Start-R pour utilisateur de statistiques

Résumé

Cette vignette introduit le logiciel libre R, son environnement et décrit les premières commandes nécessaires au démarrage d'une utilisation de méthodes statistiques avec ce logiciel. Les notions d'estimation, de variabilité et de loi d'un estimateur sont illustrées par des simulations. Les autres tutoriels abordent des fonctionnalités plus complexes.

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

Les aspect statistiques sont développés dans les différents scénarios de Wikistat et d'autres ressources sont disponbibles.

1 Introduction

1.1 Pourquoi R ?

Le logiciel R sous licence GNU est facile à installer à partir de la page du CRAN ou d'un site miroir ; ils contiennent toutes les ressources nécessaires à l'utilisateur de R, débutant ou expérimenté : fichiers d'installation, mises à jour, librairies, FAQ, newsletter, documentation... Il est le logiciel le plus utilisé de 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 nécessite un apprentissage à travers des tutoriels comme par exemple ceux listés dans le résumé mais il est facile de démarrer à partir de quelques notions de base sur son utilisation ; c'est l'objectif de ce start-R.

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*. Proposant donc une programmation matricielle, il offre des fonctionnalités analogues à Matlab et dispose également d'une très riche librairie de quasiment toutes les procédures et méthodes statistiques de la littérature. Plus précisément, toutes les recherches récentes sont d'abord développées et diffusées à l'aide de ce logiciel par la communauté scientifique.

1.2 Utilisation

Il existe de nombreuses librairies (cf. `Rcmdr`) d'interface graphique par menu mais celles-ci sont contraignantes, trop limitées dans les choix et options, elles ne peuvent éviter une utilisation par lignes de commandes ; autant s'y mettre tout de suite, c'est le choix fait ici.

Il existe également un environnement de programmation ou IDE : RStudio relativement efficace ; à l'utilisateur de faire ses choix. Nous nous limitons ici aux choix de l'interface graphique par défaut `Rgui` (*R graphical user interface*) et à celui de l'usage de RStudio si ce dernier est installé.

1.3 Ouvrir, utiliser, fermer R

- Sous Windows Cliquer sur l'icone (R ou 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 R 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 `R` ou `rstudio`.
- Dans les deux cas, le logiciel répond avec une invite de commande : > ou ouvre (RStudio) sous la forme d'une fenêtre avec 3 sous-fenêtres. En ouvrir une 4ème avec le menu : `File > New File > Sript R`. Ces 4 sous-fenêtres sont comme sos Matlab :
  - un éditeur de texte pour gérer le script à exécuter, sauvegarder pour constituer l'annexe du rapport,
  - une liste des objets (environnement) créés dans R ou l'historique des commandes,
  - la console d'exécution classique de R,
  - une fenêtre avec menu pour visualiser le répertoire, les graphiques, la liste des *packages* installés pour les charger, mettre à jour, en installer d'autres, l'aide en ligne.

`help(plot)` ou `plot?` lance la fonction `help` qui ouvre l'aide en ligne de, par exemple, la fonction `plot`. `quit()` fait quitter R après une question proposant de sauvegarder l'environnement de travail. Y répondre favorablement crée le fichier `.RData` dans le répertoire courant ainsi qu'un fichier contenant l'historique des commandes.

1.4 Les tutoriels

Les tutoriels sont organisés de la façon suivante :

- une succession de commandes à saisir est donnée. Chaque commande est mise en évidence par une graphie particulière `ls()`.
- *Attention*, lorsque R identifie une commande incomplète (parenthèse ouverte et pas fermée par exemple), le symbole "+" apparaît automatiquement sur la ligne suivante.
- Deux commandes indépendantes peuvent être tapées sur la même ligne séparées par un " ;". Elles seront traitées séquentiellement.
- Quelques remarques ou commentaires viennent parfois mettre en évidence certains points particuliers de syntaxe ou autre.
- Des questions permettent de tester l'acquisition des fonctions étudiées.
- Une réponse possible à chaque question est fournie à la suite.

2 Lignes de commande

2.1 Avec Rgui

R fonctionne comme une calculette. Vous pouvez taper les lignes ci-dessous après l'invite de commande (>). Il est recommandé d'ouvrir en plus la fenêtre d'un éditeur pour mémoriser les commandes et copier / coller ces lignes dans la fenêtre R avant de les archiver dans un fichier pour constituer l'annexe d'un rapport.

2.2 Avec RStudio

Cette fenêtre d'édition est incluse comme indiqué précédemment par l'ouverture d'un nouveau fichier de script R.

Le menu : `Session > Set Working Directory` permet de sélectionner le répertoire par défaut contenant par exemple le fichier de données à lire et où seront sauvegardés script, environnement et historique des commandes d'une session.

2.2.1 Premières commandes

Entrer les lignes ci-dessous dans la fenêtre de l'éditeur de texte. Dans Rgui, les copier / coller dans la console pour les exécuter. Dans RStudio, positionner le curseur sur une ligne et cliquer sur le bouton `Run] pour l'exécuter ou sélectionner préalablement plusieurs commandes.`

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

3 Types de variables

`La principale difficulté dans l'utilisation de R est de bien identifier les types d'objets manipulés.`

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

# 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

# Liste
x=list(mat=A, texte="testliste",vec=y)
```

```
x[[2]] ; x$vec # composants

# Base de donnée 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)
```

D'autres fonctionnalités de R sont vues dans les différents scénarios proposées sur le site Wikistat.

4 Variables et nombres aléatoires

Tout en complétant la connaissance de R, cette section propose d'illustrer, par des simulations, les propriétés des estimateurs élémentaires (moyenne, écart-type, histogramme).

4.1 Estimation

Générer n valeurs aléatoires d'une variable Y selon une loi normale de moyenne 80 et d'écart-type 5. Décrire sommairement cette série de valeurs. Associer les quantités calculées avec leur traduction en anglais *mean, median, standard error, standard déviation, standard error mean*.

```
n=10
Y=rnorm(n,80,5)  # génération
mean(Y)      # moyenne
sd(Y)        # écart-type
sd(Y)/sqrt(length(Y)) # écart-type de la moyenne
summary(Y)  # quartiles et moyenne
boxplot(Y)  # diagramme boîte
# histogramme de la densité
hist(Y, probability=T, col="blue")
# estimation par la méthode du noyau
lines(density(Y), col="red", lwd=2)
# tracer la loi théorique
x=1:100
curve(dnorm(x,mean=80,sd=5),add=TRUE,
   col="green",lwd=2)
```

4.2 Loi des grands nombres

Une moyenne et un écart-type sont la réalisation d'une variable aléatoire appelée "estimateur"; ce sont des estimations. Refaire les calculs et graphiques en posant $n = 10, n = 1000, n = 10000$; comparer les résultats obtenus, notamment les estimations des indicateurs par rapport aux valeurs théoriques. Etudier leur comportement en fonction de la taille n de l'échantillon.

```
n=100
# matrice de nombres aléatoires de
# 10 colonnes et n lignes
Y=matrix(rnorm(n*10,80,5),n,10)
# moyenne de chaque colonne
apply(Y,2,mean)
mean(apply(Y,2,mean)) # moyenne des moyennes
# écart-type de chaque colonne
apply(Y,2,sd)
mean(apply(Y,2,sd))    # moyenne des écarts-types
```

Faire varier $n = 10, 100, 1000$ et comparer les résultats obtenus.

4.3 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 la loi d'un estimateur est gaussienne lorsque n est "suffisamment" grand avec $n > 30$.