Le langage R

Gérard Govaert

14 février 2007

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*. 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 au 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-2.4.1-win32.exe`. Il suffit ensuite de lancer ce programme qui installe le logiciel R. Il est conseillé de l'installer directement sous `c:` pour faciliter l'utilisation de `Sweave`, outil permettant d'insérer des commandes R dans un fichier LaTeX.

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.

Le lancement déclenche l'exécution éventuelle :

1. des fichiers d'environnement du site (`R_ENVIRON` ou `R_HOME/etc/Renviron.site`),
2. des fichiers d'environnement de l'utilisateur (`.Renviron` recherché d'abord sur le working directory, puis sur le home directory),
3. des fichiers *profile* du site (`R_PROFILE` ou `R_HOME/etc/Rprofile.site`),
4. des fichiers *profile* de l'utilisateur (`.Rprofile` (recherché d'abord sur le working directory, puis sur le home directory),
5. du fichier `.Rdata` situé dans le working directory
6. et enfin du fichier `.First` en faisant une recherche suivant le « search path ».

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 »).

Le *Home directory* correspond à la notation `HOME` dans la documentation. Il s'agit plutôt d'un mode de recherche de certains fichiers. En positionnant la variable d'environnement `R_USER` sur un certain répertoire et en plaçant les fichiers `.Renviron`, `.Rprofile` et `Rconsole` dans ce répertoire, le démarrage de R se fera en utilisant ces fichiers, sauf si ces mêmes fichiers existent dans le working directory.

Voici un exemple de paramétrage :

1. Initialiser la variable d'environnement Windows `R_USER` à `C:\gerard\Logiciels\R\etc`
2. Créer, dans ce répertoire, le 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")
```

Les commandes `getwf()`, `?Rconsole`, `?Renviron` et `Sys.getenv("R_USER")` peuvent être utiles pour comprendre ce fonctionnement.

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`.

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

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 à 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,...

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, 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 (`logical`, `is.logical`, `as.logical`, `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`, `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 exécute 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

Tout un ensemble 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

data, library : chargement de données ou de modules

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

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

save, save.image : écriture en format R

load, data, library : lecture en format R

module foreign : fonctions permettant des données avec les 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.

Statistiques descriptives

range : étendue

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

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

quantile

fivenum : résumé d'une distribution

table : construction d'une table de contingence

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

tabulate

aggregate : Statistiques pour des sous-ensembles de données

princomp : ACP

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, dotchar, pie, boxplot, stem, sunflowerplot, stripplot, coplot, interactionplot, matplot, fourplot, assocplot, mosaicplot, pairs, qqnorm, qqline, qqplot, contour, persp, image, stars, faces (module aplpack), symbols

Paramètres 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

Fonc tions interactives : locator et identify

Paramètres 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

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