

# Statistiques et Probabilités avec R, RStudio et le Tidyverse

*Marc-André Désautels*

*2018-04-17*



# Table des Matières

|   |           |
|---|-----------|
| <b>Introduction</b>                             | <b>5</b>  |
| <b>I Les outils</b>                             | <b>7</b>  |
| <b>1 Les logiciels R et RStudio</b>             | <b>9</b>  |
| 1.1 Qu'est-ce que R?                            | 9         |
| 1.2 Qu'est-ce que RStudio?                      | 10        |
| 1.3 Les bases de RStudio                        | 10        |
| <b>2 Le tidyverse</b>                           | <b>11</b> |
| 2.1 Extensions                                  | 11        |
| 2.2 Installation                                | 12        |
| 2.3 Les tidy data                               | 12        |
| 2.4 Les tibbles                                 | 13        |
| <b>3 L'extension questionr</b>                  | <b>19</b> |
| 3.1 Mise en place                               | 19        |
| 3.2 L'interface graphique                       | 19        |
| <b>II Introduction</b>                          | <b>21</b> |
| <b>4 La démarche scientifique</b>               | <b>23</b> |
| <b>5 Les différents types de variables</b>      | <b>25</b> |
| 5.1 Introduction                                | 25        |
| 5.2 Les variables qualitatives                  | 25        |
| 5.3 Les variables quantitatives                 | 28        |
| <b>6 Construire un questionnaire</b>            | <b>31</b> |
| 6.1 Critères à respecter                        | 31        |
| 6.2 Types de questions                          | 31        |
| <b>7 Les échelles de mesure</b>                 | <b>33</b> |
| <b>8 Les techniques d'échantillonnage</b>       | <b>35</b> |
| 8.1 Techniques d'échantillonnage aléatoires     | 35        |
| 8.2 Techniques d'échantillonnage non-aléatoires | 35        |
| 8.3 Base de données pour les <i>M&amp;M's</i>   | 36        |

|   |           |
|---|-----------|
| <b>III Présentation des données</b>                             | <b>45</b> |
| <b>9 Les variables qualitatives</b>                             | <b>47</b> |
| 9.1 Mise en place . . . . .                                     | 47        |
| 9.2 Tableau de fréquences . . . . .                             | 47        |
| 9.3 Représentation graphique - Le diagramme à bandes . . . . .  | 48        |
| <b>10 Les variables quantitatives discrètes</b>                 | <b>51</b> |
| 10.1 Mise en place . . . . .                                    | 51        |
| 10.2 Tableau de fréquences . . . . .                            | 51        |
| 10.3 Représentation graphique - Le diagramme à bandes . . . . . | 55        |
| <b>11 Les variables quantitatives continues</b>                 | <b>57</b> |
| 11.1 Mise en place . . . . .                                    | 57        |
| 11.2 Tableau de fréquences . . . . .                            | 57        |
| 11.3 Représentation graphique - L'histogramme . . . . .         | 59        |
| <b>12 Les différentes mesures</b>                               | <b>61</b> |
| 12.1 Les mesures de tendance centrale . . . . .                 | 61        |
| 12.2 Les mesures de dispersion . . . . .                        | 63        |
| 12.3 Les mesures de position . . . . .                          | 64        |

# Introduction

Nous sommes constamment bombardés d'information. Que ce soit sur Internet, à la télévision ou à la radio, les résultats de sondage abondent. Comment faire pour déterminer quelle information est fiable?

Ce cours vise à faire de vous des citoyens critiques, capables d'analyser des données et d'en tirer des conclusions.



# Partie I

## Les outils





# Chapitre 1

## Les logiciels R et RStudio

Ce chapitre est inspiré de (Barnier, 2018) et de (Ismay, 2018).

### 1.1 Qu'est-ce que R?

R est un langage orienté vers le traitement et l'analyse quantitative de données. Il est développé depuis les années 90 par un groupe de volontaires de différents pays et par une large communauté d'utilisateurs. C'est un logiciel libre, publié sous licence GNU GPL. R a été créé par **Ross Ihaka** et **Robert Gentleman** en Nouvelle-Zélande à l'Université d'Auckland.

Voici les avantages les plus importants de R:

1. R est un logiciel gratuit.
2. R est un logiciel très puissant, dont les fonctionnalités de base peuvent être étendues à l'aide d'extensions développées par la communauté. Il en existe plusieurs milliers.
3. R est un logiciel dont le développement est très actif et dont la communauté d'utilisateurs et l'usage ne cessent de s'agrandir.
4. Il est possible de trouver des réponses à ses questions assez facilement grâce à l'aide incluse, à la communauté, à Google, etc. Bien que l'aide soit en anglais, il existe des communautés francophones qui utilisent le logiciel.
5. R n'est pas un logiciel au sens classique du terme, mais plutôt un langage de programmation. Il fonctionne à l'aide de scripts (des petits programmes) édités et exécutés au fur et à mesure de l'analyse. Ce point, qui peut apparaître comme un gros handicap, s'avère après un temps d'apprentissage être un mode d'utilisation d'une grande souplesse.

L'aspect langage de programmation et la difficulté qui en découle peuvent sembler des inconvénients importants. Le fait de structurer ses analyses sous forme de scripts (suite d'instructions effectuant les différentes opérations d'une analyse) présente cependant de nombreux avantages :

- le script garde par ordre chronologique l'ensemble des étapes d'une analyse, de l'importation des données à leur analyse en passant par les manipulations et les recodages
- on peut à tout moment revenir en arrière et modifier ce qui a été fait
- il est très rapide de réexécuter une suite d'opérations complexes
- on peut très facilement mettre à jour les résultats en cas de modification des données sources
- le script garantit, sous certaines conditions, la reproductibilité des résultats obtenus

Pour télécharger le logiciel R, vous allez à l'adresse suivante:

<https://www.r-project.org/>

## 1.2 Qu'est-ce que RStudio?

RStudio n'est pas à proprement parler une interface graphique pour R, il s'agit plutôt d'un *environnement de développement intégré* (*integrated development environment* en anglais), qui propose des outils et facilite l'écriture de scripts et l'usage de R au quotidien. C'est une interface bien supérieure à celles fournies par défaut lorsqu'on installe R sous Windows ou sous Mac.

Il existe plusieurs versions de RStudio:

- RStudio Desktop
- RStudio Server
- RStudio Cloud

Pour télécharger la version *Desktop* de RStudio (que vous pouvez utiliser sur votre ordinateur), vous allez à l'adresse suivante:

<https://www.rstudio.com/products/rstudio/download/#download>

## 1.3 Les bases de RStudio

# Chapitre 2

## Le tidyverse

Dans ce document, nous utiliserons l’extension `tidyverse` par (Wickham, 2017b). Ce chapitre permettra d’introduire l’extension `tidyverse` mais surtout les principes qui la sous-tendent. Ce chapitre est inspiré de (Barnier, 2018) et (Wickham and Grolemund, 2017).

```
library(tidyverse)
library(questionr)
```

### 2.1 Extensions

Le terme *tidyverse* est une contraction de *tidy* (qu’on pourrait traduire par *bien rangé*) et de *universe*. En allant visiter le site internet de ces extensions <https://www.tidyverse.org/>, voici ce que nous pouvons trouver sur la première page du site:

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

que nous pourrions traduire par:

Le tidyverse est une collection dogmatique d’extensions pour le langage R conçues pour la science des données. Toutes les extensions partagent une philosophie sous-jacente de design, de grammaire et de structures de données.

Ces extensions abordent un très grand nombre d’opérations courantes dans R. L’avantage d’utiliser le `tidyverse` c’est qu’il permet de simplifier plusieurs opérations fréquentes et il introduit le concept de **tidy data**. De plus, la grammaire du `tidyverse` étant cohérente entre toutes ses extensions, en apprenant comment utiliser l’une de ces extensions, vous serez en monde connu lorsque viendra le temps d’apprendre de nouvelles extensions.

Nous utiliserons le `tidyverse` pour:

- Le concept de **tidy data**
- L’importation et/ou l’exportation de données
- La manipulation de variables
- La visualisation

Le `tidyverse` permet aussi de:

- Travailler avec des chaînes de caractères (du texte par exemple)
- Programmer
- Remettre en forme des données

- Extraire des données du Web
- Etc.

Pour en savoir plus, nous invitons le lecteur à se rendre au site du **tidyverse** <https://www.tidyverse.org/>. Le **tidyverse** est en grande partie issu des travaux de [Hadley Wickham](#).

## 2.2 Installation

Pour installer les extensions du **tidyverse**, nous effectuons la commande suivante:

```
install.packages("tidyverse")
```

Une fois l'extension installée, il n'est pas nécessaire de la réinstaller à chaque fois que vous utilisez R. Par contre, vous devez charger l'extension à chaque fois que vous utilisez R.

Pour charger l'extension et l'utiliser dans R, nous effectuons la commande suivante:

```
library(tidyverse)
```

Cette commande va en fait charger plusieurs extensions qui constituent le **cœur** du **tidyverse**, à savoir :

- **ggplot2** (visualisation)
- **dplyr** (manipulation des données)
- **tidyr** (remise en forme des données)
- **purrr** (programmation)
- **readr** (importation de données)
- **tibble** (tableaux de données)
- **forcats** (variables qualitatives)
- **stringr** (chaînes de caractères)

Il existe d'autres extensions qui font partie du **tidyverse** mais qui doivent être chargées explicitement, comme par exemple **readxl** (pour l'importation de données depuis des fichiers Excel).

La liste complète des extensions se trouve sur le site officiel du **tidyverse** <https://www.tidyverse.org/packages/>.

## 2.3 Les tidy data

Le **tidyverse** est en partie fondé sur le concept de *tidy data*, développé à l'origine par Hadley Wickham dans un article du *Journal of Statistical Software*, voir ([Wickham, 2014](#)). Nous pourrions traduire ce concept par *données bien rangées*.

Il s'agit d'un modèle d'organisation des données qui vise à faciliter le travail souvent long et fastidieux de nettoyage et de préparation préalable à la mise en oeuvre de méthodes d'analyse. Dans ce livre, nous travaillerons toujours avec des *tidy data*. En réalité, la plupart des données rencontrées par les chercheurs ne sont pas *tidy*. Il existe une extension du **tidyverse** qui permet de faciliter la transformation de données *non tidy* en données *tidy*, l'extension **tidyr**. Nous ne verrons pas comment l'utiliser dans ce livre.

Les principes d'un jeu de données *tidy* sont les suivants :

1. chaque variable est une colonne
2. chaque observation est une ligne
3. chaque valeur doit être dans une cellule différente

La figure [2.1](#) montre ces règles de façon visuelle (l'image a été prise de ([Wickham and Grolemund, 2017](#))).

Pourquoi s'assurer que vos données sont *tidy*? Il y a deux avantages importants:

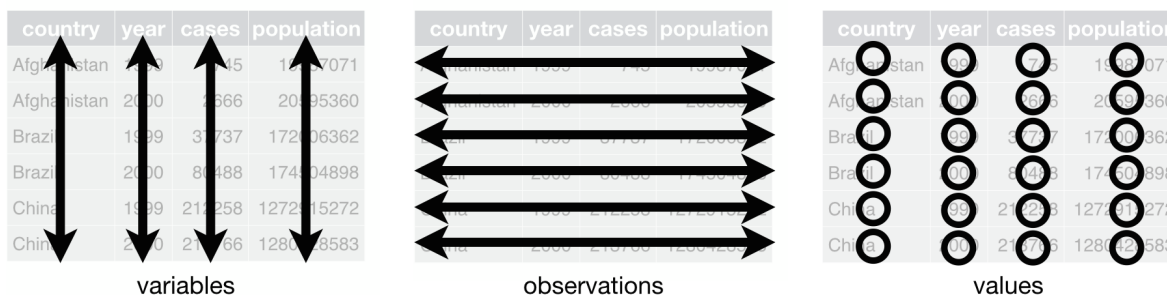


Figure 2.1: Suivre les trois principes rend les données tidy: les variables sont en colonnes, les observations sont sur des lignes, et chaque valeur est dans une cellule différente.

1. Un avantage général de choisir une seule façon de conserver vos données. Si vous utilisez une structure de données consistante, il est plus facile d'apprendre à utiliser les outils qui fonctionneront avec ce type de structure, étant donné que celles-ci possèdent une uniformité sous-jacente.
2. Un avantage spécifique de placer les variables en colonnes car ceci permet de *vectoriser* les opérations dans R. Ceci implique que vos fonctions seront plus rapides lorsque viendra le temps de les exécuter.

Voici un exemple de données *tidy* qui sont accessibles en R de base.

```
as_tibble(rownames_to_column(mtcars))
## # A tibble: 32 x 12
##   rowname      mpg   cyl  disp    hp  drat    wt   qsec    vs    am  gear
##   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Mazda RX4      21.0     6.  160.   110.  3.90  2.62  16.5     0.     1.     4.
## 2 Mazda RX4 W~  21.0     6.  160.   110.  3.90  2.88  17.0     0.     1.     4.
## 3 Datsun 710     22.8     4.  108.    93.  3.85  2.32  18.6     1.     1.     4.
## 4 Hornet 4 Dr~  21.4     6.  258.   110.  3.08  3.22  19.4     1.     0.     3.
## 5 Hornet Spor~  18.7     8.  360.   175.  3.15  3.44  17.0     0.     0.     3.
## 6 Valiant       18.1     6.  225.   105.  2.76  3.46  20.2     1.     0.     3.
## # ... with 26 more rows, and 1 more variable: carb <dbl>
```

## 2.4 Les tibbles

Une autre particularité du *tidyverse* est que ces extensions travaillent avec des tableaux de données au format *tibble*, qui est une évolution plus moderne du classique *data frame* du R de base. Ce format est fourni et géré par l'extension du même nom (`tibble`), qui fait partie du cœur du *tidyverse*. La plupart des fonctions des extensions du *tidyverse* acceptent des *data frames* en entrée, mais retournent un objet de classe `tibble`.

Pour être en mesure d'effectuer des calculs statistiques, il nous faut une structure qui soit en mesure de garder en mémoire une base de données. Ces structures se nomment des “tibbles” dans R.

### 2.4.1 Prérequis

Pour être en mesure d'utiliser le paquetage `tibble`, nous devons charger l'extension `tibble`. Pour ce faire, il suffit d'utiliser la commande suivante:

```
library(tibble)
```

## 2.4.2 Un exemple de tibble

Pour comprendre ce qu'est un **tibble**, nous allons utiliser deux librairies: `nycflights13` et `diamonds`. Si ce n'est pas déjà fait, vous devez les installer et ensuite les charger.

```
library(nycflights13)
library(ggplot2)
```

Nous allons étudier le paquetage `nycflights13` qui contient 5 bases de données contenant des informations concernant les vols intérieurs en partance de New York en 2013, à partir des aéroports de Newark Liberty International (EWR), John F. Kennedy International (JFK) ou LaGuardia (LGA). Les 5 bases de données sont les suivantes:

- `flights`: information sur les 336,776 vols
- `airlines`: lien entre les codes IATA de deux lettres et les noms de compagnies d'aviation (16 au total)
- `planes`: information de construction sur les 3 322 avions utilisés
- `weather`: données météo à chaque heure (environ 8 710 observations) pour chacun des trois aéroports.
- `airports`: noms des aéroports et localisations

## 2.4.3 La base de données flights

Pour visualiser facilement une base de données sous forme **tibble**, il suffit de taper son nom dans la console. Nous allons utiliser la base de données `flights`. Par exemple:

```
flights
## # A tibble: 336,776 x 19
##   year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>   <int>         <int>      <dbl>   <int>
## 1  2013     1     1     517             515         2.     830
## 2  2013     1     1     533             529         4.     850
## 3  2013     1     1     542             540         2.     923
## 4  2013     1     1     544             545        -1.    1004
## 5  2013     1     1     554             600        -6.     812
## 6  2013     1     1     554             558        -4.     740
## # ... with 3.368e+05 more rows, and 12 more variables:
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dtm>
```

Nous allons décortiquer la sortie console:

- `A tibble: 336,776 x 19`: un **tibble** est une façon de représenter une base de données en R. Cette base de données possède:
  - 336 776 lignes
  - 19 colonnes correspondant aux 19 variables décrivant chacune des observations
- `year month day dep_time sched_dep_time dep_delay arr_time` sont différentes colonnes, en d'autres mots des variables, de cette base de données.
- Nous avons ensuite 10 lignes d'observations correspondant à 10 vols
- `... with 336,766 more rows, and 12 more variables`: nous indique que 336 766 lignes et 12 autres variables ne pouvaient pas être affichées à l'écran.

Malheureusement cette sortie écran ne nous permet pas d'explorer les données correctement. Nous verrons à la section 2.4.5 comment explorer des **tibbles**.

### 2.4.4 La base de données diamonds

La base de données **diamonds** est composée des variables suivantes:

- **price** : prix en dollars US
- **carat** : poids du diamant en grammes
- **cut** : qualité de la coupe (Fair, Good, Very Good, Premium, Ideal)
- **color** : couleur du diamant (J (pire) jusqu'à D (meilleur))
- **clarity** : une mesure de la clarté du diamant (I1 (pire), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (meilleur))
- **x** : longueur en mm
- **y** : largeur en mm
- **z** : hauteur en mm
- **depth** :  $z / \text{mean}(x, y) = 2 * z / (x + y)$
- **table** : largeur du dessus du diamant par rapport à son point le plus large

```
diamonds
## # A tibble: 53,940 x 10
##   carat cut      color clarity depth table price     x     y     z
##   <dbl> <ord>    <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.230 Ideal    E     SI2     61.5   55.   326   3.95   3.98   2.43
## 2 0.210 Premium  E     SI1     59.8   61.   326   3.89   3.84   2.31
## 3 0.230 Good     E     VS1     56.9   65.   327   4.05   4.07   2.31
## 4 0.290 Premium  I     VS2     62.4   58.   334   4.20   4.23   2.63
## 5 0.310 Good     J     SI2     63.3   58.   335   4.34   4.35   2.75
## 6 0.240 Very Good J     VVS2     62.8   57.   336   3.94   3.96   2.48
## # ... with 5.393e+04 more rows
```

### 2.4.5 Comment explorer des “tibbles”

Voici les façons les plus communes de comprendre les données se trouvant à l'intérieur d'un “tibble”:

1. En utilisant la fonction ``View()`` de RStudio. C'est la commande que nous utiliserons le plus fréquemment.
2. En utilisant la fonction ``glimpse()`` du paquetage `knitr`
3. En utilisant la fonction ``kable()``
4. En utilisant l'opérateur ``$`` pour étudier une seule variable d'une base de données

#### 1. View():

Exécutez `View(flights)` dans la console de RStudio et explorez la base de données obtenue.

Nous remarquons que chaque colonne représente une variable différente et que ces variables peuvent être de différents types. Certaines de ces variables, comme `distance`, `day` et `arr_delay` sont des variables dites quantitatives. Ces variables sont numériques par nature. D'autres variables sont dites qualitatives.

Si vous regardez la colonne à l'extrême-gauche de la sortie de `View(flights)`, vous verrez une colonne de nombres. Ces nombres représentent les numéros de ligne de la base de données. Si vous vous promenez sur une ligne de même nombre, par exemple la ligne 5, vous étudiez une unité statistique.

#### 2. glimpse:

La seconde façon d'explorer une base de données est d'utiliser la fonction `glimpse()`. Cette fonction nous donne la majorité de l'information précédente et encore plus.

```
glimpse(flights)
## Observations: 336,776
## Variables: 19
## $ year      <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013,...
```

```
## $ month      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ day        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ dep_time   <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 55...
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 60...
## $ dep_delay  <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2...
## $ arr_time   <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 7...
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 7...
## $ arr_delay  <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -...
## $ carrier    <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", ...
## $ flight     <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79...
## $ tailnum    <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN...
## $ origin     <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR"...
## $ dest       <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL"...
## $ air_time   <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138...
## $ distance   <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 94...
## $ hour       <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 5, ...
## $ minute     <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ time_hour  <dtm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013...
```

### 3. kable():

La dernière façon d'étudier l'entièreté de la base de données est d'utiliser la fonction `kable()` de la librairie `knitr`. Nous allons explorer les codes des différentes compagnies d'aviation de deux façons.

```
library(knitr)
airlines
## # A tibble: 16 x 2
##   carrier name
##   <chr>   <chr>
## 1 9E      Endeavor Air Inc.
## 2 AA      American Airlines Inc.
## 3 AS      Alaska Airlines Inc.
## 4 B6      JetBlue Airways
## 5 DL      Delta Air Lines Inc.
## 6 EV      ExpressJet Airlines Inc.
## # ... with 10 more rows
kable(airlines)
```

| carrier | name                        |
|---------|-----------------------------|
| 9E      | Endeavor Air Inc.           |
| AA      | American Airlines Inc.      |
| AS      | Alaska Airlines Inc.        |
| B6      | JetBlue Airways             |
| DL      | Delta Air Lines Inc.        |
| EV      | ExpressJet Airlines Inc.    |
| F9      | Frontier Airlines Inc.      |
| FL      | AirTran Airways Corporation |
| HA      | Hawaiian Airlines Inc.      |
| MQ      | Envoy Air                   |
| OO      | SkyWest Airlines Inc.       |
| UA      | United Air Lines Inc.       |
| US      | US Airways Inc.             |
| VX      | Virgin America              |
| WN      | Southwest Airlines Co.      |
| YV      | Mesa Airlines Inc.          |



À première vue, les deux sorties sont semblables sauf que la seconde est beaucoup plus agréable visuellement dans un document R Markdown.

#### 4. L'opérateur \$:

Finalement, l'opérateur `$` nous permet d'explorer une seule variable à l'intérieur d'une base de données. Par exemple, si nous désirons étudier la variable `name` de la base de données `airlines`, nous obtenons:

```
airlines$name
## [1] "Endeavor Air Inc."          "American Airlines Inc."
## [3] "Alaska Airlines Inc."      "JetBlue Airways"
## [5] "Delta Air Lines Inc."      "ExpressJet Airlines Inc."
## [7] "Frontier Airlines Inc."    "AirTran Airways Corporation"
## [9] "Hawaiian Airlines Inc."    "Envoy Air"
## [11] "SkyWest Airlines Inc."     "United Air Lines Inc."
## [13] "US Airways Inc."          "Virgin America"
## [15] "Southwest Airlines Co."    "Mesa Airlines Inc."
```



## Chapitre 3

# L'extension `questionr`

L'extension `questionr` propose une interface graphique pour faciliter l'opération qui consiste à réordonner vos données.

### 3.1 Mise en place

Pour installer l'extension, vous effectuez la commande suivante:

```
install.packages("questionr")
```

Vous pouvez ensuite la charger.

```
library(questionr)
```

### 3.2 L'interface graphique

L'objectif est de permettre à l'utilisateur de saisir les nouvelles valeurs dans un formulaire, et de générer ensuite le code R correspondant au recodage indiqué.

Pour utiliser cette interface, sous RStudio vous pouvez aller dans le menu *Addins* (présent dans la barre d'outils principale) puis choisir *Levels recoding*.

Si nous utilisons l'interface graphique pour la variable `rincome` de la base de données `gts_cat`, nous obtenons:

L'interface se compose de trois onglets : l'onglet *Variable et paramètres* vous permet de sélectionner la variable à recoder, le nom de la nouvelle variable et d'autres paramètres, l'onglet *Recodages* vous permet de saisir les nouvelles valeurs des modalités, et l'onglet *Code et résultat* affiche le code R correspondant ainsi qu'un tableau permettant de vérifier les résultats.

Une fois votre recodage terminé, cliquez sur le bouton *Done* et le code R sera inséré dans votre script R ou affiché dans la console.

*Important:* cette interface est prévue pour ne pas modifier vos données. C'est donc à vous d'exécuter le code généré pour que le recodage soit réellement effectif.

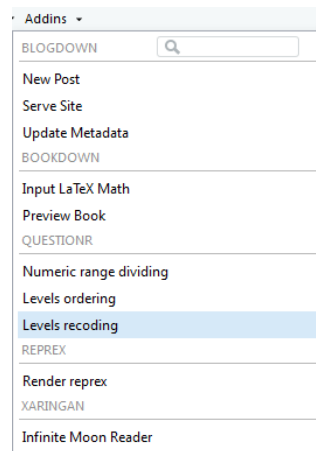


Figure 3.1: Levels recoding dans le menu Addins

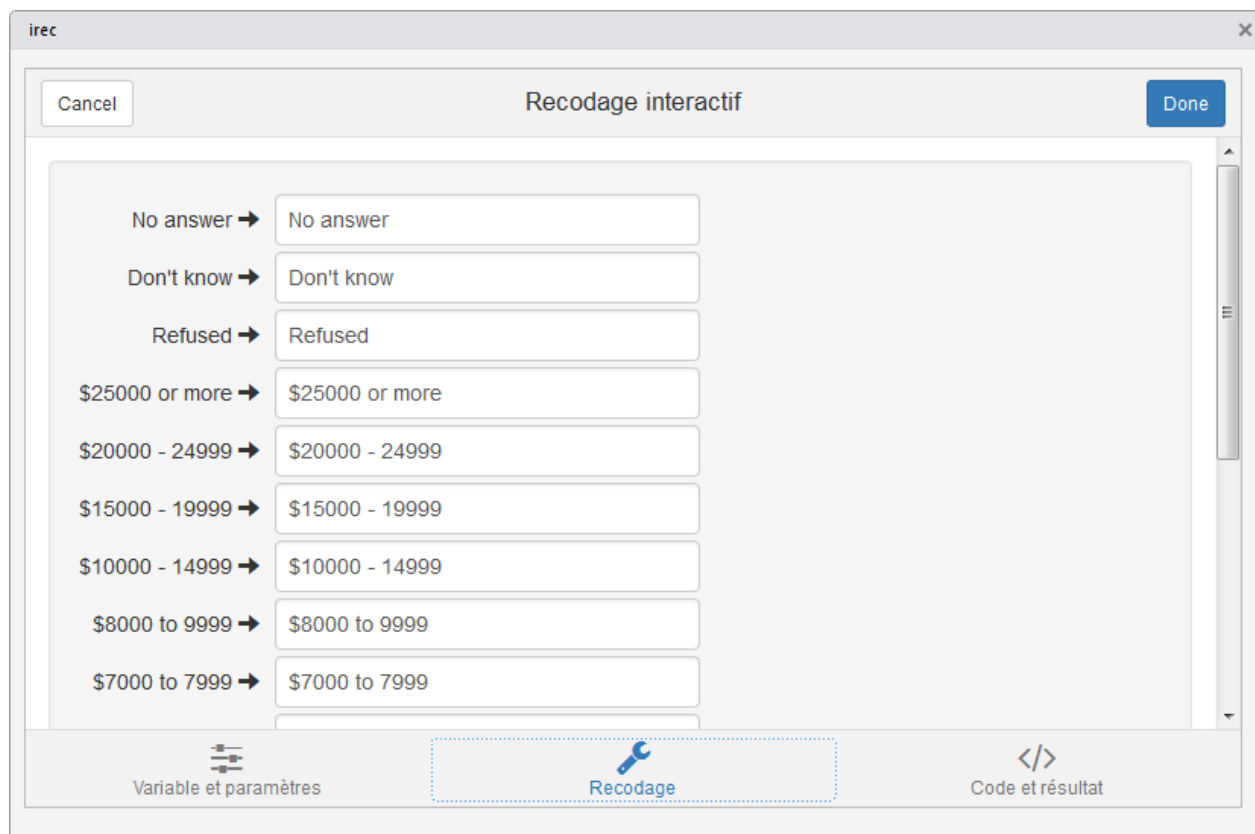


Figure 3.2: L'interface graphique de Levels recoding

## Partie II

# Introduction



## Chapitre 4

# La démarche scientifique

La démarche scientifique permet d'étudier une problématique de sorte que les résultats obtenus soient valides et reproductibles.

La démarche scientifique se divise en quatre étapes :

1. Formulation d'un sujet de recherche et méthodologie
  - Recherche documentaire
  - Formulation d'hypothèses
  - Détermination de la population et/ou de l'échantillon
2. Collecte des données
3. Traitement des données et analyse des résultats
  - Présentation des données
  - Calculs de mesures
  - Inférence statistique
4. Diffusion des résultats

Le cours de méthodes quantitatives porte entre autres sur les trois premières étapes.

Une fois le sujet de recherche choisi, il faut décider comment la recherche sera menée, c'est-à-dire établir la méthodologie. Pour ce faire, il faut d'abord déterminer s'il est préférable d'effectuer un **recensement** (étude portant sur l'ensemble de la population) ou un **sondage** (étude portant sur une partie de la population).

Il existe des avantages et des inconvénients à faire un recensement.

|               |  |
|---------------|--|
| :-----        | :-----   |
| Avantages     | Les résultats obtenus correspondent exactement à la réalité. Il est aisé d'analyser des sous-ensembles de la population. |
| Inconvénients | Les coûts sont élevés. Les résultats sont longs à obtenir.   |

Puisqu'il est généralement souhaitable que la récolte d'information soit faite à coûts moindres et le plus rapidement possible, le sondage est souvent utilisé. Ensuite, le chercheur doit définir les éléments suivants :

|                     |   |               |   |
|---------------------|---|---------------|---|
| :-----              | :-----  | La population | Ensemble des personnes, faits ou objets sur lesquels porte l'étude. Sa taille est notée $N$ |
| L'unité statistique | Une personne, fait ou objet de la population. | L'échantillon | Sous-ensemble de la population utilisé lors d'un sondage. Sa taille est notée $n$ .         |

Dans le choix de la méthodologie, il est aussi important de déterminer les variables étudiées, le type des variables, les échelles de mesure, les types de questions et les techniques d'échantillonnage. Ceux-ci seront étudiés dans les chapitres suivants.





## Chapitre 5

# Les différents types de variables

### 5.1 Introduction

Chacune des notions étudiées par le chercheur porte le nom de variable. C’est logique, puisque les données recueillies vont varier d’une unité statistique à une autre. On distingue quatre types de variables séparées en deux grandes catégories : les variables qualitatives et les variables quantitatives.

#### 5.1.1 Mise en place

Dans ce chapitre, nous introduirons les différents types de variables et les façons avec lesquelles nous pouvons les utiliser en langage R. Nous utiliserons la librairie `tidyverse` et en particulier l’extension `forcats` pour travailler avec des variables qualitatives. Puisque l’extension `forcats` fait partie du `tidyverse` de base, nous avons simplement à charger `tidyverse`.

```
library(tidyverse)
```

### 5.2 Les variables qualitatives

Une variable qualitative est une variable dont les résultats possibles sont des **mots**. Les différents **mots** que peuvent prendre une telle variable sont appelées des **modalités**. Il existe deux types de variables qualitatives.

#### 5.2.1 Les variables qualitatives à échelle nominale

On observe ce type de variable lorsqu’il n’y a pas d’ordre croissant naturel dans les **modalités** de la variable. Par exemple, la variable *couleur des cheveux* est à échelle nominale. L’ordre “blonds, bruns, roux, noirs, autre” est un ordre aussi valable que “bruns, noirs, roux, blonds, autre”.

Imaginons que vous vouliez créer une variable qui indique le mois de l’année:

```
x1 <- c("Déc", "Avr", "Jan", "Mar")
```

L’approche précédente pose deux problèmes:

1. Il n’y a que douze mois possibles et rien ne vous empêche de vous tromper dans votre entrée de modalités:

```
x2 <- c("Déc", "Avr", "Jan", "Mar")
```

2. Les modalités ne seront pas affichées dans un ordre logique

```
# La commande "sort" permet de trier les données
sort(x1)
## [1] "Avr" "Déc" "Jan" "Mar"
```

Nous pouvons résoudre ce problème en utilisant un **facteur** (**factor** en R). Pour créer un facteur, vous devez créer en premier lieu une liste avec **toutes les modalités possibles placées dans l'ordre qui vous convient** (**levels** en R):

```
niveaux_mois <- c(
  "Jan", "Fév", "Mar", "Avr", "Mai", "Jun",
  "Jui", "Aoû", "Sep", "Oct", "Nov", "Déc"
)
```

Vous pouvez maintenant créer un facteur:

```
y1 <- factor(x1, levels = niveaux_mois)
y1
## [1] Déc Avr Jan Mar
## Levels: Jan Fév Mar Avr Mai Jun Jui Aoû Sep Oct Nov Déc
sort(y1)
## [1] Jan Mar Avr Déc
## Levels: Jan Fév Mar Avr Mai Jun Jui Aoû Sep Oct Nov Déc
```

Si certaines modalités ne sont pas dans votre liste de levels, elles seront converties en NA:

```
y2 <- factor(x2, levels = niveaux_mois)
y2
## [1] Déc  Avr  <NA> Mar
## Levels: Jan Fév Mar Avr Mai Jun Jui Aoû Sep Oct Nov Déc
```

Si vous n'utilisez pas vos levels, vos modalités seront affichées en ordre alphabétique:

```
factor(x1)
## [1] Déc Avr Jan Mar
## Levels: Avr Déc Jan Mar
```

Le fait qu'il y ait un **ordre** dans les modalités n'est pas suffisant pour dire qu'une variable qualitative n'est pas nominale. Dans l'exemple précédent, bien que les mois de l'année soient toujours énumérés dans un certain ordre, il serait faux de dire que Janvier < Février par exemple.

Nous pourrions créer une variable qui contient la couleur des cheveux sans indiquer de levels. De cette façon, les données seront triées en ordre alphabétique:

```
x3 <- c("blonds", "bruns", "roux", "noirs", "autre")
sort(x3)
## [1] "autre" "blonds" "bruns" "noirs" "roux"
```

Nous allons maintenant utiliser de vraies données provenant du [General Social Survey](#), qui est un sondage produit par une organisation de recherche indépendante NORC à l'Université de Chicago. Le sondage original comporte des milliers de questions, la base de données `forcats::gss_cat` n'en contient que quelques unes. Pour en savoir plus sur la base de données `gss_cat`, consultez [Wickham \(2018\)](#).

```
gss_cat
## # A tibble: 21,483 x 9
##   year marital      age race rincome partyid  relig  denom tvhours
##   <int> <fct>      <int> <fct> <fct>   <fct>   <fct> <fct>   <int>
```

```
## 1 2000 Never married 26 White $8000 t~ Ind,near~ Prote~ South~ 12
## 2 2000 Divorced 48 White $8000 t~ Not str ~ Prote~ Bapti~ NA
## 3 2000 Widowed 67 White Not app~ Independ~ Prote~ No de~ 2
## 4 2000 Never married 39 White Not app~ Ind,near~ Ortho~ Not a~ 4
## 5 2000 Divorced 25 White Not app~ Not str ~ None Not a~ 1
## 6 2000 Married 25 White $20000 ~ Strong d~ Prote~ South~ NA
## # ... with 2.148e+04 more rows
```

Pour visualiser les levels d'une variable facilement, nous pouvons utiliser la fonction `levels` qui retourne tous les levels différents rencontrés pour cette variable. Voici par exemple les levels pour les variables `race` et `marital`

```
levels(gss_cat$race)
## [1] "Other" "Black" "White" "Not applicable"
levels(gss_cat$marital)
## [1] "No answer" "Never married" "Separated" "Divorced"
## [5] "Widowed" "Married"
```

### 5.2.2 Les variables qualitatives à échelle ordinale

On observe ce type de variable lorsqu'il existe un ordre croissant dans les modalités de la variable. Par exemple, la variable *degré de satisfaction* est à échelle ordinale. Il est possible de classer les modalités en ordre décroissant en écrivant : Très satisfait > Satisfait > Insatisfait > Très insatisfait.

Pour créer une variable qualitative à échelle ordinale en R, nous pouvons utiliser la même technique vue à la section 5.2.1. Nous pouvons donc avoir:

```
z <- c("Satisfait", "Très insatisfait", "Insatisfait", "Très insatisfait", "Insatisfait")
niveaux_satisfaction <- c("Très insatisfait", "Insatisfait", "Satisfait", "Très satisfait")
z1 <- factor(z, levels = niveaux_satisfaction)
sort(z1)
## [1] Très insatisfait Très insatisfait Insatisfait Insatisfait
## [5] Satisfait
## Levels: Très insatisfait Insatisfait Satisfait Très satisfait
```

Il est aussi possible d'utiliser des **facteurs ordonnés**. Nous devons utiliser encore la commande `factor` en ajoutant l'option `ordered=TRUE`. Par exemple:

```
z2 <- factor(z, levels = niveaux_satisfaction, ordered = TRUE)
sort(z2)
## [1] Très insatisfait Très insatisfait Insatisfait Insatisfait
## [5] Satisfait
## 4 Levels: Très insatisfait < Insatisfait < ... < Très satisfait
```

Remarquons que dans la liste **Levels**, R ajoute les symboles < pour indiquer que la variable possède un ordre. Il n'est pas nécessaire de travailler avec des facteurs ordonnés.

Nous remarquons que dans la base de données `forcats::gss_cat`, la variable `rincome` représente une variable qualitative à échelle ordinale:

```
levels(gss_cat$rincome)
## [1] "No answer" "Don't know" "Refused" "$25000 or more"
## [5] "$20000 - 24999" "$15000 - 19999" "$10000 - 14999" "$8000 to 9999"
## [9] "$7000 to 7999" "$6000 to 6999" "$5000 to 5999" "$4000 to 4999"
## [13] "$3000 to 3999" "$1000 to 2999" "Lt $1000" "Not applicable"
```

Si nous laissons de côté les modalités *No answer*, *Don't know* et *Refused*, le reste des modalités peut être placé en ordre. En effet, la modalité *\$4000 to 4999* est plus petite que la modalité *\$5000 to 5999* et ainsi de suite.

Bien que les modalités de la variable précédente soient composées de nombres, le fait que nous ayons affaire à des intervalles indique que nous avons en fait une variable qualitative à échelle ordinale.

Les modalités sont placées en ordre décroissant, si nous voulions avoir les modalités en ordre croissant, nous pourrions faire ceci:

```
a <- factor(gss_cat$rincome,
            levels=c("Lt $1000", "$1000 to 2999", "$3000 to 3999", "$4000 to 4999",
                    "$5000 to 5999", "$6000 to 6999", "$7000 to 7999", "$8000 to 9999",
                    "$10000 - 14999", "$15000 - 19999", "$20000 - 24999", "$25000 or more",
                    "No answer", "Refused", "Not applicable", "Don't know"))

levels(a)
## [1] "Lt $1000"      "$1000 to 2999" "$3000 to 3999" "$4000 to 4999"
## [5] "$5000 to 5999" "$6000 to 6999" "$7000 to 7999" "$8000 to 9999"
## [9] "$10000 - 14999" "$15000 - 19999" "$20000 - 24999" "$25000 or more"
## [13] "No answer"     "Refused"       "Not applicable" "Don't know"
```

## 5.3 Les variables quantitatives

Une variable quantitative est une variable dont les résultats possibles sont des **nombres**. Les différents nombres que peuvent prendre une telle variable sont appelées des **valeurs**.

### 5.3.1 Mise en place

Dans cette section, nous utiliserons la librairie `nycflights13` (voir [Wickham \(2017a\)](#)) qui contient cinq bases de données portant sur tous les vols aériens ayant quittés la ville de New-York en 2013.

```
library(nycflights13)
```

Les cinq base de données sont les suivantes:

- `airlines`
- `airports`
- `flights`
- `planes`
- `weather`

Pour en savoir plus sur une base de données particulière, par exemple `airlines` vous pouvez utiliser la commande `?airlines`.

### 5.3.2 Les variables quantitatives discrètes

On observe ce type de variable lorsque les valeurs sont énumérables, c'est-à-dire lorsqu'il n'existe pas de valeur possible entre deux valeurs consécutives. Par exemple, la variable *nombre de cours suivis pendant cette session* est une variable quantitative discrète. Les valeurs de ces variables peuvent être : 3, 4, 5, 6, 7,... Il est impossible de suivre 4,6 cours durant une session.

La base de données `planes` contient certaines variables quantitatives discrètes.

```
planes
## # A tibble: 3,322 x 9
##   tailnum year type      manufacturer model engines seats speed engine
##   <chr>   <int> <chr>      <chr>          <chr>   <int> <int> <int> <chr>
## 1 N10156   2004 Fixed win~ EMBRAER      EMB-1~       2    55    NA Turbo~
## 2 N102UW   1998 Fixed win~ AIRBUS INDUS~ A320--       2   182    NA Turbo~
## 3 N103US   1999 Fixed win~ AIRBUS INDUS~ A320--       2   182    NA Turbo~
## 4 N104UW   1999 Fixed win~ AIRBUS INDUS~ A320--       2   182    NA Turbo~
## 5 N10575   2002 Fixed win~ EMBRAER      EMB-1~       2    55    NA Turbo~
## 6 N105UW   1999 Fixed win~ AIRBUS INDUS~ A320--       2   182    NA Turbo~
## # ... with 3,316 more rows
```

Pour être en mesure de connaître toutes les *valeurs* différentes que peut prendre une variable, nous allons utiliser la commande `unique`. Si nous nous intéressons à la variable `engines` (qui dénombre le nombre de moteurs de l'avion):

```
unique(planes$engines)
## [1] 2 1 4 3
```

Les avions peuvent donc avoir 1, 2, 3 ou 4 moteurs.

La variable `seats` (qui dénombre le nombre de sièges de l'avion):

```
unique(planes$seats)
## [1] 55 182 149 330 178 95 290 199 20 140 2 8 400 260 255 191 375
## [18] 145 22 14 6 80 189 7 4 377 102 10 11 269 200 222 172 379
## [35] 5 147 100 16 275 292 139 9 450 179 128 300 142 12
```

Dans la sortie R les valeurs ne sont pas en ordre croissant mais elles le seront lorsque nous les représenterons sous forme de tableau ou de graphique.

Bien que la variable `seats` possède plusieurs valeurs (elle en possède 48), cela ne signifie pas qu'elle soit une variable quantitative continue, comme nous le verrons à la section [5.3.3](#).

### 5.3.3 Les variables quantitatives continues

On observe ce type de variable lorsqu'il existe une infinité de valeurs entre deux autres. Par exemple, la variable *masse d'un étudiant (en lbs)* est une variable quantitative continue. Entre 130 et 131 lbs, il existe une infinité de valeurs telles que 130,54 lbs.

Dans la base de données `weather` de l'extension `nycflights13`, nous allons observer la variable `temp`, qui représente la température en degrés Farenheit pour toutes les heures de chaque jour de l'année 2013.

```
weather
## # A tibble: 26,130 x 15
##   origin year month day hour temp dewp humid wind_dir wind_speed
##   <chr>   <dbl> <dbl> <int> <int> <dbl> <dbl> <dbl>   <dbl>      <dbl>
## 1 EWR    2013.   1.    1    0  37.0  21.9  54.0    230.     10.4
## 2 EWR    2013.   1.    1    1  37.0  21.9  54.0    230.     13.8
## 3 EWR    2013.   1.    1    2  37.9  21.9  52.1    230.     12.7
## 4 EWR    2013.   1.    1    3  37.9  23.0  54.5    230.     13.8
## 5 EWR    2013.   1.    1    4  37.9  24.1  57.0    240.     15.0
## 6 EWR    2013.   1.    1    6  39.0  26.1  59.4    270.     10.4
## # ... with 2.612e+04 more rows, and 5 more variables: wind_gust <dbl>,
## #   precip <dbl>, pressure <dbl>, visib <dbl>, time_hour <dtm>
```

Si nous utilisons la commande `unique` sur cette variable, nous obtenons la sortie suivante:

```
unique(weather$temp)
##      [1]  37.0  37.9  39.0  39.9  41.0  39.2  36.0  34.0  33.1  32.0  30.0
##      [12]  28.9  28.0  27.0  26.1  25.0  24.1  30.9  35.1  42.1  43.0  44.1
##      [23]  33.8  35.6  37.4  46.0  46.9  48.0  45.0  48.9  50.0  46.4  44.6
##      [34]  42.8  48.2  51.1  51.8  52.0  55.9  57.9  57.0  55.0  53.1  54.0
##      [45]  23.0  21.0  19.9  19.4  21.9  19.0  18.0  17.1  16.0  15.1  14.0
##      [56]  12.9  12.0  10.9  21.2  17.6  30.2  64.0  64.4  59.0  57.2  60.8
##      [67]  62.1  62.6  24.8  26.6  28.4  41.9  45.7  41.2  38.8  34.2  34.9
##      [78]  32.4  36.5  53.6  60.1  63.0  64.9  66.0  70.0  71.1  68.0  66.9
##      [89]  73.9  77.0  80.1  82.0  82.9  84.0  81.0  79.0  73.0  69.1  61.0
##     [100]  55.4  72.0  66.2  73.4  75.2  75.9  78.1  69.8  75.0  86.0  88.0
##     [111]  87.1  84.9  78.8  80.6  82.4  71.6  89.1  91.0  91.9  93.0  91.4
##     [122]  90.0  93.9  89.6  95.0  87.8  84.2  97.0  96.1  98.1 100.0  99.0
##     [133]  93.2    NA  50.7  47.1  50.5  49.1  47.3  51.3  45.3  13.1  82.6
##     [144]  84.7  83.7  81.1  54.5  53.4  50.2  48.4  43.2  40.6  42.3  43.9
##     [155]  47.8  52.5  55.6  54.1  54.9  50.9  51.4  49.5  46.6  47.5  49.8
##     [166]  50.4  52.7  56.5  58.1  57.6  51.6  60.4  60.3  59.2  55.8
```

Puisque nous avons 175 températures différentes et que nous avons affaire à une variable quantitative continue, il est souvent avantageux de placer ces données dans des classes. Nous verrons comment faire au chapitre [11](#).

## Chapitre 6

# Construire un questionnaire

La deuxième étape de la démarche scientifique est la récolte des données. La façon la plus simple de faire est sans nul doute l'utilisation du questionnaire.

Lors de la conception d'un questionnaire, il est important de choisir les questions de façon à obtenir l'information recherchée.

### 6.1 Critères à respecter

Pour construire un questionnaire qui est fiable, il existe des règles à suivre.

Les questions doivent posséder les qualités ci-dessous :

1. Claire
2. Complète
3. Neutre
4. Non-menaçante
5. Pertinente

### 6.2 Types de questions

Deux questions portant sur un même sujet ne donnent pas nécessairement la même information. C'est pourquoi il est extrêmement important de bien choisir les questions lorsqu'un questionnaire est construit.

Pour plus d'information à ce sujet, consultez le chapitre 7 portant sur les échelles de mesure.

#### 6.2.1 Question ouverte

Ce type de question ne limite pas les réponses. On l'utilise principalement pour récolter l'opinion des gens.

#### 6.2.2 Questions fermées

Ces questions restreignent les choix des répondants.

- Réponse brève : la réponse est une lettre, un chiffre ou un mot
- Dichotomique : il n'y a que deux choix de réponses

- Choix multiples : le répondant doit sélectionner une réponse parmi celles proposées
- Cafétéria : plusieurs réponses parmi celles proposées peuvent être choisies
- Nature hiérarchique : des éléments doivent être placés en ordre d'importance



## Chapitre 7

# Les échelles de mesure



## Chapitre 8

# Les techniques d'échantillonnage

Lorsqu'on souhaite effectuer un sondage plutôt qu'un recensement, il existe diverses façons pour déterminer quelles seront les  $n$  unités statistiques qui feront partie de l'échantillon. Celles-ci sont appelées techniques d'échantillonnage.

### 8.1 Techniques d'échantillonnage aléatoires

Ces techniques permettent de choisir  $n$  unités statistiques au hasard parmi la population.

En choisissant une de ces techniques, il est possible de tirer des conclusions sur une population à partir des résultats d'un sondage puisque la marge d'erreur peut être calculée.

Ces techniques nécessitent par contre une base de sondage, c'est-à-dire une liste de toutes les unités statistiques.

Il existe 4 techniques d'échantillonnage aléatoires.

- Simple: Les individus sont choisis aléatoirement parmi toutes les unités statistiques.
- Stratifié: La population est divisée en sous-ensembles ayant des caractéristiques communes (strates), puis des unités statistiques sont choisies parmi chacune des strates de façon aléatoire en respectant les proportions de la population.
- Par grappes: La population est divisée en sous-ensembles préalablement existants (grappes). Des grappes sont sélectionnées aléatoirement et toutes les unités statistiques de ces grappes sont choisies.
- Systématique: Les unités statistiques sont choisies à intervalles réguliers. Le pas de sondage, l'intervalle auquel sont choisies les unités statistiques, est  $p = \frac{N}{n}$ . La première unité statistique est choisie aléatoirement parmi les  $p$  premières unités statistiques.

### 8.2 Techniques d'échantillonnage non-aléatoires

Ces techniques d'échantillonnage ne relèvent pas du hasard, mais elles sont souvent utilisées puisqu'elles ne nécessitent pas de base de sondage et sont par le fait même généralement plus faciles à mettre en application.

- Accidentel/À l'aveuglette: Les unités statistiques sont choisies parce qu'elles sont présentes lorsque le sondage est effectué.
- Volontaire: Les unités statistiques décident elles-mêmes de participer au sondage.
- Par quotas: La population est divisée en sous-ensembles ayant des caractéristiques communes, puis des unités statistiques sont choisies parmi chacun des sous-ensembles de façon accidentelle en respectant les proportions de la population.

- Au jugé: Le chercheur choisit des unités statistiques précises parce qu'il croit qu'elles sont représentatives de la population.

## 8.3 Base de données pour les *M&M's*

Voici les librairies utilisées.

```
library(googleheets)
library(tidyverse)
library(questionr)
library(knitr)
```

### 8.3.1 Mise en place

Pour débiter, il faut remplir la feuille *Google Sheets* qui contiendra les données. Elle se trouve à l'adresse suivante [MetM](#).

### 8.3.2 Chargement de la feuille Google

Pour charger la feuille Google, nous utiliserons l'extension `googleheets`.

```
goo <- gs_url("https://docs.google.com/spreadsheets/d/16-kbE75LlCnj2GL9h6Ku9r6GnXbUfRGGxkq_vjiSoyc/edit")
MetM <- gs_read(goo)
```

Nous pouvons visualiser les données:

```
MetM
## # A tibble: 12 x 7
##   Nom      Bleu Orange Vert Jaune Brun Rouge
##   <chr>    <int> <int> <int> <int> <int> <int>
## 1 Marc-André    6     5     7     8     5     8
## 2 Mélanie      5     5     8     8     6     5
## 3 Anna        6     6     6     5     4     4
## 4 Laurent     5     5     7     5     6     5
## 5 Carolyne    7     4     4     7     6     4
## 6 Pascal      8     7     6     5     5     5
## # ... with 6 more rows
```

Pour être en mesure de travailler avec ses données, nous allons les transformer en *tidy data* avec la fonction `elongate` chargée dans cette feuille.

```
## # A tibble: 413 x 2
##   Nom      Couleur
##   <chr>    <fct>
## 1 Mélanie  Jaune
## 2 Justin  Orange
## 3 Justin  Brun
## 4 Pascal  Bleu
## 5 Laurent Vert
## 6 Marc-André Orange
## # ... with 407 more rows
```

### 8.3.3 Les M&M's

Voici les proportions des couleurs dans les paquets de M&M's en date de 2008.

| Bleu | Orange | Vert | Jaune | Brun | Rouge |
|------|--------|------|-------|------|-------|
| 24%  | 20%    | 16%  | 14%   | 13%  | 13%   |

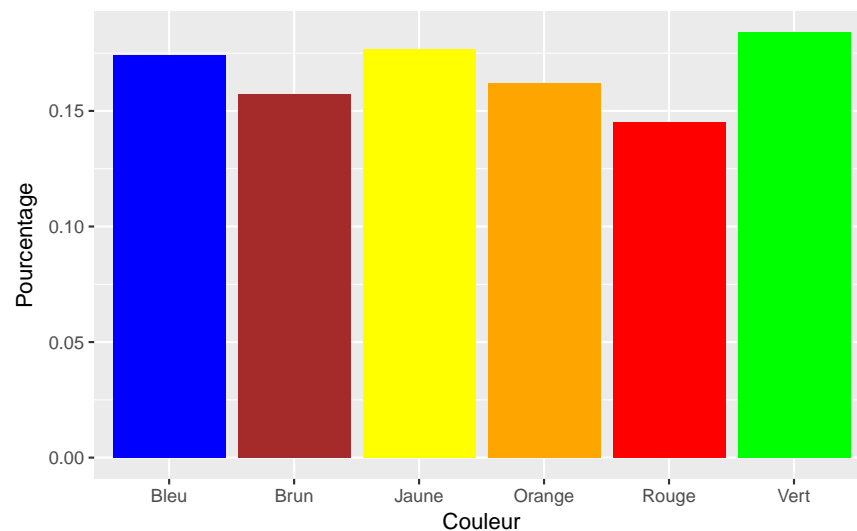
Débutons par résumer toute l'information obtenue.

Voici la distribution des M&M's par couleur.

```
kable(freq(MetMtidy$Couleur,
           total = TRUE,
           valid = FALSE))
```

|        | n   | %     |
|--------|-----|-------|
| Bleu   | 72  | 17.4  |
| Brun   | 65  | 15.7  |
| Jaune  | 73  | 17.7  |
| Orange | 67  | 16.2  |
| Rouge  | 60  | 14.5  |
| Vert   | 76  | 18.4  |
| Total  | 413 | 100.0 |

```
ggplot(MetMtidy, aes(x = Couleur, fill = Couleur))+
  geom_bar(aes(y = (..count..)/sum(..count..)))+
  scale_fill_manual(guide=FALSE,
                    values = c("Bleu"="blue",
                               "Brun"="brown",
                               "Jaune"="yellow",
                               "Orange"="orange",
                               "Rouge"="red",
                               "Vert"="green"))+
  labs(y = "Pourcentage")
```

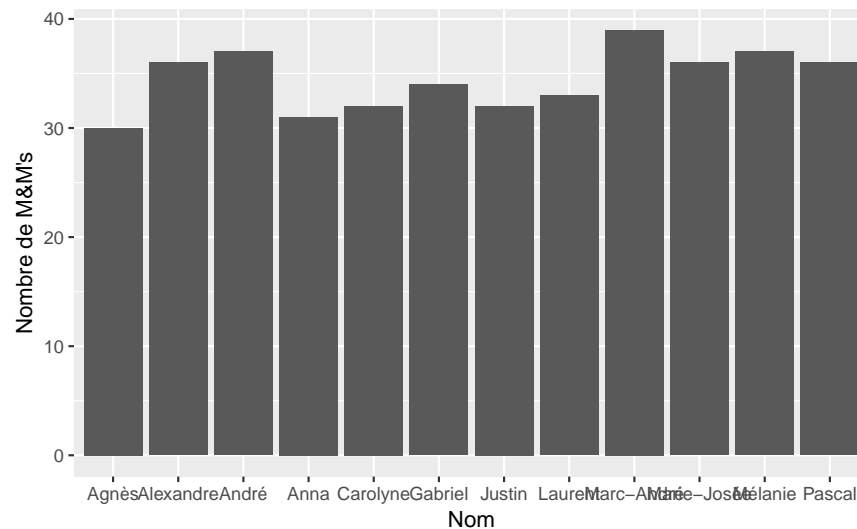


Voici la distribution du nombre de M&M's par paquet.

```
kable(freq(MetMtidy$Nom,
           total = TRUE,
           valid = FALSE))
```

|             | n   | %     |
|-------------|-----|-------|
| Agnès       | 30  | 7.3   |
| Alexandre   | 36  | 8.7   |
| André       | 37  | 9.0   |
| Anna        | 31  | 7.5   |
| Carolynne   | 32  | 7.7   |
| Gabriel     | 34  | 8.2   |
| Justin      | 32  | 7.7   |
| Laurent     | 33  | 8.0   |
| Marc-André  | 39  | 9.4   |
| Marie-Josée | 36  | 8.7   |
| Mélanie     | 37  | 9.0   |
| Pascal      | 36  | 8.7   |
| Total       | 413 | 100.0 |

```
ggplot(MetMtidy, aes(x = Nom)) +
  geom_bar() +
  labs(y = "Nombre de M&M's")
```



Nous pouvons calculer le nombre moyen de *M&M's* par paquet.

```
MetMtidy %>%
  count(Nom) %>%
  summarise(avg = mean(n))
## # A tibble: 1 x 1
##   avg
##   <dbl>
## 1  34.4
```

Voici la distribution du nombre de *M&M's* de chaque couleur séparée par paquet.

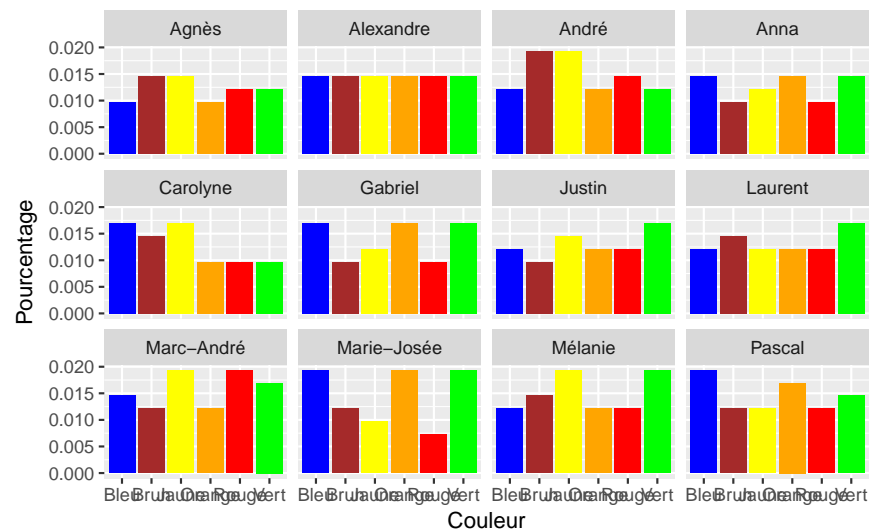
```
ggplot(MetMtidy, aes(x = Couleur, fill = Couleur)) +
  geom_bar(aes(y = (..count..)/sum(..count..))) +
  scale_fill_manual(guide=FALSE,
```

```

values = c("Bleu"="blue",
           "Brun"="brown",
           "Jaune"="yellow",
           "Orange"="orange",
           "Rouge"="red",
           "Vert"="green"))+

facet_wrap(~Nom)+
labs(y = "Pourcentage")

```



### 8.3.4 Simple

### 8.3.5 Stratifié

### 8.3.6 Par grappes

Pour utiliser des grappes, nous pouvons choisir certains paquets et observer tous les *M&M's* dans ces paquets.

Création de la grappe.

```

choix <- c("Anna", "Laurent", "Gabriel")
grappe <- MetMtidy %>%
  filter(Nom %in% choix)

```

Voici la distribution de *M&M's* par couleur.

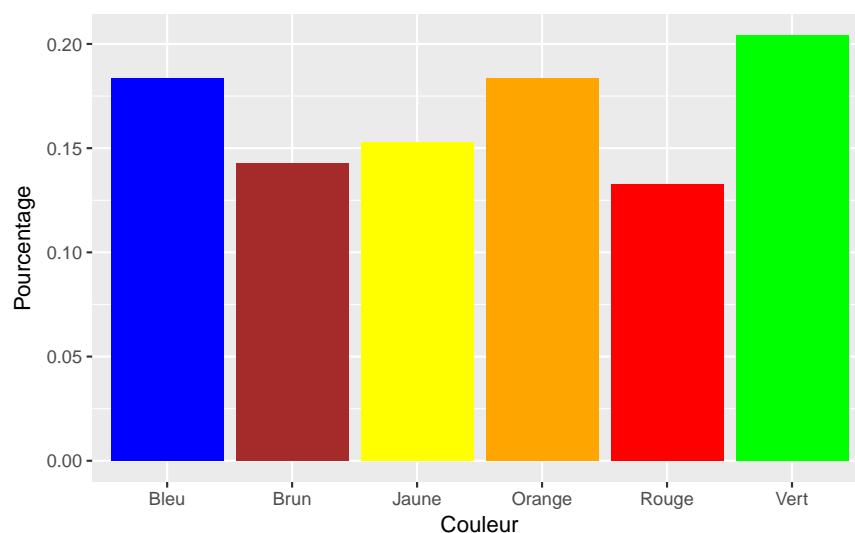
```

kable(freq(grappe$Couleur,
           total = TRUE,
           valid = FALSE))

```

|        | n  | %     |
|--------|----|-------|
| Bleu   | 18 | 18.4  |
| Brun   | 14 | 14.3  |
| Jaune  | 15 | 15.3  |
| Orange | 18 | 18.4  |
| Rouge  | 13 | 13.3  |
| Vert   | 20 | 20.4  |
| Total  | 98 | 100.0 |

```
ggplot(grappe, aes(x = Couleur, fill = Couleur))+
  geom_bar(aes(y = (..count..)/sum(..count..)))+
  scale_fill_manual(guide=FALSE,
                    values = c("Bleu"="blue",
                               "Brun"="brown",
                               "Jaune"="yellow",
                               "Orange"="orange",
                               "Rouge"="red",
                               "Vert"="green"))+
  labs(y = "Pourcentage")
```



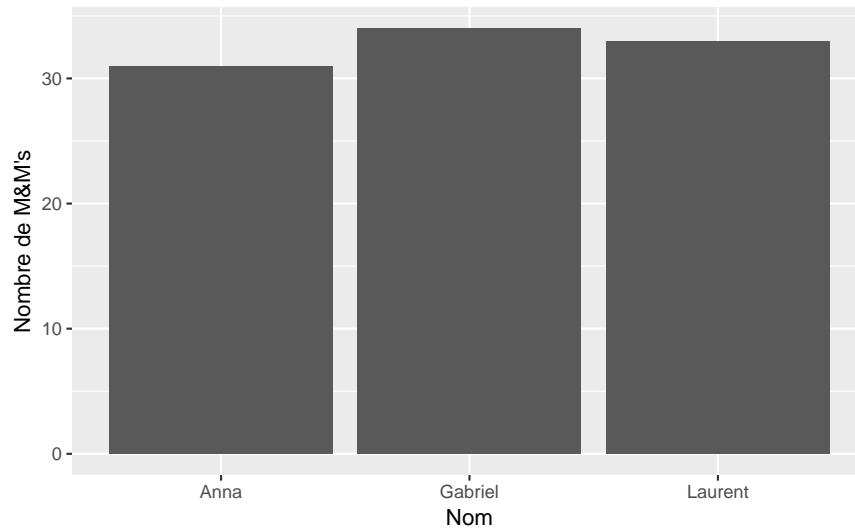
Voici la distribution du nombre de *M&M's* par paquet.

```
kable(freq(grappe$Nom,
           total = TRUE,
           valid = FALSE))
```

|         | n  | %     |
|---------|----|-------|
| Anna    | 31 | 31.6  |
| Gabriel | 34 | 34.7  |
| Laurent | 33 | 33.7  |
| Total   | 98 | 100.0 |

```
ggplot(grappe, aes(x = Nom))+
  geom_bar()+
  labs(y = "Nombre de M&M's")
```





### 8.3.7 Systématique

Pour cette méthode, nous choisissons une unité statistique au hasard, une taille d'échantillon et un pas de sondage.

```
n <- 15 # La taille de l'échantillon

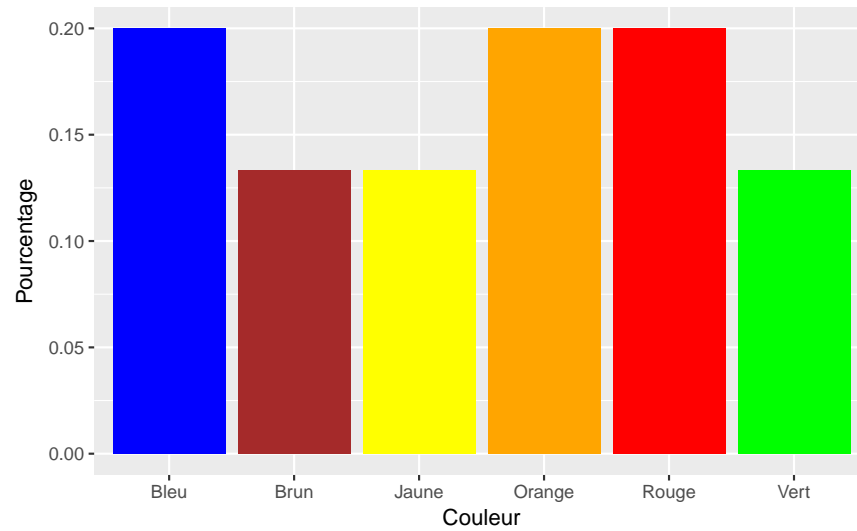
N <- nrow(MetMtidy) # La taille de la population
p <- floor(N/n) # Le pas de sondage
init <- sample(1:p, 1) # Unité statistique initiale
systematique <- MetMtidy[seq(from = init, to = N, by = p),]
```

Voici la distribution de M&M's par couleur.

```
kable(freq(systematique$Couleur,
  total = TRUE,
  valid = FALSE))
```

|        | n  | %     |
|--------|----|-------|
| Bleu   | 3  | 20.0  |
| Brun   | 2  | 13.3  |
| Jaune  | 2  | 13.3  |
| Orange | 3  | 20.0  |
| Rouge  | 3  | 20.0  |
| Vert   | 2  | 13.3  |
| Total  | 15 | 100.0 |

```
ggplot(systematique, aes(x = Couleur, fill = Couleur))+
  geom_bar(aes(y = (..count..)/sum(..count..)))+
  scale_fill_manual(guide=FALSE,
    values = c("Bleu"="blue",
      "Brun"="brown",
      "Jaune"="yellow",
      "Orange"="orange",
      "Rouge"="red",
      "Vert"="green"))+
  labs(y = "Pourcentage")
```

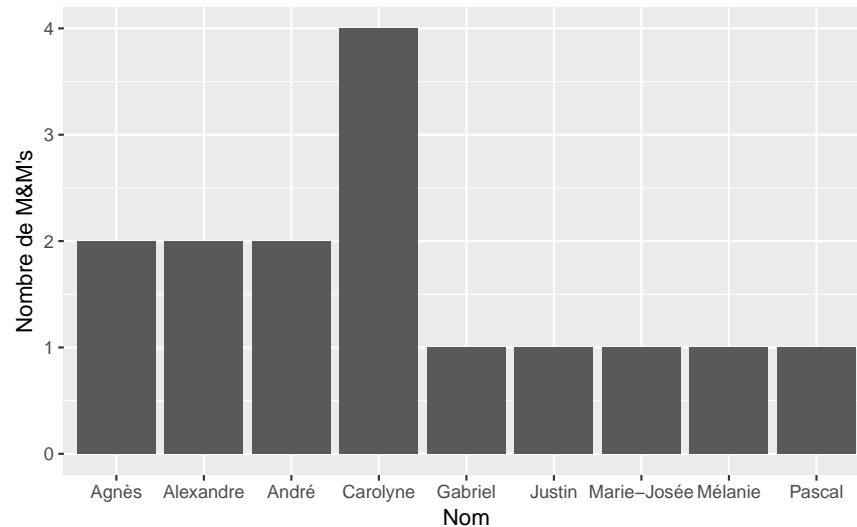


Voici la distribution du nombre de *M&M's* par paquet.

```
kable(freq(systematique$Nom,
           total = TRUE,
           valid = FALSE))
```

|             | n  | %     |
|-------------|----|-------|
| Agnès       | 2  | 13.3  |
| Alexandre   | 2  | 13.3  |
| André       | 2  | 13.3  |
| Carolynne   | 4  | 26.7  |
| Gabriel     | 1  | 6.7   |
| Justin      | 1  | 6.7   |
| Marie-Josée | 1  | 6.7   |
| Mélanie     | 1  | 6.7   |
| Pascal      | 1  | 6.7   |
| Total       | 15 | 100.0 |

```
ggplot(systematique, aes(x = Nom)) +
  geom_bar() +
  labs(y = "Nombre de M&M's")
```



## TODO

1. J'aimerais introduire les méthodes d'échantillonnage en utilisant les *M&M's*
2. Chaque étudiant reçoit une boîte
3. Les étudiants comptent le nombre de bonbons de chaque couleur
4. Je rentre l'information dans une tibble avec le nom de l'étudiant (ça permet de faire une grappe)
5. Les couleurs permettent de faire des strates

Comment construire facilement la tibble? Créer une fonction du genre

m-m(nom, qté rouge, qté bleu, ...)

```
library(tidyverse)
library(questionr)

set.seed(12345)

n <- 24 #On rentre 24 données par personnes
prenoms <- c("Marc-André", "Mélanie", "Anna", "Laurent")
couleurs <- c("Bleu", "Brun", "Orange", "Vert", "Jaune", "Rouge") #Les couleurs possibles

metm <- tibble(nom = character(), couleur = character())

for (i in prenoms){
  temp <- tibble(
    nom = rep(i, n),
    couleur = sample(couleurs, n, replace = TRUE)
  )
  metm <- rbind(metm, temp)
}

freq(metm$couleur, total = TRUE, valid = FALSE)
##      n      %
## Bleu  18  18.8
## Brun  13  13.5
## Jaune 20  20.8
## Orange 15  15.6
## Rouge 16  16.7
```

|          |    |       |
|----------|----|-------|
| ## Vert  | 14 | 14.6  |
| ## Total | 96 | 100.0 |

## Partie III

# Présentation des données



## Chapitre 9

# Les variables qualitatives

### 9.1 Mise en place

```
library(tidyverse)
library(questionr)
library(knitr)
```

### 9.2 Tableau de fréquences

Une fois les données d'un sondage recueillies, il est plus aisé d'analyser ces données si elles sont classées dans un tableau.

Le tableau de fréquences que nous utiliserons est le suivant:

| Titre              |                              |                                   |
|--------------------|------------------------------|-----------------------------------|
| Nom de la variable | Nombre d'unités statistiques | Pourcentage d'unités statistiques |
| (Modalités)        | (Fréquences absolues)        | (%)                               |
| Total              | $n$                          | 100%                              |

Important : Le titre doit toujours être indiqué lors de la construction d'un tableau de fréquence.

Lorsque les données se trouvent dans une `tibble` dans R, il est possible d'utiliser la commande `freq` de la librairie `questionr` pour afficher le tableau de fréquences. La commande `freq` prend comme argument la variable dont vous voulez produire le tableau de fréquences. Pour obtenir une sortie adéquate, il faut ajouter trois options à la commande:

- `cum = FALSE`; permet de ne pas afficher les pourcentages cumulés
- `valid = FALSE`; permet de ne pas afficher les données manquantes
- `total = TRUE`; permet d'afficher le total

Dans la base de données `forcats::gss_cat`, nous allons afficher la variable `marital`. Dans la commande ci-dessous, nous enregistrons le tableau de fréquences dans la variable `tab_marital`. Nous l'affichons ensuite à l'aide de la commande `kable`.

```
tab_marital <- freq(gss_cat$marital,
                    cum = FALSE,
```

```
valid = FALSE,
total = TRUE)
kable(tab_marital)
```

|               | n     | %     |
|---------------|-------|-------|
| No answer     | 17    | 0.1   |
| Never married | 5416  | 25.2  |
| Separated     | 743   | 3.5   |
| Divorced      | 3383  | 15.7  |
| Widowed       | 1807  | 8.4   |
| Married       | 10117 | 47.1  |
| Total         | 21483 | 100.0 |

À l'aide du tableau précédent, répondez aux questions suivantes:

1. Combien de personnes ne se sont jamais mariées dans l'échantillon? 5416
2. Quel est le pourcentage de personnes divorcées dans l'échantillon? 15.7 %
3. Quel est le nombre total d'unités statistiques? 21483

Nous pouvons produire le tableau de fréquences de la variable `race` de la façon suivante:

```
tab_race <- freq(gss_cat$race,
cum = FALSE,
valid = FALSE,
total = TRUE)
kable(tab_race)
```

|                | n     | %     |
|----------------|-------|-------|
| Other          | 1959  | 9.1   |
| Black          | 3129  | 14.6  |
| White          | 16395 | 76.3  |
| Not applicable | 0     | 0.0   |
| Total          | 21483 | 100.0 |

### 9.3 Représentation graphique - Le diagramme à bandes

Pour représenter graphiquement les variables qualitatives, nous allons utiliser les diagrammes à bandes.

Pour construire ce graphique:

- Chaque modalité est représentée par un rectangle.
- La hauteur de chaque rectangle doit être proportionnelle
  - au nombre d'unités statistiques (la fréquence absolue) OU
  - au pourcentage d'unités statistiques (la fréquence relative).
- Le titre et les fréquences (absolues ou relatives) doivent être indiqués.
- L'axe des x doit posséder un titre : le nom de la variable étudiée.
- L'axe des y doit posséder un titre : "Nombre d'unités statistiques" ou "Pourcentage d'unités statistiques".
- La graduation de l'axe des y doit commencer à zéro (l'axe ne doit pas être coupé).
- Les rectangles doivent être équidistants et de largeur égale. De plus, ils ne doivent pas être collés.

Pour produire le diagramme à bandes, nous utiliserons l'extension `ggplot2` qui est chargée avec le coeur de la librairie `tidyverse`. La grammaire graphique de `ggplot2` peut être décrite de la façon suivante:

**A statistical graphic is a mapping of data variables to aesthetic attributes of geometric objects.**



Plus spécifiquement, nous pouvons briser un graphique en trois composantes essentielles:

1. **data**: la base de données contenant les variables que nous désirons visualiser.
2. **geom**: l'objet géométrique en question. Ceci réfère au type d'objet que nous pouvons observer dans notre graphique. Par exemple, des points, des lignes, des barres, etc.
3. **aes**: les attributs esthétiques (aesthetics) de l'objet géométrique que nous affichons dans notre graphique. Par exemple, la position x/y, la couleur, la forme, la taille. Chaque attribut peut être associé à une variable dans notre base de données.

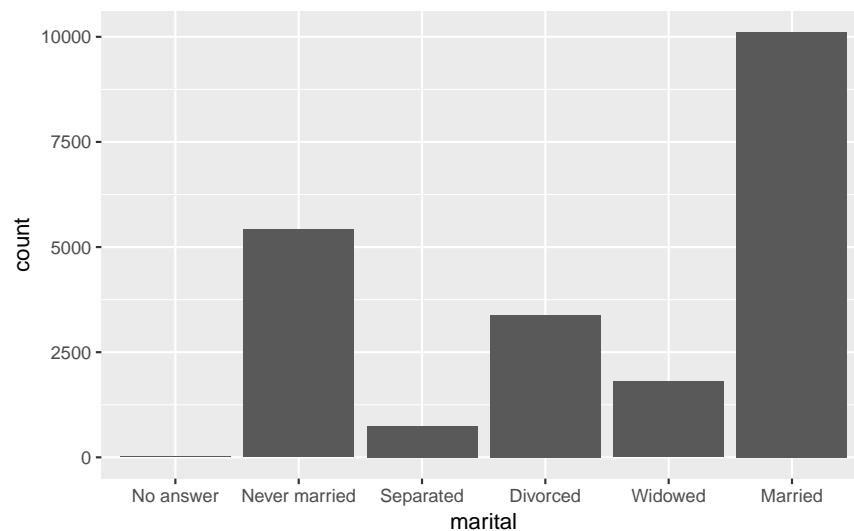
### 9.3.1 La variable `marital`

Nous allons visualiser le diagramme à bandes de la variable `marital` provenant de la base de données `forcats::gss_cat`. Nous devons spécifier:

- `data = gss_cat`: la base de données.
- `aes(x = marital)`: la variable étudiée.
- `geom_bar()`: nous voulons un diagramme à bandes

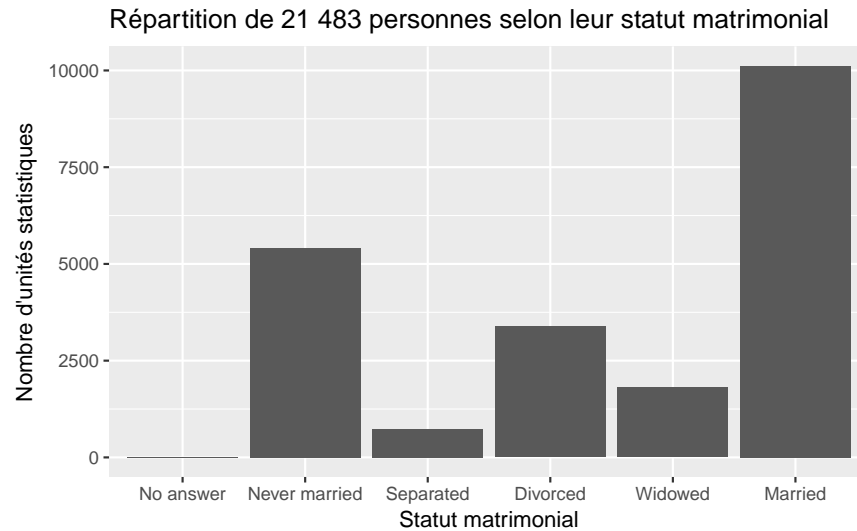
La commande est donc:

```
ggplot(data = gss_cat, aes(x = marital))+  
  geom_bar()
```



Pour ajouter un titre et indiquer les titres des axes x et y, nous utilisons la commande `labs` (pour *labels*).

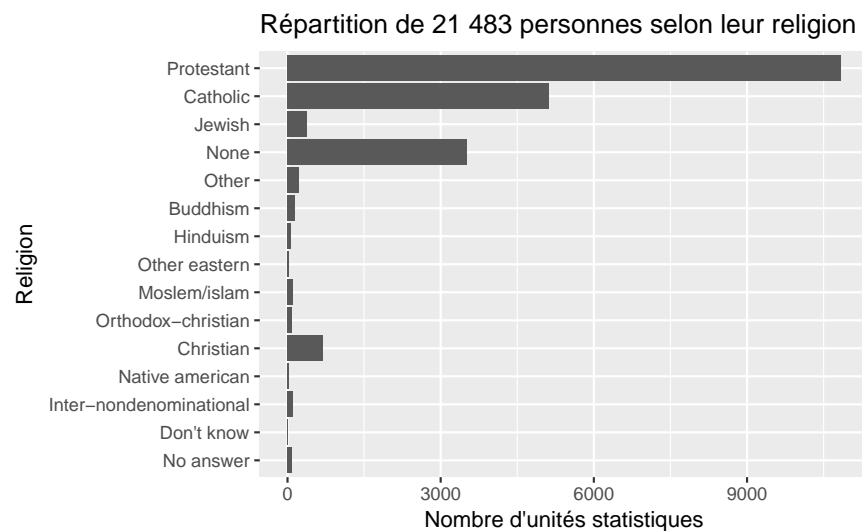
```
ggplot(data = gss_cat, aes(x = marital))+  
  geom_bar()+  
  labs(  
    title = "Répartition de 21 483 personnes selon leur statut matrimonial",  
    x = "Statut matrimonial",  
    y = "Nombre d'unités statistiques"  
  )
```



### 9.3.2 La variable `relig`

Nous pouvons afficher le diagramme à bandes horizontales de la variable `relig` en ajoutant la commande `coord_flip()`. Nous avons donc:

```
ggplot(data = gss_cat, aes(x = relig))+
  geom_bar()+
  labs(
    title = "Répartition de 21 483 personnes selon leur religion",
    x = "Religion",
    y = "Nombre d'unités statistiques"
  )+
  coord_flip()
```



# Chapitre 10

## Les variables quantitatives discrètes

### 10.1 Mise en place

```
library(tidyverse)
library(questionr)
library(nycflights13)
library(knitr)
```

### 10.2 Tableau de fréquences

Une fois les données d'un sondage recueillies, il est plus aisé d'analyser ces données si elles sont classées dans un tableau.

Le tableau de fréquences que nous utiliserons est le suivant :

| Titre                           |   |   |   |
|---------------------------------|---|---|---|
| Nom de la variable<br>(Valeurs) | Nombre d'unités statistiques<br>(Fréquences absolues) | Pourcentage d'unités statistiques (%)<br>(Fréquences relatives) | Pourcentage cumulé<br>(Fréquences relatives cumulées) |
| Total                           | $n$   | 100%  |   |

Le pourcentage cumulé permet de déterminer le pourcentage des répondants qui ont indiqué la valeur correspondante, ou une plus petite. Il sert à donner une meilleure vue d'ensemble.

Si pour la valeur  $x_i$  de la variable  $A$  la pourcentage cumulé est de  $b$  %, ceci signifie que  $b$  % des valeurs de la variable  $A$  sont plus petites ou égales à  $x_i$ .

La commande `freq` prend comme argument la variable dont vous voulez produire le tableau de fréquences. Pour obtenir une sortie adéquate, il faut ajouter trois options à la commande:

- `cum = TRUE`; permet d'afficher les pourcentages cumulés
- `valid = FALSE`; permet de ne pas afficher les données manquantes
- `total = TRUE`; permet d'afficher le total

Dans la base de données `nycflights13::planes`, nous allons afficher la variable `engines`. Dans la commande ci-dessous, nous enregistrons le tableau de fréquences dans la variable `tab_engines`. Nous l'affichons ensuite

à l'aide de la commande `kable`.

```
tab_engines <- freq(planes$engines,
  cum = TRUE,
  valid = FALSE,
  total = TRUE)
kable(tab_engines)
```

|       | n    | %     | %cum  |
|-------|------|-------|-------|
| 1     | 27   | 0.8   | 0.8   |
| 2     | 3288 | 99.0  | 99.8  |
| 3     | 3    | 0.1   | 99.9  |
| 4     | 4    | 0.1   | 100.0 |
| Total | 3322 | 100.0 | 100.0 |

Nous remarquons que le pourcentage cumulé pour les avions possédant 3 moteurs est 99.9%. Quelle est la signification de ce pourcentage? Ceci signifie que 99.9% des avions possèdent 3 moteurs ou moins.

Nous allons maintenant produire le tableau de fréquences de la variable `tvhours` de la base de données `gss_cat`. Cette variable correspond au nombre d'heures de télévision écoutées par jour (pour avoir cette information, vous pouvez utiliser la commande `?forcats::gss_cat`). Nous avons:

```
tab_tvhours <- freq(gss_cat$tvhours,
  cum = TRUE,
  valid = FALSE,
  total = TRUE)
kable(tab_tvhours)
```

|       | n     | %     | %cum  |
|-------|-------|-------|-------|
| 0     | 675   | 3.1   | 3.1   |
| 1     | 2345  | 10.9  | 14.1  |
| 2     | 3040  | 14.2  | 28.2  |
| 3     | 1959  | 9.1   | 37.3  |
| 4     | 1408  | 6.6   | 43.9  |
| 5     | 695   | 3.2   | 47.1  |
| 6     | 478   | 2.2   | 49.3  |
| 7     | 119   | 0.6   | 49.9  |
| 8     | 262   | 1.2   | 51.1  |
| 9     | 19    | 0.1   | 51.2  |
| 10    | 122   | 0.6   | 51.8  |
| 11    | 9     | 0.0   | 51.8  |
| 12    | 96    | 0.4   | 52.3  |
| 13    | 9     | 0.0   | 52.3  |
| 14    | 24    | 0.1   | 52.4  |
| 15    | 17    | 0.1   | 52.5  |
| 16    | 10    | 0.0   | 52.5  |
| 17    | 2     | 0.0   | 52.5  |
| 18    | 7     | 0.0   | 52.6  |
| 20    | 14    | 0.1   | 52.6  |
| 21    | 2     | 0.0   | 52.7  |
| 22    | 2     | 0.0   | 52.7  |
| 23    | 1     | 0.0   | 52.7  |
| 24    | 22    | 0.1   | 52.8  |
| NA    | 10146 | 47.2  | 100.0 |
| Total | 21483 | 100.0 | 100.0 |

Répondez aux questions suivantes:

1. Quel est le pourcentage des répondants qui écoutent la télévision 3 heures par jour? 9.1 %
2. Quel est le pourcentage des répondants qui écoutent la télévision 14 heures par jour? 0.1 %
3. Peut-on croire le résultat pour le pourcentage de gens qui écoutent la télévision 24 heures par jour?
4. Quelle est la signification du pourcentage cumulé pour 6 heures? Nous avons que 49.3 % des répondants écoutent la télévision 6 heures ou moins par jour.
5. Quelle est la signification du pourcentage cumulé pour 7 heures? Nous avons que 49.9 % des répondants écoutent la télévision 7 heures ou moins par jour. C'est-à-dire qu'environ la moitié des gens écoutent la télévision 7 heures ou moins par jour.

Nous pouvons produire le tableau de fréquences de la variable `seats` de la façon suivante:

```
tab_seats <- freq(planes$seats,  
                  cum = TRUE,  
                  valid = FALSE,  
                  total = TRUE)  
kable(tab_seats)
```

|       | n    | %     | %cum  |
|-------|------|-------|-------|
| 2     | 16   | 0.5   | 0.5   |
| 4     | 5    | 0.2   | 0.6   |
| 5     | 2    | 0.1   | 0.7   |
| 6     | 3    | 0.1   | 0.8   |
| 7     | 2    | 0.1   | 0.8   |
| 8     | 5    | 0.2   | 1.0   |
| 9     | 1    | 0.0   | 1.0   |
| 10    | 1    | 0.0   | 1.1   |
| 11    | 2    | 0.1   | 1.1   |
| 12    | 1    | 0.0   | 1.1   |
| 14    | 1    | 0.0   | 1.2   |
| 16    | 1    | 0.0   | 1.2   |
| 20    | 80   | 2.4   | 3.6   |
| 22    | 2    | 0.1   | 3.7   |
| 55    | 390  | 11.7  | 15.4  |
| 80    | 83   | 2.5   | 17.9  |
| 95    | 123  | 3.7   | 21.6  |
| 100   | 102  | 3.1   | 24.7  |
| 102   | 1    | 0.0   | 24.7  |
| 128   | 1    | 0.0   | 24.7  |
| 139   | 8    | 0.2   | 25.0  |
| 140   | 411  | 12.4  | 37.4  |
| 142   | 158  | 4.8   | 42.1  |
| 145   | 57   | 1.7   | 43.8  |
| 147   | 3    | 0.1   | 43.9  |
| 149   | 452  | 13.6  | 57.5  |
| 172   | 81   | 2.4   | 60.0  |
| 178   | 283  | 8.5   | 68.5  |
| 179   | 134  | 4.0   | 72.5  |
| 182   | 159  | 4.8   | 77.3  |
| 189   | 73   | 2.2   | 79.5  |
| 191   | 87   | 2.6   | 82.1  |
| 199   | 43   | 1.3   | 83.4  |
| 200   | 256  | 7.7   | 91.1  |
| 222   | 13   | 0.4   | 91.5  |
| 255   | 16   | 0.5   | 92.0  |
| 260   | 4    | 0.1   | 92.1  |
| 269   | 1    | 0.0   | 92.1  |
| 275   | 25   | 0.8   | 92.9  |
| 290   | 6    | 0.2   | 93.1  |
| 292   | 16   | 0.5   | 93.6  |
| 300   | 17   | 0.5   | 94.1  |
| 330   | 114  | 3.4   | 97.5  |
| 375   | 1    | 0.0   | 97.5  |
| 377   | 14   | 0.4   | 98.0  |
| 379   | 55   | 1.7   | 99.6  |
| 400   | 12   | 0.4   | 100.0 |
| 450   | 1    | 0.0   | 100.0 |
| Total | 3322 | 100.0 | 100.0 |

Comme nous pouvons le constater, le tableau est très grand car la variable `seats` possède 48 valeurs différentes. Nous allons donc parfois séparer nos valeurs en classes coomme nous le verrons au chapitre 11.

Important! Au chapitre 5, nous avons étudié les différents types de variables. Parmi les variables quantitatives, nous avons distingué celles qui étaient discrètes de celles qui étaient continues. Bien que cela s'applique toujours, **il est important de noter qu'une variable continue** (par exemple, l'âge) **peut être traitée comme une variable discrète** (puisque, de façon générale, les gens donnent un entier pour exprimer leur âge); **de même, une variable discrète** (par exemple, le revenu) **peut être traitée comme une variable continue** (puisque les différents revenus sont trop nombreux pour être énumérés).

## 10.3 Représentation graphique - Le diagramme à bandes

Le graphique utilisé pour représenter une variable quantitative discrète est le diagramme à bâtons.

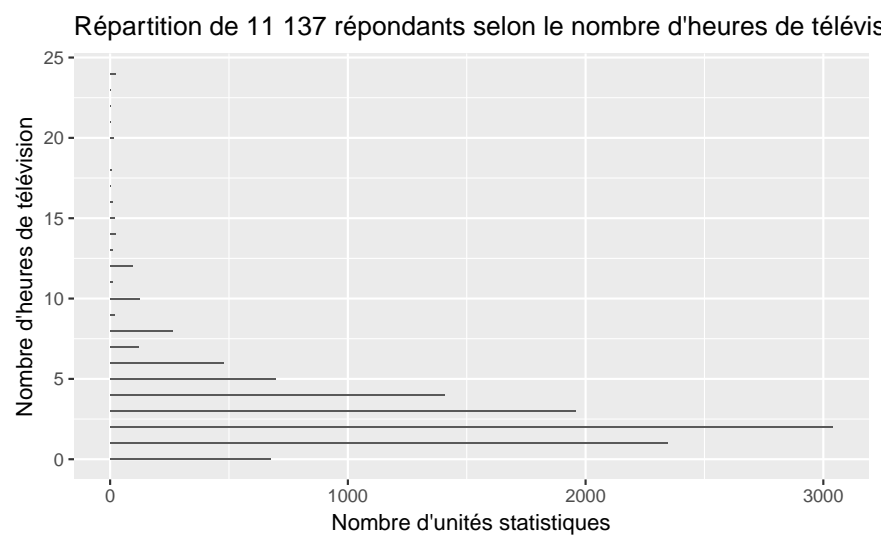
Ce graphique est construit comme le diagramme à bandes rectangulaires verticales, sauf que les rectangles sont remplacés par des bâtons très minces (généralement une simple ligne). Pour modifier la largeur de vos lignes, vous utilisez l'option `width` dans la commande `geom_bar()`.

```
ggplot(data = planes, aes(x = engines)) +
  geom_bar(width = 0.1) +
  labs(
    title = "Répartition de 3 322 avions selon le nombre de leurs moteurs",
    x = "Nombre de moteurs",
    y = "Nombre d'unités statistiques"
  )
```



Vous pouvez produire un diagramme à bâtons horizontaux en utilisant la commande `coord_flip()`.

```
ggplot(data = gss_cat, aes(x = tvhours)) +
  geom_bar(width = 0.1) +
  labs(
    title = "Répartition de 11 137 répondants selon le nombre d'heures de télévision écoutées par jour",
    x = "Nombre d'heures de télévision",
    y = "Nombre d'unités statistiques"
  ) +
  coord_flip()
## Warning: Removed 10146 rows containing non-finite values (stat_count).
```





# Chapitre 11

## Les variables quantitatives continues

### 11.1 Mise en place

```
library(tidyverse)
library(questionr)
library(nycflights13)
library(knitr)
```

### 11.2 Tableau de fréquences

Les différentes valeurs d'une variable continue étant impossibles à énumérer, nous devons regrouper celles-ci en classes. La première colonne sera donc constituée de celles-ci.

Il sera parfois utile d'ajouter une colonne supplémentaire au tableau habituel: le milieu de classe. Celui-ci est calculé en faisant la moyenne entre le début de classe et la fin de classe.

Le tableau de fréquences que nous utiliserons est le suivant :

| Titre                              |  |  |  |
|------------------------------------|--|--|--|
| Nom de la<br>variable<br>(Classes) | Nombre d'unités<br>statistiques<br>(Fréquences absolues) | Pourcentage d'unités<br>statistiques (%)<br>(Fréquences relatives) | Pourcentage cumulé<br>(Fréquences relatives<br>cumulées) |
| Total                              | $n$  | 100%   |  |

Pour être en mesure de briser une variable en classes, il faut utiliser la commande `cut`. Les options utilisées sont les suivantes:

Pour ce faire, nous devons utiliser la commande `cut` qui permet d'indiquer les frontières de ces classes. Voici un exemple où nous créons des classes de largeur 25:

```
temp_classes <- cut(weather$temp,
                     breaks = c(0, 25, 50, 75, 100, 125),
                     include.lowest = TRUE,
                     right = FALSE)
unique(temp_classes)
```

```
## [1] [25,50) [0,25) [50,75) [75,100) [100,125] <NA>
## Levels: [0,25) [25,50) [50,75) [75,100) [100,125]
```

Nous nous retrouvons donc avec 6 classes. Lorsque nous présenterons les variables sous forme de tableau, il nous sera utile d'utiliser la commande `cut`.

L'option `include.lowest` indique que nous voulons conserver ... L'option `right = FALSE` indique que nous voulons des intervalles fermés à gauche et ouverts à droite.

- `include.lowest=TRUE`: permet d'inclure les valeurs extrêmes
- `right=FALSE`: permet d'avoir des classes fermées à gauche et ouvertes à droite
- `breaks=c(0, 100, 200, 300, 400, 500, 600, 700)`: permet de couper les classes à 0, 100, 200, 300, 400, 500, 600 et 700

Pour simplifier le code, nous créons en premier lieu une variable `air_time_rec` avec les classes et nous l'affichons ensuite avec `freq`. Remarquons que nous avons ajouté l'option `valid = TRUE` car certaines valeurs sont manquantes. Rappelons que les données manquantes sont représentées par `NA` en R. Deux colonnes sont ajoutées:

- `val%`: le pourcentage en omettant les valeurs manquantes
- `val%cum`: le pourcentage cumulé en omettant les valeurs manquantes

Nous obtenons donc:

```
air_time_rec <- cut(flights$air_time,
                    include.lowest=TRUE,
                    right=FALSE,
                    breaks=c(0, 100, 200, 300, 400, 500, 600, 700))
tab_airtime <- freq(air_time_rec,
                   cum = TRUE,
                   total = TRUE,
                   valid = TRUE)
kable(tab_airtime)
```

|           | n      | %     | val%  | %cum  | val%cum |
|-----------|--------|-------|-------|-------|---------|
| [0,100)   | 105687 | 31.4  | 32.3  | 31.4  | 32.3    |
| [100,200) | 146527 | 43.5  | 44.8  | 74.9  | 77.0    |
| [200,300) | 31036  | 9.2   | 9.5   | 84.1  | 86.5    |
| [300,400) | 43347  | 12.9  | 13.2  | 97.0  | 99.8    |
| [400,500) | 48     | 0.0   | 0.0   | 97.0  | 99.8    |
| [500,600) | 132    | 0.0   | 0.0   | 97.0  | 99.8    |
| [600,700] | 569    | 0.2   | 0.2   | 97.2  | 100.0   |
| NA        | 9430   | 2.8   | NA    | 100.0 | NA      |
| Total     | 336776 | 100.0 | 100.0 | 100.0 | 100.0   |

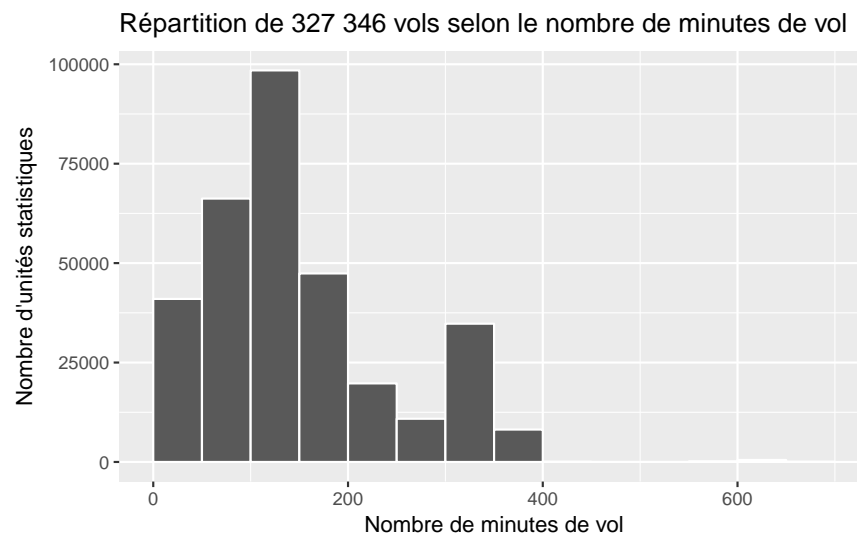
À la section 10.2, nous avons vu que la variable `seats` de la base de données `planes` contenait 48 valeurs différentes. Nous allons donc créer le tableau de fréquences avec des classes.

```
seats_rec <- cut(planes$seats,
                include.lowest=TRUE,
                right=FALSE,
                breaks=c(0, 50, 100, 150, 200, 250, 300, 350, 400, 450))
tab_seats <- freq(seats_rec,
                 cum = TRUE,
                 total = TRUE,
                 valid = TRUE)
kable(tab_seats)
```

|           | n    | %     | val%  | %cum  | val%cum |
|-----------|------|-------|-------|-------|---------|
| [0,50)    | 122  | 3.7   | 3.7   | 3.7   | 3.7     |
| [50,100)  | 596  | 17.9  | 17.9  | 21.6  | 21.6    |
| [100,150) | 1193 | 35.9  | 35.9  | 57.5  | 57.5    |
| [150,200) | 860  | 25.9  | 25.9  | 83.4  | 83.4    |
| [200,250) | 269  | 8.1   | 8.1   | 91.5  | 91.5    |
| [250,300) | 68   | 2.0   | 2.0   | 93.6  | 93.6    |
| [300,350) | 131  | 3.9   | 3.9   | 97.5  | 97.5    |
| [350,400) | 70   | 2.1   | 2.1   | 99.6  | 99.6    |
| [400,450] | 13   | 0.4   | 0.4   | 100.0 | 100.0   |
| Total     | 3322 | 100.0 | 100.0 | 100.0 | 100.0   |

## 11.3 Représentation graphique - L'histogramme

```
ggplot(flights, aes(x = air_time)) +
  geom_histogram(binwidth = 50, center = 25, color = 'white') +
  labs(
    title = "Répartition de 327 346 vols selon le nombre de minutes de vol",
    x = "Nombre de minutes de vol",
    y = "Nombre d'unités statistiques"
  )
## Warning: Removed 9430 rows containing non-finite values (stat_bin).
```





# Chapitre 12

## Les différentes mesures

Dans ce chapitre, nous verrons comment utiliser R pour calculer les mesures importantes permettant de résumer des données.

Nous allons charger les paquets que nous allons utiliser:

```
library(questionr)
library(ggplot2)
library(nycflights13)
```

### 12.1 Les mesures de tendance centrale

Les mesures de tendance centrale permettent de déterminer où se situe le « centre » des données. Les trois mesures de tendance centrale sont le mode, la moyenne et la médiane.

#### 12.1.1 Le mode

Le mode est la **modalité**, **valeur** ou **classe** possédant la plus grande fréquence. En d'autres mots, c'est la donnée la plus fréquente.

Puisque le mode se préoccupe seulement de la donnée la plus fréquente, il n'est pas influencé par les valeurs extrêmes.

Lorsque le mode est une classe, il est appelé **classe modale**.

Le mode est noté **Mo**.

Le langage R ne possède pas de fonction permettant de calculer le mode. La façon la plus simple de le calculer est d'utiliser la fonction `table` de R.

Par exemple, si nous voulons connaître le mode de la variable `cut` de la base de données `diamonds`:

```
table(diamonds$cut)
##
##      Fair      Good Very Good   Premium     Ideal
##      1610     4906     12082     13791     21551
```

Nous remarquons que le maximum est à la modalité *Ideal* avec une fréquence de 21551.

Si nous nous intéressons au mode d'une variable quantitative discrète comme `cyl` de la base de données `mtcars` nous obtenons:

```
table(mtcars$cyl)
##
##  4  6  8
## 11  7 14
```

Nous remarquons que le maximum est à la valeur 8 avec une fréquence de 14.

Dans le cas d'une variable quantitative continue, pour calculer le mode, il faut commencer par séparer les données en classes. Nous utiliserons les mêmes classes utilisées à la section

```
carat_class = cut(diamonds$carat,
                  breaks = seq(from = 0, to = 6, by = 1),
                  right = FALSE)
table(carat_class)
## carat_class
## [0,1) [1,2) [2,3) [3,4) [4,5) [5,6)
## 34880 16906 2114   34     5     1
```

La classe modale est donc la classe  $[0,1)$  avec une fréquence de 34880.

### 12.1.2 La médiane

La médiane, notée **Md**, est la valeur qui sépare une série de données classée en ordre croissant en deux parties égales.

La médiane étant la valeur du milieu, elle est la valeur où le pourcentage cumulé atteint 50%.

Puisque la médiane se préoccupe seulement de déterminer où se situe le centre des données, elle n'est pas influencée par les valeurs extrêmes. Elle est donc une mesure de tendance centrale plus fiable que la moyenne.

Important : La médiane n'est définie que pour les variables quantitatives. En effet, si vous tentez d'utiliser la médiane pour des données autres que numériques, R vous donnera un message d'erreur.

La fonction `median` permet de calculer la médiane en langage R.

Par exemple, pour calculer la médiane de la variable `carat` de la base de données `diamonds`, nous avons:

```
median(diamonds$carat)
## [1] 0.7
```

Ceci signifie que 50% des diamants ont une valeur en carat inférieure ou égale à 0.7 et que 50% des diamants ont une valeur en carat supérieure ou égale à 0.7.

Nous pouvons aussi obtenir que la médiane de la variable `price` de la base de données `diamonds` est donnée par:

```
median(diamonds$price)
## [1] 2401
```

### 12.1.3 La moyenne

La moyenne est la valeur qui pourrait remplacer chacune des données d'une série pour que leur somme demeure identique. Intuitivement, elle représente le centre d'équilibre d'une série de données. La somme des distances qui sépare les données plus petites que la moyenne devrait être la même que la somme des distances qui sépare les données plus grandes.

Important : La moyenne n'est définie que pour les variables quantitatives. En effet, si vous tentez d'utiliser la moyenne pour des données autres que numériques, R vous donnera un message d'erreur.

La fonction `mean` permet de calculer la moyenne en langage R.

Par exemple, pour calculer la moyenne de la variable `carat` de la base de données `diamonds`, nous avons:

```
mean(diamonds$carat)
## [1] 0.798
```

Nous pouvons aussi obtenir que la moyenne de la variable `price` de la base de données `diamonds` est donnée par:

```
mean(diamonds$price)
## [1] 3933
```

## 12.2 Les mesures de dispersion

Les mesures de tendance centrale (mode, moyenne et médiane) ne permettent pas de déterminer si une série de données est principalement située autour de son centre, ou si au contraire elle est très dispersée.

Les mesures de dispersion, elles, permettent de déterminer si une série de données est centralisée autour de sa moyenne, ou si elle est au contraire très dispersée.

Les mesures de dispersion sont l'étendue, la variance, l'écart-type et le coefficient de variation.

### 12.2.1 L'étendue

La première mesure de dispersion, l'étendue, est la différence entre la valeur maximale et la valeur minimale.

L'étendue ne tenant compte que du maximum et du minimum, elle est grandement influencée par les valeurs extrêmes. Elle est donc une mesure de dispersion peu fiable.

La fonction `range` permet de calculer l'étendue d'une variable en langage R.

Par exemple, pour calculer l'étendue de la variable `carat` de la base de données `diamonds`, nous avons:

```
range(diamonds$carat)
## [1] 0.20 5.01
```

Nous pouvons donc calculer l'étendue de la variable `carat` en soustrayant les deux valeurs obtenues par la fonction `range`, c'est-à-dire que l'étendue est  $5.01 - 0.2 = 4.81$ .

### 12.2.2 La variance

La variance sert principalement à calculer l'écart-type, la mesure de dispersion la plus connue.

Attention : Les unités de la variance sont des unités<sup>2</sup>.

La fonction `var` permet de calculer la variance d'une variable en langage R.

Par exemple, pour calculer la variance de la variable `carat` de la base de données `diamonds`, nous avons:

```
var(diamonds$carat)
## [1] 0.225
```

Ceci signifie que la variance de la variable `carat` est  $0.225 \text{ carat}^2$ .

### 12.2.3 L'écart-type

L'écart-type est la mesure de dispersion la plus couramment utilisée. Il peut être vu comme la « moyenne » des écarts entre les données et la moyenne.

Puisque l'écart-type tient compte de chacune des données, il est une mesure de dispersion beaucoup plus fiable que l'étendue.

Il est défini comme la racine carrée de la variance.

La fonction `sd` permet de calculer l'écart-type d'une variable en langage R.

Par exemple, pour calculer l'écart-type de la variable `carat` de la base de données `diamonds`, nous avons :

```
sd(diamonds$carat)
## [1] 0.474
```

Ceci signifie que l'écart-type de la variable `carat` est 0.474 carat.

### 12.2.4 Le coefficient de variation

Le coefficient de variation, noté C. V., est calculé comme suit :

$$C.V. = \frac{\text{ecart-type}}{\text{moyenne}} \times 100\% \quad (12.1)$$

Si le coefficient est inférieur à 15%, les données sont dites **homogènes**. Cela veut dire que les données sont situées près les unes des autres.

Dans le cas contraire, les données sont dites **hétérogènes**. Cela veut dire que les données sont très dispersées.

Important : Le coefficient de variation ne possède pas d'unité, outre le symbole de pourcentage.

Il n'existe pas de fonctions en R permettant de calculer directement le coefficient de variation. Par contre, nous pouvons utiliser en conjonction les fonctions `sd` et `mean` pour le calculer.

Par exemple, pour calculer le coefficient de variation de la variable `carat` de la base de données `diamonds`, nous avons :

```
sd(diamonds$carat)/mean(diamonds$carat)*100
## [1] 59.4
```

Le C.V. de la variable `carat` est donc 59.404 %, ce qui signifie que les données sont hétérogènes, car le coefficient de variation est plus grand que 15%.

## 12.3 Les mesures de position

Les mesures de position permettent de situer une donnée par rapport aux autres. Les différentes mesures de position sont la cote Z, les quantiles et les rangs.

Tout comme les mesures de dispersion, celles-ci ne sont définies que pour une variable quantitative.

### 12.3.1 La cote z

Cette mesure de position se base sur la moyenne et l'écart-type.

La cote Z d'une donnée x est calculée comme suit :



$$Z = \frac{x - \text{moyenne}}{\text{ecart-type}} \quad (12.2)$$

Important : La cote  $z$  ne possède pas d'unités.

Une cote  $Z$  peut être positive, négative ou nulle.

| Cote Z  | Interprétation                 |
|---------|--------------------------------|
| $Z > 0$ | donnée supérieure à la moyenne |
| $Z < 0$ | donnée inférieure à la moyenne |
| $Z = 0$ | donnée égale à la moyenne      |

Il n'existe pas de fonctions en R permettant de calculer directement la cote  $Z$ . Par contre, nous pouvons utiliser en conjonction les fonctions `sd` et `mean` pour la calculer.

Par exemple, si nous voulons calculer la cote  $Z$  d'un diamant de 3 carats, nous avons:

```
(3-mean(diamonds$carat))/sd(diamonds$carat)
## [1] 4.65
```

### 12.3.2 Les quantiles

Un quantile est une donnée qui correspond à un certain pourcentage cumulé.

Parmi les quantiles, on distingue les quartiles, les quintiles, les déciles et les centiles.

- Les quartiles  $Q_1$ ,  $Q_2$  et  $Q_3$ , séparent les données en quatre parties égales. Environ 25% des données sont inférieures ou égales à  $Q_1$ . Environ 50% des données sont inférieures ou égales à  $Q_2$ . Environ 75% des données sont inférieures ou égales à  $Q_3$ .
- Les quintiles  $V_1$ ,  $V_2$ ,  $V_3$  et  $V_4$ , séparent les données en cinq parties égales. Environ 20% des données sont inférieures ou égales à  $V_1$ . Environ 40% des données sont inférieures ou égales à  $V_2$ . Etc.
- Les déciles  $D_1$ ,  $D_2$ , ...,  $D_8$  et  $D_9$ , séparent les données en dix parties égales. Environ 10% des données sont inférieures ou égales à  $D_1$ . Environ 20% des données sont inférieures ou égales à  $D_2$ . Etc.
- Les centiles  $C_1$ ,  $C_2$ , ...,  $C_{98}$  et  $C_{99}$ , séparent les données en cent parties égales. Environ 1% des données sont inférieures ou égales à  $C_1$ . Environ 2% des données sont inférieures ou égales à  $C_2$ . Etc.

Il est utile de noter que certains quantiles se recoupent.

La fonction `quantile` permet de calculer n'importe quel quantile d'une variable en langage R. Il suffit d'indiquer la variable étudiée ainsi que le pourcentage du quantile voulu.

Par exemple, si nous voulons calculer  $D_1$  pour la variable `carat`, nous allons utiliser la fonction `quantile` avec une probabilité de 0,1.

```
quantile(diamonds$carat, 0.1)
## 10%
## 0.31
```

Ceci implique que 10% des diamants ont une valeur en carat inférieure ou égale à 0.31 carat.

Nous pouvons calculer le troisième quartile  $Q_3$  de la variable `price` en utilisant la fonction `quantile` avec une probabilité de 0,75.

```
quantile(diamonds$price, 0.75)
## 75%
## 5324
```

Ceci implique que 75% des diamants ont un prix en dollars inférieur ou égal à 5324.25 \$.

### 12.3.3 La commande `summary`

La commande `summary` produit un sommaire contenant six mesures importantes:

1. **Min** : le minimum de la variable
2. **1st Qu.** : Le premier quartile,  $Q_1$ , de la variable
3. **Median** : La médiane de la variable
4. **Mean** : La moyenne de la variable
5. **3rd Qu.** : Le troisième quartile,  $Q_3$ , de la variable
6. **Max** : Le maximum de la variable

Nous pouvons donc produire le sommaire de la variable `price` de la base de données `diamonds` de la façon suivante:

```
summary(diamonds$price)
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      326     950     2401    3933     5324    18823
```

### 12.3.4 Le rang centile

Un rang centile représente le pourcentage cumulé, *exprimé en nombre entier*, qui correspond à une certaine donnée. Nous déterminerons les rangs centiles pour les variables continues seulement.

Les rangs centiles sont donc exactement l'inverse des centiles.

Il n'existe pas de fonctions dans **R** permettant de trouver directement le rang centile, mais il est facile d'utiliser la fonction `mean` pour le trouver.

Par exemple, si nous voulons trouver le rang centile d'un diamant qui coûte 500\$, il suffit d'utiliser la commande suivante. La commande calcule la moyenne de toutes les valeurs en dollars des diamants coûtant 500\$ ou moins.

```
mean(diamonds$price<=500)
## [1] 0.0324
```

Ceci signifie que pour un diamant de 500\$, il y a 3.242 % des diamants qui ont une valeur égale ou inférieure.

# Bibliographie

- Barnier, J. (2018). *Introduction à R et au tidyverse*.
- Ismay, C. (2018). *Getting used to R, RStudio, and R Markdown*.
- Wickham, H. (2014). Tidy data. *Journal of Statistical Software, Articles*, 59(10):1–23.
- Wickham, H. (2017a). *nycflights13: Flights that Departed NYC in 2013*. R package version 0.2.2.
- Wickham, H. (2017b). *tidyverse: Easily Install and Load the 'Tidyverse'*. R package version 1.2.1.
- Wickham, H. (2018). *forcats: Tools for Working with Categorical Variables (Factors)*. R package version 0.3.0.
- Wickham, H. and Grommund, G. (2017). *R for Data Science*. O'Reilly Media Inc., 1st edition. ISBN 978-1491910399.