

# Logiciel de Statistiques R

Didier Chauveau

UFR CoST – Université d'Orléans



# Genèse : Les origines de

- 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](http://www.r-project.org)

# Quelques avantages techniques de

- 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, Scilab,...)
- Interface simple vers du code C, Fortran si besoin
- Outils de **calcul parallèle** accessibles


# “Philosophie” de

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

# Packages et sites CRAN

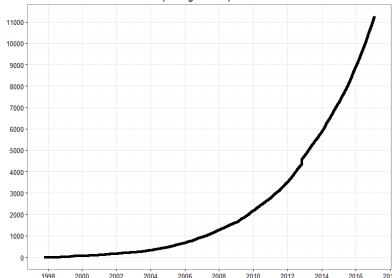
## ■ **Package** : ensemble de fonctions liées à un thème

-  inclut un environnement de développement
  - code R, C, ...
  - écriture de documentation “ $\text{\LaTeX}$ -like”
- package testé et documenté avant mise en ligne
- un package peut *dépendre* (“hériter”) d’autres packages
- Les packages “validés” sont accessibles via les sites miroirs  
**CRAN** = **C**omprehensive **R** **A**rchive **N**etwork

# Comprehensive R Archive Network

`cran.r-project.org/web/packages`

Number of R packages ever published on CRAN



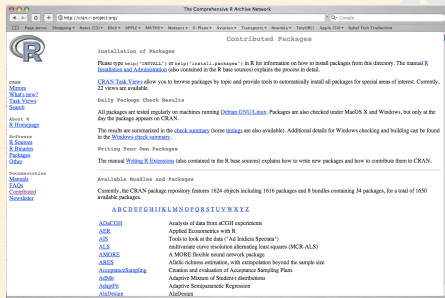
- Indicateur de l'activité en *statistical computing*
- Problèmes : qualité des packages, metadata, ...

*More than 10,000 packages !*

# Comprehensive R Archive Network et packages

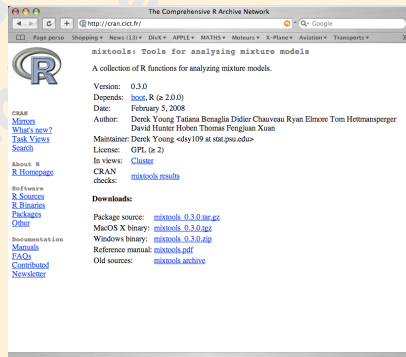
## CRAN

[cran.r-project.org/web/packages](http://cran.r-project.org/web/packages)



## Exemple : Package mixtools

[cran.cict.fr/web/packages/mixtools](http://cran.cict.fr/web/packages/mixtools)




# Pour s'y retrouver : CRAN Task Views


## CRAN Task Views



<a href="#">Bayesian</a>	Bayesian Inference
<a href="#">ChemPhys</a>	Chemometrics and Computational Physics
<a href="#">ClinicalTrials</a>	Clinical Trial Design, Monitoring, and Analysis
<a href="#">Cluster</a>	Cluster Analysis & Finite Mixture Models
<a href="#">DifferentialEquations</a>	Differential Equations
<a href="#">Distributions</a>	Probability Distributions
<a href="#">Econometrics</a>	Econometrics
<a href="#">Environmetrics</a>	Analysis of Ecological and Environmental Data
<a href="#">ExperimentalDesign</a>	Design of Experiments (DoE) & Analysis of Experimental Data
<a href="#">Finance</a>	Empirical Finance
<a href="#">Genetics</a>	Statistical Genetics
<a href="#">Graphics</a>	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
<a href="#">HighPerformanceComputing</a>	High-Performance and Parallel Computing with R
<a href="#">MachineLearning</a>	Machine Learning & Statistical Learning
<a href="#">MedicalImaging</a>	Medical Image Analysis
<a href="#">MetaAnalysis</a>	Meta-Analysis
<a href="#">Multivariate</a>	Multivariate Statistics
<a href="#">NaturalLanguageProcessing</a>	Natural Language Processing
<a href="#">NumericalMathematics</a>	Numerical Mathematics
<a href="#">OfficialStatistics</a>	Official Statistics & Survey Methodology
<a href="#">Optimization</a>	Optimization and Mathematical Programming
<a href="#">Pharmacokinetics</a>	Analysis of Pharmacokinetic Data
<a href="#">Phylogenetics</a>	Phylogenetics, Especially Comparative Methods
<a href="#">Psychometrics</a>	Psychometric Models and Methods
<a href="#">ReproducibleResearch</a>	Reproducible Research
<a href="#">Robust</a>	Robust Statistical Methods
<a href="#">SocialSciences</a>	Statistics for the Social Sciences
<a href="#">Spatial</a>	Analysis of Spatial Data
<a href="#">SpatioTemporal</a>	Handling and Analyzing Spatio-Temporal Data



# D'autres extensions/outils construits sur

 : GUI (voir TPs) <https://www.rstudio.com>

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

 R Markdown : générateur de documents (pdf, html,...) incluant code  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  et code  dynamique

 : package RHadoop pour le BigData

# 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  $n \gg p$

$$\begin{bmatrix} X_1^1 & \cdots & X_1^s & \cdots & X_1^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X_\ell^1 & \cdots & X_\ell^s & \cdots & X_\ell^p \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n^1 & \cdots & X_n^s & \cdots & X_n^p \end{bmatrix}$$

- La  $\ell$ -ième ligne  $\mathbf{X}_\ell$  est la “réponse” de l'individu  $\ell$  :
- La  $s$ -ième colonne  $\mathbf{X}^s$  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.aspell*
[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(dudi.pca) # data mining: ACP
```

# Éléments et vecteurs

## Calculatrice, affectations

`2 + 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

`a <- c(1,2,5,8,9)` # "c"ombine: différent de MATLAB!

`c <- c("toto","tu")` # vecteur de caracteres (deux c...)

`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)

`s = 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 ⇔ à `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 element-wise
log(b)        # ft math element-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 scilab)
e[c(3,6,8)]    # sous-vecteur explicite
e[-3]          # tout sauf e[3]
e[1:5][4]      # élément 4 du vecteur e[1:5]
e[1:5][2:4][1] # devinez 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
e[e > 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
?matrix                               # méthode par défaut?
m = matrix(a,nrow=4)                  # avec recycling (warning)
dim(m)                                # voir aussi nrow(m) et ncol(m)
m[,3]                                  # 3ème colonne de m
m[2,3]                                # élément m(2,3)
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)                                  # transpose
apply(m,1,sum)                        # opération (sum) par ligne (1)

```

# Exercices

- 1 Construire une matrice  $M$  à 10 lignes et 5 colonnes constituée d'entiers tirés au hasard dans  $\{0, \dots, 9\}$
- 2 Calculer la moyenne des colonnes de  $M$
- 3 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

```
lsn <- list(1:5, "toto", T)      # éléments sans noms
ls <- list(x=1:5,nom="toto",z=T) # idem avec noms
summary(ls) # default method: noms, classes et modes
ls          # liste du contenu
lsn[[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)              # basic: 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

- data.frame = liste particulière cad collection de  $(p)$  vecteurs :
- de types (mode, classe) 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.



# 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
x = 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
Apb	taux d’analphabétisme (en % de population)
Meurtre	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 jeux de données utilisé en TP a 10 variables

# 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) :

- ➊ Récupérer les données “texte” en local ou en ligne  
`StateFacts.txt`
- ➋ Sauver le fichier dans votre **répertoire de travail**
- ➌ 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 objects 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  $X = \text{Analphabétisme}$

## Exercices TP (4 à 8)

Terminer la Feuille de TP n°1.