Une introduction à RStudio pour les statistiques

Préambule

Ce TP introductif est très largement issu de Wikistat, site web sur lequel de nombreuses ressources sont disponibles, y compris plusieurs tutoriels sur R :

- *Démarrer rapidement avec R*
- *Initiation à R*
- *Fonctions graphiques de R*
- *Programmation en R*
- *MapReduce pour le statisticien*

On introduit ici l'environnement RStudio, et les bases du langage de programmation R, que nous utiliserons pour de l'analyse de données et de l'inférence statistique L'objectif est donc de se familiariser avec l'interface et les commandes que nous serons amenés à utiliser.

1 Une présentation de R et RStudio

1.1 R, un langage de programmation statistique

R, sous licence GNU, est un des logiciels les plus utilisés dans la communauté statistique académique et aussi de plus en plus dans les services R&D des entreprises industrielles en concurrence avec les logiciels commerciaux. Son utilisation récurrente nécessite un apprentissage à travers des tutoriels à l'instar ceux listés dans le préambule mais il est facile de démarrer à partir de quelques notions de base sur son utilisation ; c'est l'objectif de ce premier TP.

Dans sa structure, R est un langage de programmation d'une syntaxe voisine à celle du langage C et capable de manipuler des objets complexes sous forme de matrice, scalaire, vecteur, liste, facteur et aussi *data frame*. Il offre des fonctionnalités analogues à Matlab et dispose également de nombreuses librairies qui contiennent la plupart des procédures et méthodes statistiques de la littérature.

1.2 Premiers pas avec RStudio

RStudio est un environnement de programmation adapté au langage R, dont il existe une version gratuite et libre, téléchargeable en ligne. C'est à travers lui que nous programmerons en R.

1.2.1 Lancement RStudio

- Sous Windows Cliquer sur l'icone RStudio ou lancer le programme à partir du menu `Démarrer`. Lors de la première exécution, il est utile de préciser le répertoire de travail (menu `Fichier` de R). Il est aussi possible de lancer RStudio à partir de la session précédente en cliquant sur le fichier de sauvegarde de suffixe `.RData`.
- Sous Linux ; à partir d'une fenêtre de commande, se placer dans le bon répertoire pour taper la commande `rstudio`.

S'ouvre alors une fenêtre RStudio comprenant 4 onglets :

- en haut à gauche : un éditeur de texte pour gérer le script à exécuter, sauvegarder pour constituer l'annexe du rapport ;
- en haut à droite : une liste des objets en mémoire (*Environment*) ou l'historique des commandes (*History*) ;
- en bas à gauche : la console d'exécution de R ; on peut y exécuter directement du code, ou y lire les résultats des commandes d'un script ;
- en bas à droite : une fenêtre dans laquelle on peut notamment visualiser l'aide de RStudio ou les graphiques obtenus.

1.2.2 Gestion d'un script

Un script est un fichier d'extension *.R* consistant d'une succession de commandes écrite dans le langage de programmation R ; c'est la façon la plus simple de sauvegarder votre travail.

- Créer un script : **File > New File > R Script**.
- Enregistrer un script : **File > Save as...**
- Ouvrir un script : **File > Open File...**

- Exécuter un script :
  1. pour exécuter un script, il faut **s'être préalablement placé dans le bon répertoire** (c'est-à-dire le dossier où est enregistré le script) en effectuant la manipulation suivante : **Session > Set Working DIrectory > To Source File Location** ;
  2. pour exécuter une ligne de code : il faut placer le curseur à la ligne correspondante (avec le clavier ou la souris) et cliquer sur l'icône **Run** ou utiliser le raccourci clavier **Ctrl + Entrée** ;
  3. pour exécuter plusieurs lignes de code : il faut sélectionner le code que l'on souhaite exécuter et procéder comme ci-dessus.

1.2.3 Editer un rapport avec un fichier R Markdown

Vous pouvez également créer directement un rapport grâce a ? R Markdown, avec la possibilité d'éditer directement des documents au format .pdf ou .html (en incluant les commandes R, les sorties et les graphes). Pour cela, il faut créer un nouveau document (d'extension .Rmd) : **File > New File > R Markdown...**

Il faut alors préciser le type de document final souhaité (document au format .html ou .pdf ou présentation, etc), ainsi que les informations du document (titre et auteur). N'oubliez pas d'enregistrer votre document. Pour l'éditer, vous pouvez taper du texte (dont des commandes LateX), et insérer du code R à l'intérieur de blocs délimités par "'r et "'. Il ne reste plus qu'à compiler le document en cliquant sur le bouton **Knit**. Le fichier de sortie s'affiche alors.

Plus d'informations peuvent être trouvées sur le site internet http://rmarkdown.rstudio.com/ ou ici.

1.2.4 L'aide de RStudio

... apparaît dans la sous-fenêtre en bas à droite en cliquant sur **Help**.

Si vous cherchez à en savoir plus sur une commande particulière (notamment les arguments entrée et la nature de la sortie), il suffit, par exemple ici pour la commande *plot*, de rentrer dans la console (en bas à gauche) la commande **help(plot)** (ou **?plot**).

Il ne vous est pas demandé de connaître par coeur les innombrables commandes de R ; il est cependant très utile (et votre enseignant(e) vous en sera fort reconnaissant(e)) de savoir utiliser l'aide pour les utiliser.

2 Les commandes de base

La liste ci-dessous est loin d'être exhaustive mais les blocs de code proposés dans la suite devraient vous permettre de commencer à vous familiariser avec le langage et l'interface.

Vous pouvez taper les lignes ci-dessous après l'invite de commande **>** directement dans la console en bas à gauche. De manière générale, il vous est plutôt conseillé d'enregistrer vos commandes dans un Script R, ou éditer un document R Markdown afin de garder une trace de votre travail.

2.1 R, une calculatrice

R fonctionne comme une calculatrice, et manipule les fonctions usuelles ainsi que les opérateurs logiques.

```
# Une ligne commençant par le symbole dièse est
# considérée comme un commentaire,
# et n'est pas compilée par R.

2+2
sqrt(2)
a = exp(2)   # création d'une variable scalaire
b = a + pi
b            # affichage de la valeur
# liste des variables
ls()

#Opérations logiques
2>=4
8<99
8/3==4
8/3!=4
```

Comprenez-vous les opérations ci-dessous ?

2.2 Des objets très différents

Des objets très différents peuvent être construits et manipulés avec R... la difficulté principale résidant dans la bonne utilisation de chacun !

Tout d'abord les vecteurs et les matrices...

```
# Vecteur
x = 1:10  # définition d'une séquence
x
y = 2*x + 3
y[5] ; y[1:3] ; y[-3]  # composants d'un vecteur

z=c(3,5,9);
z
zz=c(9,z);
zz

# Matrice
A = matrix(1:15,ncol=5); A
B = matrix(1:15,nc=5,byrow=TRUE) ; B
A[1,3] ; A[,2] ; A[2,] ; A[1:3,1:3] # composants
t(A) #transposée
dim(A)
```

Les listes permettent de stocker des variables de différents types alors que les bases de données (ou *data frames*) sont des tableaux contenant de vecteurs de même taille, mais pas forcément de même type.

```
# Liste
M=list(mat=A, texte="testliste",vec=y)
M[[2]] ; M$vec # composants

# Base de données ou data frame
# Tableau contenant des vecteurs de types
# éventuellement différents
taille = c(147, 132, 156, 167, 156, 140)
poids = c( 50, 46, 47, 62, 58, 45)
sexe = c("M","F","F","M","M","F")
H = data.frame(taille,poids,sexe)
H
summary(H)
plot(H$poids,H$taille)
```

3 Des outils statistiques

Nous nous raprochons du vif du sujet : nous regardons notamment dans cette section comment simuler des variables aléatoires et construire des histogrammes ; nous proposons également une illustration graphique du théorème central limite.

3.1 Estimation

On obtient un échantillon de taille n de la loi $\mathcal{N}(80, 25)$; on en fait une rapide analyse.

```
n=10 #n à choisir
Y=rnorm(n,80,5)  # génération de l'échantillon
mean(Y)      # moyenne
sd(Y)        # écart-type
summary(Y)   # données statistiques de base
boxplot(Y)   # diagramme boîte
# histogramme de la densité
hist(Y, probability=T, col="blue")
# tracer la loi théorique
x=1:100
curve(dnorm(x,mean=80,sd=5),add=TRUE,
   col="green",lwd=2)
```

La commande rnorm vous semble-t-elle cohérente ?
Si vous répondez par la négative, c'est, au choix, que n est trop petit ou qu'il y a une erreur dans votre code... que se passe-t-il quand n augmente ?

3.2 Théorème de la limite centrale

La simulation proposée illustre le résultat fondamental du théorème de la limite centrale : une somme de variables aléatoires indépendantes et de même

loi converge vers une variable aléatoire de loi gaussienne. Le programme ci-dessous exécute les opérations suivantes :

- initialisation par des 0 d'un vecteur de taille $n = 1000$,
- chaque valeur de ce vecteur est une variable aléatoire X obtenue par la somme de N variables suivant une loi uniforme sur l'intervalle $[0, 1]$,
- estimation de la densité de X
- comparaison avec la loi théorique limite qui est la loi gaussienne de moyenne $N/2$ et de variance $N/12$.

```
n=1000
N=12
X=rep(0,n)
# n itérations
for (i in 1 : n) X[i]=sum(runif(N))
# histogramme
hist(X, col="blue", probability=T)
# estimation par méthode su noyau
lines(density(X), col="red", lwd=2)
x=X
sigma2=N/12
curve(dnorm(x,mean=N/2,sd=sqrt(sigma2)),
   add=T, col="green", lwd=2)
```

Faire varier $N = 4, 8, 12, 20$. Remarquer que la convergence est très rapide. Ceci "justifie" la pratique qui revient à considérer que l**a loi d'un estimateur est gaussienne lorsque** n **est "suffisamment" grand avec** $n > 30$.