1 Introduction

• langage interprété
• environnement de programmation orienté statistiques et calcul scientifique
• fonctions de haut niveau pour la lecture et l'écriture de données, l'affichage de graphiques, et les calculs vectoriels et matriciels.
• nombreuses librairies de calculs et d'anlyse de données
• site de documentation et mise à disposition de package : `http://cran.r-project.org/`

2 Les bases

2.1 Première session

• lancement : `R`
• le prompt : `>`
• affiche les résultats après chaque commande.
• caractère de continuation de ligne : `+`
• affectation : `<-`, `->`, `<<-`, `=`
• aide : `help()`, `help.start()`
• quitter : `q()`
• exécuter un script R : `source(''`*nomdefichier*`)''`
• Fichiers d'environnements :
`.RData` sauvegarde de l'environnement, `.Rprofile` script d'initialisation, `.First` première fonction executée.

exemples :

```
> x<-1
> x
[1] 1
> 2->x
> x
[1] 2
> 2*x+log(x)
[1] 4.693147
> x="toto"
> x
> help(q)
> q()
```

2.2 Les types de données simples

Les types gérés par R sont : `integer`, `double`, `numeric`, `complex`, `logical` et `character`.

2.3 Les opérations élémentaires

• Les opérations arithmétiques élémentaires : `+`, `-`, `*`, `/` et `^`.
• Les opérations logiques : `!`, `&`, `&&`, `|`, `||`.

3 Les structures de données

3.1 Les vecteurs

• création : les fonctions `c()`, `seq()`, `rep()`, `vector()`, `double()`, `character()`, `<`*type*`>()`.
• indexation : opérateurs 'subset' `[]`, 'element of' `[[]]`

exemples :

```
> x=1:50
> x[8]
> x=double(100)
> x=c(''toto'',''titi'', ''tutu'')
> x[2]
```

3.2 Les listes

• création : `list()`
• indexation : `[]`, `[[]]` et `$`

exemples :

```
l=list(A=c(1,2,3), B=c(T,T))
> l[2]
[[1]]
[1] TRUE TRUE
> l[[2]]
[1] TRUE TRUE
>l$B
```

3.3 Les matrices

- création :`matrix()`
- indexation :`[,]`
- fonctions `cbind` : concaténation de colonnes, `rbind` : concaténation de lignes

exemples :

```
> A=matrix(0,2,3)
> A=matrix(c(1,2,3,7,6,5),2,3)
> A[1,2]
[1] 3
> A[1,2]=2
```

3.4 Les tableaux de données

- création :`data.frame()`
- indexation : `[, ]`, et `$`

exemples :

```
> D=data.frame(noms=c("toto", "titi", "tutu"),
+ages=c(8,9,10))
> D$ages
[1]  8  9 10
> D[3,2]
[1] 10
```

3.5 Les facteurs

- création :`factor()`
- indexation : `[, ]`
- manipulation `levels()`

exemples :

```
> pain=c(0,3,2,2,1)
> fpain=factor(pain)
> levels(fpain)
> levels(fpain)=c("none", "mild", "medium", "severe")
```

3.6 Manipulation des types

- fonctions *type*(), `is.`*type*(), `as.`*type*()

, par exemple `is.matrix()`, `as.matrix()`, `matrix()`

4 Programmation vectorielle et matricielle

- le vecteur est le type de base en R
- toutes les fonctions fonctionnent vectoriellement

exemples :

```
> x=seq(-pi, pi, len=100)
> y=sin(x)
> sum(x)
```

- sous-ensemble d'un vecteur ou d'une matrice : on peut utiliser un vecteur d'indices entre `[` et `]`
- on peut aussi utiliser une condition logique entre `[` et `]`
- la fonction `which` renvoie l'ensemble d'indices des éléments satisfaisant une condition logique.
- les fonctions conditions logiques `any` et `all` sur un vecteur logique

exemples :

```
> z= y[10:36]
> y[y>0]
> which(y>0)
> A=matrix(1:16,4,4)
> A[,2] # 2eme colonne de A
> A[2,] # 2eme ligne de A
> A[A>11]=0
```

- concaténation de matrices : `cbind` et `rbind`
- multiplication de matrices : `%*%`
- transposition de matrices : `t()`
- calcul de valeurs propres et vecteurs propres : `eigen()`
- déterminant : `det()`
- inversion de matrices et résolution de systèmes linéaires : `solve()`
- décomposition QR : `qr()`

```
> A=matrix (c(2,-1,-1,2), 2,2)
> solve(A) # inverse de A
> B=t(A) # transposée de A
> b=c(1,1)
> x=solve (A, b) # résolution de Ax=b
```

5 La programmation R

On note `expr` une seule expression ou un bloc d'expressions et `cond` une expression booléenne.

5.1 Les tests

- `if (cond) expr`
- `if(cond) expr else expr`

exemples :

```
> x=1
> if (x==1) print("x = 1") else
+ print ("x != 1")
```

5.2 Les boucles

- `for(var in seq) expr`
- `while(cond) expr`

exemples :

```
> for (i in 1:10) print(i)
> noms=c("tata", "titi", "toto", "tutu")
> i=1
> while(noms[i]!="toto") {i=i+1}
> print(paste("toto est en position ", i))
```

5.3 Les fonctions

- déclaration :

nomdefonction = `function(`*nomarg=expr*`, ...)` { *expr,...*`;return(`*valeurderetour*`)`}

- Une fonction comporte des arguments obligatoires et des arguments optionnels.
- chaque argument à un nom
- les arguments optionnels ont une valeur par défaut
- appel de la fonction :

`valeur =` *nomdefonction*`(`*nomarg=expr,...*`)`

Lorsque le nom de l'argument est spécifié l'ordre des arguments est sans importance. Si la valeur argument n'est pas spécifiée, sa valeur par défaut est utiliséé par la fonction.

exemples :

```
> f=function(x,y) {
+ z=cos(y)/(1+x*x);
+ return (z)}
> val=f(1,2)
> val=f(y=2,x=1)
> f=function(x,y=1) {
+ z=cos(y)/(1+x*x);
+ return (z)}
> val=f(1)
```

6 Les graphiques

6.1 La fonction `plot`

`plot(x, y, xlim=range(x), ylim=range(y), type="p", main, xlab, ylab, ...)`

exemples :

```
> x=seq(0,10,0.2)
> plot (x, sin(x))
> plot (x, sin(x), type="l",
+ col="red", xlab="x", ylab="y")
```

En général, ne pas superposer des `plot` avec `par(new=FALSE)` Mais modifier un graphique ouvert par `plot`, avec les fonctions de bas-niveau `points`, `lines`, `rect` , `text`, `legend`, `title`, `axis`, ...

exemples :

```
> x=seq(-pi, pi, len=65)
> plot(x, sin(x), type="l", ylim=c(-1.2, 1.8),
+col=3, lty=2)
> points(x, cos(x), pch=3, col=4)
> lines(x, tan(x), type="b",
+ lty=1, pch=4, col=6)
> title("test de graphique")
> legend(-1,1.9, c("sin", "cos", "tan"),
+ col=c(3,4,6), lty=c(2,-1,1), pch=c(-1,3,4),
+  merge=T, bg="gray90")
```

6.2 Autres types de graphiques

- histogrammes : `hist()`
- graphiques barres `barplot()`
- lignes de niveaux : `contour()`
- surfaces : `persp()`
- images :`image()`

7 Exportation et importation de données

7.1 Exportation

- écriture générique dans un fichier `cat`
- écriture d'un vecteur ou d'une matrice `write`
- écriture d'une Data Frame `write.table`

exemples :

```
> x = 1:10
> write(x, file="test.dat")
> x = matrix(1:10,ncol=5)
> write(t(x), file="test2.dat")
> D=data.frame(noms=c("toto", "titi", "tutu"),
+ ages=c(8,9,10))
> write.table(D, file="test3.dat")
```

7.2 Importation

- lecture générique `scan`
- lecture d'un vecteur `scan`
- lecture d'une matrice ou d'une Data Frame `read.table`
- lecture d'un fichier texte `readLines`

exemples :

```
> scan("test.dat")
> scan("http://www.ebgm.jussieu.fr", what="")
> read.table("test3.dat")
> A=as.matrix(read.table("test2.dat"))
```