

Table des matières

Introduction	iii
1 Présentation du langage R	1
1.1 Bref historique	1
1.2 Description sommaire de R	2
1.3 Interfaces	3
1.4 Stratégies de travail	4
1.5 Éditeurs de texte	5
1.6 Anatomie d'une session de travail	8
1.7 Répertoire de travail	8
1.8 Consulter l'aide en ligne	9
1.9 Où trouver de la documentation	9
1.10 Exemples	10
1.11 Exercices	11
2 Bases du langage S	11
2.1 Commandes S	11
2.2 Conventions pour les noms d'objets	12
2.3 Les objets S	13
2.4 Vecteurs	15
2.5 Matrices et tableaux	16
2.6 Listes	17
2.7 <i>Data frames</i>	17
2.8 Indixage	18
2.9 Exemples	19
2.10 Exercices	26
3 Opérateurs et fonctions	29
3.1 Opérations arithmétiques	29

3.2	Opérateurs	30
3.3	Appels de fonctions	30
3.4	Quelques fonctions utiles	31
3.5	Structures de contrôle	34
3.6	Exemples	36
3.7	Exercices	42
4	Exemples résolus	45
4.1	Calcul de valeurs présentes	45
4.2	Fonctions de probabilité	46
4.3	Fonction de répartition de la loi gamma	48
4.4	Algorithme du point fixe	50
4.5	Exercices	51
5	Fonctions définies par l'utilisateur	53
5.1	Définition d'une fonction	53
5.2	Retourner des résultats	53
5.3	Variables locales et globales	54
5.4	Exemple de fonction	54
5.5	Fonctions anonymes	54
5.6	Débogage de fonctions	56
5.7	Styles de codage	57
5.8	Exemples	58
5.9	Exercices	61
6	Concepts avancés	67
6.1	L'argument '...'	67
6.2	Fonction apply	67
6.3	Fonctions lapply et sapply	69
6.4	Fonction mapply	70
6.5	Fonction replicate	71
6.6	Classes et fonctions génériques	72
6.7	Exemples	73
6.8	Exercices	77
7	Fonctions d'optimisation	81
7.1	Le package MASS	81
7.2	Fonctions d'optimisation disponibles	82
7.3	Exemples	87
7.4	Exercices	88

8	Générateurs de nombres aléatoires	91
8.1	Générateurs de nombres aléatoires	91
8.2	Fonctions de simulation de variables aléatoires	91
8.3	Exercices	93
9	Planification d'une simulation en S	95
9.1	Introduction	95
9.2	Première approche : avec une boucle	96
9.3	Seconde approche : avec <code>sapply</code>	96
9.4	Variante de la seconde approche	100
9.5	Comparaison des temps de calcul	101
9.6	Gestion des fichiers	102
9.7	Exécution en lot	102
9.8	Quelques remarques	103
A	GNU Emacs et ESS : la base	105
A.1	Mise en contexte	105
A.2	Configuration de l'éditeur	105
A.3	<i>Emacs-ismes</i> et <i>Unix-ismes</i>	106
A.4	Commandes d'édition de base	106
A.5	Sélection de texte	107
A.6	Mode ESS	107
A.7	Session de travail type	110
B	Utilisation de ESS et S-Plus sous Windows	113
B.1	Tout dans Emacs	113
B.2	Combinaison Emacs et S-Plus GUI	114
C	GNU Free Documentation License	115
C.1	APPLICABILITY AND DEFINITIONS	115
C.2	VERBATIM COPYING	117
C.3	COPYING IN QUANTITY	118
C.4	MODIFICATIONS	118
C.5	COMBINING DOCUMENTS	120
C.6	COLLECTIONS OF DOCUMENTS	121
C.7	AGGREGATION WITH INDEPENDENT WORKS	121
C.8	TRANSLATION	121
C.9	TERMINATION	122
C.10	FUTURE REVISIONS OF THIS LICENSE	122
	ADDENDUM: How to use this License for your documents	122

Réponses des exercices	125
Bibliographie	137
Index	139

1 Présentation du langage R

Objectifs

- > faire ceci
- > faire cela

1.1 Bref historique

À l'origine fut le S, un langage pour «programmer avec des données» développé chez Bell Laboratories à partir du milieu des années 1970 par une équipe de chercheurs menée par John M. Chambers. Au fil du temps, le S a connu quatre principales versions communément identifiées par la couleur du livre dans lequel elles étaient présentées : version «originale» (*Brown Book*; Becker et Chambers, 1984), version 2 (*Blue Book*; Becker et collab., 1988), version 3 (*White Book*; Chambers et Hastie, 1992) et version 4 (*Green Book*; Chambers, 1998) ; voir aussi Chambers (2000) et Becker (1994) pour plus de détails.

Dès la fin des années 1980 et pendant près de vingt ans, le S a principalement été popularisé par une mise en œuvre commerciale nommée S-PLUS. En 2008, Lucent Technologies a vendu le langage S à Insightful Corporation, ce qui a effectivement stoppé le développement du langage par ses auteurs originaux. Aujourd'hui, le S est commercialisé de manière relativement confidentielle sous le nom Spotfire S+ par TIBCO Software.

Ce qui a fortement contribué à la perte d'influence de S-PLUS, c'est une nouvelle mise en œuvre du langage développée au milieu des années 1990. Inspirés à la fois par le S et par Scheme (un dérivé du Lisp), Ross Ihaka et Robert Gentleman proposent un langage pour l'analyse de données et les graphiques qu'ils nomment R (Ihaka et Gentleman, 1996). À la suggestion de Martin Maechler de l'ETH de Zurich, les auteurs décident d'intégrer leur nouveau langage au projet GNU¹, faisant

1. <http://www.gnu.org>

de R un logiciel libre.

Ainsi disponible gratuitement et ouvert aux contributions de tous, R gagne rapidement en popularité là même où S-PLUS avait acquis ses lettres de noblesse, soit dans les milieux académiques. De simple dérivé «*not unlike S*», R devient un concurrent sérieux à S-PLUS, puis le surpasse lorsque les efforts de développement se rangent massivement derrière le projet libre. D'ailleurs John Chambers place aujourd'hui ses efforts de réflexion et de développement dans le projet R (Chambers, 2008).

1.2 Description sommaire de R

R est un environnement intégré de manipulation de données, de calcul et de préparation de graphiques. Toutefois, ce n'est pas seulement un «autre» environnement statistique (comme SPSS ou SAS, par exemple), mais aussi un langage de programmation complet et autonome.

Tel que mentionné précédemment, le R est un langage principalement inspiré du S et de Scheme (Abelson et collab., 1996). Le S était à son tour inspiré de plusieurs langages, dont l'APL (autrefois un langage très prisé par les actuaires) et le Lisp. Comme tous ces langages, le R est *interprété*, c'est-à-dire qu'il requiert un autre programme — l'*interprète* — pour que ses commandes soient exécutées. Par opposition, les programmes de langages *compilés*, comme le C ou le C++, sont d'abord convertis en code machine par le compilateur puis directement exécutés par l'ordinateur.

Le programme que l'on lance lorsque l'on exécute R est en fait l'interprète. Celui-ci attend que l'on lui soumette des commandes dans le langage R, commandes qu'il exécutera immédiatement, une à une et en séquence.

Par analogie, Excel est certes un logiciel de manipulation de données, de mise en forme et de préparation de graphiques, mais c'est aussi au sens large un langage de programmation interprété. On utilise le langage de programmation lorsque l'on entre des commandes dans une cellule d'une feuille de calcul. L'interprète exécute les commandes et affiche les résultats dans la cellule.

Le R est un langage particulièrement puissant pour les applications mathématiques et statistiques (et donc actuarielles) puisque précisément développé dans ce but. Parmi ses caractéristiques particulièrement intéressantes, on note :

- > langage basé sur la notion de vecteur, ce qui simplifie les calculs mathématiques et réduit considérablement le recours aux structures itératives (boucles) ;
- > pas de typage ni de déclaration obligatoire des variables ;
- > programmes généralement courts, en général quelques lignes de code seulement ;

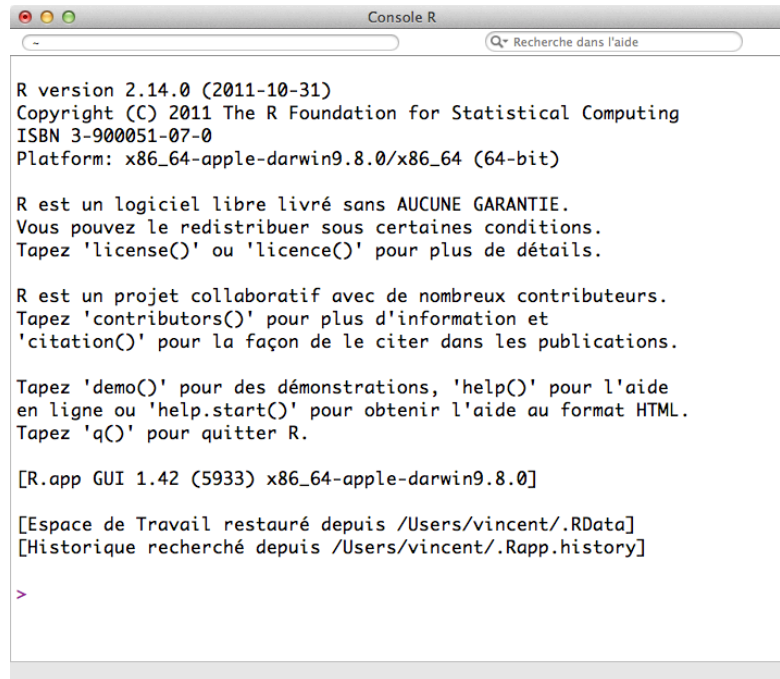


FIG. 1.1: Fenêtre de la console sous Mac OS X au démarrage de R

> temps de développement très court.

1.3 Interfaces

R est d'abord et avant tout une application n'offrant qu'une invite de commande du type de celle présentée à la figure 1.1. En soi, cela n'est pas si différent d'un tableur tel que Excel : la zone d'entrée de texte dans une cellule n'est rien d'autre qu'une invite de commande², par ailleurs aux capacités d'édition plutôt réduites.

- > Sous Unix et Linux, R n'est accessible que depuis la ligne de commande du système d'exploitation (terminal). Aucune interface graphique n'est offerte avec la distribution de base de R.
- > Sous Windows, une interface graphique plutôt rudimentaire est disponible. Elle facilite certaines opérations tel que l'installation de packages externes, mais elle offre autrement peu de fonctionnalités additionnelles pour l'édition de code R.

2. Merci à Markus Gesmann pour cette observation.

- > L'interface graphique de R sous Mac OS X est la plus élaborée. Outre la console présentée à la figure 1.1, l'application R.app comporte de nombreuses fonctionnalités, dont un éditeur de code assez complet.

1.4 Stratégies de travail

Dans la mesure où R se présente essentiellement sous forme d'une invite de commande, il existe deux grandes stratégies de travail avec cet environnement statistique.

1. On entre des expressions à la ligne de commande pour les évaluer immédiatement :

```
> 2 + 3  
[1] 5
```

On peut également créer des objets contenant le résultat d'un calcul. Ces objets sont stockés dans la mémoire de l'ordinateur :

```
> x <- exp(2)  
> x  
[1] 7.389056
```

Lorsque la session de travail est terminée, on sauvegarde une image de la mémoire sur le disque dur de l'ordinateur afin de pouvoir conserver les objets pour une future séance de travail :

```
> save.image()
```

Par défaut, l'image est sauvegardée dans un fichier nommé .RData dans le dossier de travail actif (voir la section 1.7) et cette image est automatiquement chargée en mémoire au prochain lancement de R, tel qu'indiqué à la fin du message d'accueil :

```
[Sauvegarde de la session précédente restaurée]
```

Cette approche, dite de «code virtuel et objets réels» a un gros inconvénient : le code utilisé pour créer les objets n'est pas sauvegardé entre les sessions de travail. Or, celui-ci est souvent bien plus compliqué que l'exemple ci-dessus. De plus, sans accès au code qui a servi à créer l'objet x, comment savoir ce que la valeur 7,389056 représente au juste ?

2. L'approche dite de «code réel et objets virtuels» considère que ce qu'il importe de conserver d'une session de travail à l'autre n'est pas tant les objets que le code qui a servi à les créer. Ainsi, on sauvegardera dans ce que l'on nommera

des *fichiers de script* nos expressions R et le code de nos fonctions personnelles. Par convention, on donne aux fichiers de script un nom se terminant avec l'extension `.R`.

Avec cette approche, les objets sont créés au besoin en exécutant le code des fichiers de script. Comment ? Simplement en copiant le code du fichier de script et en le collant dans l'invite de commande de R. La figure 1.2 illustre schématiquement ce que le programmeur R a constamment sous les yeux : d'un côté son fichier de script et, de l'autre, l'invite de commande R dans laquelle son code a été exécuté.

La méthode d'apprentissage préconisée dans cet ouvrage suppose que le lecteur utilisera cette seconde approche d'interaction avec R.

1.5 Éditeurs de texte

Dans la mesure où l'on a recours à des fichiers de script tel qu'expliqué à la section précédente, l'édition de code R bénéficie grandement d'un bon éditeur de texte pour programmeur. Dans certains cas, l'éditeur peut même réduire l'opération de copier-coller à un simple raccourci clavier.

- > Un éditeur de texte est différent d'un traitement de texte en ce qu'il s'agit d'un logiciel destiné à la création, l'édition et la sauvegarde de fichiers textes purs, c'est-à-dire dépourvus d'information de présentation et de mise en forme. Les applications Bloc-notes sous Windows ou TextEdit sous OS X sont deux exemples d'éditeurs de texte simples.
- > Un éditeur de texte pour programmeur saura en plus reconnaître la syntaxe d'un langage de programmation et assister à sa mise en forme : indentation automatique du code, coloration syntaxique, manipulation d'objets, etc.

Le lecteur peut utiliser l'éditeur de texte de son choix pour l'édition de code R. Certains éditeurs offrent simplement plus de fonctionnalités que d'autres.

- > GNU Emacs est un très ancien, mais aussi très puissant éditeur pour programmeur. À la question 6.2 de la foire aux questions de R (Hornik, 2011), «Devrais-je utiliser R à l'intérieur de Emacs ?», la réponse est : «Oui, absolument.»

En effet, combiné avec le mode ESS (*Emacs Speaks Statistics*), Emacs offre un environnement de développement aussi riche qu'efficace. Entre autres fonctionnalités uniques à Emacs, le fichier de script et l'invite de commandes R sont regroupés dans la même fenêtre, comme on peut le voir à la figure 1.3.

Logiciel porte étendard du mouvement libre, Emacs est disponible gratuitement et à l'identique sur toutes les plateformes supportées par R, dont Windows, OS X et Linux.

```
## Fichier de script simple contenant des expressions R pour
## faire des calculs et créer des objets.
2 + 3

## Probabilité d'une loi de Poisson(10)
x <- 7
10^x * exp(-10) / factorial(x)

## Petite fonction qui fait un calcul trivial
f <- function(x) x^2

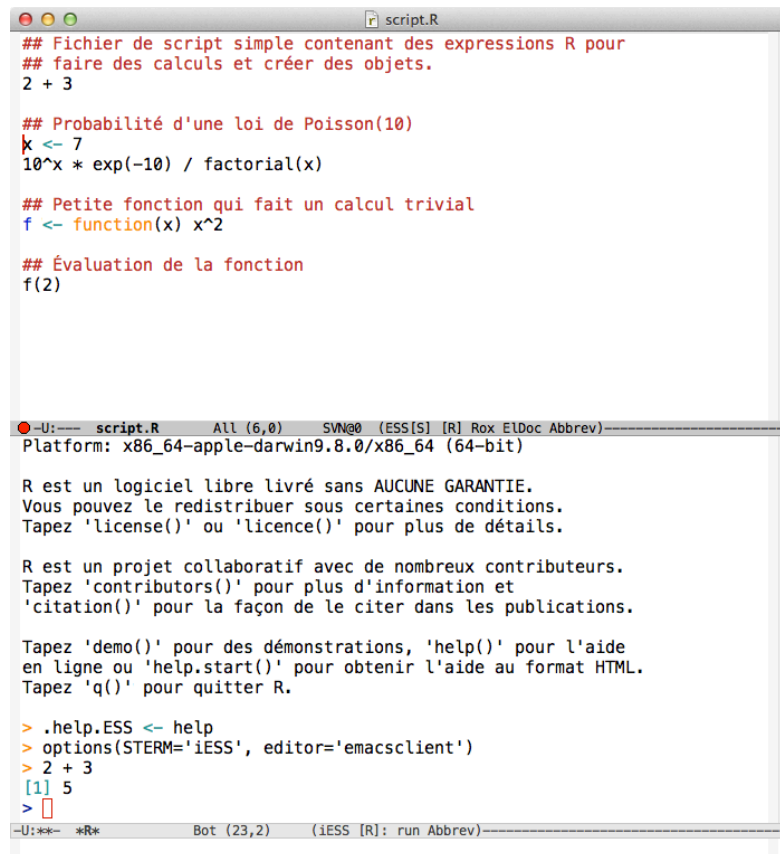
## Évaluation de la fonction
f(2)
```

```
R version 2.14.0 (2011-10-31)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

[...]

> ## Fichier de script simple contenant des expressions R pour
> ## faire des calculs et créer des objets.
> 2 + 3
[1] 5
>
> ## Probabilité d'une loi de Poisson(10)
> x <- 7
> 10^x * exp(-10) / factorial(x)
[1] 0.09007923
>
> ## Petite fonction qui fait un calcul trivial
> f <- function(x) x^2
>
> ## Évaluation de la fonction
> f(2)
[1] 4
```

FIG. 1.2: Fichier de script (en haut) et invite de commande R dans laquelle les expressions R ont été exécutées (en bas). Les lignes débutant par # dans le fichier de script sont des commentaires ignorés par l'interprète de commandes.



```

## Fichier de script simple contenant des expressions R pour
## faire des calculs et créer des objets.
2 + 3

## Probabilité d'une loi de Poisson(10)
k <- 7
10^x * exp(-10) / factorial(x)

## Petite fonction qui fait un calcul trivial
f <- function(x) x^2

## Évaluation de la fonction
f(2)

```

```

-U:--- script.R All (6,0) SVN@0 (ESS[S] [R] Rox EIDoc Abbrev)
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

R est un logiciel libre livré sans AUCUNE GARANTIE.
Vous pouvez le redistribuer sous certaines conditions.
Tapez 'license()' ou 'licence()' pour plus de détails.

R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.

> .help.ESS <- help
> options(STERM='iESS', editor='emacsclient')
> 2 + 3
[1] 5
> 
-U:*** *R* Bot (23,2) (iESS [R]: run Abbrev)

```

FIG. 1.3: Fenêtre de GNU Emacs sous OS X en mode d'édition de code R. Dans la partie du haut, on retrouve le fichier de script de la figure 1.2 et dans la partie du bas, l'invite de commandes R.

- > Le présent auteur distribue des versions de GNU Emacs pour Windows et OS X simples à installer et livrées avec le mode ESS et quelques autres extensions. Consulter le site <http://vgoulet.act.ulaval.ca/emacs/>.
- > Il existe plusieurs autres options plus faciles à apprivoiser que Emacs pour éditer du code R. Nous recommandons :
 - sous Windows, l'éditeur Notepad++ muni de l'extension NppToR (Redd, 2010), tous deux des logiciels libres ;
 - toujours sous Windows, le logiciel WinEdt muni de l'extension libre R-WinEdt (Ligges, 2003) ;

- sous OS X, tout simplement l'éditeur de texte très complet intégré à l'application R. app, ou alors l'éditeur de texte commercial TextMate (essai gratuit de 30 jours) ;
- sous Linux, Vim et Kate semblent les choix les plus populaires après Emacs dans la communauté R.

1.6 Anatomie d'une session de travail

Dans ses grandes lignes, toute session de travail R se réduit aux étapes ci-dessous.

1. Ouvrir un fichier de script existant ou en créer un nouveau à l'aide de l'éditeur de texte de son choix.
2. Démarrer une session R en cliquant sur l'icône de l'application si l'on utilise une interface graphique, ou alors en suivant la procédure expliquée à l'annexe A si l'on utilise GNU Emacs.
3. Au cours de la phase de développement, on fera généralement de nombreux aller-retours la ligne de commande où l'on testera des commandes et le fichier de script où l'on consignera le code R que l'on souhaite sauvegarder et les commentaires qui nous permettront de s'y retrouver plus tard.
4. Sauvegarder son fichier de script et quitter l'éditeur.
5. Si nécessaire — et c'est rarement le cas — sauvegarder l'image mémoire de la session R avec `save.image()`. En fait, on ne voudra sauvegarder nos objets R que lorsque ceux-ci sont très longs à créer comme, par exemple, les résultats d'une simulation.
6. Quitter R en tapant `q()` à la ligne de commande ou en fermant l'interface graphique par la procédure usuelle. Encore ici, la manière de procéder est quelque peu différente dans GNU Emacs ; voir l'annexe A.

Évidemment, les étapes 1 et 2 sont interchangeables, tout comme les étapes 4, 5 et 6.

1.7 Répertoire de travail

Le répertoire de travail (*workspace*) de R est le dossier par défaut dans lequel le logiciel 1) va rechercher des fichiers de script ou de données ; et 2) va sauvegarder l'image mémoire dans le fichier `.RData`. Le dossier de travail est déterminé au lancement de R.

- > Les interfaces graphiques démarrent avec un répertoire de travail par défaut. Pour le changer, utiliser l'entrée appropriée dans le menu Fichier (Windows) ou Divers (Mac OS X). Consulter aussi les foires aux questions spécifiques aux interfaces graphiques (Ripley et Murdoch, 2011; Iacus et collab., 2011) pour des détails additionnels sur la gestion des répertoires de travail.
- > Avec GNU Emacs, la situation est un peu plus simple puisque l'on doit spécifier un répertoire de travail chaque fois que l'on démarre un processus R.

1.8 Consulter l'aide en ligne

Les rubriques d'aide des diverses fonctions disponibles dans R contiennent une foule d'informations ainsi que des exemples d'utilisation. Leur consultation est tout à fait essentielle.

- > Pour consulter la rubrique d'aide de la fonction `foo`, on peut entrer à la ligne de commande
 - > `?foo`
 - ou
 - > `help(foo)`

1.9 Où trouver de la documentation

La documentation officielle de R se compose de six guides accessibles depuis le menu Aide des interfaces graphiques ou encore en ligne dans le site du projet R³. Pour le débutant, seuls *An Introduction to R* et, possiblement, *R Data Import/Export* peuvent s'avérer des ressources utiles à court terme.

Plusieurs livres — en versions papier ou électronique, gratuits ou non — ont été publiés sur R. On en trouvera une liste exhaustive dans la section Documentation du site du projet R.

Depuis plusieurs années maintenant, les ouvrages de Venables et Ripley (2000, 2002) demeurent des références standards *de facto* sur les langages S et R. Plus récent, Braun et Murdoch (2007) participe du même effort que le présent ouvrage en se concentrant sur la programmation en R plutôt que sur ses applications statistiques.

3. <http://www.r-project.org>

1.10 Exemples

```
### Générer deux vecteurs de nombres pseudo-aléatoires issus
### d'une loi normale centrée réduite.
x <- rnorm(50)
y <- rnorm(x)

### Graphique des couples (x, y).
plot(x, y)

### Graphique d'une approximation de la densité du vecteur x.
plot(density(x))

### Générer la suite 1, 2, ..., 10.
1:10

### La fonction 'seq' sert à générer des suites plus générales.
seq(from = -5, to = 10, by = 3)
seq(from = -5, length = 10)

### La fonction 'rep' sert à répéter des valeurs.
rep(1, 5)      # répéter 1 cinq fois
rep(1:5, 5)    # répéter le vecteur 1,...,5 cinq fois
rep(1:5, each = 5) # répéter chaque élément du vecteur cinq fois

### Arithmétique vectorielle.
v <- 1:12      # initialisation d'un vecteur
v + 2          # additionner 2 à chaque élément de v
v * -12:-1     # produit élément par élément
v + 1:3        # le vecteur le plus court est recyclé

### Vecteur de nombres uniformes sur l'intervalle [1, 10].
v <- runif(12, min = 1, max = 10)
v

### Pour afficher le résultat d'une affectation, placer la
### commande entre parenthèses.
( v <- runif(12, min = 1, max = 10) )

### Arrondi des valeurs de v à l'entier près.
( v <- round(v) )

### Créer une matrice 3 x 4 à partir des valeurs de
### v. Remarquer que la matrice est remplie par colonne.
```

```
( m <- matrix(v, nrow = 3, ncol = 4) )

### Les opérateurs arithmétiques de base s'appliquent aux
### matrices comme aux vecteurs.
m + 2
m * 3
m ^ 2

### Éliminer la quatrième colonne afin d'obtenir une matrice
### carrée.
( m <- m[, -4] )

### Transposée et inverse de la matrice m.
t(m)
solve(m)

### Produit matriciel.
m %*% m           # produit de m avec elle-même
m %*% solve(m)    # produit de m avec son inverse
round(m %*% solve(m)) # l'arrondi donne la matrice identité

### Consulter la rubrique d'aide de la fonction 'solve'.
?solve

### Liste des objets dans l'espace de travail.
ls()

### Nettoyage.
rm(x, y, v, m)
```

1.11 Exercices

- 1.1 Démarrer un processus S-Plus ou R à l'intérieur de Emacs.
- 1.2 Exécuter un à un les exemples de la section précédente. Une version électronique du code de cette section est disponible dans le site mentionné dans la préface.
- 1.3 Consulter les rubriques d'aide d'une ou plusieurs des fonctions rencontrées lors de l'exercice précédent. Observer d'abord comment les rubriques d'aide sont structurées — elles sont toutes identiques — puis exécuter quelques lignes d'exemples.
- 1.4 Lire le chapitre 1 de Venables et Ripley (2002) et exécuter les commandes de l'exemple de session de travail de la section 1.3. Bien que davantage orienté

vers les applications statistiques que vers la programmation, cet exemple démontre quelques-unes des possibilités du langage S.

Bibliographie

- Abelson, H., G. J. Sussman et J. Sussman. 1996, *Structure and Interpretation of Computer Programs*, 2^e éd., MIT Press, ISBN 0-26201153-0.
- Becker, R. A. 1994, «A brief history of S», cahier de recherche, AT&T Bell Laboratories. URL <http://cm.bell-labs.com/cm/ms/departments/sia/doc/94.11.ps>.
- Becker, R. A. et J. M. Chambers. 1984, *S: An Interactive Environment for Data Analysis and Graphics*, Wadsworth, ISBN 0-53403313-X.
- Becker, R. A., J. M. Chambers et A. R. Wilks. 1988, *The New S Language: A Programming Environment for Data Analysis and Graphics*, Wadsworth & Brooks/Cole, ISBN 0-53409192-X.
- Braun, W. J. et D. J. Murdoch. 2007, *A First Course in Statistical Programming with R*, Cambridge University Press, ISBN 978-0-52169424-7.
- Chambers, J. M. 1998, *Programming with Data: A Guide to the S Language*, Springer, ISBN 0-38798503-4.
- Chambers, J. M. 2000, «Stages in the evolution of S», URL <http://cm.bell-labs.com/cm/ms/departments/sia/S/history.html>.
- Chambers, J. M. 2008, *Software for Data Analysis: Programming with R*, Springer, ISBN 978-0-38775935-7.
- Chambers, J. M. et T. J. Hastie. 1992, *Statistical Models in S*, Wadsworth & Brooks/Cole, ISBN 0-53416765-9.
- Hornik, K. 2011, «The R FAQ», URL <http://cran.r-project.org/doc/FAQ/R-FAQ.html>, ISBN 3-900051-08-9.
- Iacus, S. M., S. Urbanek et R. J. Goedman. 2011, «R for Mac OS X FAQ», URL <http://cran.r-project.org/bin/macosx/RMacOSX-FAQ.html>.

- Ihaka, R. et R. Gentleman. 1996, «R: A language for data analysis and graphics», *Journal of Computational and Graphical Statistics*, vol. 5, n° 3, p. 299–314.
- Ligges, U. 2003, «R-winedt», dans *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*, édité par K. Hornik, F. Leisch et A. Zeileis, TU Wien, Vienna, Austria, ISSN 1609-395X. URL <http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/>.
- Redd, A. 2010, «Introducing NppToR: R interaction for Notepad++», *R Journal*, vol. 2, n° 1, p. 62–63. URL http://journal.r-project.org/archive/2010-1/RJournal_2010-1.pdf.
- Ripley, B. D. et D. J. Murdoch. 2011, «R for Windows FAQ», URL <http://cran.r-project.org/bin/windows/base/rw-FAQ.html>.
- Venables, W. N. et B. D. Ripley. 2000, *S Programming*, Springer, New York, ISBN 0-38798966-8.
- Venables, W. N. et B. D. Ripley. 2002, *Modern Applied Statistics with S*, 4^e éd., Springer, New York, ISBN 0-38795457-0.

Index

Les numéros de page en caractères gras indiquent les pages où les concepts sont introduits, définis ou expliqués.

!, **30**
!=, **30**
*, **30**
, **30
+, **30**
-, **30**
->, **11**
..., **67**, **98**
/, **30**
<, **30**
<-, **11**
<<-, **54**
<=, **30**
=, **11**
==, **30**
>, **30**
>=, **30**
[, **133**
[[]], **17**
[], **15**, **16**, **18**
%%%, **30**, **63**
%/%, **30**
%/, **30**
%in%, **32**, **38**
%o%, **34**, **40**
&, **30**
^, **30**
_, **12**
abs, **55**, **58**, **59**
affectation, **11**
apply, **35**, **40**, **42**, **67**, **67**, **68**, **69**, **73**,
97–99, **101**
array, **16**, **21**, **23**, **73**
arrondi, **33**
as.data.frame, **18**
attach, **18**, **24**
attribut, **14**

bibliothèque, **32**
boucle, **35**, **51**, **96**
break, **35**, **42**, **55**, **58**, **59**
by, **10**, **37**
byrow, **31**

c, **15**
cat, **76**
cbind, **17**, **18**, **23**, **39**
ceiling, **33**, **38**
character, **15**, **22**
character (mode), **13**, **15**, **18**
choose, **46**
class, **20–24**, **72**, **76**
class (attribut), **14**
colMeans, **34**, **42**, **68**, **73**

- colnames, 18
- colSums, **34**, 40, 42, 49, 68
- colVars, **34**, 103
- compilé (langage), 2
- complex (mode), **13**, 18
- cummax, **33**, 39
- cummin, **33**, 39
- cumprod, **33**, 39, 46
- cumsum, **33**, 39
- data, 31, 37, 88
- data frame, 84
- data frame, **18**
- data.frame, **18**
- data.frame (classe), 18
- dbeta, 87
- dbinom, 47
- density, 10
- detach, **18**, 25
- dgamma, 87, 88
- diag, **33**, 39, 76
- diff, **33**, 39
- différences, 33
- dim, 20–24, 39, 73
- dim (attribut), **14**, 16
- dimension, 14, 26
- dimnames, 21, 31, 103
- dimnames (attribut), **14**
- distribution
 - bêta, 92
 - binomiale, 46, 92
 - binomiale négative, 92
 - Cauchy, 92
 - exponentielle, 92
 - F, 92
 - gamma, 48, 64, 92
 - géométrique, 92
 - hypergéométrique, 92
 - khi carré, 92
 - log-normale, 92, 93
 - logistique, 92
 - mélange discret, 93
 - mélange Poisson/gamma, 93
 - normale, 62, 64, 92
 - Pareto, 64, 78
 - Poisson, 46, 51, 92
 - t, 92
 - uniforme, 92
 - Weibull, 92
 - Wilcoxon, 92
- dnorm, 62
- Dossier de travail, voir répertoire de travail
- dpois, 47
- écart type, 33
- ecdf, 134
- else, **34**, 40, 41, 59, 76
- Emacs, 5, 54, 56, 57, 105–107
 - C-_, 107
 - C-g, 107
 - C-r, 107
 - C-s, 107
 - C-SPC, 107
 - C-w, 107
 - C-x 0, 107
 - C-x 1, 107
 - C-x 2, 107
 - C-x b, 107
 - C-x C-c, 106
 - C-x C-f, 106, 110
 - C-x C-s, 106, 110
 - C-x C-w, 106
 - C-x k, 106
 - C-x o, 107, 110
 - C-x u, 107
 - C-y, 107
- configuration, 105
- et S-Plus, 113–114
- M-%, 107

- M-w, 107
- M-y, 107
- nouveau fichier, 106
- rechercher et remplacer, 107
- sélection, 107
- sauvegarder, 106
- sauvegarder sous, 106
- ESS, 5, 54, 56, 107–109
 - C-c C-e, 108, 110
 - C-c C-f, 54, 109, 110
 - C-c C-j, 108
 - C-c C-l, 109
 - C-c C-n, 56, 108, 110
 - C-c C-o, 108, 110
 - C-c C-q, 108, 111
 - C-c C-r, 109, 110
 - C-c C-s, 109
 - C-c C-v, 108, 109
- h, 109
- l, 109
- M-h, 108
- M-n, 108
- M-p, 108
- n, 109
- p, 109
- q, 109
- r, 109
- x, 109
- étiquette, 14, 26
- eval, 59
- exists, 24, 25
- exp, 47, 49
- expression, 11
- expression, 59
- extraction, voir aussi indîçage
 - dernières valeurs, 32
 - éléments différents, 32
 - premières valeurs, 32
- F, voir FALSE
- factorial, 42, **47**
- FALSE, 12
- fitdistr, 87
- floor, **33**, 38
- fonction
 - anonyme, 54
 - appel, 30
 - débogage, 56
 - définie par l'utilisateur, 53
 - générique, 72
 - maximum local, 83
 - minimum, 84, 85
 - minimum local, 83
 - optimisation, 86
 - racine, 82
 - résultat, 53
- for, **35**, 36, 40, 41, 60, 70, 96–98
- function, **53**, 55, 58–61, 74–76, 87, 88, 98–101
- function (mode), **13**
- gamma, 42, **47**
- head, **32**, 38
- hist, 75, 78
- if, **34**, 36, 40–42, 55, 58, 59, 76
- ifelse, **35**
- Im, 87
- indîçage
 - liste, **17**, 26
 - matrice, 16, **18**, 27
 - vecteur, **18**, 26
- interprété (langage), 2
- is.na, **15**, 25, 41
- is.null, **15**
- lapply, 35, 67, 69, **69**, 70, 74, 96
- length, 10, **13**, 19–23, 37–39
- lfactorial, 42
- lgamma, 42

- library, 88
- list, **17**, 21, 23, 24, 74, 88, 98, 99, 101
- list (mode), **13**, 17
- liste, 17
- log, 88
- logical, **15**, 22
- logical (mode), **13**, 14, 15, 18
- longueur, **13**, 26
- lower, 87
- ls, 11

- mapply, **70**, 74
- match, **32**, 38
- matrice, 16, 43, 62, 63, 67
 - diagonale, 33
 - identité, 33
 - inverse, 33
 - moyennes par colonne, 34
 - moyennes par ligne, 34
 - somme par colonne, 34
 - sommes par ligne, 34
 - transposée, 33
 - variance par colonne, 34
 - variance par ligne, 34
- matrix, 11, **16**, 20–24, 37, 62, 73, 74, 76, 97, 98
- max, 10, **33**, 39, 73
- maximum
 - cumulatif, 33
 - d'un vecteur, 33
 - local, 83
 - parallèle, 33
 - position dans un vecteur, 32
- mean, 14, 21, **33**, 39, 73, 75, 97–100
- median, **33**, 39, 97–100
- médiane, 33
- methods, **72**, 76
- min, 10, **33**, 39
- minimum
 - cumulatif, 33
 - d'un vecteur, 33
 - d'une somme, 84
 - fonction non linéaire, 84, 85
 - local, 83
 - parallèle, 33
 - position dans un vecteur, 32
- mode, **13**, 26
- mode, **13**, 21, 23, 24, 76
- moyenne
 - arithmétique, 33
 - harmonique, 51
 - pondérée, 51, 77
 - tronquée, 33
- ms, **84**, 88

- NA, **14**
- na.rm, **14**, 21, 73
- names, 18, 21, 22, 25
- names (attribut), **14**
- nchar, 20
- ncol, 11, 20, 22, 31, **34**, 37, 39, 73, 97, 98
- next, **35**
- nlm, **85**, 88
- nlmin, **84**, 85, 88
- nlminb, **86**
- noms d'objets
 - conventions, 12
 - réservés, 12
- Notepad++, 7
- nrow, 11, 20–22, 31, **34**, 37, 39, 73, 97, 98
- NULL, **15**
- NULL (mode), **15**
- numeric, **15**, 20, 22, 40, 41, 60
- numeric (mode), **13**, 15, 18

- optim, **86**, 88
- optimize, **83**, 87
- order, **32**, 38

- outer, **34**, 40, 49, 54, 56
- package, 32
 - MASS, 81, 86, 87
- paste, 78
- pgamma, 49
- plot, 10, 20, 72, 93
- pmax, **33**, 39, 40
- pmin, **33**, 39
- pnorm, 62
- point fixe, 50, 54
- polyroot, **83**, 87
- print, 36, 40–42, 56, 58, 59, 72, 73, 76
- prod, **33**, 34, 39, 40, 73
- produit, 33
 - cumulatif, 33
 - extérieur, 34
- q, 8
- quantile, 33
- quantile, **33**, 39
- Répertoire de travail, 8–9
- racine
 - d'un polynôme, 83
 - d'une fonction, 82
- .Random.seed, 91
- rang, 32
- range, **33**, 39, 97–100
- rank, **32**, 38
- rbind, **17**, 18, 23
- Re, 87
- Recall, 62
- renverser un vecteur, 32
- rep, 10, **32**, 37, 40, 42, 74, 75
- repeat, **35**, 41, 50, 55, 58, 59
- répétition de valeurs, 32
- replace, 25, 39
- replicate, **71**, 75, 100, 101
- return, **53**
- rev, **32**, 38, 39, 46
- rgamma, 88
- rm, 11
- rnorm, 10, 75
- round, 10, 11, **33**, 38
- row.names, 18, 103
- rowMeans, **34**, 42, 68, 97–99, 101
- rownames, 18, 97, 98, 103
- rowSums, **34**, 40, 42, 68, 73
- rowVars, **34**, 103
- runif, 10, **91**, 97–100
- S, 1, 2
- S+, 1
- S-PLUS, 1
- sample, 20, 25, 39, 43, 73–75, **93**
- sapply, 35, 67, 69, **69**, 70, 71, 74, 75, 96, 98, 99, 131
- save.image, 4, 8
- Scheme, 1, 2
- sd, **33**, 39, 75
- seq, 10, 24, **32**, 37, 42, 74
- set.seed, **91**
- simulation
 - nombres uniformes, 91
 - planification, 95–103
 - variables aléatoires, 91
- solve, 11, **33**, 39
- somme, 33
 - cumulative, 33
- sort, **32**, 37
- source, 102
- start, 55, 58, 59, 88
- Startup, 109
- style, 57
- suite de nombres, 32
- sum, 14, **33**, 39, 40, 73, 74, 88
- summary, **33**, 39, 72
- switch, **35**
- sys.time, 60, 101
- system.time, 61, 101

T, voir TRUE
t, 11, **33**, 39
table, 93
tableau, 16, 43, 67
tail, **32**, 38
tri, 32
TRUE, 12
trunc, **33**, 38

unique, **32**, 38
uniroot, **82**, 87
unlist, **17**, 24, 74
upper, 87

valeur présente, 45, 51, 52
var, **33**, 39, 61, 97–99, 101
variable
 globale, 54
 locale, 54
variance, 33
vecteur, 15, 29
vide, voir NULL
vraisemblance, 84, 87

which, **32**, 38
which.max, **32**, 38
which.min, **32**, 38
while, **35**, 41, 61
WinEdt, 7

