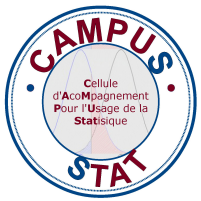
Introduction au logiciel R

Sébastien Déjean

math.univ-toulouse.fr/~sdejean

Institut de Mathématiques de Toulouse UMR 5219
Université Paul-Sabatier (Toulouse III)



Documents complémentaires et mises à jour :

math.univ-toulouse.fr/~sdejean

INSTITUT de MATHEMATIQUES de TOULOUSE — INSTITUT DE MATHÉMATIQUES DE TOULOUSE
PLATEFORME DE BIOSTATISTIQUE — geno toul biostat

Pour se donner un peu d'R

SÉBASTIEN DÉJEAN

29 octobre 2008
Mises à jour et compléments :
math.univ-toulouse.fr/~sdejean

Institut de Mathématiques de Toulouse UMR 5219 — Université de Toulouse et CNRS
Plateforme de Biostatistique — Génopôle Toulouse Midi-Pyrénées

INSTITUT de MATHEMATIQUES de TOULOUSE — INSTITUT DE MATHÉMATIQUES DE TOULOUSE
PLATEFORME DE BIOSTATISTIQUE — geno toul biostat

Encore besoin d'R ?

SÉBASTIEN DÉJEAN(*), THIBAULT LAURENT(+)

(*) Institut de Mathématiques de Toulouse, Sebastien.Dejean@math.univ-toulouse.fr
(+) Toulouse School of Economics, Thibault.Laurent@univ-tlse1.fr

9 décembre 2013
Mises à jour et compléments :
math.univ-toulouse.fr/~sdejean

Institut de Mathématiques de Toulouse UMR 5219 — Université de Toulouse et CNRS
Plateforme de Biostatistique — Génopôle Toulouse Midi-Pyrénées

Sommaire

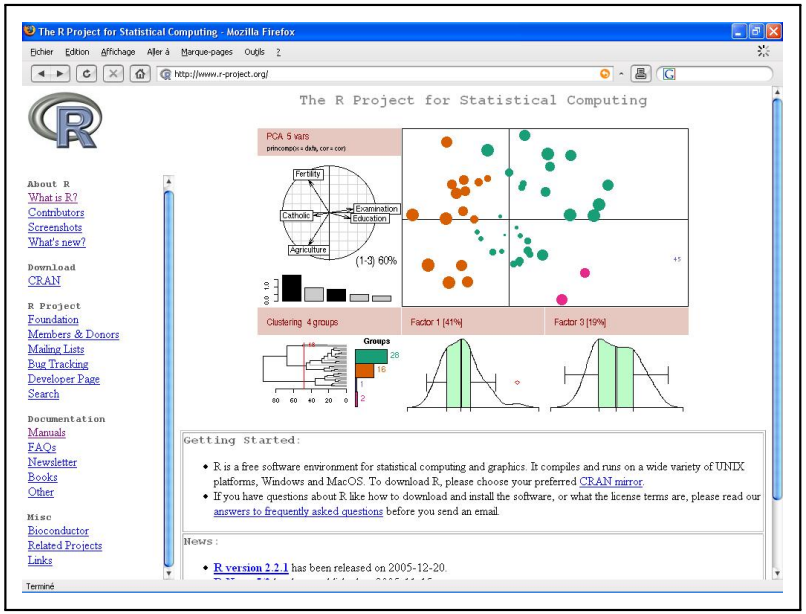
R ? ? ? ?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques : linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage (`www.r-project.org`*) for further information.*

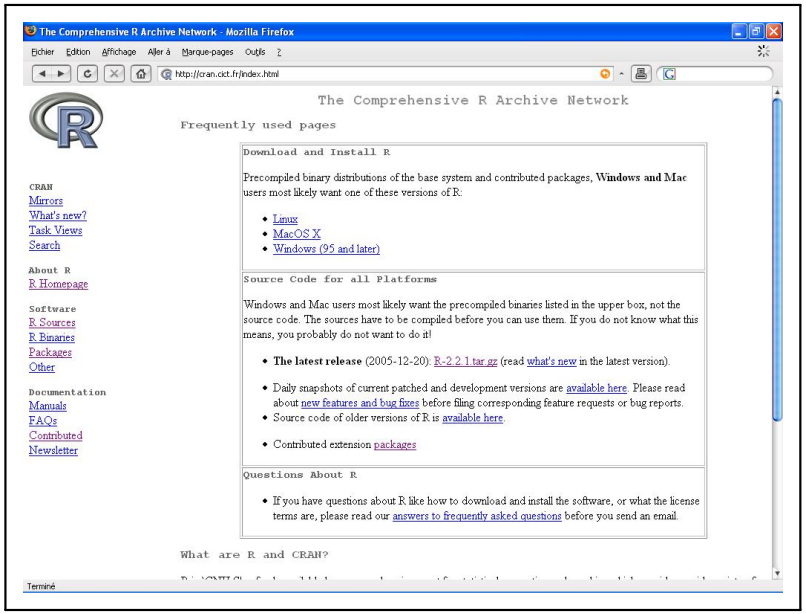
CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.

Ressources

Fichiers d'installation, mises à jour, packages, FAQ, newsletter, documentation...



www.r-project.org



cran.r-project.org

Ligne de commande

- `R >` Prompt en attente de commande
- `+` indique une commande non terminée
- Pas de " clic-bouton " (on évite l'impression de facilité et donc des bêtises)
- Appel à une fonction avec ses paramètres entre parenthèses
- L'absence de parenthèses provoque l'affichage du code de la fonction
- le caractère ; sépare deux commandes qui seront exécutées séquentiellement
- Le caractère # permet d'insérer un commentaire

Aide en ligne

Rubriques :

- Description
- Usage
- Arguments
- Details
- Value
- Note
- Authors
- Reference(s)
- See also
- Examples

```
R > help(plot)
R > ?plot
R > help.search("plot")
R > ??plot
R > help(help.search)
R > help(help)
```

Espace de travail

- les objets manipulables dans R (variables créées, jeux de données importés, résultats de fonction...) peuvent être stockés dans un **espace de travail** (*workspace*) qui apparaîtra dans le **répertoire de travail** (*working directory*) comme un fichier `.RData`.
- on peut charger un **espace de travail** avec `load()`, lister son contenu avec `ls()` et le sauvegarder avec `save()` ou `save.image()`.
- on peut connaître le **répertoire de travail** grâce à la commande `getwd()` et le modifier avec `setwd()`.
- le contenu du **répertoire de travail** (c'est à dire la liste des fichiers telle qu'elle peut apparaître dans un explorateur ou gestionnaire de fichiers) est accessible avec la commande `dir()`.

Aides pratiques

- Un éditeur basique pour préparer son script et copier-coller dans la console R
- Editeur Tinn-R(1) : éditeur gratuit permettant notamment une coloration syntaxique et l'interaction avec la console R
- Rstudio(2) : environnement de développement intégré
- Package `Rcmdr`(3) : interface graphique avec menus déroulants et zones "script" et "sortie"

(1) Tinn Is Not Notepad, `www.sciviews.org/Tinn-R`

(2) RStudio, `www.rstudio.com`

(3) J. Fox (2005) - The R Commander : A Basic-Statistics Graphical User Interface to R, Journal of Statistical Software, 14(9) - (`socserv.mcmaster.ca/jfox/Misc/Rcmdr`)

Scalaire

- entier, réel, logique, chaîne de caractères
- affectation $<-$ ou $=$
- `ls()` liste les variables de l'environnement de travail
- `rm()` efface une ou plusieurs variables

```
R > 2+2
R > exp(10)
R > a = log(2)
R > b <- cos(10)
R > a+b
R > a
R > 2==3
R > b = 2<3
R > ls()
R > rm(a)
R > a
R > a="toto"
```

Vecteur

- tous les éléments sont de même nature (tout numérique ou tout caractère ou tout ...)
- construction de vecteurs
- séquence, répétition
- extraction [ ]
- nommer les éléments d'un vecteur
- erreurs ou avis (*warning*) volontaires dans certaines commandes

```
R > d = c(2,3,5,8,4,6);d
R > is.vector(d)
R > 1:10
R > seq(from=1,to=20,by=2)
R > rep("bleu",times=10)
R > d[2];d[2:3];d[-3]
R > f = c(a=12,b=26,c=32,d=41);f
R > names(f);f["a"]
R > d[2] <- "toto";d;d+1
R > f>30;f[f>30];which(f>30)
R > d[2]="texte";d
R > f[2] = 22;f+100;f+d
R > cos(f);length(f);sort(d)
```

Matrice

- tous les éléments sont de même nature
- Construction, extraction de parties
- Fonction `apply()` !
- Produit matriciel %*%

```
R > A = matrix(1:15,ncol=5);A
R > B = matrix(1:15,nc=5,byrow=T)
R > cbind(A,B);rbind(A,B)
R > A[1,3];A[,2];A[1:3,2:4]
R > g = seq(0,1,length=20)
R > C = matrix(g,nrow=4)
R > C[C[,1]>0.1,]
R > A+B;A*B
R > cos(A); cos(A[1:2,1:2])
R > apply(A,2,sum)
R > apply(D,1,max)
```

```
R > C[C[,1]>0.1,]
```

`C[,1]` : 1ère colonne de `C`
`C[,1]>0.1` : vecteur logique de longueur le nombre de lignes de `C` contenant `TRUE` si la valeur est supérieure à 0.1 et `FALSE` sinon.
`C[C[,1]>0.1,]` extrait de la matrice `C` les lignes où les éléments sur la première colonne sont supérieurs à 0.1 et toutes les colonnes.

```
R > C
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	0.000	0.211	0.421	0.632	0.842
[2,]	0.053	0.263	0.474	0.684	0.895
[3,]	0.105	0.316	0.526	0.737	0.947
[4,]	0.158	0.368	0.579	0.789	1.000

```
R > C[,1]>0.1
```

```
[1] FALSE FALSE  TRUE  TRUE
```

```
R > C[C[,1]>0.1,]
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	0.105	0.316	0.526	0.737	0.947
[2,]	0.158	0.368	0.579	0.789	1.000

```
R > apply(A,2,sum)
```

Arguments de la fonction `apply()` :

`A` : matrice de travail

`2` : on s'intéresse aux colonnes (1 pour les lignes)

`sum` : fonction à appliquer sur les colonnes de la matrice de travail

`apply(A,2,sum)` : calcule la somme en colonne de la matrice `A`

```
R > C
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1	4	7	10	13
[2,]	2	5	8	11	14
[3,]	3	6	9	12	15

```
R > apply(A,2,sum)
```

[1] 6 15 24 33 42

Liste - *list*

- Objet "fourre-tout" : scalaire, vecteur, chaînes de caractères, listes...
- Accès aux composants d'un objet de type `list` soit par le nom (ou un raccourci non ambigû) soit par le numéro entre double-crochets `[[ ]]`
- La longueur d'un objet de type `list` est le nombre de ces composants
- Utile pour renvoyer les résultats d'une fonction

```
R > x = list("bidon",1:8);x
R > x[[1]];x[[1]]+1;x[[2]]+10
R > y = list(matrice=A,vecteur=f,
+ texte="bidon",scalaire=8)
R > names(y);y[[1]]
R > y$matrice;y$vec
R > y[c("texte","scal")]
R > y[c("texte","scalaire")]
R > length(y)
R > length(y$vecteur)
R > cos(y$scal)+y[[2]][1]
```

Data frame

- Structure spéciale pour les jeux de données de type *Individus* $\times$ *Variables*
- Analogies avec les matrices et les listes pour l'accès aux colonnes (composants)
- Les colonnes peuvent être de natures différentes (variables quantitatives et qualitatives)

```
R > taille = runif(12,150,180)
R > masse = runif(12,50,90)
R > sexe = rep(c("M","F","F","M"),3)
R > H = data.frame(taille,masse,sexe)
R > H;summary(H)
R > H[1,];H$taille
R > H$sexe
```

Importation de données

- Créer les 2 fichiers ci-dessous dans le répertoire de travail

`fic1.csv`

```
5,2.5,3.8
8,3.2,3.4
12,4.6,5
```

`fic2.txt`

```
X1 ;X2 ;X3
5 ;2,5 ;3,8
8 ;3,2 ;3,4
12 ;4,6 ;5
```

```
R > dir()
R > fic1 <- read.table("fic1.csv",
+ sep=",")
R > fic1b <- read.csv("fic1.csv")
R > fic2 <- read.table("fic2.txt",
+ sep=";",dec=",",header=TRUE)
R > help(read.table)
```

Exportation d'objets R

- `write.table()`, fonction réciproque de `read.table()`
- `sink()` : redirection du résultat des commandes vers un fichier (pas d'affichage à l'écran).
  ☞ Ne pas oublier de fermer le fichier en rappelant `sink()` sans argument.

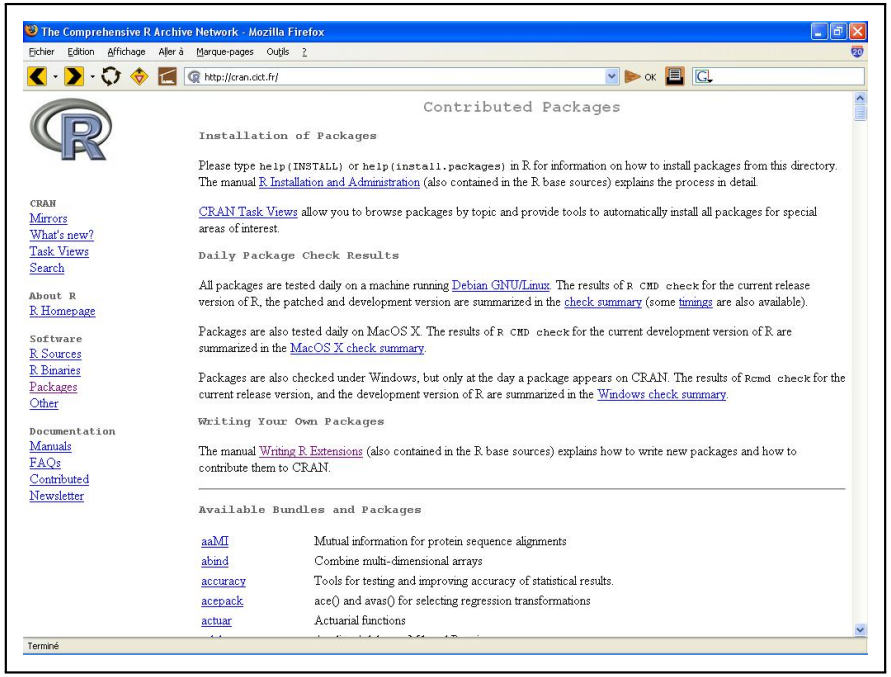
```
R > A=seq(1,10,l=50)
R > write.table(A,"A.txt")
R > sink("A2.txt")
R > A;summary(A)
R > sink()
R > summary(A)
```

Liens avec d'autres logiciels

La passerelle liant R à un autre logiciel scientifique (ou tableur) est le format texte (ASCII). R peut importer et exporter du format texte. Et c'est également le cas de la plupart des logiciels permettant de traiter des données.
Le package `foreign` permet de simplifier la communication avec les logiciels statistiques Minitab, S, SAS, SPSS, Stata, Systat, Octave.

Où trouver des extensions (*packages*) ?

Rubrique *Packages* sur cran.r-project.org ou un site mirroir plus proche.

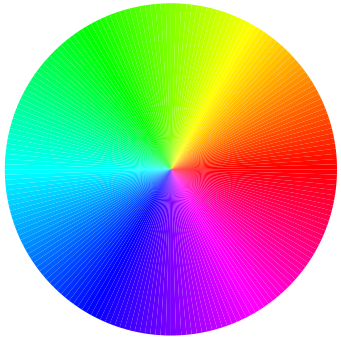
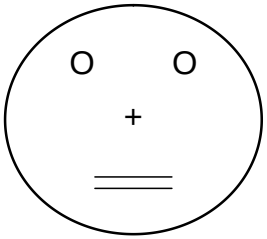
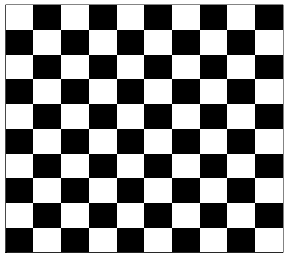
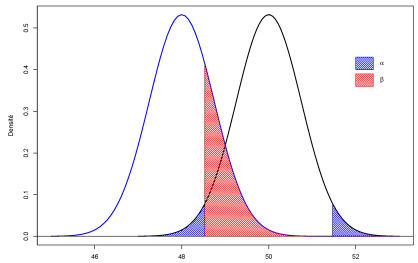


Consulter la rubrique *Task Views* pour un regroupement thématique de packages (Finance, Genetics, Medical Imaging...)

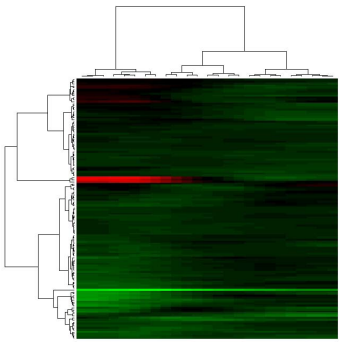
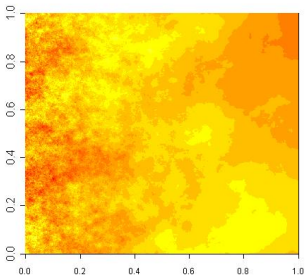
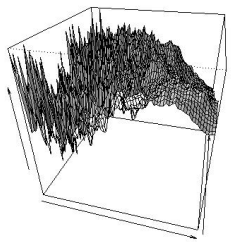
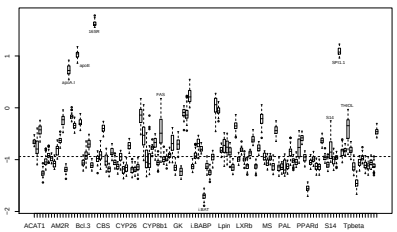
Utiliser un package

- Installation (en ligne) : Menu `Packages` (sous Windows), choix d'un site miroir puis choix du package
- Installation (en local) : Menu `Packages` (sous Windows), à partir d'un fichier Zip
- Pour gérer les packages en ligne de commande, utiliser l'ensemble des fonctions `install.packages()`, `update.packages()...`
- Chargement soit par menu soit par la fonction `library()`

Sommaire

```
R > help.search("plot")
```

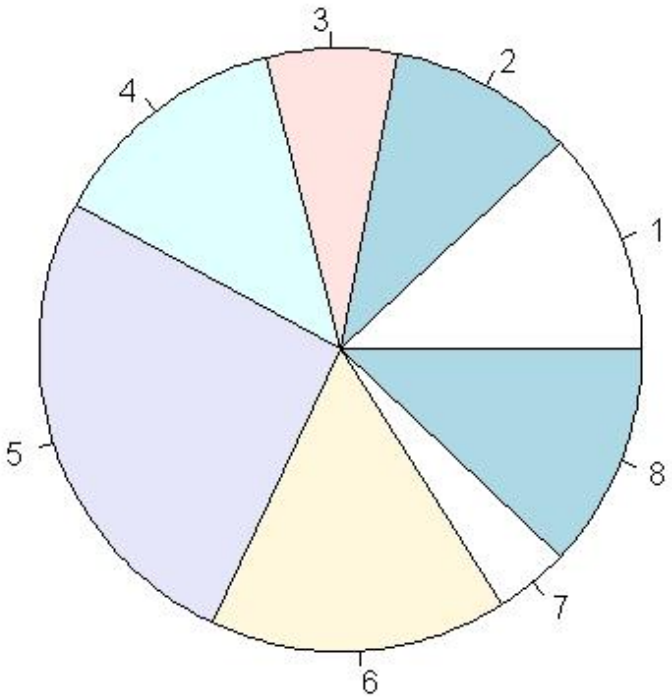





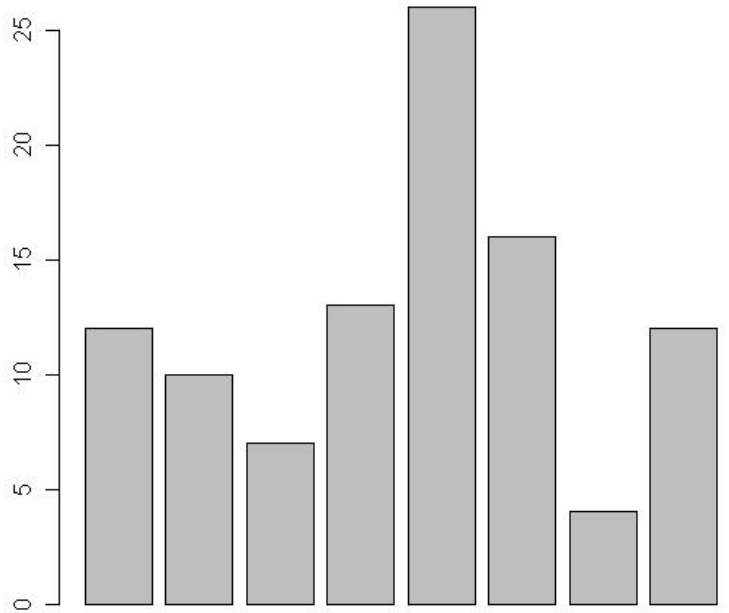
Une variable qualitative (Effectif)

Ex :

A	B	C	D	E	F	G	H
12	10	7	13	26	16	4	12

```
R > data=c(12,10,7,13,26,16,4,12)
R > pie(data);?pie
R > barplot(data);?barplot
```

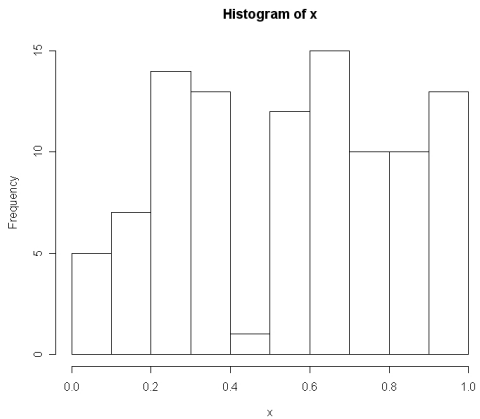


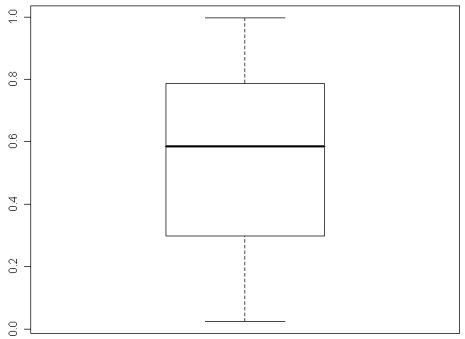


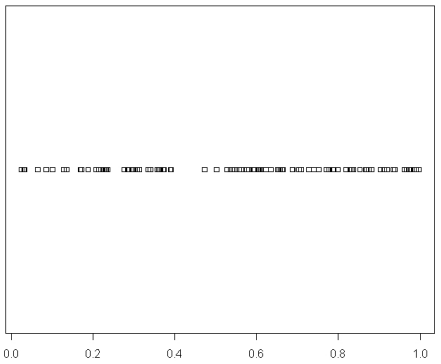
Une variable quantitative

Ex : Tirage aléatoire d'un échantillon de taille 100 issu d'une loi uniforme sur l'intervalle [0,1]

```
R > x=runif(100)
R > hist(x);?hist
R > boxplot(x);?boxplot
R > stripchart(x);?stripchart
```



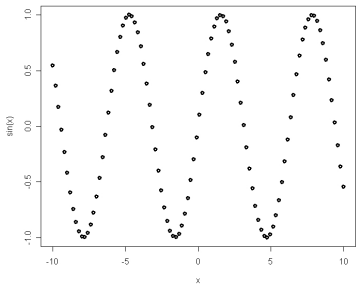
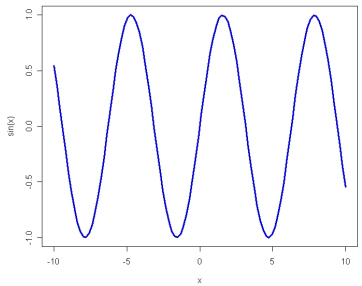
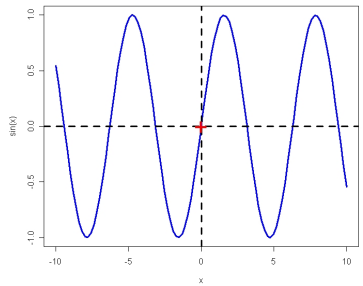
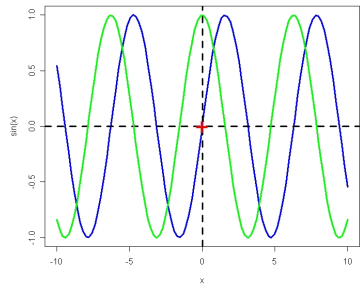




Deux variables quantitatives

Ex : Représentation de la fonction sinus sur l'intervalle [-10,10]

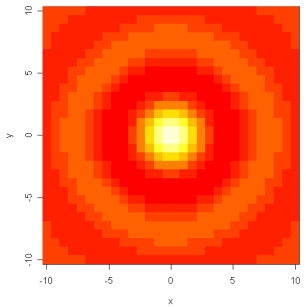
```
R > x=seq(-10,10,l=100)
R > plot(x,sin(x))
R > plot(x,sin(x),type="l",col="blue")
R > abline(h=0,v=0,lty=2)
R > points(0,0,pch="+",cex=3,col="red")
R > lines(x,cos(x),col="green")
```

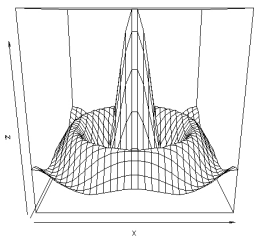





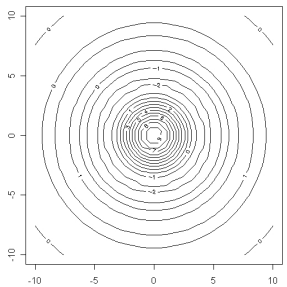
Trois variables

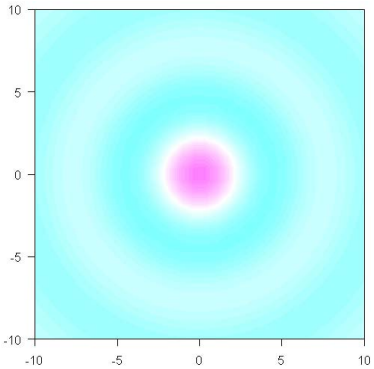
Ex : Représentation de la fonction sinus cardinal sur $[-10,10]^2$

```
R > ?persp # Construction de x, y, et z
R > image(x,y,z)
R > persp(x,y,z)
R > contour(x,y,z)
R > filled.contour(x,y,z)
R > ?image;?contour;?filled.contour
```









- Créer un graphique :

```
plot(),image()...
```

- Ajouter à un graphique existant :

```
lines(), abline(), points(), text(), rect()...
```

- Récupérer les coordonnées d'un point en cliquant :

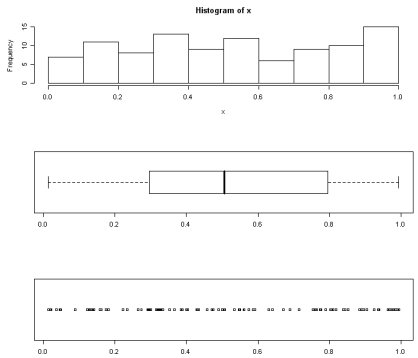
```
locator(1), text(locator(1),"ici")
```

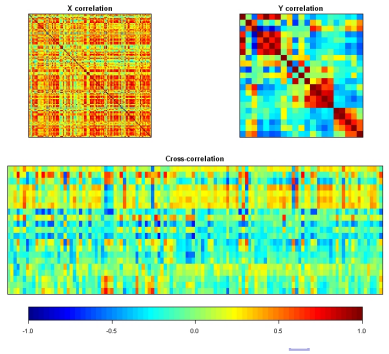
- Ouvrir une nouvelle fenètre graphique :

```
windows(), X11()
```

- Découper une fenètre graphique :

```
par(mfrow=c(lig,col)), layout()
```





- copier-coller : menu `Fichier > Copier vers le presse-papier` puis coller dans le logiciel de son choix ✶
- sauvegarder : menu `Fichier`, rubrique `Sauver sous`. Formats : emf, ps, pdf, png, bmp, jpeg ... ✶
- utiliser les fonctions associées à la sauvegarde de fichiers graphiques : `bmp(), jpeg(), pdf() ...`

1. Redirection de la sortie graphique vers un fichier

```
R > jpeg("fichier.jpg")
```

2. Tracé du graphique qui n'apparait pas à l'écran

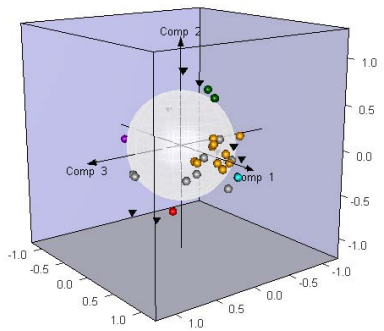
```
R > plot(1:100);text(20,80,"abcdef")
```

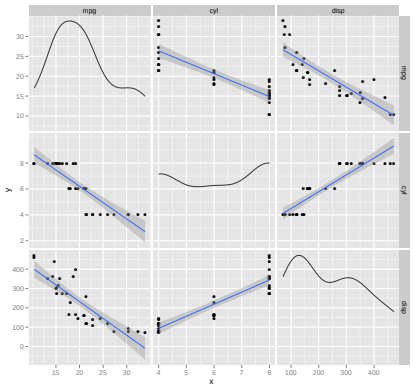
3. Fermer le fichier. ☞ Ne pas oublier cette étape !

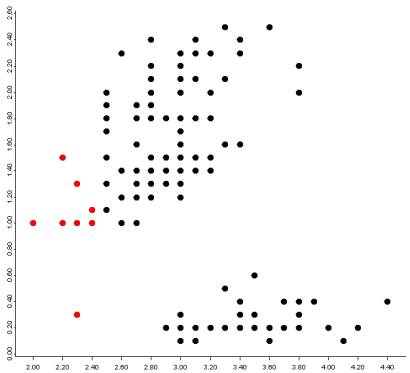
```
R > dev.off()
```

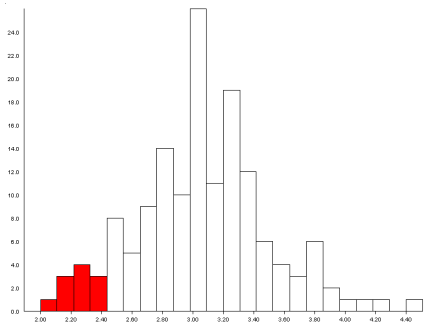
✶ Windows uniquement

- 3D (`rgl`), interactivité (`iplots`), facilité de création de graphiques complexes (`ggplot2`), représentation de réseaux (`igraph`) ...



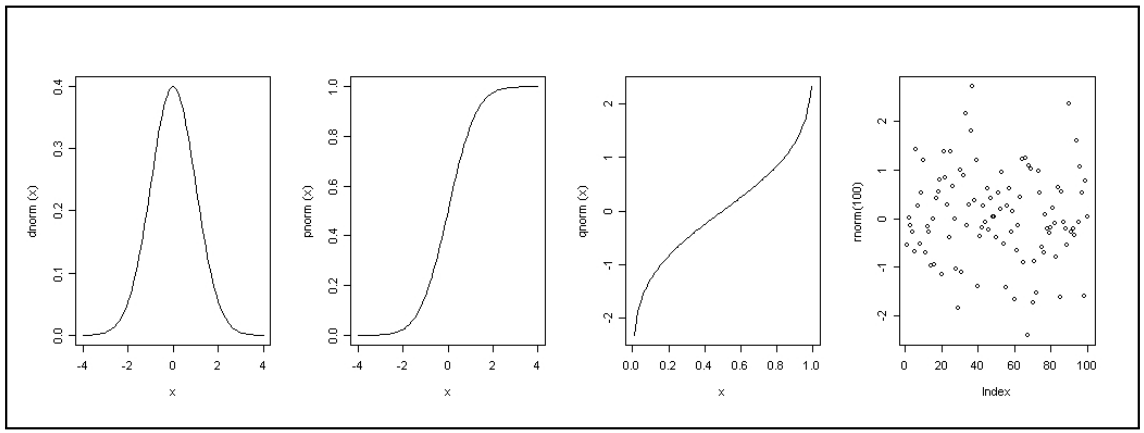






Sommaire

- Les distributions courantes sont programmées : Beta, Binomiale, Cauchy, Normale, Uniforme, Weibull...
- Plusieurs fonctions pour chaque distribution. Par exemple, pour la loi normale :
  - `dnorm()` : fonction densité (*density*)
  - `pnorm()` : fonction de répartition (*probability*)
  - `qnorm()` : fonction quantile (*quantile*)
  - `rnorm()` : générateur aléatoire (*random*)




```
R > help.search("Distribution")
```

- La plupart des tests statistiques courants (et bien d'autres) sont programmés dans R.
- Test de Student pour la comparaison de moyennes.
- Test de Fisher pour la comparaison de variances.
- Test de nullité du coefficient de corrélation.
- Test de Kolmogorov-Smirnov
- ...

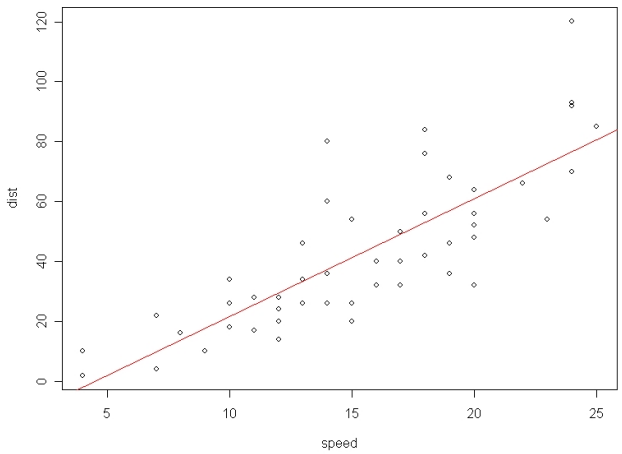
```
R > x=rnorm(100)
R > y=rnorm(100,mean=1)
R > t.test(x,y)
R > var.test(x,y)
R > t.test(x,y,var.equal=T)
R > cor.test(x,y)
R > ks.test(x,y)
R > ks.test(x,"pnorm")
R > ks.test(y,"pnorm")
R > ks.test(y,"pnorm",1)
```

```
R > help.search("test",package="stats")
```

- Les fonctions `boxplot()` et `hist()` peuvent ne pas produire de graphique (option `plot=FALSE`).
- La fonction `stem()` produit une diagramme *stem-and-leaf* (tige et feuille) qui donne un aperçu de la répartition des données de façon plus « rustique » qu'un histogramme
- La fonction `summary()` est une fonction générique (comme `plot()` par exemple) qui s'adapte à la classe (fonction `class()`) de l'objet passé en paramètre (vecteur, matrice, *data frame*, résultat d'une fonction...) ✶

```
R > x=runif(100)
R > y=runif(100)
R > mean(x);var(x);sd(x)
R > min(x);max(x)
R > quantile(x);median(x)
R > quantile(x,0.9)
R > boxplot(x,plot=FALSE)
R > cov(x,y)
R > summary(x)
R > stem(x);stem(y)
R > hist(x,plot=F)
```

- Liste les jeux de données disponibles dans le package *datasets* attaché par défaut au lancement de R. ✶
- Le résultat de la fonction `lm()` est un objet de classe `"lm"`, ce dont tient compte la fonction `summary()`. ✶



✶

```
R > search()
R > ls(pos=7) ✶
R > help(cars)
R > res1 = lm(dist ~ speed,
+ data=cars);res1
R > class(res1) ✶
R > plot(cars) ✶
R > abline(res1) ✶
R > names(res1)
R > summary(res1) ✶
R > anova(res1)
```

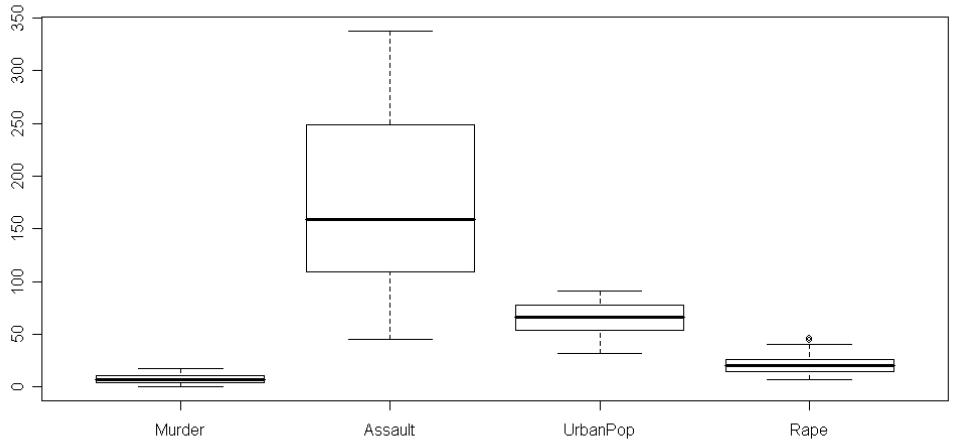
Jeu de données : USArrests

Criminalité aux Etats-Unis : 50 états, 4 variables

```
R > summary(USArrests)
```

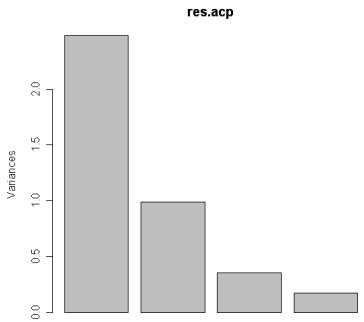
```
     Murder          Assault         UrbanPop          Rape
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

```
R > boxplot(USArrests)
```

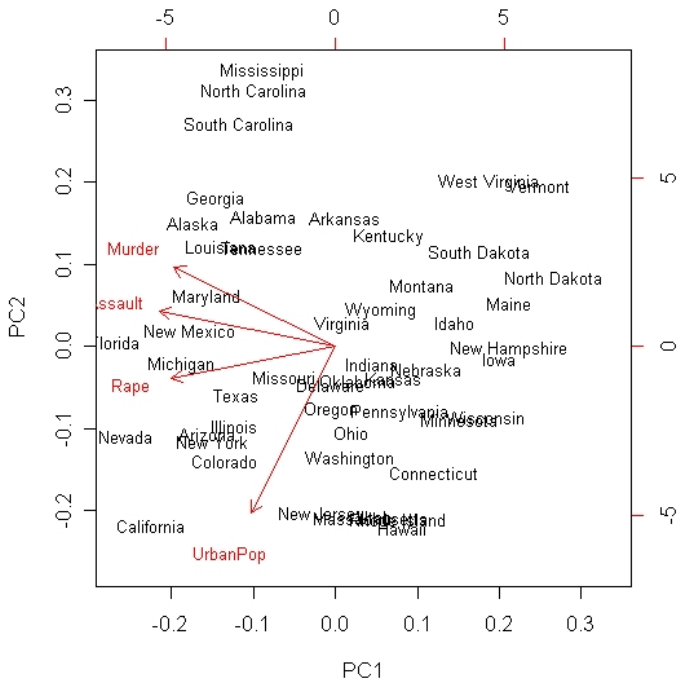


Analyse en composantes principales (ACP)

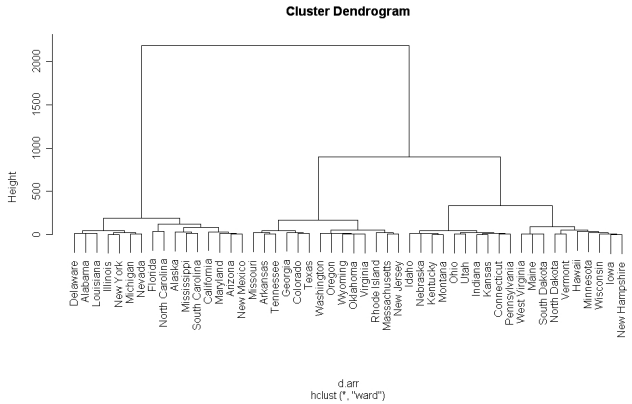
```
R > res.acp=prcomp(USArrests,scale=T)
R > plot(res.acp)
R > summary(res.acp)
R > biplot(res.acp)
```

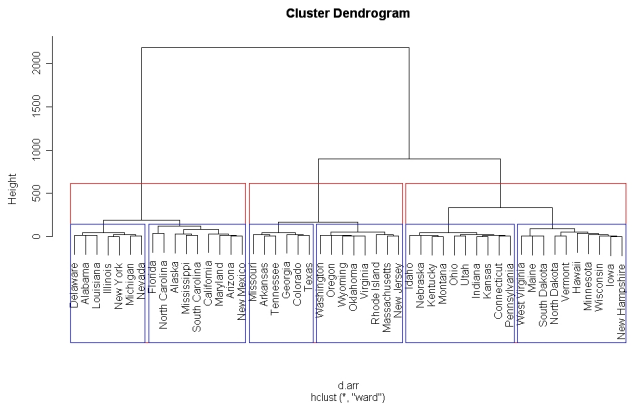


	PC1	PC2	PC3	PC4
Standard deviation	1.57	0.99	0.60	0.42
Proportion of Variance	0.62	0.25	0.09	0.04
Cumulative Proportion	0.62	0.87	0.96	1.00



Classification hiérarchique





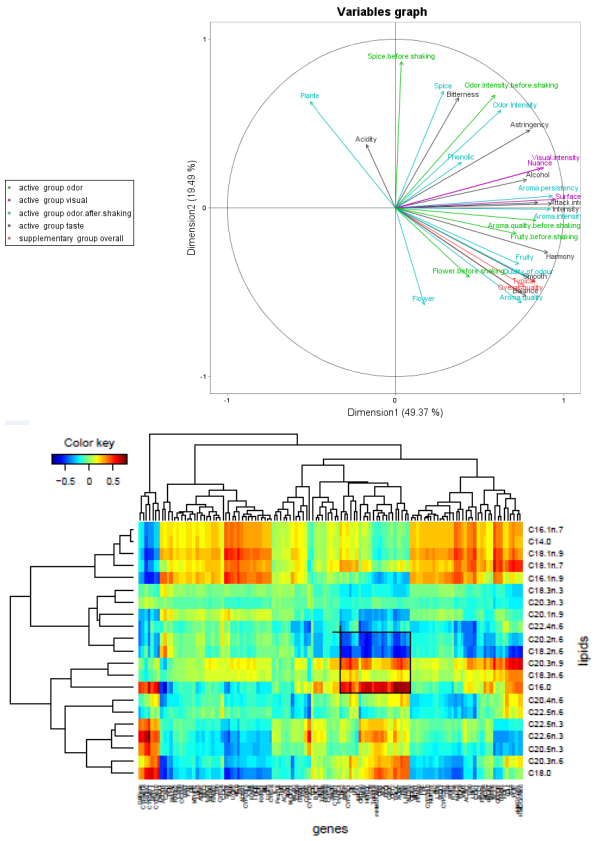

```
R > d=dist(USArrests)
R > clas=hclust(d,meth="ward")
R > plot(clas)
```

```
R > rect.hclust(clas,k=3)
R > rect.hclust(clas,k=6,bord="blue")
```

```
R > plot(hclust(dist(USArrests),method="ward"))
```

Quelques packages

- `FactoMineR` : analyse statistique exploratoire factominer.free.fr/
- `cluster` : classification cran.r-project.org/web/packages/cluster
- `mixOmics` : intégration de données www.math.univ-toulouse.fr/biostat/mixOmics
- ...



Sommaire

Répétition

Formes classiques de la répétition :

- Nombre de répétitions pré-défini : `for`
- Répétition jusqu'à obtention d'un critère : `while`
- `repeat`, `break`, `next`
- `R > help("for")` renvoie une aide en ligne commune pour les structures de contrôle (répétition et condition)

```
R > for (i in 1:10) print(i)
R > som=0
R > for (j in -5:5){
+ som=som+j
+ print(som)}
R > for (i in c(2,4,5,8)) print(i)
R > i=0
R > while (i<10){
+ print(i)
+ i=i+1}
```

Condition

- Structure classique : `if ... else`
- Structure particulière `ifelse(test, oui, non)`. Renvoie un objet de la même forme que `test`.

```
R > y=z=0;
R > for (i in 1:10) {
+ x=runif(1)
+ if (x>0.5) y=y+1
+ else z=z+1 }
R > y;z
R > x = rnorm(10) ✶
R > y = ifelse(x>0, 1, -1) ✶
```

Ex (✶) :

x	=	0.6	-0.4	-1.8	-0.5	-0.7	-0.2	0.3	-1.3	0.1	-0.4
y	=	1	-1	-1	-1	-1	-1	1	-1	1	-1

- Création de fonctions : `function(arg1,...){corps}`
- Affectation de valeurs par défaut à des arguments ✶
- Utilité du type `list` pour renvoyer plusieurs informations de natures différentes ✶
- Reconnaissance du paramètre si raccourci non ambigû ✶

```
R > f1=function(x){x+2}
R > f1(3)
R > x = f1(4)
R > f2 = function(a,b=0){a+b}
R > f2(a=2,b=3)
R > f2(5)
R > calc.rayon=function(r){
+ p=2*pi*r;s=pi*r*r;
+ list(ray=r,perim=p,surf=s)
R > resultat=calc.rayon(3)
R > resultat$r
```

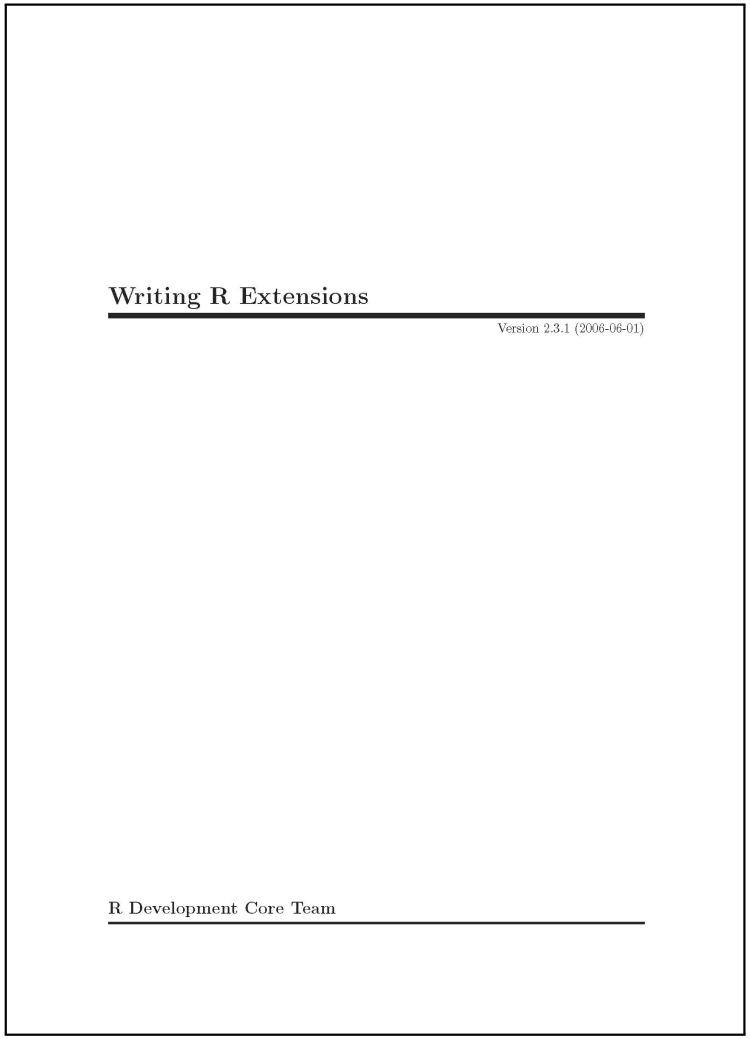
Le mode non-interactif

Le mode non-interactif (*BATCH*) peut-être utile pour lancer des calculs qui demanderont a priori beaucoup de temps. Le principe consiste en la lecture d'un fichier d'instructions et au renvoi des résultats dans un fichier de sortie sans interaction avec l'utilisateur.

```
R > help(BATCH)
```

- créer le fichier `fic.in` contenant les instructions 2+2 et 3+3
- dans un terminal de commande (DOS ou Linux), saisir la commande `R CMD BATCH fic.in fic.out`
- Vérifier la création d'un fichier `fic.out` qui contient les résultats attendus

Créer un package


Writing R Extensions

Version 2.3.1 (2006-06-01)

R Development Core Team

- Documentation : *Writing R extensions*
- Structure d'un package :

```
R > help(package.skeleton)
```

☞ Création des fichiers et des répertoires (R, man, data, src...) requis pour la construction du package à partir des éléments `R` (fonctions, données) passés en paramètre.

- Vérification : `R CMD check`
- Construction : `R CMD build`
- Soumission à CRAN (ftp, mail)

Interface avec C et FORTRAN

Référence : *Writing R extensions* (très technique !)

1. Fichier convolve.c

```
void convolve(double *a, int *na, double *b, int *nb, double *ab){
  int i, j, nab = *na + *nb - 1;
  for(i = 0; i < nab; i++) ab[i] = 0.0;
  for(i = 0; i < *na; i++) for(j = 0; j < *nb; j++) ab[i + j] += a[i] * b[j];}
```

2. Création d'une librairie dynamique (Unix, `.o` et `.so`) : `R CMD SHLIB convolve.c`

3. Création d'une fonction R qui fait appel à la librairie (pas obligatoire, mais plus clair)

```
conv = function(a, b)
  .C("convolve", as.double(a), as.integer(length(a)), as.double(b),
     as.integer(length(b)), ab = double(length(a) + length(b) - 1))$ab
```

4. Chargement de la librairie dynamique dans R :

```
R > dyn.load("convolve.so")
```

5. Utilisation :

```
R > res = conv(1:10,seq(0,1,l=10))
```

Introduction au logiciel R

Sébastien Déjean

math.univ-toulouse.fr/~sdejean

Institut de Mathématiques de Toulouse UMR 5219
Université Paul-Sabatier (Toulouse III)