

Initiation au langage R

Séance 1 d'introduction

Groupe ElementR

Joséphin Béraud, Léa Christophe, Aurélie Douet

2 décembre 2022



Programme

- **Qu'est-ce qu'on peut faire avec R ?**
- **R et RStudio : c'est quoi la différence ?**
- **Créer un projet et un script**
- **Manipuler des objets : assignation, indexation, fonctions**
- **Les packages**

Programme



Qu'est-ce qu'on peut faire avec R ?



R et RStudio : c'est quoi la différence ?



Créer un projet et un script

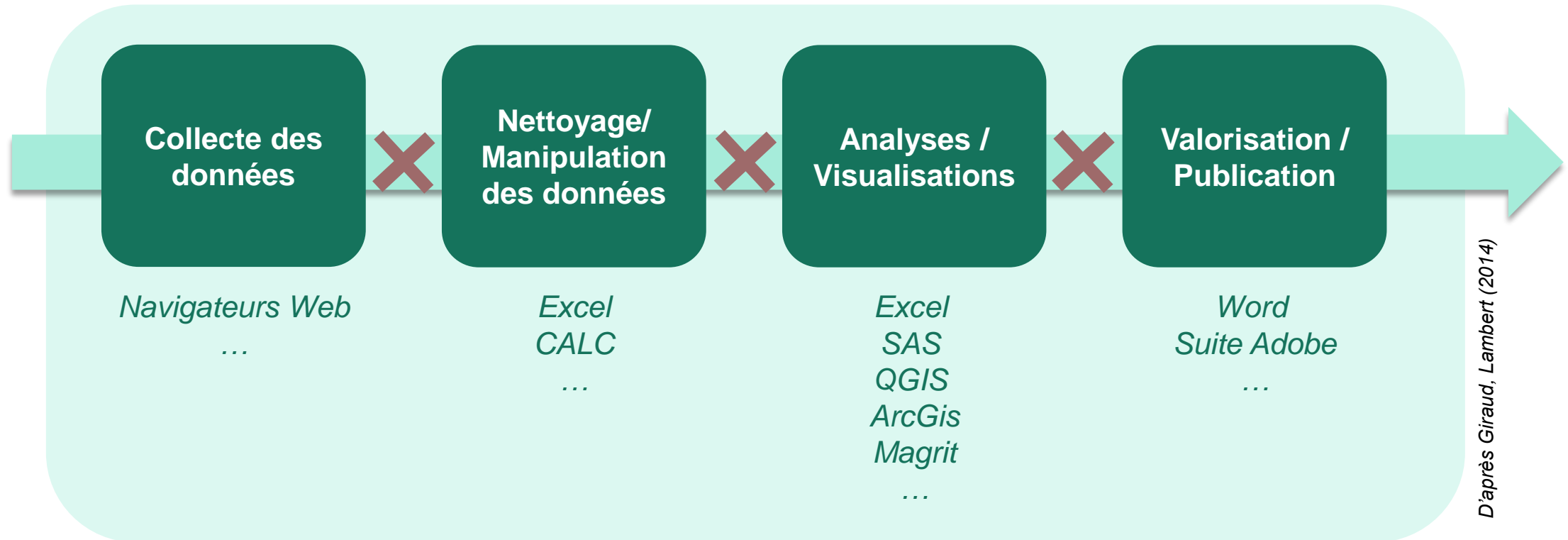


Manipuler des objets : assignation, indexation, fonctions



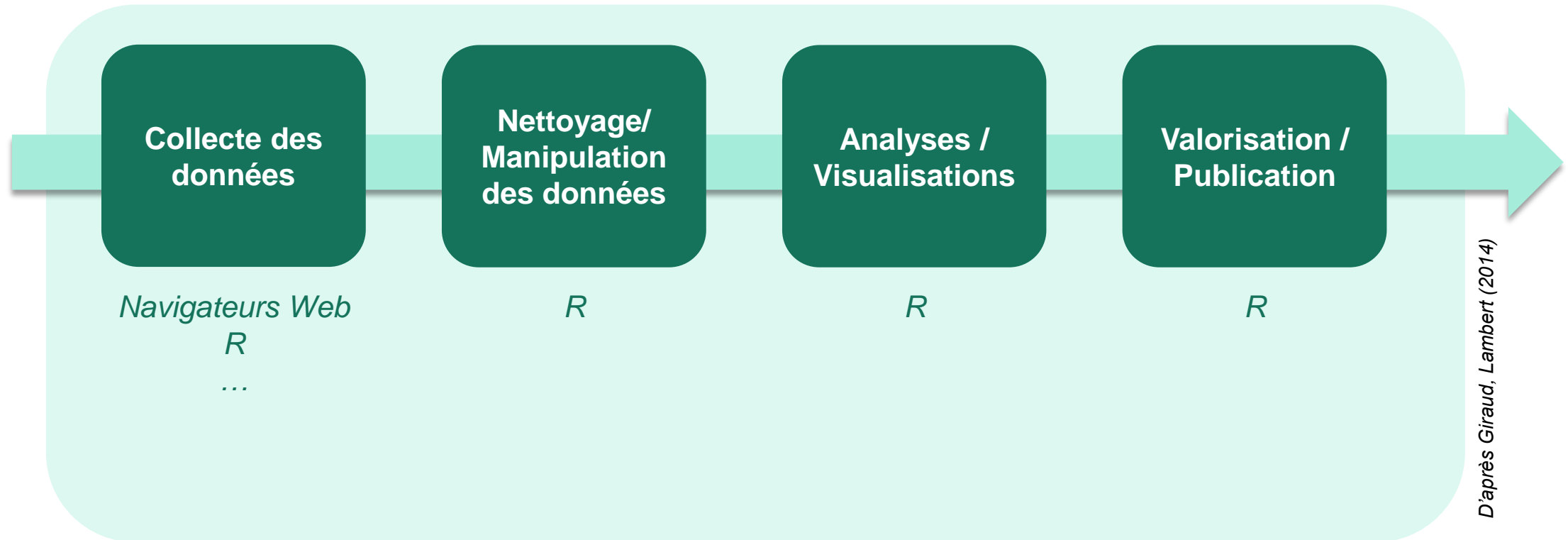
Les packages

De la collecte à la valorisation sans R ...



- Utilisation de **différents outils** pour effectuer l'intégralité de la chaîne d'analyses de données
- **Pas de transparence et de reproductibilité** de la chaîne de traitement

De la collecte à la valorisation avec R ...



- Utilisation d'un **seul outil** pour effectuer l'intégralité de la chaîne d'analyses de données
- **Transparence et reproductibilité** de la chaîne de traitement

Exemple : Production de diaporamas à partir de questionnaires

Contexte : La conférence des financeurs de la prévention de la perte d'autonomie (CFPPA) réalise, tous les ans, un questionnaire auprès des bénéficiaires des structures qu'elle finance.

Objectif : Produire une synthèse générale des résultats du questionnaire et une par structure tous les ans.

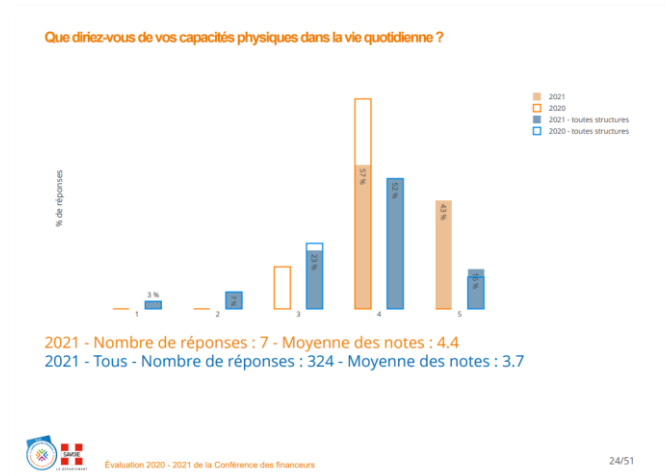
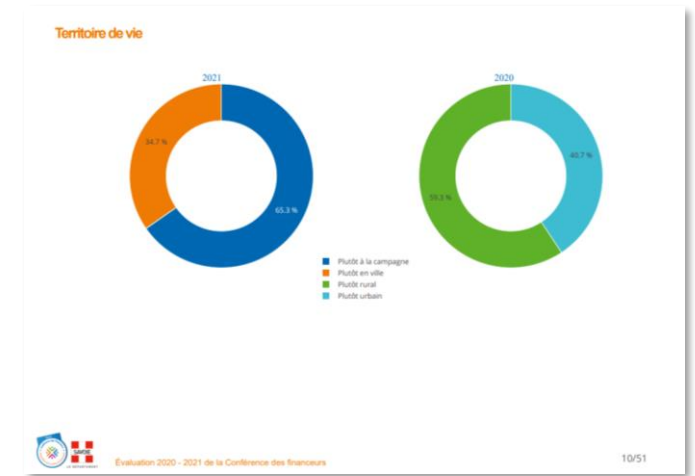
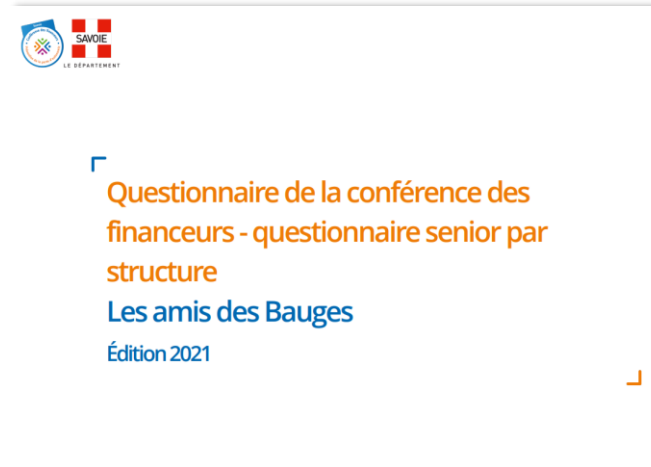
Import des résultats du questionnaire

Nettoyage des tableaux + Manipulation des données

Sélection des indicateurs et des représentations (carto)graphiques

Production de diaporama finalisé

Extraits de slides obtenus pour l'une des structures (sur 50 slides)



Pourquoi utiliser R dans cet exemple?

- **Lisibilité du traitement** : Effectuer la chaîne de traitement de l'import des données au diaporama dans un seul outil
- **Gain de temps** : Production automatisée d'un diaporama général et un par structure (soit 30 diaporamas)

Programme



Qu'est-ce qu'on peut faire avec R ?



R et RStudio : c'est quoi la différence ?



Créer un projet et un script



Manipuler des objets : assignation, indexation, fonctions



Les packages

R n'est pas RStudio



Langage-programme



Interface graphique

R est à la fois un logiciel et un langage



- Un logiciel libre intégré au projet GNU, à l'origine développé dans le milieu académique pour l'analyse des données et leur visualisation en graphique
- Un langage de programmation interprété : les commandes (écrites en R) sont interprétées (par le logiciel R) et exécutées (par la machine)
 - immédiatement et
 - l'une après l'autre

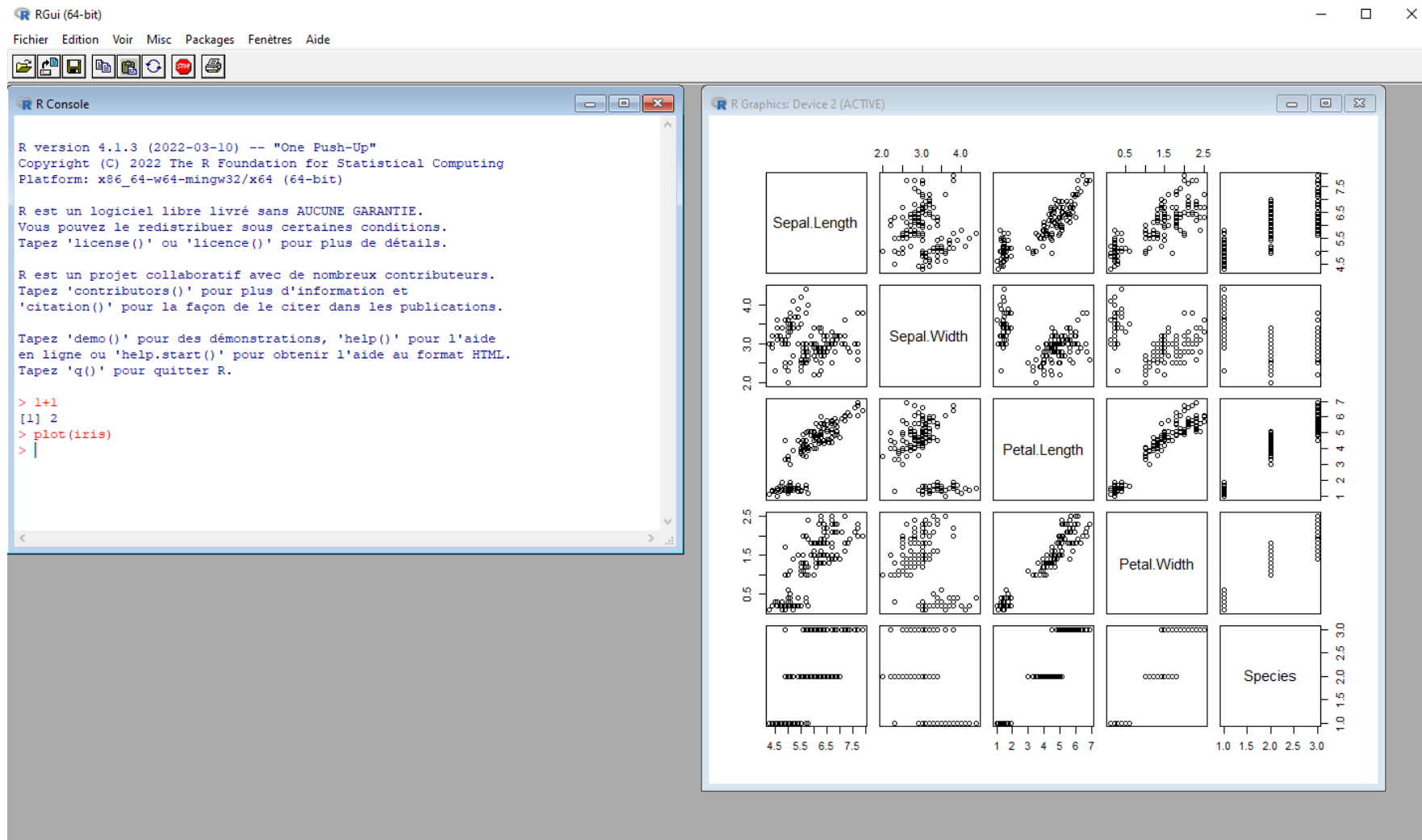
Guide d'installation de R :

<https://quanti.hypotheses.org/1813#installer-r>

L'interface graphique de R

RGui est l'interface du logiciel R.

Elle se présente sous la forme d'une simple invite de commande



L'interface graphique RStudio

RStudio est un environnement de développement intégré (IDE) dédié à R



- Une interface pour R - parmi d'autres - largement populaire car aboutie et conviviale
- Le produit d'une entreprise commerciale avec :
 - Une version open source et gratuite (RStudio Desktop / RStudio Server)
 - Une version payante (RStudio Desktop Pro/ RStudio Server Pro)

Guide d'installation de RStudio :

<https://quanti.hypotheses.org/1813#installer-rstudio>

Paramétrer l'interface : l'encodage

The image shows the RStudio interface with the **Tools** menu open and **Global Options...** selected. The **Options** dialog is open, with the **Saving** tab active. Under the **Serialization** section, the **Default text encoding** is set to **UTF-8**. A **Change...** button is visible next to the encoding field. A **Choose Encoding** dialog is also open, showing a list of encodings with **UTF-8** selected.

Tools menu items:

- Install Packages...
- Check for Package Updates...
- Version Control
- Shell...
- Terminal
- Background Jobs
- Addins
- Memory
- Keyboard Shortcuts Help (Alt+Shift+K)
- Modify Keyboard Shortcuts...
- Edit Code Snippets...
- Show Command Palette (Ctrl+Shift+P)
- Project Options...
- Global Options...

Options dialog tabs: Editing, Display, **Saving**, Completion, Diagnostics

General

- Ensure that source files end with newline
- Strip trailing horizontal whitespace when saving
- Restore last cursor position when opening file

Serialization

Line ending conversion: Platform Native

Default text encoding: UTF-8

Auto-save

- Always save R scripts before sourcing
- Automatically save when editor loses focus
- When editor is idle: Backup unsaved changes
- Idle period: 1000ms

Choose Encoding

- [Ask]
- ISO-8859-1 (System default)
- ASCII
- BIG5
- GB18030
- GB2312
- ISO-2022-JP
- ISO-2022-KR
- ISO-8859-2
- ISO-8859-7
- SHIFT-JIS
- UTF-8**
- WINDOWS-1252

Show all encodings

Buttons: OK, Cancel

Paramétrer l'interface : ne pas enregistrer de .RData par

The screenshot shows the RStudio interface with the 'Global Options' dialog box open. The 'Workspace' tab is selected, and the 'Restore .RData into workspace at startup' checkbox is unchecked. The 'Save workspace to .RData on exit' dropdown menu is open, showing 'Never' selected. The 'History' section has 'Always save history (even when not saving .RData)' checked. The 'Other' section has 'Wrap around when navigating to previous/next tab', 'Automatically notify me of updates to RStudio', and 'Send automated crash reports to RStudio' checked.

Global Options - Workspace

- Restore .RData into workspace at startup
- Save workspace to .RData on exit: **Never**
- Always save history (even when not saving .RData)
- Remove duplicate entries in history

Other

- Wrap around when navigating to previous/next tab
- Automatically notify me of updates to RStudio
- Send automated crash reports to RStudio

Paramétrer l'interface : la langue

The image shows the RStudio interface with the 'Tools' menu open and the 'Options' dialog box displayed. The 'Advanced' tab is selected in the Options dialog, and the 'User Interface Language' dropdown menu is open, showing 'French (Français)' as the selected option.

RStudio Tools Menu:

- Install Packages...
- Check for Package Updates...
- Version Control
- Shell...
- Terminal
- Background Jobs
- Addins
- Memory
- Keyboard Shortcuts Help (Alt+Shift+K)
- Modify Keyboard Shortcuts...
- Edit Code Snippets...
- Show Command Palette (Ctrl+Shift+P)
- Project Options...
- Global Options...

Options Dialog - Advanced Tab:

- Debugging:**
 - Use debug error handler only when my code contains errors
- OS Integration:**
 - Rendering engine: Auto-detect (recommended)
 - Use GPU exclusion list (recommended)
 - Use GPU driver bug workarounds (recommended)
 - Show full path to project in window title
- Other:**
 - Show .Last.value in environment listing
 - Help panel font size: 10
- Experimental Features:**
 - User Interface Language: English (dropdown menu open, showing English and French (Français))

Buttons: OK, Cancel, Apply

Paramétrer l'interface : les parenthèses arc-en-ciel

The image shows the RStudio interface with the 'Tools' menu open and the 'Options' dialog box displayed. The 'Tools' menu includes options like 'Install Packages...', 'Check for Package Updates...', 'Version Control', 'Shell...', 'Terminal', 'Background Jobs', 'Addins', 'Memory', 'Keyboard Shortcuts Help', 'Modify Keyboard Shortcuts...', 'Edit Code Snippets...', 'Show Command Palette', 'Project Options...', and 'Global Options...'. The 'Options' dialog box is set to the 'Display' tab, and the 'Rainbow parentheses' checkbox is checked. A tooltip explains that this option is for highlighting parentheses in various colors. The 'Fold Style' dropdown is also visible.

Tools

- Install Packages...
- Check for Package Updates...
- Version Control
- Shell...
- Terminal
- Background Jobs
- Addins
- Memory
- Keyboard Shortcuts Help Alt+Shift+K
- Modify Keyboard Shortcuts...
- Edit Code Snippets...
- Show Command Palette Ctrl+Shift+P
- Project Options...
- Global Options...

Options

Editing **Display** Saving Completion Diagnostics

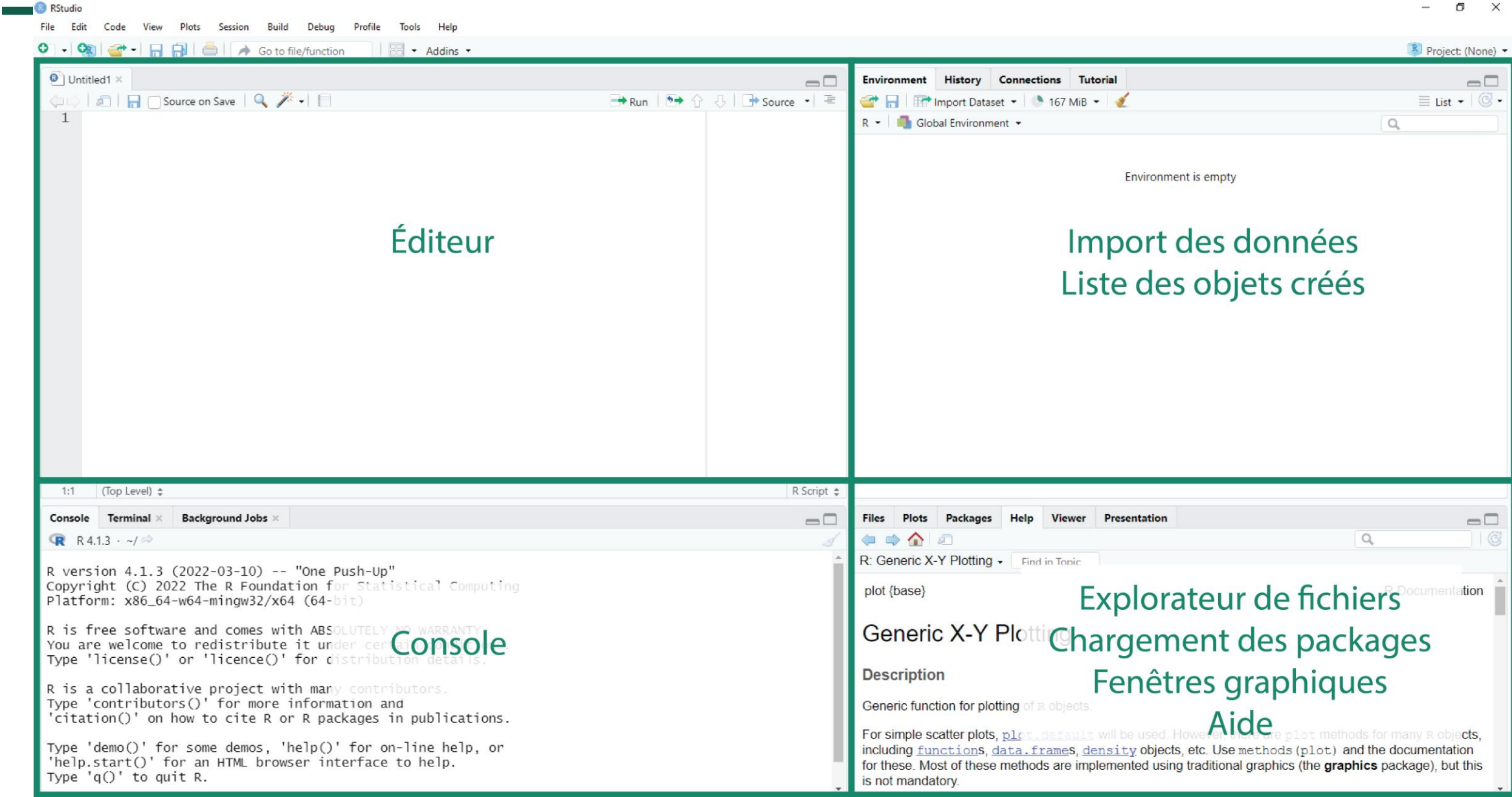
General

- Highlight selected word
- Highlight selected line
- Show line numbers
- Relative line numbers
- Show margin
- Margin column
- Show whitespace characters
- Show indent guides
- Blinking cursor
- Allow scroll past end of document
- Allow drag and drop of text
- Highlight R function calls
- Rainbow parentheses
- Fold Style

Whether to highlight parentheses in a variety of colors.

OK Cancel Apply

Les 4 volets de RStudio



Les 4 volets de RStudio

1. L'éditeur : zone réservée aux scripts dans lesquels on écrit le code.

Pour lancer une commande, autrement dit envoyer son code dans la console, on peut soit utiliser le bouton Run, soit utiliser ctrl+enter

2. La console : c'est l'interpréteur R qui exécute les commandes.

Il est également possible d'écrire ses commandes directement dans la console

Les 4 volets de RStudio

3. onglet Environment : liste des objets créés dans une session. Permet également d'importer des données au clic bouton

History : historique des commandes exécutées depuis le début d'une session

Connections : interface pour se connecter à des bases de données

Tutorials : lancer des tutos du package learnr

A savoir : il est également possible d'associer un projet RStudio à un projet Git. Dans ce cas, c'est dans ce volet qu'apparaît l'onglet dédié au Git

4. onglet Files : pour naviguer dans les dossiers de son ordinateur

Plots : onglet réservé à l'affichage des sorties graphiques statiques

Packages : liste des packages installés dans R

Help : fiches de documentation sur les packages

Viewer : pour l'affichage de carto/graphiques interactifs

Programme



Qu'est-ce qu'on peut faire avec R ?



R et RStudio : c'est quoi la différence ?



Créer un projet et un script



Manipuler des objets : assignation, indexation, fonctions



Les packages

Téléchargement du diaporama et du script

Vous pouvez accéder au diaporama et au script à partir du lien suivant :
<https://elementr.gitpages.huma-num.fr/website/posts/seance2.html>

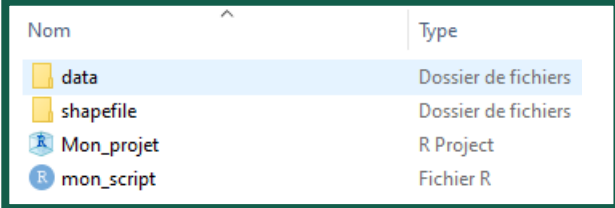
Créer un Projet R

Projet R Studio :

- meilleure organisation de son travail,
- meilleure reproductibilité,
- meilleure portabilité.

Dans un dossier, mettre :

- Fichier .Rproj
- Les éléments du projet (données, shapefile, documentation...)



A screenshot of a file explorer window showing a directory structure. The table below represents the content of the screenshot:

Nom	Type
data	Dossier de fichiers
shapefile	Dossier de fichiers
Mon_projet	R Project
mon_script	Fichier R

Projet R Studio est un fichier qui remplace le répertoire par défaut par le dossier où il est situé.

Chemin absolu :

- *Propre à chaque machine*
- *Doit être mis à jour (déplace, renomme...)*

C:/Users/Prénom/Desktop/Mon_projet/mon_script

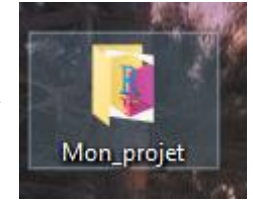
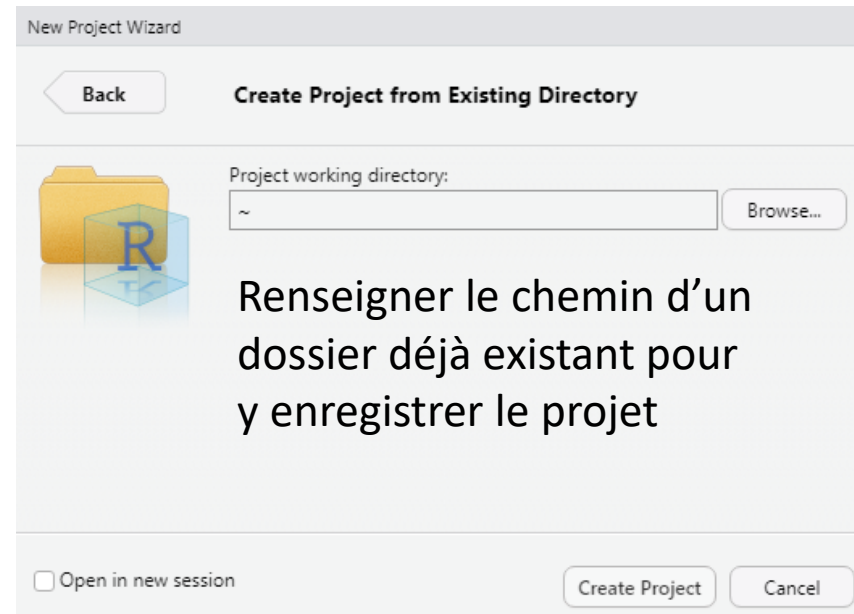
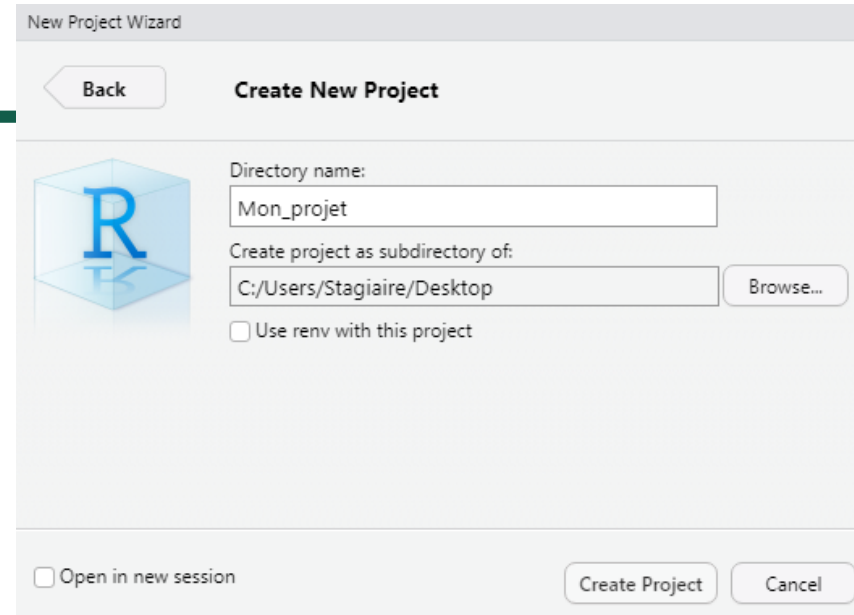
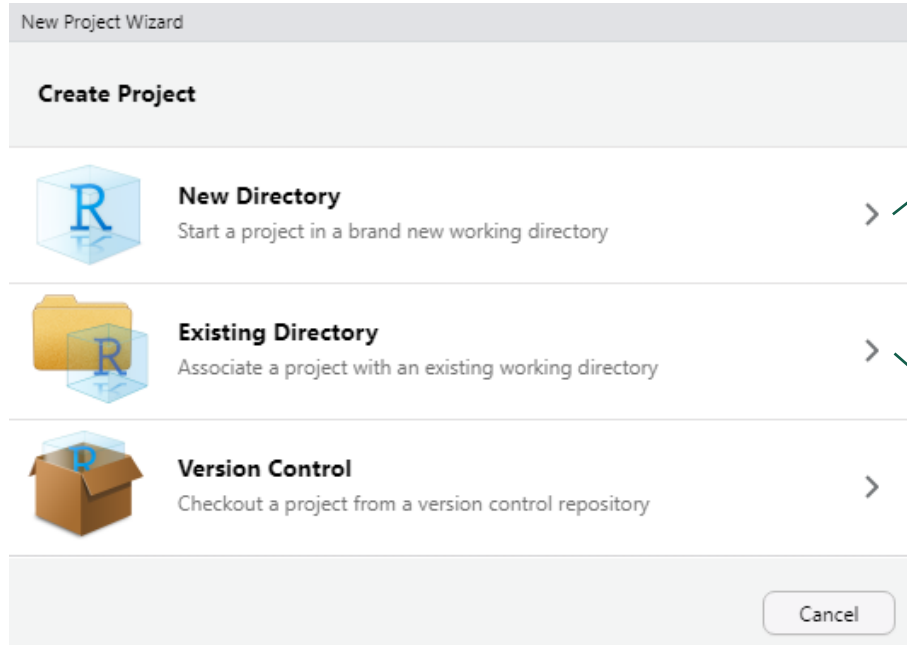
Chemin relatif :

- *Est relatif à l'emplacement du fichier .Rproj*
- *Se met à jour automatiquement du côté machine*

./mon_script

Créer un Projet R

Fichier > Nouveau projet



Le dossier est créé sur le bureau

Créer un Projet R

The screenshot displays the RStudio interface for a project named "Mon_projet". The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for running, saving, and navigating. The main editor window shows a script file named "mon_script.R" with the following content:

```
1 # Mon script
2 |
```

A green box highlights the script file name in the top-left corner, and a green arrow points from this box to the file explorer view. The file explorer shows the project directory structure:

Nom	Type
..	
Mon_projet.Rproj	218 B
data	Dossier de fichiers
shapefile	Dossier de fichiers
mon_script.R	15 B
Mon_projet	R Project
mon_script	Fichier R

The console window at the bottom shows the R version (4.1.3) and the project path (~/.Mon_projet/). The terminal output includes the R license and help information.

Script situé dans le dossier du projet

Explorateur de fichiers se met automatiquement dans le dossier où se situe le projet

Créer un script R

Fichier > Nouveau Fichier > Script R va créer un nouveau script dans l'éditeur, untitled1 (ou untitledN)

```
1 # Mon premier script
2 # 2/12/22, ElementR, Campus Condorcet
3
4 # Ma première opération ####
5 1 + 1
6
7 # Les choses sérieuses ----
8 2 + 2 # devrait faire 4
9
10 # La suite
11
12
13 # La fin ----
14
15
16
```

Ma première opération
Les choses sérieuses
La suite
La fin

Index

Une section peut être réduite

Naviguer entre les sections

Tout ce qui est après un # n'est pas lu par la machine (commentaire)

---- ou #### permet de faire un section et d'organiser son travail

Enregistrer son script : Fichier > Enregistrer sous > et choix de l'emplacement et du nom du script, en .R

Programme



Qu'est-ce qu'on peut faire avec R ?



R et RStudio : c'est quoi la différence ?



Créer un projet et un script



Manipuler des objets : assignation, indexation, fonctions



Les packages

Manipuler des objets

Plan

1. Assignment / création d'objet
2. Type d'objet et nature des données
3. Indexation et opérateurs
4. Fonctions de base : structure des données
5. Fonctions de base : description statistique des variables
6. Manipuler un dataframe

1. Assignment

L'assignation sert à la création d'un objet dans l'environnement : elle permet de stocker un résultat pour la réutilisation de celui-ci plus tard *dans la même session*. Elle se fait avec l'opérateur `<-`.

Raccourci clavier
(Windows/Linux)
Insérer l'opérateur
d'assignation : alt+-

The screenshot displays the RStudio interface. The main editor window contains the following R code:

```
9
10 # 1. Assignation/création d'objet ----
11
12 # R, s'il-te-plaît, calcule 21*2
13 21*2
14
15 # Création d'un objet avec assignation dans l'environnement
16 monObjet <- 21*2
17
18 # Attention, R est sensible à la casse
19 monobjet <- 21*2
20
21 # deux objets distincts ont été ajoutés dans mon environnement,
22 # pour afficher la liste de mes objets dans la console :
23 ls()
24
25
26
27
28
29
30
31
32
33
34
```

The Environment pane on the right shows the following values:

Variable	Value
monobjet	42
monObjet	42

The Console pane at the bottom shows the output of the code:

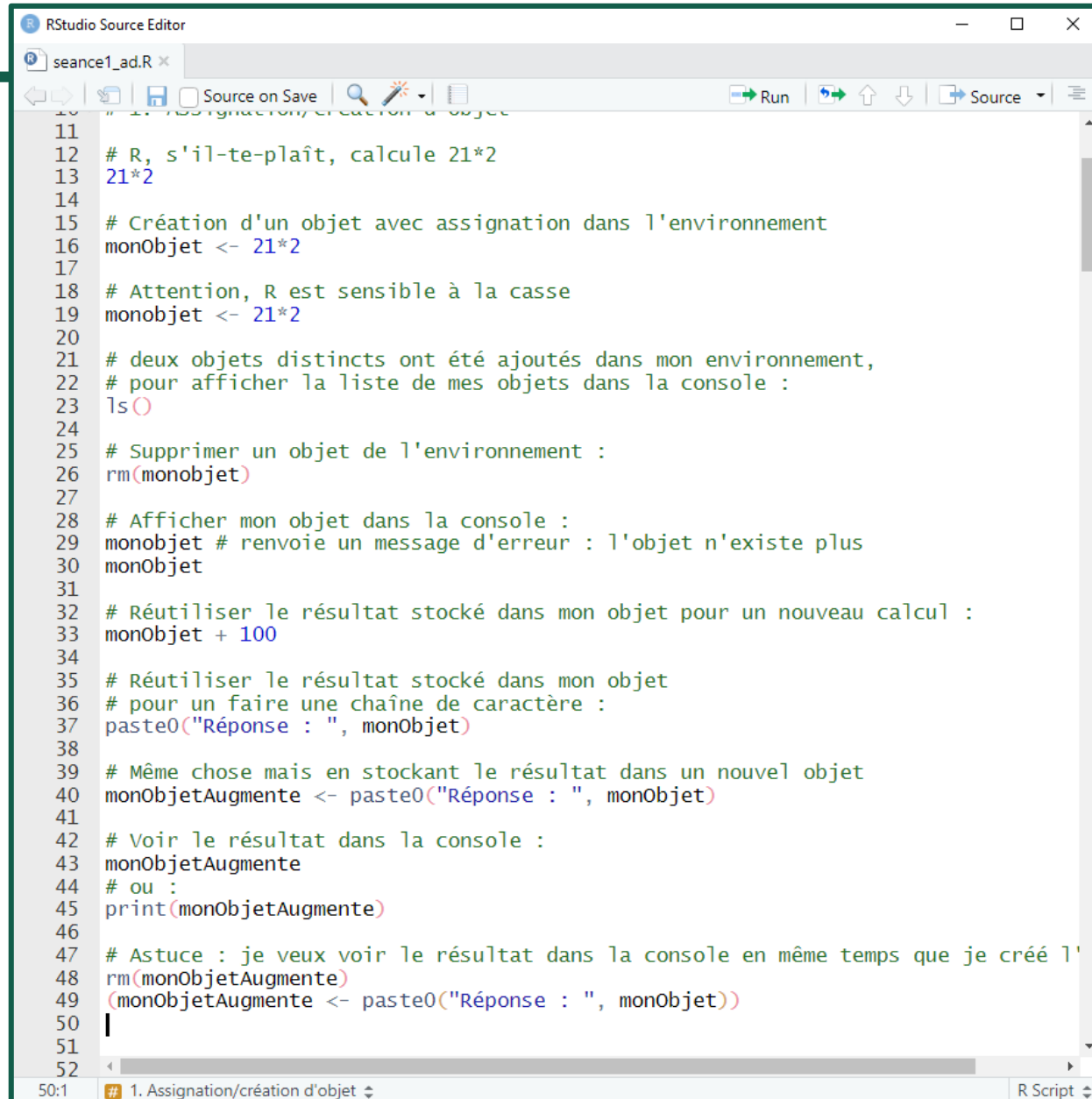
```
R 4.1.3 ~ /
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> # R, s'il-te-plaît, calcule 21*2
> 21*2
[1] 42
> # Création d'un objet avec assignation dans l'environnement
> monObjet <- 21*2
> # Attention, R est sensible à la casse
> monobjet <- 21*2
> # deux objets distincts ont été ajoutés dans mon environnement,
> # pour afficher la liste de mes objets dans la console :
> ls()
[1] "monobjet" "monObjet"
>
```

1. Assignment

Exercice de manipulation :

créer un objet simple et observer la console et l'environnement global



```
10 # 1. Assignment/création d'objet
11
12 # R, s'il-te-plaît, calcule 21*2
13 21*2
14
15 # Création d'un objet avec assignment dans l'environnement
16 monObjet <- 21*2
17
18 # Attention, R est sensible à la casse
19 monobjet <- 21*2
20
21 # deux objets distincts ont été ajoutés dans mon environnement,
22 # pour afficher la liste de mes objets dans la console :
23 ls()
24
25 # Supprimer un objet de l'environnement :
26 rm(monobjet)
27
28 # Afficher mon objet dans la console :
29 monobjet # renvoie un message d'erreur : l'objet n'existe plus
30 monObjet
31
32 # Réutiliser le résultat stocké dans mon objet pour un nouveau calcul :
33 monObjet + 100
34
35 # Réutiliser le résultat stocké dans mon objet
36 # pour un faire une chaîne de caractère :
37 paste0("Réponse : ", monObjet)
38
39 # Même chose mais en stockant le résultat dans un nouvel objet
40 monObjetAugmente <- paste0("Réponse : ", monObjet)
41
42 # Voir le résultat dans la console :
43 monObjetAugmente
44 # ou :
45 print(monObjetAugmente)
46
47 # Astuce : je veux voir le résultat dans la console en même temps que je créé l'
48 rm(monObjetAugmente)
49 (monObjetAugmente <- paste0("Réponse : ", monObjet))
50 |
51
52
```

Raccourci clavier
(Windows/Linux)

Insérer l'opérateur
d'assignment : alt+-

Exécution d'une ligne
de code dans le volet
Editeur : Ctrl + entrée

Autocomplétion : Tab

Mémo sur les opérateurs arithmétiques

Opérateur	Description	Exemple	Résultat
+	addition	2+2	4
-	soustraction	2-2	0
*	multiplication	2*2	4
/	division	2/2	0
^	puissance	2^2	4
%%	modulo	2%%2	0
%/%	quotient décimal	2%/2	1

2. Les types d'objets

Il existe plusieurs types d'objet dans R. Pour cette séance d'initiation, nous en aborderons trois :

- Les vecteurs
- Les facteurs
- Les dataframes

2. Les types d'objets : les vecteurs

Un vecteur est une collection à une dimension d'éléments de même nature

Les éléments se combinent avec la fonction `c()`

```
> # Vecteur de nombres
> vNum <- c(41.5, 38, 37)
> vNum
[1] 41.5 38.0 37.0
```

```
> # vecteur de chaînes de caractères
> vChar <- c("Joséphine", "Léa", "Aurélie")
> vChar
[1] "Joséphine" "Léa"      "Aurélie"
```

```
> # vecteur de booléens
> vBoo <- c(FALSE, TRUE, TRUE)
> vBoo
[1] FALSE TRUE TRUE
```

Uni-dimensionnel

```
> vNum
[1] 41.5 38.0 37.0
```

```
> vChar
[1] "Joséphine" "Léa"      "Aurélie"
```

```
> vBoo
[1] FALSE TRUE TRUE
```

De même nature

```
> class(vNum)
[1] "numeric"
```

```
> class(vChar)
[1] "character"
```

```
> class(vBoo)
[1] "logical"
```


Mémo sur la nature des données (non exhaustif)

Grand type	Type	Description	Exemple
numeric	integer	nombres entiers	10
	double	nombres réels	10,56
character	character	Chaîne de caractère	"Hello Word"
Logical/boolean	logical ou bolean	Vrai/ faux/manquant	TRUE/FALSE/NA

2. Les types d'objets : les facteurs

Un facteur est également un vecteur d'éléments mais avec des modalités prédéfinies, les *levels*

Pour créer un facteur, on utilise la fonction `factor()`

```
> # Création d'un facteur
> f <- factor(c("homme", "homme", "femme", "femme", "femme",
+             "femme", "femme", "femme", "homme", "homme"))
> f
[1] homme homme femme femme femme femme femme femme homme homme
Levels: femme homme
```

Le facteur `f` contient 10 éléments :

```
> length(f)
[1] 10
```

Et 2 modalités possibles :

```
> levels(f)
[1] "femme" "homme"
```

2. Les types d'objets : les dataframes

Un dataframe est un tableau de données à deux dimensions (lignes et colonnes) :

- Chaque colonne (ou variable) est un vecteur nommé :
 - les données stockées doivent être de même nature ;
 - la 1ère ligne correspond aux noms des variables
- Un dataframe peut combiner des colonnes de types différents ...
- ... mais elles doivent avoir la même longueur (i.e. le même nombre de lignes)

Pour créer un dataframe, on utilise la fonction `data.frame()`

```
> (df <- data.frame(NOM = vChar, FEMME = vBoo, POINTURE = vNum))  
  NOM FEMME POINTURE  
1 Joséphin FALSE    41.5  
2 Léa    TRUE    38.0  
3 Aurélie TRUE    37.0
```

3. L'indexation

L'indexation permet d'intervenir dans un vecteur pour transformer, extraire, supprimer ou ajouter des éléments

Indexation par position : chaque élément est implicitement lié à un index qui correspond à sa position dans le vecteur, $v[i]$

```
> vNum[2]  
[1] 38
```

Dans un dataframe, on accède à un élément en indiquant sa ligne i et sa colonne j , $df[i, j]$

```
> df[1,3]  
[1] 41.5
```

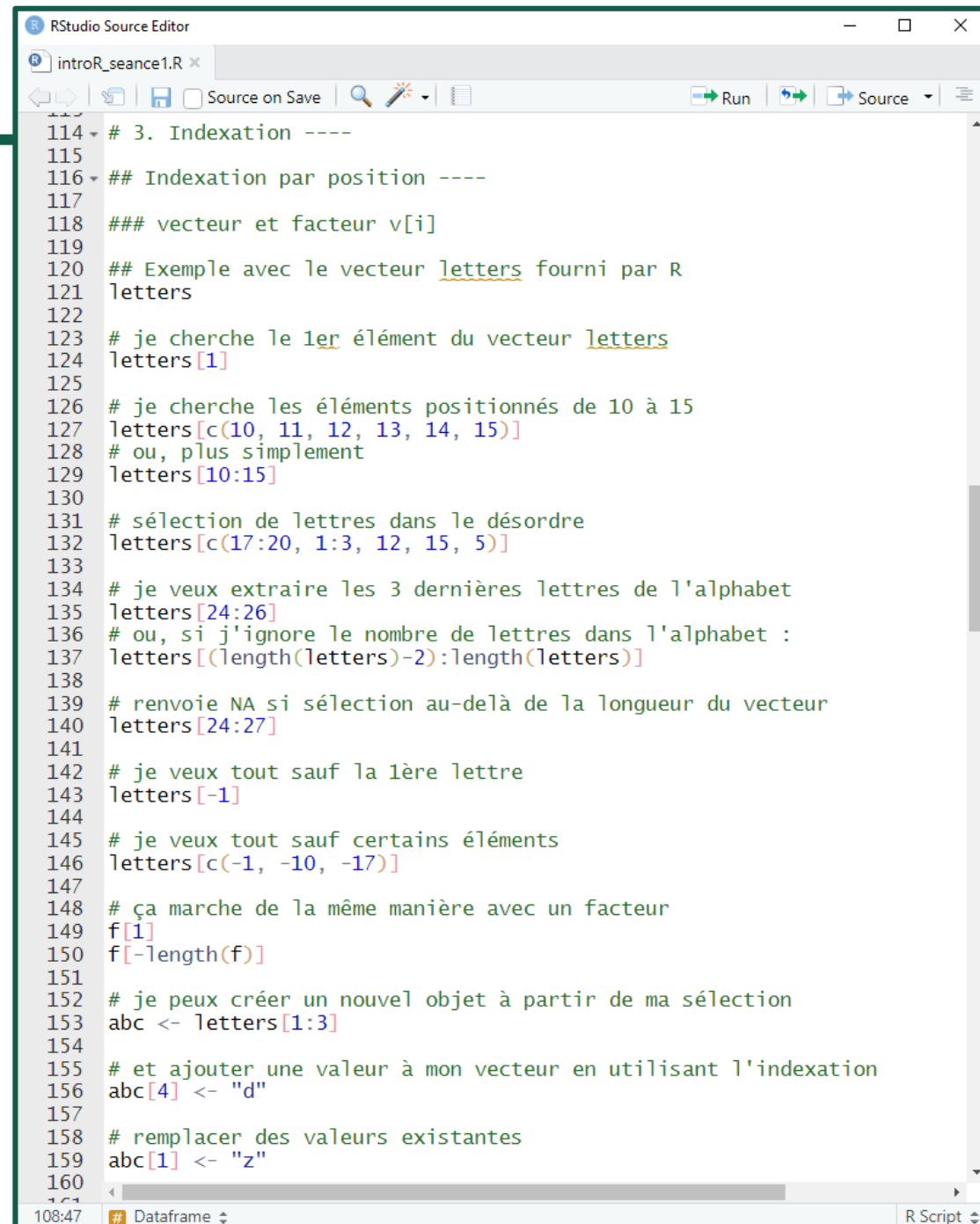
Indexation par condition : il est possible d'atteindre les éléments d'un vecteur en utilisant une condition. Si celle-ci est remplie, le ou les éléments sont renvoyés.

```
> vNum[vNum==38]  
[1] 38
```

3. L'indexation par position

Exercice de manipulation sur des vecteurs :

Reproduire le code ci-contre et observer les résultats

The image shows a screenshot of the RStudio Source Editor window. The title bar reads "RStudio Source Editor" and the active file is "introR_seance1.R". The code is as follows:

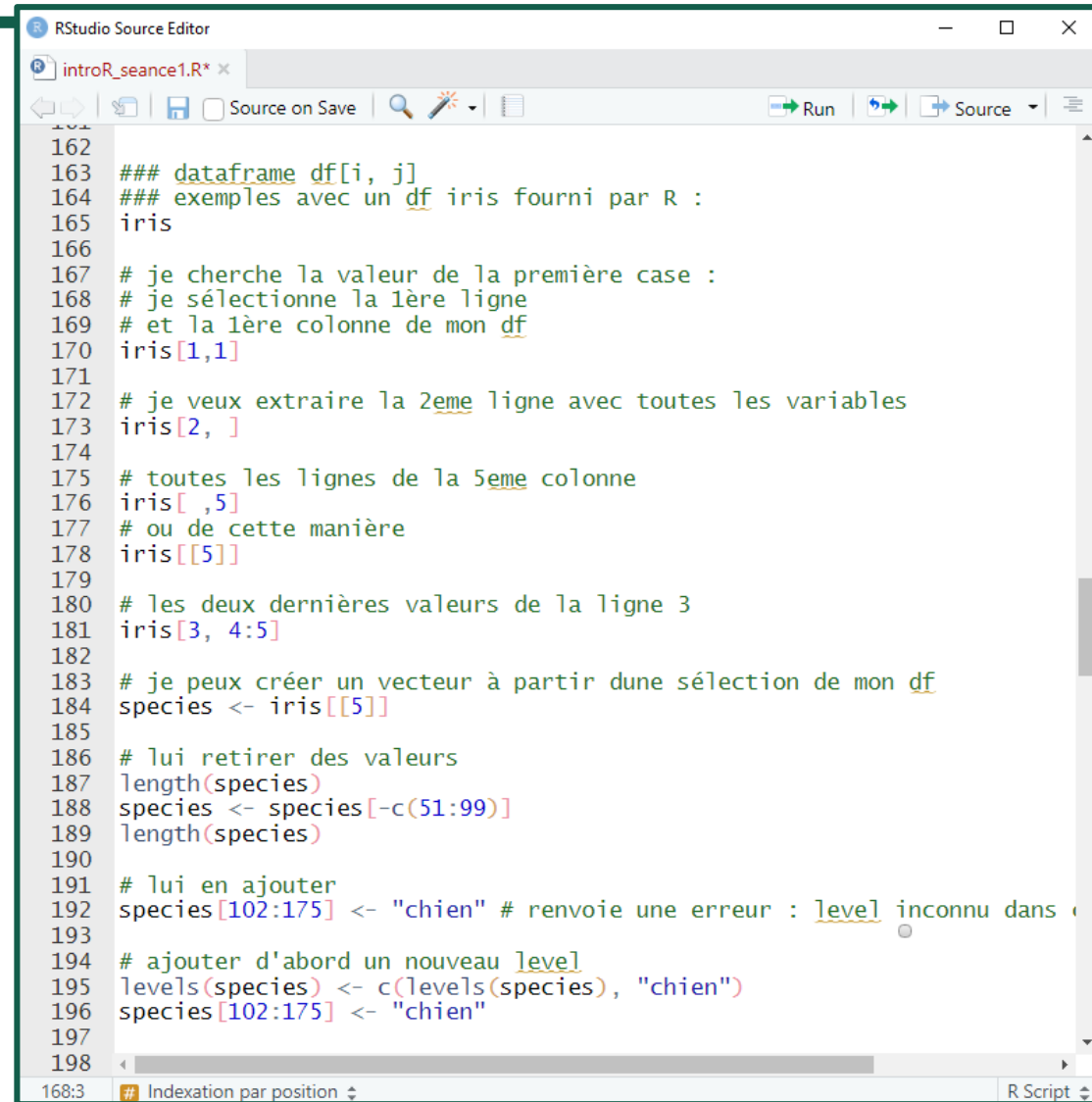
```
114 # 3. Indexation ----
115
116 ## Indexation par position ----
117
118 ### vecteur et facteur v[i]
119
120 ## Exemple avec le vecteur letters fourni par R
121 letters
122
123 # je cherche le 1er élément du vecteur letters
124 letters[1]
125
126 # je cherche les éléments positionnés de 10 à 15
127 letters[c(10, 11, 12, 13, 14, 15)]
128 # ou, plus simplement
129 letters[10:15]
130
131 # sélection de lettres dans le désordre
132 letters[c(17:20, 1:3, 12, 15, 5)]
133
134 # je veux extraire les 3 dernières lettres de l'alphabet
135 letters[24:26]
136 # ou, si j'ignore le nombre de lettres dans l'alphabet :
137 letters[(length(letters)-2):length(letters)]
138
139 # renvoie NA si sélection au-delà de la longueur du vecteur
140 letters[24:27]
141
142 # je veux tout sauf la 1ère lettre
143 letters[-1]
144
145 # je veux tout sauf certains éléments
146 letters[c(-1, -10, -17)]
147
148 # ça marche de la même manière avec un facteur
149 f[1]
150 f[-length(f)]
151
152 # je peux créer un nouvel objet à partir de ma sélection
153 abc <- letters[1:3]
154
155 # et ajouter une valeur à mon vecteur en utilisant l'indexation
156 abc[4] <- "d"
157
158 # remplacer des valeurs existantes
159 abc[1] <- "z"
160
```

The status bar at the bottom shows "108:47", "# Dataframe", and "R Script".

3. L'indexation par position

Exercice de manipulation sur un dataframe :

Reproduire le code ci-contre et observer les résultats



```
162
163 ### dataframe df[i, j]
164 ### exemples avec un df iris fourni par R :
165 iris
166
167 # je cherche la valeur de la première case :
168 # je sélectionne la 1ère ligne
169 # et la 1ère colonne de mon df
170 iris[1,1]
171
172 # je veux extraire la 2ème ligne avec toutes les variables
173 iris[2, ]
174
175 # toutes les lignes de la 5ème colonne
176 iris[, 5]
177 # ou de cette manière
178 iris[[5]]
179
180 # les deux dernières valeurs de la ligne 3
181 iris[3, 4:5]
182
183 # je peux créer un vecteur à partir d'une sélection de mon df
184 species <- iris[[5]]
185
186 # lui retirer des valeurs
187 length(species)
188 species <- species[-c(51:99)]
189 length(species)
190
191 # lui en ajouter
192 species[102:175] <- "chien" # renvoie une erreur : level inconnu dans
193
194 # ajouter d'abord un nouveau level
195 levels(species) <- c(levels(species), "chien")
196 species[102:175] <- "chien"
197
198
168:3 # Indexation par position R Script
```

3. L'indexation par condition

Exercice de manipulation
avec une seule condition :

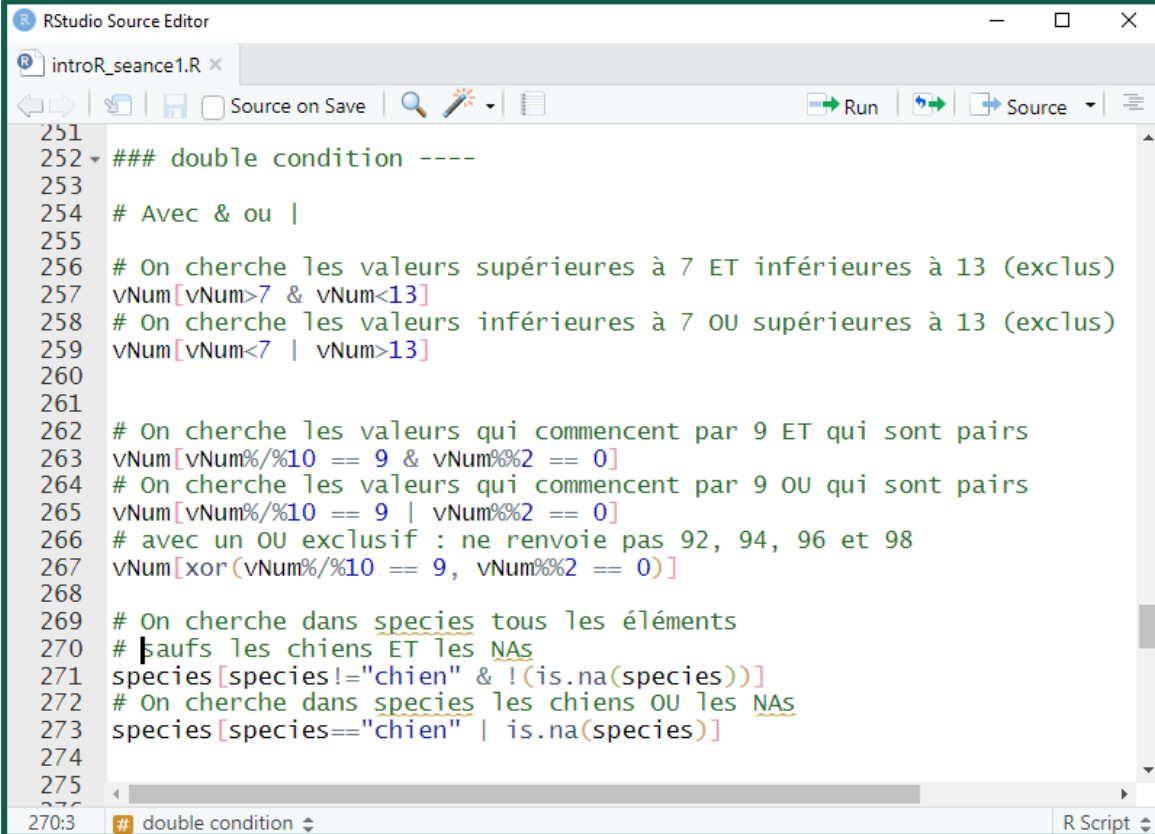
Reproduire le code ci-contre
et observer les résultats

```
RStudio Source Editor
introR_seance1.R x
Source on Save
Run Source
200 ## Indexation par condition ----
201
202 ### une seule condition ----
203
204 # Création d'un nouveau vecteur numérique avec seq()
205 vNum <- seq(from=1, to=100, by=1)
206
207 # Sélection des valeurs supérieures à 51
208 vNum[vNum>51]
209 # Sélection des valeurs égales ou supérieures à 51
210 vNum[vNum>=51]
211
212 # Sélection des valeurs inférieures à 51
213 vNum[vNum<51]
214 # Sélection des valeurs égales ou inférieures à 51
215 vNum[vNum<=51]
216
217 # Sélection des valeurs égales à 25
218 vNum[vNum==25]
219
220 # sélection des valeurs qui ne sont pas égales à 51
221 vNum[vNum!=25]
222
223 # sélection des valeurs qui sont présentes dans un autre vecteur
224 vNum[vNum %in% c(1, 19, 53, 31)]
225 # ou l'inverse
226 vNum[!vNum %in% c(1, 19, 53, 31)]
227
228
229 # Sélection dans un facteur :
230 species[species=="chien"]
231 species[species!="chien"]
232
233 # Sélection dans un vecteur character
234 letters[letters %in% c("a", "b", "c")]
235
236 # Gérer les valeurs manquantes
237 # on introduit d'abord des NA dans un vecteur
238 species[176:180] <- "chat"
239 # Recherche de NA : renvoie TRUE pour chaque élément manquant
240 # FALSE pour chaque élément présent
241 is.na(species)
242 # On ne garde que les éléments avec une valeur
243 #species[!is.na(species)]
244
168:3 # Indexation par position ↕ R Script ↕
```

3. L'indexation par condition

Exercice de manipulation
avec une double condition :

Reproduire le code ci-contre
et observer les résultats



```
251
252 ### double condition ----
253
254 # Avec & ou |
255
256 # On cherche les valeurs supérieures à 7 ET inférieures à 13 (exclus)
257 vNum[vNum>7 & vNum<13]
258 # On cherche les valeurs inférieures à 7 OU supérieures à 13 (exclus)
259 vNum[vNum<7 | vNum>13]
260
261
262 # On cherche les valeurs qui commencent par 9 ET qui sont pairs
263 vNum[vNum%%10 == 9 & vNum%%2 == 0]
264 # On cherche les valeurs qui commencent par 9 OU qui sont pairs
265 vNum[vNum%%10 == 9 | vNum%%2 == 0]
266 # avec un OU exclusif : ne renvoie pas 92, 94, 96 et 98
267 vNum[xor(vNum%%10 == 9, vNum%%2 == 0)]
268
269 # On cherche dans species tous les éléments
270 # sauf les chiens ET les NAs
271 species[species!="chien" & !(is.na(species))]
272 # On cherche dans species les chiens OU les NAs
273 species[species=="chien" | is.na(species)]
274
275
276
```


Mémo sur les opérateurs logiques (1)

Est-ce qu'une proposition est vraie ou fausse ?

Opérateur	Description	Exemple	Résultat
==	identique à	1==2	FALSE
<	strictement inférieur à	1<2	TRUE
>	strictement supérieur à	1>1	FALSE
<=	inférieur ou égal à	2<=5	TRUE
>=	supérieur ou égal à	1>=1	TRUE
!=	différent de	1!=2	TRUE
%in%	présent dans	4 %in% c(1, 2, 3)	FALSE

Mémo sur les opérateurs logiques (2)

Est-ce qu'une proposition est vraie ou fausse ?

Opérateur	Description	Exemple	Résultat
&	et	<code>1<2 & 2>1</code>	TRUE
	ou	<code>1>2 2>1</code>	TRUE
xor()	ou exclusif	<code>xor(1<2, 2>1)</code>	FALSE

<code>is.na()</code>	est manquant	<code>is.na(c(1, 2, NA))</code>	FALSE FALSE TRUE
<code>is.null()</code>	est nul (vide)	<code>is.null(c())</code>	TRUE
<code>isTRUE()</code>	est vrai	<code>isTRUE(FALSE)</code>	FALSE
<code>isFALSE</code>	est faux	<code>isFALSE(FALSE)</code>	TRUE
!	l'inverse de	<code>!(is.na(NA))</code>	FALSE
		<code>!4 %in% c(1, 2, 3)</code>	TRUE

4. Fonctions de base : structure des données

Fonctions de base pour explorer la structure et le contenu de ses données :

Obtenir des infos sur une fonction chargée dans la session, ex :
`?length`

class() : renvoie le type d'objet (numeric, character...)

str() : renvoie la structure (type d'objet, contenu, nature des variables)

```
> str(df)
'data.frame': 3 obs. of 3 variables:
 $ NOM      : chr  "Joséphin" "Léa" "Aurélie"
 $ FEMME    : logi  FALSE TRUE TRUE
 $ POINTURE : num  41.5 38 37
```

dim() : renvoie la dimension d'un df (n ligne et n colonne)

```
> dim(df)
[1] 3 3
```

length() : renvoie la longueur de l'objet (n éléments d'un vecteur ; n colonne d'un df)

nrow() : renvoie le nombre de ligne d'un df

```
> nrow(df)
[1] 3
```

print() : renvoie le contenu d'un objet

View() : affiche l'objet dans le volet « édition source »

5. Fonctions de base : description des variables

Fonctions de base pour explorer et décrire des données quantitatives

Obtenir des infos sur
une fonction chargée
dans la session, ex :
?IQR

summary() : renvoie résumé stat d'un vecteur

```
> summary(vNum)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.00  25.75   50.50   50.50  75.25  100.00
```

min() : minimum

max() : maximum

mean() : moyenne

median() : médiane

sd() : écart-type

IQR() : intervalle interquartile

quantile() : quantile

```
> # par défaut, quantiles
> quantile(vNum)
  0%    25%   50%   75%  100%
 1.00 25.75 50.50 75.25 100.00
```

```
> # déciles
> quantile(vNum, probs = seq(from = 0, to = 1, by = .1 ))
  0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
 1.0 10.9 20.8 30.7 40.6 50.5 60.4 70.3 80.2 90.1 100.0
```

6. Manipuler un dataframe

Pour voir les 10 premières/dernières lignes de son tableau : head() / tail()

Pour l'ouvrir et le manipuler : view() (ou cliquer sur son nom dans l'environnement)

Un tableau a deux dimensions : df[Ligne, Colonne] :

Double crochet : colonne

```
> df[[1]]  
[1] "Joséphin" "Léa" "Aurélie"
```

```
> df[1,]  
      NOM FEMME POINTURE  
1 Joséphin FALSE    41.5
```

Ligne : [L,]

Colonne : [,C]

```
> df[,1]  
[1] "Joséphin" "Léa" "Aurélie"
```

	NOM	FEMME	POINTURE
1	Joséphin	FALSE	41.5
2	Léa	TRUE	38.0
3	Aurélie	TRUE	37.0

```
> df[c(1,3),]  
      NOM FEMME POINTURE  
1 Joséphin FALSE    41.5  
3  Aurélie  TRUE    37.0
```

Sélection multiple avec c()

Nom de la colonne

```
> df["NOM"]  
      NOM  
1 Joséphin  
2 Léa  
3 Aurélie
```

```
> df[c(1, 2), c(2,3)]  
      FEMME POINTURE  
1 FALSE    41.5  
2  TRUE    38.0
```

```
> df[3,2]  
[1] TRUE
```

Ligne et colonne : [L,C]

Sélection multiple avec c()

6. Manipuler un dataframe

Pour ajouter une ligne, on crée une liste que l'on lie à notre df avec rbind()

```
> nouveau <- list("Hugues", F, 45)
> df <- rbind(df, nouveau)
> df
```

	NOM	FEMME	POINTURE
1	Joséphin	FALSE	41.5
2	Léa	TRUE	38.0
3	Aurélie	TRUE	37.0
4	Hugues	FALSE	45.0

Pour ajouter une colonne, cbind() suit la même logique

```
> taille <- c(169, 168, 167, 180)
> cbind(df, taille)
```

	NOM	FEMME	POINTURE	taille
1	Joséphin	FALSE	41.5	169
2	Léa	TRUE	38.0	168
3	Aurélie	TRUE	37.0	167
4	Hugues	FALSE	45.0	180

Le \$ permet aussi d'accéder aux colonnes

```
> df$taille <- c(169, 168, 167, 180)
```

Par défaut, le nom des lignes sont un numéro, on y accède par row.names()

```
> row.names(df)
[1] "1" "2" "3" "4"
```

On peut facilement créer une colonne identifiant en prenant le nom des lignes

```
> df$id <- row.names(df)
> df
```

	NOM	FEMME	POINTURE	taille	id
1	Joséphin	FALSE	41.5	169	1
2	Léa	TRUE	38.0	168	2
3	Aurélie	TRUE	37.0	167	3
4	Hugues	FALSE	45.0	180	4

Le nom des colonnes peut être manipulé avec la fonction names()

```
> names(df)[c(1, 4)] = c("PRENOM", "TAILLE")
> df
```

	PRENOM	FEMME	POINTURE	TAILLE	id
1	Joséphin	FALSE	41.5	169	1
2	Léa	TRUE	38.0	168	2
3	Aurélie	TRUE	37.0	167	3
4	Hugues	FALSE	45.0	180	4

6. Manipuler un dataframe

On peut faire des opérations sur les colonnes numériques

```
> df$calcul <- df$TAILLE - df$POINTURE
> df
  PRENOM FEMME POINTURE TAILLE id calcul
1 Joséphin FALSE    41.5   169  1  127.5
2 Léa TRUE     38.0    168  2  130.0
3 Aurélie TRUE    37.0    167  3  130.0
4 Hugues FALSE    45.0    180  4  135.0
```

```
> df$POINTURE <- df$POINTURE + 1
> df
  PRENOM FEMME POINTURE TAILLE id calcul
1 Joséphin FALSE    42.5   169  1  127.5
2 Léa TRUE     39.0    168  2  130.0
3 Aurélie TRUE    38.0    167  3  130.0
4 Hugues FALSE    46.0    180  4  135.0
```

La fonction order() trie l'ensemble du tableau selon la colonne voulue

```
> df <- df[order(df$TAILLE, decreasing = TRUE), ]
> df
  PRENOM FEMME POINTURE TAILLE id calcul
4 Hugues FALSE    46.0   180  4  135.0
1 Joséphin FALSE    42.5   169  1  127.5
2 Léa TRUE     39.0   168  2  130.0
3 Aurélie TRUE    38.0   167  3  130.0
```

La sélection permet aussi de réorganiser l'ordre des colonnes

```
> df <- df[, c("id", "PRENOM", "TAILLE")]
> df
  id PRENOM TAILLE
4  4 Hugues   180
1  1 Joséphin  169
2  2 Léa     168
3  3 Aurélie  167
```

Programme



Qu'est-ce qu'on peut faire avec R ?



R et RStudio : c'est quoi la différence ?



Créer un projet et un script



Manipuler des objets : assignation, indexation, fonctions



Les packages

C'est quoi un package ?

R base

mean() , print() ou encore length() sont des fonctions incluses dans la base du langage R.

Elle font parties des 7 packages standards installés et chargés automatiquement dans R.

→ Installer R suffit pour utiliser les fonctions des packages :

- base
- utils
- stats
- grDevices
- Graphics
- methods
- datasets



Packages à ajouter

De nombreuses fonctions ont été développées par la communauté R. Elles sont regroupées dans différents packages que l'on peut installer et charger dans R en fonction des besoins.

→ Il faut installer et charger ces packages dans R pour pouvoir utiliser ces fonctions. Par exemple les packages :

- mapsf
- ggplot2
- openxlsx
- questionR
- palmerpenguins
- ...

Le CRAN

CRAN est un réseau de serveurs à travers le monde qui stockent des versions identiques à jour du code et de la documentation de R. On y retrouve par exemple :

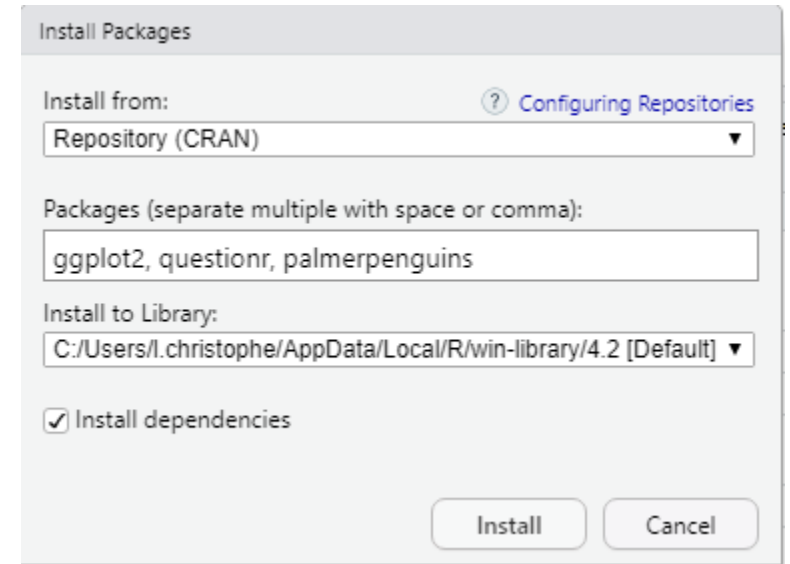
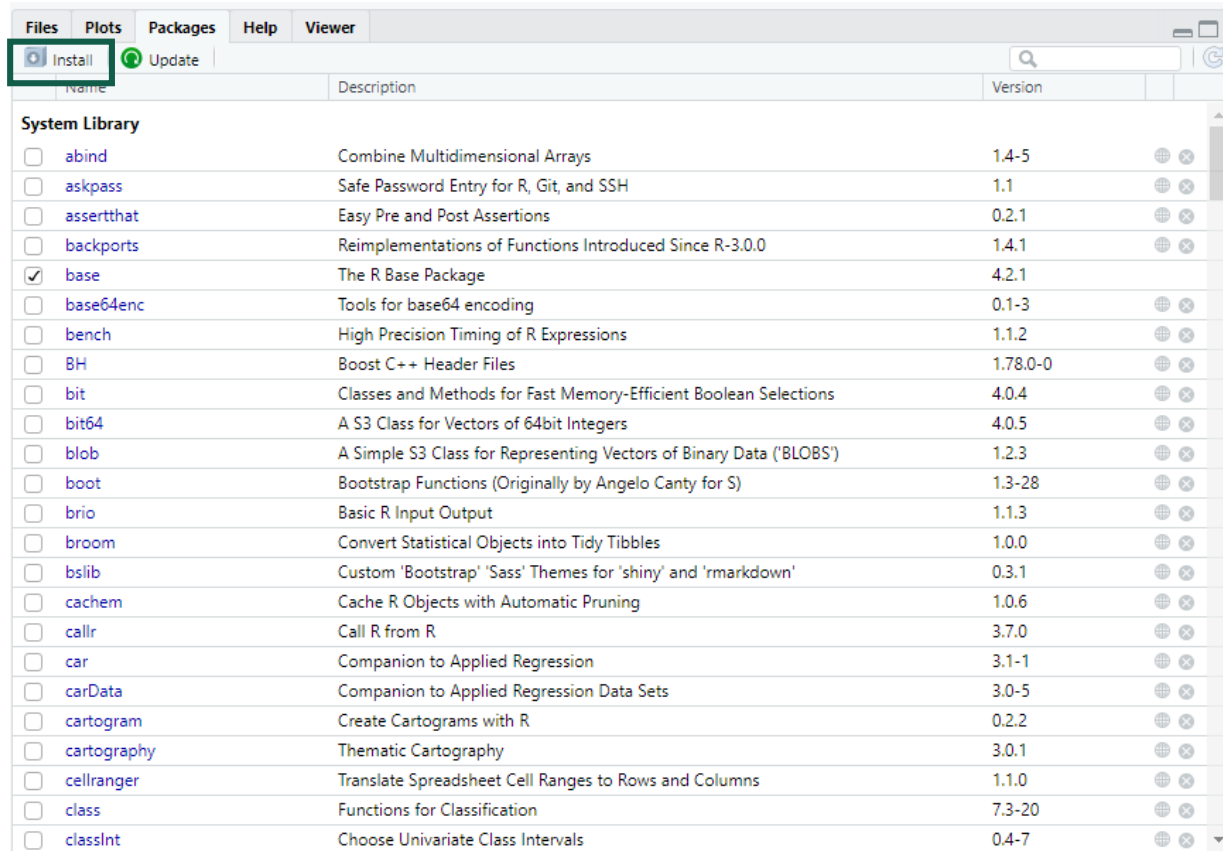
- Le logiciel R
- La liste des packages déposés par les développeurs
- De la documentation sur ces packages



→ Depuis RStudio, nous allons pouvoir “aller chercher” les packages disponibles sur le CRAN pour les installer.

Comment installer-t-on un package ?

1^{ère} solution : Depuis la fenêtre de chargement des packages



Comment installe-t-on un package ?

2^{de} solution : En code

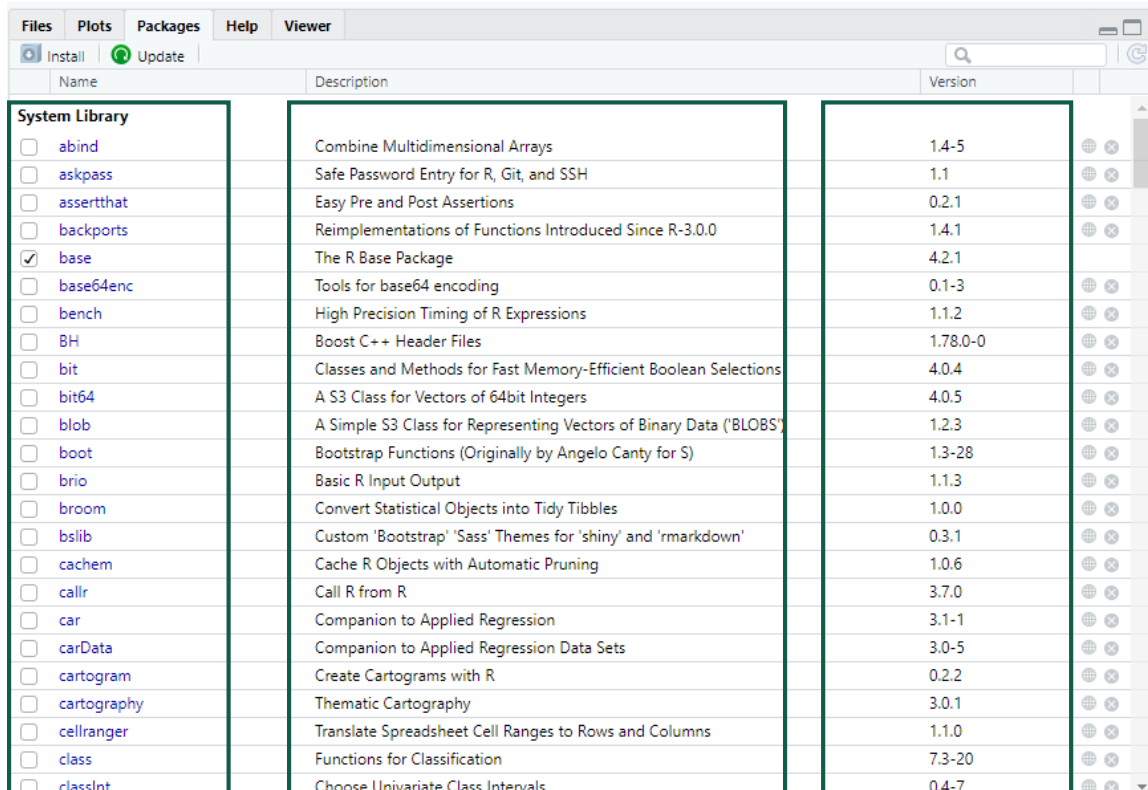
```
package_a_installer <- c('ggplot2', 'questionr', 'palmerpenguins')  
install.packages(package_a_installer)
```

=

```
install.packages(c('ggplot2', 'questionr', 'palmerpenguins'))
```

Quels sont les packages installés ?

Depuis la fenêtre de chargement des packages



The screenshot shows the R package manager window with a list of packages. The 'base' package is checked, indicating it is installed. The table below summarizes the visible data from the screenshot.

NOM	DESCRIPTION	VERSION
System Library		
<input type="checkbox"/> abind	Combine Multidimensional Arrays	1.4-5
<input type="checkbox"/> askpass	Safe Password Entry for R, Git, and SSH	1.1
<input type="checkbox"/> assertthat	Easy Pre and Post Assertions	0.2.1
<input type="checkbox"/> backports	Reimplementations of Functions Introduced Since R-3.0.0	1.4.1
<input checked="" type="checkbox"/> base	The R Base Package	4.2.1
<input type="checkbox"/> base64enc	Tools for base64 encoding	0.1-3
<input type="checkbox"/> bench	High Precision Timing of R Expressions	1.1.2
<input type="checkbox"/> BH	Boost C++ Header Files	1.78.0-0
<input type="checkbox"/> bit	Classes and Methods for Fast Memory-Efficient Boolean Selections	4.0.4
<input type="checkbox"/> bit64	A S3 Class for Vectors of 64bit Integers	4.0.5
<input type="checkbox"/> blob	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')	1.2.3
<input type="checkbox"/> boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28
<input type="checkbox"/> brio	Basic R Input Output	1.1.3
<input type="checkbox"/> broom	Convert Statistical Objects into Tidy Tibbles	1.0.0
<input type="checkbox"/> bslib	Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown'	0.3.1
<input type="checkbox"/> cachem	Cache R Objects with Automatic Pruning	1.0.6
<input type="checkbox"/> callr	Call R from R	3.7.0
<input type="checkbox"/> car	Companion to Applied Regression	3.1-1
<input type="checkbox"/> carData	Companion to Applied Regression Data Sets	3.0-5
<input type="checkbox"/> cartogram	Create Cartograms with R	0.2.2
<input type="checkbox"/> cartography	Thematic Cartography	3.0.1
<input type="checkbox"/> cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
<input type="checkbox"/> class	Functions for Classification	7.3-20
<input type="checkbox"/> classInt	Choose Univariate Class Intervals	0.4-7

En ligne de code

```
> installed.packages()
```

```
> installed.packages()
  Package      LibPath      Version
abind         "abind"      "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "1.4-5"
askpass       "askpass"    "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "1.1"
assertthat    "assertthat" "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "0.2.1"
backports     "backports"  "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "1.4.1"
base64enc     "base64enc"  "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "0.1-3"
bench         "bench"      "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "1.1.2"
BH            "BH"         "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "1.78.0-0"
bit           "bit"        "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "4.0.4"
bit64         "bit64"      "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "4.0.5"
blob          "blob"       "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "1.2.3"
brio          "brio"       "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "1.1.3"
broom         "broom"      "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "1.0.0"
bslib         "bslib"      "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "0.3.1"
cachem        "cachem"     "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "1.0.6"
callr         "callr"      "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "3.7.0"
car           "car"        "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "3.1-1"
carData       "carData"    "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "3.0-5"
cartogram     "cartogram"  "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "0.2.2"
cartography   "cartography" "C:/Users/l.christophe/AppData/Local/R/win-library/4.2" "3.0.1"
```

Pensez à mettre à jour vos packages régulièrement. Via le bouton Update de la fenêtre de chargement vous pouvez vérifier si des mises à jour sont disponibles. La mise à jour se fait de la même manière que l'installation.

Installation et chargement : c'est quoi la différence ?

C'est quoi la différence entre **installer** et **charger** ?

- Installer : télécharger le package sur internet, puis installation sur l'ordinateur (dans un dossier connu de R)
- Charger : indiquer à R le(s) package(s) que l'on souhaite utiliser dans la session en cours.

→ Il n'est pas nécessaire d'installer le package à chaque fois MAIS il est obligatoire de charger le package dès qu'on lance une nouvelle session de R (lorsque l'on ouvre le logiciel).

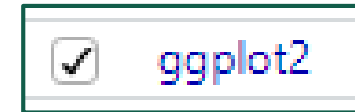
Exemple : Nous avons installé le package « ggplot2 », ce package contient les fonctions ggplot() et geom_col() que l'on souhaite utiliser pour faire un graphique à partir du tableau df créé précédemment. Peut-on utiliser cette fonction ?

```
> ggplot(df) + geom_col(aes(x = NOM, y = POINTURE))  
Error in ggplot(df) : could not find function "ggplot"
```

→ **NON** il faut charger le package

Comment charge-t-on un package ?

1^{ère} solution : Depuis la fenêtre de chargement des packages en cliquant sur la case à gauche (fortement déconseillé pour la reproductibilité du code)



Comment charge-t-on un package ?

2^{ème} solution : en code

Un par un

```
library('ggplot2')  
library('questionr')  
library('palmerpenguins')
```

OU

Pour plusieurs packages

```
lapply(c('ggplot2', 'questionr', 'palmerpenguins'), library, character.only = TRUE)
```


Exemple : Utilisation d'un package

Sur l'exemple précédent, il était impossible d'utiliser les fonctions `ggplot()` et `geom_col()` du package `ggplot2`

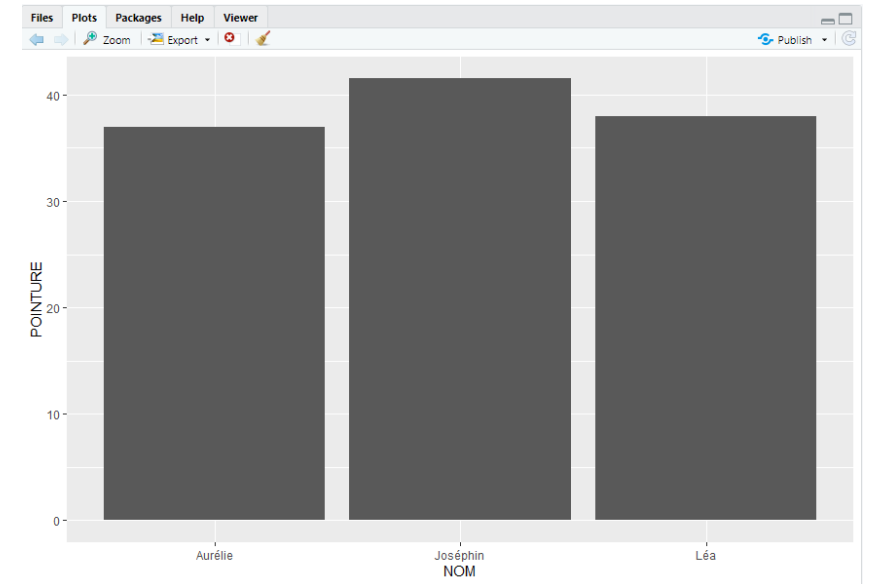
```
> ggplot(df) + geom_col(aes(x = NOM, y = POINTURE))  
Error in ggplot(df) : could not find function "ggplot"
```

→ Erreur précédente

Après chargement, le code s'exécute sans erreur :

```
>>>  
> library('ggplot2')  
> ggplot(df) + geom_col(aes(x = NOM, y = POINTURE))  
>>>
```

→ Résultat du code :



Trouver de l'aide pour utiliser un package : le CRAN

Sur le CRAN il est possible de trouver la documentation complete du package.

Par exemple :

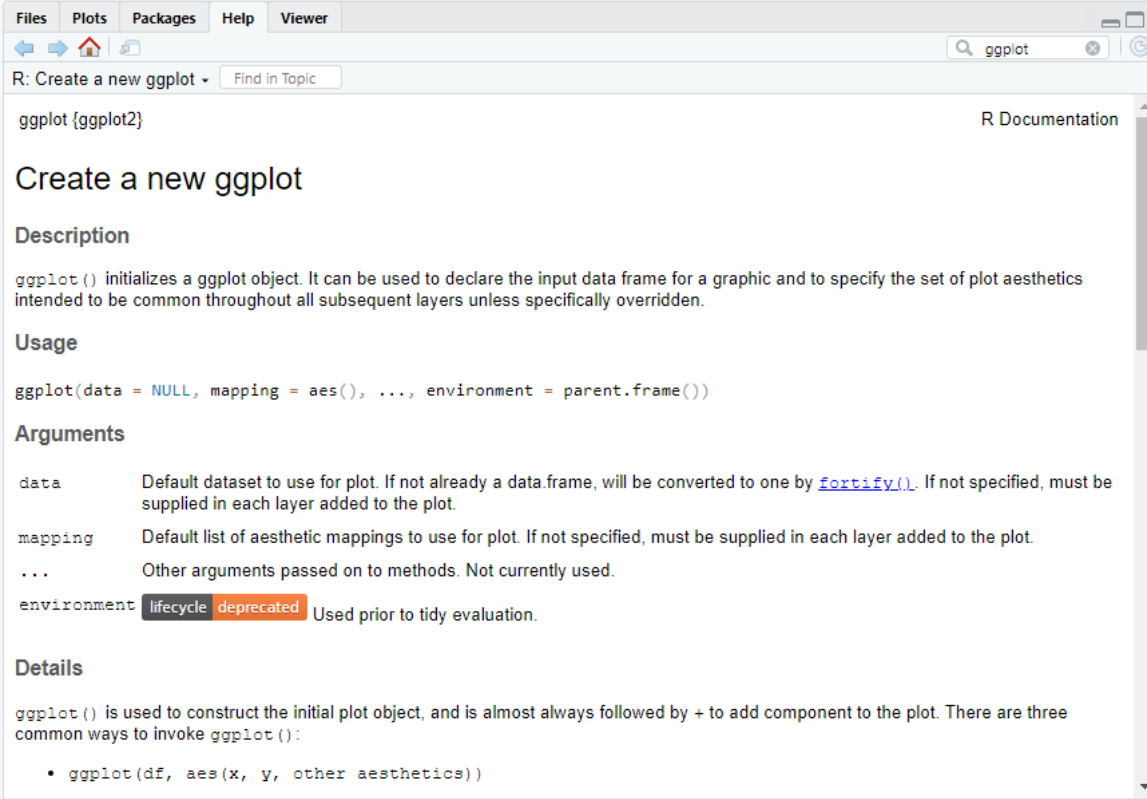
- Pour ggplot2 <https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf>
- Pour cartography <https://cran.r-project.org/web/packages/cartography/cartography.pdf>

Sur le forum Stack Overflow : <https://stackoverflow.com/>

Pour trouver de l'aide sur une fonction precise il est possible (une fois le package chargé):

- Via une ligne de code : `?ggplot2`
- d'utiliser la fenêtre "Help" dans Rstudio

→ **Un bon package est un package documenté.**



The screenshot shows the R Documentation page for the `ggplot` function in the `ggplot2` package. The page is titled "Create a new ggplot" and includes sections for Description, Usage, Arguments, and Details. The Description section states that `ggplot()` initializes a `ggplot` object. The Usage section shows the function signature: `ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())`. The Arguments section lists `data`, `mapping`, `...`, and `environment` (marked as deprecated). The Details section explains that `ggplot()` is used to construct the initial plot object and is followed by `+` to add components. A common invocation is shown as `ggplot(df, aes(x, y, other_aesthetics))`.

Trouver de l'aide pour utiliser un package : les cheat sheets

Les *cheat sheets* ou feuilles de triche offrent une synthèse des fonctions disponibles dans un package.

Data Visualization with ggplot2 : CHEAT SHEET



Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms** - visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (aesthetics) like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

```
ggplot(data = DATA) +
  geom_function(mapping = aes(MAPPING)) +
  coord_function() +
  facet_function() +
  scale_function() +
  theme_function()
```

ggplot2(data = mpg, aes(x = city, y = hwy)) Draws a plot that you finish by adding layers. Add size geom function per layer.

geom_function maps **data** to **geom**

coord(x = city, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("my_plot.png", width = 5, height = 8) Saves last plot as 5 x 8 inch named "my_plot.png" in working directory. Matches file type to file extension.

Geoms Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function inherits a layer.

GRAPHICAL PRIMITIVES (Default for expanding limits)

```
a = geom_blank()
b = geom_curve(aes(xend = lat + 1, yend = hwy + 1, aes(linetype = "solid", color = "red", size = 2)))
a = geom_point(aes(x = "Paris", y = 48, size = 100, color = "red", group = "Paris"))
a = geom_polygon(aes(group = "group", x, y, alpha, color, fill, linetype, size, stroke))
b = geom_rect(aes(xmin = long, xmax = long + 1, ymin = hwy, ymax = hwy + 1, aes(linetype = "solid", color = "red", size = 2)))
a = geom_ribbon(aes(ymin = unemploj, ymax = unemploj + 10, aes(linetype = "solid", color = "red", size = 2)))
```

LINE SEGMENTS

```
b = geom_abline(aes(intercept = slope, x = 1, y = 1))
b = geom_line(aes(intercept = lat))
b = geom_path(aes(xorder = long, y = hwy))
b = geom_segment(aes(xend = lat + 1, yend = hwy + 1))
b = geom_spath(aes(geom = "line", color = "red", size = 2))
```

ONE VARIABLE continuous

```
c = geom_area(aes("area", x, y, alpha, color, fill, linetype, size))
c = geom_density(aes("density", x, y, alpha, color, fill, group, linetype, size, weight))
c = geom_density_2d(aes("2d", x, y, alpha, color, fill, linetype, size, weight))
c = geom_dotplot(aes("dotplot", x, y, alpha, color, fill))
c = geom_histogram(aes("hist", x, y, alpha, color, fill, linetype, size, weight))
c = geom_histogram_binwidth(aes("binwidth", x, y, alpha, color, fill, linetype, size, weight))
c = geom_histogram_binwidth2(aes("binwidth2", x, y, alpha, color, fill, linetype, size, weight))
```

ONE VARIABLE discrete

```
d = geom_bar()
d = geom_bar(aes("bar", x, y, alpha, color, fill, linetype, size, weight))
```

TWO VARIABLES continuous x, continuous y

```
e = ggplot(mpg, aes(city, hwy))
a = geom_label(aes(label = city, radius = 1, aes(x, y, alpha, color, fill, linetype, size, weight))
a = geom_density_2d(aes("density", x, y, alpha, color, fill, linetype, size, weight))
a = geom_hex(aes("hex", x, y, alpha, color, fill, size))
```

continuous bivariate distribution

```
f = ggplot(mtcars, aes(wt, mpg))
a = geom_bland(aes(wt, mpg, aes(x = wt, y = mpg, aes(alpha, color, fill, linetype, size, weight))
a = geom_density_2d(aes("density", x, y, alpha, color, fill, linetype, size, weight))
a = geom_hex(aes("hex", x, y, alpha, color, fill, size))
```

continuous function

```
i = geom_area(aes("area", x, y, alpha, color, fill, linetype, size))
i = geom_line(aes("line", x, y, alpha, color, group, linetype, size))
i = geom_step(aes("step", x, y, alpha, color, group, linetype, size))
```

visualizing error

```
o = data.frame(x = c("A", "B"), fit = 4.5, se = 1.2)
j = ggplot(o, aes(x, fit, ymin = fit - se, ymax = fit + se))
j = geom_crossbar(aes("crossbar", x, y, ymin, ymax, alpha, color, fill, group, linetype, size))
j = geom_errorbar(aes("errorbar", x, y, ymin, ymax, alpha, color, group, linetype, size, width))
j = geom_linerange(aes("linerange", x, y, ymin, ymax, alpha, color, group, linetype, size))
j = geom_pointrange(aes("pointrange", x, y, ymin, ymax, alpha, color, fill, group, linetype, size, shape, size))
j = geom_violin(aes("violin", x, y, alpha, color, fill, group, linetype, size, weight))
```

maps

```
data = data.frame(ordered = USStates$ordered, state = ordered(lower.names(USStates)))
map = map_data("state")
k = ggplot(data, aes(long, lat))
k = geom_map(aes(map_id = state, map = map, expand_limits(aes(mapping = "map_data", map_id, alpha, color, fill, linetype, size))
```

THREE VARIABLES

```
soaSz = with(mtcars, sqrt((cyl * long)^2 + (displ * lat)^2))
l = geom_contour(aes("contour", x, y, alpha, color, fill, linetype, size, weight))
l = geom_raster(aes("raster", x, y, alpha, color, fill, linetype, size, weight))
l = geom_tile(aes("tile", x, y, alpha, color, fill, linetype, size, weight))
```

Thematic maps with mapsf : CHEAT SHEET



Create and integrate thematic maps in your workflow.

Base map

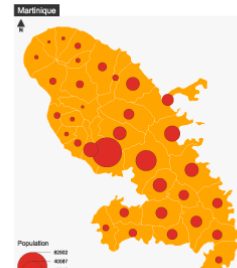
Import library
Library(mapsf)

Import the sample data set
mtq <- mf_get_mtg()

Initiate a base map centered on a specific extent
mf_map(x = mtq, col = "orange", border = "white")

Plot styling
mf_map(x = mtq, type = "prop", var = "POP", leg_title = "Population", add = TRUE)

Complete layout (credits, title, north arrow, scale bar)
mf_layout(credits = "Maptastic", credits = "Sources: IGN, 2018")



Colors

mapsf can use color palettes from **hcl.colors()**.
mf_get_pal() is useful to create well-balanced asymmetric diverging palettes

```
mf_get_pal(n = c(7, 2), pal = c("burg", "mint"))
```

Symbology

The x argument should be an sf object. Input geometries can be polygons, lines or points.

Choropleth (ratio)
mf_map(x = mtq, type = "choro", var = "var", method = "quantile")

Typology (category)
mf_map(x = mtq, type = "type", var = "var", inches = 0.1, symbol = "circle")

Proportional Symbols (stock)
mf_map(x = mtq, type = "prop", var = "var", inches = 0.1, symbol = "circle")

Graduated Symbols (stock)
mf_map(x = mtq, type = "grad", var = "var", pch = 24)

Symbols (category)
mf_map(x = mtq, type = "sym", var = "var", pch = c(21:23))

Choropleth proportional symbols (stocks - ratio)
mf_map(x = mtq, type = "prop_choro", var = c("var1", "var2"))

Colorized proportional symbols (stocks - cate)
mf_map(x = mtq, type = "prop_choro", var = c("var1", "var2"))

Choropleth symbols (ratio - category)
mf_map(x = mtq, type = "ratio_choro", var = c("var1", "var2"), pch = c(21:23))

Raster
mf_raster(x = raster)

Legends

Default legends are plotted along maps. Customization parameters are available with mf_legend()



Map Layout

Along with cartographic functions, other functions are dedicated to customize the layout design.



- Set a map theme (figure margins, colors, etc.)
mf_theme(theme = "mapsf", size = TRUE, mar = c(0,0,0,0), pos = "left")
Auto themes are available: default, ink, dark, applight, candy, default, color, green, newsworld, jet or basemap.
- Init a map centered on a specific area
mf_init(x = mtq, col = "orange", border = "white", add = TRUE)
- Import external image for background
mf_background(filename = "img/sea.jpg")
- Create a shadow effect
mf_shadow(...)
- Create a custom inset
mf_inset_on(x = mtq, pos = "bottomleft", mf_inset_off())
- Create a world inset
mf_inset_on(x = "worldmap", mf_worldmap(mtg), mf_inset_off())
- Plot title
mf_title("A beautiful beach")
- Plot labels
mf_label(...)
- Plot annotation (in specific locations)
mf_annotation(...)
- North arrow
mf_arrow(...)
- Scale (in km)
mf_scale(...)
- Credits
mf_credits(...)

Export Maps

mf_export() exports maps in PNG or SVG formats.

The exported map width/height ratio will match the one of a spatial object.

Additionally, mf_export() can be used to set a theme, to extend the map space on one or several side of the figure, or to center a map on a specific area.

Simple export (PNG)
mf_export(x = mtq, width = 500, filename = "my_export.png", dev.off())

Export with a theme (SVG)
mf_export(x = mtq, width = 5, export = "svg", theme = "newsworld", dev.off())

Extra space on the figure (bottom, left, top, right)
mf_export(x = mtq, width = 500, filename = "my_export.png", dev.off())

Export a map centered on a specific area
mf_export(x = mtq, add = TRUE, filename = "my_export.png", dev.off())

Further documentation

- Vignettes on mapsf website: rielab.github.io/mapsf
- Get started
 - How to Use Themes
 - How to Export Maps
 - How to Create Inset Maps
 - How to Create Faceted Maps
 - How to Use a Custom Font Family



Bonus

Obtenir des informations sur les versions utilisées

❖ Connaître sa version R installée :

- À l'ouverture de RStudio dans la console
- Avec la commande `R.version()`
- Clic bouton : `Tools>General>Basic`

❖ Connaître sa version Rstudio installée :

- Avec la commande `rstudioapi::versionInfo()`
- Clic bouton : `Help>About RStudio`

❖ Connaître les versions R et packages chargées dans sa session :

- `sessionInfo()`

❖ Connaître les versions de packages installées :

- Dans le volet en bas à droite, onglet Packages
- Clic bouton : `Tools>Check for Packages Updates...`
- Avec la commande `installed.packages()`

❖ Guide de mise à jour de R et de RStudio :

- <https://delladata.fr/tutoriel-mise-a-jour-de-r/>