TP OUMOBIO 1: Introduction à l'environnement R-Rstudio

Mathieu Brevet

2024-09-26

Bienvenue dans ce premier TP sur R ! Cette première séance aura pour but de se familiariser avec les environnements R et Rstudio, en apprenant à créer et gérer son espace de travail.

Qu'est-ce que R ?

R est un **langage de programmation** spécialisé dans le **traitement des données** et leur **analyse statistique**. R permet notamment de réaliser des opérations mathématiques ou logiques et contient de nombreuses fonctions de manipulation/gestion de données et de traitement statistiques. Ces fonctions / opérations se réalisent sur des objets divers allant de tableaux de données à différents types de variables (numériques, entières, chaines de caractères ou booléens). Les fonctions prennent des objets en entrée ainsi que différentes options (le tout formant l'ensemble des arguments de la fonction) et permettent d'obtenir en sortie de nouveaux objets, des objets modifiés ou des informations sur la nature de ces derniers. R contient un ensemble de fonction de base présentes par défaut dans l'environnement, mais il existe également des fonctions plus poussées disponibles sur R en téléchargeant des "paquets" thématiques dans votre environnement de travail.

Au cours de ces TP nous allons utiliser une table de données composée de différentes variables et l'analyser en utilisant divers outils statistiques

Créer un environnement de travail

Créer un dossier de travail

La première étape est de créer un dossier sur votre espace réseau, sur lequel vous allez déposer tous les fichiers utiles à vos travaux et enregistrer vos codes et les sorties du logiciel R.

La marche à suivre est la suivante:
Positionnez vous dans votre espace réseau (cliquez sur l'explorateur de fichier, allez dans "Ce PC", puis dans "Espace personnel") → **Créez un nouveau dossier** (attention à lui donner un nom explicite: voir encadré ci-dessous) → **Venez y déposer le jeu de donnée** que nous utiliserons au cours de ces séances de TP ("Suivi_lezard_vivipare.csv"), présent sur E-learn

BONNES PRATIQUES

Prenez garde à donner des **noms explicites** à vos dossiers, fichiers de travail et objet R, qui vous permettront de facilement identifier de quoi il s'agit. **Évitez d'utiliser des caractères spéciaux ou des espaces** dans vos noms de documents ou de dossier: ces derniers risquerait d'être mal lu par le logiciel. À la place des espaces préférez les symboles "-" ou "_" et utilisez des chiffres et lettres sans caractères, en majuscule ou non. Un exemple de bonne dénomination dans le cas présent serait: "TP-OUMOBIO_Semestre1".

Définir son environnement de travail sur R

Maintenant que votre dossier de travail est prêt il vous faut indiquer à R l'endroit depuis lequel vous souhaitez travailler et charger sur R les fichiers nécessaires pour vos analyses.

La première étape est d'ouvrir le logiciel Rstudio qui est une interface de développement du langage de programmation informatique R, vous permettant de gérer vos données, écrire du code R et manipuler différents objets R.

Vous allez tout d'abord ouvrir un nouveau fichier **script R** (fichier où vous allez écrire et enregistrer toutes vos lignes de codes) en suivant le chemin suivant:

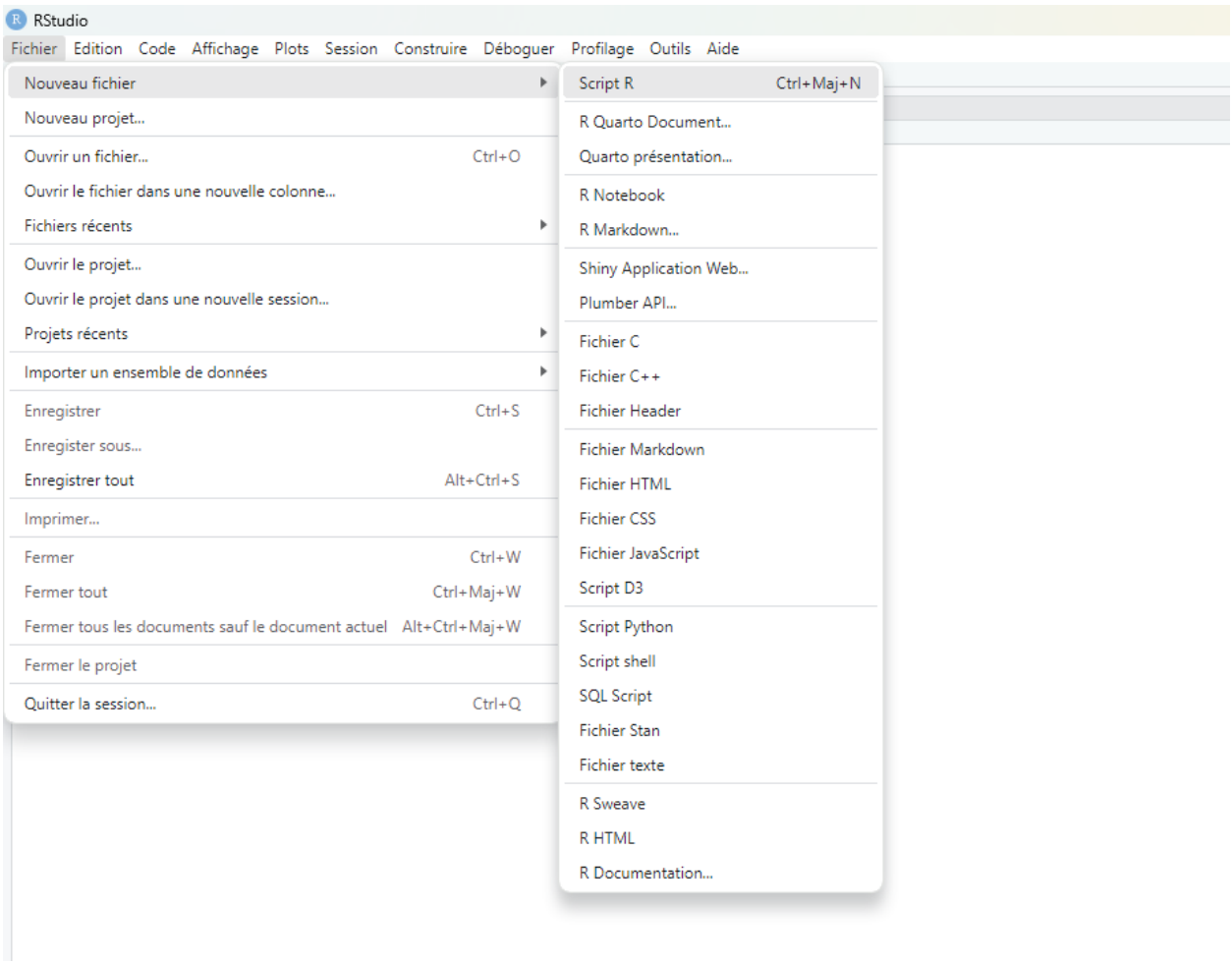

Figure 1: Création d'un script R

Vous retrouverez l'interface suivante:

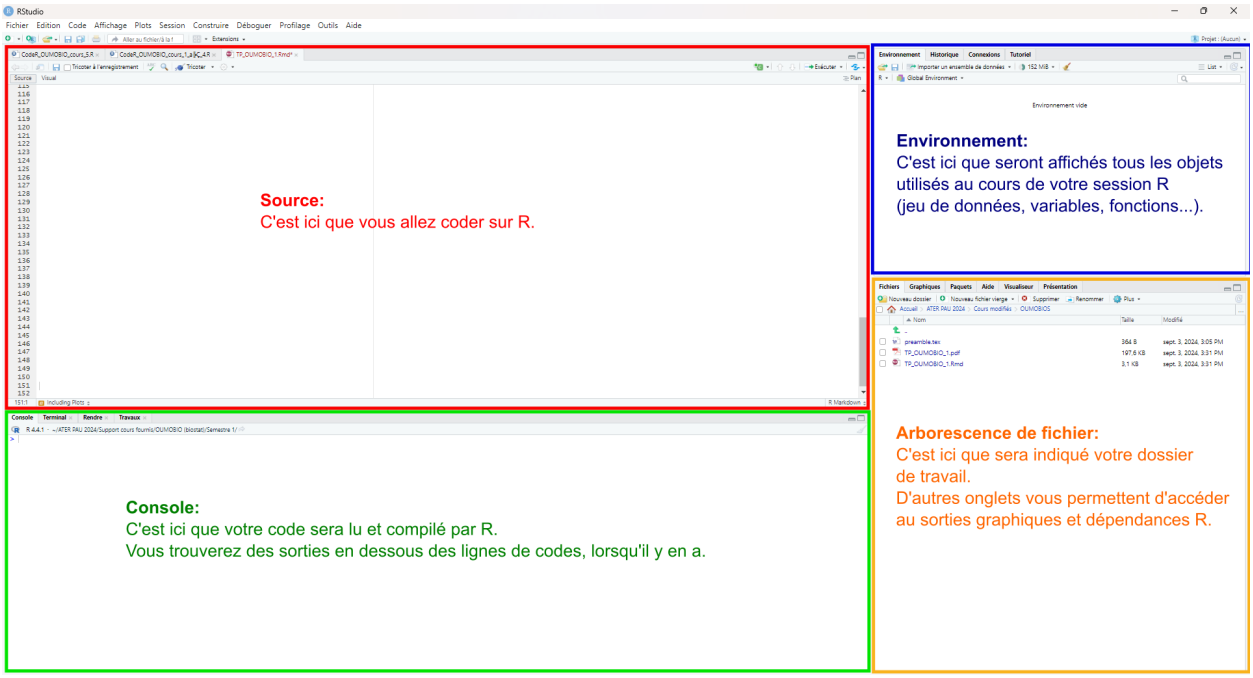

Figure 2: Panneaux Rstudio

Nous allons commencer par indiquer à R notre dossier de travail pour que l'on puisse par la suite importer des fichiers de travail depuis ce dossier vers Rstudio et enregistrer nos scripts et sorties depuis Rstudio vers ce dossier. Voici les commandes à écrire dans votre code source puis à copier dans votre console R:

```
setwd("~/ATER PAU 2024/Cours modifiés/OUMOBIOS")
# 'setwd()' est une fonction R, elle permet de définir l'espace de travail en
# prenant en argument la voie d'accès au dossier de travail

getwd()
```

```
## [1] "C:/Users/savma/Documents/ATER PAU 2024/Cours modifiés/OUMOBIOS"
```

```
# cette commande permet de vérifier dans quel dossier de travail vous vous
# trouvez
```

ASTUCES

L'espace de travail peut aussi être défini directement à partir de Rstudio en utilisant l'arborescence de fichier, il vous suffit de vous positionner dans votre dossier de travail puis de le définir comme votre répertoire de travail (voir image ci-dessous). Cela peut être utile lorsque vous ne connaissez pas parfaitement la voie d'accès à votre dossier de travail.

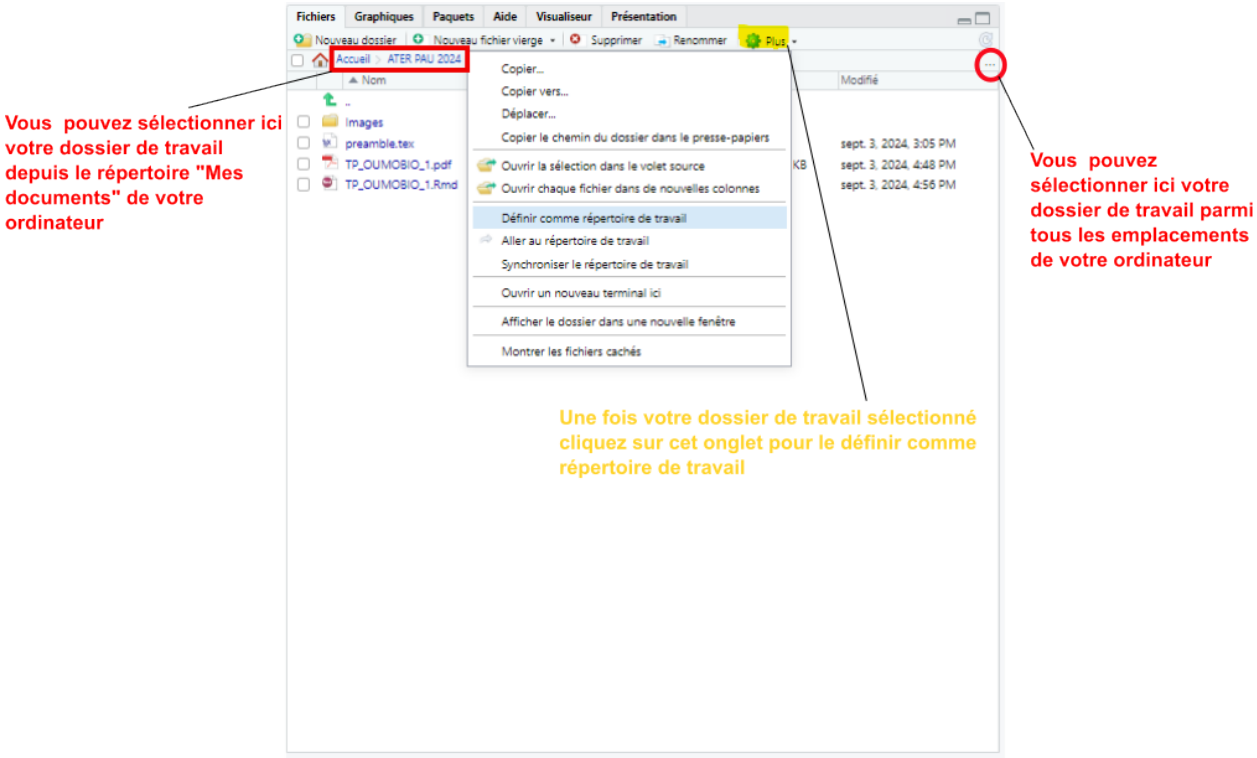

Figure 3: Sélection répertoire de travail

Importer un fichier de données dans R

Maintenant l'espace de travail défini, nous allons pouvoir importer notre **jeu de données** (sous forme d'un tableau de données) dans R. Ce jeu de données correspond à un suivi scientifique du lézard vivipare (*Zootoca vivipara*) en France, dans le massif des Cévennes. Il s'agit plus particulièrement d'un suivi des individus dès leur naissance, et d'une étude de leur choix de dispersion en fonction de leur environnement social. Ce jeu de données va nous permettre de nous **poser différentes questions scientifiques** quant aux phénotypes et aux choix de dispersion de ces individus juvéniles, à partir d'un échantillon de la population de lézard vivipare dans les Cévennes.

Pour importer ce jeu de données dans R nous allons utiliser la fonction **"read.table()"**. Commençons par en apprendre un peu plus sur cette fonction, pour cela entrez dans la console la commande suivante:

```
help("read.table")
# fonctionne également avec: ?read.table
```

La **commande "?"** suivi d'un nom de fonction permet d'afficher l'**aide de la fonction** dans le panel inférieur droit de votre écran, on peut ainsi voir les différents arguments que prend la fonction, avec une description pour chacun d'entre eux, ainsi qu'une description et des détails sur le fonctionnement de la fonction, puis les sorties de cette fonction. Pour cette fonction il y a trois arguments clés à retenir:

- "header": définit la présence ou non de noms de colonnes dans le jeu de données
- "sep": définit quel type de séparateur est utilisé entre chaque colonne du jeu de données
- "dec": définit quel type de séparateur décimal est utilisé dans le jeu de données

Pour vérifier quels paramètres sont à utiliser dans notre cas **ouvrez le fichier avec un éditeur de texte** (bloc-note). Dans notre cas, on observe que des noms de colonnes sont bien présents (donc: header=TRUE), que les colonnes sont séparées par des retours chariots (codés par "\t" en langage informatique, donc: sep="\t") et que les nombres décimaux ont une virgule pour séparateur décimal (donc: dec=","). On peut donc entrer la commande suivante pour importer notre jeu de données dans R et l'enregistrer dans une variable de notre environnement que l'on nommera "data_lezard":

```
data_lezard = read.table("Suivi_lezard_vivipare.csv", header = T, sep = "\t", dec = ",")
# fonctionne également avec:
# data_lezard=read.delim2('Suivi_lezard_vivipare.csv') (read.delim est un
# wrapper de read.table ayant par défaut les arguments qui correspondent à
# notre cas, d'autres wrappers sont disponibles, comme vous pouvez le voir avec
# ?read.table)

# vous pouvez également utiliser la commande '<-' à la place de '=', R n'est
# pas sensible aux espaces, vous pouvez donc séparer ces commandes ou non par
# des espaces si vous le souhaitez
```

Il est important de noter que **ces paramètres peuvent varier suivant les jeux de données**, par exemple les jeux de données anglo-saxons utiliseront plus souvent des virgules comme séparateurs de colonnes et des points comme séparateurs de décimales. Il faudra donc vous adapter à la nature du jeu de données avant de lancer l'ouverture du ficher !

Vous pouvez maintenant voir votre jeu de données apparaître dans le panel supérieur droit, avec son nombre de ligne (obs.) et son nombre de colonne (variables).

POUR ALLER PLUS LOIN

Une source d'erreur très commune lors de l'ouverture des fichiers dans R est l'erreur dans le choix d'encodage. En effet, tous les fichiers ne sont pas encodé de la même manière, avec des disparités notamment suivant les zones géographiques puisque selon les langues de nouveaux caractères sont encodés. Si le fichier s'affiche dans R avec des caractères étranges apparaissant dans les textes, une vérification de l'encodage du fichier semble nécessaire (celui-ci s'affiche en bas à droite de la plupart des éditeurs de texte comme le bloc-note), il faut ensuite indiquer le bon encodage avec l'argument "encoding" dans la fonction d'ouverture de fichier.

Compiler et enregistrer son script R

L'ensemble du code que vous avez écrit jusqu'à maintenant constitue votre **script R**. Ce script peut ensuite être **compilé** dans R pour produire les sorties souhaitées. Dans notre cas, vous devez normalement avoir les deux lignes suivantes écrites dans votre code source:

```
setwd("~/ATER PAU 2024/Cours modifiés/OUMOBIOS")


data_lezard = read.table("Suivi_lezard_vivipare.csv", header = T, sep = "\t", dec = ",")
```

Pour compiler notre script nous avons pour le moment fait des copier-coller depuis la source vers la console, ligne par ligne. Vous pouvez également copier directement un bloc de ligne depuis la source pour compiler plusieurs lignes à la suite dans la console. Il existe également un raccourci plus rapide pour réaliser cela: vous pouvez directement sélectionner les lignes que vous souhaitez exécuter puis cliquer sur la **commande**

“Exécuter” (voir image ci-dessous) pour compiler ces lignes de codes ou bien choisir l’option “Exécuter tout” (voir image ci-dessous) pour exécuter l’ensemble de votre script.

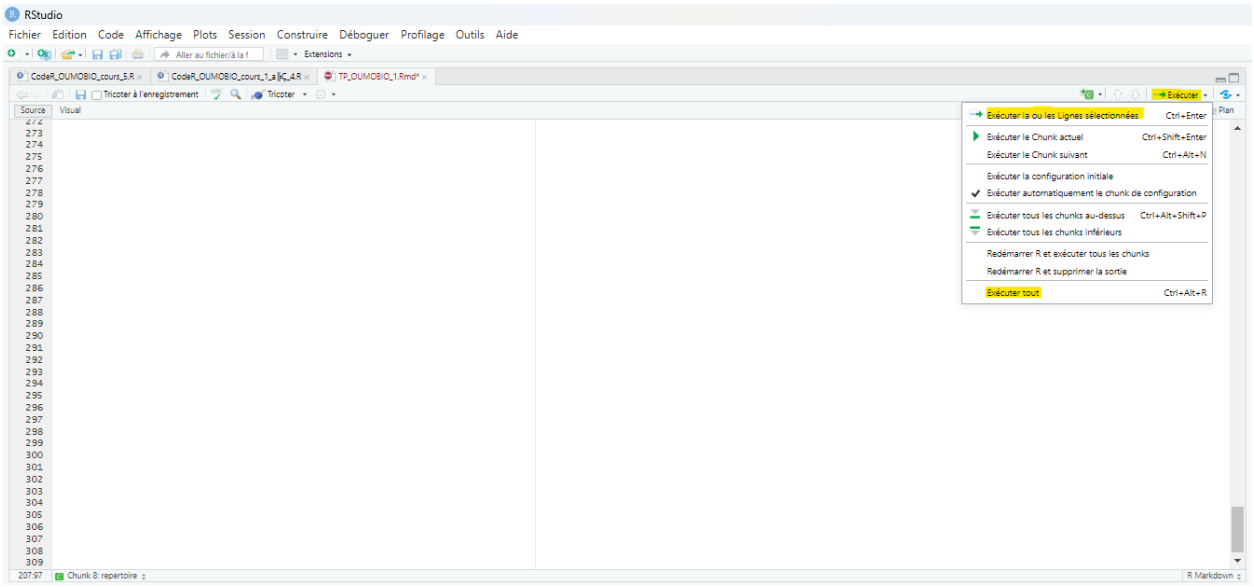

Figure 4: Exécution de lignes de scripts

ASTUCES

N’hésitez pas à utiliser les raccourcis claviers pour compiler votre code (exécuter le code sélectionné: **Ctrl+Entrée**, exécuter tout le script: Ctrl+Alt+R), le gain de temps peut être considérable !

Il faut maintenant **sauvegarder votre script R** dans votre environnement de travail, pour cela allez dans le menu déroulant “Fichier” → “Enregistrer sous” puis sélectionner votre dossier de travail pour y enregistrer votre script.

BONNES PRATIQUES

Pensez à **sauvegarder régulièrement votre travail** en cliquant sur la petite disquette en haut à gauche (voir image ci-dessous). Lorsque votre travail n’est pas sauvegardé le nom de votre fichier R en haut à gauche apparaît en rouge. Il est fortement conseillé d’éviter de rester longtemps sans sauvegarder au cas où R planterait (ce qui peut malheureusement arriver assez souvent !).

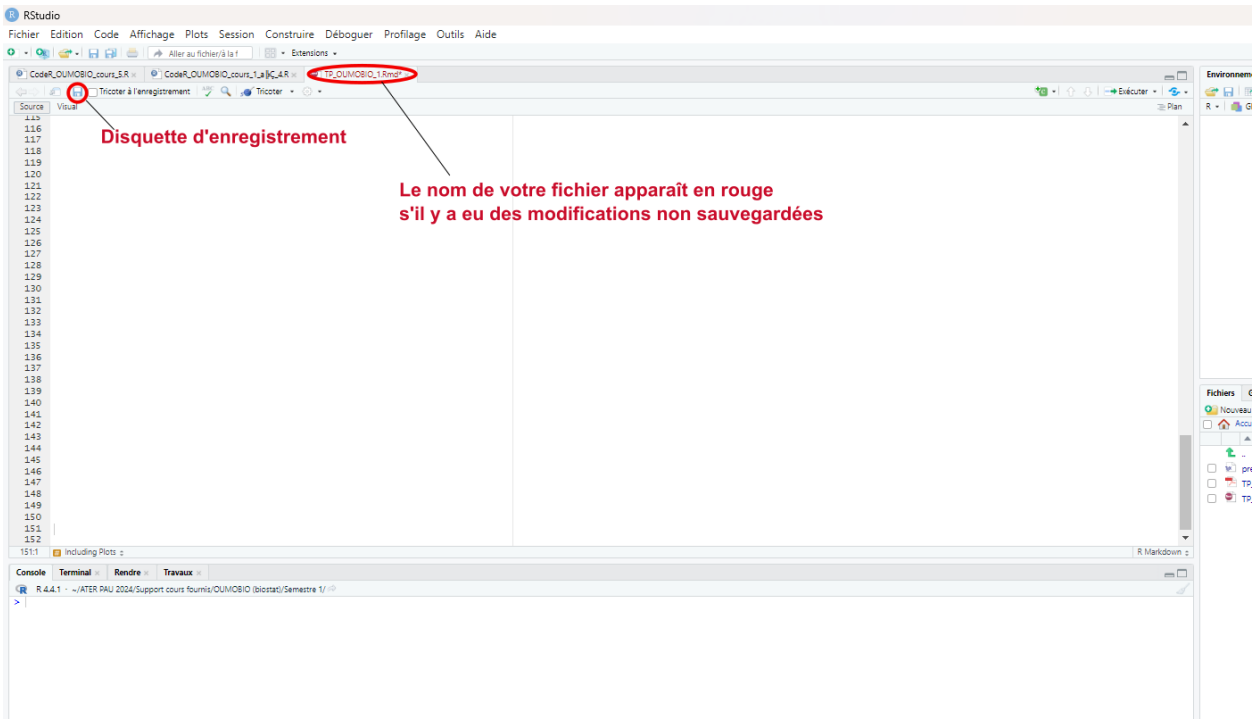

Figure 5: Enregistrement du script

Manipuler un jeu de données

Analyser son contenu

Nous allons maintenant nous familiariser avec notre jeu de données, ouvrez-le dans R en tapant son nom dans la console ou en tapant la ligne suivante:

```
View(data_lezard)
# Vous pouvez également cliquez sur le jeu de données dans le panel supérieur
# droit pour obtenir la même vue
```

Chaque colonne du tableau correspond à une variable, vous pouvez retrouver un descriptif de chaque variable dans le fichier "Index_Table_suivi_lezard.odf" (déposé sur E-learn). **Chaque ligne du tableau correspond à une observation**, c'est-à dire **ici un individu**, qui est décrit par les différentes **variables en colonnes** (taille, masse de l'individu et de sa mère, nourrissage, choix de dispersion au cours de l'expérience, etc.).

Pour avoir une vue synthétique du jeu de données vous pouvez utiliser les commandes suivantes:

```
str(data_lezard)
```

```
## 'data.frame':    168 obs. of  19 variables:
##  $ ID_IND         : chr  "210+0" "219" "227" "271" ...
##  $ BIRTH_DATE     : chr  "02/07/19" "02/07/19" "02/07/19" "02/07/19" ...
##  $ SEX            : chr  "f" "f" "m" "f" ...
##  $ SVL_IND        : int  20 19 21 21 19 19 21 20 20 20 ...
##  $ M_IND          : num  0.16 0.14 0.17 0.18 0.15 0.14 0.18 0.18 0.17 0.16 ...
##  $ ID_MOTHERS     : chr  "203" "205" "201" "214" ...
##  $ CAPT_DATE      : chr  "19/06/19" "19/06/19" "19/06/19" "19/06/19" ...
##  $ POP            : chr  "MON" "MON" "MON" "MON" ...
##  $ SVL_MOTHERS    : int  65 61 61 69 65 61 61 69 65 65 ...
##  $ M_MOTHERS      : num  3.86 3.19 2.62 4.27 3.86 3.19 2.62 4.27 3.86 3.86 ...
##  $ EXP_DATE       : chr  "05/07/19" "05/07/19" "05/07/19" "05/07/19" ...
##  $ EXP_ID         : int  1 1 1 2 2 2 3 3 3 4 ...
##  $ H_BEGIN        : chr  "08:17:00" "08:17:00" "08:17:00" "09:57:00" ...
##  $ H_END          : chr  "09:52:00" "09:52:00" "09:52:00" "10:52:00" ...
##  $ ROLE           : chr  "LI" "F" "RI" "LI" ...
##  $ FED            : logi  FALSE FALSE TRUE TRUE FALSE FALSE ...
##  $ EATEN_CRICKETS: int  NA NA 2 1 NA NA NA NA 2 1 ...
##  $ CHOICE         : chr  "" "None" "" "" ...
##  $ ID_LIGNE       : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
# structure du jeu de données: donne son format, ses dimensions, puis pour
# chaque variable: son format et ses premières valeurs

class(data_lezard)
```

```
## [1] "data.frame"
```

```
# format du jeu de données

dim(data_lezard)
```

```
## [1] 168  19
```

```
# dimensions du jeu de données: nombre de lignes, puis de colonnes

colnames(data_lezard)
```

```
##  [1] "ID_IND"         "BIRTH_DATE"     "SEX"            "SVL_IND"
##  [5] "M_IND"          "ID_MOTHERS"     "CAPT_DATE"      "POP"
##  [9] "SVL_MOTHERS"    "M_MOTHERS"      "EXP_DATE"       "EXP_ID"
## [13] "H_BEGIN"        "H_END"          "ROLE"           "FED"
## [17] "EATEN_CRICKETS" "CHOICE"         "ID_LIGNE"
```

```
# noms des colonnes du jeu de données

head(data_lezard)
```

```
##   ID_IND BIRTH_DATE SEX SVL_IND M_IND ID_MOTHERS CAPT_DATE POP SVL_MOTHERS
## 1  210+0   02/07/19   f      20  0.16        203  19/06/19 MON          65
## 2    219   02/07/19   f      19  0.14        205  19/06/19 MON          61
## 3    227   02/07/19   m      21  0.17        201  19/06/19 MON          61
## 4    271   02/07/19   f      21  0.18        214  19/06/19 MON          69
## 5    214   02/07/19   m      19  0.15        203  19/06/19 MON          65
## 6    224   02/07/19   f      19  0.14        205  19/06/19 MON          61
##   M_MOTHERS EXP_DATE EXP_ID  H_BEGIN    H_END ROLE   FED EATEN_CRICKETS CHOICE
## 1      3.86 05/07/19      1 08:17:00 09:52:00   LI FALSE             NA
## 2      3.19 05/07/19      1 08:17:00 09:52:00    F FALSE             NA   None
## 3      2.62 05/07/19      1 08:17:00 09:52:00   RI  TRUE              2
## 4      4.27 05/07/19      2 09:57:00 10:52:00   LI  TRUE              1
## 5      3.86 05/07/19      2 09:57:00 10:52:00    F FALSE             NA   Left
## 6      3.19 05/07/19      2 09:57:00 10:52:00   RI FALSE             NA
##   ID_LIGNE
## 1        1
## 2        2
## 3        3
## 4        4
## 5        5
## 6        6
```

```
# affiche les premières lignes du jeu de données

tail(data_lezard)
```

```
##     ID_IND BIRTH_DATE SEX SVL_IND M_IND              ID_MOTHERS CAPT_DATE POP
## 163  621+0   23/07/19   m      20  0.14                  545+9  13/06/19 PIM
## 164  631+0   24/07/19   f      19  0.12                    0+6  12/06/19 PIM
## 165  635+0   24/07/19   f      21  0.13                 1295+1  12/06/19 PIM
## 166  624+0   23/07/19   f      21  0.14                  545+9  13/06/19 PIM
## 167  633+0   24/07/19   f      20  0.14                    0+6  12/06/19 PIM
## 168  627+0   24/07/19   m      21  0.18 1000+700+400+200+70+3+1  17/06/19 PIM
##     SVL_MOTHERS M_MOTHERS EXP_DATE EXP_ID  H_BEGIN    H_END ROLE   FED
## 163          61      4.15 27/07/19      4 13:37:00 14:49:00   LI FALSE
## 164          57      3.69 27/07/19      4 13:37:00 14:49:00    F FALSE
```

```
## 165          60      2.72 27/07/19      4 13:37:00 14:49:00   RI  TRUE
## 166          61      4.15 27/07/19      5 14:57:00 16:10:00   LI  TRUE
## 167          57      3.69 27/07/19      5 14:57:00 16:10:00    F FALSE
## 168          53      2.64 27/07/19      5 14:57:00 16:10:00   RI FALSE
##     EATEN_CRICKETS CHOICE ID_LIGNE
## 163             NA             163
## 164             NA   Left      164
## 165              0             165
## 166              1             166
## 167             NA   None      167
## 168             NA             168
```

```
# affiche les dernières lignes du jeu de données
```

La fonction "str" vous permet notamment de visualiser **le format de chaque variable** contenue dans le jeu de donnée. Dans R, vous allez être confronté à 5 types de variable (en dehors des jeux de données): les nombres **entiers** ("integer", ou "int"), les nombres **réels** ("numeric", ou "num"), les **chaînes de caractères** ("character", ou "chr"), les **facteurs** ("factor") et les **booléens** ("logical", ou logi). Globalement toutes les **variables quantitatives** seront soient des nombres entiers, soient des nombres réels; les **variables qualitatives** peuvent être booléennes (c'est à dire répondre à une question "vrai ou faux ?" avec seulement deux valeurs possibles: TRUE ou FALSE) ou des chaînes de caractères (c'est-à dire du texte, **ce dernier type de format est toujours donné entre guillemet " " dans R**). Lorsque la chaîne de caractères prend uniquement certaines **valeurs définies et ordonnées** (on peut leur attribuer des numéros) on parle alors de facteur. Les différentes valeurs d'un facteur sont appelés niveaux et sont donc ordonnées dans un ordre précis. Ces variables peuvent être contenu dans différents ensembles de données, lorsqu'une variable est composé de plusieurs valeurs on parle de **vecteur** (en dimension 1, de la forme "c()" dans R; mais on peut aussi utiliser des matrices de dimension 2: ce cas ne sera pas abordé en TP), on peut aussi associer plusieurs variables ensembles comme c'est le cas dans **les tableaux de données** (format "data.frame") ou dans les listes (format "list", ce format ne sera pas abordé en TP). La fonction **"class"** peut être utilisé sur tout objet R pour déterminer son format. Il est également possible de tester le format d'une variable en utilisant les fonctions de la forme "is.format()", en remplaçant le mot "format" par le format que l'on souhaite tester (par exemple "is.numeric()" permet de tester si une variable est en format numérique ou non).

NOTE IMPORTANTE

Il peut arriver que certaines **valeurs** d'une variable soient **manquantes** dans le jeu de données (pour différentes raisons: une absence de mesure, une impossibilité de mesurer un cas précis, etc.), ces valeurs sont indiqués par le sigle **"NA"** dans R (pour "not applicable, not available, not assessed, or no answer"), quelque soit le format de la variable. C'est le cas par exemple pour la variable "EATEN_CRICKETS" dans notre jeu de donnée: dans le cas présent la mesure n'est pas faite pour certains individus car tous les individus n'ont pas été nourri lors de l'expérience (la mesure du nombre de grillons mangés ne s'applique donc pas à ces derniers). La fonction "na.omit" permet de supprimer tous les NA d'un objet R (dans la cas d'un vecteur toutes les occurrences NA, dans le cas d'un tableau de données toutes les lignes avec au moins un NA apparaissant dedans).

NOTE IMPORTANTE

Il peut parfois être utile de **convertir** une variable d'un format à un autre (par exemple d'une chaîne de caractères à un facteur ou d'un nombre à une chaîne de caractères), dans ce cas on peut utiliser les fonctions de la forme **"as.format()"**, en remplaçant le mot "format" par le format dans lequel on souhaite convertir la variable en entrée (par exemple "as.character()" permet de convertir une variable en format chaîne de caractères). Attention: il faut être prudent avec ce type de manipulation pour éviter des conversions forcés qui n'ont pas lieu d'être.

Sélectionner des sous-ensembles

Comme nous l'avons vu précédemment le jeu de données est composés de lignes (les individus) et de colonnes (les variables). Pour pouvoir analyser certains éléments en particulier (par exemple les caractéristiques d'un individu en particulier ou toutes les valeurs d'une unique variable) nous allons devoir apprendre à **extraire des sous-ensembles du jeu de données**. La sélection de sous-ensemble dans R se réalise grâce aux crochets **"[ ]"**, pour un jeu de données on positionne une virgule entre les crochets pour distinguer la **sélection des lignes (avant la virgule)** de celle des **colonnes (après la virgule)**. La sélection peut s'opérer soit à l'aide des **numéros** de colonnes ou de lignes soit à l'aide des **noms** de colonnes ou de lignes (quand il y en a). Voici les commandes utiles à la sélection de lignes ou colonnes:

```
# sélection à l'aide des identifiants lignes/colonnes:

data_lezard[3, ]
# sélection de la troisième ligne du jeu de données

data_lezard[, 5]
# sélection de la cinquième colonne du jeu de données

data_lezard[, -4]
# sélection de toutes les colonnes sauf la quatrième



data_lezard[c(3, 6, 7), ]
# sélection des lignes N°3, 6 et 7 du jeu de données

data_lezard[c(6, 3, 7), ]
# l'ordre de sélection est important ! Ici on a inversé la position des lignes
# 6 et 3

data_lezard[-c(3, 6, 7), ]
# sélection de toutes les lignes saufs les lignes N°3, 6 et 7 du jeu de données

data_lezard[, c(2, 4, 6)]
# sélection des colonnes N°2, 4 et 6 du jeu de données

data_lezard[, c(4, 2, 6)]
# l'ordre de sélection est important ! Ici on a inversé la position des
# colonnes 4 et 2
```

```r
data_lezard[2, c(2, 4, 6)]
# sélection des deuxièmes valeurs des colonnes N°2, 4 et 6 du jeu de données




# sélection à l'aide des noms de colonnes:

data_lezard[, "BIRTH_DATE"]
# sélection des dates de naissances des lézards


data_lezard[c(2, 3), "BIRTH_DATE"]
# date de naissance des deuxième et troisième individus


data_lezard[, c("SVL_IND", "M_IND")]
# sélection de la taille et de la masse des individus

data_lezard[, c("M_IND", "SVL_IND")]
# l'ordre de sélection est important ! Ici on a inversé la position des
# variables masses et tailles


data_lezard$BIRTH_DATE
# sélection des dates de naissance des lézards
```

On peut également sélectionner des sous-ensembles **sous certaines conditions** (par exemple sélectionner tous les individus, c-à-d lignes, provenant de la population "MON"). Cela nécessite de maîtriser les **opérateurs logiques** de R dont voici les principaux:

- >/<: supérieur/ inférieur à
- == : égal à
- >=/<= : supérieur ou égal / inférieur ou égal à
- %in% : est contenu dans
- is.na() : est une valeur manquante
- ! : "non" logique
- & : "et" logique
- | : "ou" logique

L'idée va être ensuite d'indiquer la **condition entre crochets**, encore une fois avant la virgule si elle porte sur les lignes, ou après si elle porte sur la colonne. Pour notre exemple, la condition porte sur la population d'origine des individus, on va donc indiquer que l'on souhaite faire une sélection en fonction de la variable "POP" (code correspondant: data_lezard$POP) et que l'on souhaite sélectionner les occurrences égales à "MON" (code correspondant: == "MON") on indiquera donc: data_lezard[data_lezard$POP == "MON",], en positionnant la condition entre crochet et avant la virgule pour spécifier qu'elle porte sur les lignes du jeu de données.

Voici quelques autres exemples de sélections conditionnelles:

```r
data_lezard[data_lezard$SVL_IND > 20, "M_IND"]
# poids des individus dont la taille est supérieure à 20 mm
```

```r
data_lezard[data_lezard$SVL_IND <= 18 & data_lezard$M_IND <= 0.15, ]
# sélection de tous les individus de taille inférieure ou égale à 18 mm et de
# poids inférieur ou égal à 0.15 g

data_lezard[data_lezard$SVL_IND == 18 | data_lezard$SEX == "m", ]
# sélection de tous les individus de taille égale à 18 mm ou de sexe masculin

data_lezard[!data_lezard$POP == "MON", ]
data_lezard[data_lezard$POP != "MON", ]
# sélection de tous les individus provenant d'une autre population que 'MON'

data_lezard[data_lezard$POP %in% c("JOC", "JON"), ]
# sélection de tous les individus provenant des populations JOC et JON

data_lezard[data_lezard$FED == FALSE, ]
data_lezard[is.na(data_lezard$EATEN_CRICKETS), ]
# sélection de tous les individus n'ayant pas été nourris

data_lezard[, colnames(data_lezard) %in% c("SVL_IND", "M_IND")]
# sélection des variables (=colonnes) taille et poids des individus

data_lezard[, !colnames(data_lezard) %in% c("SVL_IND", "M_IND")]
# sélection de toutes les variables sauf la taille et le poids des individus
```

EXERCICE

- Sélectionnez les individus nés le 20 et 21 juillet 2019
- Sélectionnez les individus dont les mères pesaient strictement moins de 4 g et mesuraient une taille égale ou supérieure à 60 mm
- Sélectionnez le poids et la taille des mères pour les individus de sexe féminin
- Sélectionnez la population d'origine des individus dont la taille est différente de 20 mm
- Expliquez ce que fait la ligne suivante: "data_lezard[!is.na(data_lezard$EATEN_CRICKETS),]"

```r
data_lezard[data_lezard$BIRTH_DATE == "20/07/19" | data_lezard$BIRTH_DATE == "21/07/19",
    ]
data_lezard[data_lezard$BIRTH_DATE %in% c("20/07/19", "21/07/19"), ]

data_lezard[data_lezard$M_MOTHERS < 4 & data_lezard$SVL_MOTHERS >= 60, ]

data_lezard[data_lezard$SEX == "f", c("SVL_MOTHERS", "M_MOTHERS")]

data_lezard[data_lezard$SVL_IND != 20, "POP"]

# cette ligne de code sélectionne l'ensemble des individus pour lesquels la
# donnée du nombre de criquets mangés est présente (c'est à dire les individus
# qui ont été nourri)
```

ASTUCES

Vous pouvez facilement retrouvez les anciennes lignes de code que vous avez entrées dans la console en utilisant les flèches haut-bas ou le raccourci Ctrl+R lorsque vous êtes dans la console. S'affiche alors l'historique des lignes de codes que vous avez entrées, avec Ctrl+R vous pouvez directement rechercher celles qui vous intéressent en utilisant les premières lettres de la commande pour lancer une recherche. Il ne vous reste plus qu'à sélectionner la ligne que vous souhaitez récupérer.

MISE EN APPLICATION

Dans l'espace E-learn vous trouverez un autre jeu de données appelé "Interactions_dauphins_bateaux.txt". Cette table de données décrit le comportement de dauphins à proximité de bateaux (colonne boat.dist: "no"= pas de réponse, "approach"= s'approche du bateau, "avoidance"= s'éloigne du bateau, "response"= interagit avec le bateau) et peut être utile à des gestionnaires pour comprendre le potentiel dérangement généré par ces interactions. L'objectif est pour vous d'importer ce jeu de données dans R et d'utiliser les outils qui vous ont été présentés au cours de cette séance pour explorer ces données et vous les approprier. Voici quelques exemples ci-dessous d'objectifs que vous pouvez chercher à réaliser lors de votre exploration.

- Quel est la taille de ce jeu de données ? Quelles sont les différentes variables de ce jeu de données ? Quelles sont leur classe ? Leurs valeurs ? Y a-t-il des variables contenant des valeurs manquantes ?
- Quels sont les réponses et comportements observés en fonction de l'âge, du sexe ?
- Quels sont les comportements et réponses observés sur des groupes de petite taille (<5) et de grande taille (>4) ?

Bilan

Nous avons vu au cours de cette séance de TP comment **créer son environnement de travail et un script sous R** et comment **ouvrir et manipuler un jeu de données**. Il est important de garder en tête les bonnes pratiques abordées au cours de cette séance: savoir **utiliser la documentation** présente dans R (help("fonction") ou ?fonction), utiliser des **dénominations pertinentes** pour les fichiers et dossiers (**sans caractères spéciaux**), et **enregistrer régulièrement** son script R.

Les principales commandes à retenir sont les suivantes:

- **setwd("Chemin_d_acces/Dossier_travail")**, permet de définir l'espace (ou dossier) de travail sur R pour importer des fichiers depuis celui-ci ou enregistrer des objets R dans celui-ci; prends en argument le chemin d'accès du dossier
- **read.table("Data.csv")**, permet d'ouvrir un jeu (=tableau) de données dans R (ici "Data.csv"); prends en argument les options "header" (présence ou nom de noms de colonnes dans le jeu de données), "sep" (type de séparateur entre colonnes) et "dec" (séparateur décimal utilisé dans le de données)
- **str()**: donne la structure d'un jeu de données
- **class()**: permet de déterminer la classe d'une variable; soit un "character" (catégorielle: un nom, un mot, une phrase...), un "factor" (ordinale: un classement, une note...) ou un "logical" (booléenne: TRUE/T ou FALSE/F) pour une valeur unique ou un vecteur qualitatif, soit un "numeric" (réels) ou un "integer" (entiers) pour une valeur unique ou un vecteur quantitatif, soit "data.frame" pour un jeu de données
- **as. "Nom de classe"()**: permet de convertir une variable d'une classe à une autre (la classe d'arrivée étant définie par "Nom de classe"), par exemple as.character() permet de convertir une variable numérique ou logique en une variable chaîne de caractère
- **Data[,]**: les crochets permettent de réaliser une sélection dans le jeu de données "Data", soit en utilisant les **numéros** de ligne et de colonne, soit en utilisant les **noms** de variables (=colonnes) ou de lignes (=individus ou observations); la sélection peut se faire via des **conditions** (=tests) logiques en utilisant les différents **opérateurs logiques** existants (à connaître !)