

Logiciel de Statistiques R

Didier Chauveau

UPR CaST - Université d'Orléans



Didier Chauveau - UPR CaST - Orléans

R statistique

1 / 32

Genèse : Les origines de R

- Initié par Ross Ihaka et Robert Gentleman (1993) (University of Auckland, New Zealand)
- Implémentation du langage "S" (Chambers et al., 1984-1998) S-PLUS est une autre implémentation, commerciale, du langage S (Insightful Corporation)
- ACM Software System Award, 1998.
- De facto devenu :
 - le standard de la recherche en Computational Statistics et exploration de données
 - l'environnement idéal pour populariser les nouvelles méthodologies en statistique

Homepage www.r-project.org

Didier Chauveau - UPR CaST - Orléans

R statistique

2 / 32

Quelques avantages techniques de R

- Gratuit (GPL2) et Open Source (écrit en C)
- Multi-plateforme (UNIX, LINUX, MacOS X, Windows...)
- Développé et maintenu par les meilleurs experts en "Statistical Computing" : **The R Core Team** (B. Ripley, L. Tierney, J. Chambers...)
- Langage interactif, orienté objet, extensible
- **Pensé pour l'exploration et la modélisation de données** (à la différence de e.g. MATLAB, Solab,...)
- Interface simple vers du code C, Fortran si besoin
- Outils de **calcul parallèle** accessibles

Didier Chauveau - UPR CaST - Orléans

R statistique

3 / 32

"Philosophie" de R

Une analyse statistique implique :

- Exploration des données (résumés numériques, graphiques, modèles)
- Choix des outils guidés par la visualisation des données
- Possibilité d'adapter les outils existants : l'écriture de fonctions est naturelle

Didier Chauveau - UPR CaST - Orléans

R statistique

4 / 32

Packages et sites CRAN

■ **Package** : ensemble de fonctions liées à un thème

- Inclut un environnement de développement
 - code R, C, ...
 - écriture de documentation "text-like"
- package testé et documenté avant mise en ligne
- un package peut dépendre ("hériter") d'autres packages
- Les packages "valides" sont accessibles via les sites miroirs

CRAN = Comprehensive R Archive Network

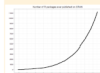
Didier Chauveau - UPR CaST - Orléans

R statistique

5 / 32

Comprehensive R Archive Network

cran.r-project.org/web/packages



More than 10,000 packages!

- Indicateur de l'activité en statistical computing
- Problèmes : qualité des packages, metadata...

Didier Chauveau - UPR CaST - Orléans

R statistique

6 / 32

Comprehensive R Archive Network et packages

CRAN

Exemple : Package mixtools

cran.r-project.org/web/packages



Didier Chauveau - UPR CaST - Orléans

R statistique

7 / 32

Pour s'y retrouver : CRAN Task Views

Thème	Contenu
Bayesian	Bayesian Inference
Classical	Classical Statistical Inference
Cluster	Cluster Analysis
Computation	Computational Statistics
Confounders	Confounders
Control	Control
Correlation	Correlation
Covariance	Covariance
Decision	Decision
Design	Design
Diagnostics	Diagnostics
Distribution	Distribution
Distance	Distance
Dynamic	Dynamic
Ecology	Ecology
Economics	Economics
Education	Education
Energy	Energy
Environment	Environment
Finance	Finance
Forecasting	Forecasting
Genetics	Genetics
Health	Health
History	History
Image	Image
Language	Language
Law	Law
Life Sciences	Life Sciences
Machine Learning	Machine Learning
Mathematics	Mathematics
Medical	Medical
Meta-analysis	Meta-analysis
Modeling	Modeling
Networks	Networks
Optimization	Optimization
Programming	Programming
Psychology	Psychology
Quality Control	Quality Control
Reliability	Reliability
Simulation	Simulation
Statistics	Statistics
Teaching	Teaching
Time Series	Time Series
Visualization	Visualization
Web	Web
Workflows	Workflows

Didier Chauveau - UPR CaST - Orléans

R statistique

8 / 32

D'autres extensions/outils construits sur R

GUI (voir TP) <http://www.rstudio.com>

Interface web dynamique de visualisation <http://shiny.rstudio.com/gallery/>

R Markdown : générateur de documents (pdf, html...) incluant code R et code dynamique

package RHadop pour le BigData

Didier Chauveau - UPR CaST - Orléans

R statistique

9 / 32

Données = table individus-caractères

On "pose" p "questions" (mesures) à n "individus" = unités statistiques (personne, animal, item, jour, lieu...)

Données sous la forme d'une
Table individus-caractères
table ou matrice ($n \times p$)
souvent $n \gg p$

$$\begin{bmatrix} X_1^1 & \dots & X_1^p & \dots & X_1^n \\ \vdots & & \vdots & & \vdots \\ X_i^1 & \dots & X_i^p & \dots & X_i^n \\ \vdots & & \vdots & & \vdots \\ X_n^1 & \dots & X_n^p & \dots & X_n^n \end{bmatrix}$$

- La i -ième ligne X_i est la "réponse" de l'individu i :
- La s -ième colonne X^s est le s -ième **caractère** ou **variable statistique**

Format des données pour les méthodes de Data Mining

Rappel : Nature des variables statistiques

Une variable (colonne) de la table individus-caractères peut être de 2 natures :

- Qualitative** : (facteur)
 - à valeur dans un ensemble fini de modalités
 - pas de relation d'ordre entre les modalités
 - pas d'opérations numériques entre modalités
 exemples : CSP, groupe sanguin, région, code postal...
- Quantitative** : à valeur dans \mathbb{R} (ou \mathbb{N} , \mathbb{Z})
 - exemples : mesure physique, revenu, taux de CO_2 ...
- Qualitative ordonnée**, variables pouvant avoir les deux statuts (nb d'enfants d'un ménage, classement subjectif d'un parfum...)

Objets

Structures comme tous les langage "de haut niveau" :

Vecteur, matrice, tableau multidim, objet structuré...

Structures spécifiques :

Classes d'objets spécialisés pour les statistiques, eg :

- factor** : vecteur à valeurs modalités sans ordre
- facteur qualitatif** (e.g. Région, CSP, sexe...)
- data.frame** : liste de vecteurs de même longueur, de classes quelconques
- = **table individus-caractères**

Classes et méthodes

Langage orienté objet

Les fonctions peuvent disposer de **méthodes adaptées aux classes** de leur argument

Une fonction se comporte différemment suivant la classe de l'argument avec lequel on l'utilise.

Exemple : méthodes définies pour la fonction `summary` qui résume un objet :

```
> methods(summary)
[1] summary.aov          summary.aovlist       summary.apell+
[4] summary.connection    summary.data.frame    summary.Date
[7] summary.default       summary.ecdf+         summary.factor
...

```

Objets

Comme tout langage de haut niveau :

- numeric** : valeur numérique
- vector** : collection d'objets de même mode (type)
- matrix** : tableau de dimension 2
- array** : tableau de dimension $d \geq 2$

Mais aussi :

- factor** : **vecteur à valeurs modalités**
- list** : collection d'objets de types quelconques
- data.frame** : **liste de vecteurs de même longueur**

Langage spécialisé pour les statistiques

data.frame = structure d'une table individus-caractères
factor = **facteur qualitatif** (sexe, groupe sanguin, CSP...)

Pour commencer

Informations générales

```
# ceci est un commentaire
R.version # infos techniques...
citation() # pour bibliographie d'articles...
citation # sans "()", définition de la fonction!
```

aide en ligne

```
help() # aide de l'aide
?citation # aide de la fonction citation
?*=* # aide pour un opérateur
```

Chargement de packages et démo de fonctions

```
install.packages("ade4") # download & install
library(ade4) # charge ce package
example(sudi.pca) # data mining: ACP
```

Éléments et vecteurs

Calculs, affectations

```
x = 2
x <- 1; y <- 2 # ";" sépare les commandes
x = 4 # "presque toujours" équivalent à x <- 4
rm(x) # suppression de x du "workspace"
```

Création de vecteurs

```
c <- c(1,2,5,8,9) # "c" combine; différent de MATLAB!
c <- c("toto","titi") # vecteurs de caractères (deux ...)
length(a) # longueur du vecteur

a = rep(1,10) # répétition, voir ?rep
b = rep(c(10,12),5) # idem sur (10,12)
s = 1:10 # boucle (cf Matlab, Scilab)
seq = seq(1,10,by=2) # séquences, voir ?seq
seq(1,10,length=20) # discrétisation en 20 points
seq(1,10,len=20) # idem, "Partial Matching" !!!
```

Règles concernant les arguments de fonctions

Exemple : définition de la fonction

```
seq(from = 1, to = 1, by = ((to-from)/(length.out-1)),
length.out = NULL, ...)
```

- pour les arguments fournis sans nom **l'ordre compte** :
`seq(1,10,2) + seq(from=1, to=10, by=2)`
- pour les arguments nommés l'ordre ne compte pas :
`seq(to=10, by=2, from=1)` est valide
- les arguments non précisés prennent les valeurs par défaut indiquées dans l'aide de la fonction :
`seq(to=10, by=2)` est `seq(1,10,2)`
- le "partial matching" permet de ne spécifier que partiellement le nom d'un paramètre
`"len" => "length.out"`, car pas d'ambiguïté

Manipulations sur les vecteurs (1)

Opérations

```
a+b; a*b; a/b # "élément-wise", même longueur
a<b # test élément-wise
log(b) # fait math élément-wise
a+s # pas de même longueur: "recycling" !
```

Quelques fonctions statistiques élémentaires

```
min(a) # minimum
mean(a) # moyenne empirique
sd(a) # écart-type (standard deviation)
sort(a) # tri
```

Échantillonnage dans un ensemble

```
a=sample(1:10) # par défaut permutation de 1:10
?sample # aide, arguments...
sample(a,2) # tirage de 2 éléments de a
```

Manipulations sur les vecteurs (2)

Extraction d'éléments :

```
e = seq(1,20,by=2)
e[3] # élément d'un vecteur
e[1:5] # 1:5 vaut {1,2,3,4,5} (cf acilab)
e[c(3,6,8)] # sous-vecteur explicite
e[-3] # tout sauf e[3]
e[1:5][4] # élément i du vecteur e[1:5]
e[1:5][2:4][1] # dévies les résultats! (QCM...)
```

Opérateurs logiques

```
a <- c(T,T,F,F) # T ou TRUE, F ou FALSE
!a # not a
b <- c(T,F); a & b # ET logique, b recyclé ici
# extraction de sous-vecteurs par tests
a[a > 5] # seules les valeurs testées à TRUE
e[e != 5] # sauf les e[i]=5
```

Manipulations sur les matrices

Construction et extraction

```
a=1:10 # on reprend
7matrix # méthode par défaut?
m = matrix(a,nrow=6) # avec recycling (warning)
dim(m) # voir aussi nrow(m) et ncol(m)
m[,3] # ligne colonne de m
m[2,3] # élément m[i,j]
class(m) # différent de mode(m)
```

Opérations

```
p = matrix(a,ncol=2) # par défaut byrow = FALSE
q = matrix(a,nrow=2)
q %*% p # produit de matrices
q %/% m # pb dimensions
diag(m) # diagonale, même si pas carrée
t(m) # transposée
apply(m,1,sum) # opération (sum) par ligne (1)
```

Exercices

- Construire une matrice M à 10 lignes et 5 colonnes constituée d'entiers tirés au hasard dans $\{0, \dots, 9\}$
- Calculer la moyenne des colonnes de M
- Calculer l'écart-type des lignes de M

Manipulations sur les listes

Une liste est une collection d'objets

- ordonnés
- de types (mode, class) quelconques

Construction et extraction

```
ls <- list(1:5, "toto", 7) # éléments sans noms
ls <- list(x=1:5, nom="toto", z=7) # idem avec noms
summary(ls) # default method: noms, classes et modes
ls # liste du contenu
ls[[1]] # extraction élément 1 de la liste
ls$x # opérateur % lorsque le nom est connu
```

Fonctions génériques, méthodes et classes

Fonctions qui s'adaptent à leurs arguments

Certaines fonctions ont des **méthodes** dont le résultat dépend de la **classe** de leurs arguments.

Exemple : résumé de structures

```
class(a); summary(a) # a est "numeric"
class(c); summary(c) # c est "character"
summary(ls) # résumé d'une liste (default)
methods(summary) # méthodes associées
```

Exemple : graphiques élémentaires (et beaucoup d'autres...)

```
plot(a) # base/r série des observations
x <- rnorm(100); y <- x + rnorm(100) # x iid ~ N(0,1)...
plot(x, y, pch=20) # nuage de points
```

Data.frame = table individus-caractères

Construction à la main

```
a=sample(1:10); b=rep(c(1,2),5);
c = c(rep("M",6), rep("F",4)) # facteur qualitatif
m = data.frame(a,b,c) # création de la structure
class(x) # data.frame
class(c); class(x$c) # conversion par défaut
```

Manipulations de base

```
names(x) # noms des colonnes = variables
row.names(x) # noms des lignes = individus
x[, x$a]; x$a[2] # affiche x, la variable a, extraction
x[[1]] # idem car c'est une liste!
```

→ Poursuivre avec des données réelles...

Exemple simple : Données State Facts

Source : Bureau of the Census, US (1977).

Sélection de 7 variables en cours :


```
Etat Noms des "individus" = 50 états (code à 2 lettres)
Pop Population estimée en 1975
Revenu Revenu moyen par habitant
Aph taux d'analphabétisme (en % de population)
Mourtr taux de criminalité pour 100 000 habitants
Diplome % de population de niveau équivalent au Bac
Region région d'appartenance des états (Northeast, South, North Central, West)
```

le jeu de données utilisé en TP à 10 variables

data.frame = table individus-caractères

data.frame = liste particulière (et collection) de (p) vecteurs :
de types (mode, class) quelconques
de même longueur (n)

Les data.frame peuvent se manipuler comme des listes, mais aussi comme des tableaux.


C'est une structure fondamentale pour la finalité de  : manipuler et analyser des tableaux de données.

Importation de données

Fichier texte "brut" des données State Facts complètes

```
Abb Etat Pop ... Aire Region
AL Alabama 3615 ... 50708 South
AK Alaska 365 ... 566432 West
...
```

TP (1) :

- Récupérer les données "texte" en local ou en ligne
StateFacts.txt
- Sauver le fichier dans votre **répertoire de travail**
- Indiquer à  le répertoire de travail, cf menu
Fichier → Changer de répertoire de travail...

Importation de données

Importation sous forme de data.frame

```
states <- read.table("StateFacts.txt",
  header=T, # ligne 1 = noms des variables
  row.names=1) # labels individus colonne 1

ls() # liste des objets chargés/existants
head(states) # affiche "le début" d'un objet
```

Méthode de travail après import de données :

Sauvegarde d'un dataframe au format R binaire compressé

```
save(states, file="StateFacts.Rdata")
```

Chargement d'un dataframe au format R binaire compressé

```
load("StateFacts.Rdata")
```

Premières manipulations

Taille de la table, attributs...

```
dim(states) # p=6 variables + labels individus
nrow(states) # n, aussi par dim(states)[1]
attach(states) # rend "visible" les variables de states
detach(states) # opération inverse
```

```
colnames(states) # noms des variables
row.names(states) # labels individus
```

Extractions...

```
states$Pop # car c'est aussi une "list"
states[2,] # car structure ordonnée
states[1,3] # car c'est aussi une "matrix"
class(states$Region) # facteur créé au chargement
```

Exemples de statistiques (résumés) numériques

Fonctions agissant sur les variables

```
mean(Apb)
var(Meurtre)
max(Revenu) # statistiques numériques...
```

Fonctions agissant sur le data.frame

```
summary(states) # min, max, quantiles...
sd(states) # pas défini pour un facteur!
sd(states[, -6]) # ni pour un data.frame...
cor(states[, -6]) # corrélations, sauf facteur Region
```

Statistiques par niveaux de facteur(s)

```
tapply(Meurtre, Region, mean) # mean(Meurtre) par Region
by(states[, -6], Region, colMeans) # pour le dataframe
```

Exercices TP (3)

En une seule commande à chaque fois :

- 1 Afficher la moyenne empirique de la variable Revenu
- 2 Afficher l'écart-type de Meurtre
- 3 Combien y-a-t-il d'États de Revenu moyen > 5000 ?
- 4 Afficher les moyennes empiriques des variables quantitatives de la table states
- 5 changer le nom de la variable Analphabétisme en Apb (attention à attach()...)
- 6 Calculer $\sum_{i=1}^n X_i^2$ pour la variable $X = \text{Alphabétisme}$

Exercices TP (4 à 8)

Terminer la Feuille de TP n°1.