

Initiation aux logiciels R et RStudio

Pour l'analyse de données et les représentations graphiques

Benoît Simon-Bouhet

Conservatoire Botanique National

9, 10 et 11 décembre 2024

1^{er} contact : R et R

1. Les logiciels : Téléchargement, installation



1. Les logiciels : Téléchargement, installation



1. Les logiciels : Téléchargement, installation



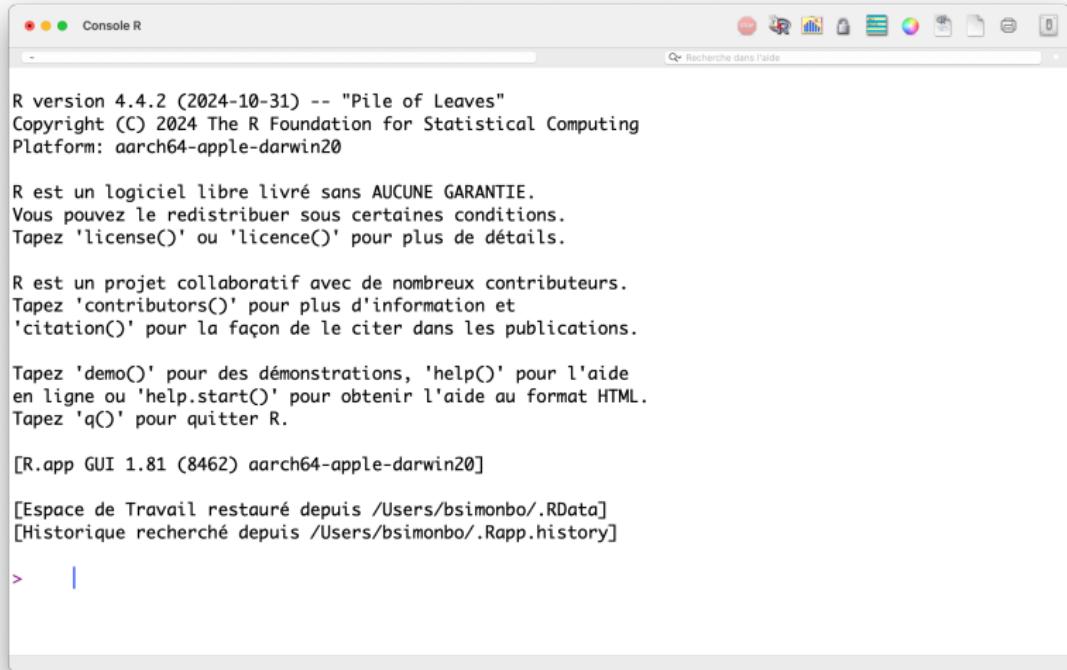
→ <https://cran.r-project.org>



→ <https://posit.co>

2. l'interface graphique

2. L'interface de R



R version 4.4.2 (2024-10-31) -- "Pile of Leaves"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20

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

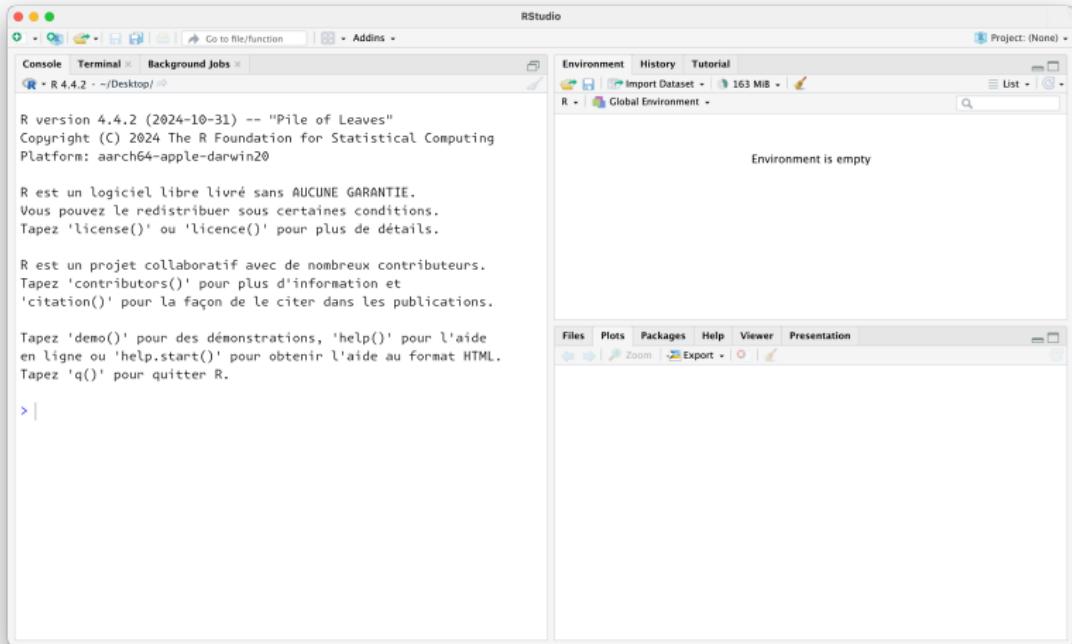
R est un projet collaboratif avec de nombreux contributeurs.
Tapez 'contributors()' pour plus d'information et
'citation()' pour la façon de le citer dans les publications.

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.

[R.app GUI 1.81 (8462) aarch64-apple-darwin20]
[Espace de Travail restauré depuis /Users/bsimonbo/.RData]
[Historique recherché depuis /Users/bsimonbo/.Rapp.history]

> |

2. L'interface de R



2. L'interface de R

Les **avantages** par rapport à R :

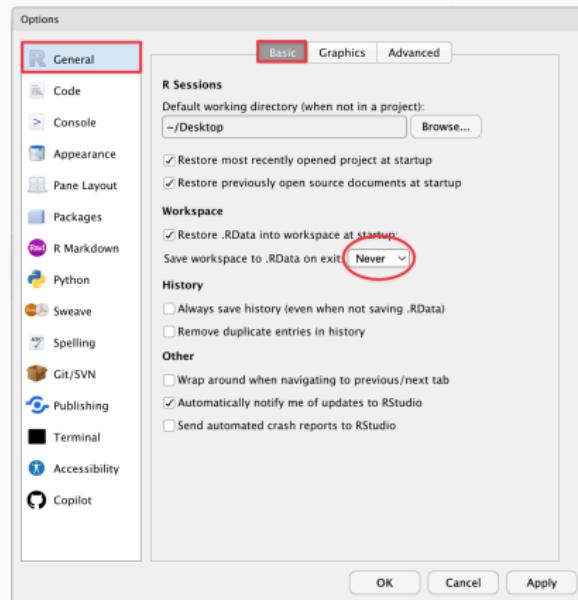
1. plus de choses sont accessibles en cliquant (installation des packages, consultation de l'aide)
2. toutes les fenêtres sont regroupées
3. l'interface est plus facile à personnaliser (agencement des fenêtres, thèmes)
4. l'éditeur de scripts est **bien supérieur** : coloration syntaxique, auto-complétion, parenthèses, arguments des fonctions, R Notebook...

Les **inconvénients** par rapport à R :

1. consomme parfois un peu plus de ressources
2. fenêtre graphiques plus "lourde"
3. plus facile de prendre de mauvaises habitudes (trop de clics possibles!)
4. consomme plus de batterie

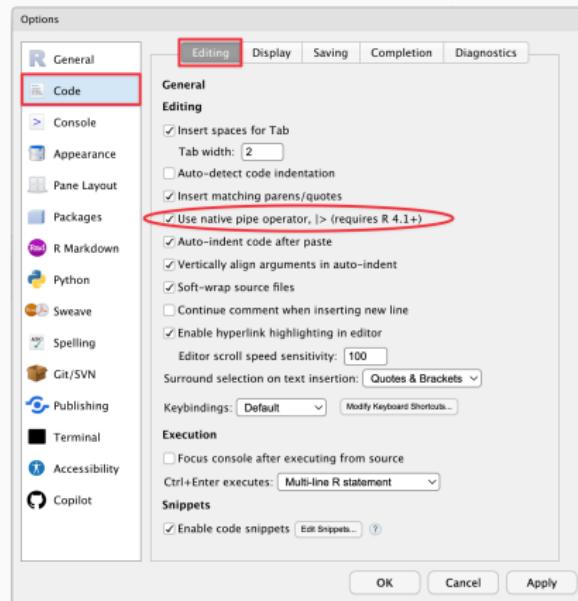
2. L'interface de R

Dans le menu Tools > Global Options..., prenez le temps de modifier ces réglages :



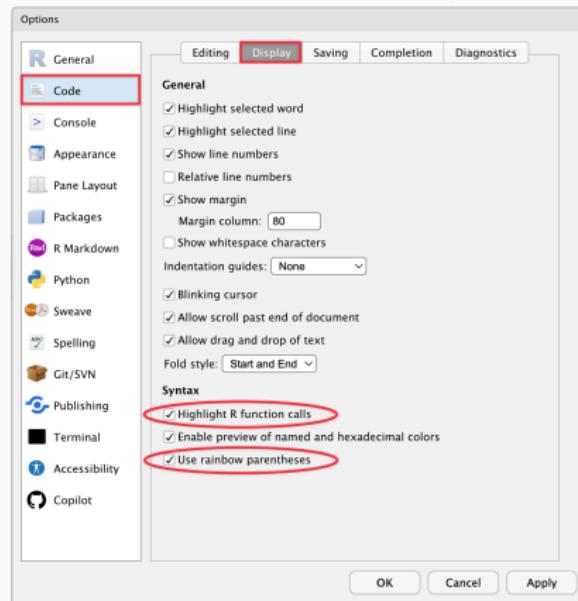
2. L'interface de R

Dans le menu Tools > Global Options..., prenez le temps de modifier ces réglages :



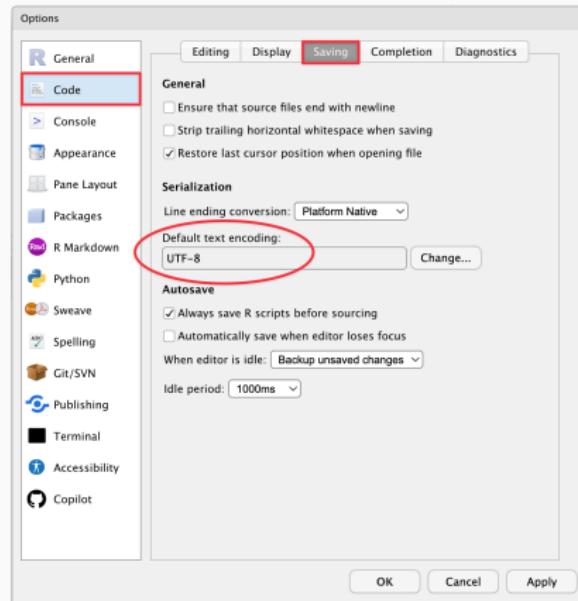
2. L'interface de R

Dans le menu Tools > Global Options..., prenez le temps de modifier ces réglages :



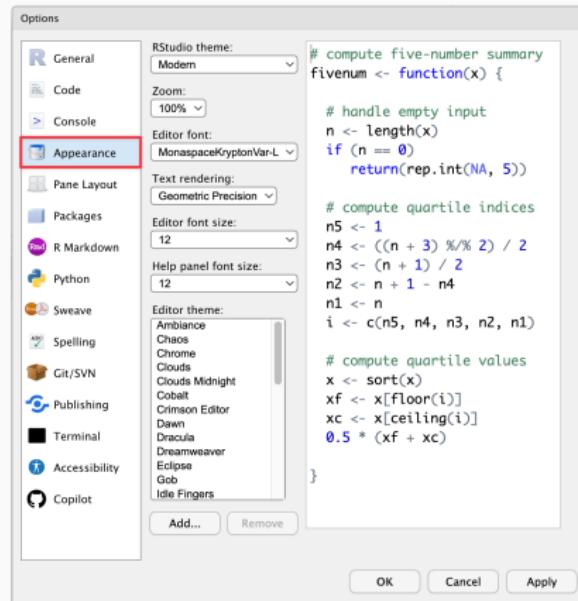
2. L'interface de R

Dans le menu Tools > Global Options..., prenez le temps de modifier ces réglages :



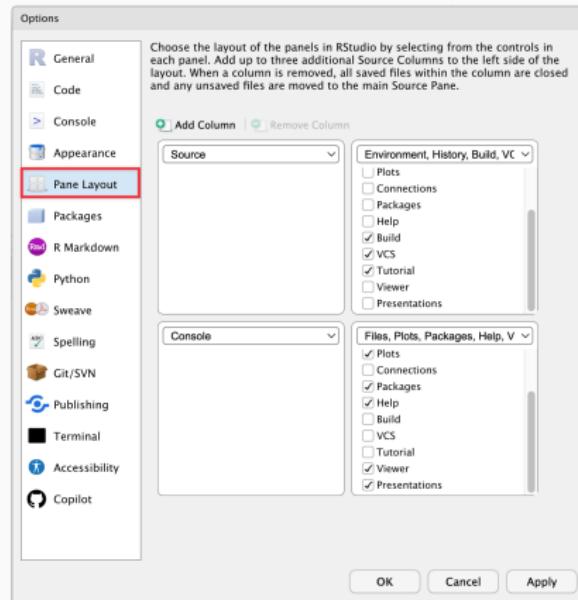
2. L'interface de R

Dans le menu Tools > Global Options..., prenez le temps de modifier ces réglages :



2. L'interface de R

Dans le menu Tools > Global Options..., prenez le temps de modifier ces réglages :



3. la `console`

3. La console

On tape des commandes puis on presse Entrée :

```
3 * 4
```

```
[1] 12
```

```
4^2 # Puissance
```

```
[1] 16
```

```
sqrt(81) # Racine carrée
```

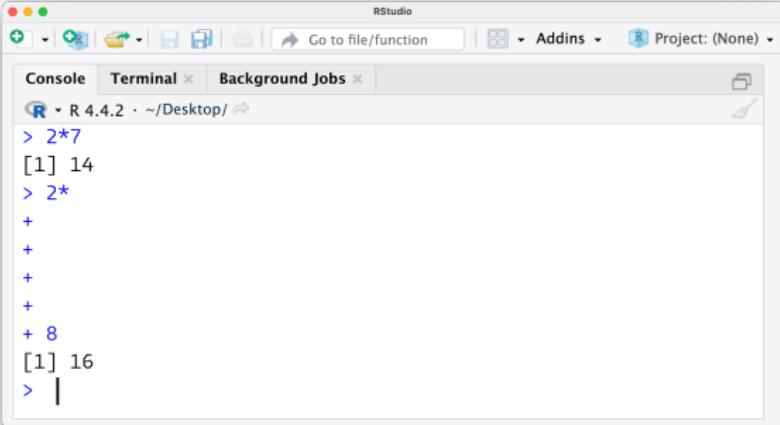
```
[1] 9
```

Le symbole “#” marque le début d'un commentaire.

3. La console

Par défaut, l'invite de commande est représentée par “>”

Si le symbole “+” apparaît à la place, c'est que la commande précédente était incomplète :



The screenshot shows the RStudio interface with the 'Console' tab selected. The window title is 'RStudio'. The console area displays the following session history:

```
R 4.4.2 · ~/Desktop/ 
> 2*7
[1] 14
> 2*
+
+
+
+
+
+
+
[1] 16
> |
```

The command `2*` was entered but left incomplete. Subsequent lines show the addition of several plus signs (+) before the final command `[1] 16`. A cursor is visible at the end of the line `> |`.

3. La console

Si  comprend la commande, il affiche le résultat sur la ligne suivante.

Sinon, il dit haut et fort ce qui ne lui plaît pas¹ :

```
log("bonjour")
```

```
Error in log("bonjour"): argument non numérique pour une fonction mathématique
```

Si  n'affiche rien lorsque vous validez, c'est que la commande tapée a été comprise, exécutée, mais que l'affichage du résultat n'a pas été demandé :

```
a <- 10
```

1. Selon la version du logiciel et la langue du système,  s'exprime en Français ou en Anglais...

4. le système d'aide

4. Trouver de l'aide

En ligne

1. Documents pdf : <https://www.r-project.org>
Documentation > Manuals > Contributed documentation
2. Forums, mailing lists, R-Help, Google : R CRAN + mots clés
3. Avant de poser des questions, fouillez² !

*“Getting **flamed** for asking dumb questions on a public mailing list is all part of growing up and being a man/woman.”*

— Michael Watson, 2004
(in a discussion on whether answers on R-help should be more polite)

2. RTFM...

4. Trouver de l'aide

En ligne dans 

1. Recherche par mot clé dans les fichiers d'aide installés

```
help.search("mot.clé")
```

```
help.search("wilcoxon")
```

2. Ouvrir un fichier d'aide : ?nom.de.la.fonction

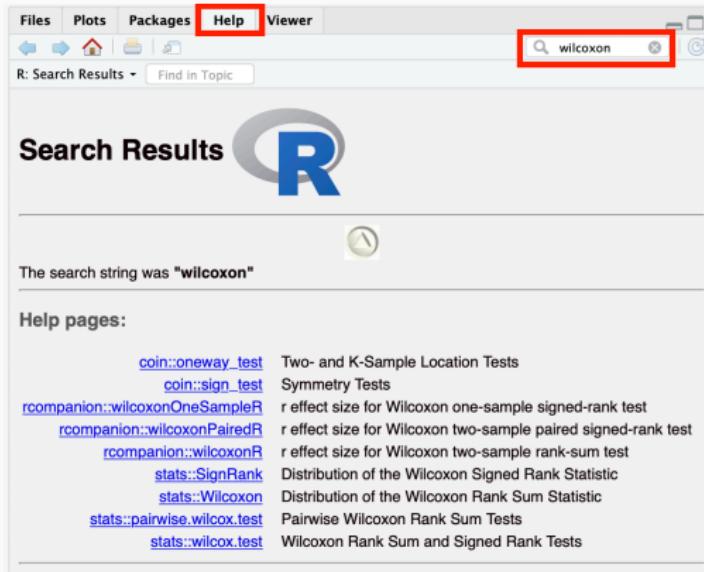
```
?wilcox.test
```

3. L'onglet "Help" de l'interface de 

4. Trouver de l'aide

En ligne dans 

L'onglet "Help" de l'interface de 



The screenshot shows the R Help interface with the "Help" tab selected. A search bar at the top right contains the text "wilcoxon". The main area displays search results for "wilcoxon".

Search Results 

The search string was "**wilcoxon**"

Help pages:

coin::oneway.test	Two- and K-Sample Location Tests
coin::sign_test	Symmetry Tests
rcompanion::wilcoxonOneSampleR	r effect size for Wilcoxon one-sample signed-rank test
rcompanion::wilcoxonPairedR	r effect size for Wilcoxon two-sample paired signed-rank test
rcompanion::wilcoxonR	r effect size for Wilcoxon two-sample rank-sum test
stats::SignRank	Distribution of the Wilcoxon Signed Rank Statistic
stats::Wilcoxon	Distribution of the Wilcoxon Rank Sum Statistic
stats::pairwise.wilcox.test	Pairwise Wilcoxon Rank Sum Tests
stats::wilcox.test	Wilcoxon Rank Sum and Signed Rank Tests

5. le répertoire de travail

5. Le répertoire de travail

Afficher le chemin du répertoire de travail (**get working directory**)

```
getwd()  
[1] "/Users/bsimonbo/TAF/Formations/Initiation_R"
```

Spécifier un nouveau répertoire de travail (**set working directory**)

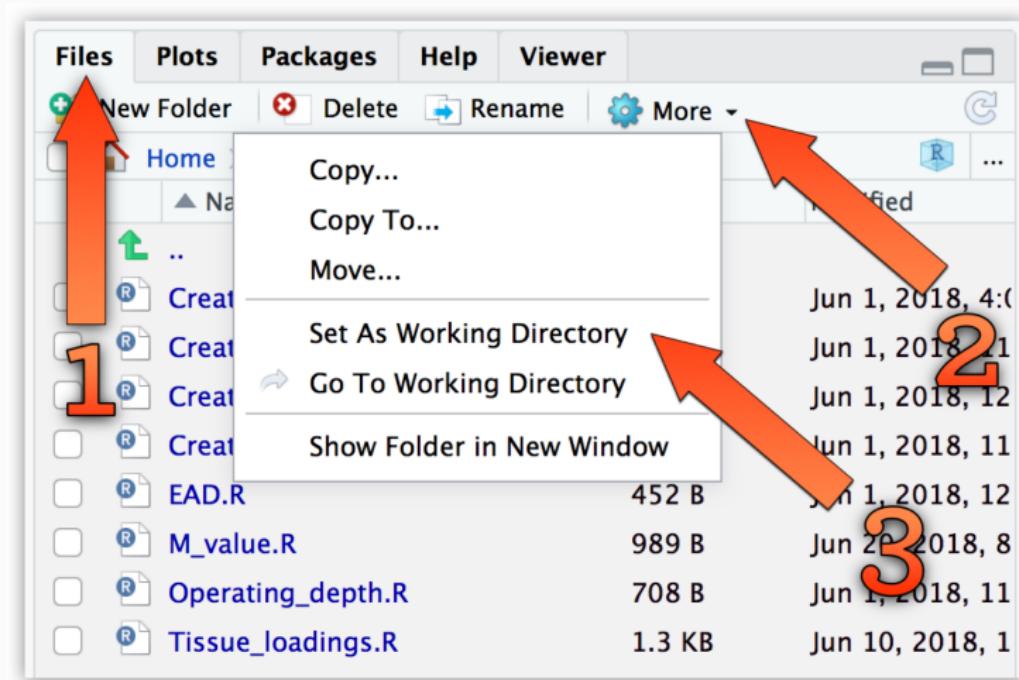
```
setwd("~/Desktop/MonDossier")
```

Ou, dans  *via* le menu Session > Set Working Directory > Choose Directory...³

Penser à changer le répertoire de travail (ou “répertoire courant”) **au début de chaque nouvelle session de travail.**

3. L'intitulé des menus peut changer selon le système d'exploitation, la langue ou la version de  utilisée.

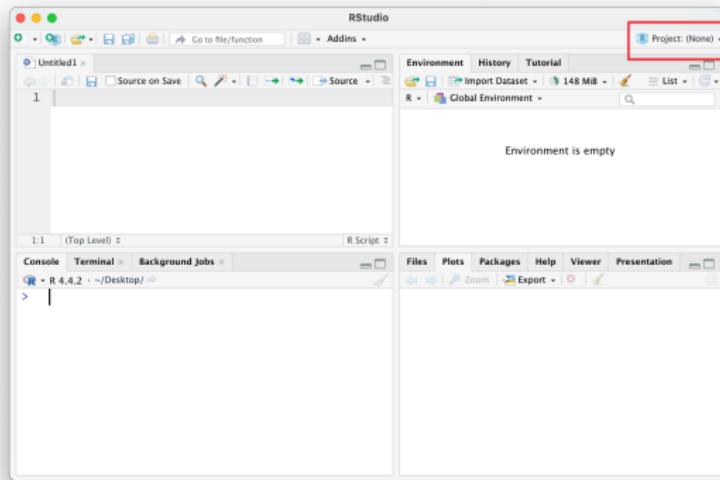
5. Le répertoire de travail



6. les Rprojects

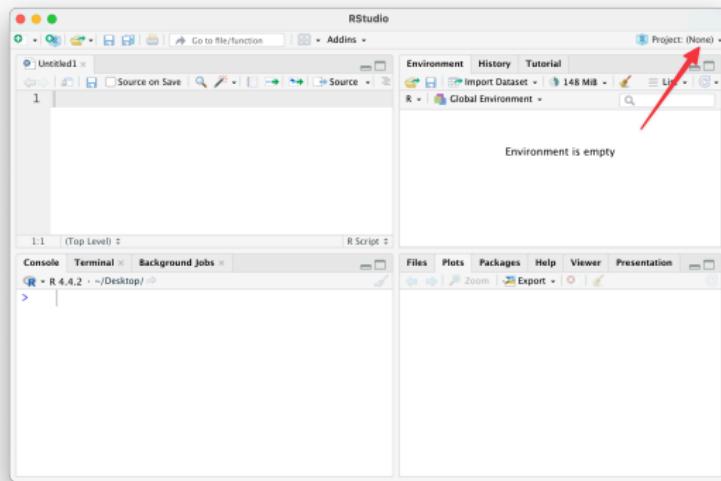
6. Les Rprojects

Une excellente façon d'**organiser** les fichiers d'un projet et d'oublier les complications liées aux répertoires de travail.



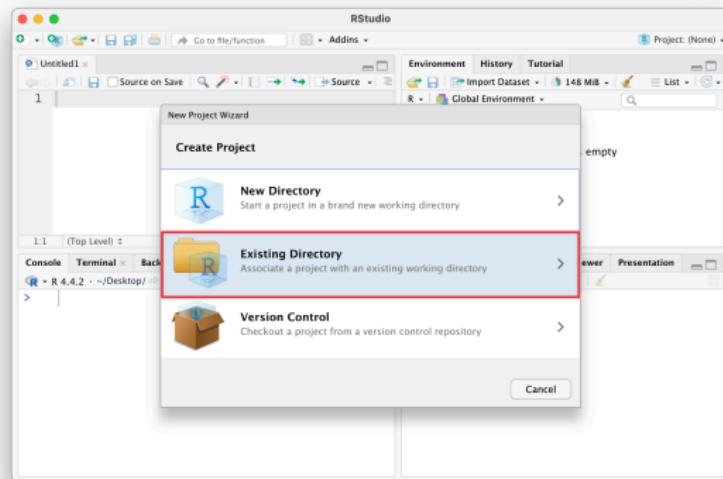
6. Les Rprojects

Une excellente façon d'**organiser** les fichiers d'un projet et d'oublier les complications liées aux répertoires de travail.



6. Les Rprojects

Une excellente façon d'**organiser** les fichiers d'un projet et d'oublier les complications liées aux répertoires de travail.



7. les packages

7. Les packages additionnels

Ce sont des extensions, des groupes de fonctions qui ajoutent des possibilités spécifiques au logiciel.⁴

À ce jour, il en existe plus de 21 000⁵ !

On peut les trouver ici :

<https://cran.r-project.org/web/packages/>

L'installation se fait directement dans R grâce à la fonction :

```
install.packages("nom.du.package")
```

4. analyses multivariées, visualisation 3D, analyses génétiques, cartographie, réseaux neuronaux, ... La liste est très longue.

5. près de 120 000 si on compte les packages d'autres dépôts (*i.e.* Bioconductor, R-forge, GitHub, voir <https://rdrr.io>)

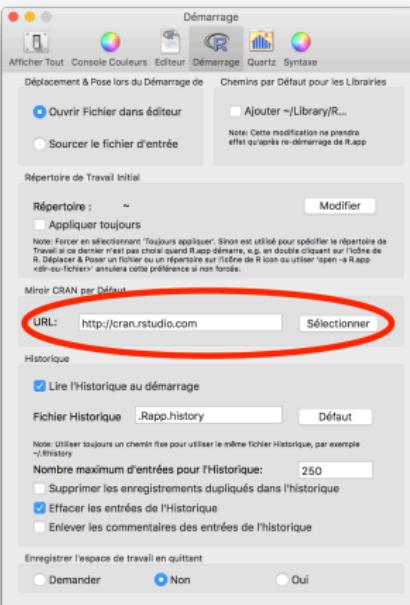
7. Les packages additionnels

La première fois que la fonction `install.packages()` est utilisée,  peut vous demander de sélectionner un site miroir :



7. Les packages additionnels

Choisir un site miroir français (Lyon 1 fonctionne très bien) ou s'en tenir au choix par défaut. Changer plus tard est toujours possible :



7. Les packages additionnels

`install.packages()` récupère les fichiers nécessaires sur internet et installe le package.

Cette opération est donc à réaliser **une seule fois** pour chaque package.

Pour utiliser le package, il faut ensuite charger ses fonctions en mémoire, **une fois par session de travail** :

```
library(nom.du.package)
```

Attention, aux **guillemets** !

- ▶ `install.packages("nom.du.package")` : guillemets obligatoires
- ▶ `library(nom.du.package)` : guillemets optionnels

7. Les packages additionnels

La liste des fonctions d'un package est accessible en tapant :

```
library(help = "nom.du.package")
```

Certains packages disposent d'une description de leur contenu accessible en tapant :

```
?nom.du.package
```

Enfin, certains packages disposent d'une ou plusieurs vignettes :

```
# Afficher la liste des vignettes d'un package  
vignette(package = "nom.du.package")  
# Afficher le contenu d'une vignette  
vignette("nom.de.la.vignette")
```

7. Les packages additionnels

The screenshot shows the RStudio interface with the 'Packages' tab selected in the top navigation bar. The main pane displays a list of packages from the User Library, including 'babynames', 'checkmate', 'cowplot', 'cranlogs', 'cumstats', 'curl', 'DeepCCR', 'dendextend', 'dunn.test', 'EDAWR', 'ellipse', 'emmmeans', 'estimability', 'factoextra', 'FactoMineR', 'flashClust', 'forecast', and 'formatR'. Each package entry includes its name, description, version, and three circular icons.

Name	Description	Version	Icons
babynames	US Baby Names 1880-2017	1.0.1	...
checkmate	Fast and Versatile Argument Checks	2.3.2	...
cowplot	Streamlined Plot Theme and Plot Annotations for 'ggplot2'	1.1.3	...
cranlogs	Download Logs from the 'RStudio' 'CRAN' Mirror	2.1.1	...
cumstats	Cumulative Descriptive Statistics	1.0	...
curl	A Modern and Flexible Web Client for R	5.2.3	...
DeepCCR	Deco Planning of CCR Deep Dives	0.1.0	...
dendextend	Extending 'dendrogram' Functionality in R	1.19.0	...
dunn.test	Dunn's Test of Multiple Comparisons Using Rank Sums	1.3.6	...
EDAWR	Expert Data Analysis with R	0.1	...
ellipse	Functions for Drawing Ellipses and Ellipse-Like Confidence Regions	0.5.0	...
emmmeans	Estimated Marginal Means, aka Least-Squares Means	1.10.5	...
estimability	Tools for Assessing Estimability of Linear Predictions	1.5.1	...
factoextra	Extract and Visualize the Results of Multivariate Data Analyses	1.0.7	...
FactoMineR	Multivariate Exploratory Data Analysis and Data Mining	2.11	...
flashClust	Implementation of optimal hierarchical clustering	1.01-2	...
forecast	Forecasting Functions for Time Series and Linear Models	8.23.0	...
formatR	Format R Code Automatically	1.14	...

7. Les packages additionnels

The screenshot shows the RStudio interface with the 'Packages' tab selected in the top navigation bar. The 'Install' button in the toolbar is highlighted with a red box. The main pane displays a list of packages from the User Library, including 'babynames', 'checkmate', 'cowplot', 'cranlogs', 'cumstats', 'curl', 'DeepCCR', 'dendextend', 'dunn.test', 'EDAWR', 'ellipse', 'emmmeans', 'estimability', 'factoextra', 'FactoMineR', 'flashClust', 'forecast', and 'formatR'. Each package entry includes its name, description, version, and three circular icons.

Name	Description	Version	Icons
babynames	US Baby Names 1880-2017	1.0.1	...
checkmate	Fast and Versatile Argument Checks	2.3.2	...
cowplot	Streamlined Plot Theme and Plot Annotations for 'ggplot2'	1.1.3	...
cranlogs	Download Logs from the 'RStudio' 'CRAN' Mirror	2.1.1	...
cumstats	Cumulative Descriptive Statistics	1.0	...
curl	A Modern and Flexible Web Client for R	5.2.3	...
DeepCCR	Deco Planning of CCR Deep Dives	0.1.0	...
dendextend	Extending 'dendrogram' Functionality in R	1.19.0	...
dunn.test	Dunn's Test of Multiple Comparisons Using Rank Sums	1.3.6	...
EDAWR	Expert Data Analysis with R	0.1	...
ellipse	Functions for Drawing Ellipses and Ellipse-Like Confidence Regions	0.5.0	...
emmmeans	Estimated Marginal Means, aka Least-Squares Means	1.10.5	...
estimability	Tools for Assessing Estimability of Linear Predictions	1.5.1	...
factoextra	Extract and Visualize the Results of Multivariate Data Analyses	1.0.7	...
FactoMineR	Multivariate Exploratory Data Analysis and Data Mining	2.11	...
flashClust	Implementation of optimal hierarchical clustering	1.01-2	...
forecast	Forecasting Functions for Time Series and Linear Models	8.23.0	...
formatR	Format R Code Automatically	1.14	...

7. Les packages additionnels

The screenshot shows the RStudio interface with the 'Packages' tab selected in the top navigation bar. The main pane displays a list of packages in the 'User Library'. A red box highlights the first column of package names. The table includes columns for Name, Description, Version, and several small circular icons.

Name	Description	Version	Icons
babynames	US Baby Names 1880-2017	1.0.1	...
checkmate	Fast and Versatile Argument Checks	2.3.2	...
cowplot	Streamlined Plot Theme and Plot Annotations for 'ggplot2'	1.1.3	...
cranlogs	Download Logs from the 'RStudio' 'CRAN' Mirror	2.1.1	...
cumstats	Cumulative Descriptive Statistics	1.0	...
curl	A Modern and Flexible Web Client for R	5.2.3	...
DeepCCR	Deco Planning of CCR Deep Dives	0.1.0	...
dendextend	Extending 'dendrogram' Functionality in R	1.19.0	...
dunn.test	Dunn's Test of Multiple Comparisons Using Rank Sums	1.3.6	...
EDAWR	Expert Data Analysis with R	0.1	...
ellipse	Functions for Drawing Ellipses and Ellipse-Like Confidence Regions	0.5.0	...
emmmeans	Estimated Marginal Means, aka Least-Squares Means	1.10.5	...
estimability	Tools for Assessing Estimability of Linear Predictions	1.5.1	...
factoextra	Extract and Visualize the Results of Multivariate Data Analyses	1.0.7	...
FactoMineR	Multivariate Exploratory Data Analysis and Data Mining	2.11	...
flashClust	Implementation of optimal hierarchical clustering	1.01-2	...
forecast	Forecasting Functions for Time Series and Linear Models	8.23.0	...
formatR	Format R Code Automatically	1.14	...

7. Les packages additionnels

The screenshot shows the RStudio interface with the 'Packages' tab selected in the top navigation bar. The 'User Library' section is highlighted with a red box, listing various R packages and their details. The packages listed are:

Name	Description	Version
babynames	US Baby Names 1880-2017	1.0.1
checkmate	Fast and Versatile Argument Checks	2.3.2
cowplot	Streamlined Plot Theme and Plot Annotations for 'ggplot2'	1.1.3
cranlogs	Download Logs from the 'RStudio' 'CRAN' Mirror	2.1.1
cumstats	Cumulative Descriptive Statistics	1.0
curl	A Modern and Flexible Web Client for R	5.2.3
DeepCCR	Deco Planning of CCR Deep Dives	0.1.0
dendextend	Extending 'dendrogram' Functionality in R	1.19.0
dunn.test	Dunn's Test of Multiple Comparisons Using Rank Sums	1.3.6
EDAWR	Expert Data Analysis with R	0.1
ellipse	Functions for Drawing Ellipses and Ellipse-Like Confidence Regions	0.5.0
emmmeans	Estimated Marginal Means, aka Least-Squares Means	1.10.5
estimability	Tools for Assessing Estimability of Linear Predictions	1.5.1
factoextra	Extract and Visualize the Results of Multivariate Data Analyses	1.0.7
FactoMineR	Multivariate Exploratory Data Analysis and Data Mining	2.11
flashClust	Implementation of optimal hierarchical clustering	1.01-2
forecast	Forecasting Functions for Time Series and Linear Models	8.23.0
formatR	Format R Code Automatically	1.14

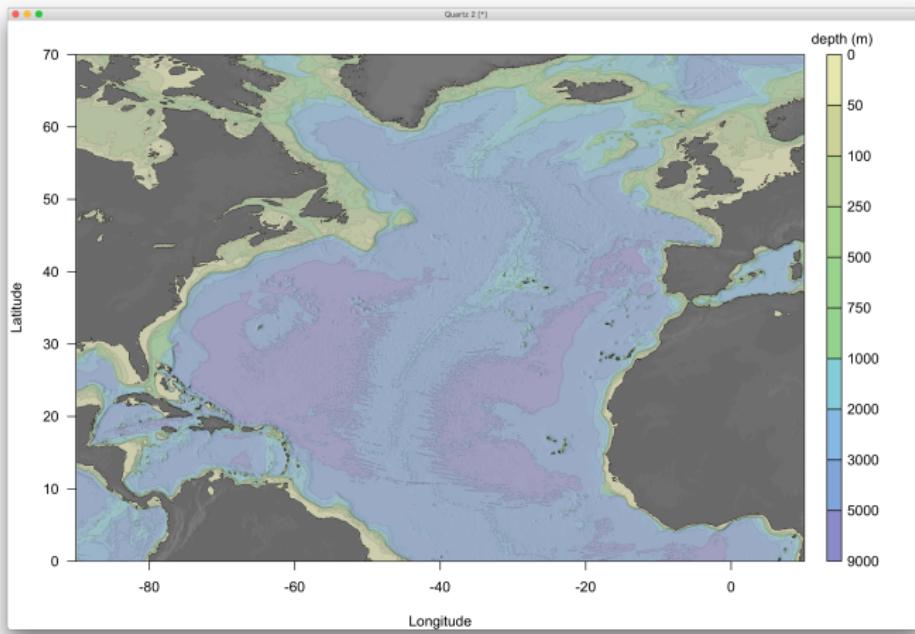
7. Les packages additionnels

The screenshot shows the RStudio interface with the 'Packages' tab selected in the top navigation bar. The main pane displays a list of packages in the 'User Library'. A vertical red box highlights the right edge of the list, where small circular icons are located next to each package entry.

Name	Description	Version
babynames	US Baby Names 1880-2017	1.0.1
checkmate	Fast and Versatile Argument Checks	2.3.2
cowplot	Streamlined Plot Theme and Plot Annotations for 'ggplot2'	1.1.3
cranlogs	Download Logs from the 'RStudio' 'CRAN' Mirror	2.1.1
cumstats	Cumulative Descriptive Statistics	1.0
curl	A Modern and Flexible Web Client for R	5.2.3
DeepCCR	Deco Planning of CCR Deep Dives	0.1.0
dendextend	Extending 'dendrogram' Functionality in R	1.19.0
dunn.test	Dunn's Test of Multiple Comparisons Using Rank Sums	1.3.6
EDAWR	Expert Data Analysis with R	0.1
ellipse	Functions for Drawing Ellipses and Ellipse-Like Confidence Regions	0.5.0
emmmeans	Estimated Marginal Means, aka Least-Squares Means	1.10.5
estimability	Tools for Assessing Estimability of Linear Predictions	1.5.1
factoextra	Extract and Visualize the Results of Multivariate Data Analyses	1.0.7
FactoMineR	Multivariate Exploratory Data Analysis and Data Mining	2.11
flashClust	Implementation of optimal hierarchical clustering	1.01-2
forecast	Forecasting Functions for Time Series and Linear Models	8.23.0
formatR	Format R Code Automatically	1.14

7. Les packages additionnels : exercice

Le package `marmap` permet de télécharger des données bathymétriques, de les analyser et de construire des cartes :



7. Les packages additionnels : exercice

1. Installez le package marmap.
2. Chargez-le en mémoire afin de pouvoir accéder à ses fonctions, ses fichiers d'aide, etc.
3. Affichez la liste de ses fonctions.
4. Affichez la liste de ses vignettes. Combien y en a-t-il?
5. Ouvrez la vignette marmap-DataAnalysis pour avoir un aperçu des possibilités offertes par le package.
6. Exécutez les exemples du fichier d'aide de la fonction `create.buffer()` en tapant :

```
example(create.buffer)
```

8. les **scripts**

8. Les scripts

Un script est un fichier **texte brut** portant l'extension “.R”.

Pour créer un script : menu File > New File > R Script, ou en pressant les touches ctrl + shift + N

Il contient :

1. des commandes qui peuvent être comprises par 
2. des commentaires, plein de commentaires⁶...

```
# Ceci est un exemple de script
# Création d'un objet contenant les entiers de 1 à 5
a <- 1:5

# affichage d'une table contenant ces entiers,
# leur carré et leur cube
data.frame(x = a, x2 = a^2, x3 = a^3)
```

6. Ne jamais sous-estimer à quel point “le futur soi” peut être **amnésique**. Comprendre ce que vous tapez aujourd’hui ne signifie pas que vous le comprendrez toujours dans 6 mois...

8. Les scripts : un exemple

Parmi les documents de la formation se trouve un fichier nommé alt.R.

Cliquez sur File > Open file..., puis naviguez jusqu'à ce fichier.

Il doit s'ouvrir dans un nouvel onglet de R, en haut à gauche. Vous pouvez envoyer chaque ligne du script dans la console en effectuant des "copier-coller".

Beaucoup plus pratique et rapide, on peut aussi presser les touches **ctrl+Entrée** pour envoyer la ligne active (ou plusieurs lignes sélectionnées) directement dans la console.

8. Les scripts : exercice

1. Créez un nouveau script et donnez-lui un nom (sauvegardez-le dans votre répertoire de travail).
2. Tapez toutes les commandes permettant d'installer marmap, de le charger en mémoire, d'afficher la liste de ses fonctions, et toutes les autres étapes vues précédemment.
3. N'oubliez pas d'ajouter autant de commentaires que nécessaire.
4. Sauvegardez régulièrement.

9. les sauvegardes

9. Que faut-il sauvegarder?

3 choses...

1. Indispensable : **vos scripts**!
2. Parfois : votre environnement de travail.
3. Rarement : l'historique des commandes.

10. Les objets dans R

i. Les **vecteurs**

Les vecteurs : la fonction `c()`

`c()` : collection d'éléments qui sont tous du même type

Vecteur numérique (d'entiers) :

```
c(2, 4, 1, 6, 7, 8)  
[1] 2 4 1 6 7 8
```

Vecteur de caractères :

```
c("rouge", "vert", "bleu", "bleu")  
[1] "rouge" "vert"   "bleu"   "bleu"
```

Vecteur logique (vrais/faux) :

```
c(TRUE, TRUE, FALSE, FALSE, T, T, F)  
[1]  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE
```

Les vecteurs : la fonction c()

c() : collection d'éléments qui sont tous du même type

Si les éléments sont incompatibles, R essaie de se débrouiller au mieux :

```
c(1, 2, 4, "rouge")  
[1] "1"      "2"      "4"      "rouge"
```

```
c(1, 0, 3, TRUE, FALSE, TRUE)  
[1] 1 0 3 1 0 1
```

```
c(1, 0, 3, TRUE, FALSE, TRUE, "vert")  
[1] "1"      "0"      "3"      "TRUE"   "FALSE"  "TRUE"   "vert"
```

Les vecteurs : la fonction `c()`

`c()` : collection d'éléments qui sont tous du même type

`c()` est une fonction.

Les fonctions possèdent (presque toutes) des arguments séparés par des virgules.

Pour ré-utiliser les résultats produits par une fonction, on leur donne un nom : on les assigne dans un objet grâce à la notation “`<-`” :

```
taille <- c(165, 162, 184, 191, 174, 174, 188, 157)
taille
[1] 165 162 184 191 174 174 188 157
mean(taille)
[1] 174.375
```

Les vecteurs : fonctions utiles

ls() et rm()

ls(), pour lister les objets créés au cours d'une session de travail :

```
ls()
[1] "a"           "make_dfs"      "taille"
[4] "theme_benoit" "thm"
```

rm(), pour supprimer (remove) des objets :

```
rm("a")
a
Error in eval(expr, envir, enclos): objet 'a' introuvable
```

```
ls()
[1] "make_dfs"      "taille"       "theme_benoit"
[4] "thm"
```

Les vecteurs : fonctions utiles

`class()`

Affiche la **classe** d'un objet :

```
x <- c(1, 4, 12, 8)
class(x)

[1] "numeric"
```

```
y <- c("Petit", "Moyen", "Grand", "Grand", "Petit")
class(y)

[1] "character"
```

```
z <- c(TRUE, TRUE, FALSE, FALSE, TRUE)
class(z)

[1] "logical"
```

Les vecteurs : fonctions utiles

Opérations classiques

Lorsqu'on dispose d'un vecteur numérique, les opérations arithmétiques classiques s'appliquent à chaque élément du vecteur :

```
x  
[1] 1 4 12 8  
  
2 * x  
[1] 2 8 24 16  
  
x^3  
[1] 1 64 1728 512  
  
log(x, base = 10)  
[1] 0.000000 0.602060 1.079181 0.903090
```

Les vecteurs : fonctions utiles

length() et dim()

Dans R, un vecteur n'a pas de dimension. Il a en revanche une longueur :

```
y  
[1] "Petit" "Moyen" "Grand" "Grand" "Petit"  
  
dim(y)  
NULL  
  
length(y)  
[1] 5
```

Comme TRUE et FALSE, NULL est un **mot réservé**.

Les vecteurs : fonctions utiles

Le recyclage

On peut additionner, soustraire, multiplier et diviser des vecteurs entre eux. Ces opérations se font **terme à terme**.

Les vecteurs doivent avoir la **même longueur**.

Sinon, R recycle le vecteur le plus court et affiche un message d'avertissement :

```
a <- c(1, 2, 3, 4)
b <- c(3, 6, 9)
a + b
```

Warning in a + b: la taille d'un objet plus long n'est pas multiple de la taille d'un objet plus court

```
[1] 4 8 12 7
```

Les vecteurs : fonctions utiles

L'opérateur ":"

Il sert essentiellement à générer des suites d'entiers, croissantes ou décroissantes :

```
croiss <- 1:10  
decroiss <- 30:17  
neg <- -4:3
```

```
croiss
```

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

```
decroiss
```

```
[1] 30 29 28 27 26 25 24 23 22 21 20 19 18 17
```

```
neg
```

```
[1] -4 -3 -2 -1 0 1 2 3
```

Les vecteurs : fonctions utiles

L'opérateur ":"

Il est aussi possible d'utiliser des nombres à virgule, mais le pas est toujours de 1 :

```
1.3:9.3
```

```
[1] 1.3 2.3 3.3 4.3 5.3 6.3 7.3 8.3 9.3
```

Si ça ne tombe pas juste, le vecteur est tronqué :

```
1.3:9.2
```

```
[1] 1.3 2.3 3.3 4.3 5.3 6.3 7.3 8.3
```

Les vecteurs : fonctions utiles

La fonction seq()

Elle permet de créer des **séquences** régulières :

```
seq(from = 0, to = 25, by = 5)
```

```
[1] 0 5 10 15 20 25
```

Comme précédemment, si ça ne tombe pas juste, le vecteur est tronqué :

```
seq(from = 0, to = 25, by = 4)
```

```
[1] 0 4 8 12 16 20 24
```

Les vecteurs : fonctions utiles

La fonction seq()

Les séquences décroissantes sont possibles également, mais attention au signe du pas :

```
seq(from = 25, to = 0, by = 5)
```

```
Error in seq.default(from = 25, to = 0, by = 5): signe incorrect de l'argument 'by'
```

```
seq(from = 25, to = 0, by = -5)
```

```
[1] 25 20 15 10 5 0
```

Les pas non entiers sont possibles également :

```
seq(from = 1, to = 3, by = 0.2)
```

```
[1] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0
```

Les vecteurs : fonctions utiles

La fonction seq()

Autre argument utile de cette fonction : length.out

```
seq(from = 0, to = 100, length.out = 11)
[1] 0 10 20 30 40 50 60 70 80 90 100
```

```
seq(from = 0, to = 100, length.out = 6)
[1] 0 20 40 60 80 100
```

À quoi sert-il? Faites des essais, consultez l'aide de la fonction...

```
?seq
```

Les vecteurs : fonctions utiles

La fonction rep()

Permet de répliquer les valeurs fournies en guise de premier argument :

```
rep(c(1, 7, 6, 7), times = 4)  
[1] 1 7 6 7 1 7 6 7 1 7 6 7 1 7 6 7
```

On peut répéter chaque élément un nombre de fois différent :

```
rep(c(1, 7, 6), times = c(4, 3, 2))  
[1] 1 1 1 1 7 7 7 6 6
```

Enfin, il est possible de combiner les arguments times et each :

```
rep(c(1, 7, 6), each = 3, times = 2)  
[1] 1 1 1 7 7 7 6 6 6 1 1 1 7 7 7 6 6 6
```

Les vecteurs

Exercices

À l'aide des fonctions `c()`, `rep()`, `seq()` et de l'opérateur “`:`”, créez les objets suivants :

```
vec1
```

```
[1] 0 6 6 12 12 12 18 18 18 18 24 24 24 24 24 24
```

```
vec2
```

```
[1] 1 1 2 2 8 8 10 10 1 1 2 2 8 8 10 10
```

```
vec3
```

```
[1] 8 8 8 8 7 7 6 6 6 5 5 5 4 4 4 4 3 3 3
```

Les vecteurs : solutions

Création du vecteur 1

```
vec1 <- rep(seq(from = 0, to = 24, by = 6), times = 1:5)
```

Création du vecteur 2

```
vec2 <- rep(c(1, 2, 8, 10), each = 2, times = 2)
```

Création du vecteur 3

```
vec3 <- rep(8:3, times = c(4, 3, 4, 3, 4, 3))
```

ou alors :

```
vec3 <- rep(8:3, times = rep(4:3, 3))
```

10. Les objets dans R

ii. Les facteurs

Les facteurs

La fonction `factor()`

Un facteur est un vecteur qui possède un **attribut supplémentaire** :

```
vec <- c(1, 4, 7, 5, 6, 4, 1, 7, 7)
vec
[1] 1 4 7 5 6 4 1 7 7
```

```
fac <- factor(vec)
fac
[1] 1 4 7 5 6 4 1 7 7
Levels: 1 4 5 6 7
```

L'attribut “`Levels`” indique quelles sont les valeurs qui peuvent être observées dans le facteur.

Les facteurs sont utilisées pour coder des **variables catégorielles** dont le nombre de catégories est fini et connu.

Les facteurs

La fonction `levels()`

Elle permet d'afficher et de modifier les niveaux d'un vecteur :

```
fac  
[1] 1 4 7 5 6 4 1 7 7  
Levels: 1 4 5 6 7  
  
levels(fac)  
[1] "1" "4" "5" "6" "7"
```

```
levels(fac) <- c("a", "b", "c", "d", "e")  
fac  
[1] a b e c d b a e e  
Levels: a b c d e
```

Les facteurs : un exemple

La fonction `factor()`

200 juges ont goûté à l'aveugle un vin de Bordeaux grand cru classé. Ils ont attribué une note de 1 à 4, 1 étant la note minimale et 4 la note maximale. Le vecteur `wine` contient les notes des 200 juges.

```
wine
```

```
[1] 3 4 4 3 3 4 2 2 3 3 3 3 3 3 3 3 3 3 2 3 2 4 4 3 2 3 3 3 3 3 3  
[28] 3 3 3 3 3 3 1 3 4 3 3 3 4 3 2 4 3 2 3 3 3 3 3 2 3 3 3 4 4 4  
[55] 3 3 4 4 2 4 3 3 4 3 3 2 2 3 3 3 3 4 4 4 3 3 4 3 3 3 3 3 3 3 3  
[82] 3 2 2 3 3 2 3 3 2 3 3 3 3 3 3 4 3 4 4 3 3 3 3 3 3 3 4 1 2  
[109] 3 4 3 3 3 4 2 2 4 3 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3 4  
[136] 3 3 3 3 3 4 4 3 3 1 4 4 3 2 3 3 3 4 3 3 3 3 3 3 2 4 2 3  
[163] 3 4 3 4 4 3 3 3 3 4 2 4 4 2 1 3 3 3 4 3 3 4 3 3 2 4 3  
[190] 3 3 3 4 4 3 1 4 3 4 3
```

Les facteurs : un exemple

La fonction `summary()`

La fonction `summary()` permet d'afficher un résumé des données :

```
summary(wine)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.000	3.000	3.000	3.055	3.000	4.000

```
class(wine)
```

```
[1] "integer"
```

`wine` est un vecteur numérique. La fonction `summary()` calcule donc la moyenne, la médiane, les valeurs extrêmes, etc.

Les facteurs : un exemple

La fonction `factor()`

Les notes attribuées correspondent en fait à des **catégories** plus qu'à une variable continue. Le résumé suivant est plus approprié :

```
wine.f <- factor(wine)
class(wine.f)
[1] "factor"
```

```
summary(wine.f)
 1   2   3   4
 5 24 126 45
```

`summary()` est une fonction **générique** : elle ne renvoie pas la même chose selon le type d'argument fourni. Pour un facteur, elle présente le nombre d'observations pour chaque valeur possible (**niveau**) du facteur.

Les facteurs : un exemple

```
levels()
```

Pour rendre les notes plus compréhensibles, on peut utiliser des niveaux non numériques :

```
levels(wine.f) <- c("Mauvais", "Moyen", "Bon", "Excellent")
head(wine.f, 20) # Affiche les 20 premières valeurs

[1] Bon      Excellent Excellent Bon      Bon
[6] Excellent Moyen     Moyen     Bon      Bon
[11] Bon      Bon       Bon       Bon      Bon
[16] Bon      Bon       Moyen    Bon      Moyen
Levels: Mauvais Moyen Bon Excellent
```

```
summary(wine.f)
```

Mauvais	Moyen	Bon	Excellent
5	24	126	45

Les facteurs : un autre exemple

Ordonner un facteur

Soit le facteur suivant :

```
mois <- c("jan", "fev", "jan", "mar", "mai", "avr")
mois.f <- factor(mois)
mois.f

[1] jan fev jan mar mai avr
Levels: avr fev jan mai mar
```

Les niveaux ne sont pas dans le bon ordre.

C'est un problème pour les graphiques. Pour y remédier :

```
mois.f <- factor(mois, levels = c("jan", "fev", "mar",
                                    "avr", "mai"))
mois.f

[1] jan fev jan mar mai avr
Levels: jan fev mar avr mai
```

10. Les objets dans R

iii. Les matrices

Les matrices

La fonction `matrix()`

Comme un vecteur, une matrice contient des éléments qui sont tous du même type.

Contrairement aux vecteurs, les matrices ont une dimension : un nombre de ligne et un nombre de colonnes.

```
mat <- matrix(1:12, ncol = 4)
mat

      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

dim(mat)
[1] 3 4
```

Les matrices

La fonction `matrix()`

La fonction `matrix()` peut prendre jusqu'à 5 arguments (consultez l'aide de la fonction).

On peut ainsi spécifier :

1. un **vecteur** contenant les coefficients de la matrice,
2. un nombre de lignes (`nrow`)...
3. ... et/ou de colonnes (`ncol`),
4. le sens de remplissage (par défaut, `byrow = FALSE`)
5. une **liste** contenant les noms de lignes et de colonnes de la matrice

Les matrices

La fonction `matrix()`

```
coef <- c(1, 3, 8, 12, 13, 14)
matrix(coef, ncol = 2)

[,1] [,2]
[1,]    1   12
[2,]    3   13
[3,]    8   14

matrix(coef, nrow = 2)

[,1] [,2] [,3]
[1,]    1    8   13
[2,]    3   12   14

matrix(coef, nrow = 2, byrow = TRUE)

[,1] [,2] [,3]
[1,]    1    3    8
[2,]   12   13   14
```

Les matrices

La fonction `matrix()`

Si les dimensions indiquées ne correspondent pas à la longueur du vecteur, R fait au mieux :

- ▶ si le vecteur est trop long, il le tronque
- ▶ si le vecteur est trop court, il recycle

```
matrix(coef, ncol = 2, nrow = 2)

Warning in matrix(coef, ncol = 2, nrow = 2): data length differs from size of
matrix: [6 != 2 x 2]

[,1] [,2]
[1,]    1    8
[2,]    3   12

matrix(coef, ncol = 3, nrow = 3)

Warning in matrix(coef, ncol = 3, nrow = 3): data length differs from size of
matrix: [6 != 3 x 3]

[,1] [,2] [,3]
[1,]    1   12    1
[2,]    3   13    3
[3,]    8   14    8
```

Les matrices

La fonction `matrix()`

Comme pour les vecteurs, on peut créer des matrices de vrais/faux et de caractères :

```
matrix(c("rouge", "bleu", "jaune", "violet"), ncol = 2)

[,1]      [,2]
[1,] "rouge"  "jaune"
[2,] "bleu"   "violet"

matrix(c(T, F, T, T, F, T), ncol = 2, byrow = TRUE)

[,1]      [,2]
[1,] TRUE FALSE
[2,] TRUE  TRUE
[3,] FALSE TRUE
```

10. Les objets dans R

iv. Les tableaux

Les tableaux de données

La fonction `data.frame()`

Dans R, la plupart des tableaux que vous manipulerez seront des `data.frame()` et non des `matrix()`.

Les variables peuvent avoir des classes différentes et sont présentées en colonnes. Exemple : création de 2 facteurs et d'un vecteur :

```
day <- factor(c("lundi", "mardi", "mercredi", "jeudi",
               "vendredi", "samedi", "dimanche"))
sun <- factor(c(TRUE, TRUE, FALSE, FALSE, FALSE, TRUE, TRUE))
wind <- c(0, 12, 32, 45, 21, 11, 14)
```

Les tableaux de données

La fonction `data.frame()`

On peut combiner ces 3 objets dans un `data.frame` :

```
meteo <- data.frame(jour = day, soleil = sun, vent = wind)
meteo

      jour soleil vent
1     lundi   TRUE    0
2     mardi   TRUE   12
3 mercredi FALSE   32
4     jeudi FALSE   45
5 vendredi FALSE   21
6     samedi  TRUE   11
7 dimanche  TRUE   14

class(meteo)

[1] "data.frame"
```

Les tableaux de données

La fonction `summary()`

Les tableaux de données que nous importerons bientôt dans  à partir de tableurs auront ce format.

La fonction `summary()` renvoie un résumé pour chaque colonne d'un `data.frame`. Le type de résumé dépend du contenu des colonnes.

```
summary(meteo)
```

	jour	soleil	vent
dimanche:1	FALSE:3	Min. : 0.00	
jeudi :1	TRUE :4	1st Qu.:11.50	
lundi :1		Median :14.00	
mardi :1		Mean :19.29	
mercredi:1		3rd Qu.:26.50	
samedi :1		Max. :45.00	
vendredi:1			

Les tableaux de données

Exercice

Sur la diapo précédente, les jours de la semaine sont classés par ordre alphabétique. Créez un data.frame nommé meteo2 contenant les mêmes données mais dans lequel les jours sont ordonnés chronologiquement.

La fonction `summary()` doit renvoyer ceci :

```
summary(meteo2)
```

	jour	soleil	vent
lundi	:1	FALSE:3	Min. : 0.00
mardi	:1	TRUE :4	1st Qu.:11.50
mercredi	:1		Median :14.00
jeudi	:1		Mean :19.29
vendredi	:1		3rd Qu.:26.50
samedi	:1		Max. :45.00
dimanche	:1		

10. Les objets dans R v. Les listes

Les listes

La fonction list()

Les listes sont les objets les plus “souples” de R.

On peut regrouper dans une liste tous les autres types d'objets vus précédemment.

La plupart des fonctions statistiques du logiciel renvoient leurs résultats sous forme de liste.

```
my.list <- list(month = mois.f, size = taille,  
                 weather=meteo2)  
summary(my.list)
```

	Length	Class	Mode
month	6	factor	numeric
size	8	-none-	numeric
weather	3	data.frame	list

Les listes

La fonction list()

```
my.list  
  
$month  
[1] jan fev jan mar mai avr  
Levels: jan fev mar avr mai  
  
$size  
[1] 165 162 184 191 174 174 188 157  
  
$weather  
      jour soleil vent  
1     lundi   TRUE    0  
2     mardi   TRUE   12  
3 mercredi FALSE   32  
4     jeudi FALSE   45  
5 vendredi FALSE   21  
6     samedi  TRUE   11  
7 dimanche  TRUE   14
```

Exercice

Listes et tirages aléatoires

Créez une liste contenant 3 éléments :

1. Un vecteur nommé uniforme qui contient 30 valeurs tirées au hasard entre 0 et 100 dans une loi de distribution uniforme.
2. Un vecteur nommé uniforme.crois qui contient ces mêmes 30 valeurs triées en ordre croissant.
3. Un vecteur nommé uniforme.decr qui contient ces mêmes 30 valeurs triées en ordre décroissant.

Vous aurez besoin de rechercher sur internet et dans l'aide de  comment tirer des valeurs dans une loi de distribution donnée (ici, la loi uniforme, chaque valeur entre 0 et 100 aura la même probabilité d'être tirée au hasard), et comment trier un vecteur en ordre croissant et décroissant.

11. L'indexation

L'indexation

Par position

L'indexation permet d'accéder à des éléments spécifiques au sein d'objets en contenant plusieurs.

On utilise la notation “`objet[position]`”

Par exemple, pour accéder au 4^e élément du vecteur `mois` :

```
mois[4]
```

```
[1] "mar"
```

Pour mémoire :

```
mois
```

```
[1] "jan" "fev" "jan" "mar" "mai" "avr"
```

L'indexation

Par position

On peut utiliser des vecteurs en guise de position.

```
mois  
[1] "jan" "fev" "jan" "mar" "mai" "avr"  
  
mois[c(2, 4)]  
[1] "fev" "mar"
```

```
mois[c(2, 4, 4, 4)]  
[1] "fev" "mar" "mar" "mar"
```

```
mois[6:2]  
[1] "avr" "mai" "mar" "jan" "fev"
```

L'indexation

Par position

La notation -position permet de retirer un ou des éléments :

```
mois  
[1] "jan" "fev" "jan" "mar" "mai" "avr"  
mois[-1]  
[1] "fev" "jan" "mar" "mai" "avr"  
mois[-c(1, 3)]  
[1] "fev" "mar" "mai" "avr"
```

Attention à ne pas donner une position qui n'existe pas :

```
mois[10]  
[1] NA
```

L'indexation

Par position

Pour les matrices et les tableaux, il faut 2 positions : le numéro de la ligne **puis** le numéro de la colonne :

```
mat  
      [,1] [,2] [,3] [,4]  
[1,]    1     4     7    10  
[2,]    2     5     8    11  
[3,]    3     6     9    12
```

```
mat[1, 2]
```

```
[1] 4
```

```
mat[c(1, 3), -2]
```

```
      [,1] [,2] [,3]  
[1,]    1     7    10  
[2,]    3     9    12
```

L'indexation

Par position

Pour les matrices et les tableaux, il faut 2 positions : le numéro de la ligne **puis** le numéro de la colonne :

```
meteo2
```

	jour	soleil	vent
1	lundi	TRUE	0
2	mardi	TRUE	12
3	mercredi	FALSE	32
4	jeudi	FALSE	45
5	vendredi	FALSE	21
6	samedi	TRUE	11
7	dimanche	TRUE	14

```
meteo2[6:7, -2]
```

	jour	vent
6	samedi	11
7	dimanche	14

L'indexation

Par position

Omettre le numéro de ligne (ou de colonne) signifie que l'on souhaite récupérer **toutes** les lignes (ou toutes les colonnes) :

```
meteo2[6:7, ]
```

	jour	soleil	vent
6	samedi	TRUE	11
7	dimanche	TRUE	14

```
meteo2[, 3]
```

```
[1] 0 12 32 45 21 11 14
```

```
meteo2[, 2]
```

```
[1] TRUE TRUE FALSE FALSE FALSE TRUE TRUE  
Levels: FALSE TRUE
```

L'indexation

Par position

Pour les tableaux et matrices disposant de **noms de colonnes**, il est possible d'utiliser la notation nom.objet\$nom.colonne :

```
meteo2[, 2]
[1] TRUE  TRUE  FALSE FALSE FALSE TRUE  TRUE
Levels: FALSE TRUE

meteo2$soleil
[1] TRUE  TRUE  FALSE FALSE FALSE TRUE  TRUE
Levels: FALSE TRUE
```

```
meteo2$jour
[1] lundi    mardi    mercredi jeudi    vendredi samedi
[7] dimanche
7 Levels: lundi mardi mercredi jeudi vendredi ... dimanche
```

L'indexation

Par position

Pour les listes, on peut utiliser le nom des éléments ou la notation plus complexe “[[]]” :

```
my.list$size  
[1] 165 162 184 191 174 174 188 157  
  
my.list[[2]]  
[1] 165 162 184 191 174 174 188 157
```

Ici, pour accéder à certaines tailles :

```
my.list$size[c(1, 3, 4)]  
[1] 165 184 191  
  
my.list[[2]][-1]  
[1] 162 184 191 174 174 188 157
```

L'indexation

Par condition

Lorsqu'on travaille avec des objets de grande taille, on a fréquemment besoin de récupérer des éléments en fonction de **critères précis**, sans nécessairement savoir où ces éléments se trouvent dans l'objet.

Par exemple, dans le vecteur `wind`, quelles sont les valeurs supérieures à 30 ?

```
wind[wind > 30]
```

```
[1] 32 45
```

L'indexation

Par condition

Les opérateurs permettant de spécifier les conditions sont les suivants :

>	→	supérieur
\geq	→	supérieur ou égal
<	→	inférieur
\leq	→	inférieur ou égal
$=$	→	égal
\neq	→	différent

Par ailleurs, on peut enchaîner plusieurs conditions avec :

&	→	ET logique
	→	OU logique

L'indexation

Par condition

Dans meteo2, quelles sont les lignes pour lesquelles on avait du soleil?

```
meteo2[meteo2$soleil == TRUE, ]
```

	jour	soleil	vent
1	lundi	TRUE	0
2	mardi	TRUE	12
6	samedi	TRUE	11
7	dimanche	TRUE	14

Y a-t-il des lignes pour lesquelles il y avait du soleil et plus de 10 km/h de vent?

```
meteo2[meteo2$soleil == TRUE & meteo2$vent > 10, ]
```

	jour	soleil	vent
2	mardi	TRUE	12
6	samedi	TRUE	11
7	dimanche	TRUE	14

L'indexation

Exercice

Installez le package ade4 et chargez-le en mémoire.

Tapez ensuite les commandes suivantes :

```
data(deug)
notes <- deug$tab
```

Vous disposez maintenant d'un tableau contenant les notes de 104 étudiants dans 9 disciplines :

```
dim(notes)
[1] 104   9

names(notes)
[1] "Algebra"      "Analysis"      "Proba"        "Informatic"
[5] "Economy"       "Option1"       "Option2"       "English"
[9] "Sport"
```

L'indexation

Exercice

On cherche à savoir si, comme le veut la croyance populaire, les meilleurs élèves en math sont les plus mauvais en sport....

Pour cela :

1. calculez la moyenne des notes de sport de toute la promotion
2. identifiez à quoi correspond une “bonne note” en algèbre. On considérera comme “bonne”, une note obtenue par le meilleur quart des étudiants.
3. calculez la moyenne des notes de sport pour les étudiants ayant obtenu une note d’algèbre supérieure ou égale à la bonne note identifiée à l’étape précédente

12. Import / Export...

Importer des données dans R

Les étapes

Historiquement, pour importer dans R des données issues de tableur, plusieurs étapes devaient être respectées :

1. préparer les données dans Excel (ou Open Office Calc)
2. Exporter le fichier au format texte brut
3. Fermer le tableur
4. Importer les données dans R avec la fonction `read.table()`

Nous verrons très vite que dans R, tout cela est beaucoup plus simple.

Importer des données dans R

1. Préparation des données dans Excel

Avant toute chose, faire une **copie** du fichier brut. Ne **jamais** travailler directement sur l'**original**!

1. Placer les variables en colonnes et les observations en ligne
2. Pas de cases vides : remplir avec des NA
3. Placer les noms de colonnes sur la première ligne de la feuille Excel
4. Aucun caractère spécial tel que #, \$, %, &, ^, {}, [], accents, cédille, guillemets, ...
5. Pas d'espaces dans les noms de variables ou de catégories. Les remplacer par des points ou tirets bas
6. Les noms de lignes doivent être uniques
7. Privilégier les noms courts

Importer des données dans R

2. Exporter au format texte brut

Enregistrez dans le répertoire de travail de R le fichier au format texte brut en utilisant :

- ▶ soit Fichier > Enregistrer sous > texte csv
- ▶ soit Fichier > Enregistrer sous > texte séparateur tabulation

Si le format csv est choisi, il faut s'assurer (par exemple en ouvrant le fichier dans le bloc notes de Windows) que le séparateur de colonnes n'est pas le même symbole que celui utilisé pour les décimales.

Importer des données dans R

`read.table()`

R est capable d'importer des fichiers au format **texte brut** grâce à la fonction `read.table()`.

Il existe de nombreuses fonctions dérivées de `read.table()` :

- ▶ `read.delim()`
- ▶ `read.delim2()`
- ▶ `read.csv()`
- ▶ `read.csv2()`
- ▶ ...

Elles font toutes la même chose que `read.table()` mais leurs arguments utilisent des valeurs par défaut différentes.

Importer des données dans R

`read.table()`

La fonction `read.table()` prend plusieurs arguments :

- ▶ `file` : c'est le seul argument qui n'a pas de valeur par défaut. Il faut spécifier le nom (ou le chemin) du fichier à importer. Cela peut être un fichier issu du web.
- ▶ `header` : le fichier contient-il une ligne de titres de colonnes
- ▶ `dec` : quel est le symbole utilisé dans le fichier pour les décimales
- ▶ `sep` : quel est le symbole utilisé dans le fichier pour séparer les colonnes
- ▶ `row.names` : quel est le numéro de colonne contenant les titres des lignes (à supposer qu'il y en ait un)

Importer des données dans R

`read.table()`

Exemple fictif :

```
dat <- read.table("donnees.txt", header = TRUE,  
                  sep = "\t", dec = ",")
```

Dans la pratique, on utilise souvent l'assistant d'importation de R.

Importer des données dans R

L'assistant d'importation de R

Tout est expliqué ici : importer des données dans Rstudio

Tout les réglages sont faits grâce à une interface graphique. En général, on s'assure au moins que :

- ▶ le séparateur de colonnes est correct
- ▶ la nature des données contenues dans chaque colonne est la bonne. Cela suppose souvent de changer le symbole utilisé pour les décimales des variables numériques

Enfin, il est important de ne pas cliquer sur le bouton Import, mais au contraire de copier-coller dans le script le code généré automatiquement par l'assistant d'importation.

Exporter des données depuis R

`write.table()`

Cette fonction est très simple à utiliser :

```
write.table(objet, "NomFichier.txt")
```

Cette commande crée un fichier nommé “NomFichier.txt” dans le répertoire de travail. Ce fichier contiendra les données contenues dans objet.

Comme pour `read.table()`, les arguments sont nombreux :

- ▶ `sep`
- ▶ `dec`
- ▶ `row.names`
- ▶ `col.names`
- ▶ `...`