Formation R & RStudio

Prise en main et acquisition des bases

Formation délivrée le : 12/12/2024

Mise à jour du support le : 26/11/2024

Blondel Eriann
LAPPS

Table des matières

1. Généralité sur l'environnement R & RStudio

A) Distinction entre R et RStudio

1. R

R est le logiciel contenant toutes les instructions permettant l'exécution des lignes de codes. Il ne contient qu'une console, où sont exécutée les lignes de code) ainsi qu'un éditeur de script très rudimentaire.

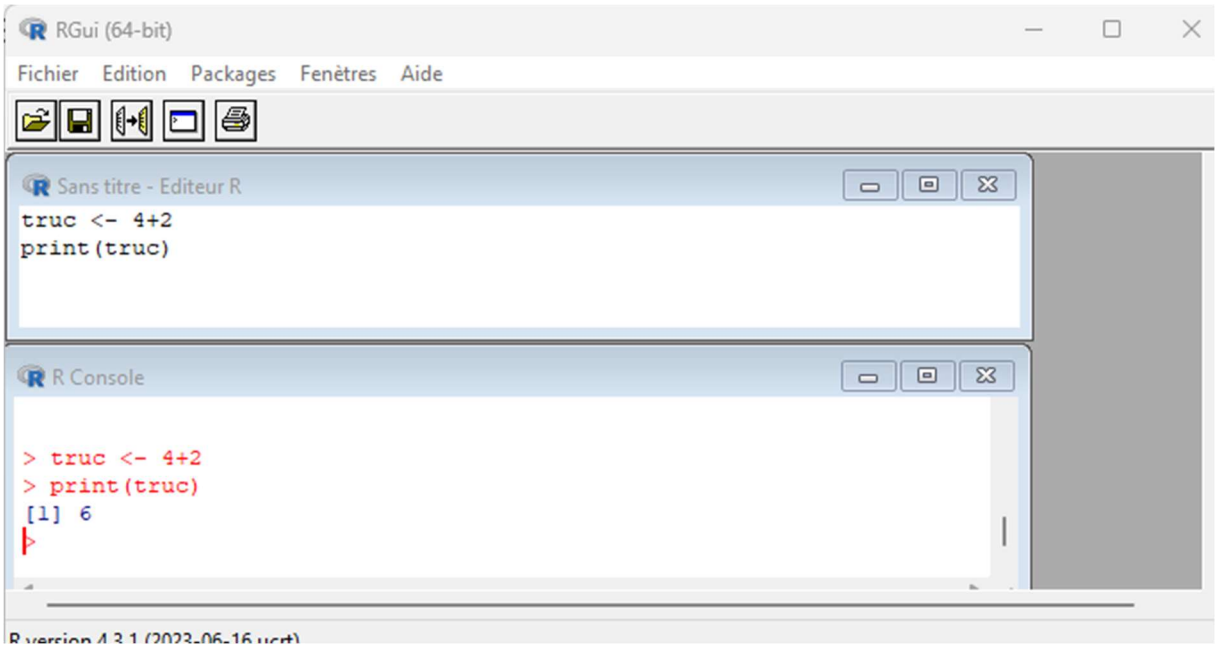

Figure 1: Interface

2. RStudio

RStudio est une interface plus ergonomique permettant de communiquer avec R. Il permet une visualisation des objets que nous créons, l'éditeur de script est plus ergonomique, il existe même un éditeur spécial qui permet la génération automatique de rapport au format pdf ou d'un logiciel de traitement de texte.

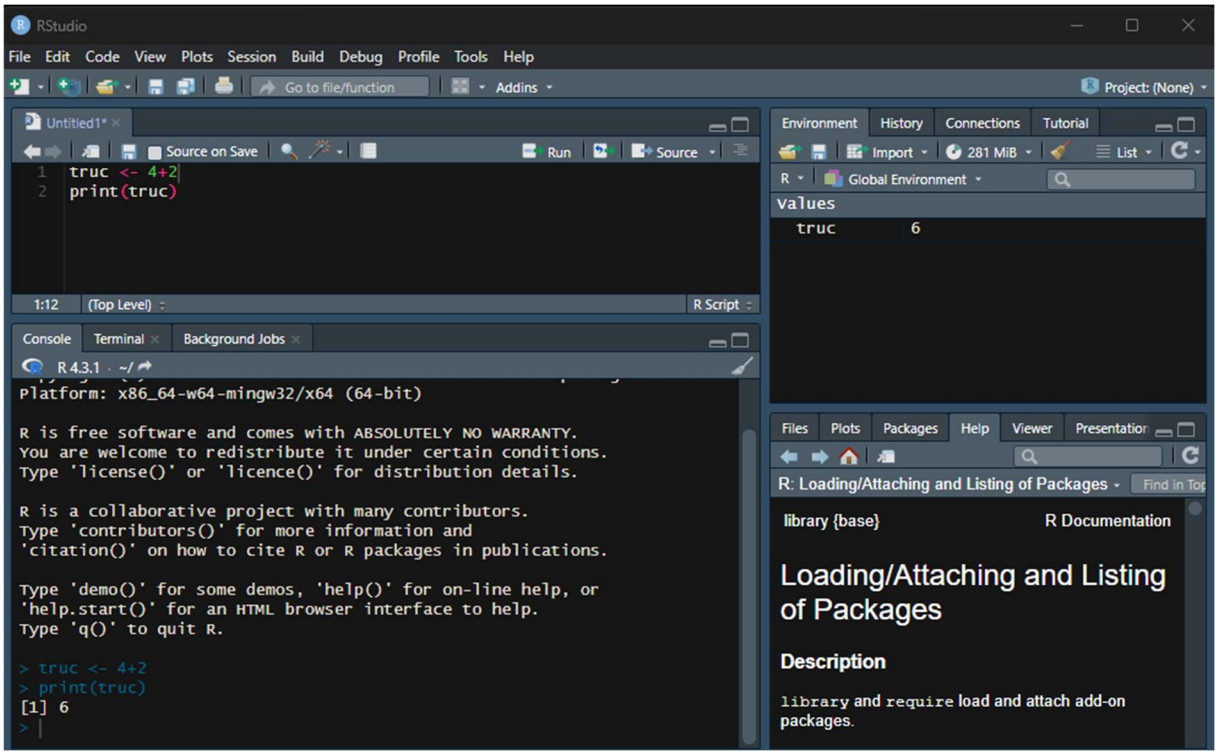

Figure 2 : Interface RStucio (thème sombre)

B) Présentation de l'interface RStudio

Sur l'image précédente, nous avons vu que l'interface était découpée en quatre sections. Il est possible de définir la taille de chaque section par rapport aux autres en utilisant une fonction « cliquer-glisser » sur la zone de jonction entre deux sections.

[Exercice 1]

1. La console

La console est la section principale. Elle se situe en bas à gauche de l'écran C'est ici que s'exécutera le code R. Il est possible d'écrire directement une ligne de code dans la console. Cependant, à partir du moment où l'on actionne la touche entrée, R comprendra qu'il faut exécuter tout ce qui vient d'être tapé. Nous ne pouvons donc pas y écrire un script entier.

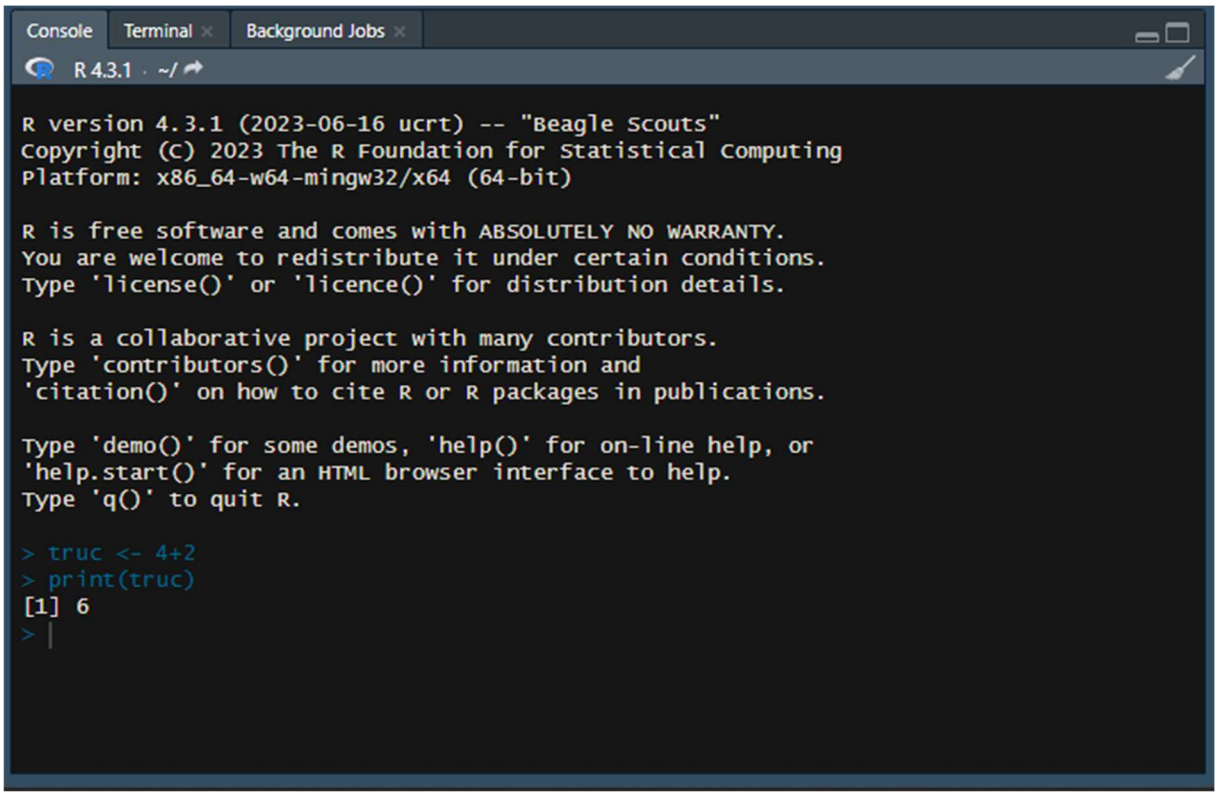

Figure 3 : Console RStudiio (Thème sombre)

[Exercice 2]

2. L'espace de travail

Il s'agit de la zone dans laquelle nous allons pouvoir créer nos scripts. Nous pouvons ouvrir simultanément plusieurs scripts, lesquels bénéficieront d'aides visuels pour identifier la nature des différentes expressions.

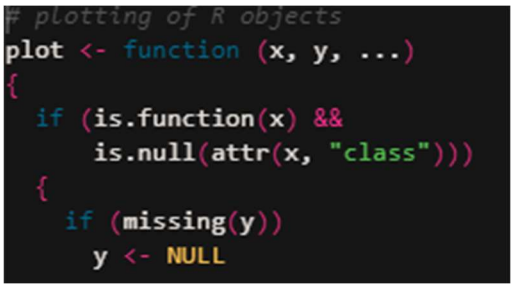

Figure 4 : exemple de script (RStudio - Thème sombre)

Il est également possible d'ouvrir un ou plusieurs fichier Markdown ou Quarto[1].

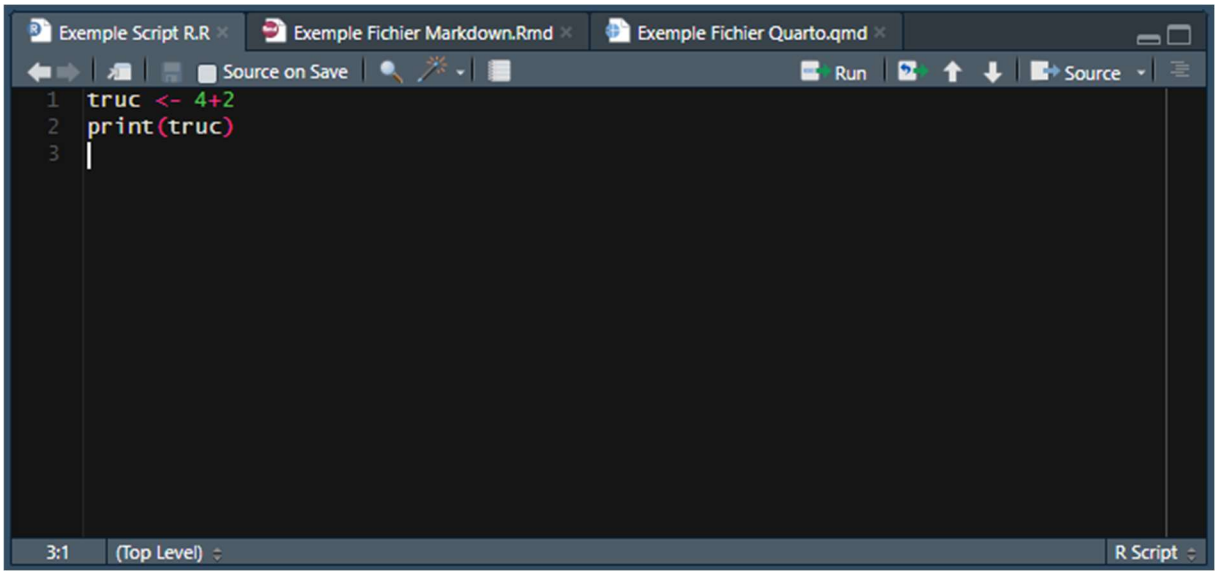

Figure 5 : espace de travail (RStudio - Thème sombre)

Par défaut, les fichiers ne sont sauvegardés. Il faut penser à les enregistrer. Il suffit de se cliquer sur le script / fichier Markdown / fichier Quarto qui nous intéresse et d'aller dans la rubrique « file » et cliquer sur « Save as ».

[Exercice 3]

Remarque : il existe des éditeurs de codes tiers qui permettent de créer des scripts en-dehors de RStudio, si vous préférez des espaces programmation moins étriqués.

[1] Ces formats de fichier seront présentés plus tard.

3. L'environnement de travail

Nous pouvons avoir un œil sur tous les objets créés dans RStudio grâce à l'environnement de travail. Il se situe en haut à droite.

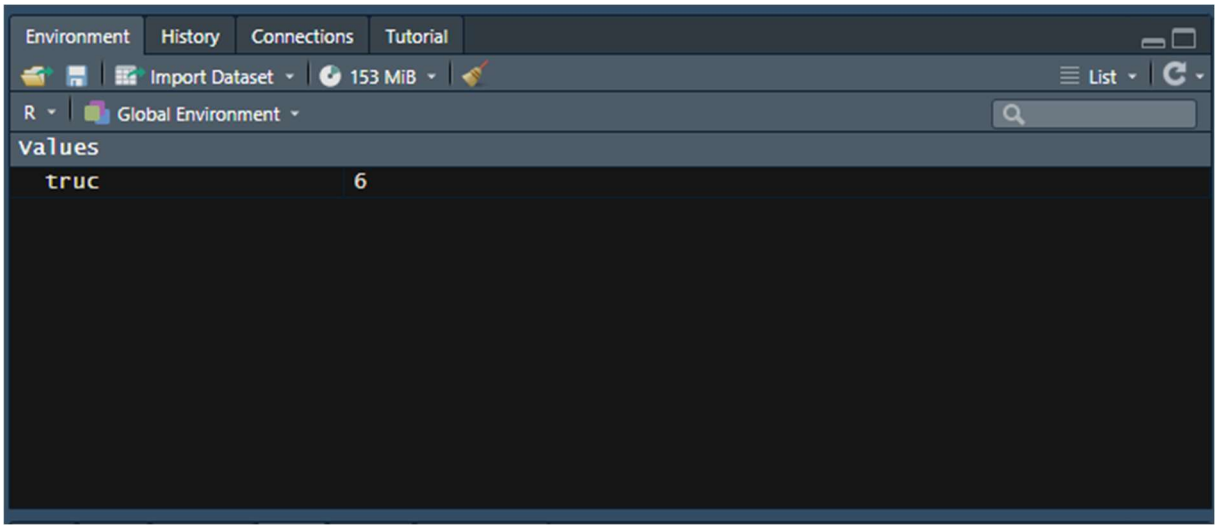

Figure 6 : Environnement de travail (RStudio - Thème sombre)

Remarque : pensez à vérifier dans les options que R ne sauvegarde pas tout l'historique.

[Exercice4]

4. Ressources & visuels

En bas à droit de l'écran, se situe une section avec différents onglets. Cette section permet de consulter différents éléments. Voici les plus importants

- Explorateur de fichier :

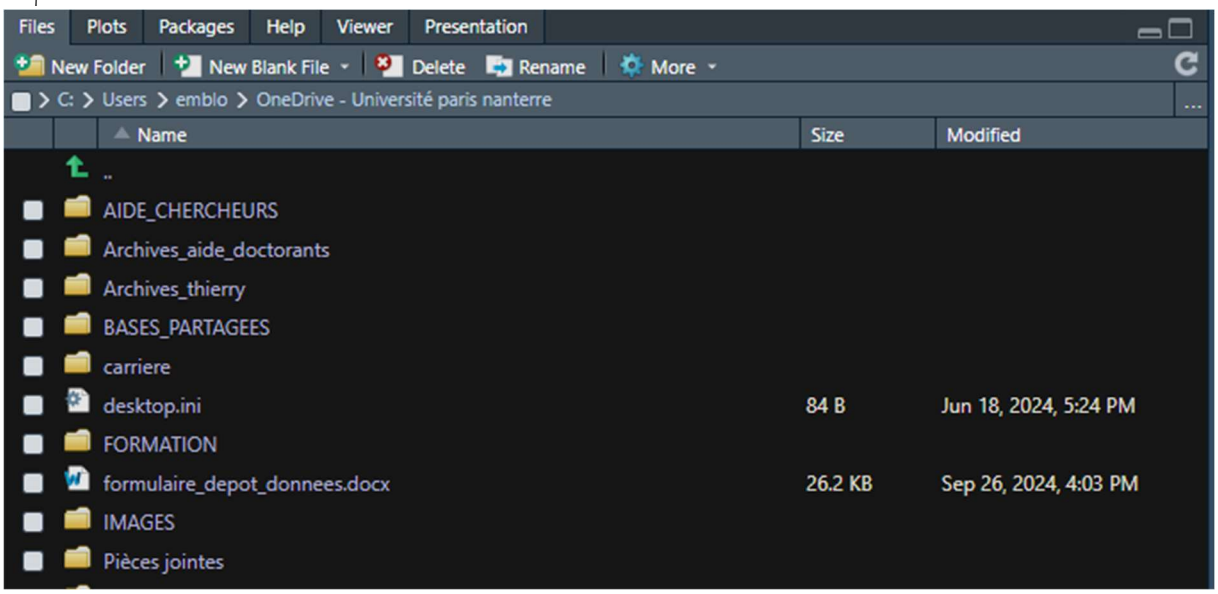

Figure 7 : explorateur de fichier (RStudio - Thème sombre)

- L'aide :

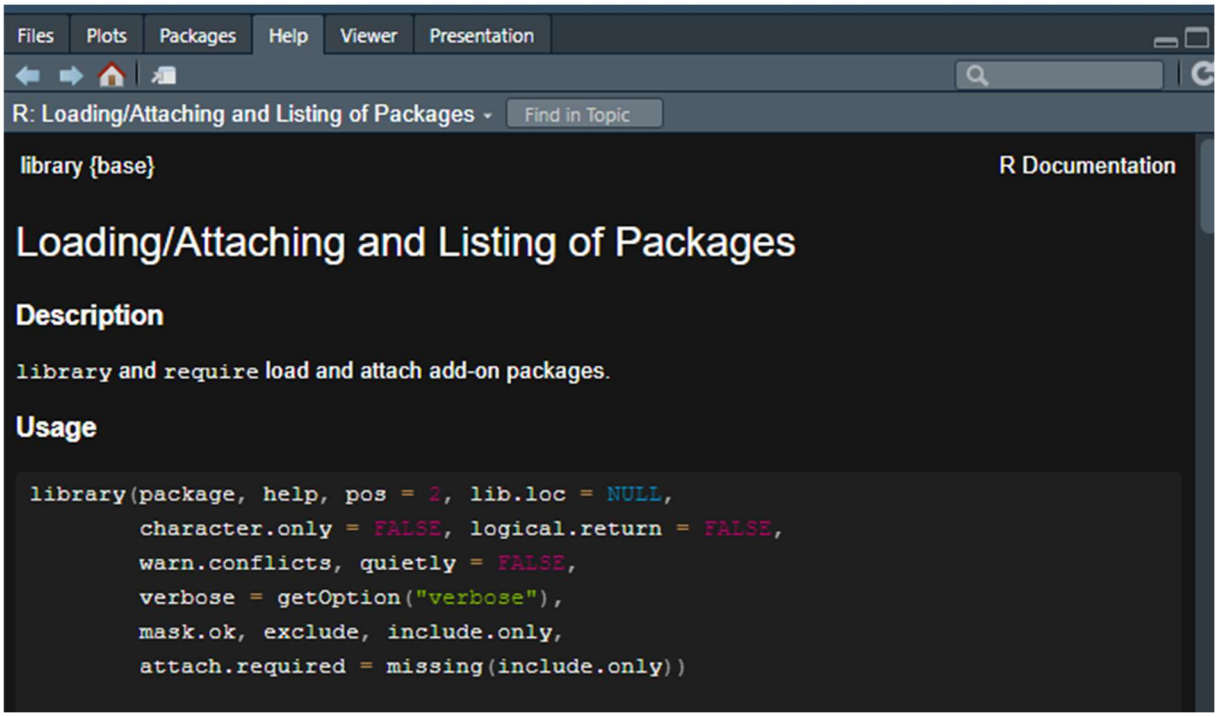

Figure 8 : Aide (RStudio - Thème Sombre)

- « Graphiques » (permet de voir les graphiques produits)

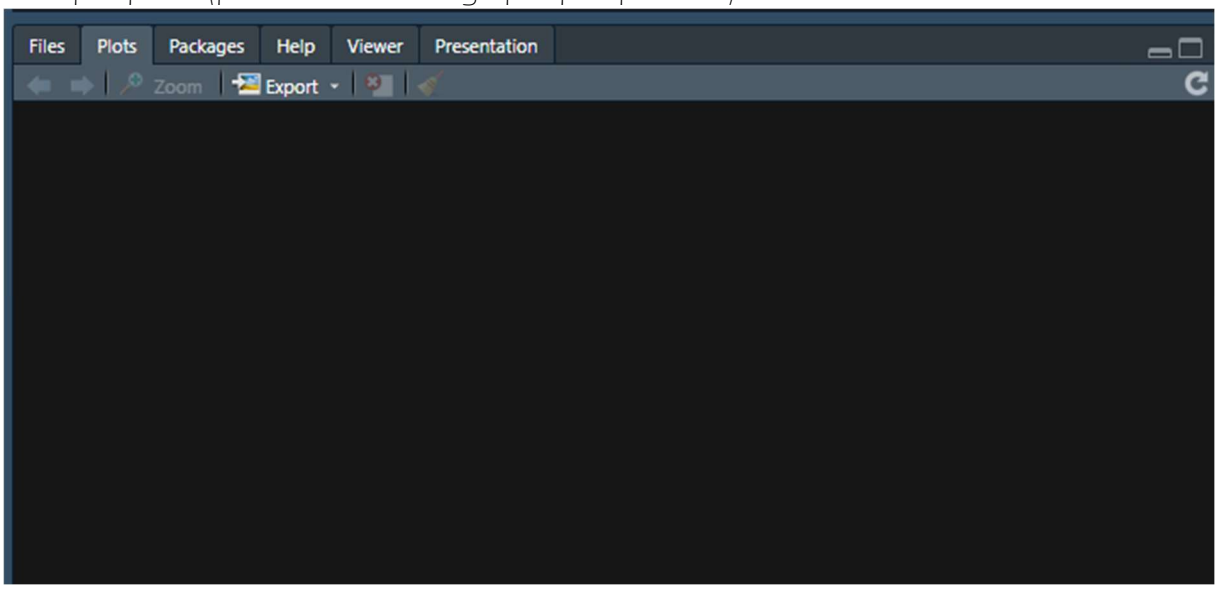

Figure 9 : "Graphiques" (RStudio - Thème sombre)

2. Les règles de saisie

A) La création d'objets

Les objets seront toutes les entités que vous créerez et stockerez dans l'environnement de travail RStudio (base de données, résultats de traitement, variables etc.).

1. Sensibilité aux majuscules

R/RStudio fait une distinction entre une lettre écrite en minuscule et écrite en majuscule. Il est donc important de faire attention aux majuscules, si vous décidez d'en ajouter

Par exemple, sms et SMS seront considérés comme deux objets différents.

2. Caractères autorisés

Les lettres (minuscules et majuscules), les nombres, underscore (_) et les points (.) sont autorisés.

Remarques :

- Pour le point, si votre base à vocation à être utilisée sur d'autres logiciels de traitement, je recommande d'éviter. En effet, ce dernier ne sera pas forcément accepté par le logiciel choisi. Par ailleurs, si vous utilisez certaines fonctionnalité avancées (tel que des packages SQL), il risque d'y avoir des conflits.
- Lors de la création de vos questionnaires, pensez à paramétrer le codage de vos variables pour que celles-ci soient bien retranscrites sur R. Ce dernier remplacera automatiquement les caractères qu'il ne reconnaît pas (notamment par des points).

3. Différentiation entre objets/bases de données et variables

Une base de données (DB) est un type d'objet. Il s'agit donc d'une suite de caractères (correspondant aux contraintes précédemment citées). Une variable est un morceau particulier d'une DB, donc d'un objet. Pour faire référence à une variable en particulier, il faut écrire nom_DB$nom_variable. R, comprendra qu'on parle de la variable nom_variable appartenant à l'objet nom_DB.

B) Autres cas

1. Chaînes de caractères

Lorsque nous faisons tapons une chaîne de caractères qui n'est pas un objet, il faut utiliser des guillemets anglais ou des apostrophes. En effet, si nous ne mettons pas de guillemets ou des apostrophes, R croira que nous faisons référence à un objet ou à une expression clé liée à une fonction.

Par exemple : "ma chaîne de caractères" ou 'ma chaîne de caractères'

[Exercice 5]

2. Les commentaires

Il est possible d'indiquer qu'une ligne est du code non exécutable. Cela est très utile pour ajouter des commentaires, permettant une meilleure compréhension du fonctionnement du script. Pour ce faire, il suffit d'insérer en début de ligne le caractère #.

Par exemple :

```
#Cette ligne est un commentaire.
```

[Excercice 6]

3. Rédaction d'une fonction

Un programme R n'est qu'un enchaînement de différentes fonctions qui seront appliquées les unes à la suite des autres.

D'un point de vue formel, une fonction se rédige de la manière suivante :

```
maFonction(instruction1, instruction2, [...])
```

Le nombre et la syntaxe des instructions disponibles différent selon la fonction. La section aide vous aidera dans vos essais.

[Exercice 7]

Il est possible de stocker le résultat d'une fonction dans l'espace de travail grâce à la création d'objet.

```
monObjet <- maFonction(instruction1, instruction2, [...])
```

[Exercice 8]

4. Configurer son espace RStudio

A) Choisir un thème couleur

Chacun ayant une sensibilité différente (thème claire VS thème sombre), il est possible de personnaliser son interface.

Pour ce faire, rendez-vous dans « Global Options », via l'onglet « Tools ». Un menu s'affichera. Allez à la rubrique « Appearance ». Vous pouvez désormais choisir un thème et l'appliquer.

B) Définir son répertoire de travail

1. La base

Lorsque nous exportons ou importons un document sur R, il faut qu'il sache dans quel répertoire le chercher.

Il existe une commande pour prédéfinir dans quel dossier précis R devra travailler. Il s'agit de la commande **setwd("chemin de mon dossier")** .

Si je me prends en exemple :

```
setwd("C:/Users/OneDrive - Université paris nanterre/AIDE_CHERCHEURS")
```

Si vous avez l'habitude de faire des copiés-collés de chemin de fichier, vous remarquerez que R exige qu'on remplace tous les antislash (\) par des slash (/). Normalement, mon chemin de fichier Windows est : C:\Users\OneDrive - Université paris nanterre\AIDE_CHERCHEURS.

Il faudra donc penser à remplacer tous les \ par des /, quand vous ferez un copié-collé du chemin du dossier.

2. Optimisation

Il est possible que le dossier à partir duquel vous allez importer vos bases de données soit différent de celui où vous exporterez vos résultats.

Je vous propose d'utiliser la création d'objets afin d'optimiser la déclaration du dossier de travail. Nous créerons deux objets, un pour les imports, l'autre pour les exports (vous pouvez en créer plus si vous avez des exports de différentes natures, par exemple).

- Importations :

```
import_way <- "C:/Users/OneDrive - Université paris nanterre/AIDE_CHERCHEURS/import"

setwd(import_way)
```

- Exportations :

```
export_way <- "C:/Users/OneDrive - Université paris nanterre/AIDE_CHERCHEURS/export"

setwd(export_way)
```

Remarques importantes :

Si vous déclarez différents dossiers de travail, R ne prendra en compte que le dernier dossier déclaré. Faîte donc attention à l'ordre dans lequel vous déclarez les choses.

Même pour les exports, il faut que le dossier déclaré soit pré-existant. R ne créera pas de dossier si celui déclaré n'existe pas.

[Exercice 9]

C) Les packages

R étant un logiciel libre, il bénéficie du travail partagé de milliers d'utilisateurs, qui ont créé des milliers de fonctions pour enrichir le logiciel. Ces fonctions se retrouvent sous forme de packages à installer.

Un package ne s'installe qu'une seule fois (sauf mises-à-jour et erreurs de compatibilité). Cependant, à chaque fois que vous lancerez RStudio, il faudra charger les packages que vous souhaitez utiliser.

1. Installation

Pour installer un package, il suffit de cliquer sur « Install Packages… » dans l'onglet « Tools ». Un menu s'affiche alors et il suffit de chercher le nom du package qui nous intéresse et de cliquer sur « Install ».

[Exercice 10]

2. Chargement

Pour charger un package, il suffit de taper la ligne de code suivante :

```
library(nom_du_package)
```

[Exercice 11]

Remarque importante :

Je vous conseille d'appeler tous les packages nécessaires au début de votre script (en précisant en commentaire leur utilité dans le script).

3. Trouver de la documentation

Si vous avez un doute sur la syntaxe d'une fonction, vous pouvez solliciter de la documentation grâce à la ligne de code suivante :

```
help(nom_de_ma_fonction)
```

[Exercice 12]

Si vous souhaitez obtenir une documentation complète sur un package, il suffit de taper le nom du package sur un moteur de recherche, vous retrouverez normalement toute la documentation en ligne.

5. Importation de la base de données et vérifications

A) Importation

Nativement, R ne permet l'importation de base que sous certains formats peu usités dans nos pratiques (par exemple : le CSV).

J'ai l'habitude d'utiliser des bases, soit au format Excel qui est communément admis. Soit au format SPSS, qui permet de conserver la richesse d'information, notamment les étiquettes nominales pour les échelles numériques.

Pour pallier au manque de R, nous allons utiliser le package openxlsx (précédemment installé), qui permet d'importer des base de données au format Excel grâce à la fonction read.xlsx()

Afin de pouvoir travailler sur la base, lors son importation, il faudra créer un objet qui la stockera.

[Exercice 13]

B) Vérification visuelle

Dans la fenêtre d'environnement de travail, je peux constater l'existence de l'objet créé, ayant les caractéristiques d'une base de données (« obs » et « variables »).

Figure 10 : environnement de travail contenant une base de travail

Si on clique dessus, elle s'ouvrira dans la fenêtre dédiée aux scripts. Cela nous permettra de vérifier le contenu.

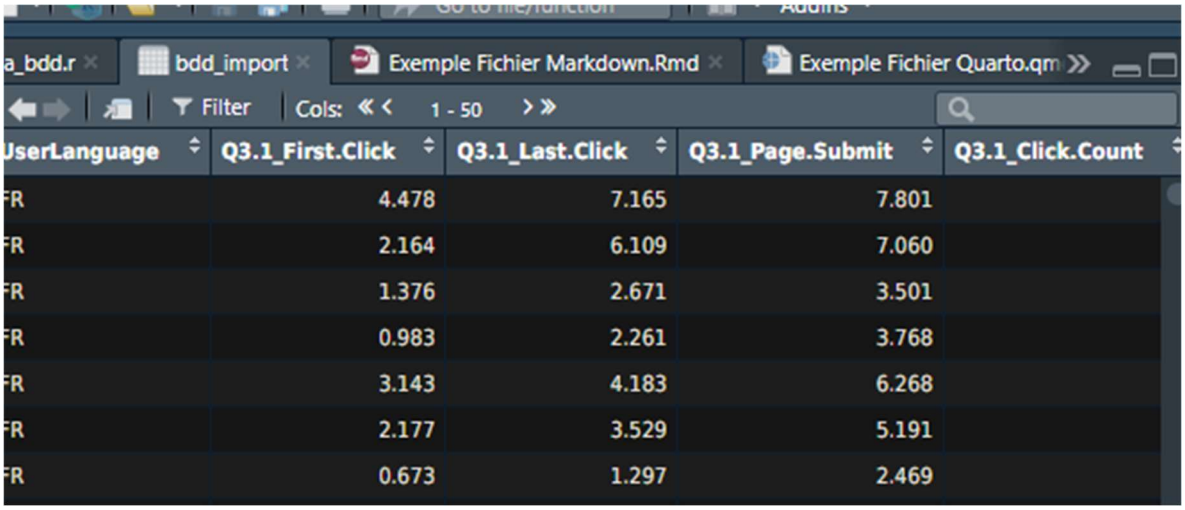

Figure 11 : aperç de la base de données

C) Vérification complète

Il existe des fonctions permettant de vérifier l'intégralité du contenu d'une base afin de vérifier la nature et les modalités des variables. Il s'agit d'une étape indispensable lorsque le nombre d'individus et/ou de variables est important. C'est le cas de la fonction summary().

```
summary(ma_bdd)
```

[Exercice 14]

6. *Transformer sa base de données*

A) Nettoyage : ne prendre que les lignes qui nous intéressent

1. Sélectionner des individus précis

Il va s'agir ici de ne sélectionner que les lignes/individus qui correspondent à nos critères d'étude.

Pour ce faire, la fonction subset nous permettra de sélectionner les individus en fonctions de critères qui nous établiront.

Nous pouvons enregistrer la nouvelle base dans l'objet contenant la base sur laquelle nous allons appliquer un filtre. Cependant, cela écrasera cette dernière. Il vaut mieux créer un nouvel objet (au moins le temps des tests).

```
ma_nouvelle_base <- subset(ma_base_importee, critere)
```

2. Sélection des variables précises

Il est possible d'utiliser la fonction subset() pour opérer une sélection sur les variables.

```
ma_nouvelle_base <- subset(ma_base_importee, select = c(variable1, variable2, [...]))
```

Si vous n'avez besoin d'exclure que quelques variables sur plusieurs dizaines, il sera plus rapide de dire à R de sélectionner toutes les variables sauf une liste restreinte.

```
ma_nouvelle_base <- subset(ma_base_importee, select = -(c(variable1, variable2, [...])))
```

Remarque, si on connaît le numéro de l'ordre d'apparition d'une variable, on peut mentionner son numéro plutôt que son code. Si les variables se suivent, il est même possible de les référencer de la manière suivante

```
(1:10)
```

Ci-dessus R comprendra que je fais référence au variable numérotées 1 à 10.

[Exercices 15 et 16]

B) Créer une variable

La création de variable se fait dans le même esprit que la création d'un objet contenant une base de données.

```
ma_base$ma_viarable <- ma_fonction(instruction1,[...])
```

Il existe une quantité importante de fonctions permettant de générer de nouvelles variables à partir d'un jeu de données existant. L'étape la plus important est d'arriver à traduire le résultat que l'on souhaite en une procédure informatique.

[Exercices 15 et 16]

C) Supprimer des objets, bases ou variables

Si vous êtes sûr de ne plus avoir besoin d'un objet, une base ou une variable, vous pouvez directement les supprimer avec la commande suivante.

```
rm(objet_a_supprimer1, objet_a_supprimer2, [...])
```

[Exercice 17]

D) Attribuer des libellés aux variables quantitatives

Imaginons que vous ne disposiez pas d'une base de données au format SPSS/Jamovi, mais au format Excel. Vos différentes échelles sont codées de manière quantitative. Mais pour certaines traitements (des tableaux ou des graphiques) d'étiquettes de valeur. C'est possible de le faire sur R.

Nous allons utiliser la fonction labelled du package éponyme.

```
ma_bdd$ma_variable <- labelled(ma_bdd$ma_variable, c("étiquette 1"=1,
                                                     "étiquette 2" = 2,
                                                     [...]
                                                     )
                               )
```

La fonction val_labels() permet de vérifier les étiquettes créées.

[Exercice 18]

E) Renommer une variable

La fonction rename.variable() du package questionr permet de renommer une variable.

```
ma_bdd <- rename.variable(ma_bdd, "ancien_nom_variable", "nouveau_nom_variable")
```

[Exercice 19

F) Attribuer une étiquette à une variable

Grâce au package labelled, il est également possible d'attribuer une étiquette à une variable. Cela permet d'apporter une description à la variable mais peut être également affiché à la place du nom/code de la variable dans les résultats de traitement.

```
var_label(ma_bdd$ma_variable) <- "Mon étiquette 1"
```

[Exercice 20]

7. Décrire la base de données

A) Obtenir le nombre d'individus

La première information à générer lorsque nous souhaitons décrire une base de données est le nombre d'individus.

Pour ce faire, nous allons utiliser la fonction nrow. Cette dernière va compter le nombre de lignes contenues dans la base de données. Attention, comme cela fonctionne sur le nombre de lignes, cette technique ne fonction que si la structure de votre base correspond au format suivant une ligne = un individu. Si vous disposez d'une base issue d'une étude expérimentale avec x lignes pour un individu, il faudra utiliser d'autres méthodes.

```
nrow(ma_bdd)
```

[Exercice 21]

B) Variables qualitatives : créer des tableaux d'effectifs et de fréquences

Pour décrire une variable qualitative, il faut connaître les effectifs et la fréquence de chaque modalité de la variable.

1. Les effectifs

La fonction table() nous permet de générer un tableau d'effectifs

```
table(ma_bdd$ma_variable)
```

[Exercice 22]

2. Les fréquences

La fonction prop.table() permet de transformer un tableau d'effectifs (généré par la fonction table ici) en un tableau de proportion. Attention, le tableau généré de cette manière affichera des valeurs entre 0 et 1, et non des pourcentages.

```
prop.table(table(ma_bdd$ma_variable))
```

Si l'on souhaite afficher un pourcentage, il faut faire la manipulation suivante

```
prop.table(table(ma_bdd$ma_variable))*100
```

Il est également possible d'arrondir le résultat avec la fonction round().

```
round(prop.table(table(ma_bdd$ma_variable))*100, digits = 2)
```

[Exercice 23]

C) Croiser deux variables qualitatives

Les fonctions précédemment citées peuvent être utilisées afin de créer des tableaux croisant 2 variables.

1. Les effectifs

```
table(ma_bdd$ma_variable_ligne, ma_bdd$ma_variable_colonne)
```

[Exercice 24]

2. Les fréquences

```
Prop.table(table(ma_bdd$ma_variable_ligne, ma_bdd$ma_variable_colonne), margin = 1)
```

Le paramètre "margin =" permet de désigner si les pourcentages sont calculés sur le total « lignes » (1) ou sur les « colonnes » (2). Si cet argument n'est pas renseigné, R calculera les pourcentages sur le total global

[Exercice 25]

D) Variables quantitatives : statistiques descriptives

Pour décrire une variable quantitative, il faut calculer ses statistiques descriptives. La fonction summary() permet de le faire.

```
summary(ma_bdd$ma_variable)
```

Il existe des fonctions dédiées pour calculer séparément chaque statistique (voir la fiche mémo avec les fonctions clés).

[Exercice 26]

E) Croiser une variable quantitative avec une variable qualitative

Parfois, nous souhaitons connaître les statistiques descriptives d'une variable quantitative en fonction d'une variable qualitative. La fonction aggregate() vous permettra de lancer une fonction tiers sur une variable X (quantitative) en fonction d'une variable Y (qualitative).

```
aggregate(x = ma_bdd$ma_variable_quanti, by = list(ma_bdd$ma_variable_quali), FUN = "ma_fonction")
```

[Exercice 27]

8. Illustrer sa base de données

Il pourra vous arriver de vouloir générer des graphiques en-dehors d'analyses statistiques poussées. R dispose de différentes fonctions permettant de créer des graphiques.

Dans les paragraphes qui vont suivre, le paramétrage des graphiques restera basique. Cependant, il existe une multitude d'options permettant de personnaliser des graphiques jusqu'au moindre détail.

A) Une variable qualitative

La fonction barplot() permet de mettre sous forme de diagrammes en barres un tableau de contingence.

1. Méthode 1 : je n'ai pas encore d'objet contenant mon tableau de contingence

```
barplot(table(ma_bdd$ma_variable), main = "Titre de mon graphique",  xlab = NULL, ylab = "Unité de valeur", ylim = 1500)
```

Remarque : pour l'exemple, la fonction table est utilisée. Mais il est tout à fait possible d'utiliser la fonction prop.table().

2. Méthode 2 : j'ai un objet contenant mon tableau de contingence

```
barplot(monObjet, main = "Titre de mon graphique",  xlab = NULL, ylab = "Unité de valeur", ylim = 1500)
```

Il est à noter que puisque la fonction se base sur un tableau préalablement généré, en fonction du type de tableau renseigné (effectifs ou pourcentages), nous pourrons générer soit un graphique des effectifs, soit un graphique des pourcentages.

[Exercice 28]

B) Deux variables qualitatives

La fonctionne barplot fonctionne également lorsque nous souhaitons croiser deux variables qualitatives. Il suffit d'utiliser un tableau croisant deux variables au lieu d'un tableau à une seule variable.

3. Méthode 1 : je n'ai pas encore d'objet contenant mon tableau de contingence

```
barplot(table(ma_bdd$ma_variable_ligne, ma_bdd$ma_variable_colonne), beside = TRUE, main = "Titre de mon graphique", legend.text = malistedemodalites)
```

Remarque : pour l'exemple, la fonction table est utilisée. Mais il est tout à fait possible d'utiliser la fonction prop.table().

4. Méthode 2 : j'ai un objet contenant mon tableau de contingence

```
barplot(monObjet, beside = TRUE, main = "Titre de mon graphique", legend.text = malistedemodalites)
```

Comme pour le graphique univarié, en fonction du type de tableau renseigné (effectifs ou pourcentages), cela génèrera un graphique d'effectifs ou de pourcentages.

Pour le cas des pourcentages, comme nous croisons deux variables, il faudra faire attention au paramétrage du calcul des pourcentages (lignes/colonne/total général) pour que les données soient cohérentes avec la construction du graphique.

[Exercice 29]

C) Une variable quantitative

Les histogrammes sont une manière assez simple de représenter la répartition des individus selon une variable quantitatives. La fonction hist() permet de le faire.

```
hist(ma_bdd$ma_variable, main = "Titre graphique", xlab = "Titre des abscisses ", ylab = "Titre des ordonnées")
```

[Exercice 30]

Il est également possible de générer une boîte à moustache (boxplot).

```
boxplot(ma_bdd$ma_variable, main = "Titre graphique", ylab = "Titre des ordonnées")
```

[Exercice 31]

D) Deux variables quantitatives

Pour croiser deux variables quantitatives, il suffit de créer un graphique en nuages de points. La fonction plot() permet d'en générer.

```
plot(x = ma_bdd$ma_variable_en_abscisses, y = ma_bdd$ma_variable_en_ordonneées, main = "Titre du graphique", xlab = "Libellé des abscisses", ylab = "Libellé des ordonnées")
```

[Exercice 32]

E) Une variable quantitative croisée avec une variable qualitative

Les boîtes à moustache peuvent être couplées à une variable qualitative afin de permettre une représentation graphique d'une variable quantitative croisée avec une variable qualitative.

```
boxplot(ma_variable_quanti ~ ma_variable_quali, data = ma_bdd, main = "Titre du graphique", xlab="Nom variable quali", ylab = "Nom variable quanti", outline = FALSE, horizontal = TRUE)
```

[Exercice 33]

9. Exporter ses résultats

A) Exporter une base de données.

1. Au format Excel (xslx)

Le package openxlsx que nous avons précédemment vu permet également de faire des exportations de base grâce à la fonction write.xlsx().

```
wirte.xlsx(mon_objet_contenant_ma_bdd, "nom_du_fichier_excel.xslx", sheetName = "nom_de_longlet")
```

Il est à noter que vous pouvez exporter tous les objets que vous souhaitez au format Excel, ils n'ont pas besoin d'être sous la forme d'une base de données.

[Exercice 34]

2. Au format SPSS/Jamovi (sav)

Le package haven permet d'importer et d'exporter de bases au format .sav, reconnu par SPSS et Jamovi. Pour l'exportation, il faut utiliser la fonction write_sav().

```
write_sav(mon_objet_contenant_ma_bdd, "nom_du_fichier.sav")
```

[Exercice 35]

B) Exporter des graphiques

Il est possible d'exporter un graphique au format png (ou d'autres formats) lorsque nous le générons sur R.

```
png(filename = "nom_du_fichier.png", width = 600, height = 600)

ma_fonction_générant_un_graphique(mes_instruction, [..])

dev.off()
```

Remarque : width correspond à la largeur de l'image, height à sa hauteur en pixels.

[Exercice 36]