Statistiques et Probabilités avec R, RStudio et le Tidyverse

Marc-André Désautels

2018-04-21

Table des Matières

Introduction **7**

I Les outils 9

1 Les logiciels R et RStudio **11**
1.1 Qu'est-ce que R? . . . 11
1.2 Qu'est-ce que RStudio? . . . 12
1.3 Les bases de RStudio . . . 12

2 Le tidyverse **15**
2.1 Extensions . . . 15
2.2 Installation . . . 16
2.3 Les tidy data . . . 16
2.4 Les tibbles . . . 17

3 L'extension `questionr` **23**
3.1 Mise en place . . . 23
3.2 L'interface graphique . . . 23

II Introduction 25

4 La démarche scientifique **27**

5 Les différents types de variables **29**
5.1 Introduction . . . 29
5.2 Les variables qualitatives . . . 29
5.3 Les variables quantitatives . . . 32

6 Construire un questionnaire **35**
6.1 Critères à respecter . . . 35
6.2 Types de questions . . . 35

7 Les échelles de mesure **37**

8 Les techniques d'échantillonnage **39**
8.1 Techniques d'échantillonnage aléatoires . . . 39
8.2 Techniques d'échantillonnage non-aléatoires . . . 39
8.3 Base de données pour les *M&M's* . . . 40

III Présentation des données **41**

9 Les variables qualitatives **43**
9.1 Mise en place . . . 43
9.2 Tableau de fréquences . . . 43
9.3 Représentation graphique - Le diagramme à bandes . . . 44

10 Les variables quantitatives discrètes **47**
10.1 Mise en place . . . 47
10.2 Tableau de fréquences . . . 47
10.3 Représentation graphique - Le diagramme à bandes . . . 51

11 Les variables quantitatives continues **53**
11.1 Mise en place . . . 53
11.2 Tableau de fréquences . . . 53
11.3 Représentation graphique - L'histogramme . . . 55

12 Deux variables **57**
12.1 Mise en place . . . 57
12.2 Croisement de deux variables qualitatives . . . 57
12.3 Croisement d'une variable qualitative et d'une variable quantitative . . . 61
12.4 Croisement de deux variables quantitatives . . . 64

IV Les mesures **65**

13 Les proportions **67**
13.1 Mise en place . . . 67

14 Les mesures de tendance centrale **69**
14.1 Mise en place . . . 69
14.2 Le mode . . . 69
14.3 La médiane . . . 70
14.4 La moyenne . . . 71

15 Les mesures de dispersion **73**
15.1 L'étendue . . . 73
15.2 La variance . . . 73
15.3 L'écart-type . . . 74
15.4 Le coefficient de variation . . . 74

16 Les mesures de position **75**
16.1 La cote z . . . 75
16.2 Les quantiles . . . 75
16.3 La commande `summary` . . . 76
16.4 Le rang centile . . . 76

V Les données construites **79**

17 Les séries chronologiques **81**
17.1 Mise en place . . . 81

18 Les données construites **83**
18.1 Mise en place . . . 83

VI L'analyse de lien **85**

19 La corrélation linéaire **87**
19.1 Mise en place . . . 87
19.2 Le nuage de points . . . 88
19.3 Fake data . . . 91
19.4 Le quartet d'Anscombe . . . 92
19.5 DatasauRus . . . 93

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 **R**oss Ihaka et **R**obert Gentleman en Nouvelle-Zélande à l'Université d'Auclkand.

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/

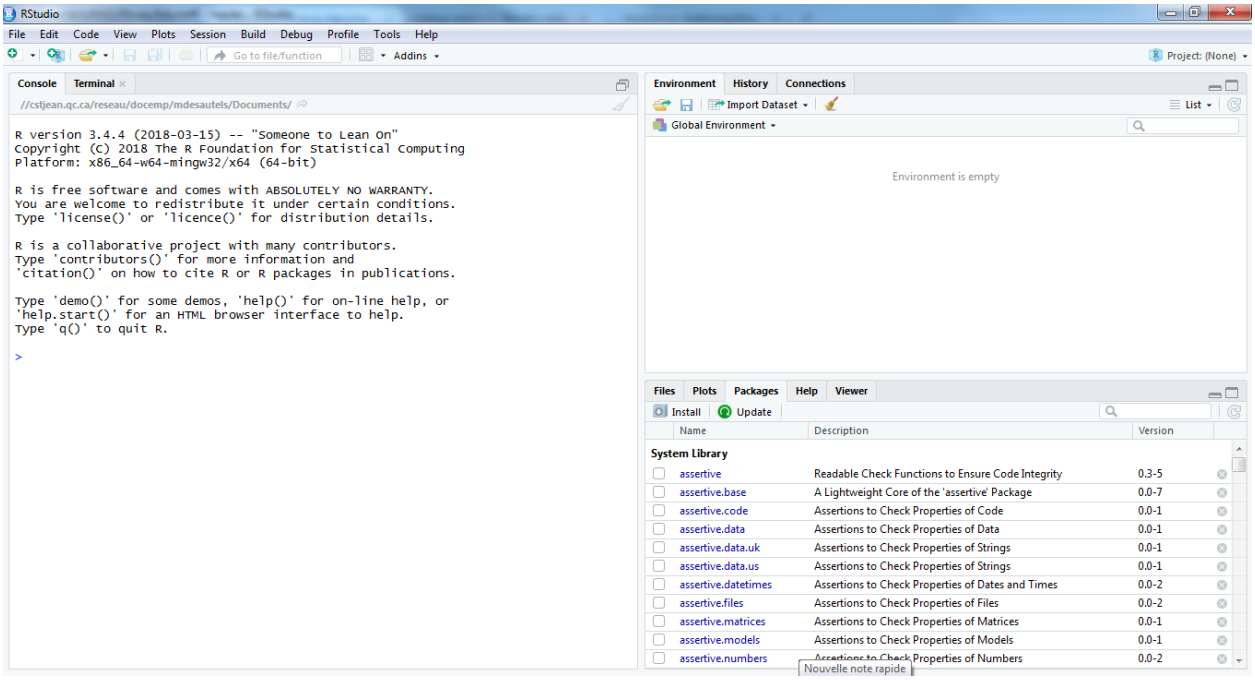


Figure 1.1: L'interface de RStudio

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 utilisez sur votre ordinateur), vous allez à l'adresse suivante:

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

1.3 Les bases de RStudio

1.3.1 La console

Au premier lancement de RStudio, l'écran est séparé en trois grandes zones:

La zone de gauche se nomme *Console*. À son démarrage, RStudio a lancé une nouvelle session de R et c'est dans cette fenêtre que nous allons pouvoir interagir avec lui.

Vous devriez voir un texte ressemblant à celui-ci:

```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
```

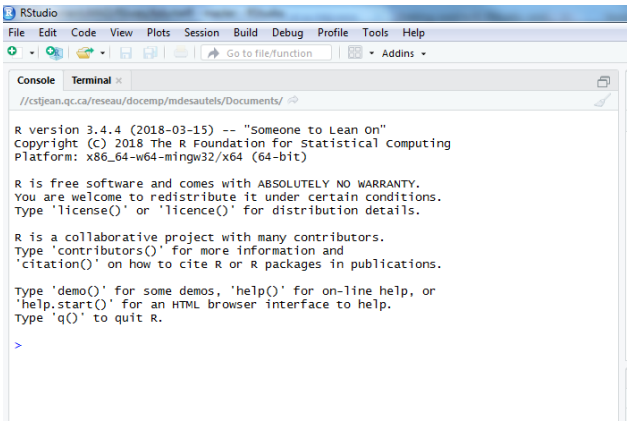


Figure 1.2: La console R

```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

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

La console ressmble à ceci:

La ligne qui débute avec le symbole `>` est appelée l'invite de commande (ou prompt en anglais). Elle signifie que R est disponible et en attente de votre prochaine commande. C'est à cet endroit que nous pouvons entrer des commandes et les éxécuter en appuyabt sur `Entrée`.

1.3.2 Environment/History/Connections

TODO

1.3.3 Files/Plots/Packages/Help/Viewer

TODO

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 philosphie 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 **coeur** 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:

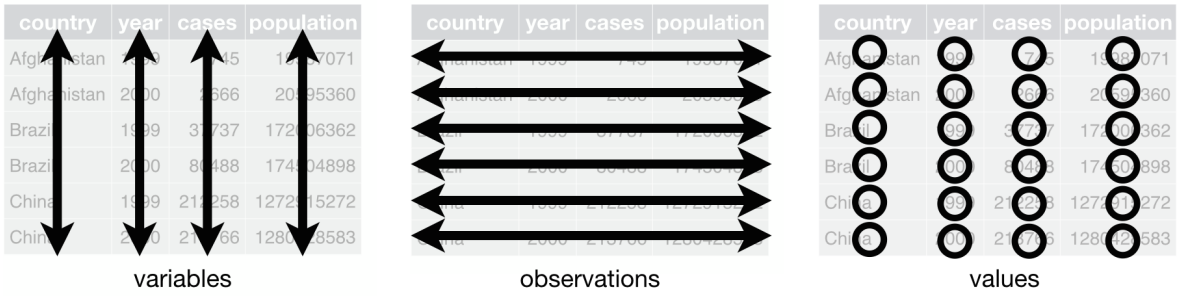


Figure 2.1: Suivre les trois principes rend les données tidy: les variables sont en colonnes, les observations sont sur des lignes, et chaques valeurs sont dans des cellules différentes.

1. Un avantage général de choisir une seule façon de conserver vos données. Si vous utilisez une structure de données consitante, il est plus facile d'apprendre à utiliser les outils qui fonctionneront avec ce type de structure, étant donné que celles-ci possède 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 est géré par l'extension du même nom (`tibble`), qui fait partie du coeur 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 <dttm>
```

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'obervations 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 / 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 fr?quemm
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()`:

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

Nous remarquons que chaque colonnes représentent 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,...
```

```
#> $ 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, 6, 5,...
#> $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, ...
#> $ time_hour      <dttm> 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.

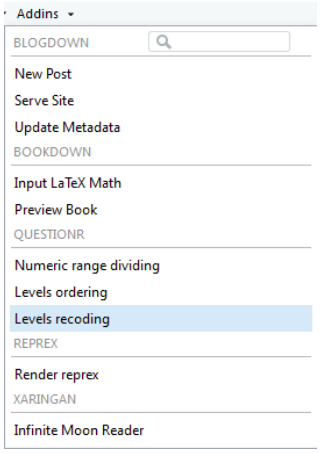


Figure 3.1: Levels recoding dans le menu Addins

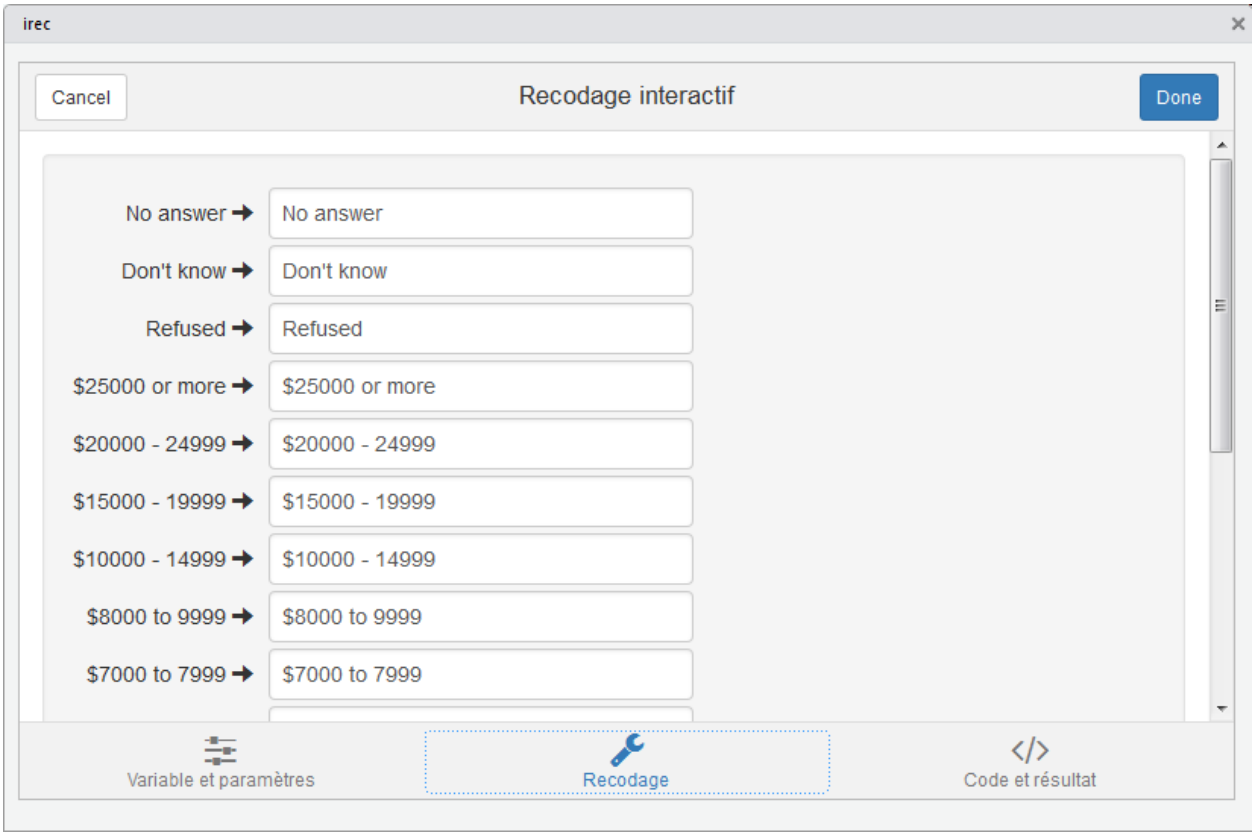


Figure 3.2: L'interface graphique de Levels recoding

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", "Jam", "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 problèmes 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 donnéee `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 example:

```
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 utilisez 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 chaques jours 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 <dttm>
```

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*

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*)	(*Fréquences relatives*)
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.

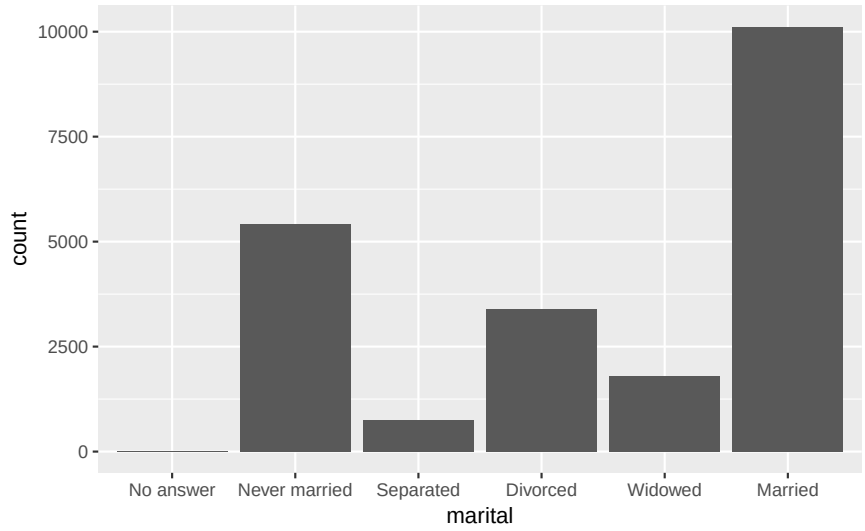
9.3.1 La variable `marital`

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

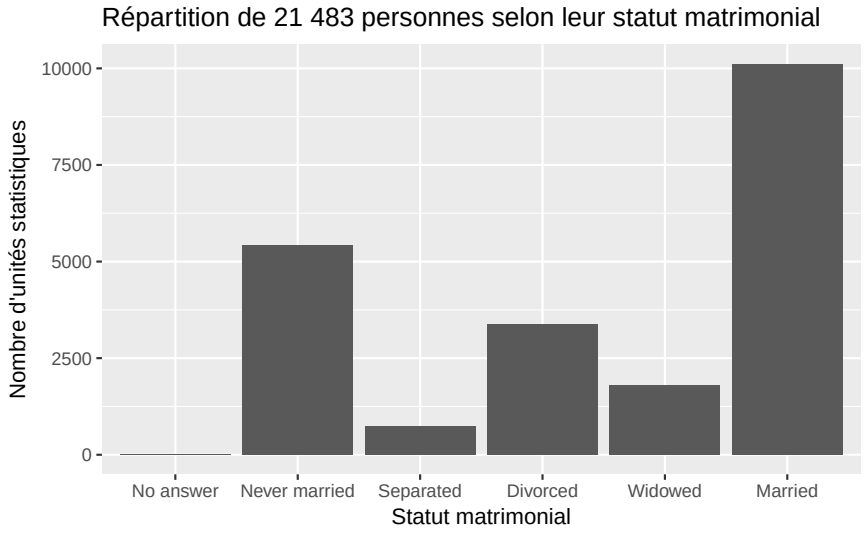
La commande est donc:

```
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*).

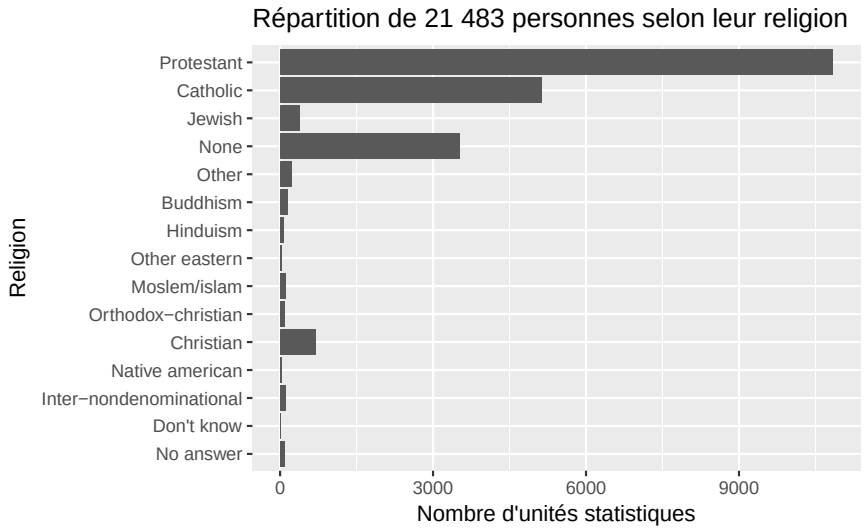
```
ggplot(data = gss_cat, aes(x = marital))+
  geom_bar()+
  labs(
    title = "Répartition de 21 483 personnes selon leur statut matrimonial",
    x = "Statut matrimonial",
    y = "Nombre d'unités statistiques"
  )
```



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 (*Valeurs*)	**Nombre d'unités statistiques** (*Fréquences absolues*)	**Pourcentage d'unités statistiques** (%) (*Fréquences relatives*)	**Pourcentage cumulé** (*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.

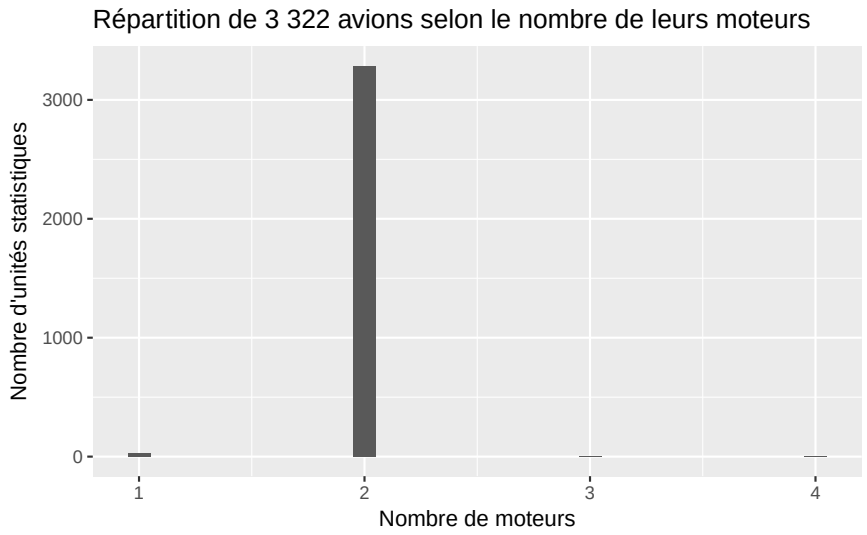
Important! Au chapitre 5, nous avons étudié les différents types de variables. Parmi les variables quantitatives, nous avons distingué celles qui étaient discrètes de celles qui étaient continues.Bien que cela s'applique toujours, **il est important de noter qu'une variable continue** (par exemple, l'âge) **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.

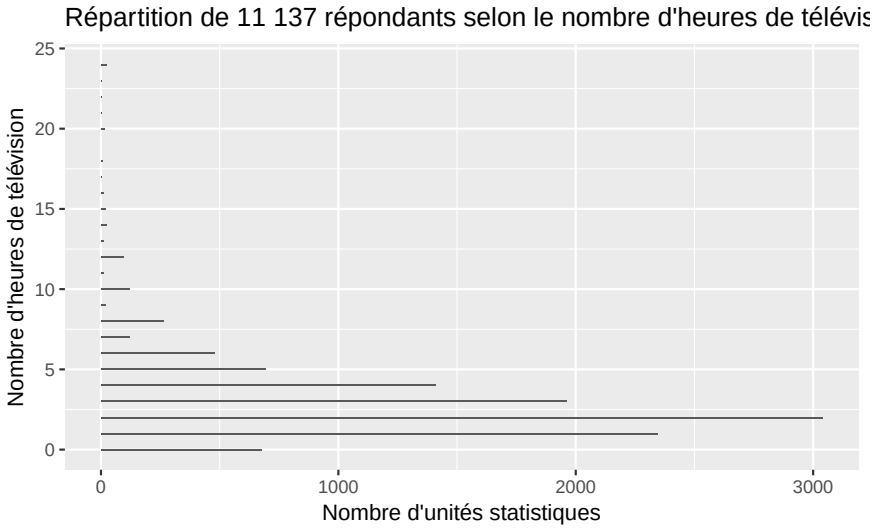
Ce graphique est construit comme le diagramme à bandes rectangulaires verticales, sauf que les rectangles sont remplacés par des bâtons très minces (généralement une simple ligne). Pour modifier la largeur de vos lignes, vous utilisez l'option `width` dans la commande `geom_bar()`.

```
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).
```


Répartition de 11 137 répondants selon le nombre d'heures de télévis
Nombre d'heures de télévision
25
20
15
10
5
0
0
1000
2000
3000
Nombre d'unités statistiques

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 devrons 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 variable (*Classes*)	**Nombre d'unités statistiques** (*Fréquences absolues*)	**Pourcentage d'unités statistiques** (%) (*Fréquences relatives*)	**Pourcentage cumulé** (*Fréquences relatives 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 extrèmes
- `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

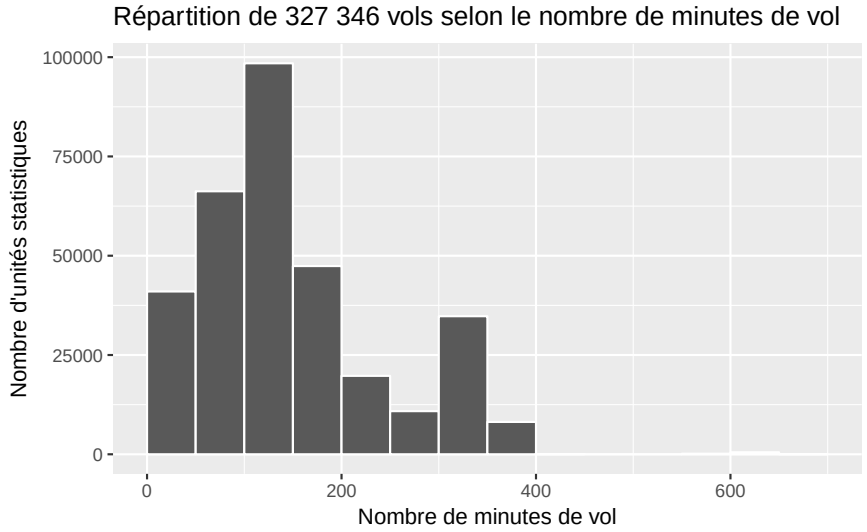
À la section 10.2, nous avons vu que la variable `seats` de la base de données `planes` contenait 48 valeurs différentes. Nous allons donc créer le tableau de fréquences avec des classes.

```
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

Deux variables

12.1 Mise en place

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

12.2 Croisement de deux variables qualitatives

Quand on veut croiser deux variables qualitatives, on fait un tableau croisé.

12.2.1 Tableaux à double entrée

Lorsque deux variables (peu importe leur type) sont étudiées simultanément, on construit un tableau à double entrée.

	Nom de la variable 2	
Nom de la variable 1 (Modalités de la variable 1) **Total**	(Modalités de la variable 2) (Fréquences) (Sous-totaux)	**Total** (Sous-totaux) (Total)

Nous utilisons la commande `table` à laquelle on passe cette fois deux variables en argument. Par exemple, en utilisant la base de données `gss_cat`, nous pouvons croiser les variables `marital` et `race`.

```
table(gss_cat$marital,gss_cat$race)
#>
#>                 Other Black White Not applicable
#>   No answer         2     2    13               0
#>   Never married   633  1305  3478               0
#>   Separated       110   196   437               0
#>   Divorced        212   495  2676               0
#>   Widowed          70   262  1475               0
#>   Married         932   869  8316               0
```

Nous pouvons exclure certaines modalités en utilisant l'option `exclude`. Par exemple, on peut exclure les modalités *Not applicable* de la façon suivante:

```
table(gss_cat$marital,gss_cat$race, exclude = c("Not applicable"))
#>
#>                 Other Black White
#>   No answer         2     2    13
#>   Never married   633  1305  3478
#>   Separated       110   196   437
#>   Divorced        212   495  2676
#>   Widowed          70   262  1475
#>   Married         932   869  8316
```

Nous pouvons obtenir un tableau à double entrée comportant des pourcentages à l'aide de la commande `prop`.

```
prop(table(gss_cat$marital,gss_cat$race))
#>
#>                 Other Black White Total
#>   No answer       0.0   0.0   0.1   0.1
#>   Never married   2.9   6.1  16.2  25.2
#>   Separated       0.5   0.9   2.0   3.5
#>   Divorced        1.0   2.3  12.5  15.7
#>   Widowed         0.3   1.2   6.9   8.4
#>   Married         4.3   4.0  38.7  47.1
#>   Total           9.1  14.6  76.3 100.0
```

Pour connaître toutes les options de la commande `prop`, vous pouvez utilisez la commande `?prop` dans la console.

Nous pouvons également obtenir les totaux des lignes et des colonnes en utilisant la commande `addmargins`:

```
addmargins(table(gss_cat$marital,gss_cat$race))
#>
#>                 Other Black White Not applicable   Sum
#>   No answer         2     2    13              0    17
#>   Never married   633  1305  3478              0  5416
#>   Separated       110   196   437              0   743
#>   Divorced        212   495  2676              0  3383
#>   Widowed          70   262  1475              0  1807
#>   Married         932   869  8316              0 10117
#>   Sum            1959  3129 16395              0 21483
```

Pour pouvoir interpréter ce tableau on doit passer du tableau en effectifs au tableau en pourcentages ligne ou colonne. Pour cela, on peut utiliser les fonctions `lprop` et `cprop` de l'extension `questionr`, qu'on applique au tableau croisé précédent.

Pour calculer des pourcentages lignes.

```
lprop(table(gss_cat$marital,gss_cat$race))
#>
#>                 Other Black White Total
#>   No answer      11.8  11.8  76.5 100.0
#>   Never married  11.7  24.1  64.2 100.0
#>   Separated      14.8  26.4  58.8 100.0
#>   Divorced        6.3  14.6  79.1 100.0
#>   Widowed         3.9  14.5  81.6 100.0
#>   Married         9.2   8.6  82.2 100.0
```

```
#>   All             9.1  14.6  76.3 100.0
```

Pour calculer des pourcentages colonnes.

```
cprop(table(gss_cat$marital,gss_cat$race))
#>
#>                 Other Black White All
#>   No answer       0.1   0.1   0.1   0.1
#>   Never married  32.3  41.7  21.2  25.2
#>   Separated       5.6   6.3   2.7   3.5
#>   Divorced       10.8  15.8  16.3  15.7
#>   Widowed         3.6   8.4   9.0   8.4
#>   Married        47.6  27.8  50.7  47.1
#>   Total         100.0 100.0 100.0 100.0
```

Comme vous pouvez le constater, les commandes `lprop` et `cprop` enlève les lignes ou colonnes dont la somme des effectifs est zéro. Pour empêcher ce comportement, vous devez utilisez l'option `drop = FALSE`. Par exemple:

```
lprop(table(gss_cat$marital,gss_cat$race), drop = FALSE)
#>
#>                 Other Black White Not applicable Total
#>   No answer      11.8  11.8  76.5   0.0          100.0
#>   Never married  11.7  24.1  64.2   0.0          100.0
#>   Separated      14.8  26.4  58.8   0.0          100.0
#>   Divorced        6.3  14.6  79.1   0.0          100.0
#>   Widowed         3.9  14.5  81.6   0.0          100.0
#>   Married         9.2   8.6  82.2   0.0          100.0
#>   All             9.1  14.6  76.3   0.0          100.0
```

Pour connaître toutes les options de ces deux commandes, vous pouvez taper `?lprop` ou `?cprop` dans la console.

12.2.2 Représentation graphique - diagramme à bandes chevauchées

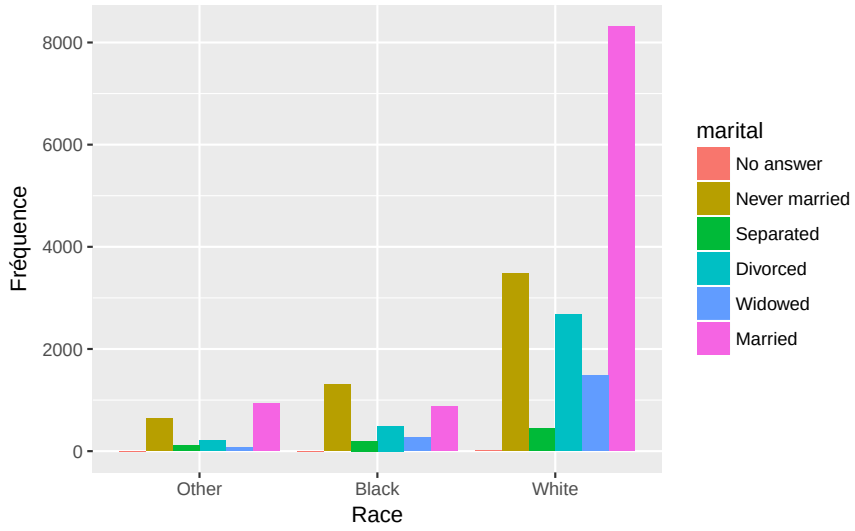
Le graphique utilisé pour représenter simultanément deux variables qualitatives est le diagramme à bandes rectangulaires chevauchées.

Ce graphique ressemble au diagramme à bandes rectangulaires verticales, à la différence qu'il y aura au moins deux rectangles pour chacune des modalités ainsi qu'une légende.

Encore une fois, ce graphique peut être construit à partir des fréquences absolues ou relatives.

Nous pouvons comparer les variable `race` et `marital` avec des diagrammes à bandes chevauchées en utilisant l'option `position = "dodge"`.

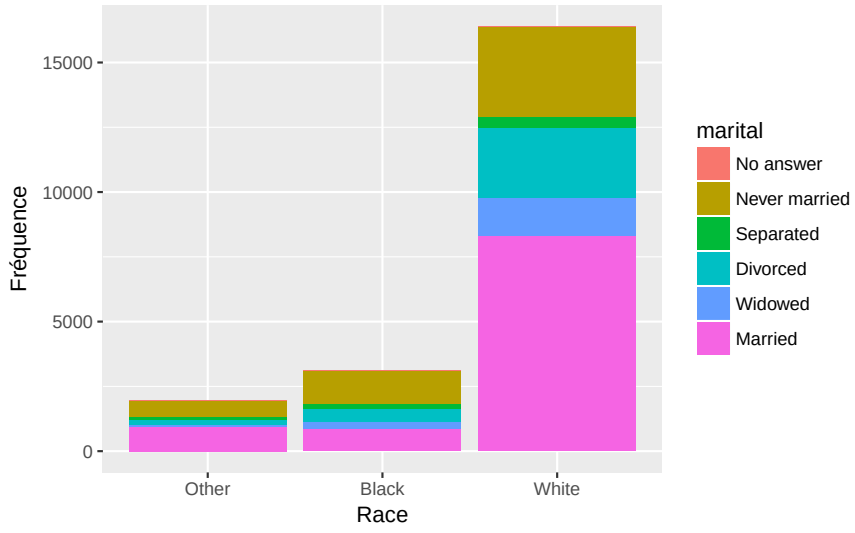
```
ggplot(data = gss_cat, aes(x = race, fill = marital))+
  geom_bar(position = "dodge")+
  labs(x = "Race",
       y = "Fréquence")
```



12.2.3 Représentation graphique - diagramme à bandes superposées

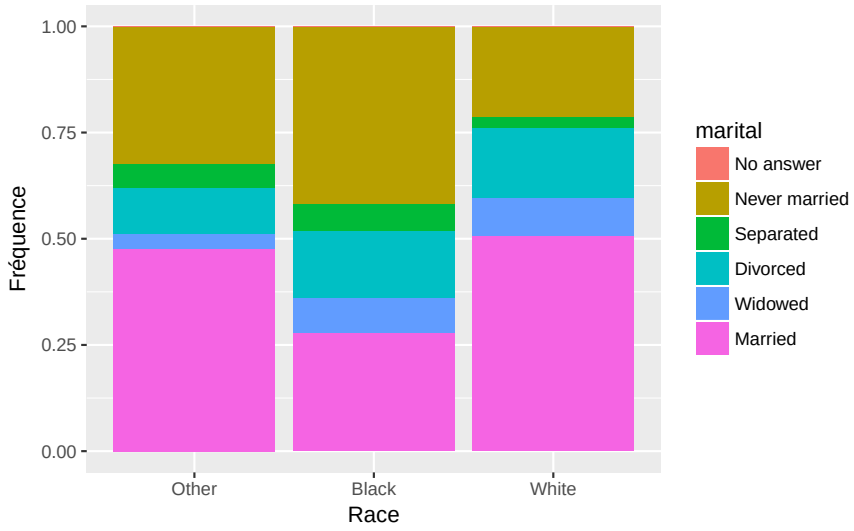
Si nous n'utilisons pas l'option `position = "dodge"`, nous obtenons des diagrammes à bandes superposées.

```
ggplot(data = gss_cat, aes(x = race, fill = marital))+
  geom_bar()+
  labs(x = "Race",
       y = "Fréquence")
```



Pour obtenir des diagrammes comportant des fréquences relatives, nous utilisons l'option `position = "fill"`.

```
ggplot(data = gss_cat, aes(x = race, fill = marital))+
  geom_bar(position = "fill")+
  labs(x = "Race",
       y = "Fréquence")
```



12.3 Croisement d'une variable qualitative et d'une variable quantitative

12.3.1 Représentation graphique - boîte à moustaches

Croiser une variable quantitative et une variable qualitative, c'est essayer de voir si les valeurs de la variable quantitative se répartissent différemment selon la catégorie d'appartenance de la variable qualitative.

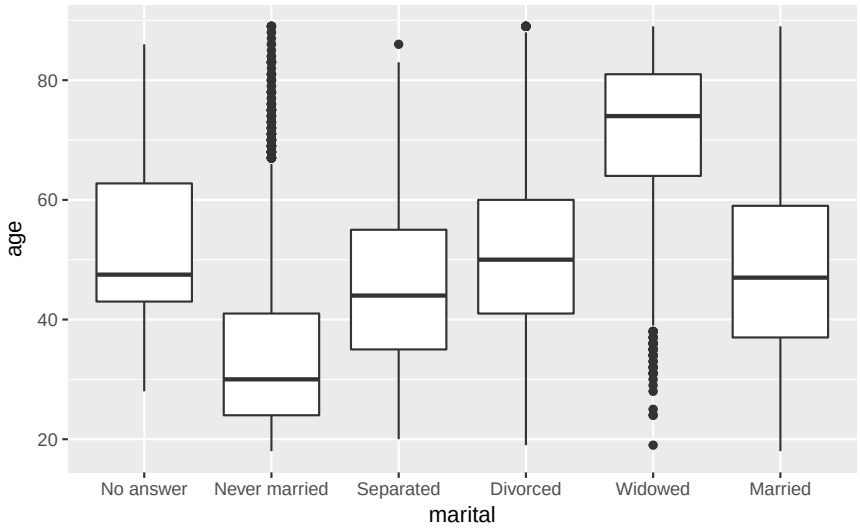
Pour cela, l'idéal est de commencer par une représentation graphique de type "boîte à moustache".

> L'interprétation d'une boîte à moustaches est la suivante : Les bords inférieurs et supérieurs du carré central représentent le premier et le troisième quartile de la variable représentée sur l'axe vertical. On a donc 50% de nos observations dans cet intervalle. Le trait horizontal dans le carré représente la médiane. Enfin, des "moustaches" s'étendent de chaque côté du carré, jusqu'aux valeurs minimales et maximales, avec une exception : si des valeurs sont éloignées du carré de plus de 1,5 fois l'écart interquartile (la hauteur du carré), alors on les représente sous forme de points (symbolisant des valeurs considérées comme "extrêmes").

Nous discuterons plus en détail des toutes ces mesures au chapitre 16.

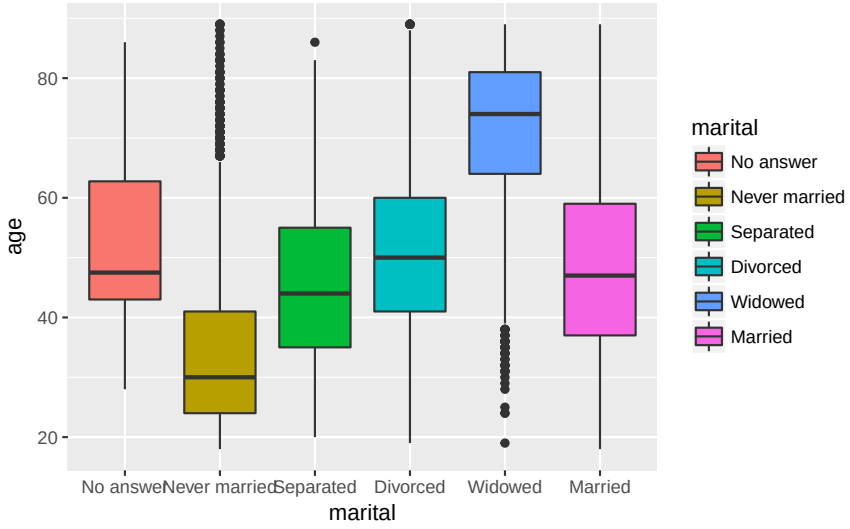
Voici le graphique boîte à moustaches représentant les variables `marital` et `age`.

```
ggplot(data = gss_cat, aes(x = marital, y = age))+
  geom_boxplot()
#> Warning: Removed 76 rows containing non-finite values (stat_boxplot).
```



Pour mieux visualiser vos boîtes à moustache, vous pouvez les colorier. Si nous voulons colorier les boîtes en fonction de la variable `marital`, nous ajoutons l'option `fill = marital`..

```
ggplot(data = gss_cat, aes(x = marital, y = age, fill = marital))+
  geom_boxplot()
#> Warning: Removed 76 rows containing non-finite values (stat_boxplot).
```



12.3.2 Représentation graphique - diagramme en violon

Nous pouvons également visualiser le lien entre une variable qualitative et une variable quantitative à l'aide d'un diagramme en violon.

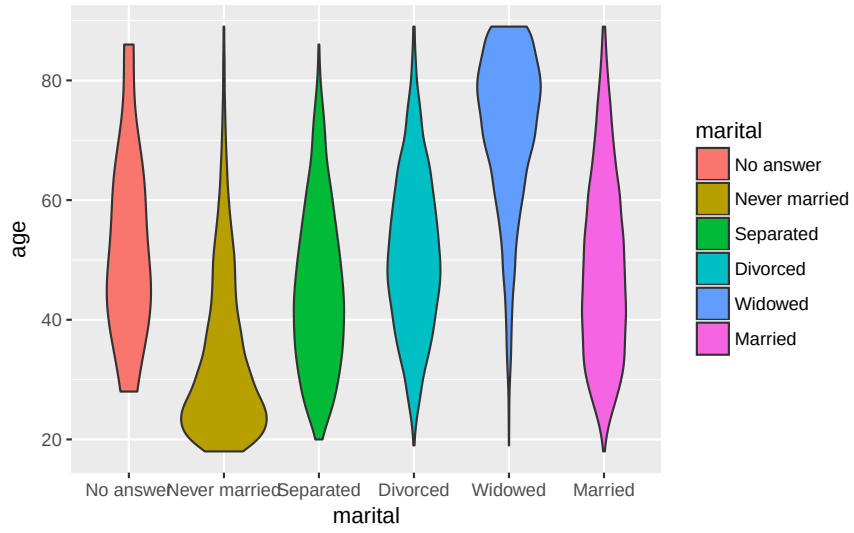
> L'interprétation du diagramme en violon est la suivante: La largeur du diagramme nous renseigne sur la fréquence d'apparition de la variable. Plus ils sont larges, plus la valeur de la variable est fréquente et inversement.

Nous pouvons représenter les diagrammes en violon de la variable `marital` et de la variable `age`.

```
ggplot(data = gss_cat, aes(x = marital, y = age))+
  geom_violin()
#> Warning: Removed 76 rows containing non-finite values (stat_ydensity).
```

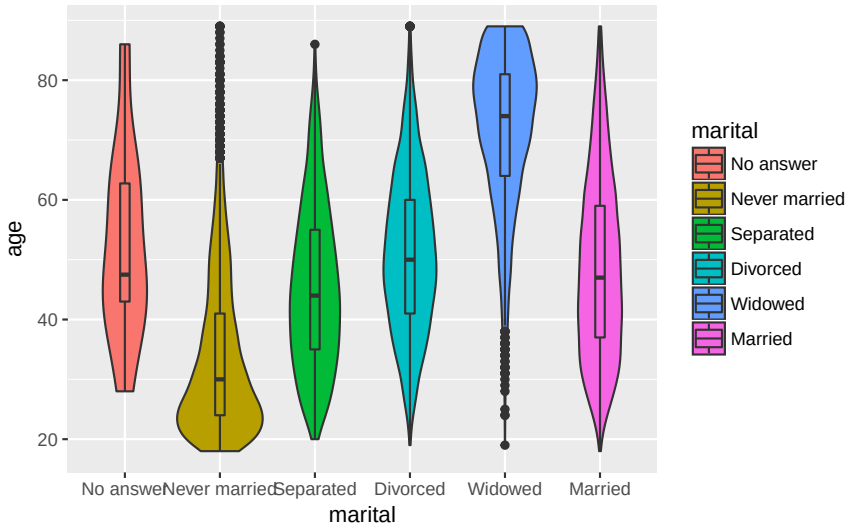
Nous pouvons ajouter de la couleur avec l'option `fill`.

```
ggplot(data = gss_cat, aes(x = marital, y = age, fill = marital))+
  geom_violin()
#> Warning: Removed 76 rows containing non-finite values (stat_ydensity).
```



Nous pouvons bien sûr superposer des boîtes à moustaches et des diagrammes en violon.

```
ggplot(data = gss_cat, aes(x = marital, y = age, fill = marital))+
  geom_violin()+
  geom_boxplot(width = 0.1)
#> Warning: Removed 76 rows containing non-finite values (stat_ydensity).
#> Warning: Removed 76 rows containing non-finite values (stat_boxplot).
```



12.4 Croisement de deux variables quantitatives

```
#ggplot(data = flights, aes(x = dep_delay, y = arr_delay))+
#  geom_point(alpha = 0.25)
```

Partie IV

Les mesures

Chapitre 13

Les proportions

Les proportions, exprimées en pourcentage (%), permettent de comparer la taille de deux ensembles. Le symbole utilisé pour représenter une proportion dépend d'où proviennent les données.

- Si les données proviennent d'une population, la proportion sera notée π.
- Si elles proviennent d'un échantillon, la proportion sera notée $\hat{p}$.

  Attention: π indique une proportion et ne représente donc pas 3,1415...

13.1 Mise en place

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

Chapitre 14

Les mesures de tendance centrale

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

Nous allons charger les librairies que nous allons utiliser:

14.1 Mise en place

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

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.

14.2 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 `marital` de la base de données `gss_cat`:

```
table(gss_cat$marital)
#>
#>     No answer Never married     Separated      Divorced       Widowed
#>            17          5416           743          3383          1807
#>       Married
#>         10117
```

Nous remarquons que le maximum est à la modalité *Married* avec une fréquence de 10117.

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

```
table(gss_cat$age)
#>
#>  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
#>  91 249 251 278 298 361 344 396 400 385 387 376 433 407 445 425 425 417
#>  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
#> 428 438 426 415 452 434 405 448 432 404 422 435 424 417 430 390 400 396
#>  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
#> 387 365 384 321 326 323 338 307 310 292 253 259 231 271 205 201 213 206
#>  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
#> 189 152 180 179 171 137 150 135 127 119 105  99 100  75  74  54  57 148
```

Nous remarquons que le maximum est à la valeur *40* avec une fréquence de 452.

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.

14.3 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
```

14.4 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
```

Chapitre 15

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.

15.1 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.

15.2 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^2.

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 carat^2.

15.3 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.

15.4 Le coefficient de variation

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

$$C.V. = \frac{\text{ecart-type}}{\text{moyenne}} \times 100\% \tag{15.1}$$

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%.

Chapitre 16

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.

16.1 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 :

$$Z = \frac{x - \text{moyenne}}{\text{ecart-type}} \tag{16.1}$$

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
```

16.2 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 $.

16.3 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
```

16.4 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.

Partie V

Les données construites

Chapitre 17

Les séries chronologiques

Une série chronologique est un ensemble de valeurs observées d'une variable quantitative. Elle permet d'analyser l'évolution de cette variable dans le temps dans le but éventuel de faire des prévisions. Le tableau utilisé pour représenter les données d'une série chronologique comporte une colonne pour la période ainsi qu'une colonne pour la valeur observée.

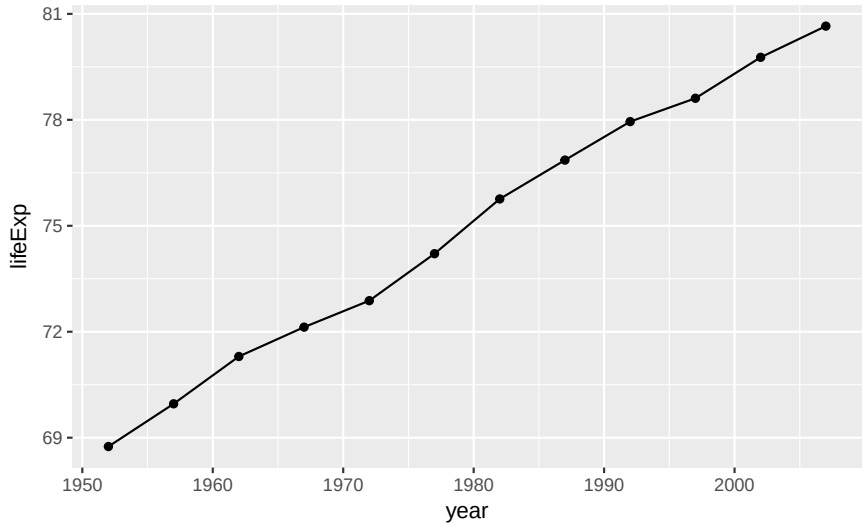
Pour ce chapitre, nous utiliserons la librairie `gapminder`.

17.1 Mise en place

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

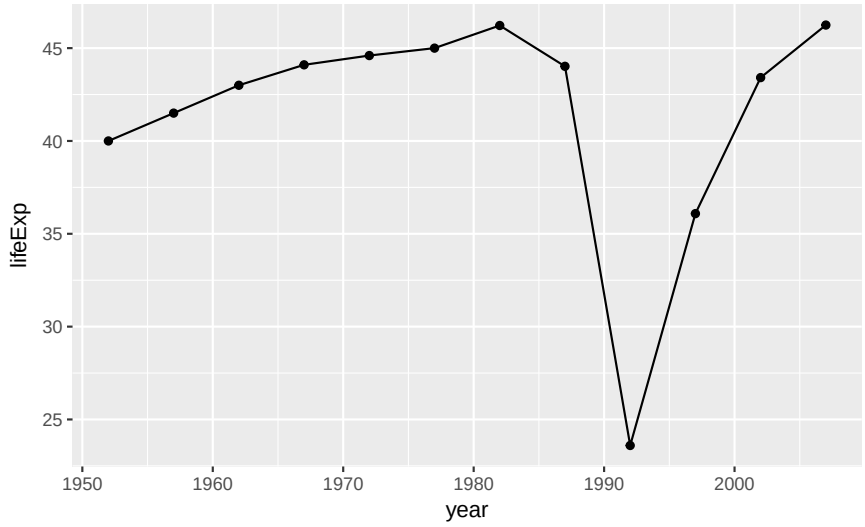
```
canada <-
  gapminder %>%
  filter(country %in% c("Canada")) %>%
  select(year,lifeExp)
canada
#> # A tibble: 12 x 2
#>    year lifeExp
#>   <int>   <dbl>
#> 1  1952    68.8
#> 2  1957    70.0
#> 3  1962    71.3
#> 4  1967    72.1
#> 5  1972    72.9
#> 6  1977    74.2
#> # ... with 6 more rows
```

```
ggplot(data = canada, aes(x = year, y = lifeExp))+
  geom_line()+
  geom_point()
```



Le génocide Rwandais...

```
gapminder %>%
  filter(country %in% c("Rwanda")) %>%
  select(year,lifeExp) %>%
  ggplot(aes(x = year, y = lifeExp))+
    geom_line()+
    geom_point()
```



Chapitre 18

Les données construites

18.1 Mise en place

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

Partie VI

L'analyse de lien

Chapitre 19

La corrélation linéaire

La corrélation est l'étude de l'intensité du lien entre deux variables. Elle permet de quantifier la relation entre deux variables **quantitatives**.

Bien qu'à priori le lien de dépendance ne soit pas toujours évident entre ces deux variables, il est pratique, à des fins d'analyse, de définir la variable indépendante X et la variable dépendante Y.

19.1 Mise en place

Nous utiliserons les librairies suivantes pour ce chapitre.

```
library(tidyverse)
library(datasauRus)
library(knitr)
```

Dans ce chapitre, nous utiliserons la base de données `starwars` qui est chargée par l'extension `dplyr` du `tidyverse`. Les informations présentes dans cette base de données sont:

variable	description
name	Name of the character
height	Height (cm)
mass	Weight (kg)
hair_color,skin_color,eye_color	Hair, skin, and eye colors
birth_year	Year born (BBY = Before Battle of Yavin)
gender	male, female, hermaphrodite, or none.
homeworld	Name of homeworld
species	Name of species
films	List of films the character appeared in
vehicles	List of vehicles the character has piloted
starships	List of starships the character has piloted

Nous pouvons avoir un meilleur apercu des différentes variables de la base de données en utilisant la commande `glimpse`.

```
glimpse(starwars)
#> Observations: 87
#> Variables: 13
#> $ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", ...
#> $ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188...
```

```
#> $ mass       <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 8...
#> $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "b...
#> $ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "l...
#> $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue",...
#> $ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0...
#> $ gender     <chr> "male", NA, NA, "male", "female", "male", "female",...
#> $ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alder...
#> $ species    <chr> "Human", "Droid", "Droid", "Human", "Human", "Human...
#> $ films      <list> [<"Revenge of the Sith", "Return of the Jedi", "Th...
#> $ vehicles   <list> [<"Snowspeeder", "Imperial Speeder Bike">, <>, <>,...
#> $ starships  <list> [<"X-wing", "Imperial shuttle">, <>, <>, "TIE Adva...
```

19.2 Le nuage de points

Dans le cours de méthodes quantitatives, nous nous intéressons à la corrélation linéaire. Un nuage de points permet de visualiser les données.

Pour construire ce graphique:

- Le titre doit être indiqué : “Lien entre variable 1 et variable 2”.
- La variable indépendante est placée sur l’axe des x et la variable dépendante est placée sur l’axe des y.
- Les titres des axes doivent être présents.
- Un point doit être placé à la coordonnée (x_i, y_i) pour chacun des couples de données.
- Les axes x et y peuvent être coupés pour améliorer la présentation.

Nous allons débuter par représenter les deux variables étudiées:. Puisque la base de données contient 87, nous n’afficherons que les premières données.

height	mass
172	77
167	75
96	32
202	136
150	49
178	120
...	...

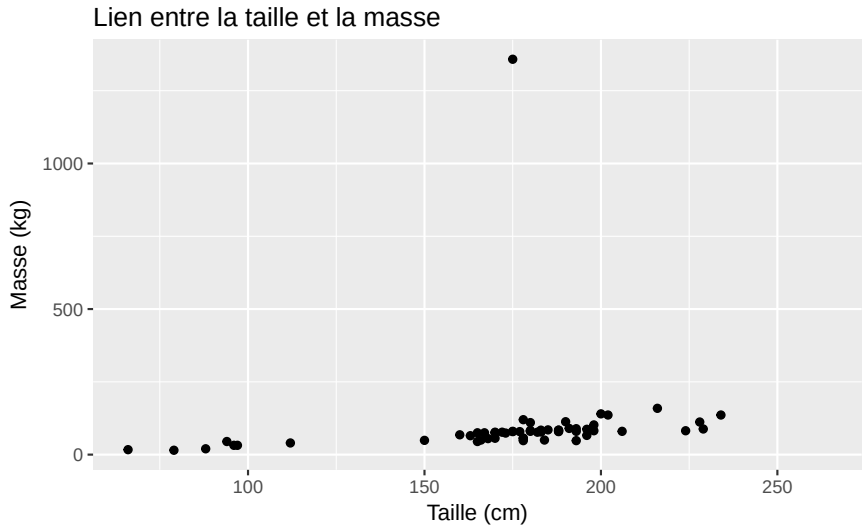
Nous allons nous intéresser aux variables `height` et `mass` de la base de données `starwars`. Puisque la masse (`mass`) dépend de la taille (`height`), la variable `mass` est la variable dépendante et la variable `height` est la variable indépendante.

Nous affichons ces données dans un plan cartésien et chaque ligne du tableau précédent correspond aux coordonnées cartésiennes de nos points. Par exemple, puisque le premier personnage à une taille de 172 et un poids de 77, nous allons afficher le point (172,77). Si nous le faisons pour toutes les données, nous obtenons un nuage de points.

Nous avons donc:

```
ggplot(data = starwars, aes(x = height, y = mass))+
  geom_point()+
  labs(
    x = "Taille (cm)",
    y = "Masse (kg)",
    title = "Lien entre la taille et la masse"
```

```
  )
#> Warning: Removed 28 rows containing missing values (geom_point).
```



Nous remarquons une donnée qui semble aberrante dans le graphique précédent. Le personnage de Star Wars semble avoir une masse très importante par rapport à sa taille. Puisque sa masse dépasse 1000kg, nous allons filtrer les données pour trouver le nom du personnage en question.

```
starwars %>%
  filter(mass > 1000)
#> # A tibble: 1 x 13
#>   name     height  mass hair_color skin_color  eye_color birth_year gender
#>   <chr>     <int> <dbl> <chr>      <chr>       <chr>           <dbl> <chr>
#> 1 Jabba D~    175 1358. <NA>       green-tan,~ orange          600. herma~
#> # ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
#> #   vehicles <list>, starships <list>
```

Le personnage est bien sûr Jabba Desilijic Tiure.

Il est possible de quantifier la force de la corrélation linéaire entre deux variables à l'aide d'une mesure. Cette mesure est appelée le coefficient de corrélation et est noté r.

La formule pour calculer r est la suivante:

$$r = \frac{\sum xy - n\bar{x}\bar{y}}{(n-1)s_x s_y}$$

Nous pouvons utiliser la commande `cor` dans R pour trouver le coefficient de corrélation. Par exemple, si nous voulons trouver le coefficient de corrélation entre les variables `height` et `mass`:

```
cor(starwars$height,starwars$mass)
#> [1] NA
```

Nous obtenons comme réponse `NA`. Ceci signifie que des données sont manquantes dans nos observations. Pour calculer un coefficient de corrélation en omettant les données manquantes, nous pouvons utiliser l'option `use = "complete.obs"`. Nous obtenons donc:

```
cor(starwars$height, starwars$mass, use = "complete.obs")
#> [1] 0.134
```

Figure 19.1: Jabba

Le coefficient de corrélation est donc de 0.134.

Nous pouvons calculer le coefficient de corrélation lorsque nous enlevons l'observation de Jabba Desilijic Tiure. Nous obtenons:

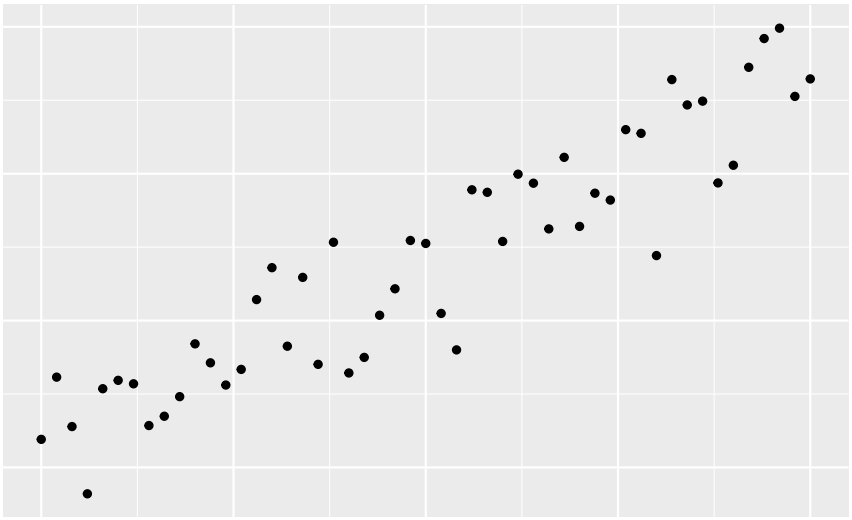
```
no_jabba <- starwars %>%
  filter(mass < 1000)
cor(no_jabba$height, no_jabba$mass)
#> [1] 0.761
```

Le coefficient de corrélation est maintenant de 0.761.

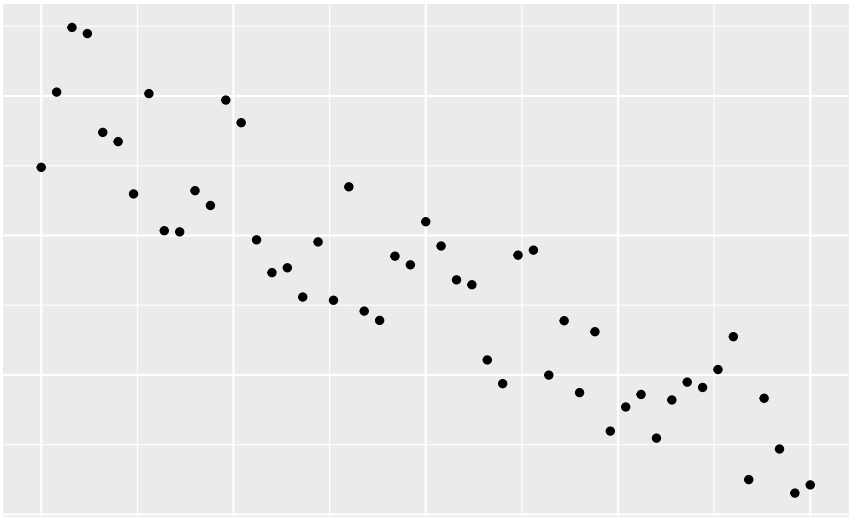
La valeur de r permet de quantifier la force de la corrélation entre X et Y et permet aussi de déterminer si cette corrélation est positive ou négative.

19.3 Fake data

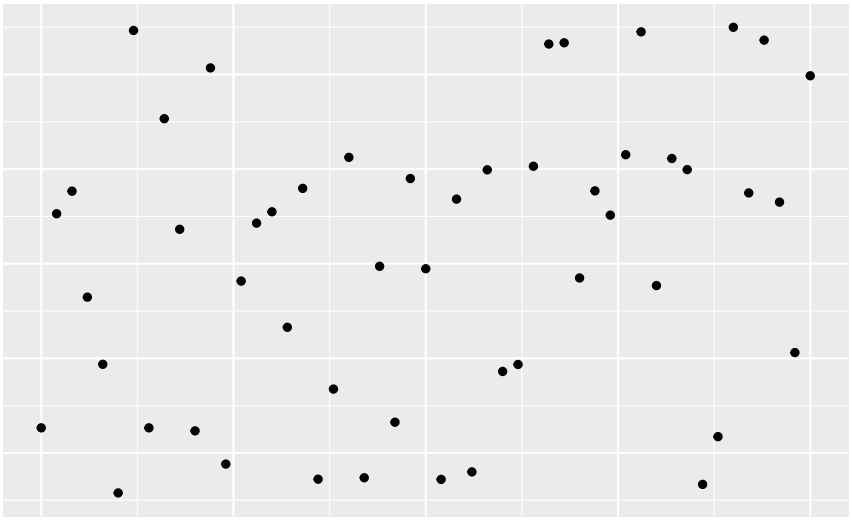
19.3.1 Corrélation positive



19.3.2 Corrélation négative



19.3.3 Aucune corrélation



19.4 Le quartet d'Anscombe

Le quartet d'Anscombe est constitué de quatre ensembles de données qui ont les mêmes propriétés statistiques simples mais qui sont en réalité très différents, ce qui se voit facilement lorsqu'on les représente sous forme de graphiques. Ils ont été construits en 1973 par le statisticien Francis Anscombe dans le but de démontrer l'importance de tracer des graphiques avant d'analyser des données, car cela permet notamment d'estimer l'incidence des données aberrantes sur les différentes indices statistiques que l'on pourrait calculer.

Le quartet d'Anscombe est disponible dans R sous le nom `anscombe`.

```
kable(anscombe)
```

x1	x2	x3	x4	y1	y2	y3	y4
10	10	10	8	8.04	9.14	7.46	6.58
8	8	8	8	6.95	8.14	6.77	5.76
13	13	13	8	7.58	8.74	12.74	7.71
9	9	9	8	8.81	8.77	7.11	8.84
11	11	11	8	8.33	9.26	7.81	8.47
14	14	14	8	9.96	8.10	8.84	7.04
6	6	6	8	7.24	6.13	6.08	5.25
4	4	4	19	4.26	3.10	5.39	12.50
12	12	12	8	10.84	9.13	8.15	5.56
7	7	7	8	4.82	7.26	6.42	7.91
5	5	5	8	5.68	4.74	5.73	6.89

Les observations x_i sont reliées aux observations y_i. Pour visualiser ces quatre ensembles de données, nous avons produit une nouvelle base de données `anscombe_tidy` sous la forme de *tidy data*.

Avant d'afficher les ensembles de données, nous allons calculer quelques mesures sur chacun de ces ensembles, à savoir, la moyenne des x, la moyenne des y, la variance des x, la variance des y et le coefficient de corrélation.

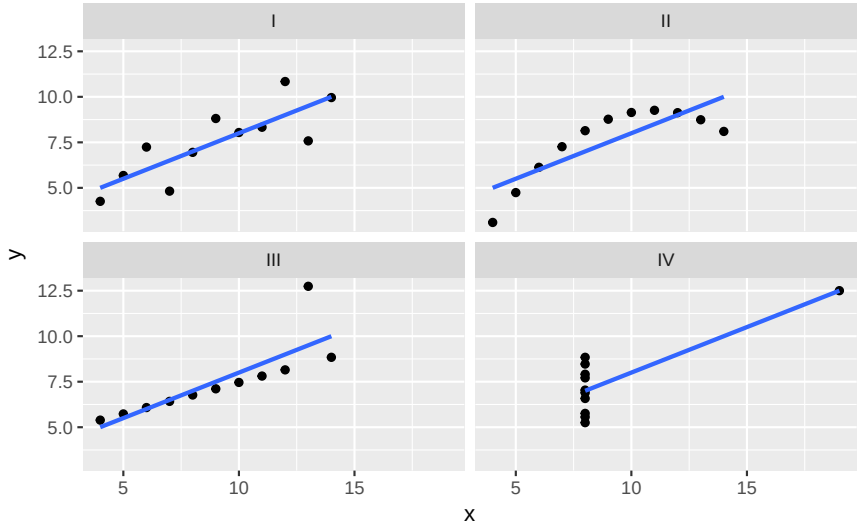
```
anscombe_tidy %>%
  group_by(ensemble) %>%
  summarise("moyenne des $x$"=mean(x),
            "variance des $x$"=var(x),
```

```
            "moyenne des $y$"=mean(y),
            "variance des $y$"=var(y),
            "coeff. de corrélation"=cor(x,y)) %>%
  kable
```

ensemble	moyenne des x	variance des x	moyenne des y	variance des y	coeff. de corrélation
I	9	11	7.5	4.13	0.816
II	9	11	7.5	4.13	0.816
III	9	11	7.5	4.12	0.816
IV	9	11	7.5	4.12	0.817

Comme nous pouvons le remarquer, les quatre ensembles de données possèdent les mêmes mesures. Par contre, lorsque nous affichons ensuite les quatre ensembles de données, nous remarquons que ces ensembles sont très différents.

```
ggplot(anscombe_tidy, aes(x, y)) +
    geom_point() +
    facet_wrap(~ ensemble) +
    geom_smooth(method = "lm", se = FALSE)
```

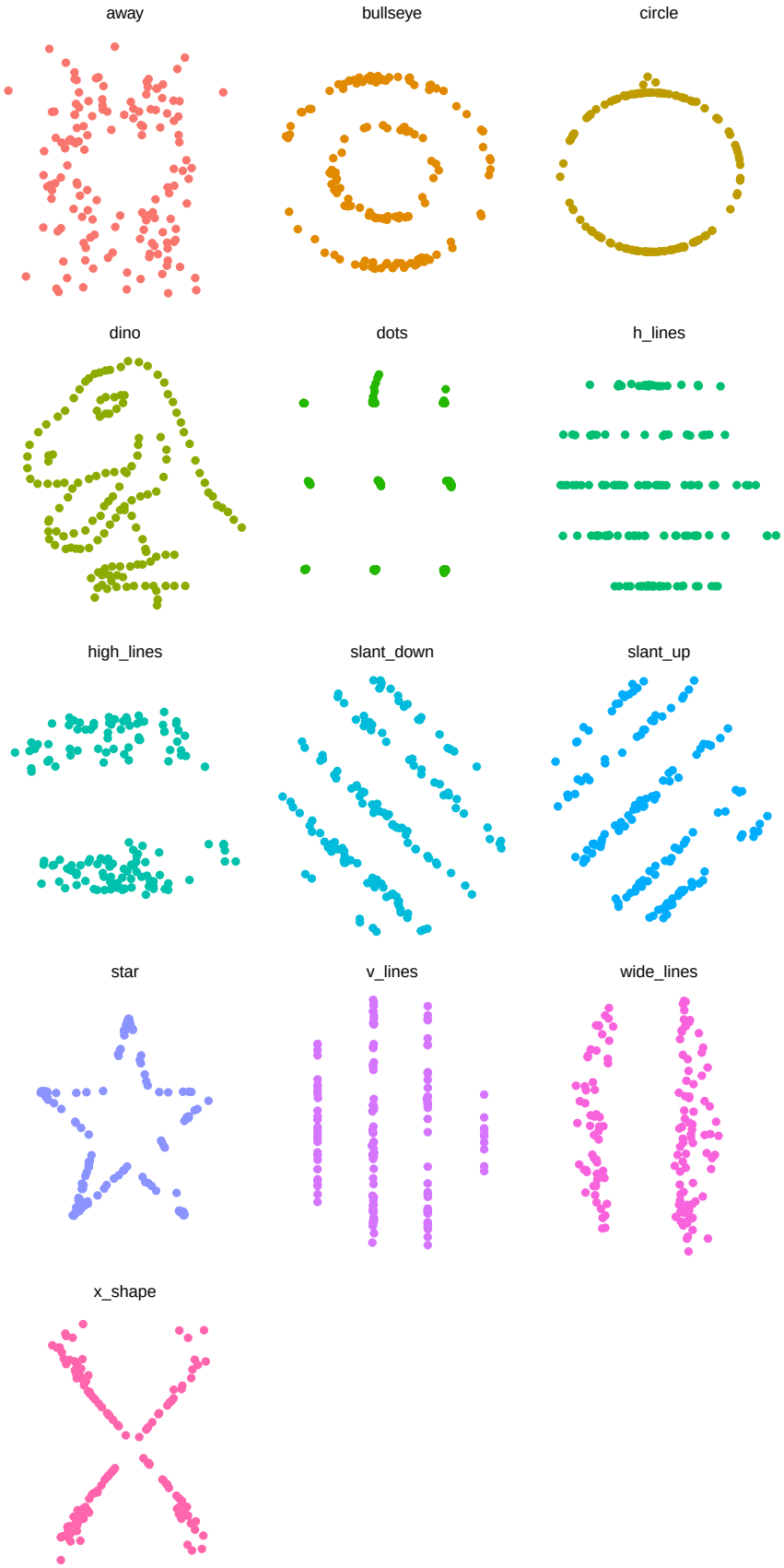


19.5 DatasauRus

```
datasaurus_dozen %>%
  group_by(dataset) %>%
  summarise("moyenne des $x$"=mean(x),
            "variance des $x$"=var(x),
            "moyenne des $y$"=mean(y),
            "variance des $y$"=var(y),
            "coeff. de corrélation"=cor(x,y)) %>%
  kable
```

dataset	moyenne des x	variance des x	moyenne des y	variance des y	coeff. de corrélation
away	54.3	281	47.8	726	-0.064
bullseye	54.3	281	47.8	726	-0.069
circle	54.3	281	47.8	725	-0.068
dino	54.3	281	47.8	726	-0.064
dots	54.3	281	47.8	725	-0.060
h_lines	54.3	281	47.8	726	-0.062
high_lines	54.3	281	47.8	726	-0.069
slant_down	54.3	281	47.8	726	-0.069
slant_up	54.3	281	47.8	726	-0.069
star	54.3	281	47.8	725	-0.063
v_lines	54.3	281	47.8	726	-0.069
wide_lines	54.3	281	47.8	726	-0.067
x_shape	54.3	281	47.8	725	-0.066

```
ggplot(datasaurus_dozen, aes(x=x, y=y, colour=dataset))+
  geom_point()+
  theme_void()+
  theme(legend.position = "none")+
  facet_wrap(~dataset, ncol=3)
```


away
bullseye
circle
dino
dots
h_lines
high_lines
slant_down
slant_up
star
v_lines
wide_lines
x_shape

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 Grolemund, G. (2017). *R for Data Science*. O'Reilly Media Inc., 1st edition. ISBN 978-1491910399.