Cours d'Analyse de Données avec R

Priestley

2022-04-21

Contents

Chapter 1

The pool of tears

Chapter 2

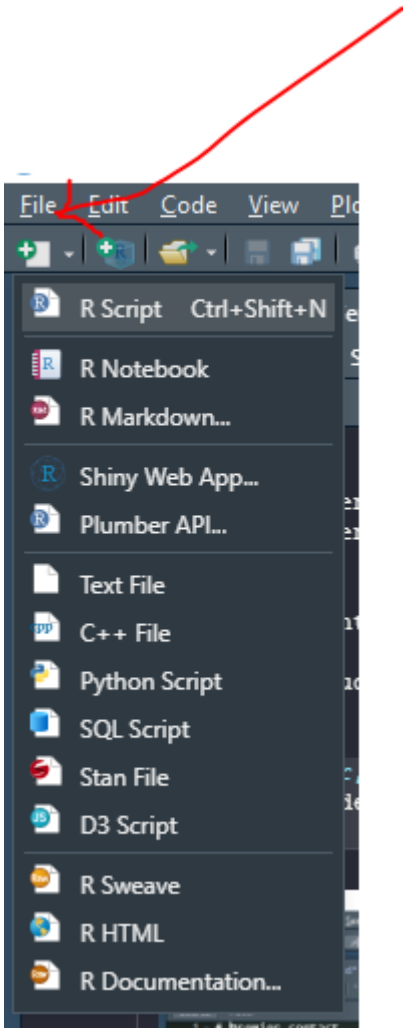
PREMIER TRAVAIL AVEC LES DONNEES

2.1 Regrouper les commandes dans des scripts

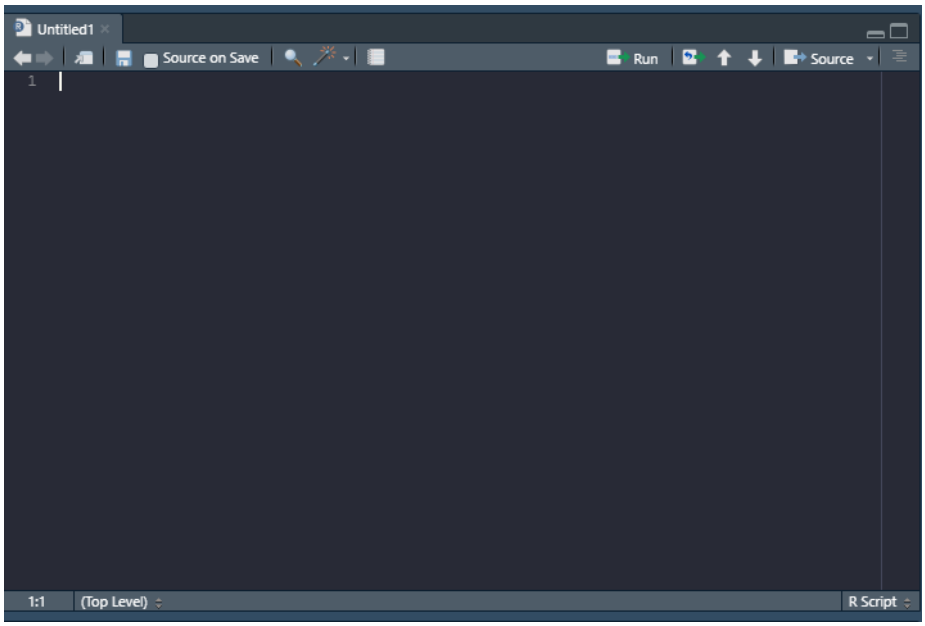
Jusqu'à maintenant nous avons utilisé uniquement la console pour communiquer avec R via l'invite de commandes. Le principal problème de ce mode d'interaction est qu'une fois qu'une commande est tapée, elle est pour ainsi dire « perdue », c'est-à-dire qu'on doit la saisir à nouveau si on veut l'exécuter une seconde fois. L'utilisation de la console est donc restreinte aux petites commandes « jetables », le plus souvent utilisées comme test.

La plupart du temps, les commandes seront stockées dans un fichier à part, que l'on pourra facilement ouvrir, éditer et exécuter en tout ou partie si besoin. On appelle en général ce type de fichier un script.

Pour comprendre comment cela fonctionne, dans RStudio cliquez sur l'icône en haut à gauche représentant un fichier avec un signe plus vert, puis choisissez R script.



Un nouvel onglet apparaît dans le quadrant supérieur gauche.



Nous pouvons désormais y saisir des commandes. Par exemple, tapez sur la première ligne la commande suivante : 2 + 2. Ensuite, cliquez sur l'icône Run (en haut à droite de l'onglet du script) ou bien pressez simulatément les touches CTRL et Entrée.

Les lignes suivantes ont dû faire leur apparition dans la console :

```
2+2
```

```
## [1] 4
```

Voici donc comment soumettre rapidement à R les commandes saisies dans votre fichier. Vous pouvez désormais l'enregistrer, l'ouvrir plus tard, et en exécuter tout ou partie. À noter que vous avez plusieurs possibilités pour soumettre des commandes à R :

vous pouvez exécuter la ligne sur laquelle se trouve votre curseur en cliquant sur Run ou en pressant simulatément les touches CTRL et Entrée ; vous pouvez sélectionner plusieurs lignes contenant des commandes et les exécuter toutes en une seule fois exactement de la même manière ; vous pouvez exécuter d'un coup l'intégralité de votre fichier en cliquant sur l'icône Source. La plupart du travail sous R consistera donc à éditer un ou plusieurs fichiers de commandes et à envoyer régulièrement les commandes saisies à R en utilisant les raccourcis clavier ad hoc.

2.1.1 Ajouter des commentaires

Un commentaire est une ligne ou une portion de ligne qui sera ignorée par R. Ceci signifie qu'on peut y écrire ce qu'on veut et qu'on va les utiliser pour ajouter tout un tas de commentaires à notre code permettant de décrire les différentes étapes du travail, les choses à se rappeler, les questions en suspens, etc.

Un commentaire sous R commence par un ou plusieurs symboles # (qui s'obtient avec les touches Alt Gr et 3 sur les claviers de type PC). Tout ce qui suit ce symbole jusqu'à la fin de la ligne est considéré comme un commentaire. On peut créer une ligne entière de commentaire en la faisant débuter par ##. Par exemple :

```
## Tableau croisé de la CSP par le nombre de livres lus.
## Attention au nombre de non réponses !
```

On peut aussi créer des commentaires pour une ligne en cours :

```
x <- 2 # On met 2 dans x, parce qu'il le vaut bien
```

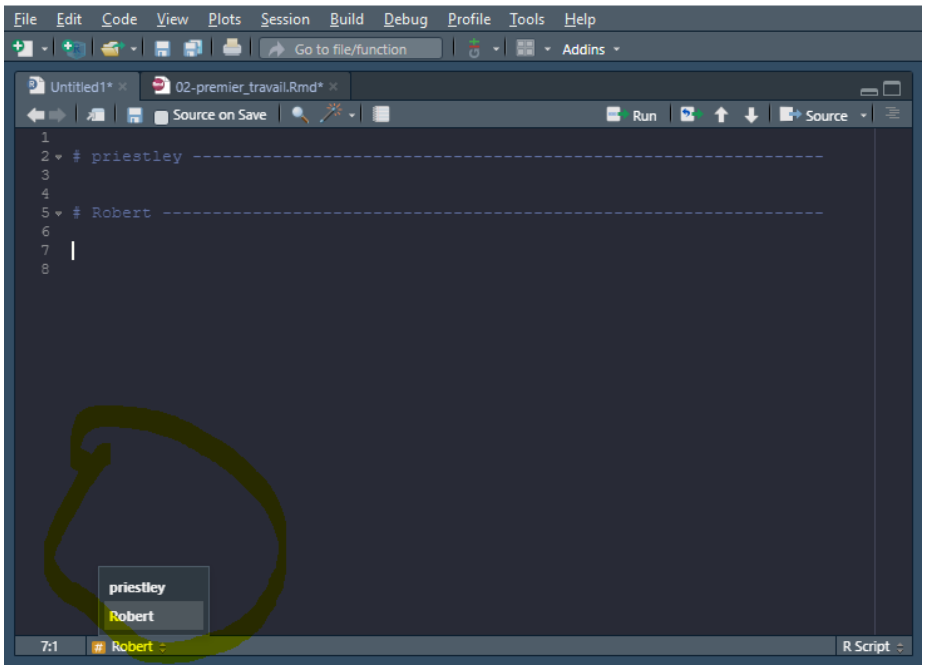
NB:*Dans tous les cas, il est très important de documenter ses fichiers R au fur et à mesure, faute de quoi on risque de ne plus y comprendre grand chose si on les reprend ne serait-ce que quelques semaines plus tard.*

Avec RStudio, vous pouvez également utiliser les commentaires pour créer des sections au sein de votre script et naviguer plus rapidement. Il suffit de faire suivre une ligne de commentaires d'au moins 4 signes moins (—-). Par exemple, si vous saisissez ceci dans votre script :

```
## Créer les objets ----

x <- 2
y <- 5

## Calculs ----

x + y
```

```
## [1] 7
```

Vous verrez apparaître en bas à gauche de la fenêtre du script un symbole dièse orange. Si vous cliquez dessus, un menu de navigation s'affichera vous permettant de vous déplacez rapidement au sein de votre script.



Note : on remarquera au passage que le titre de l'onglet est affiché en rouge et suivi d'une astérisque (*), nous indiquant ainsi qu'il y a des modifications non enregistrées dans notre fichier

2.2 Tableaux de données

Dans cette partie nous allons utiliser un jeu de données inclus dans l'extension questionr. L'installation d'extension est décrite dans le chapitre Extensions. Le jeu de données en question est un extrait de l'enquête Histoire de vie réalisée par l'INSEE en 2003. Il contient 2000 individus et 20 variables. Pour pouvoir utiliser ces données, il faut d'abord charger l'extension questionr (après l'avoir installée, bien entendu). Le chargement d'une extension en mémoire se fait à l'aide de la fonction library. Sous RStudio, vous pouvez également charger une extension en allant dans l'onglet Packages du quadrant inférieur droit qui liste l'ensemble des packages disponibles et en cliquant la case à cocher située à gauche du nom du package désiré.

```
library(questionr)
```

Puis nous allons indiquer à R que nous souhaitons accéder au jeu de données hdv2003 à l'aide de la fonction data :

```
data(hdv2003)
```

Bien. Et maintenant, elles sont où mes données ? Et bien elles se trouvent dans un objet nommé hdv2003 désormais chargé en mémoire et accessible directement. D'ailleurs, cet objet est maintenant visible dans l'onglet Environment du quadrant supérieur droit.

Essayons de taper son nom à l'invite de commande :

```
hdv2003
```

Le résultat (non reproduit ici) ne ressemble pas forcément à grand-chose… Il faut se rappeler que par défaut, lorsqu'on lui fournit seulement un nom d'objet, R essaye de l'afficher de la manière la meilleure (ou la moins pire) possible. La réponse à la commande hdv2003 n'est donc rien moins que l'affichage des données brutes contenues dans cet objet.

Ce qui signifie donc que l'intégralité de notre jeu de données est inclus dans l'objet nommé hdv2003 ! En effet, dans R, un objet peut très bien contenir un simple nombre, un vecteur ou bien le résultat d'une enquête tout entier. Dans ce cas, les objets sont appelés des data frames, ou tableaux de données. Ils peuvent être manipulés comme tout autre objet. Par exemple :

Résumons

Comme nous avons désormais décidé de saisir nos commandes dans un script et non plus directement dans la console, les premières lignes de notre fichier de travail sur les données de l'enquête Histoire de vie pourraient donc ressembler à ceci :

```
## Chargement des extensions nécessaires ----
library(questionr)
## Jeu de données hdv2003 ----
data(hdv2003)
d <- hdv2003
```

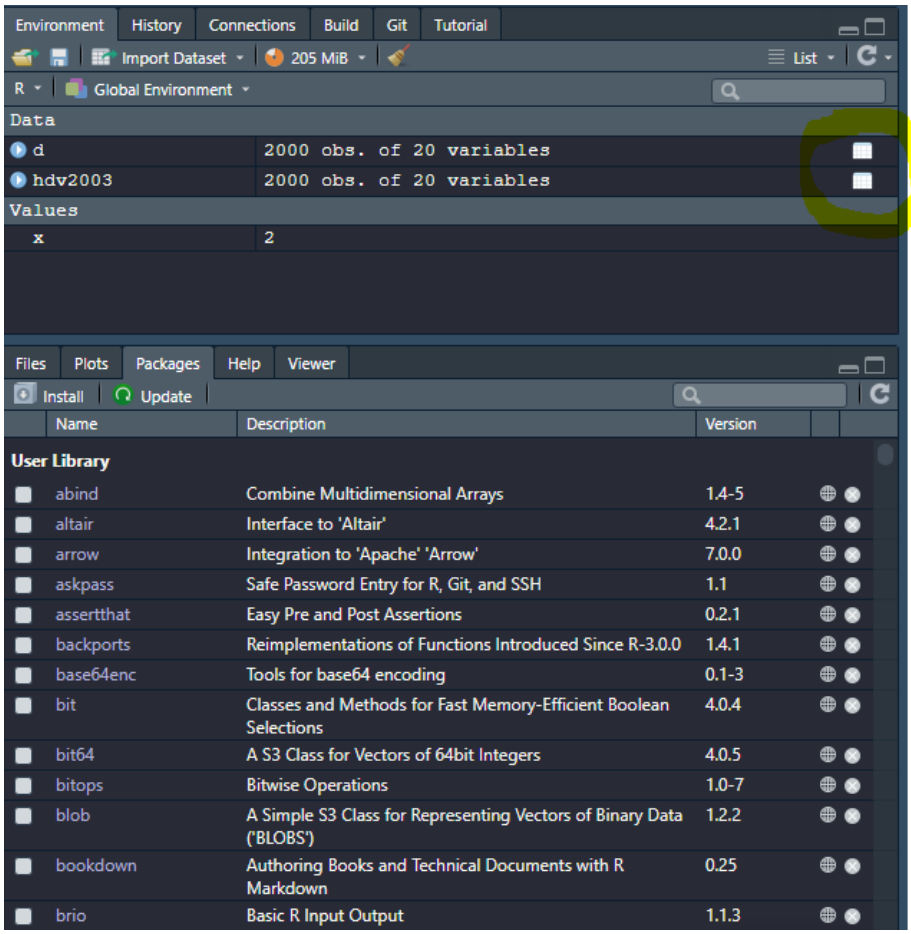
2.2.1 Inspection visuelle des données

La particularité de R par rapport à d'autres logiciels comme **Modalisa ou SPSS** est de ne pas proposer, par défaut, de vue des données sous forme de tableau. Ceci peut parfois être un peu déstabilisant dans les premiers temps d'utilisation, même si l'on perd vite l'habitude et qu'on finit par se rendre compte que « voir » les données n'est pas forcément un gage de productivité ou de rigueur dans le traitement.

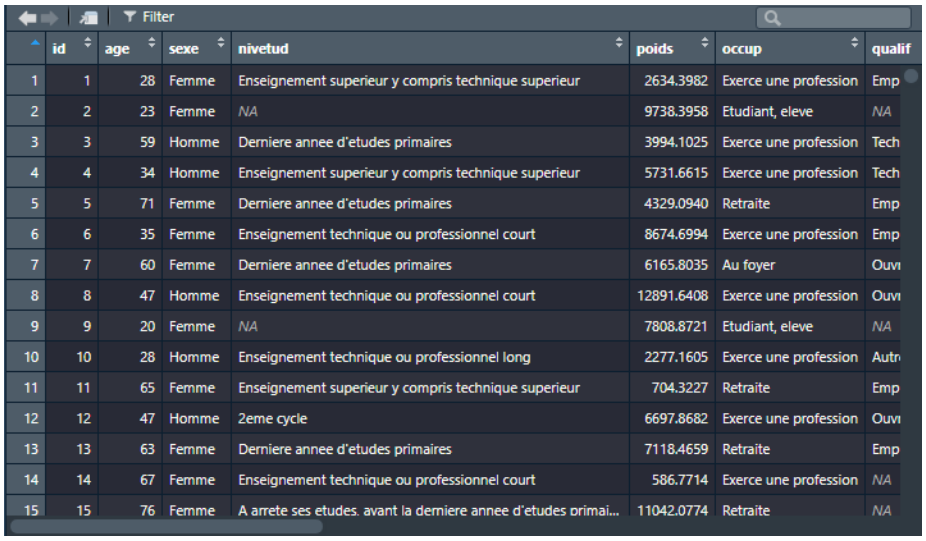
Néanmoins, R propose une interface permettant de visualiser le contenu d'un tableau de données à l'aide de la fonction **View** :

```
View(d)
```

Sous RStudio, on peut aussi afficher la visionneusee (viewer) en cliquant sur la petite icône en forme de tableau située à droite de la ligne d'un tableau de données dans l'onglet Environment du quadrant supérieur droit (cf. figure ci-après).



Dans tous les cas, RStudio lancera le viewer dans un onglet dédié dans le quadrant supérieur gauche. Le visualiseur de RStudio est plus avancé que celui-de base fournit par R. Il est possible de trier les données selon une variable en cliquant sur le nom de cette dernière. Il y a également un champs de recherche et un bouton Filter donnant accès à des options de filtrage avancées.



Filter

	id	age	sexe	nivetud	poids	occup	qualif
1	1	28	Femme	Enseignement superieur y compris technique superieur	2634.3982	Exerce une profession	Emp
2	2	23	Femme	*NA*	9738.3958	Etudiant, eleve	*NA*
3	3	59	Homme	Derniere annee d'etudes primaires	3994.1025	Exerce une profession	Tech
4	4	34	Homme	Enseignement superieur y compris technique superieur	5731.6615	Exerce une profession	Tech
5	5	71	Femme	Derniere annee d'etudes primaires	4329.0940	Retraite	Emp
6	6	35	Femme	Enseignement technique ou professionnel court	8674.6994	Exerce une profession	Emp
7	7	60	Femme	Derniere annee d'etudes primaires	6165.8035	Au foyer	Ouvr
8	8	47	Homme	Enseignement technique ou professionnel court	12891.6408	Exerce une profession	Ouvr
9	9	20	Femme	*NA*	7808.8721	Etudiant, eleve	*NA*
10	10	28	Homme	Enseignement technique ou professionnel long	2277.1605	Exerce une profession	Autr
11	11	65	Femme	Enseignement superieur y compris technique superieur	704.3227	Retraite	Emp
12	12	47	Homme	2eme cycle	6697.8682	Exerce une profession	Ouvr
13	13	63	Femme	Derniere annee d'etudes primaires	7118.4659	Retraite	Emp
14	14	67	Femme	Enseignement technique ou professionnel court	586.7714	Exerce une profession	*NA*
15	15	76	Femme	A arrete ses etudes, avant la derniere annee d'etudes primai...	11042.0774	Retraite	*NA*

Structure du tableau

Avant de travailler sur les données, nous allons essayer de comprendre comment elles sont structurées. Lors de l'import de données depuis un autre logiciel (que nous aborderons dans un autre chapitre), il s'agira souvent de vérifier que l'importation s'est bien déroulée.

Nous avons déjà vu qu'un tableau de données est organisé en lignes et en colonnes, les lignes correspondant aux observations et les colonnes aux variables. Les fonctions nrow, ncol et dim donnent respectivement le nombre de lignes, le nombre de colonnes et les dimensions de notre tableau. Nous pouvons donc d'ores et déjà vérifier que nnombre des lignes

```
nrow(d)#Nombre de lignes:
```

```
## [1] 2000
```

```
ncol(d)#nombre de colonnes :
```

```
## [1] 20
```

```
dim(d)# lignes x colonnes
```

```
## [1] 2000   20
```

La fonction **names** donne les noms des colonnes de notre tableau, c'est-à-dire les noms des variables :

```
names(d)
```

```
##  [1] "id"            "age"           "sexe"          "nivetud"
##  [5] "poids"         "occup"         "qualif"        "freres.soeurs"
##  [9] "clso"          "relig"         "trav.imp"      "trav.satisf"
## [13] "hard.rock"     "lecture.bd"    "peche.chasse"  "cuisine"
## [17] "bricol"        "cinema"        "sport"         "heures.tv"
```

2.2.2 Accéder aux variables

d représente donc l'ensemble de notre tableau de données. Nous avons vu que si l'on saisit simplement d à l'invite de commandes, on obtient un affichage du tableau en question. Mais comment accéder aux variables, c'est à dire aux colonnes de notre tableau ? La réponse est simple : on utilise le nom de l'objet, suivi de l'opérateur $, suivi du nom de la variable, comme ceci :

```
d$sexe
```

Au regard du résultat (non reproduit ici), on constate alors que R a bien accédé au contenu de notre variable sexe du tableau d et a affiché son contenu, c'est-à-dire l'ensemble des valeurs prises par la variable.

Les fonctions head et tail permettent d'afficher seulement les premières (respectivement les dernières) valeurs prises par la variable. On peut leur passer en argument le nombre d'éléments à afficher :

```
head(d$nivetud)# 6 premierès observations
```

```
## [1] Enseignement superieur y compris technique superieur
## [2] <NA>
## [3] Derniere annee d'etudes primaires
## [4] Enseignement superieur y compris technique superieur
## [5] Derniere annee d'etudes primaires
## [6] Enseignement technique ou professionnel court
## 8 Levels: N'a jamais fait d'etudes ...
```

```
tail(d$age, 10)#10 dernières observations
```

```
##  [1] 52 42 50 41 46 45 46 24 24 66
```

À noter que ces fonctions marchent aussi pour afficher les lignes du tableau d :

```
head(d,2)
```

```
##   id age  sexe                                                nivetud    poids
## 1  1  28 Femme Enseignement superieur y compris technique superieur 2634.398
## 2  2  23 Femme                                                  <NA> 9738.396
##                   occup  qualif freres.soeurs clso                      relig
## 1 Exerce une profession Employe             8  Oui Ni croyance ni appartenance
## 2       Etudiant, eleve    <NA>             2  Oui Ni croyance ni appartenance
##        trav.imp    trav.satisf hard.rock lecture.bd peche.chasse cuisine bricol
## 1 Peu important Insatisfaction       Non        Non          Non     Oui    Non
## 2          <NA>           <NA>       Non        Non          Non     Non    Non
##   cinema sport heures.tv
## 1    Non   Non         0
## 2    Oui   Oui         1
```

2.2.3 La fonction str

La fonction str est plus complète que names. Elle liste les différentes variables, indique leur type et donne le cas échéant des informations supplémentaires ainsi qu'un échantillon des premières valeurs prises par cette variable :

```
str(d)
```

```
## 'data.frame':    2000 obs. of  20 variables:
##  $ id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ age          : int  28 23 59 34 71 35 60 47 20 28 ...
##  $ sexe         : Factor w/ 2 levels "Homme","Femme": 2 2 1 1 2 2 2 1 2 1 ...
##  $ nivetud      : Factor w/ 8 levels "N'a jamais fait d'etudes",..: 8 NA 3 8 3 6 3 6
##  $ poids        : num  2634 9738 3994 5732 4329 ...
##  $ occup        : Factor w/ 7 levels "Exerce une profession",..: 1 3 1 1 4 1 6 1 3 1
##  $ qualif       : Factor w/ 7 levels "Ouvrier specialise",..: 6 NA 3 3 6 6 2 2 NA 7
##  $ freres.soeurs: int  8 2 2 1 0 5 1 5 4 2 ...
##  $ clso         : Factor w/ 3 levels "Oui","Non","Ne sait pas": 1 1 2 2 1 2 1 2 1 2
##  $ relig        : Factor w/ 6 levels "Pratiquant regulier",..: 4 4 4 3 1 4 3 4 3 2
##  $ trav.imp     : Factor w/ 4 levels "Le plus important",..: 4 NA 2 3 NA 1 NA 4 NA 3
##  $ trav.satisf  : Factor w/ 3 levels "Satisfaction",..: 2 NA 3 1 NA 3 NA 2 NA 1 ...
##  $ hard.rock    : Factor w/ 2 levels "Non","Oui": 1 1 1 1 1 1 1 1 1 1 ...
##  $ lecture.bd   : Factor w/ 2 levels "Non","Oui": 1 1 1 1 1 1 1 1 1 1 ...
##  $ peche.chasse : Factor w/ 2 levels "Non","Oui": 1 1 1 1 1 1 2 2 1 1 ...
##  $ cuisine      : Factor w/ 2 levels "Non","Oui": 2 1 1 2 1 1 2 2 1 1 ...
##  $ bricol       : Factor w/ 2 levels "Non","Oui": 1 1 1 2 1 1 1 2 1 1 ...
##  $ cinema       : Factor w/ 2 levels "Non","Oui": 1 2 1 2 1 2 1 1 2 2 ...
##  $ sport        : Factor w/ 2 levels "Non","Oui": 1 2 2 2 1 2 1 1 1 2 ...
##  $ heures.tv    : num  0 1 0 2 3 2 2.9 1 2 2 ...
```

La première ligne nous informe qu'il s'agit bien d'un tableau de données avec 2000 observations et 20 variables. Vient ensuite la liste des variables. La première se nomme id et est de type entier (int). La seconde se nomme age et est de type numérique. La troisième se nomme sexe, il s'agit d'un facteur (factor).

Un facteur est une variable pouvant prendre un nombre limité de modalités (levels). Ici notre variable a deux modalités possibles : « Homme » et « Femme ». Ce type de variable est décrit plus en détail dans le chapitre sur la manipulation de données.

Important

La fonction str est essentielle à connaître et peut s'appliquer à n'importe quel type d'objet. C'est un excellent moyen de connaître en détail la structure d'un objet. Cependant, les résultats peuvent être parfois trop détaillés et on lui priviligiera dans certains cas la fonction describe que l'on abordera dans les prochains chapitres, cependant moins générique puisque ne s'appliquant qu'à des tableaux de données et à des vecteurs, tandis que str peut s'appliquer à absolument tout objet, y compris des fonctions.

```
describe(d)
```

```
## [2000 obs. x 20 variables] tbl_df tbl data.frame
##
## $id:
## integer: 1 2 3 4 5 6 7 8 9 10 ...
## min: 1 - max: 2000 - NAs: 0 (0%) - 2000 unique values
##
## $age:
## integer: 28 23 59 34 71 35 60 47 20 28 ...
## min: 18 - max: 97 - NAs: 0 (0%) - 78 unique values
##
## $sexe:
## nominal factor: "Femme" "Femme" "Homme" "Homme" "Femme" "Femme" "Femme" "Homme" "Femme" "Homme
## 2 levels: Homme | Femme
## NAs: 0 (0%)
##
## $nivetud:
## nominal factor: "Enseignement superieur y compris technique superieur" NA "Derniere annee d'et
## 8 levels: N'a jamais fait d'etudes | A arrete ses etudes, avant la derniere annee d'etudes pri
## NAs: 112 (5.6%)
##
## $poids:
## numeric: 2634.3982157 9738.3957759 3994.1024587 5731.6615081 4329.0940022 8674.6993828 6165.80
## min: 78.0783403 - max: 31092.14132 - NAs: 0 (0%) - 1877 unique values
##
## $occup:
```

```
## nominal factor: "Exerce une profession" "Etudiant, eleve" "Exerce une profession" "
## 7 levels: Exerce une profession | Chomeur | Etudiant, eleve | Retraite | Retire des
## NAs: 0 (0%)
##
## $qualif:
## nominal factor: "Employe" NA "Technicien" "Technicien" "Employe" "Employe" "Ouvrier
## 7 levels: Ouvrier specialise | Ouvrier qualifie | Technicien | Profession intermedia
## NAs: 347 (17.3%)
##
## $freres.soeurs:
## integer: 8 2 2 1 0 5 1 5 4 2 ...
## min: 0 - max: 22 - NAs: 0 (0%) - 19 unique values
##
## $clso:
## nominal factor: "Oui" "Oui" "Non" "Non" "Oui" "Non" "Oui" "Non" "Oui" "Non" ...
## 3 levels: Oui | Non | Ne sait pas
## NAs: 0 (0%)
##
## $relig:
## nominal factor: "Ni croyance ni appartenance" "Ni croyance ni appartenance" "Ni croy
## 6 levels: Pratiquant regulier | Pratiquant occasionnel | Appartenance sans pratique
## NAs: 0 (0%)
##
## $trav.imp:
## nominal factor: "Peu important" NA "Aussi important que le reste" "Moins important q
## 4 levels: Le plus important | Aussi important que le reste | Moins important que le
## NAs: 952 (47.6%)
##
## $trav.satisf:
## nominal factor: "Insatisfaction" NA "Equilibre" "Satisfaction" NA "Equilibre" NA "In
## 3 levels: Satisfaction | Insatisfaction | Equilibre
## NAs: 952 (47.6%)
##
## $hard.rock:
## nominal factor: "Non" "Non" "Non" "Non" "Non" "Non" "Non" "Non" "Non" "Non" ...
## 2 levels: Non | Oui
## NAs: 0 (0%)
##
## $lecture.bd:
## nominal factor: "Non" "Non" "Non" "Non" "Non" "Non" "Non" "Non" "Non" "Non" ...
## 2 levels: Non | Oui
## NAs: 0 (0%)
##
## $peche.chasse:
## nominal factor: "Non" "Non" "Non" "Non" "Non" "Non" "Oui" "Oui" "Non" "Non" ...
## 2 levels: Non | Oui
```

```
## NAs: 0 (0%)
##
## $cuisine:
## nominal factor: "Oui" "Non" "Non" "Oui" "Non" "Non" "Oui" "Oui" "Non" "Non" ...
## 2 levels: Non | Oui
## NAs: 0 (0%)
##
## $bricol:
## nominal factor: "Non" "Non" "Non" "Oui" "Non" "Non" "Non" "Oui" "Non" "Non" ...
## 2 levels: Non | Oui
## NAs: 0 (0%)
##
## $cinema:
## nominal factor: "Non" "Oui" "Non" "Oui" "Non" "Oui" "Non" "Non" "Oui" "Oui" ...
## 2 levels: Non | Oui
## NAs: 0 (0%)
##
## $sport:
## nominal factor: "Non" "Oui" "Oui" "Oui" "Non" "Oui" "Non" "Non" "Non" "Oui" ...
## 2 levels: Non | Oui
## NAs: 0 (0%)
##
## $heures.tv:
## numeric: 0 1 0 2 3 2 2.9 1 2 2 ...
## min: 0 - max: 12 - NAs: 5 (0.2%) - 30 unique values
```

2.2.4 Quelques calculs simples

Maintenant que nous savons accéder aux variables, effectuons quelques calculs simples comme la moyenne, la médiane, le minimum et le maximum, à l'aide des fonctions mean, median, min et max.

```
mean(d$age)
```

```
## [1] 48.157
```

```
median(d$age)
```

```
## [1] 48
```

```
min(d$age)
```

```
## [1] 18
```

```
max(d$age)
```

```
## [1] 97
```

::: {#hello .greeting .message style="color: red;"} Au sens strict, il ne s'agit pas d'un véritable âge moyen puisqu'il faudrait ajouter 0,5 à cette valeur calculée, un âge moyen se calculant à partir d'âges exacts et non à partir d'âges révolus. On peut aussi très facilement obtenir un tri à plat à l'aide la fonction table :

NOTICE!

Thank you for noticing this **new notice**! Your noticing it has been noted, and *will be reported to the authorities*!