

Introduction au logiciel

L. BELLANGER

lise.bellanger@univ-nantes.fr

Page web: <http://www.math.sciences.univ-nantes.fr/~bellanger/>

Master 1 Ingénierie Statistique
Dpt de Mathématiques - Université de Nantes

1

Plan

1. Qu'est-ce que R ?
Exemples, historique, fournisseurs
2. Introduction à RMarkdown
3. Les bases du langage
4. Introduction à l'étude de données avec R
5. Calcul matriciel sous R : pour aller plus loin

2

1. Qu'est-ce que ?

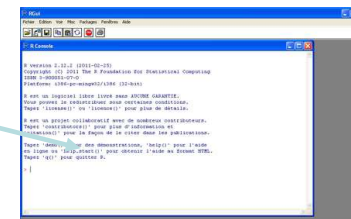
Exemple, historique,
distributeurs

3

R est un logiciel **statistique** et **graphique**

- R : logiciel multiplateforme
- R : un logiciel et un langage
- R : logiciel libre et gratuit téléchargeable gratuitement à partir de: <http://www.r-project.org>
- R : un logiciel au développement très actif

Console

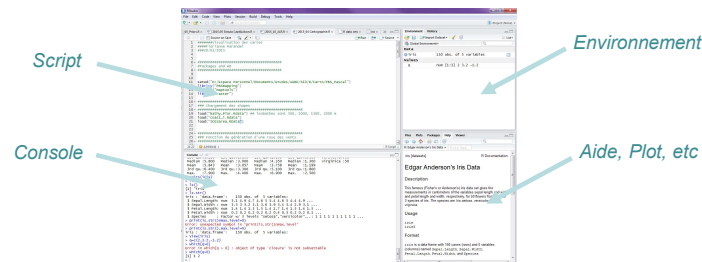


```
R version 3.5.1 (2018-07-02)
Copyright (C) 2018 The R Foundation for Statistical Computing
This program is distributed under the terms of the GNU General Public License
You should have received a copy of the GNU General Public License
along with this program. See the file 'COPYING' for more details.
R is a free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
For more details see the 'COPYING' file.
>
```

4

Rstudio : un environnement de travail pour R

- Rstudio : nécessite l'installation de R
- Rstudio : même langage que R
- Rstudio : logiciel libre et gratuit



5

Exemples de syntaxe

```
x <- c(3,6,7,8)
```

Crée un vecteur x

```
mean(x)
```

Affiche la moyenne de x

```
var(x)
```

Affiche la variance de x

```
y <- c(4.5,3,2,-0.5)
```

Crée un autre vecteur y

```
plot(x,y)
```

Graphique de x versus y

```
lm(y ~ x)
```

Régression linéaire de y sur x

```
x.matrix <-
```

```
matrix(runif(9), ncol=3)
```

Crée une matrice 3x3

```
solve(x.matrix)
```

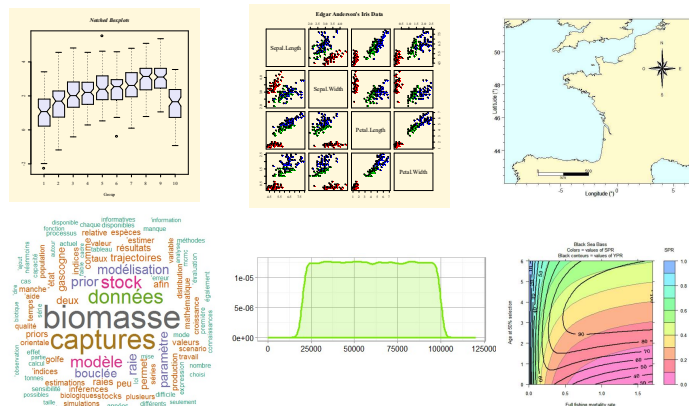
Inverse de la matrice x.matrix

```
cube <- function(z)
  z^3
```

Crée une fonction "mettre au cube"

6

Un aperçu des possibilités graphiques



7

Ressources pour R

Le site WWW principal : <http://www.r-project.org/>

Voir aussi : <http://cran.r-project.org/>

FAQ: <http://cran.r-project.org/doc/FAQ/R-FAQ.html>
ou directement dans R en tapant `help.start()`

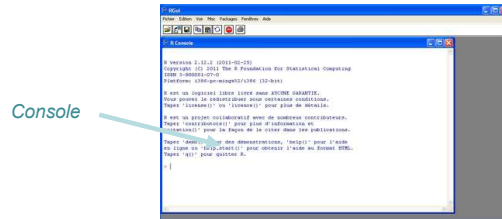
Une introduction en français :
http://cran.r-project.org/doc/contrib/Rdebuts_fr.pdf

Une autre vue sur la science "statistique et ordinateur":
<http://www.omegahat.org/>

8

Démarrage et arrêt de R

- Obtenir le logiciel (version Windows)
<http://cran.r-project.org/bin/windows/base/R-?.?.?.win.exe>
- Installer le logiciel
Lancez le programme **R-?.?.?.win.exe**, puis suivez les instructions affichées à l'écran.



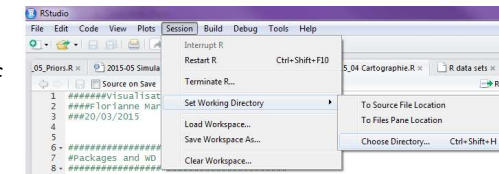
- Pour quitter : **q("yes")**
- Pour interrompre des calculs : taper **ESC** ou appuyer sur la touche "Echap"

9

Le répertoire de travail

- Par défaut, R lit et écrit dans le **répertoire de travail**
- Connaître le répertoire de travail de R :
`getwd()`
`[1] "C:/Documents and Settings/toto/Mes documents"`
Pour retrouver vos données soyez attentif au répertoire de travail
- Changer le répertoire de travail de R :
Utiliser le menu déroulant: 'Fichier' ⇒ 'Changer le répertoire courant'

Ou avec
Rstudio



10

Changer de répertoire de travail : fonction `setwd()`

- Le chemin du répertoire de travail peut être fastidieux à écrire
▪ `setwd("C:/Documents and Settings/toto/Mes documents/Enquetes")`
- Deuxième solution :
▪ `setwd("~/")` #le répertoire perso de l'utilisateur est : ~
▪ `setwd("~/Enquetes")` #erreur si ce répertoire n'existe pas

Remarque : pour connaître le contenu d'un répertoire : utiliser `dir()`

- `dir("~/rgis")`

```
[1] "Carto.pdf" "departement"
[3] "FRA.dbf"   "FRA.prj"
[5] "FRA.qpj"   "FRA.shp"
[7] "FRA.shx"   "france"
[9] "LR.png"    "maps.pdf"
```

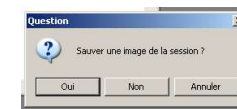
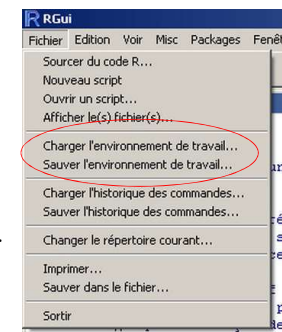
- `dir("~/rgis", pattern=".shp")`

```
[1] "FRA.shp"
```

11

L'environnement de travail

- Les données créées au cours d'une session peuvent être sauvegardées. Les fichiers de données R portent l'extension **.Rdata**
- La commande « **Sauver l'environnement de travail** » copie toutes les données en mémoire dans un fichier à l'extension **.Rdata**
- À la fermeture, R vous propose de sauvegarder l'environnement de travail.



12

Demander de l'aide

Toutes les fonctions sont décrites de manière très détaillée dans l'aide "on-line".

- En savoir plus sur une fonction

`?mean`
`?median`
`?IQR`

- Comment importe-t-on des fichiers textes ?

`?read.table`

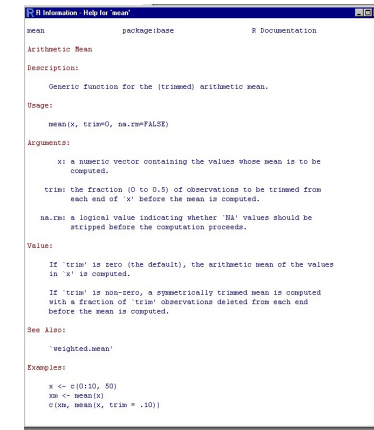
13

Comment lire les pages de l'aide "on-line"

Lisez la courte introduction au début.

Descendez à la fin et répéter les exemples !
 On comprend souvent rapidement l'idée de la fonction.

Lisez les détails de la fonction plus tard.



Écrire des scripts avec un éditeur de texte

Première solution : éditeur de script R

- Ouvrez l'éditeur de texte fourni par R
- Saisissez vos commandes

```
# 1000 valeurs suivant une loi normale {0,1}
```

```
x<-rnorm(1000,mean=0, sd=1)
```

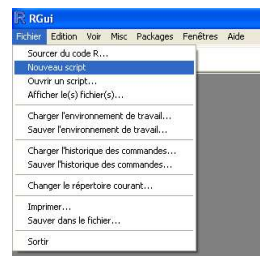
```
#Calcul de la moyenne
```

```
mean(x)
```

```
#Calcul de la variance
```

```
var(x)
```

- Transférez-les dans la fenêtre R
 - Ctrl-A tout sélectionner
 - Ctrl-R coller vers R-console

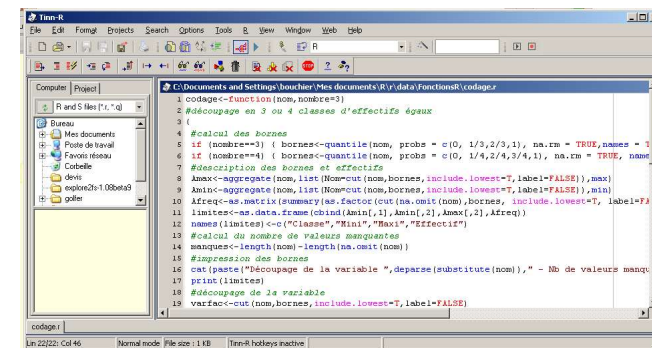


15

Écrire des scripts avec un éditeur de texte

Deuxième solution : Tinn-R (uniquement pour Windows)

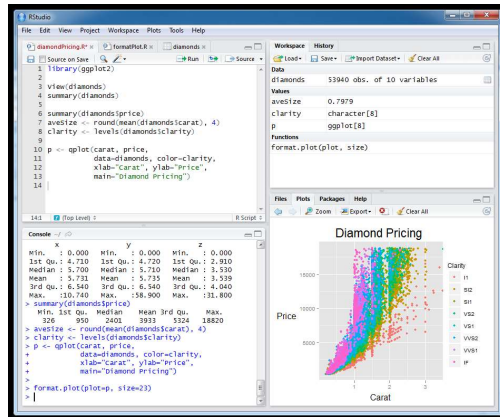
- disponible à cette adresse <http://www.sciviews.org/Tinn-R/>
- avec coloration syntaxique et copie-coller direct vers R



Écrire des scripts avec un éditeur de texte

Troisième solution : Rstudio

- multiplateforme, disponible à cette adresse : <http://rstudio.org/>



Utiliser des bibliothèques de fonctions

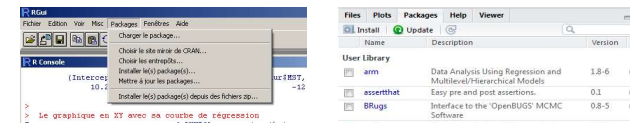
- Une fonction est un objet R.
 - Un gd nb sont prédéfinies: **mean**, **min**, ...
 - On peut en créer
 - On peut installer des bibliothèques/paquets/packages qui complètent les fonctionnalités de R : **MASS**, ...

- Si la bibliothèque est déjà installée sur votre machine :

1) On peut utiliser le menu de R

Packages -> Charger le package, puis cliquez sur la bibliothèque désirée

2) On peut aussi taper l'instruction suivante dans la console
library(nom de la bibliothèque)



R

R Studio

18

les bibliothèques disponibles

- Pour consulter les bibliothèques installées sur votre machine cliquez sur **aide** -> **aide HTML**, puis, dans la page **html** : **Packages**
(sous Gnu/Linux tapez : **help.start()** dans la console R)
- Pour consulter les bibliothèques disponibles sur internet rendez-vous sur le site : <http://lib.stat.cmu.edu/R/CRAN/> dans le menu de gauche, cliquez sur **Packages**

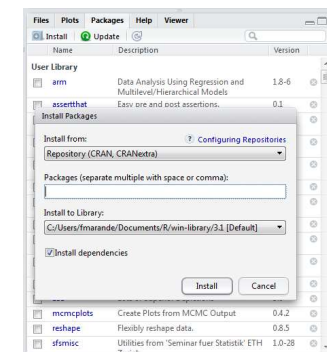
19

Installer une bibliothèques de fonctions

- Vous avez une connexion internet, c'est le cas le plus simple. On peut utiliser le menu de R :

package -> installer le(s) package(s)
puis suivez les instructions

- Vous n'avez pas de connexion internet :
vous installerez une bibliothèque à partir d'un fichier local (sur CD-Rom, extension .zip). Attention aux dépendances !
package -> installer le(s) package(s)
depuis des fichiers zip



20

2. Introduction à RMarkdown



Création d'un document, bases du langage, insertion de code R,

...

<https://statistique-et-logiciel-r.com/guide-de-demarrage-en-r-markdown/>

21

Qu'est-ce que Rmarkdown ?

- Un générateur de **rapports dynamiques** incorporant
 - **du code R** : on peut montrer comment les analyses ont été faites, par exemple en termes de jeu de données, ou de fonctions utilisées.
 - **les sorties de ce code** : par exemple des sorties d'un modèle de régression, ou encore d'un plot ...
et
 - **des commentaires** : qui permettent, par exemple, d'ajouter une interprétation aux résultats.
- **Rapport dynamique** : rapport qui
 - se **génère automatiquement à partir d'un fichier** écrit dans un certain format ici R markdown (.Rmd). Divers formats possibles : html, pdf ou docx
 - assure un travail reproductible
- **Format « .Rmd »** : format qui contient des **balises**, un peu comme du « html », mais en plus simple, et qui permet de concevoir un rapport dynamique.

22

Création d'un document RMarkdown

1. **Créer un dossier** sous windows à l'emplacement souhaité, puis dans R Studio :

File -> New Project -> Existing Directory -> Indiquez l'emplacement -> Create Project.

2. **Installer et charger les packages :**
"rmarkdown", "markdown" et "knitr"

```
1 install.packages("knitr")
2 install.packages("rmarkdown")
3 install.packages("markdown")
4
5 library(knitr)
6 library(rmarkdown)
7 library(markdown)
```

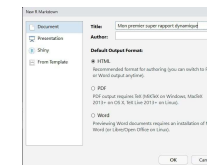
3. **Générer un premier rapport dynamique**

23

Création d'un premier rapport dynamique

1. **Création d'un nouveau fichier R markdown et enregistrement :**
File -> New File -> Rmarkdown

- Laissez toutes les options par défaut (document et html), et remplissez le champs "Title", ce titre correspondra au titre du document, il peut être long.



- Un fichier Rmd apparaît ensuite dans la fenêtre d'édition. Ce fichier est un exemple, il comporte du code et du texte, ainsi que le titre donné.
- Enregistrez le, en lui donnant un nom de fichier cette fois-ci ("RappDyn1" par exemple)

24

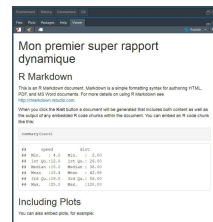
Création d'un premier rapport dynamique

2. Lancement de la génération

- cliquez sur le bouton avec la pelote de laine et les aiguilles : « knit »



- Un fichier html portant le même nom que le fichier .Rmd est alors créé dans le **working directory** (là où vous avez créé votre projet R), et en même temps le **rapport généré s'ouvre dans le viewer de R Studio** :



25

Création d'un premier rapport dynamique

3. Choix du format de sortie : html, word, pdf

- cliquez sur la flèche vers le bas, du bouton knit vous pourrez **changer le format de sortie** du rapport dynamique.
- format par défaut celui choisi à l'étape précédente, ie html, ici.
 - Vous pourrez choisir Word ou pdf.
 - Pour obtenir votre rapport au format pdf, vous devez d'abord télécharger MikTeX et lors de l'installation, choisir "install missing packages". Vous pouvez aussi choisir cette option après l'installation en allant dans "settings".

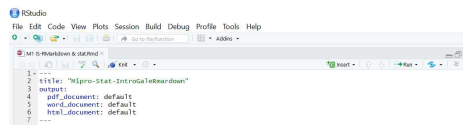
4. Comment ça marche ? La production d'un rapport se réalise en deux étapes :

- Création du fichier .Rmd contenant des blocs de code R (que l'on appelle « chunk ») et du texte. Le package **knitr** va
 - exécuter les codes R afin d'obtenir les sorties, et
 - créer un document au format markdown (.Rmd) qui contient alors les codes R, les résultats (ou les sorties), et les textes de commentaires.
- Conversion du fichier .Rmd vers le format souhaité ("html", "docx" ou encore "pdf"), par le package **markdown**.

26

Bases du langage : organisation du fichier Rmd

• L'en-tête



- contenu entre **deux séries de pointillés**.
- Par défaut, contient deux types d'éléments : le **titre du document**, et son **format de sortie**.
- possible d'**ajouter d'autres éléments**: auteur, date de création, ..., une table des matières, un lien vers un fichier de références bibliographiques ... cf https://statistique-et-logiciel-r.com/comment-inserer-des-references-bibliographiques-gerees-sous-mendeley-dans-un-document-rmarkdown_v2/

27

Bases du langage : organisation du fichier Rmd

• Les chunks contenant le code R :

```
```{r cars}
summary(cars)
```
```

- Les parties de code R sont contenues dans des blocs, appelés « chunks ». Ces chunks commencent et finissent par les balises `
- Entre les deux, se trouve **une accolade contenant la lettre r**. C'est dans cette accolade, après la lettre **r** (à ne pas l'enlever) que les **options** vont pouvoir être passées, pour choisir de faire apparaître, ou non, le code dans le rapport dynamique, ainsi que les résultats, ou encore pour définir la taille des plots.
- les chunks peuvent être nommés. Dans l'exemple précédent, le nom du chunk est **cars**.

28

Bases du langage : organisation du fichier Rmd

• Le chunk de set up

```
11 # (r setup, include=FALSE)
12 knitr::opts_chunk$set(echo = TRUE)
13
14
```

Ce chunk se trouve en dessous de l'en-tête, il permet de régler les options par défaut de tous les chunks.

Exemple : indiquer que l'on ne veut pas garder les messages et les warnings générés lors de l'exécution des chunk. Au lieu de le faire dans chaque chunk, on peut le faire une seule fois ici.

```
10 # (r setup, include=FALSE, message=FALSE, warning=FALSE)
11 knitr::opts_chunk$set(echo = TRUE)
12
13
```

Et si, pour un chunk donné, on veut faire apparaître les warnings et les messages, on utilisera `message=TRUE` et `warnings=TRUE` dans l'accolade du chunk concerné.

29

Bases du langage : organisation du fichier Rmd

• Les parties texte

```
15 ## R Markdown
16
17 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS
18 word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
19
20 when you click the "knit" button a document will be generated that includes both content as well as
21 the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
```

- Elles peuvent être insérées partout en **dehors des chunks**.
- Il est possible de mettre en gras, ou en italique, certaines parties du texte.

- Un aide-mémoire des fonctions principales : dans Rstudio via Help -> Cheatsheets -> R Markdown Cheat Sheet

30

Bases du langage : éléments pour débiter

• Titre, liste, gras, italique

Pour faire des titres, des listes, des puces, mettre en gras et en italique :

```
15 ## R Markdown
16
17 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS
18 word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
19
20 #> #> de gros titre
21 #> #> 1 un plus petit titre
22 #> #> 1.1 un titre encore plus petit
23
24 Pour faire une liste:
25 1. premier point
26 2. 2ème point
27
28 Pour mettre des puces:
29 • premier élément
30   ◦ avec son premier sous élément
31   ◦ avec son deuxième sous élément
32 • deuxième élément
33   ◦ avec ça
34   ◦ et ça
35
36 Pour mettre des puces:
37 * premier élément
38   + avec son premier sous élément
39   + avec son deuxième sous élément
40 * deuxième élément
41   + avec ça
42   + et ça
43
44 Le chiffre "trois" est en gras et le chiffre "deux" est en italique
45
46
```

1. Un gros Titre

1.1 Un plus petit Titre

1.1.1 Un titre encore plus petit

Pour faire une liste:

1. premier point
2. 2ème point

Pour mettre des puces:

- premier élément
 - avec son premier sous élément
 - avec son deuxième sous élément
- deuxième élément
 - avec ça
 - et ça

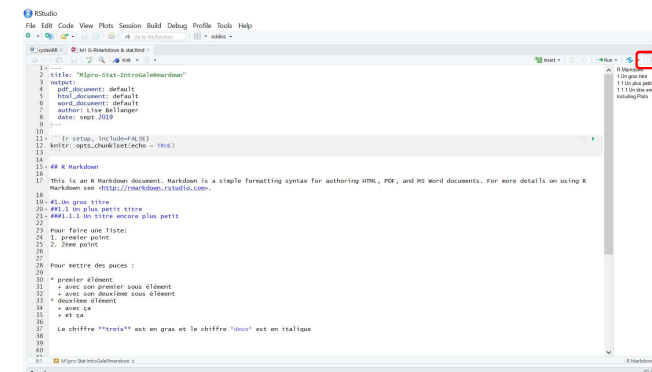
Le chiffre trois est en gras, et le chiffre deux est en italique.

Le rendu

31

Bases du langage : éléments pour débiter

La table des matières du document peut être visualisée en cliquant sur le bouton suivant :



32

Bases du langage : éléments pour débiter

• Gestion du code

- Affichage du code

Par défaut, le code R est affiché lors de la génération du rapport, cela correspond à l'option `echo=TRUE`.

Pour supprimer l'affichage du code, il faut utiliser l'argument `echo=FALSE`.

```

41
42
43 When you click the "knit" button a document will be generated that includes both content as well as the output of any embedded R code chunks within
44 the document. You can embed an R code chunk like this:
45 ```{r cars}
46 summary(cars)
47
48 le code n'est pas affiché; mais le résultat l'est
49 ```{r, echo=FALSE}
50 summary(cars)
51
52
53
54

```

Regarder le rendu en cliquant sur le bouton « knit »

33

Bases du langage : éléments pour débiter

• Gestion du code

- Affichage des résultats du code

Par défaut, le résultat du code est affiché lors de la génération du rapport, cela correspond à l'option `results="markup"`.

Pour le supprimer, il faut utiliser l'argument `results="hide"`.

```

41
42
43 When you click the "knit" button a document will be generated that includes both content as well as the output of any embedded R code chunks within
44 the document. You can embed an R code chunk like this:
45 ```{r cars}
46 summary(cars)
47
48 le code n'est pas affiché; mais le résultat l'est
49 ```{r, echo=FALSE}
50 summary(cars)
51
52 le code est affiché mais pas le résultat :
53 ```{r, results="hide"}
54 summary(cars)
55
56
57
58

```

Regarder le rendu en cliquant sur le bouton « knit »

34

Bases du langage : éléments pour débiter

• Gestion du code

- Gérer la taille des figures



35

Bases du langage : éléments pour débiter

• Insérer une image

Pour insérer une image, il suffit de placer l'image dans le working directory et d'utiliser, en dehors d'un chunk, la commande suivante :

```

```

• Créer une table de données

Pour cela, on utilise, dans un chunk, la fonction `kable` du package `knitr`.

Ici, une sortie sans le formatage :

Sous RMarkdown:

```
## insérer une table
```

```
```${x}``
head(iris)
```\n`
```

Sous R, pour obtenir une table :

```
library(knitr)
kable(head(iris))
```

36

Remarque

- Utiliser le R markdown pour autre chose:
 - pour réaliser des diapositives de présentation (avec le package slidify),
 - des tableaux de bord (avec shiny),
 - du contenu pour un site web (package blogdown), ou encore
 - pour écrire un article une publication ou un livre (package bookdown).

37

Pour aller plus loin

- [R Markdown Cheat Sheet](#)
- [R Markdown Reference Guide](#)
téléchargeable directement depuis l'onglet Help → Cheatsheet
- [l'add on remedy](#), développé par ThinkR, qui s'intègre à R Studio pour simplifier l'écriture des fichiers Rmd
- le livre "[R Markdown: The Definitive Guide](#)", ou en version papier:
- ?????

38

3. Les bases du langage

Conventions, vecteurs,
listes, matrices,
opérations, "data frames", ...

39

Quelques conventions de langage

`x, X, z1, Z1, Y.star`

Noms possibles pour les variables
Attention: `x` et `X` sont deux objets
différents

`c, t, q, s, I, TRUE, FALSE,`
`pi, ...`
`1Z`

Noms réservés, ne pas les utiliser !
Interdit !

`1-pi+exp(1.7)`
[1] 3.332355

Evaluation d'une expression

40

Prise en main

- Taper **Ctrl L** pour nettoyer la **fenêtre « Rconsole »**
- Faire des opérations : `5+9` ; `10^2` ; `2^0.5` ; `sqrt(2)`
- Utiliser la flèche (clavier) **↑** pour faire défiler les commandes déjà tapées
- La souris permet de sélectionner (de copier-coller) les lignes dans « **Rconsole** »
- Utiliser les parenthèses
 - `4+9*2-1 = 21`
 - `(4+9)*2-1 = 25`
 - `(4+9)*(2-1) = 13`

41

Stocker les résultats dans des variables

- On construit une flèche avec **< et - ;** ou **=** :
 - `a<-10`
 - `b<-3`
 - `c<-a+b` ou `a+b->c`
- Afficher le contenu de la variable **c** :
 - `c`
 - `[1] 13`
- L'ensemble peut s'écrire :
 - `a<-10 ; b<-3 ; c<-a+b ; c`

42

Quelques opérations

- **Arithmétique**
 - `+` ; `-` ; `*` ; `/` ; `^` (puissance)
- **Logique**
 - `>` ; `<` ; `<=` ; `>=` ;
 - `==` (égal) ; `!=` (différent) ;
 - `&` (et) ; `|` (ou) ; `!` (non) ; `xor` (ou exclusif)
- **exemple**
 - `8^(1/3)` = racine cubique de 8
- **Attention aux parenthèses !**
 - `8^1/3` = 2.666667
 - `8^(1/3)` = 2

43

Lecture de données (données d'exemple)

- R est fourni avec des fichiers de données d'exemple
liste des fichiers disponibles : `data()`
- Charger en mémoire le tableau de données "iris"
`data(iris)`
- Que contient ce tableau ?
`iris` ou `head(iris)`
- Un résumé numérique
`summary(iris)`
- Une présentation graphique
`pairs(iris)`
- En savoir plus sur ce tableau de données
`?iris`

44

Lecture de données (format binaire R)

- On peut stocker des données au format R (extension `.Rdata`)
- Lecture du fichier de données R « `voit2005.Rdata` »
`load(file.choose())` ou `read.table()`
- Le fichier de données a-t-il été chargé en mémoire ?
`ls()`

```
[1] "voit2005"
```
- Sauver un tableau de données
 Un fichier de données `.Rdata` peut contenir plusieurs `data.frames`
`save(iris, voit2005, file= "test.Rdata")`
`dir(pattern=".Rdata")`

```
[1] "test.Rdata"
```

45

Objets en mémoire

- La fonction `ls()` permet de lister les objets en mémoire
`ls()`
- Plus de détails avec `ls.str()`
`ls.str()`
`print(ls.str(), max.level = 0)`
- Effacer des objets en mémoire
`rm(a,b)`
- Effacer tous les objets en mémoire ou pinceau
`rm(list=ls())`



sous Rstudio

46

Types d'objets : les classes

- **vector** : variable dans le sens général
`is.vector(x)` ; `as.vector(x)`
- **factor** : variable qualitative (facteur)
`is.factor()` ; `as.factor()`
- **array** : matrice (données du même type)
`is.matrix()` ; `as.matrix()`
- **list** : Ensemble ordonné d'objets hétérogènes
`is.list()` ; `as.list()`
- **data.frame** : jeu de données composé de vecteurs de même dimension.
`is.data.frame()` ; `as.data.frame()`

47

Types d'objets : Vecteurs

| | |
|--|---|
| <code>Mydata <- c(2,3.5,-0.2)</code> | Vecteurs (<code>c="concatenate"</code>) |
| <code>Colors <- c("Red", "Green", "Red")</code> | Vecteurs de caractères |
| <code>x1 <- 25:30</code>
<code>x1</code>
<pre>[1] 25 26 27 28 29 30</pre> | Séquences de nombres |
| <code>Colors[2]</code>
<pre>[1] "Green"</pre> | Extraction d'une composante |
| <code>x1[3:5]</code>
<pre>[1] 27 28 29</pre> | Plusieurs composantes |

48

Vecteurs: Opérations sur les composantes

```
Mydata
[1] 2 3.5 -0.2

Mydata > 0
[1] TRUE TRUE FALSE

Mydata[Mydata>0]
[1] 2 3.5          Extrait les valeurs positives

Mydata[-c(1,3)]
[1] 3.5           Enlever des composantes
```

49

Quelques opérations sur des vecteurs

```
x <- c(5,-2,3,-7)
y <- c(1,2,3,4)*10
y
[1] 10 20 30 40

sort(x)
[1] -7 -2 3 5      Ordonner un vecteur

order(x)
[1] 4 2 3 1        L'ordre des composantes pour ordonner un vecteur

y[order(x)]
[1] 40 20 30 10    Opération sur toutes les composantes

rev(x)
[1] -7 3 -2 5       Ordre inverse d'un vecteur
```

50

Types d'objets : Matrices

```
x <- c(3,-1,2,0,-3,6)
x.mat <- matrix(x,ncol=2) # Crée une matrice avec 2 colonnes
                             (par défaut : création par colonnes)
      [,1] [,2]
[1,]    3    0
[2,]   -1   -3
[3,]    2    6

x.mat <- matrix(x,ncol=2, byrow=T) # Création par lignes
x.mat
      [,1] [,2]
[1,]    3   -1
[2,]    2    0
[3,]   -3    6
```

51

Extraction de composantes d'une matrice

```
x.mat[,2]          2ème colonne
[1] -1 0 6

x.mat[c(1,3),]     1ère et 3ème ligne
      [,1] [,2]
[1,]    3   -1
[2,]   -3    6

x.mat[-2,]         Sans 2ème ligne
      [,1] [,2]
[1,]    3   -1
[2,]   -3    6
```

52

Opérations matricielles I

```
dim(x.mat)           Dimension d'une matrice
[1] 3 2
t(x.mat)             Transposée d'une matrice
      [,1] [,2] [,3]
[1,]    3    2   -3
[2,]   -1    0    6

x.mat %*% t(x.mat)    Multiplication matricielle
      [,1] [,2] [,3]
[1,]   10    6  -15
[2,]    6    4   -6
[3,]  -15   -6   45
```

53

Opérations matricielles II

Autres fonctions utiles:

```
solve()              Inverse d'une matrice carrée
eigen()              Valeurs et vecteurs propres d'une matrice carrée

cbind(x.mat,x.mat)
      [,1] [,2] [,3] [,4]
[1,]    3   -1    3   -1
[2,]    2    0    2    0
[3,]   -3    6   -3    6

rbind(x.mat,x.mat[1,])
      [,1] [,2]
[1,]    3   -1
[2,]    2    0
[3,]   -3    6
[4,]    3   -1
```

54

Utiliser un data.frame (1)

Les `data.frame` ressemblent à des matrices, mais sont beaucoup plus flexibles. Ils sont utilisés dans la plupart des techniques statistiques à disposition dans R.

- Connaître le nom des variables du data.frame
`names(voit2005)`
- Les dimensions du data.frame [lignes , Colonnes]
`dim(voit2005)`
`dim(voit2005)[1]` # nombre de lignes
`dim(voit2005)[2]` # nombre de colonnes
- Le nombre de colonnes
`length(voit2005)`

55

Utiliser un data.frame (2)

Il existe plusieurs façons d'accéder aux variables du data.frame

- En précisant le nom du data.frame pour chaque variable
`plot(voit2005$Longueur, voit2005$Largeur)`
- Par leur numéro (voir `names(voit2005)`)
`plot(voit2005[,3], voit2005[,4])`
- En attachant le data.frame (*uniquement pour la lecture*)
`attach(voit2005)`
`plot(Longueur, Largeur)`
`detach()`

56

Extraire des données d'un data.frame (indexation)

- Les 5 premières lignes avec les variables 2 à 5
`voit2005[1:5, 2:5]`
- Les 5 premières lignes mais avec les variables 1, 3 et 6
`voit2005[1:5, c(1,3,6)]`
- Toutes les lignes (sauf la 3ème), toutes les variables (sauf la 1ère)
`voit2005[-3, -1]`
- Les 5 premières lignes, toutes les variables sauf les n° 1, 3, et 5
`voit2005[1:5, c(-1,-3,-5)]`
- On conserve les individus pour lesquels la puissance > 10
`voit2005[voit2005$Puissance>10,]`

57

Extraire des données d'un data.frame (fonction `subset()`)

- Ne conserver que les variables "Puissance" et "Vitesse" et que les véhicules dont la vitesse maxi est supérieure à 200 km/h

```
subset(voit2005, Vitesse > 200, select = c(Puissance, Vitesse))
```

| | Puissance | Vitesse |
|----------------------|-----------|---------|
| Alfa-Romeo 155 2.0 | 10 | 205 |
| Alfa-Romeo 164 2.5 T | 7 | 202 |
| BMW 730i | 16 | 222 |
| Citroen XM 2.0i | 11 | 201 |
| Citroen XM V6 | 16 | 222 |
| Ford Scorpio 2900i | 15 | 201 |
| Peugeot 605 Sv24 | 16 | 235 |

Exercice : en utilisant la fonction `subset()`, sélectionnez les véhicules dont la consommation est inférieure à 6l/100 et la puissance fiscale égale = 4 CV.

58

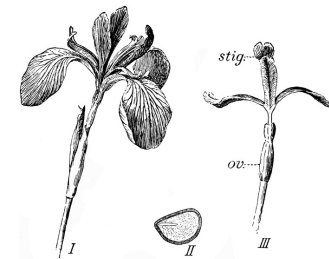
Identifier les lignes du tableau de données

- Toutes les lignes d'un data.frame ont un identificateur unique
`row.names(voit2005)`
- On peut accéder à:
 - un individu en particulier
`voit2005["Renault 21 Prima TD",]`
- ou
- une collection d'individus
`voit2005[c("Renault 21 Prima TD", "BMW 518i"), 1:3]`

59

Exercice

- Représentez graphiquement l'ensemble des relations x-y des données quantitatives du data.frame `iris`
- Extraire les données de la variété `virginica`. Quelles sont les moyennes des variables pour cette variété (fonction `mean()`)



60

Connaître le type de données

- Une classe d'objets peut être composée de données de différents types

numérique, caractère, entier, réel, logique

- Utiliser la fonction `typeof()` ou la fonction `str()`

```
a<-"inra"
typeof(a)
[1] "character"
a<-4
typeof(a)
[1] "double"
a<-as.integer(a)
typeof(a)
[1] "integer"
```

61

Types de données dans un vecteur

```
is.numeric(iris$Petal.Length)
[1] TRUE
# test sur toutes les variables : sapply() retourne une liste
sapply(iris, is.numeric)
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
TRUE TRUE TRUE TRUE FALSE
# numéro des variables numériques
which(sapply(iris, is.numeric))
Sepal.Length Sepal.Width Petal.Length Petal.Width
1 2 3 4
# noms des variables numériques
names(which(sapply(iris, is.numeric)))
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
# Récupération des numéros de variables
as.vector(which(sapply(iris, is.numeric)))
[1] 1 2 3 4
# Représentation graphique
numero<-which(sapply(iris, is.numeric))
pairs(iris[, numero])
```

62

Quelques fonctions utiles (I)

```
seq(2,12,by=2)
[1] 2 4 6 8 10 12
seq(4,5,length=5)
[1] 4.00 4.25 4.50 4.75 5.00
rep(4,10)
[1] 4 4 4 4 4 4 4 4 4 4
rep(1:3,1:3)
[1] 1 2 2 3 3 3
paste("V",1:5,sep="")
[1] "V1" "V2" "V3" "V4" "V5"
LETTERS[1:7]
[1] "A" "B" "C" "D" "E" "F" "G"
letters[20:26]
[1] "t" "u" "v" "w" "x" "y" "z"
```

63

Quelques fonctions utiles (II)

```
x1 <- c(2,1,4,2,1,2)
length(x1)
[1] 6
unique(x1)
[1] 2 1 4
table(x1)
1 2 4
2 3 2
```

64

La règle du "recycling"

L'unité de base de R est un vecteur, par exemple

```
x <- c(1,2,3,4,5)
```

Observez:

```
x + 2
```

```
[1] 3 4 5 6 7
```

```
y <- c(x,100,x) ; y
```

```
[1] 1 2 3 4 5 100 1 2 3 4 5
```

```
x+y
```

```
[1] 2 4 6 8 10 101 3 5 7 9 6
```

Warning message:

```
longer object length
```

```
is not a multiple of shorter object length in: x + y
```

Le vecteur **x** a été **répété 2.2 fois** pour donner la même longueur que **y**.
 ↳ R donne un message d'avertissement !

Ne les ignorez pas !

65

Les opérations mathématiques

- Opérations usuelles : `+` `-` `*` `/`
 - Puissances : `2^5` ou bien `2**5`
 - Fonctions standards :
`abs()`, `sign()`, `log()`, `log10()`, `sqrt()`,
`exp()`, `sin()`, `cos()`, `tan()`
`gamma()`, `lgamma()`, `choose()`
 - Pour arrondir :
 - `round(x,3)` arrondi à 3 chiffres après la virgule
- et aussi :
- `floor(2.5)` donne 2,
 - `ceiling(2.5)` donne 3

66

Opérations sur des vecteurs

```
vec <- c(5,4,6,11,14,19)
```

```
sum(vec)
```

```
[1] 59
```

```
prod(vec)
```

```
[1] 351120
```

```
mean(vec)
```

```
[1] 9.833333
```

```
median(vec)
```

```
[1] 8.5
```

```
var(vec)
```

```
[1] 34.96667
```

```
sd(vec)
```

```
[1] 5.913262
```

```
summary(vec)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.   Max.
 4.000  5.250   8.500   9.833  13.250  19.000
```

Et aussi : `min()` `max()`
`cummin()` `cummax()`
`range()`

67

Des fonctions logiques

R contient deux valeurs logiques: **TRUE** (ou **T**) et **FALSE** (ou **F**).

Exemple:

```
3 == 4
```

```
[1] FALSE
```

```
4 > 3
```

```
[1] TRUE
```

```
x <- -4:3
```

```
x > 1
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE
```

```
sum(x[x>1])
```

```
[1] 5
```

```
sum(x>1)
```

```
[1] 2
```

} Notez la différence !

| | |
|--------------------|--------------------|
| <code>==</code> | exactement égal |
| <code><</code> | plus petit |
| <code>></code> | plus grand |
| <code><=</code> | plus petit ou égal |
| <code>>=</code> | plus grand ou égal |
| <code>!=</code> | différent |
| <code>&</code> | "et" ("and") |
| <code> </code> | "ou" ("or") |

68

Valeurs manquantes: NA

Le symbole **NA** ("non-available") est utilisé pour désigner des valeurs manquantes :

```
x <- c(3.5,-1,4,6,-2,6,2)
x[c(3,6)] <- NA ; x
[1] 3.5 -1.0 NA 6.0 -2.0 NA 2.0
is.na(x)
[1] FALSE FALSE TRUE FALSE FALSE TRUE FALSE
x.sans.NA <- x[!is.na(x)]
x.sans.NA
[1] 3.5 -1.0 6.0 -2.0 2.0
mean(x)
[1] NA
mean(x,na.rm=TRUE)
[1] 1.7
c(1:length(x)) [is.na(x)]
[1] 3 6
na.omit(x) # Supprimer les données manquantes de tout le tableau
```

69

Importer les données d'un fichier texte

- Lire un fichier de données texte avec données manquantes codées **M**, le séparateur décimal est une virgule (fichier **bledur.txt**)

```
bledur <- read.table("~/bledur.txt", header=T, na.string="M", dec=",", sep=" ")
```

- Si vous ne connaissez pas le format du fichier texte des données :

```
file.show(file.choose()) # affichez le contenu d'un fichier texte
```

- Plus simplement, de façon interactive :

```
bledur <- read.table(file.choose(), header=T, na.string="M", dec=",", sep=" ")
```

70

Variables du fichier exemple **bledur**

- Ces données sont extraites d'une enquête agronomique

Numero identifiant de la parcelle

RDT rendement en grains à la récolte

PLM nombre de plantes levées par m²

ZON zone géographique

ARG taux d'argile dans le sol

LIM taux de limon dans le sol

SAB taux de sable dans le sol

VRT codes des 6 variétés cultivées

PGM poids de 1000 grains à la récolte

MST matière sèche totale à la récolte (aérien + racinaire)

AZP taux d'azote dans la plante

VRTC variétés cultivées après regroupement en 3 classes

71

Exercices

En utilisant l'éditeur de script de R :

- Importez le fichier texte **voit2005.txt**
 - Combien de lignes et de variables possède-t-il ?
- Quelles sont les moyennes des variables Longueur, Largeur et Surface du data.frame **voit2005** ?
- Combien de véhicules ont une cylindrée plus petite que 1000 cm³ ?
- Quel est l'écart-type de la variable « Longueur » ?
- Quelle est sa médiane, sa moyenne, son IQR ?

72