Le langage R

Gérard Govaert

5 mars 2013

1 Introduction

R est un environnement de programmation statistique interfaçable avec C et Fortran. Il s'agit d'un langage interprété et orienté objet semblable au langage statistique S (ou S+). Son exécution et sa sémantique sont proches du langage Scheme, variante du Lisp. Il permet la lecture, la manipulation et le stockage de données. Il intègre de nombreuses méthodes statistiques et des outils graphiques variés avec sortie sur écran ou sur fichier. La gestion des fonctions se fait à l'aide de la notion de modules (*packages*). Le logiciel R est un logiciel libre disponible sous Windows, Unix, Linux et Macintosh. Il est diffusé sous licence GNU [1].

Le langage S a été développé dans les années 1970 par John Chambers & co aux Bell labs. R a été initialement écrit par Robert Gentleman et Ross Ihaka du département de statistique de l'université d'Auckland pour illustrer l'enseignement des statistiques. Le binaire a été mis à disposition en 1993 dans la bibliothèque Statlib et le source est disponible depuis 1995. Le groupe de développement a été élargi en 1997. Depuis, la participation active de nombreux chercheurs du domaine a permis une croissance exponentielle du logiciel R.

La meilleure source d'information est le site Internet de l'équipe de développement de R (*R core-development Team* : `www.r-project.org/`. Le logiciel est disponible depuis les sites du CRAN (*Comprehensive R Archive Network*).

2 Utilisation de R

Installation

À partir du site R, le chargement peut se faire sans difficulté : choisir `Download : CRAN`, un site et le type de matériel (par exemple, Windows) ; ensuite sélectionner `base` et `R-2.15.3-win.exe`. Il suffit ensuite de lancer ce programme qui installe le logiciel R. Pour faciliter l'utilisation de `Sweave`, outil permettant d'insérer des commandes R dans un fichier LaTeX, il est conseillé de l'installer directement sous `c:`

Démarrer et Terminer

Il suffit de cliquer sur l'icône R, créé lors de l'installation, pour lancer l'interface `Rgui.exe`. Il suffit alors de taper les commandes R dans la fenêtre `Console`. La commande `q()` permet de terminer.

Paramétrage

Différents fichiers interviennent au moment du lancement et de l'arrêt de R. Il y a les fichiers `Renviron.site` et `.Renviron` qui permettent de définir les variables d'environnement de R, les fichiers `Rprofile.site` et `.Rprofile` qui sont des scripts R, le fichier `Rconsole` qui permet de configurer la console, le fichier `Rdata` qui sauvegarde l'environnement de travail et enfin le fichier `.Rhistory` qui conserve un historique des commandes utilisés dans la session. Pour plus de précisions, il est possible d'effectuer la commande `help("Startup")`. Une solution possible est de placer les fichiers `.Rprofile` et `Rconsole` dans le répertoire `HOME` de Windows. Voici un exemple de fichier `.Rprofile` :

```
setwd("C:/Gerard/Logiciels/R/Work")
cat("\n   Bienvenue dans R!\n\n")
cat("\n   Vous êtes dans le répertoire : ",
     getwd(),"\n\n")
```

Remarquons qu'avant le lancement de ce fichier, le *working directory*, dont la valeur peut être obtenue à l'aide de la commande `getwd()`, est défini dans l'icône Rgui de lancement de R. Ce répertoire peut être modifié (clic droit sur l'icône, propriétés, modifier le fichier correspondant à « Démarrer dans »).

Les commandes `setwf`, `getwf`, `?Rconsole`, `?Renviron` et `Sys.getenv("R_USER")` peuvent être utiles pour comprendre la procédure de lancement de R.

Aide en ligne

Différentes aides sont disponibles. Elles peuvent être obtenues à l'aide du menu `Aide` ou à l'aide de commandes R. Une aide générale est obtenue avec la commande `help.start()` ou à l'aide du menu `Aide -> Aide Html`. La description d'une commande est obtenue avec la commande `?commande` (ou `\help(commande)`) ou à l'aide du menu `Functions R`. On dispose aussi d'une recherche par mots-clés grâce à la commande `apropos(mot-clé)` ou à l'aide du menu `Rechercher dans l'aide`.

Entrée des commandes

Les commandes peuvent être entrées de différentes façons :

1. Directement dans la fenêtre `Rconsole`.
2. À l'aide de l'historique des commandes (utilisation des flèches et édition des commandes à la Emacs).
3. En les éditant dans un fichier script, dont le nom est souvent suffixé par .R. Ce script peut être lancé à l'aide de la commande `source` où à l'aide du menu `Fichier`. On peut éditer ce fichier à l'aide de l'éditeur R, accessible par la commande `edit` ou à l'aide du menu, ou de tout autre éditeur de texte.

Commandes système

Différentes commandes permettant d'accéder à Windows sont disponibles. Par exemple, les fonctions `getwd` et `setwd` permettent respectivement de connaître et de modifier le répertoire courant, la fonction `dir` permet de lister les fichiers contenus dans le répertoire courant, la fonction `sink` permet de rediriger les sorties,...

1. Projet créé en 1984 pour développer un système d'exploitation complet et libre

3 Éléments de base du langage

Généralités

- *Commentaires :* texte situé après le caractère « # »
- Noms : constitués de lettres, chiffres et du caractère « . » et ne commençant pas par un chiffre
- *Constantes :* 5, 5.23, `pi`, `Inf`, `NaN`, `TRUE` ou `T`, `FALSE` ou `T`, `NA` (not available : valeur manquante), `NULL`, "exemple", `LETTERS`, `letters`, `month.abb`, `month.name`
- *Séparateurs de commandes :* ; ou saut de ligne
- *Opérateurs d'affectation :* = ou <-
- *Opérateurs arithmétiques :* +, -, *, /, ^
- *Division entière et modulo :* %/%, %%
- *Prod. matr., ext. et de Kronecker :* %*%, %o%, %x%
- *Opérateurs logiques :* <, <=, >, >=, ==, !=, !, &, &&, |, ||, xor
- *Expression groupée :* {exp1 ; exp2 ;... ; expm}

Les objets

Les objets sont les éléments de base de R. Un objet est caractérisé par son nom, son type, son mode de stockage, son mode qui décrit le contenu, sa classe qui décrit la structure et des attributs. Les fonctions `typeof`, `storage.mode`, `mode`, `class`, `attributes`, `attr` permettent de manipuler ces informations. Le type et le mode de stockage sont souvent identiques.

Les types

Les valeurs possibles sont NULL, symbol, pairlist, closure, environment, promise, language, special, builtin, logical, integer, double, complex, character, expression, list, ...

Les modes

Numérique pour représenter les nombres (`numeric`, `is.numeric`, `as.numeric`, `is.finite`, `is.infinite`, `is.nan`)

Complexe pour représenter les nombres complexes (`complex`, `is.complex`, `as.complex`)

Logique pour représenter les valeurs logiques (`logical`, `is.logical`, `as.logical`)

Caractère pour représenter les chaînes de caractères (`character`, `is.character`, `as.character`, `is.na`)

Les principales classes

Vecteur (*vector*) : collection ordonnée d'éléments de même mode.
- Attributs intrinsèques : mode (logical, numeric, complex ou character) et length
- Attributs : names (optionnel)
- Indexation :
  - Forme : `[...]`
  - Index :
    - vecteur logique : sélection des valeurs correspondant à la valeur `TRUE`
    - vecteur de valeurs numériques positives : sélection
    - vecteur de valeurs numériques négatives : exclusion
    - vecteur de chaînes (si l'attribut `names` existe)
- Fonctions associées : `vector`, `is.vector`, `as.vector`, `mode`, `length`, `c`, `names`

Matrices (*matrix*) vecteur structuré en ligne et colonne.
- Attributs
  - `dim` : vecteur contenant le nombre de lignes et de colonnes
  - `byrow` : rangé par colonne (FALSE) ou par ligne (TRUE)
  - `dimnames`
- Indexation :
  - Comme pour les vecteurs, mais en utilisant 2 index.
  - Exemples :
    - `(m[2:6,c(5,7)])`
    - `(m[,c(5,7)])` pour sélectionner les colonnes 5 et 7
- Fonctions associées : `matrix`, `is.matrix`, `as.matrix`, `dim`, `nrow`, `ncol`

Tableaux (*array*) : vecteur structuré en plusieurs dimensions (généralisation de la classe `matrix`).
- Fonctions associées : `array`, `is.array`, `as.array`, `outer`

Facteur (*factor*) : Cette classe permet de traiter les variables qualitatives nominales ou ordinales. Elle est implémentée à l'aide d'un vecteur d'entiers et d'un vecteur de noms.
- Type de stockage : entier
- Mode : numérique
- Attributs :
  - `levels`
- Fonctions associées : `levels`, `cut`, `nlevels`, `factor`, `is.factor`, `ordered`, `is.ordered`, `tapply`, `contrasts`

Liste (*list*) collection ordonné d'objets de modes pouvant être différents.
- Attributs intrinsèques : `mode` (`list`) et `length`
- Attributs
  - `names`
- Indexation :
  - `[...]` pour atteindre une sous-liste
  - `[[...]]` ou `$` pour atteindre un composant
- Fonctions associées : `list`, `is.list`, `as.list`, `is.null`, `as.null`, `outer`, `c`

Structure de données (*data.frame*) cette classe permet de traiter les tableaux de données. Il s'agit d'une liste dont chaque composant correspond à une variable. Ces composants, qui doivent avoir la même longueur, peuvent être des vecteurs, des facteurs, des matrices, des listes ou des structures de données.
- Attributs
  - `names` : nom des colonnes
  - `row.names` : nom des lignes
- Indexation : `$`, `[`
- Fonctions associées : `data.frame`, `mode`, `length`, `dim`, `nrow`, `ncol`, `attach`, `detach`, `search`

4 Les fonctions

Fonctions génériques

Il s'agit de fonctions qui s'appliquent à tous types d'objets mais qui exécutent une commande spécifique en fonction de la classe de l'objet considéré. Par exemple `print(x)` lance `print.matrix(x)` si `x` est de classe matrix et si aucune fonction n'est trouvée, c'est la fonction `print.default` qui sera lancée. Les principales fonctions génériques sont les fonctions `print`, `plot` et `summary`. La fonction `methods` permet de connaître les différentes méthodes associées à une fonction générique.

Écriture de fonction

On peut créer une fonction grâce à la commande :

```
name <- function(arg_1,arg_2,...) expression
```

La valeur retournée par la fonction est précisée à l'aide de la fonction `return`. Si cette dernière est absente, la valeur retournée est la dernière valeur calculée. À l'appel de la fonction, la valeur des arguments peut être définie en utilisant l'ordre de ces arguments, le nom des arguments ou encore un mélange des deux. Il est possible de passer des arguments à une sous-fonction grâce à l'argument « ... ». La commande `args` permet de lister les arguments d'une fonction.

Structures de contrôle

Les structures de contrôle sont les suivantes :
- `if(cond) expr`,
- `if(cond) expr1 else expr2`,
- `for(var in seq) expr`
- `repeat expr`
- `while(cond) expr`
- `break`
- `next`

Les Modules (ou packages)

Les fonctions sont regroupées en modules. La fonction `library()` permet de connaître la liste des modules installés. La fonction `install.package` permet d'installer un nouveau module, à partir d'un fichier `zip` ou directement depuis le CRAN. La fonction `remove.packages` permet des-installer un module. Les fonctions `library(mod)` et `detach("package:mod")` permettent respectivement de charger un module installé et de décharger un module. La fonction `help(package=mod)` donne la liste des fonctions du module `mod`. Toutes ces fonctions sont accessibles dans le menu `Packages`. La fonction `library(help=nom_module)` permet d'obtenir une documentation sur un module. Le menu `\packages-->Charger le package` permet de connaître les modules pré-installés. En dehors des modules pré-installés, voici quelques modules qui peuvent être utiles :

xtable : permet d'exporter des tableaux au format LaTeX ou HTML

rggobi : permet d'échanger des données avec le logiciel `ggobi`

5 Les graphiques

De nombreuses de fonctions sont disponibles pour créer des graphiques. En plus de ces fonctions, dites de haut niveau, il existe une famille de fonctions, dites de bas niveau, permettant d'ajouter des éléments à un graphique existant et quelques fonctions graphiques interactives. Il existe en outre la fonction `par` qui permet de régler un nombre très important de paramètres à appliquer au graphique en construction.

6 Sweave

Il s'agit d'un outil permettant d'insérer des commandes R dans un fichier LaTeX. La commande R `Sweave("ex.rnw")` permet de transformer le fichier `ex.rnw` contenant le code Latex et les appels aux fonctions R en un fichier `ex.tex`, et éventuellement en fichiers graphiques. Il suffit alors de compiler le fichier `ex.tex`. Voici quelques exemples d'appel à R :

```
<<>>=
data(airquality)
kruskal.test(Ozone ~ Month, data = airquality)
@

<<fig=TRUE,echo=FALSE>>=
boxplot(Ozone ~ Month, data = airquality)
@
```

7 Principales fonctions

Divers

q() : arrêt
dir() : contenu du répertoire de travail
getwd() : nom du répertoire de travail
setwd("d") changement du répertoire de travail
source("s") : exécution d'un fichier script
methods(f), methods(class=c) : liste les méthodes associées à une fonction générique ou à une classe
system, system.file, system.time, as.date
Op. arith. : +, -, *, /, ^, %*%, %/%, %%, %o%
Op. log. : <, <=, >, >=, ==, !=, &, &&, |, ||, !+, %in%
Index. des vecteurs : x[n], x[-n], x[1 :n], x[-(1 :n)], x[c(3,5,9)], x["name"], x[x>2 & x <20]
Index. des listes : x[n], x[[n]], x[["name"]], x$name
Index. des matrices : x[i,j], x[i,],x[,j],x[,c(1,3)], x["name",]
Index. des data frames : matrices + x[["name"]] et x$name
attach, detach : ajout ou suppression de sd au chemin de recherche

Aide

help(sujet), ?sujet : documentation associée à une fonction
help.start() : aide html
help.search("str") : liste des aides contenant "str"
apropos("str") : liste des fonctions contenant "str"
index.search, example, demo

Création

c(1,2,3), 1 : 3, seq, rep : création de vecteur
list, array, matrix, factor
data.frame, expand.grid : création de sd
gl : création des modalités d'un facteur

Information et manipulation des objets

ls ou objects : objets de l'environnement
rm ou remove : suppression d'un objet
summary : résumé associé à un objet
str, ls.str : structure interne d'un objet
typeof : type d'un objet
storage.mode : mode de stockage d'un objet
mode : lecture ou modification du mode d'un objet
class, unclass : classe d'un objet
attributes, attr : attributs d'un objet
as.array, as.data.frame, as.numeric, as.logical, as.character
is.array, is.data.frame, is.numeric, is.logical, is.character
dim, dimnames, lenghh, nrow, ncol
rbind, cbind : concaténation en ligne ou en colonne
sort, order, rev, rank
cut, split, findInterval
subset, unique, sample : extraction
which : indices des éléments prenant la valeur TRUE
na.omit, na.fail, na.exclude, na.pass : données manquantes
match, pmatch, %in% : recherche
merge : fusion de 2 sd
stack, unstack : concaténation de vecteurs d'une sd ou d'une liste
paste, substr, strsplit : concaténation, substitution de carac.
chartr, tolower, toupper : conversion de caractères
grep, sub, gsub, regexpr : recherche de caractères
nchar : nombre de caractères

Entrées-sorties

library : chargement de modules (*packages*)

cat, print, format, write, write.table, sprintf : écriture ASCII

read.table, scan, read.csv, read.delim, read.fwf : lecture ASCII

save, save.image : écriture

load, data : lecture

module foreign : fonctions permettant de lire des données provenant des principaux logiciels statistiques

data.entry : édition d'un objet avec un tableur

file : création, ouverture et ouverture d'un fichier

sink("file") : envoi des sorties sur un fichier

readBin, writeBin : lecture et écriture au format binaire

list.files, dirname, basename, path.expand

file.choose, file.show, file.exists, file.path,...

Mathématiques

abs, round, sqrt, log, log10, exp, sin, cos, tan, acos, asin, atan

sum, cumsum, rowSums, colSums

%*%, prod, cumprod, crossprod

scale : centrage, réduction

min, cummin, which.min, pmin

max, cummax, which.max, pmax

diff calcul des différences entre les éléments d'un vecteur

t, aperm : transposition

diag : diagonale d'une matrice

union, intersect, setdiff, setequal, is.element : fonction d'ensembles

beta, gamma, choose, factorial

eigen, svd, qr, solve

Distribution de probabilités

d<loi>, p<loi>, q<loi>, r<loi> : densité, fdR, quantile et simulation. `<loi>` peut prendre les valeurs : binom, cauchy, chisq, exp, F, gamma, geom, hyper, lnorm, logis, nbinom, norm, pois, t, unif, weibull, wilcox.

Statistique exploratoire

summary : résumé numérique

range : étendue

mean, rowMeans, colMeans, weighted.mean, ave, median : moyenne et médiane

boxplot : diagramme en boîte

stem ou stem.leaf (aplpack) : diagramme en tige et feuille

hist : histogramme

var, sd, cor : variance, covariance, écart-type et corrrélation

quantile

fivenum : résumé d'une distribution

pairs : graphique matriciel

faces (aplpack) : visages de Chernoff

stars : polygones

barplot : diagrammes en bâton

pie : diagrammes en camenbert

table : table de contingence

prop.table : fréquences relatives (générale, ligne ou colonne)

mosaicplot :

tabulate

aggregate : Statistiques pour des sous-ensembles de données

princomp : ACP

biplot : Représentation simultanée des individus et des variables

- **biplot.princomp :** Représentation simultanée des individus et des variables sur les axes de l'ACP ; option scale=0 pour avoir la représentation standard.

kmeans : algorithme des k-means

Statistique décisionnelle

t.text, power.t.test, var.test : tests de student et de Fisher à 1 ou 2 populations

wilcox.test : test non-paramétrique de Wilcoxon à 1 ou 2 populations

binom.test, prop.test : tests sur la proportion

ks.test tests d'adéquation de Kolmogorov-Smirnov

chisq.test tests d'adéquation du χ^2 et test du χ^2 de contingence

mcnemar, cor.test, fisher.test, friedman.test : divers tests

shapiro.test : test de normalité

aov, anova : analyse de la variance

density : estimation non paramétriques

optim, nlm, lm, glm, approx, nls, approx, spline, loess : ajustement de modèles

predict, df.residual, coef, residuals, deviance, fitted, logLik, AIC : fonctions génériques associées

Graphiques

Gestion des fenêtres graphiques : window, postscript, pictex, dev.off, dev.list, dev.next, dev.prev, dev.set, dev.copy, dev.print, graphics.off

Partitionnement de la femêtre graphique : layout, layout.show

Fonctions de haut niveau : plot, hist, barplot, dotchart, pie, boxplot, stem, sunflowerplot, stripplot, coplot, interactionplot, matplot, fourplot, assocplot, mosaicplot, pairs, qqnorm, qqline, qqplot, contour, persp, image, stars, symbols

Paramêtres associés : add, axes, type, xlim, ylim, xlab, ylab, main, sub

Fonctions de bas niveau : box, points, lines, text, mtext, segments, arrowx, abline, rect, polygone, legend, title, axis, rug

Fonctions interactives : locator et identify

Paramêtres graphiques : les principaux paramètres graphiques, gérés par la fonction `par`, sont les suivants : adj, bg, bty, cex, col, font, las, lty, lwd, mar, mfcol, mfrow, pch, ps, pty, tck, tcl, xast, yast

Autres fonctions colors, palette, split.screen, screen, erase.screen, close.screen

Module aplpack stem.leaf, faces

Module lattice barchart, bwplot, densityplot, dotplot, histogram, qqmath, stripplot, qq, xyplot, levellot, contourplot, cloud, ...

Programmation

nom <- function(arguments) exp : création d'une fonction

return

if(cond) expr, if(cond) expr1 else expr2

for(x in seq) expr, while(cond) expr, repeat expr, switch

break

next

ifelse

do.call

apply, lapply, tapply, sapply, by

8 Exemples

Vecteurs

```
x1<-c(20,3,30,25)           # Création
x2<-rep(3,4)
l1<-c(T,F,T,F)
l2<-x1==x2
c1<-c('a','b','c','d')
c3<-rep(c('a','b'),3)
x3<-rep(c(1,2),c(2,5))
x4<-seq(1,50,by=5)
x5<-scan('ex1.txt')         # à partir d'un fichier
x6<-scan('')                # à partir de la console
str(x1)
class(x1)
attributes(x1)
ode(x1)
mode(x2)
mode(x3)

x1+x2                       # Utilisation
x1/3
xbar<-sum(x1)/length(x1)    # Moyenne et écart-type
xbar
mean(x1)
sqrt(sum((x1-xbar)^2)
  /(length(x1)-1))
sd(x1)
freq<-c(130,100,50,         # Vecteurs avec noms
  45,90,86)
freq
names(freq)<-c('GI','GM',
  'GSM','GTU','GC','GB')
freq
attributes(freq)
freq['GTU']
freq[4]
```

Matrices

```
y<-sample(1:100,30,rep=T)          # Création
y
y<-matrix(y,nrow=5,byrow=T)
nomi=paste("A",1:5,sep="")
nomj=paste("B",1:6,sep="")
dimnames(y)=list(nomi,nomj)
y
attributes(y)
dimnames(y)[[1]][2]
y1<-y[,3:5]                        # Utilisation
z1<-z[c(1,3,5),,]
u<-sort(y[1,])
o<-order(y[1,])
y<-y[,o]
y
```

Tableaux

```
z<-array(1:30,dim=c(5,2,3))       # Création
z
z[2,1,2]                          # Utilisation
z[2,,2]
z[2,,2]<-c(0,0)
z
```

Facteurs

```
y=factor(c(rep('v1',10),rep('v2',10))) # Création d'une var. qual.
z=gl(3,4,48,labels=c('a','b','c'))     # Création d'une var. qual.
x=rnorm(48)                            # Création d'une var. quan.

tapply(x,z,mean)                       # Description de la var. x
boxplot(x~z)                           # regroupée suivant les
stripchart(x~z)                        # modalités de la
stripchart(x~z,method='jitter')        # var. qual. z
```

Listes

```
v1=c(10,2,17)                     # Création d'une liste constituée
v2=c('yes','no','oui','non')      # de 3 éléments
v3=1:5
l<-list(a=v1,b=v2,c=v3)
l
mode(l)

l$y                               # Extraction et manipulation
l[[2]]                            # d'élément de la liste
l$y[3]
l[[1]][3]

l[1]                              # Sous-liste
l[c(1,3)]

y<-matrix(rnorm(12,0,1),nrow=3)   # Liste créée par une fonction
r<-svd(y)
r
mode(r)
attributes(r)
r$d
r$u
r$v
```

Data.frame

Création et sauvegarde

```
# Données simulées
x1=runif(30,0,1)
x2=runif(30,5,10)
x3=x1+rnorm(30,0,0.05)
x4=rbinom(30,3,0.2)
x=data.frame(v1=x1,v2=x2,v3=x3,v4=x4)
x$v4=factor(x$v4,labels=c("A","B","C"))
save(x,file="exemple1.Rdata")

# A partir du fichier ascii exemple2.txt suivant :
#     sexe taille poids yeux age cheveux
#     Pierre       M 1.80 85 bleu     23 chatain
#     Gabrielle    F 1.65 55 marron   18 brun
#     Paul         M 1.60 60 marron   34 chatain
#     Clara        F 1.50 45 bleu     17 blond
#     Gaspard      M 1.92 78 marron   28 brun
#     Jean         M 1.63 54 marron   40 brun
#     Michel       M 1.69 80 bleu     37 blond
#     Alain        M 1.73 74 bleu     42 brun
#     Louise       F 1.59 48 marron   33 chatain
#     Karine       F 1.63 47 marron   25 chatain
x=read.table('exemple2.txt')
save(x,file="exemple2.Rdata")

#  A partir du fichier Excel exemple3.xls suivant :
#
#         A    B         C       D      E
# 1        var 1    var 2   var 3   var 4
# 2     ind1    8       6       5       12,78
# 3     ind2    7       7       3       45,34
# 4     ind3    6       5       4       45,94
# 5     ind4    7       5       4       12,52
# 6     ind5    4       8       6       80,45
# 7     ind6    2      12       9       55,23
#
# Il est possible de recopier ce fichier Excel au format csv dans
# le fichier exemple3.csv.puis de le lire de la façon suivante :
data<-read.csv2("exemple3.csv",row.names=1)
save(notes,file="exemple3.Rdata")
```

Information, sélection et édition

```
load("exemple2.Rdata")
attributes(x)
class(x$poids)
class(x$taille)
class(x$cheveux)
names(x)
rownames(x)
length(x)
n=dim(x)[1]                       # Taille de l'échantillon
d=dim(x)[2]                       # Nombre de variables

load("exemple1.Rdata")
x$v2
x[[2]]
x[,2]
x[,'v2']
x.qan=x[sapply(x,is.numeric)]   # Sélection des var. quant.
x.qal=x[sapply(x,is.factor)]    # Sélection des var. qual.

y<-edit(x)
```

Fonction

```
norme<-function(v,p=2){              # Création d'une fonction
# Calcul de la norme Lp d'un vecteur
# v : vecteur
# p = 2 par défaut
#
(sum(v^p))^(1/p)
}
norme                                # Utilisation
norme()
x=c(1,2,3]
norme(x,3)
norme(x)
norme(v=x,p=3)
norme(p=3,v=x)
norme(x,p=3)

stat<- function(v){                  # Création d'une autre fonc.
m=mean(x)
e=sd(x)
min=min(x)
ax=max(x)
res=list(m=m,e=e,min=min,max=max)
}
r<-stat(x)                           # Utilisation
r
mode(r)
r$min
```

Graphiques

```
x<-seq(0,5,length=10)       # Création de deux vecteurs
y<-2*x

plot(x,y)                   # Affichage des points (x,y)

postscript('essai.ps')      # Même chose mais le graphe est
plot(x,y)                   # envoyé sur un fichier
dev.off()

old.par=par(no.readonly=T)  # Sauvegarde des par.graphiques
par(mfrow=c(2,2))
plot(x,y)
title("Sans option")
plot(x,y,type="l")
title("Avec l'option 'l'")
plot(x,y,type="b")
title("Avec l'option 'b'")
plot(x,y,type="n")
text(x,y,letters[1:10])
par(old.par)                # Restauration des par.graphiques

x<-seq(0,2*pi,length=100)   # Utilisation de la souris
plot(x,sin(x),type="l")     # pour placer un texte
abline(h=0)
text(locator(1),"Point M")

x=seq(-3,3,by=0.1)          # Autre exemple
par(mfrow=c(2,2))           # Découpage de la fenêtre graphique
plot(x,sin(x),type="l")
plot(x,tan(x),type="l",
 xlim=c(0,3),ylim=c(-10,10))
points(x,tan(x))
plot(x,dnorm(x),type="l")
plot(x,pnorm(x),type="l")
par(mfrow=c(1,1))           # Suppression du découpage

                            # Affichage d'un nuage de points avec
                            # des classes
x=c(1,5,3,4,8,5)            # Création de deux variables quantitatives
y=c(2,7,4,5,3,2)
z=factor(c(2,1,2,3,1,2))    # Création d'une variable qualitative
coul=c("red","green","blue")# Création d'un vecteur de couleurs
symb=c(19,22,24,25)         # Création d'un vecteur de symboles
plot(x,y,pch=19,col=coul[z])# Affichage avec des couleurs différentes
legend(7,6,levels(z),pch=19,col=coul)
plot(x,y,pch=symb[z])       # Affichage avec des symboles différents
legend(7,6,levels(z),pch=symb)
plot(x,y,pch=symb[z],col=coul[z])# Affichage avec des couleurs et
                                 # des symboles différents
legend(7,6,levels(z),pch=symb,col=coul[z])
```

Probabilités

Comportement d'un échantillon gaussien

```
mu=5; sd=1            # Simulation de 3 échan. gaussiens
t1=rnorm(10,mu,sd)    # de tailles 10, 100 et 1000
t2=rnorm(100,mu,sd)
t3=rnorm(1000,mu,sd)

x=seq(0,10,by=0.1)    # Détermination des points (x,f(x)
y=dnorm(x,mu,sd)      # où f est la densité de la loi normale

par(mfrow=c(2,3))     # Tracé des histogrammes, de l'estimation
hist(t1);hist(t2);    # de la fonction de densité par la méthode
 hist(t3)             # des noyaux et de la densité théorique
plot(density(t1))     # pour les 3 échantillons
lines(x,y,col='red')
plot(density(t2))
 lines(x,y,col='red')
plot(density(t3))
lines(x,y,col='red')
par(mfrow=c(1,1))
```

Calcul de prob. par simul. de Monte Carlo

On cherche à calculer $P(X < x)$ sachant que $X \sim \mathcal{N}(5, 1)$

```
mu=5 ; sd=2 ; x=4                # Données

p1=pnorm(x,mu,sd)                # Calcul théorique

m=100000                         # Calcul par simulation
t=rnorm(m,mu,sd)
p2=length(t[t<x])/m

paste('Valeur théorique : ',p1,  # Affichage des résultats
 ' Valeur approchée : ',p2)
```

Comportement de $\overline{X}$

On étudie, à l'aide de simulation, le comportement de la moyenne empirique d'un échantillon issu d'une loi uniforme $U(a, b)$.

```
a=0 ; b=1                              # Données

ln=c(20,50,100,200,500,1000)           # Déf. de 6 tailles d'échan.
E=rep((a+b)/2,6)                       # E(xbar) pour les 6 tailles
V=(b-a)^2/(12*ln)                      # Var(xbar) pour les 6 tailles.

res=matrix(c(E,rep(0,6),V,             # Init. du tableau de résultat
  rep(0,6)),nrow=4,byrow=T)
dimnames(res)=list(c('Esp. théorique',
  'Esp. simulée','Var. théorique',
  'Var. simulée'),ln)

par(mfrow=c(2,3))                      # Découpage de la fenêtre graphique
for(i in 1:length(ln)) {               # Boucle sur les 6 tailles
 n=ln[i]
 t<-matrix(runif(n*1000,a,b),nrow=1000)
 xbar=apply(t,1,mean)
 hist(xbar,xlim=c(0,1),main=paste('n= ',n),
   col='blue',xlab="",ylab="")
 res[2,i]=mean(xbar)
 res[4,i]=var(xbar)
}

res                                    # Affichage des résultats
par(mfrow=c(1,1))                      # Suppression du découpage
```

Statistique exploratoire

```
x=read.table('exemple2.txt')
x.qan=x[sapply(x,is.numeric)]          # Sélection des var. quant.
x.qal=x[sapply(x,is.factor)]           # Sélection des var. qual.
n=dim(x)[1]                            # Taille de l'échantillon
attach(x)

x                                      # Les données
summary(x)                             # Résumé numérique
boxplot(x.qan)                         # Diagramme en boîte
print(mean(x.qan),digits=3)            # Moyennes
print(sd(x.qan),digits=3)              # Ecarts-types
print(cor(x.qan),digits=2)             # Matrice de corrélation
stem(poids)                            # Diagramme en tige et feuille
stem.leaf(poids)                       # Idem (aplpack)
hist(taille,col="blue",main="Taille")  # Histogramme
plot(taille,poids,type='n')            # Affichage du plan (taille,poids)
text(taille,poids,rownames(x))
plot(sort(taille),(1:n)/n,type='s',    # Tracé de la f.d.r
                  ylim=c(0,1))
pairs(x.qan)                           # Graphique matriciel
faces(x.qan)                           # Visages de Chernoff (aplpack)
stars(x.qan)                           # Polygones
layout(matrix(c(1,3,5,2,4,6),2,3, byrow=T)) # Description de var. qual.
for (j in 1:3){
  barplot(table(x.qal[,j]))            # Diagrammes en bâton
  pie(table(x.qal[,j]))                # Diagrammes en camenbert
}
table(yeux,cheveux)                    # Table de contingence
mosaicplot(table(yeux,cheveux))
boxplot(poids~cheveux)                 # Var. quan.  vs. var. qual.
aggregate(x[,c(2,3,5)],list(cheveux=x$cheveux),mean)
detach(d)
```

Statistique décisionnelle

Intervalle de confiance

On cherche à déterminer un intervalle de confiance à 0.95 sur la moyenne d'une loi normale à partir d'un échantillon. La variance est inconnue.

```
x=c(161,155,142,157,150,192,156)  # Données
r=t.test(x,conf.level=0.95)       # Calcul de l'IC
r$conf.int                        # Affichage de l'IC
```

Test du χ^2

On cherche à tester l'équiprobabilité du sexe à la naissance. Pour ceci, un échantillon portant sur 320 familles ayant 5 enfants a donné les résultats suivants : 18 familles n'ont eu aucune fille, 56 en ont eu 1, 110 en ont eu 2, 88 en ont eu 3, 40 en ont eu 4 et 8 en ont eu 5. Sous l'hypothèse d'équiprobabilité, les probabilité de chacun de ces types de famille est respectivement 1/32, 5/32, 10/32, 10/32, 5/32, 1/32.

```
x=c(18,56,110,88,40,8)         # Données
proba=c(1,5,10,10,5,1)/32      # Probabilités théoriques sous H0
chisq.test(x,p=proba)          # Test
```

Test de Kolmogorov Smirnov

```
x=rnorm(30,mean=5,sd=3) # Données
ks.test(x,"pnorm",5,3)  # Test d'adéquation à une loi normale
ks.test(x,"pexp",1/5)   # Test d'adéquation à une loi exponentielle
```

Test de Normalité

```
x1=rnorm(100, mean = 5, sd = 3)            # Echantillon gaussien
x2=runif(100, min = 2, max = 4)            # Echantillon uniforme
shapiro.test(x1)                           # Test pour le 1er échan.
shapiro.test(x2)                           # Test pour le 2nd échan.

par(mfrow=c(1,2))                          # Tracé des droites de
qqnorm(x1,main='Echantillon gaussien')     # Henri
qqline(x1,col='red')
qqnorm(x2,main='Echantillon uniforme')
qqline(x2,col='red')
par(mfrow=c(1,1))
```

Test du χ^2 de contingence

```
x=matrix(c(60,60,42,18,12,2),nrow=3,byrow=T) # Données
chisq.test(x)                                # Test
```

Comparaison à une moyenne donnée

Pour tester l'égalité de la moyenne d'un échantillon à une valeur de référence donnée, on peut utiliser le test de Student, qui est applicable pour un échantillon gaussien ou de grande taille. Sinon, et si l'échantillon est symétrique, on peut appliquer le test de Wilcoxon. L'hypothèse nulle est ici $\mu = 12$ et on a choisi le test bilatéral avec $\alpha^* = 0.05$.

```
x=rnorm(100,10,6)    # Données
t.test(x,mu=12)      # Test de stduent
wilcox.test(x,mu=12) # Test de Wilcoxon
```

Comparaison de deux moyennes

Pour tester l'égalité des moyennes de deux échantillons, on peut utiliser le test de Student, qui est applicable pour des échantillons gaussiens, ou de grandes tailles, et de même variance. Sinon, et si les échantillons sont symétriques, on peut appliquer le test non paramétrique de Wilcoxon, équivalent au test de Mann-Whitney. L'hypothèse nulle est ici $\mu_1 = \mu_2$ et on a choisi le test bilatéral avec $\alpha^* = 0.05$.

```
x1=rnorm(100,10,6)                 # 1er échantillon
x2=rnorm(80,12,6)                  # 1er échantillon
t.test(x1,x2)                      # Test de Student
x=c(x1,x2)                         # Autre solution
f=as.factor(c(rep(1,100),rep(2,80)))
t.test(x~f)
wilcox.test(x1,x2)                 # Test de Wilcoxon
```

Comparaison de deux variances

Pour tester l'égalité des variances de deux échantillons, on peut utiliser le test de Fisher applicable pour un échantillon gaussien. L'hypothèse nulle est ici $\sigma_1^2 = \sigma_2^2$ et on a choisi le test bilatéral avec $\alpha^* = 0.05$.

```
x1=rnorm(100,10,6)                 # 1er échantillon
x2=rnorm(80,15,6)                  # 2nd échantillon
var.test(x1,x2)                    # Test
```

Comparaison de plusieurs moyennes

Pour tester l'égalité des moyennes de plusieurs échantillons, on peut utiliser d'analyse de la variance à 1 facteur, généralisation du test de Student. Ce test est applicable pour des échantillons gaussiens, ou de grandes tailles, et de même variance. Sinon, et si les échantillons sont symétriques, on peut appliquer le test non paramétrique de Kruskal-Wallis.

```
x=read.table('nerveux.txt',header=T)    # Lecture des données
y=c(x$homme,x$boeuf,x$rat,x$grenouille) # Transformation des données
f=as.factor(c(rep(1,5),rep(2,5),
  rep(3,5),rep(4,5)))
oneway.test(y~f,var.equal=T)            # Test anova
kruskal.test(y~f)                       # Test non paramétrique
```

Comparaison de plusieurs variances

Pour tester l'égalité des variances de plusieurs échantillons, on peut utiliser le test de Bartlett qui est applicable pour des échantillons gaussiens. Ce test peut être utilisé, par exemple, pour vérifier l'égalité des variances, hypothèse nécessaire pour le test de l'analyse de la variance décrit dans le paragraphe précédent. On reprend dans l'exemple qui suit les données précédentes.

```
bartlett.test(y~f)
```

Régression linéaire

Dans l'exemple qui suit, on cherche à vérifier s'il existe une relation linéaire entre la moyenne obtenu au baccalauréat et le QI par 10 élèves. Pour ceci, on utilise le modèle de régression linéaire.

```
note=c(8.8,9.6,11.2,10.4,12.8,   # Création du tableau de données
 + 15.2,12.0,16.0,8.0,9.2)
QI=c(108,112,115,118,121,
  125,122,130,96,113)
d=data.frame(note=note,QI=QI)
r<-lm(QI~note)                   # Appel de la fonction de régression
summary(r)                       # Affichage des résultats
r$coef
plot(note,QI)                    # Tracé de la droite de régression
abline(r)
r$residuals                      # Analyse des résidus
par(mfrow=c(2,2))
plot(r)
par(mfrow=c(1,1))
```