Une boucle `for` a la syntaxe suivante

```
R> vec=c()
R> for (i in 1:10){
R> vec=c(vec,i)
R> cat('iteration numero: ',i,'/n')
R> }
```

8.3 Écrire ses propres fonctions

Un des grands intérêts du logiciel R est qu'il est possible de créer ses propres fonctions. Voici ci-dessous un exemple permettant de présenter la syntaxe. Les arguments de la fonction sont donnés après l'instance `function` ; une valeur par défaut à un argument peut être donnée en indiquant cette valeur lorsque les arguments sont définis (par exemple `arg2=0` indique que l'argument `arg2` aura la valeur nulle par défaut). Le résultat de la fonction peut être de différente nature (scalaire, vecteur, matrice, liste...).

```
R> mafonction <- function(arg1,arg2=0,arg3=1){
R> tmp=1/sqrt(2*pi*arg3) * exp(-1/2 * ((arg1-arg2)/(sqrt(arg3)))^2
)
R> return(res=list(argument1=arg1,densite=tmp))
R> }
R> x=seq(-3,5,0.01)
R> y=mafonction(x,arg2=1)
R> plot(x,y$densite,type='l',col=3)
```

9 Memento des principales fonctions R

Création de données

- `read.table` : lit un data frame à partir d'un fichier. Arguments : `header=TRUE` si la première ligne correspond aux intitulés des variables ; `sep=","` pour indiquer le séparateur de variables dans le fichier ; `skip=n` pour ne pas lire les n premières lignes.
- `write.table` : sauvegarde un data frame dans un fichier.
- `c` : concatène des scalaires en un vecteur.
- `rbind, cbind` : concatène en ligne ou en colonne des vecteurs en une matrice.
- `list` : crée une liste.
- `matrix` : crée une matrice à `nrow` lignes et `ncol` colonnes.
- `data.frame` : crée un data frame.
- `array` : crée un tableau dont l'argument `dim` permet de préciser le nombre de dimensions ainsi que la taille de chaque dimension.
- `seq` : créer une séquence d'entiers.
- `rnorm, runif` : simule la génération d'une variable aléatoire normale, uniforme.

Manipulation de données
- `x[n]` : n-ème élément du vecteur `x`.
- `x[n:m]` : n-ème au m-ème éléments du vecteur `x`.
- `x[c(k,l,m)]` : k-ème, l-ème et m-ème éléments du vecteur `x`.
- `x[x>m & x<n]` : éléments de `x` compris entre m et n.
- `l$x` ou `l[["x"]]` : élément `x` de la liste `l`.
- `M[i,j]` : élément ligne i et colonne j de la matrice `M`.
- `M[i,]` : i-ème ligne de la matrice `M`.
- `t(M)` : transposée de la matrice `M`.
- `solve(M)` : inverse de la matrice `M`.
- `M%*%N` : produit des matrices `M` et `N`.
- `sort(x)` : tri du vecteur `x`.

Information sur les variables
- `length` : longueur d'un vecteur.
- `ncol`, `nrow` : nombre de colonnes et de lignes d'une matrice.
- `str` : affiche le type d'un objet.
- `as.numeric`, `as.character` : change un objet en un nombre ou une chaine de caractères.
- `is.na` : teste si la variable est de type 'NA' (valeur manquante).

Statistiques
- `sum` : somme d'un vecteur.
- `mean` : moyenne d'un vecteur.
- `sd`, `var` : écart-type et variance d'un vecteur (dénominateur $n-1$)
- `rowSums`, `rowMeans`, `colSums` ou `colSums` : somme et moyenne en ligne ou en colonne d'une matrice.
- `max, min` : maximum et minimum d'un vecteur.
- `quantile(x,0.1)` : quantile d'ordre 10% du vecteur `x`.

Graphiques
- `plot(x)` : représente une série de points (ordonnée `x` et numéro d'indice en abscisse).
- `plot(x,y)` : représente un nuage de points d'abscisse `x` et d'ordonnée `y`.
- `image(x,y,z)` : représente en niveau de couleur une image où `z` représente l'intensité au point `x,y` (`z` est une matrice dont le nombre de ligne est la longueur de `x` et le nombre de colonne celle de `y`).
- `lines, points` : ajoute une ligne ou des points sur un graphique existant.
- `hist` : histogramme.
- `barplot` : graphique en barre.
- `abline` : représente une ligne en précisant la pente `b` et l'ordonnée à l'origine `a`. Une ligne verticale d'abscisse `x` (`v=x`) ou horizontale d'ordonnée `y` (`v=y`)

- `legend` : ajoute une légende en précisant les symboles (`lty` ou `pch` et `col`), le texte (`text`) et l'emplacement (`x="topright"`).
- `axis` : ajoute un axe. Argument : `side` (1 : bas, 2 : gauche, 3 : haut, 4 : droite).
- `grid` : ajoute un quadrillage.
- `par(mfrow=c(n,p))` : partage la fenêtre graphique en n×p sous graphiques.

Paramètres graphiques

- `type` : `"l"` pour ligne et `"p"` pour points.
- `col` : `"black"`, `"red"`, `"blue"`, `"red"` ... (ou 1, 2, 3, 4...)
- `lty` : type de lignes (1 : solide, 2 : pointillée...).
- `pch` : type de points (1 : cercle, 2 : triangle...).
- `main` : titre principale.
- `xlab, ylab` : titre des axes.
- `log` : échelle logarithmique (`"x"` pour l'axe des abscisse, `"y"` pour l'axe des ordonnées, `"xy"` pour les deux axes).