Introduction à R

Florence Yerly

Dept. de mathématiques, Université de Fribourg (CH)

SP 2011

Qu'est ce que R ?

- Un logiciel de statistiques libre et gratuit ;
- Un logiciel multi-plateforme (UNIX, Windows MacOS X)
- R permet de faire des calculs statiques avancés ainsi que des graphiques de haute qualité ;
- Très utilisé dans les domaines académiques, mais aussi dans le domaine privé ;
- Un clone du logiciel commercial S.

On peut télécharger R ici : `http://cran.r-project.org`
On y trouve aussi des packages supplémentaires, des tutoriels (en plusieurs langues), etc.

Avantages et désavantages

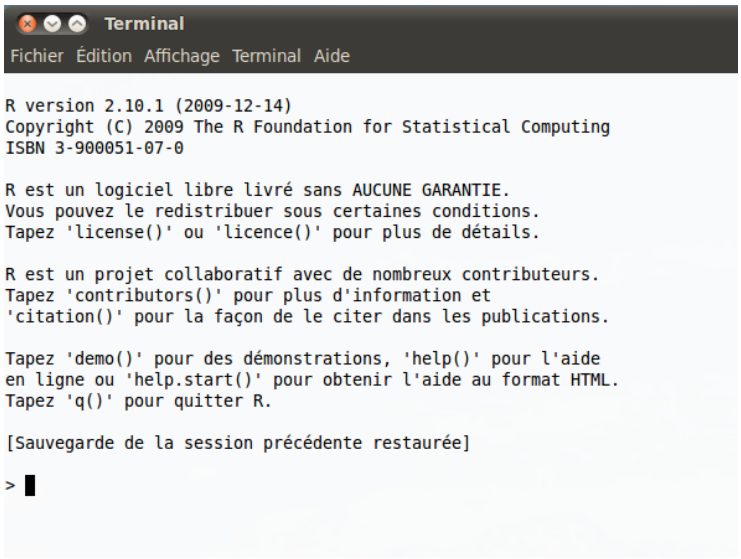
Avantages :

- Logiciel libre, donc tout le monde peut contribuer et modifier les fichiers à sa convenance.
- Rapidité d'insertion des nouveaux outils statistiques

Désavantages :

- L'interface graphique est très pauvre.
- Pas vraiment d'unité entre les différents packages.

Voici à quoi ressemble R :



`>` invite l'utilisateur à entrer une commande.

`+` indique que la commande précédente n'est pas terminée. On peut stopper un processus avec la combinaison Ctrl+C.

`q()` est la commande pour quitter R, on peut ensuite sauver ou non son travail en tapant `y/n/c` (oui, non, annuler)

- Tous les objets créés en R sont stocké (si demandé) dans un fichier .RData.
  - > ls() donne la liste des objets existants
  - > rm(x) enlève l'objet x
  - > rm(list=ls()) enlève tous les objets existants
- On navigue dans les anciennes commandes avec les flèches ↑ et ↓ du clavier.
- getwd() donne le dossier de travail de R, on peut le modifier en utilisant setwd(*path*).

R se résume à une console qui exécute des commandes. Pour garder une trace du travail, il est vivement conseillé de copier les commandes dans un script.

Pour cela, un simple éditeur de texte suffit (Bloc-Note de Windows, par ex.).

Un très bon éditeur pour Windows : Tinn-R
http://www.sciviews.org/Tinn-R/

La commande source(*file.r*) permet d'executer un script *file.r* dans la console. Les commentaires sont signalés par #.

Commande d'aide

`> help.start(.)`	lance l'aide HTML en ligne de `R`.
`> help(help)`	documentation de la fonction `help`.
`> ?help`	idem
`> help.search("test")`	cherche dans tous les packages installés les fonctions qui correspondent à `test`.
`> example(mean)`	exécute les exemples liés à la fonction `mean`.

Convention en R

- `R` ignore les espaces et tout ce qui se trouve après #.
- `R` fait la différence entre les majuscules et les minuscules.
- On peut utiliser `R` comme une calculatrice

```
> 3+4
[1] 7
```

Le résultat n'est pas mis en mémoire.

- `> x= 3+4` ou `> x<- 3+4`
  n'affiche rien, mais le résultat est mis en mémoire, on y accède ainsi :

```
> x
[1] 7
```

Vecteurs en R

`> x=c(1,3.2,4,6,10.2,-11)`	crée un vecteur avec une **c**ollection de nombres.
`> y=1:10`	
`> x[2]`	renvoie la deuxième valeur de `x`.
`> y[-3]`	renvoie le vecteur `y` sans la troisième valeur.
`> x[1:4]`	renvoie les 4 premières valeurs de `x`.

Autres fonctions importantes pour créer des vecteurs :

```
seq(from=0,to=1,by=0.25)
rep(1:3,1:3)
```


Opérations et fonctions

Opérations (terme par terme)	`'+','-','*','/','^'`
Multiplication matricielle	`'%*%'`
Division entière	`'%/%'`
x Modulo y	`'x%%y'`

Quelques fonctions :

```
abs, exp, log, log10, sqrt, sin, cos, tan, max, min,
sum, sort, var, mean, gamma, sign, length, sinh, ...
```

Matrices en R

3 manières de procéder :

- en combinant des vecteurs lignes, `rbind`
  ```
  > x.mat=rbind(1:5,6:10,1)
  ```
- en combinant des vecteurs colonnes, `cbind`
  ```
  > y.mat=cbind(1:5,6:10,1)
  ```
- en transformant un vecteur en matrice, `matrix`
  ```
  > z.mat=matrix(1:12, ncol=4)
  ```

`x.mat[1,2]`	retourne l'élément de la première ligne, deuxième colonne de la matrice
`x.mat[1,]`	retourne la première ligne de la matrice
`t(.)`	transpose la matrice
`solve(.)`	calcule l'inverse de la matrice

Matrices en R

`nrow(.)`	retourne le nombre de lignes
`ncol(.)`	retourne le nombre de colonnes
`dim(.)`	retourne (n, m) où n est le nombre de lignes et m le nombre de colonnes
`length(.)`	retourne le nombre d'éléments

Autres fonctions : `diag, eigen, chol, qr, ...`

Chaîne de caractères et listes

Il est possible de créer des vecteurs avec des chaînes de caractères :

```
> x=c("Lundi","Mardi","Mercredi")
```

Une liste permet de stocker plusieurs objets de différents types :

```
> mylist=list(Day=x,Location="Fribourg",Temp=c(3,6,2))
> mylist
$Day
[1] "Lundi" "Mardi" "Mercredi"
$Location
[1] "Fribourg"
$Temp
[1] 3 6 2
```

Accès aux données :

`mylist$Day` ou `mylist[[1]]`

`mylist$Temp[2]` ou `mylist[[3]][2]`

Tableau de données : Data frames

C'est certainement l'objet R le plus utilisé ! Il permet de stocker des données de différents types mais de même grandeur.

```
> classeAB=data.frame(Nom=c("Alain","Zoé","Anne"),
                      Note=c(3.5,5.5,4))
```

```
> names(classeAB)
[1] "Nom" "Note"
```

donne le nom des colonnes

```
> classeAB$Note
[1] 3.5 5.5 4
```

accès à une colonne

Valeurs logiques

==	égal à
!=	n'est pas égal à
> (<)	plus grand que (plus petit que)
>= (<=)	plus grand ou égal à (plus petit ou égal à)
&	et logique
|	ou logique
!	négation

```
x=c(1,3.2,4,6,10.2,-11)
> x > 3
[1] FALSE TRUE TRUE TRUE TRUE FALSE
> x[x>3 & x<=6]
[1] 3.2 4 6
```


Importer et exporter des données

Avec R, on peut importer des données enregistrées dans des fichiers .txt, .dat, .csv :

- `> file.dat=read.table("file.dat",header=TRUE)`
  Importe les données du fichier file.dat qui contient aussi les entêtes des colonnes.
- `> file.csv=read.csv("file.csv",header=TRUE,sep="\t")`
- `> file.txt=read.table("file.txt",header=TRUE,sep=" ")`

Pour exporter un data frame : `write.table(x,file,...)` ou `write.csv(x,file,...)`

Packages

Les packages sont des modules supplémentaires qui peuvent contenir des fonctions et des bases de données.

`library()`	indique les packages installés sur la machine
`install.packages("ISwR")`	installe le package ISwR sur la machine ou voir dans les menus
`library(ISwR)`	charge le package, le rend disponible
`require(ISwR)`	idem
`help(package=ISwR)`	donne des informations sur le package ISwR
`data(package="ISwR")`	répertorie les bases de données de ISwR
`data(tlc)`	charge (rend disponible) la base de données tlc
`summary(tlc)`	résume les données contenues dans tlc

Variables de distribution standard

R possède de nombreuses fonctions de distribution déjà implémentées.

Code	Distribution	paramètre
`binom`	Loi binomiale	n, p
`geom`	Loi géométrique	p
`pois`	Loi de Poisson	λ
`unif`	Loi uniforme	min, max
`exp`	Loi exponentielle	rate
`norm`	Loi normale	mean, sd
`chisq`	Loi du χ^2	degré de liberté
`t`	Loi de Student	degré de liberté

Il suffit de mettre une des 4 lettres suivantes devant le nom de code de la distribution pour obtenir :

- `d` la densité, `dnorm`
- `p` la probabilité, `pnorm`
- `q` le quantile, `qnorm`
- `r` une variable aléatoire, `rnorm`

Exemples :

`> dexp(1,2)`	donne la valeur en 1 de la densité pour une v.a. exponentielle de paramètre 2
`> pbinom(2,10,0.1)`	probabilité que la binomiale de paraètre 10 et 0.1 soit égale à 2
`> qnorm(0.05)`	donne le t pour lequel $\mathrm{P}(X < t) = 0.05$ pour une v.a. normale
`> rpois(10,2)`	donne 10 réalisations d'une v.a. de Poisson de paramètre 2

Outils graphiques

`R` possède de nombreux outils graphiques de haute qualité (2D et 3D).

`windows()`	pour ouvrir une fenêtre graphique (Windows)
`quartz()`	pour ouvrir une fenêtre graphique (MacOS X)
`x11()`	pour ouvrir une fenêtre graphique (UNIX)
`par(...)`	spécifie les options graphiques de la fenêtre
`par(mfrow=c(2,2))`	crée une fenêtre graphique avec 4 sous-graphiques arrangés sur 2 lignes et 2 colonnes
`par(pty="s")`	le graphique sera carré
`help(par)`	pour plus d'informations ...

Différents type de graphiques

Quelques exemples :

`plot(x,y)`	dessine x en fonction de y (points)
`plot(x,y,type="l")`	avec l'option "ligne"
`plot(x,y,col="red")`	avec l'option couleur rouge
`barplot,hist`	dessine un "barplot", un histogramme
`boxplot`	dessine un "boxplot"
`pie`	dessine un camenbert
`pairs`	dessine tous les couples de coordonnées à partir d'un jeu de données.

...

Ce que l'on peut ajouter à un graphique existant ...

`abline`	ajoute une droite d'ordonnée à l'origine et de pente donnés
`axis`	ajoute des axes
`box`	ajoute une boîte autour du graphique
`legend`	ajoute une légende
`lines, points`	ajoute des lignes, des points
`title`	ajoute un titre
`mtext, text`	ajoute du texte dans la marge ou dans le graphique

...

Programmation en R

Plusieurs commandes R peuvent être groupées :

```
{expression1 ; expression2 ; ...}
```

⇝ la valeur du groupe est la valeur de la dernière expression.

Boucle "For" :

```
for (index in values) {expressions}
```

Boucle "While" :

```
while (condition) {expressions}
```

Evaluation conditionnelle "If" :

```
if (condition) {expression1} else {expression2}
```

Exemples :

```
for (i in 1:10) print(i)
while(i< 11) {print(i) ; i=i+1}
if (x>0) y=x*log(x) else y=0
```


Créer des fonctions en R

```
name=function(arg1,arg2,...){expressions}
```

Exemple:

```
> fun1=function(x,y){ (x+y)^2}
> fun1(1,1)
[1] 4
> fun1(1:4,5)
[1] 36 49 64 81
```