Analyser des données avec R

Pierre-Yves de Müllenheim

2023-11-02

Sommaire

Liste des tableaux

Liste des figures

Introduction

Ce livre vise à servir de support à des enseignements auprès d'étudiants qui découvrent les analyses statistiques en même temps que le langage de programmation R. Ce livre n'est donc pas dédié exclusivement aux statistiques, ni exclusivement dédié à R. D'autres ouvrages proposent des contenus plus spécialisés, et donc plus poussés, et c'est vers ces ouvrages qu'il convient d'aller pour davantage maîtriser les fondamentaux des statistiques ou toutes les subtilités du langage R.

Bien que le langage R puisse être utilisé via le logiciel R en tant que tel, les procédures décrites dans ce livre supposent que l'utilisation du logiciel R s'effectue en réalité à l'aide de l'environnement proposé par le logiciel RStudio, qui est plus confortable en matière d'utilisation que l'interface initialement associée au logiciel R. Que cela soit via l'interface de base associée au logiciel R ou via l'environnement proposé par RStudio, l'utilisation du langage R pour obtenir le résultat d'une analyse nécessite d'écrire des lignes de code. Si cela peut s'avérer plus complexe et/ou fastidieux à utiliser au départ qu'un logiciel classique où il suffit de cliquer sur des boutons pour obtenir le résultat de l'analyse, cela vaut la peine de prendre le temps d'apprendre à programmer avec R (et préférentiellement via RStudio), au moins pour les raisons suivantes : les logiciels R et RStudio sont en accès libre sur internet ; il existe de nombreuses aides documentaires, notamment sur internet, qui permettent de programmer n'importe quelle analyse ou manipulation de données, aussi sophistiquée soit elle ; il est possible de conserver les lignes de code pour pouvoir refaire plus tard les analyses, ou pour pouvoir appliquer ces lignes de code à d'autres données, ou encore pour partager ces lignes de code avec d'autres personnes. De par sa gratuité et sa capacité à permettre le partage des analyses réalisées, R et RStudio sont ainsi une très bonne option pour embrasser la tendance actuelle, encore timide, de l'*open science*, consistant à permettre à tout un chacun de savoir ce qui a été fait en matière d'analyses et à pouvoir reproduire ces analyses. Enfin, le langage R, notamment via l'utilisation de RStudio, permet de faire bien plus que des analyses de données (e.g., automatisation d'analyses, développement d'applications web, construction de sites internet et de curriculum vitae, etc.), ce qui en fait un outil de travail polyvalent et donc particulièrement intéressant.

L'appropriation du contenu de ce livre suppose une lecture relativement linéaire des chapitres, du moins en vue de comprendre comment utiliser le langage R. Plusieurs exemples de code, en particulier pour la conception de graphiques, utilisent ce qu'on appelle des *packages* qui sont à télécharger et à installer sur l'ordinateur, puis à charger lorsque le logiciel est ouvert. Une fois que les consignes d'installation et de chargement de ces packages ont été explicitement données au cours d'un chapitre, leur utilisation dans le chapitre en cours suppose implicitement que ces packages ont été chargés et sont donc prêts à être utilisés.

Dans ce livre, les exemples de code apparaissent dans des zones grisées, et les résultats qui pourraient apparaître à l'écran dans RStudio une fois le code activé sont précédés d'un double dièse (##). Par moment, les noms des fonctions sont écrits à la suite des noms des packages dont elles dépendent, comme ceci : `package::fonction()`.

Une version PDF de ce livre est disponible en ligne ici.[1]

[1] https://pydemull.github.io/Analyser-des-donnees-avec-R/Analyser-des-données-avec-R.pdf

partie I

Appropriation de R et RStudio

Chapitre 1

Prérequis

1.1 Installation de R et RStudio

R et RStudio sont deux logiciels en libre accès sur internet. Le logiciel R peut être utilisé indépendamment du logiciel RStudio. En revanche, l'utilisation du logiciel RStudio requiert au préalable l'installation du logiciel R. En effet, RStudio est un logiciel qui permet d'utiliser les fonctionnalités de R tout en proposant une interface d'utilisation plus agréable et fonctionnelle que l'interface à l'origine proposée pour le logiciel R, qui est très basique. Les explications présentées au cours de ce document considèrent que l'utilisateur fonctionne avec RStudio.

1.1.1 Installer R sur WINDOWS

- Aller sur le site https://cran.rstudio.com.
- Sur la page web qui s'affiche, l'encart du haut **Download and Install R** montre les différents liens de téléchargement possibles selon le système d'exploitation utilisé. Cliquer sur le lien **Download R for Windows**.
- Dans la nouvelle fenêtre qui vient de s'ouvrir, cliquer sur **install R for the first time**.
- Dans la nouvelle fenêtre qui vient de s'ouvrir, cliquer sur le premier lien en haut de la page : **Download R-X.X.X. for Windows**. Exécuter le fichier s'il est proposé de le faire. Si ce n'est pas le cas, il est probable que le téléchargement du fichier ait été lancé automatiquement. Retrouver le fichier ainsi téléchargé sur le PC (une fois son enregistrement terminé), puis exécuter le fichier.
- Suivre la procédure d'installation par défaut en cliquant à chaque fois sur **Suivant**.
- Une fois l'installation terminée, double-cliquer sur l'icône du bureau (**R x64 X.X.X**) pour vérifier que l'ouverture du logiciel R s'effectue correctement.

1.1.2 Installer R sur MAC

- Aller sur le site https://cran.rstudio.com.
- Sur la page web qui s'affiche, l'encart du haut **Download and Install R** montre les différents liens de téléchargement possibles selon le système d'exploitation utilisé. Cliquer sur le lien **Download R for macOS**.
- Dans la nouvelle fenêtre qui s'ouvre, cliquer sur le lien qui correspond à votre version OS (le clic entraînera le début du téléchargement du fichier).
- Sur le Mac, chercher dans le dossier **Téléchargements** le fichier téléchargé.
- Double-cliquer sur le fichier téléchargé pour lancer l'installation du logiciel R.
- Suivre la procédure par défaut et terminer l'installation.

- Lorsque l'installation est terminée, aller dans le dossier **Applications** du Mac pour rechercher le logiciel R. Double-cliquer sur l'icône pour ouvrir le logiciel et vérifier que l'ouverture se déroule correctement.

1.1.3 Installer RStudio sur WINDOWS ou MAC

- Aller sur le site https://posit.co/.
- En haut à droite de la page d'accueil, cliquer sur le bouton **DOWNLOAD RSTUDIO**.
- Dans la nouvelle fenêtre qui vient de s'ouvrir, faire défiler la page vers le bas jusqu'à aller dans la section RStudio Desktop puis cliquer sur le bouton **DOWNLOAD RSTUDIO**.
- Dans la nouvelle fenêtre qui vient de s'ouvrir, cliquer sur le bouton **DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS** si vous avez une machine Windows. Si vous avez un MAC et que le bouton n'est pas mis à jour automatiquement pour votre MAC, alors il faut faire défiler un peu la page vers le bas pour ensuite cliquer sur le lien de téléchargement de RStudio adapté à votre version de système d'exploitation.
- **Sur WINDOWS :** Exécuter le fichier s'il est proposé de le faire. Si le clic entraîne automatiquement le téléchargement du fichier, retrouver alors ce fichier sur le PC (une fois son enregistrement terminé), puis exécuter le fichier et suivre la procédure d'installation par défaut. Une fois le logiciel installé, retrouver le fichier d'exécution du logiciel sur le PC (chemin d'accès possible : **Ordinateur > Windows (C:) > Programmes > RStudio > bin > rstudio.exe**). Créer un raccourci pour le fichier **rstudio.exe** (clic droit sur le fichier > Créer un raccourci) et mettre le raccourci sur le bureau du PC. Double-cliquer sur l'icône RStudio afin de vérifier si l'ouverture s'effectue correctement.
- **Sur MAC :** Le téléchargement débute lorsque vous cliquez sur le lien. Une fois le téléchargement terminé, sur votre MAC, réaliser l'installation en double-cliquant sur le fichier téléchargé et en suivant la procédure indiquée. Une fois l'installation terminée, aller dans le dossier **Applications** du MAC et double-cliquer sur l'icône de RStudio pour vérifier si le logiciel s'ouvre correctement.

1.2 Prise en main de RStudio

1.2.1 Fonctionnement général

Basiquement, RStudio s'utilise de la manière suivante : on écrit une instruction (i.e., une ligne de code) dans une fenêtre, on lance cette instruction, et le logiciel nous donne le résultat, qu'il s'agisse d'un calcul, d'un graphique, d'une modification d'un jeu de données, etc. Quand on ouvre RStudio pour la première fois, la fenêtre principale qui se présente est la **Console**. Cette fenêtre permet d'y écrire des lignes de code et de les lancer en appuyant sur la touche **Entrée**. Lorsque l'on souhaite conserver les lignes de code que l'on a écrites, ou que l'on souhaite écrire des lignes de code sans forcément les lancer, il est possible d'utiliser une fenêtre **Script** (chemin d'accès : **File > New File > R Script**). Pour lancer les lignes de code qui sont écrites dans une fenêtre de script, il suffit de se placer n'importe où sur la ligne de code et de cliquer sur l'icône **Run** du logiciel (raccourci : **Ctrl + Entrée**). Une fois le code activé, celui-ci est montré dans la Console, et selon la nature de la commande, le résultat peut apparaître lui aussi dans la Console ou dans l'un des encarts disposés par défaut à droite de l'écran, selon qu'il s'agisse notamment d'un nouvel objet (e.g., une variable), d'une demande d'aide, ou de la création d'un graphique.

1.2.2 Manipuler des objets (valeurs, vecteurs, et tableaux de données)

R permet tout d'abord d'effectuer des opérations simples avec des nombres, telles que des additions avec le symbole `+`, des soustractions avec le symbole `-`, des multiplications avec le symbole `*`, des divisions avec le symbole `/`, des racines carrées avec la fonction `sqrt()`, ou encore des élévations à la puissance avec le symbole `^`.

```
(9 + 3 - 5) * 5 / 2 + sqrt(9) ^ 2
```

```
## [1] 26.5
```

De manière plus élaborée, R permet aussi de créer des vecteurs (i.e., des suites de valeurs), notamment grâce à la fonction `c()`, et de les manipuler avec différentes sortes d'opérations. Lorsqu'une opération ou une série d'opérations est appliquée à un vecteur, chaque valeur du vecteur subit les opérations spécifiées. Dans l'exemple ci-dessous, on voit par exemple que chaque valeur du vecteur a été multipliée par 2 et s'est vue ajouter la valeur 3.

```
c(0, 1, 2, 3, 4, 5) * 2 + 3
```

```
## [1]  3  5  7  9 11 13
```

Si des vecteurs peuvent contenir des nombres, ils peuvent également contenir des caractères, tels que de simples lettres ou des mots, ces vecteurs ne pouvant cependant pas, par nature, subir des opérations mathématiques. Pour pouvoir être crées, les caractères doivent être écrits à l'intérieur du vecteur avec des guillemets (`" "`).

```
c("a", "b", "c")
```

```
## [1] "a" "b" "c"
```

```
c("Pierre", "Marie", "Jean")
```

```
## [1] "Pierre" "Marie"  "Jean"
```

De manière encore plus élaborée, R permet de créer des tableaux de données à partir de vecteurs à l'aide de la fonction `data.frame()`. Dans l'exemple ci-dessous, les noms x, y, et z, marqués à gauche du signe `=`, sont les noms des vecteurs que contiendra le tableau de données. À droite du signe `=`, on retrouve la fonction `c()` qui permet de créer un vecteur avec des valeurs à l'intérieur.

```
data.frame(
  x = c(0, 1, 2, 3),
  y = c(3, 5, 7, 9),
  z = c("a", "b", "c", "d")
)
```

```
##   x y z
## 1 0 3 a
## 2 1 5 b
## 3 2 7 c
## 4 3 9 d
```

1.2.3 Manipuler des objets via des noms

L'une des particularités de R, c'est de permettre d'associer des objets (e.g., des valeurs, des vecteurs, ou encore des tableaux de données) à des noms. Pour ce faire, R utilise la fonction d'assignation `<-`. Cette fonction s'utilise en écrivant à droite de la flèche l'objet à créer (ou qui est déjà crée), et en écrivant à gauche de la flèche le nom auquel on veut que l'objet soit associé (Attention : Toujours utiliser seulement des caractères alphanumériques et des points `.` ou des tirets du bas `_` pour écrire un nom ; ne pas commencer par un chiffre ; avoir à l'esprit que R est sensible à la casse, ce qui veut dire qu'un nom commençant par une majuscule sera un nom différent de celui qui a les mêmes lettres mais qui commence par une minuscule.) L'utilisation de noms associés à des objets permet de grandement faciliter les analyses par la suite. Lorsqu'on réalise une assignation, il est possible de voir le nouveau nom et l'objet associé dans la fenêtre **Environnement** de RStudio. Lorsqu'on lance le code permettant d'assigner un objet à un nom, R ne montre pas le contenu de l'objet. Pour le voir, il faut écrire le nom associé à l'objet dans une ligne de code et lancer la commande.

Il est donc possible d'associer à un nom un objet qui serait une valeur numérique…

```
a <- 9
a
```

```
## [1] 9
```

… ou encore une succession de caractères …

```
Prenom <- "Pierre"
Prenom
```

```
## [1] "Pierre"
```

… ou encore un vecteur …

```
Taille <- c(178, 191, 178, 182, 167, 151)
Taille
```

```
## [1] 178 191 178 182 167 151
```

```
Poids <- c(60, 89, 92, 67, 80, 70)
Poids
```

```
## [1] 60 89 92 67 80 70
```

```
Sexe <- c("M", "M", "F", "F", "M", "F")
Sexe
```

```
## [1] "M" "M" "F" "F" "M" "F"
```

… et même un tableau de données, qui aurait été soit conçu à la main, soit conçu à partir d'objets de type vecteurs qui auraient été crées auparavant, comme ci-dessous. À noter que les vecteurs doivent contenir le même nombre de valeurs pour pouvoir être combinés dans un tableau de données avec la fonction `data.frame()`.

```
df <- data.frame(Taille, Poids, Sexe)
df
```

```
##   Taille Poids Sexe
## 1    178    60    M
## 2    191    89    M
## 3    178    92    F
## 4    182    67    F
## 5    167    80    M
## 6    151    70    F
```

Lorsqu'un objet de type tableau de données est assigné à un nom, il est possible d'afficher le contenu d'une seule colonne de ce tableau à partir du nom associé au tableau, du symbole `$`, et du titre de la colonne désirée.

```
df$Taille
```

```
## [1] 178 191 178 182 167 151
```

Une fois que des objets sont liés à des noms, il est possible, comme montré initialement avec des valeurs, d'utiliser ces noms pour réaliser des opérations. Par exemple, via des noms, on peut manipuler des objets contenant simplement une valeur numérique…

```
a <- 7
b <- 3
c <- 2
(a + b) / c
```

```
## [1] 5
```

ou alors des objets contenant un vecteur …

```
vec1 <- c(0, 2, 4, 6, 8)
vec2 <- c(1, 4, 5, 9, 0)
vec1 * 10
```

```
## [1]  0 20 40 60 80
```

```
vec1 * vec2
```

```
## [1]  0  8 20 54  0
```

ou encore des objets contenant un tableau de données, en créant par exemple une variable à partir d'autres variables du tableau.

```
df$IMC <- df$Poids / (df$Taille / 100) ^ 2
df
```

```
##   Taille Poids Sexe      IMC
## 1    178    60    M 18.93700
## 2    191    89    M 24.39626
## 3    178    92    F 29.03674
## 4    182    67    F 20.22703
## 5    167    80    M 28.68514
## 6    151    70    F 30.70041
```

Lorsque plusieurs objets ont été assignés à des noms, il est possible de vouloir supprimer certaines assignations, par exemple en raison du fait qu'un objet aurait été assigné par erreur. Pour supprimer une assignation, il est possible d'utiliser la fonction `rm()`.

```
rm(vec1)
```

L'instruction `rm(list = ls())` supprime toutes les assignations qui ont été réalisées auparavant.

1.2.4 Utiliser des fonctions

Dans les exemples de code précédents, nous avons utilisé plusieurs fonctions : la fonction `sqrt()`, la fonction `c()`, la fonction `data.frame()`, et la fonction `rm()`. Par la suite, nous serons amenés à voir comment l'on crée une fonction et comment l'on arrive finalement à n'avoir qu'une expression suivie de parenthèses à utiliser pour faire un ensemble d'actions automatiquement. Mais avant cela, il est important de savoir globalement comment une fonction s'utilise. Cela est important car il est rapidement possible de se rendre compte qu'utiliser R, c'est utiliser des fonctions. De plus, lorsque l'on souhaite réaliser une nouvelle analyse avec une fonction que l'on n'a jamais utilisée auparavant, il est nécessaire de pouvoir en comprendre la structure et d'être en mesure d'en comprendre le fonctionnement pour pouvoir l'utiliser.

Pour expliquer comment s'utilise une fonction, commençons directement par un exemple, cette fois avec la fonction `plot()` :

```
plot(x = iris$Sepal.Length, y = iris$Petal.Length)
```

Comme nous pouvons l'observer ci-dessus, pour utiliser une fonction, il faut d'abord écrire son nom, puis mettre des parenthèses pour qu'on puisse écrire des informations à l'intérieur. Ces informations, elles sont de deux natures. D'un côté il y a les **arguments** (qui sont `x` et `y` dans l'exemple ci-dessus), et d'un autre côté il y a les **valeurs** (qui sont les variables `Sepal.Length` et `Petal.Length` du jeu de données `iris` dans l'exemple ci-dessus). Notons ici que le concept de valeur est à prendre au sens général du terme. Dans ce cadre là, une valeur pourrait tout aussi bien désigner des nombres ou des lettres, des vecteurs, des jeux de données, etc. Avec certaines fonctions, nous aurions pu mettre le nom de la variable seul et mettre le nom du jeu de données en face d'un autre argument, mais cela n'était pas possible ici. Comme nous pouvons le voir également, l'argument et la valeur sont toujours mis en lien par le biais du signe `=`. (Attention : Nous verrons plus tard qu'avec certaines fonctions, l'écriture qui est à gauche du signe `=` est en fait le nom d'une nouvelle variable à créer, mais laissons cela de côté pour le moment.) Le nombre d'arguments dépend des fonctions. Certaines n'en n'ont qu'un, d'autres peuvent en avoir un très grand nombre. Dans une fonction, certains arguments doivent obligatoirement recevoir une valeur indiquée par nos soins, alors que d'autres arguments ne seront tout simplement pas utilisés si on ne leur associe pas une valeur particulière. Enfin, certains arguments prendront une valeur par défaut associée à la fonction si l'on écrit rien les concernant dans la fonction. Dans la fonction `plot()`, seul l'argument `x` doit obligatoirement recevoir une valeur pour que la fonction puisse être utilisée. Dans notre exemple, le fait d'avoir en plus associé une valeur à

l'argument `y` permet à la fonction de non pas montrer uniquement les données de la variable `x`, mais de réaliser un graphique en montrant les données de `y` en fonction de `x`, comme illustré sur la Figure 1.1 :

```
plot(x = iris$Sepal.Length, y = iris$Petal.Length)
```

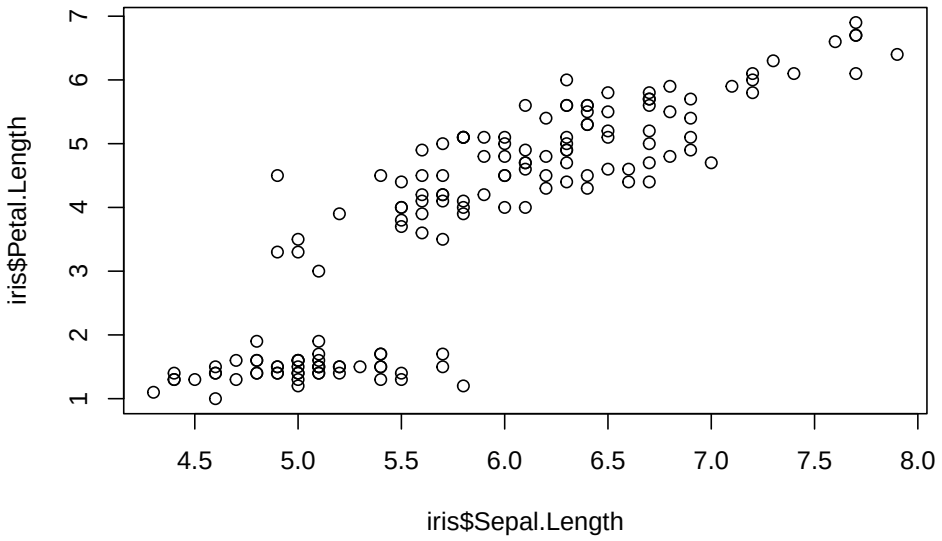

Figure 1.1: Graphique obtenu avec un paramétrage explicite de la fonction plot()

Lorsque les arguments sont explicitement précisés comme pour `x` et `y` de l'exemple ci-dessus, il est en réalité possible de les écrire dans l'ordre que l'on veut. Nous aurions par exemple très bien pu écrire les choses de la manière suivante sans que cela ne change rien au résultat de la commande :

```
plot(y = iris$Petal.Length, x = iris$Sepal.Length)
```

Ce changement d'ordre n'est possible que lorsque les arguments sont explicitement précisés dans la fonction. Il est aussi possible de configurer une fonction en mettant des valeurs sans avoir à écrire les noms des arguments. Cependant, lorsque les noms des arguments ne sont pas précisés, R associe les valeurs (celles que l'on a mises) aux arguments de la fonction en suivant l'ordre par défaut des arguments avec lequel la fonction a été configurée. Ainsi, si nous voulons avoir, pour le graphique associé à notre exemple, la variable `Sepal.Length` en `x`, et la variable `Petal.Length` en `y`, on peut très bien écrire la fonction comme ceci :

```
plot(iris$Sepal.Length, iris$Petal.Length)
```

En revanche, si nous avions inversé l'ordre d'écriture des variables dans la fonction sans préciser les noms des arguments (cf. code ci-dessous), nous aurions eu un résultat différent du graphique précédent (cf. Figure 1.2, où les variables en x et en y ont été inversées par rapport au graphique précédent).

```
plot(iris$Petal.Length, iris$Sepal.Length)
```

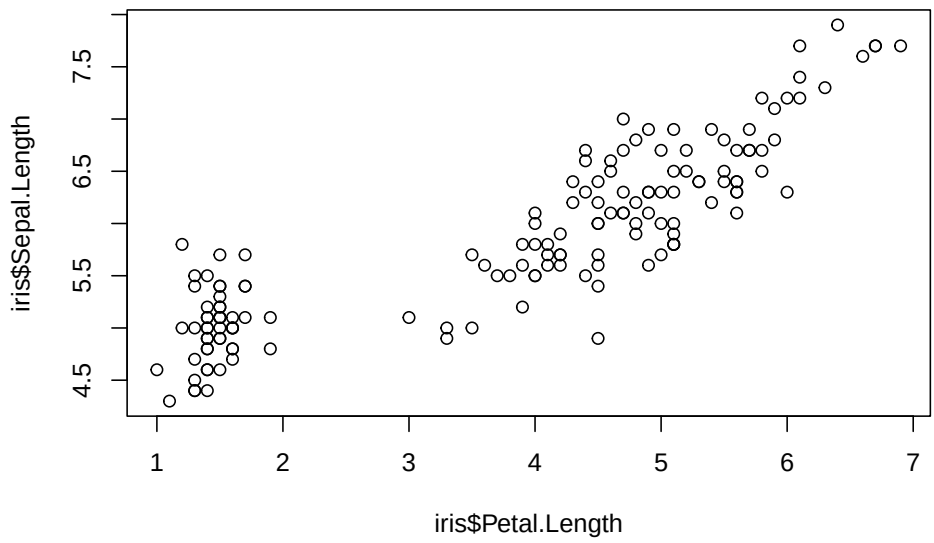

Figure 1.2: Graphique obtenu avec un paramétrage implicite de la fonction plot()

Dans la suite de ce livre, les arguments configurés par nos soins seront en général précisés lorsque nous leur donnerons une valeur. Cela étant dit, beaucoup d'arguments seront aussi régulièrement laissés de côté lorsque cela ne sera pas nécessaire de les préciser pour l'exemple.

Au regard de ce qu'il vient d'être expliqué, il est donc une bonne pratique, avant d'utiliser une fonction, de connaître les arguments qu'elle contient, non seulement pour savoir comment configurer ses arguments, mais aussi pour savoir ce que font les arguments de la fonction lorsque l'on ne touche pas à leur configuration par défaut. En général, toute fonction utilisable et opérationnelle avec R dispose d'une aide directement accessible via RStudio. Pour consulter l'aide associée à une fonction, il suffit d'écrire dans la Console le signe `?` suivi du nom de la fonction qui pose question, comme ci-dessous :

```
?plot
```

Toutefois, paradoxalement, l'aide n'est pas toujours facile à comprendre pour ceux qui n'ont pas un niveau d'expertise avancé avec R. Pour bien comprendre comment on peut utiliser une fonction, internet peut être une excellente ressource car il regorge de sites et d'exemples crées par la communauté R. L'un des sites sur lesquels on tombe souvent lors d'une recherche internet est le site https://stackoverflow.com. Une bonne partie des problèmes de compréhension et d'utilisation des fonctions de R que l'on rencontre peuvent être résolus en consultant des exemples venant de ce site.

1.2.5 Installer et charger des *packages*

R fonctionne en bonne partie sur la base de fonctions qui permettent de réaliser automatiquement différents types de calculs. Ces fonctions sont regroupées dans des ensembles qu'on appelle des

packages. La version d'installation initialle du logiciel R dispose d'un ensemble de *packages* de base qui permettent de réaliser un très grand nombre d'analyses. Toutefois, la version de base de R impose parfois des manières d'écrire certaines instructions qui sont peu intuitives ou qui parfois ne permettent tout simplement pas de faire les analyses souhaitées. Pour palier ces problèmes, des *packages* sont régulièrement crées et actualisés par la communauté R. Pour pouvoir les utiliser, il est nécessaire de d'abord installer le *package* additionnel grâce à la fonction `install.packages()`. L'une des collections de *packages* les plus utiles pour manipuler des tableaux de données et effectuer des analyses statistiques et graphiques est celle du `tidyverse`, qui a été pensée notamment pour faciliter l'écriture des lignes de code.

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

Une fois que le *package* a été installé (ou l'ensemble de *packages* s'il s'agit d'une collection comme dans le cas du `tidyverse`), une étape supplémentaire est nécessaire pour pouvoir utiliser les fonctions qu'il contient : il faut le charger dans l'environnement R. Pour cela, il est possible d'utiliser la fonction `library()`.

```
library(tidyverse)
```

Lorsque l'on charge la collection de *packages* `tidyverse`, on peut observer dans la Console que plusieurs *packages* sont chargés en même temps : `ggplot2`, pour la visualisation de données ; `dplyr`, pour la manipulation de données ; `tidyr`, pour l'organiation des tableaux de données ; `readr`, pour l'importation de jeux de données ; `purrr`, pour la programmation ; `tibble`, pour le formatage de tableaux de données ; `stringr`, pour la gestion des chaînes de caractères ; `forcats`, pour la gestion de variables qualitatives. Si nous avons téléchargé et chargé l'ensemble des *packages* du `tidyverse`, nous aurions pu aussi installé et chargé un seul de ces *packages* à la fois, comme pour la plupart des *packages* qui existent.

1.2.6 Divers

Au fur et à mesure que l'on écrit un script, une bonne pratique consiste à régulièrement créer des sections avec des titres et d'ajouter des commentaires pour certaines analyses. Afin de ne pas rendre activable les lignes de code qui ne serviraient qu'à écrire des titres ou des commentaires, il convient d'utiliser le symbole `#` devant l'écriture du code.

```
# Titre de section 1 ----------------------------------------------
## Sous-titre 1
## Sous-titre 2
```

Étant donné que l'erreur est difficile à éviter à un moment donné ou à un autre lorsqu'on commence à écrire son propre code ou à utiliser un code qui vient de quelqu'un d'autre, il est utile de reconnaître les situations dans lesquelles une erreur est survenue. La situation la plus évidente est l'apparition d'un texte en rouge dans la Console. Lorsqu'il s'agit bel et bien d'une erreur (car il peut ne s'agir parfois que d'un message d'alerte ou d'information), le texte en rouge décrit l'erreur qui a été détectée et qui empêche le code d'être entièrement activé. De manière moins visible, il est possible parfois d'observer un `+` tout en bas de la Console. Cela survient lorsque le code lancé à l'instant est incomplet (e.g., une parenthèse a été oubliée). Si cela arrive, il vaut mieux appuyer sur **Echap**, trouver l'erreur dans le code, et relancer la commande. Avant de relancer la commande, il faut s'assurer que R donne effectivement la main pour lancer une nouvelle instruction. C'est le cas lorsque le symbole `>` est observé dans la Console. Enfin, RStudio permet d'utiliser un certain nombre de raccourcis clavier. Pour en avoir une vue d'ensemble, appuyez sur **Alt+Shift+K**.

1.3 Résumé

- R permet de faire des opérations sur des valeurs : additions, soustractions, multiplications, divisions, etc.
- R permet de faire des opérations sur des objets : valeurs uniques, vecteurs, tableaux de données, etc.
- Pour créer un vecteur, qu'il contienne des nombres, du texte, ou les deux, il est possible d'utiliser la fonction `c()`.
- Pour créer un tableau de données, il est possible d'utiliser la fonction `data.frame()`.
- Pour afficher une colonne particulière d'un tableau de données, utiliser le symbole `$`.
- Pour associer un objet à un nom, utiliser la fonction d'assignation `<-`.
- Pour supprimer une assignation, utiliser la fonction `rm()`.
- L'utilisation de R repose sur l'utilisation de fonctions. Une fonction s'utilise en écrivant son nom, suivi de parenthèses à l'intérieur desquelles on peut préciser les arguments qui nous intéressent et indiquer les valeurs nécessaires selon les besoins des analyses.
- Pour demander l'aide de R à propos d'une fonction, écrire `?` suivi du nom de la fonction.
- Pour installer un *package*, utiliser la fonction `install.packages()`.
- Un code peut être écrit directement dans la **Console**, ou dans une fenêtre **Script**. Pour activer le code à partir d'un script, utiliser le bouton **Run** ou **Ctrl + Entrée**.
- Les noms et les objets associés apparaissent dans la fenêtre **Environnement**.
- Pour écrire des titres et des commentaires dans un script, utiliser un ou plusieurs `#` avant l'écriture du code.
- Dans la Console, le symbole `>` signifie que le logiciel est prêt à lancer une nouvelle instruction. Le symbole `+` indique que l'instruction initialement lancée est incomplète. Mieux vaut alors faire **Echap**, modifier le code, et recommencer.

Chapitre 2

Importation et manipulation d'une base de données

2.1 Comprendre ce qu'est une base de données

Lorsqu'on souhaite répondre à une question, la démarche scientifique classique consiste à effectuer une série de mesures ou d'observations selon un protocole qui a été conçu en cohérence avec la question posée. En principe, ces mesures ou observations donnent lieu à l'obtention de valeurs. Ces valeurs peuvent être de forme numérique (e.g., les valeurs de taille de différents individus) ou de forme littérale (e.g., les valeurs de sexe de différents individus). Quelle que soit leur forme, les valeurs que l'on obtient dans un contexte qui est connu, comme dans le cas d'un protocole de mesures, ont un sens bien défini car elles sont associées à des choses que l'on a cherché à caractériser. Lorsqu'une valeur est porteuse d'un sens bien défini, on peut alors considérer qu'il s'agit d'une **donnée**. Très souvent, pour répondre à une question, il est nécessaire d'acquérir plusieurs données qui seraient relatives à différentes choses que l'on a cherché à caractériser (e.g., la taille, la couleur, le poids, etc.), et qui seraient relatives également à différents individus chez qui l'on aurait souhaité caractériser ces choses. Afin de conduire les analyses qui permettraient de répondre à la question posée, il convient alors de répertorier toutes les données acquises dans un même document, et plus exactement dans un même fichier, qui serait la base de données, telle que présentée dans le Tableau 2.1.

Tableau 2.1: Exemple de base de données

id	genre	taille	nb_victoires	niveau
1	H	1.80	45	1
2	H	1.93	90	3
3	H	1.50	100	4
4	F	1.95	43	1
5	F	1.52	34	2
6	H	1.87	67	2
7	H	1.83	79	3

La base de données prend donc la forme d'un tableau. Plusieurs principes sont à respecter en

général lors de la création d'une base de données. Tout d'abord, les lignes de la base de données (qu'on appelle des **observations**) doivent correspondre le cas échéant à des individus bien identifiés. Ensuite, chaque colonne doit correspondre à une variable. L'ensemble des données contenues dans une même ligne correspond donc aux données relatives aux différentes variables (e.g., la taille, le poids, le sexe, etc.) qui auraient été obtenues chez un même individu. Dans le cas d'études où l'on évaluerait une ou plusieurs variables plusieurs fois chez un même individu (i.e., à différentes moments, dans différentes conditions), il peut convenir de créer autant de lignes que de fois où les variables auraient été évaluées. Par exemple, le Tableau 2.2 représente une base de données, certes très sommaire, qui contient des données d'individus dont on aurait évalué le poids deux fois, avant et après un programme de prise en charge. On remarque alors qu'il y a deux lignes par individu qui correspondent aux deux temps d'évaluation. La taille, elle, n'a été évaluée qu'une seule fois, en début de programme, mais pour éviter de laisser des cellules vides, la valeur initiale de la taille a été reproduite dans la seconde ligne.

Tableau 2.2: Organisation d'une base de données avec des mesures répétées

id	taille	temps_eval	poids
1	1.75	pre	75
1	1.75	post	73
2	1.89	pre	90
2	1.89	post	88

En principe, les données de la base qui ont été obtenues selon la même procédure d'acquisition représentent le même type de choses. Ces choses sont appelées des **variables** car elles varient selon les individus qui ont été étudiés et les conditions de mesure qui ont été mises en oeuvre (dans le cas où il y en aurait plusieurs). Lorsque ces choses ne sont pas censées varier, on parle de **constantes**. Une base de données peut comporter des variables de types différents (Tableau 2.3).

Tableau 2.3: Les différents types de variables

Type	Continue	Discrète
Quantitative		
Intervalle	X	X
Ratio	X	X
Qualitative		
Nominale		X
Ordinale		X

Les **variables quantitatives** (aussi dites numériques) comportent des nombres dont les différences entre eux ont un sens. Parmi les variables quantitatives, il est possible de plus précisément distinguer celles qui peuvent être associées seulement à une **échelle d'intervalles**, et celles qui peuvent être aussi associées à une **échelle de ratios**. Les variables étant associables seulement à une **échelle d'intervalles** ont la particularité de ne pas avoir de zéro naturel, de sorte que multiplier ou diviser les valeurs de cette variable n'aurait pas de sens. Un exemple

de ce type de variables pourrait être la température exprimée en degrés Celsius. Avec cette variable, il est seulement possible de décrire le fait qu'il fait x degrés plus chaud ou plus froid à un endroit par rapport à un autre, mais cela n'aurait pas de sens de dire qu'il ferait deux fois plus chaud à un endroit où il y aurait 20°C par rapport à un endroit où il y aurait seulement 10°C. Il serait possible de le dire si 0°C correspondrait à "pas de température du tout", mais cela n'est évidemment pas le cas (cet exemple est repris de Danielle Navarro (2018)). En revanche, les variables pouvant être associées à une échelle de ratios présentent des valeurs qui, en plus de permettre de calculer des différences numériques qui ont un sens, peuvent être multipliées ou divisées, comme par exemple le temps de réaction à un stimulus donné.

En plus de la distinction échelle d'intervalles / échelle de ratios, les variables quantitatives peuvent être considérées comme étant soit **continues**, soit **discrètes**. Les variables quantitatives continues contiennent des nombres pouvant comporter théoriquement un nombre infini de décimales (e.g., la taille, le poids, etc.). Au contraire, les variables quantitatives discrètes ne peuvent contenir théoriquement que des nombres finis (e.g., le nombre de victoires sportives au cours d'une année). Certaines variables en théorie discrètes sont cependant souvent considérées comme continues tant le nombre de valeurs théoriquement possibles pour la variable est grand, tel que pour le nombre de globules blancs mesurés dans le sang (Labreuche, 2010).

Les **variables qualitatives** (aussi dites catégorielles), elles, contiennent des valeurs désignant non pas des quantités mais des modalités. Ces variables sont donc forcément discrètes. Les modalités peuvent être exprimées sous forme littérale ou numérique. Parmi les variables qualitatives, on distingue celles qui sont **nominales** et celles qui sont **ordinales**. Les variables qualitatives nominales contiennent des modalités qui ne peuvent pas être ordonnées (e.g., les couleurs, les genres, etc.). Au contraire, les variables qualitatives ordinales contiennent des modalités qui peuvent être ordonnées (e.g., les niveaux de compétition sportive : départemental ; régional ; interrégional ; national ; international). Les variables qualitatives ordinales qui prendraient des valeurs numériques pour indiquer par exemple un niveau d'expertise (e.g., 1, 2, 3, et 4) se différencient des variables quantitatives discrètes par l'absence d'information sur la distance qui sépare les nombres de cette variable (Labreuche, 2010).

2.2 Fixer le répertoire de travail

Lorsque l'on souhaite réaliser l'analyse d'une base de données avec RStudio, il peut être utile et plus fonctionnel pour la suite de créer un dossier spécifique, sur l'ordinateur, relatif à son projet de travail. Ce dossier pourrait alors contenir au moins trois sous-dossiers appelés : "data" ; "out" ; et "R". L'idée ici est d'organiser tous les fichiers qui vont être utilisés et produits au cours du travail d'analyse. Le dossier "data" devrait ne contenir que les fichiers à analyser. Le dossier "out" ne devrait contenir que les fichiers qui sont exportés au cours des analyses réalisées. Et le dossier "R" ne devrait contenir que les fichiers .R qui servent à écrire et activer le code nécessaire aux analyses.

Une fois la structure de travail créée, il est préférable de faire en sorte que l'emplacement sur le PC du dossier du projet en cours soit le point de départ des chemins d'accès qui serviront à importer des données dans RStudio ou à exporter des fichiers à partir de RStudio. Pour faire cela, il faut créer un fichier .Rproj dans le dossier général du projet. Pour créer ce fichier, il suffit de suivre, dans RStudio, le chemin suivant : **Fichier > New Project... > Existing Directory > Sélectionner le chemin d'accès au dossier souhaité > Cliquer sur Create Project**. Un grand avantage de cette procédure est que cela permet de ne pas écrire, dans le script, le chemin d'accès complet d'un fichier sur son PC (de la racine au fichier lui-même) et ainsi de ne pas révéler dans le script des informations qui sont relativement privées. Cette solution est donc celle à préférer, surtout lorsqu'on envisage de partager son script avec d'autres collaborateurs qui risqueraient qui plus est d'être gênés par cette ligne de code d'accès au fichier qui ne leur servirait à rien, car le chemin d'accès au dossier de travail d'un collègue sera très probablement

différent de celui des autres collaborateurs.

Une fois le projet bien configuré, il faut ensuite, dans RStudio, ouvrir un fichier **Script** où toutes les commandes seront écrites et enregistrables (chemin d'accès : **File > New File > R Script**). Une fois ouvert, il est possible d'enregistrer le script en appuyant sur **Ctrl+S**.

2.3 Importer la base de données

Il existe plusieurs fonctions pour importer une base de données dans RStudio. La fonction `read_csv2()` du package `readr` permet d'importer par exemple des fichiers .csv qui, structurellement, séparent les données avec des points-virgules. C'est généralement le type de structure de fichier .csv que l'on obtient après avoir réalisé un export .csv à partir du logiciel Excel. Pour illustrer ici l'importation d'une base de données, il est d'abord possible d'en créer une dans le répertoire de travail actif, cela en exportant un tableau de données qui existe déjà avec le logiciel R. Le logiciel R dispose en effet d'un grand nombre de jeux de données différents que tout utilisateur peut consulter et manipuler. L'ensemble des jeux de données disponibles suite à l'installation par défaut de R est visible en lançant dans la Console la commande `data()`. Au fur et à mesure de la découverte des analyses montrées dans ce document, différents jeux de données seront utilisés en fonction des besoins. Pour le moment, il est possible d'utiliser le jeu de données qui s'appelle `iris`. Même si on ne le voit pas dans la fenêtre Environnement de RStudio, il est bel et bien là, disponible, prêt à être utilisé. Afin d'exporter ce jeu de données dans le répertoire de travail fixé au préalable, il est possible d'utiliser la fonction `write_csv2()` du package `readr`. Pour cela, il suffit d'utiliser le nom du jeu de données, puis d'indiquer entre guillemets le nom que l'on veut que le fichier exporté ait, tout en n'oubliant pas de mettre l'extension .csv à la fin du nom pour indiquer le format d'export, comme ci-dessous.

```
library(readr)
write_csv2(x = iris, file = "out/iris.csv")
```

Si la commande ci-dessus est activée dans RStudio et que l'on jette ensuite un oeil dans le répertoire de travail (dossier "out"), il est alors possible d'y voir un nouveau fichier .csv du nom de `iris`. Maintenant qu'il existe une base de données dans le répertoire de travail actif, il est possible de concrétiser la procédure de son importation dans RStudio. Comme évoqué plus tôt dans ce document, il est intéressant, et en réalité nécessaire, d'assigner cette base de données à un nom pour pouvoir plus facilement manipuler le jeu de données par la suite. Ici, nous allons tout simplement associer ce nouvel objet au nom `iris`, tel que montré ci-dessous.

```
iris <- read_csv2(file = "out/iris.csv")
```

Suite à l'activation de la commande, RStudio nous montre un message d'information sur la manière dont la fonction `read_csv2()` a configuré le jeu de données importé. Ce message apparaît car la fonction importe le jeu de données non pas sous la forme d'un *data frame* comme nous avons pu en créer auparavant, mais sous la forme d'un *tibble*, qui désigne un format de tableau que l'on ne peut obtenir qu'en passant par le biais de fonctions associées à l'ensemble de packages `tidyverse`. Pour comprendre l'intérêt d'un *tibble*, revenons au format classique d'un *data frame* à l'aide de la fonction `as.data.frame()`.

```
iris <- as.data.frame(x = iris)
```

À présent, regardons ce qu'il se passe si on lance le nom `iris` dans la Console…

```
iris
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 2            4.9         3.0          1.4         0.2     setosa
## 3            4.7         3.2          1.3         0.2     setosa
## 4            4.6         3.1          1.5         0.2     setosa
## 5            5.0         3.6          1.4         0.2     setosa
## 6            5.4         3.9          1.7         0.4     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 8            5.0         3.4          1.5         0.2     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 12           4.8         3.4          1.6         0.2     setosa
## 13           4.8         3.0          1.4         0.1     setosa
## 14           4.3         3.0          1.1         0.1     setosa
## 15           5.8         4.0          1.2         0.2     setosa
## 16           5.7         4.4          1.5         0.4     setosa
## 17           5.4         3.9          1.3         0.4     setosa
## 18           5.1         3.5          1.4         0.3     setosa
## 19           5.7         3.8          1.7         0.3     setosa
## 20           5.1         3.8          1.5         0.3     setosa
## 21           5.4         3.4          1.7         0.2     setosa
## 22           5.1         3.7          1.5         0.4     setosa
## 23           4.6         3.6          1.0         0.2     setosa
## 24           5.1         3.3          1.7         0.5     setosa
## 25           4.8         3.4          1.9         0.2     setosa
## 26           5.0         3.0          1.6         0.2     setosa
## 27           5.0         3.4          1.6         0.4     setosa
## 28           5.2         3.5          1.5         0.2     setosa
## 29           5.2         3.4          1.4         0.2     setosa
## 30           4.7         3.2          1.6         0.2     setosa
## 31           4.8         3.1          1.6         0.2     setosa
## 32           5.4         3.4          1.5         0.4     setosa
## 33           5.2         4.1          1.5         0.1     setosa
## 34           5.5         4.2          1.4         0.2     setosa
## 35           4.9         3.1          1.5         0.2     setosa
## 36           5.0         3.2          1.2         0.2     setosa
## 37           5.5         3.5          1.3         0.2     setosa
## 38           4.9         3.6          1.4         0.1     setosa
## 39           4.4         3.0          1.3         0.2     setosa
## 40           5.1         3.4          1.5         0.2     setosa
## 41           5.0         3.5          1.3         0.3     setosa
## 42           4.5         2.3          1.3         0.3     setosa
## 43           4.4         3.2          1.3         0.2     setosa
## 44           5.0         3.5          1.6         0.6     setosa
## 45           5.1         3.8          1.9         0.4     setosa
## 46           4.8         3.0          1.4         0.3     setosa
## 47           5.1         3.8          1.6         0.2     setosa
## 48           4.6         3.2          1.4         0.2     setosa
## 49           5.3         3.7          1.5         0.2     setosa
## 50           5.0         3.3          1.4         0.2     setosa
## 51           7.0         3.2          4.7         1.4 versicolor
```

```
## 52          6.4         3.2          4.5         1.5 versicolor
## 53          6.9         3.1          4.9         1.5 versicolor
## 54          5.5         2.3          4.0         1.3 versicolor
## 55          6.5         2.8          4.6         1.5 versicolor
## 56          5.7         2.8          4.5         1.3 versicolor
## 57          6.3         3.3          4.7         1.6 versicolor
## 58          4.9         2.4          3.3         1.0 versicolor
## 59          6.6         2.9          4.6         1.3 versicolor
## 60          5.2         2.7          3.9         1.4 versicolor
## 61          5.0         2.0          3.5         1.0 versicolor
## 62          5.9         3.0          4.2         1.5 versicolor
## 63          6.0         2.2          4.0         1.0 versicolor
## 64          6.1         2.9          4.7         1.4 versicolor
## 65          5.6         2.9          3.6         1.3 versicolor
## 66          6.7         3.1          4.4         1.4 versicolor
## 67          5.6         3.0          4.5         1.5 versicolor
## 68          5.8         2.7          4.1         1.0 versicolor
## 69          6.2         2.2          4.5         1.5 versicolor
## 70          5.6         2.5          3.9         1.1 versicolor
## 71          5.9         3.2          4.8         1.8 versicolor
## 72          6.1         2.8          4.0         1.3 versicolor
## 73          6.3         2.5          4.9         1.5 versicolor
## 74          6.1         2.8          4.7         1.2 versicolor
## 75          6.4         2.9          4.3         1.3 versicolor
## 76          6.6         3.0          4.4         1.4 versicolor
## 77          6.8         2.8          4.8         1.4 versicolor
## 78          6.7         3.0          5.0         1.7 versicolor
## 79          6.0         2.9          4.5         1.5 versicolor
## 80          5.7         2.6          3.5         1.0 versicolor
## 81          5.5         2.4          3.8         1.1 versicolor
## 82          5.5         2.4          3.7         1.0 versicolor
## 83          5.8         2.7          3.9         1.2 versicolor
## 84          6.0         2.7          5.1         1.6 versicolor
## 85          5.4         3.0          4.5         1.5 versicolor
## 86          6.0         3.4          4.5         1.6 versicolor
## 87          6.7         3.1          4.7         1.5 versicolor
## 88          6.3         2.3          4.4         1.3 versicolor
## 89          5.6         3.0          4.1         1.3 versicolor
## 90          5.5         2.5          4.0         1.3 versicolor
## 91          5.5         2.6          4.4         1.2 versicolor
## 92          6.1         3.0          4.6         1.4 versicolor
## 93          5.8         2.6          4.0         1.2 versicolor
## 94          5.0         2.3          3.3         1.0 versicolor
## 95          5.6         2.7          4.2         1.3 versicolor
## 96          5.7         3.0          4.2         1.2 versicolor
## 97          5.7         2.9          4.2         1.3 versicolor
## 98          6.2         2.9          4.3         1.3 versicolor
## 99          5.1         2.5          3.0         1.1 versicolor
## 100         5.7         2.8          4.1         1.3 versicolor
## 101         6.3         3.3          6.0         2.5  virginica
## 102         5.8         2.7          5.1         1.9  virginica
## 103         7.1         3.0          5.9         2.1  virginica
## 104         6.3         2.9          5.6         1.8  virginica
## 105         6.5         3.0          5.8         2.2  virginica
```

```
## 106          7.6         3.0          6.6         2.1  virginica
## 107          4.9         2.5          4.5         1.7  virginica
## 108          7.3         2.9          6.3         1.8  virginica
## 109          6.7         2.5          5.8         1.8  virginica
## 110          7.2         3.6          6.1         2.5  virginica
## 111          6.5         3.2          5.1         2.0  virginica
## 112          6.4         2.7          5.3         1.9  virginica
## 113          6.8         3.0          5.5         2.1  virginica
## 114          5.7         2.5          5.0         2.0  virginica
## 115          5.8         2.8          5.1         2.4  virginica
## 116          6.4         3.2          5.3         2.3  virginica
## 117          6.5         3.0          5.5         1.8  virginica
## 118          7.7         3.8          6.7         2.2  virginica
## 119          7.7         2.6          6.9         2.3  virginica
## 120          6.0         2.2          5.0         1.5  virginica
## 121          6.9         3.2          5.7         2.3  virginica
## 122          5.6         2.8          4.9         2.0  virginica
## 123          7.7         2.8          6.7         2.0  virginica
## 124          6.3         2.7          4.9         1.8  virginica
## 125          6.7         3.3          5.7         2.1  virginica
## 126          7.2         3.2          6.0         1.8  virginica
## 127          6.2         2.8          4.8         1.8  virginica
## 128          6.1         3.0          4.9         1.8  virginica
## 129          6.4         2.8          5.6         2.1  virginica
## 130          7.2         3.0          5.8         1.6  virginica
## 131          7.4         2.8          6.1         1.9  virginica
## 132          7.9         3.8          6.4         2.0  virginica
## 133          6.4         2.8          5.6         2.2  virginica
## 134          6.3         2.8          5.1         1.5  virginica
## 135          6.1         2.6          5.6         1.4  virginica
## 136          7.7         3.0          6.1         2.3  virginica
## 137          6.3         3.4          5.6         2.4  virginica
## 138          6.4         3.1          5.5         1.8  virginica
## 139          6.0         3.0          4.8         1.8  virginica
## 140          6.9         3.1          5.4         2.1  virginica
## 141          6.7         3.1          5.6         2.4  virginica
## 142          6.9         3.1          5.1         2.3  virginica
## 143          5.8         2.7          5.1         1.9  virginica
## 144          6.8         3.2          5.9         2.3  virginica
## 145          6.7         3.3          5.7         2.5  virginica
## 146          6.7         3.0          5.2         2.3  virginica
## 147          6.3         2.5          5.0         1.9  virginica
## 148          6.5         3.0          5.2         2.0  virginica
## 149          6.2         3.4          5.4         2.3  virginica
## 150          5.9         3.0          5.1         1.8  virginica
```

RStudio nous montre tout le jeu de données dans la Console, ce qui n'est pas très utile, d'autant plus que l'on peut perdre de vue la première ligne de titre lorsque le jeu de données contient beaucoup de lignes. Retournons donc au format *tibble* grâce à la fonction `as_tibble()` du package `tibble`, et voyons ce qu'il se passe lorsqu'on lance à nouveau le nom `iris` dans la Console.

```
library(tibble)
iris <- as_tibble(x = iris)
```

```
iris
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <chr>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # i 140 more rows
```

Cette fois, RStudio n'affiche que les premières lignes du jeu de données, et il fournit en plus de cela des informations quant aux types de variables présentes dans le jeu de données, en-dessous de la ligne de titres. Maintenant que la base de données a été importée, il ne reste plus qu'à voir différentes fonctions pour pouvoir configurer la base de données telle qu'on la voudrait pour réaliser confortablement les analyses.

2.4 Manipuler la base de données

2.4.1 Vérifier le succès de l'importation de la base

Avant de débuter les analyses de la base de données, une bonne pratique est de vérifier si la base de données a été correctement importée avec RStudio. Une manière rapide de faire cela est de regarder les nombres d'observations (i.e., de lignes) et de variables (i.e., de colonnes) associés à l'objet crée lors de l'importation et visibles dans la fenêtre Environnement de RStudio, puis de cliquer sur le nom associé à l'objet. Lors de l'étape précédente, nous avons importé le jeu de données `iris` en l'appelant ainsi. Lorsque l'on cherche le nom `iris` dans la fenêtre Environnement, on peut voir que l'objet associé contient 150 observations et 5 variables, signes que la structure du jeu de données a été bien interprétée par R si l'on sait que ce sont effectivement les dimensions du jeu de données en question. Puis, lorsque l'on clique sur le nom `iris` dans la liste des noms montrés dans la fenêtre Environnement, RStudio ouvre un onglet qui contient les données. Il est alors possible de voir d'un simple coup d'oeil si les données sont bien présentes et organisées en lignes et en colonnes comme attendu.

2.4.2 Vérifier et reconfigurer les types des variables de la base

Il convient de vérifier que les types des variables que RStudio a associés aux variables du jeu de données importé soient bien en accord avec ce qui était attendu. Pour vérifier les types des variables, il est possible d'utiliser la fonction `str()` avec le nom auquel on a associé la base de données.

```
str(iris)
```

```
## tibble [150 x 5] (S3: tbl_df/tbl/data.frame)
```

```
##  $ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num [1:150] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : chr [1:150] "setosa" "setosa" "setosa" "setosa" ...
```

Après avoir activé la commande contenant la fonction `str()`, la Console affiche plusieurs lignes d'information (cf. texte des résultats ci-dessus), avec à chaque fois le nom de la variable, son type, et les premières valeurs de la variable. Plusieurs termes peuvent être rencontrés selon la manière dont R a interprété les variables du jeu de données, notamment:

- num : désigne une variable quantitative continue ;
- int : dégine une variable quantitative discrète (avec des nombres entiers) ;
- Factor : désigne une variable qualitative ;
- chr : désigne une variable texte ;
- Date : désigne une variable date.

Le logiciel R s'appuie donc sur une classification des types de variables plus complexe que celle que nous avons présentée précédemment. On peut noter que les abbréviations montrées pour indiquer le type de variable en utilisant la fonction `str()` sont différentes de celles montrées lorsque l'on observe un jeu de données au format *tibble* dans la Console, mais ces différences reflètent en réalité principalement une divergence dans les stratégies d'écriture de l'information par les concepteurs des packages et des fonctions. En outre, si l'on veut déterminer le type d'une seule variable, ou plus globalement le type de l'objet qui nous intéresse, on peut utiliser la fonction `class()`. Utiliser un nom de variable avec cette fonction renverra le type de la variable, comme montré ci-dessous.

```
class(x = iris$Sepal.Length)
```

```
## [1] "numeric"
```

Lorsque le type d'une variable ne correspond pas à celui attendu après avoir importé la base de données dans RStudio, il peut être utile de se questionner sur les erreurs qui ont pu causer cela. Lorsque l'on obtient une variable de type *Factor* ou de type *chr* alors qu'une variable de type *num* étaient attendue, une cause possible est que l'importation du jeu de données a été réalisée avec une fonction d'importation mal configurée par rapport au contenu du jeu de données. Par exemple, il est possible que la fonction d'importation du jeu de données reconnaissait les nombres décimaux seulement lorsqu'ils avaient des points (e.g., 24.3) alors qu'en réalité les nombres décimaux étaient écrits avec des virgules (e.g., 24,3) dans la base de données. Une autre possibilité est que l'on n'ait pas indiqué, dans la fonction d'importation, sous quelle forme se présentaient les valeurs manquantes de la base de données. Par exemple, avec des valeurs manquantes qui seraient notées "NA" dans des variables numériques de la base de données, l'usage de certaines fonctions d'importation sans indiquer à l'intérieur que "NA" désigne "valeur manquante" conduira R à interpréter les variables concernées comme des variables *chr*. En utilisant la fonction `read_csv2()` du package `readr`, ces écueils sont plus facilement évités car les paramètres par défaut de la fonction nous facilitent le travail. En revanche, d'autres fonctions, plus anciennes, comme `read.csv2()` qui est une fonction de base de R, nécessitent plus de vigilance.

Lorsque la modification du type de la variable est nécessaire, une stratégie possible est de créer une variable portant exactement le même nom à partir de la variable initiale et à laquelle on applique une fonction capable d'imposer un certain type de variable. Il existe une fonction pour chaque type de variable à définir, notamment :

- La fonction `as.numeric()` pour obtenir un type de variable quantitative ;
- La fonction `as.factor()` pour un obtenir un type de variable qualitative ;
- La fonction `as.character()` pour un obtenir un type de variable texte ;
- La fonction `as.Date()` pour obtenir un type de variable date.

Par exemple, nous aurions pu vouloir faire en sorte que toutes les variables du jeu de données `iris` soient de type texte :

```
iris$Sepal.Length <- as.character(x = iris$Sepal.Length)
iris$Sepal.Width <- as.character(x = iris$Sepal.Width)
iris$Petal.Length <- as.character(x = iris$Petal.Length)
iris$Petal.Width <- as.character(x = iris$Petal.Width)
iris$Species <- as.character(x = iris$Species)
```

Remarquons qu'à chaque fois, le nom de variable écrit à gauche de la flèche d'assignation est exactement le même que celui qui est écrit à droite de la flèche d'assignation dans la fonction `as.character()`, ce qui implique que la création de la nouvelle variable entraîne la suppression et le remplacement de la précédente qui portait le même nom. Il est possible de vérifier la conséquence de ces commandes avec la fonction `str()`.

```
str(iris)
```

```
## tibble [150 x 5] (S3: tbl_df/tbl/data.frame)
##  $ Sepal.Length: chr [1:150] "5.1" "4.9" "4.7" "4.6" ...
##  $ Sepal.Width : chr [1:150] "3.5" "3" "3.2" "3.1" ...
##  $ Petal.Length: chr [1:150] "1.4" "1.4" "1.3" "1.5" ...
##  $ Petal.Width : chr [1:150] "0.2" "0.2" "0.2" "0.2" ...
##  $ Species     : chr [1:150] "setosa" "setosa" "setosa" "setosa" ...
```

Cette stratégie de modification du type de la variable peut convenir lorsqu'il y a peu de variables à modifier. Cependant, lorsque la liste s'allonge, il peut être plus lisible, en matière de code, de fonctionner avec le symbole `|>` (qu'on appelle *pipe*) lorsque sa version de R est >= 4.1.0 (ou à défaut avec le symbole `%>%` du package `magrittr`), et la fonction `mutate()` du package `dplyr`.

```
library(dplyr)
iris <-
  iris |>
  mutate(
    Sepal.Length = as.numeric(x = Sepal.Length),
    Sepal.Width = as.numeric(x = Sepal.Width),
    Petal.Length = as.numeric(x = Petal.Length),
    Petal.Width = as.numeric(x = Petal.Width),
    Species = as.factor(x = Species)
  )
```

Ici, le symbole `|>` permet d'indiquer à R que toutes les fonctions qui sont écrites après ce symbole s'appliquent à ce qui a été défini avant ce symbole. La fonction `mutate()`, dont nous reparlerons peu après, permet de créer de nouvelles variables dans le cadre de cette stratégie, soit en écrasant les anciennes variables si les anciens noms sont conservés, soit en créant de nouvelles variables si de nouveaux noms sont utilisés. Remarquons également qu'avec ce code, nous venons de créer un nouvel objet (en l'assignant à nouveau au nom `iris`) à partir de l'ancien objet, mais dont on a transformé les types des variables, perdant dans le même temps l'ancien objet.

Tableau 2.4: Data summary

Name	iris
Number of rows	150
Number of columns	5
_____	
Column type frequency:	
factor	1
numeric	4
_____	
Group variables	None

On pourrait penser que le bloc de code montré juste ci-dessus n'est pas très satisfaisant car on répète finalement plusieurs fois une même action. Ici il y a seulement 5 variables dont 4 numériques donc cela peut encore aller. Mais s'il y en avait des dizaines et plus, devrait-on tout écrire comme ci-dessus ? Heureusement non. Pour éviter de réécrire la même action pour plusieurs variables, on peut utiliser la fonction `accross()` du package `dplyr`, comme ceci :

```
iris <-
  iris |>
  mutate(across(c(Sepal.Length:Petal.Width), as.numeric),
         Species = as.factor(Species))
```

Enfin, si la fonction `str()` a l'intérêt de faire partie des fonctions de base de R, on peut noter l'existence de fonctions associées à d'autres packages qui sont particulièrement intéressantes pour découvrir un jeu de données. C'est le cas notamment de la fonction `skim()` du package `skimr` :

```
library(skimr)
skim(iris)
```

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Species	0	1	FALSE	3	set: 50, ver: 50, vir: 50

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Sepal.Length	0	1	5.84	0.83	4.3	5.1	5.80	6.4	7.9	
Sepal.Width	0	1	3.06	0.44	2.0	2.8	3.00	3.3	4.4	
Petal.Length	0	1	3.76	1.77	1.0	1.6	4.35	5.1	6.9	
Petal.Width	0	1	1.20	0.76	0.1	0.3	1.30	1.8	2.5	

2.4.3 Sélectionner des variables avec `select()`

Certains jeux de données peuvent être très larges, c'est-à-dire qu'ils peuvent contenir beaucoup de colonnes, parfois inutiles, et qui peuvent être gênantes lorsque l'on veut avoir une vue claire du contenu du jeu de données. La fonction `select()` du package `dplyr` permet de sélectionner des colonnes facilement.

```
iris |>
  dplyr::select(Petal.Length, Petal.Width, Species)
```

```
## # A tibble: 150 x 3
##    Petal.Length Petal.Width Species
##           <dbl>       <dbl> <fct>
##  1          1.4         0.2 setosa
##  2          1.4         0.2 setosa
##  3          1.3         0.2 setosa
##  4          1.5         0.2 setosa
##  5          1.4         0.2 setosa
##  6          1.7         0.4 setosa
##  7          1.4         0.3 setosa
##  8          1.5         0.2 setosa
##  9          1.4         0.2 setosa
## 10          1.5         0.1 setosa
## # i 140 more rows
```

2.4.4 Renommer des variables avec `rename()`

Il est possible que certains noms de variables ne soient pas clairs ou trop longs, voire mal écrits, ce qui peut être gênant pour écrire un code le plus lisible possible. La fonction `rename()` du package `dplyr` permet de gérer cela. Dans l'exemple ci-dessous, on observe que le nouveau nom doit être écrit à gauche du signe `=`, alors que l'ancien nom doit être écrit à droite du signe `=`. Si le nom d'origine contient au moins une espace entre deux mots, ou encore s'il contient des caractères spéciaux, il convient d'encadrer le nom à remplacer par des guillemets (" ").

```
iris |>
  rename(Sepal_long = Sepal.Length,
         Sepal_lar = Sepal.Width,
         Petal_long = Petal.Length,
         Petal_lar = Petal.Width,
         Especes = Species)
```

```
## # A tibble: 150 x 5
##    Sepal_long Sepal_lar Petal_long Petal_lar Especes
##         <dbl>     <dbl>      <dbl>     <dbl> <fct>
##  1        5.1       3.5        1.4       0.2 setosa
##  2        4.9       3          1.4       0.2 setosa
##  3        4.7       3.2        1.3       0.2 setosa
##  4        4.6       3.1        1.5       0.2 setosa
##  5        5         3.6        1.4       0.2 setosa
##  6        5.4       3.9        1.7       0.4 setosa
##  7        4.6       3.4        1.4       0.3 setosa
##  8        5         3.4        1.5       0.2 setosa
##  9        4.4       2.9        1.4       0.2 setosa
## 10        4.9       3.1        1.5       0.1 setosa
## # i 140 more rows
```

Cette méthode visant à renommer les variables pour modifier un caractère ou plusieurs caractères peut être à nouveau très fastidieuse en présence de nombreuses variables à renommer car mal écrite au départ. La fonction `clean_names()` du package `janitor` pourrait alors faire gagner

beaucoup de temps. Cette fonction réécrit les noms des variables pour qu'elles soient titrées (nommées) conformément à certains standards, à savoir : pas de points en général dans les noms des variables, plutôt des tirets du bas ; éviter les majuscules dans les noms d'objet ; éviter les caractères spéciaux ; etc. Voilà ce que cela donnerait :

```
library(janitor)
iris |>
  clean_names()
```

```
## # A tibble: 150 x 5
##    sepal_length sepal_width petal_length petal_width species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # i 140 more rows
```

2.4.5 Créer des variables avec `mutate()`

Certaines analyses peuvent nécessiter d'ajouter des variables à partir de calculs réalisés sur des variables qui existent déjà dans le jeu de données. La fonction `mutate()`, du package `dplyr`, et que nous avons déjà rencontrée précédemment, permet cela. Dans l'exemple ci-dessous, on observe que le nom de la nouvelle variable à créer est à gauche du signe `=` et que le calcul créant les nouvelles valeurs est décrit à droite du signe `=`.

```
iris |>
  mutate(ratio_sepal = Sepal.Length / Sepal.Width,
         ratio_petal = Petal.Length / Petal.Width)
```

```
## # A tibble: 150 x 7
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # i 140 more rows
## # i 2 more variables: ratio_sepal <dbl>, ratio_petal <dbl>
```

2.4.6 Sélectionner des lignes avec `filter()`

En fonction des besoins de l'analyse, on peut vouloir ne retenir que certaines lignes du fichier de données. La fonction `filter()` du package `dplyr` est faite pour réaliser ce filtrage. Plusieurs opérateurs sont disponibles pour ne retenir que les lignes que l'on veut (cf. Tableau 2.5).

Tableau 2.5: Les opérateurs utilisables avec la fonction filter()

Opération	Opérateur
Égal	==
Inférieur ou égal	<=
Supérieur ou égal	>=
Différent de	!=

De plus, dans la configuration du code, ces opérateurs peuvent être couplés à l'opérateur `|` (OU) et à l'opérateur `&` (ET). Dans l'exemple ci-dessous, le code permet, à partir du jeu de données `iris`, de ne garder que les lignes du jeu de données qui contiennent les noms d'espèce *setosa* OU *virginica*, ET en même temps qui affichent une longueur de sépale inférieure ou égale à 5.

```
iris |>
  filter((Species == "setosa" | Species == "virginica") &
           Sepal.Length <= 5)
```

```
## # A tibble: 29 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.9         3            1.4         0.2 setosa
##  2          4.7         3.2          1.3         0.2 setosa
##  3          4.6         3.1          1.5         0.2 setosa
##  4          5           3.6          1.4         0.2 setosa
##  5          4.6         3.4          1.4         0.3 setosa
##  6          5           3.4          1.5         0.2 setosa
##  7          4.4         2.9          1.4         0.2 setosa
##  8          4.9         3.1          1.5         0.1 setosa
##  9          4.8         3.4          1.6         0.2 setosa
## 10          4.8         3            1.4         0.1 setosa
## # i 19 more rows
```

2.4.7 Réordonner les lignes avec `arrange()`

On peut vouloir trier les lignes du jeu de données selon un certain ordre, en fonction des valeurs d'une variable donnée. La fonction `arrange()` du package `dplyr` est très utile pour gérer ce genre de réalisation. L'exemple ci-dessous conduit à trier les données selon un ordre croissant en fonction des valeurs de la variable `Sepal.Length`. Le fait de mettre le symbole `-` devant le nom de la variable aurait conduit à un tri décroissant.

```
iris |>
  arrange(Sepal.Length)
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          4.3         3            1.1         0.1 setosa
##  2          4.4         2.9          1.4         0.2 setosa
##  3          4.4         3            1.3         0.2 setosa
##  4          4.4         3.2          1.3         0.2 setosa
##  5          4.5         2.3          1.3         0.3 setosa
##  6          4.6         3.1          1.5         0.2 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          4.6         3.6          1           0.2 setosa
##  9          4.6         3.2          1.4         0.2 setosa
## 10          4.7         3.2          1.3         0.2 setosa
## # i 140 more rows
```

2.4.8 Résumer des variables avec `group_by()` et `summarize()`

Bien qu'une base de données puisse contenir énormément de lignes, on peut n'en vouloir que la version résumée. Les fonctions `group_by()` et `summarize()` du package `dplyr` permettent de faire cela aisément. Dans l'exemple ci-dessous, la fonction `group_by()` permet d'indiquer que les calculs réalisés par la suite avec la fonction `summarize()` doivent être exécutés pour les modalités de la variable `Species` prises séparément. La fonction `summarize()`, quant à elle, permet d'exéctuer différents calculs. Dans l'exemple ci-dessous, il s'agit de moyennes, obtenues à l'aide de la fonction `mean()`. De plus, la fonction `summarize()` permet, comme montré ci-dessous, d'indiquer à gauche du `=` le nom du titre du calcul alors effectué .

```
iris |>
  group_by(Species) |>
  summarize(mean_sep_len = mean(Sepal.Length),
            mean_sep_wid = mean(Sepal.Width))
```

```
## # A tibble: 3 x 3
##   Species    mean_sep_len mean_sep_wid
##   <fct>             <dbl>        <dbl>
## 1 setosa             5.01         3.43
## 2 versicolor         5.94         2.77
## 3 virginica          6.59         2.97
```

Au cours des illustrations montrant l'usage des fonctions `select()` jusqu'à `summarize()`, il aura été possible de noter que les commandes n'écrasaient pas le jeu de données initial, ni ne créaient de nouveaux jeux de données, car aucune assignation à un nom n'était faite. Lorsqu'une assignation est réalisée, il est conseillé d'utiliser un nouveau nom, différent de celui utilisé pour le jeu de données initial, pour pouvoir revenir au jeu de données originel lorsque cela est souhaité. Ci-dessous un exemple de création d'un nouvel objet de type tableau (assigné au nom `iris2`) à partir de l'utilisation de la plupart des fonctions que nous venons de voir et qui peuvent être utilisées dans un même bloc de code grâce au *pipe* (`|>`) :

```
iris2 <-
  iris |>
  dplyr::select(Petal.Length, Petal.Width, Species) |>
  clean_names() |>
  mutate(petal_ratio = petal_length / petal_width) |>
```

```
  filter((species == "setosa" |
            species == "virginica") & petal_ratio > 3) |>
  arrange(-petal_ratio)
iris2
```

```
## # A tibble: 65 x 4
##    petal_length petal_width species petal_ratio
##           <dbl>       <dbl> <fct>         <dbl>
##  1          1.5         0.1 setosa         15
##  2          1.5         0.1 setosa         15
##  3          1.4         0.1 setosa         14
##  4          1.4         0.1 setosa         14
##  5          1.1         0.1 setosa         11
##  6          1.9         0.2 setosa          9.5
##  7          1.7         0.2 setosa          8.5
##  8          1.6         0.2 setosa          8
##  9          1.6         0.2 setosa          8
## 10          1.6         0.2 setosa          8
## # i 55 more rows
```

2.4.9 Passer d'une disposition en lignes à une disposition en colonnes et inversement avec `pivot_wider()` et `pivot_longer()`

Il convient de respecter certaines règles de base lors de la conception d'une base de données (e.g., mettre les observations en lignes et les variables en colonnes). Toutefois, dans certains cas, même après avoir bien respecté les règles, la manière selon laquelle la base de données a été organisée peut ne pas être encore adéquate pour pouvoir utiliser certaines fonctions. Prenons par exemple le cas où toutes les valeurs numériques d'une variable quantitative auraient été mises dans une même colonne en regard d'une variable qualitative pour que chaque valeur numérique corresponde à une modalité de cette variable qualitative (c'est le cas, par exemple, avec le jeu de données `iris`), et que la fonction à utiliser nécessiterait que l'on ait une colonne pour chacune des modalités de la variable qualitative, avec des colonnes mises côte à côte. Une fonction qui permet alors de passer d'un format "long" (i.e., toutes les valeurs numériques sont dans la même colonne) à un format "large" (i.e., les valeurs numériques sont réparties dans différentes colonnes selon la modalité à laquelle elles sont associées), est la fonction `pivot_wider()` du package `tidyr`. Pour pouvoir utiliser cette fonction, il faut qu'il y ait une variable permettant d'identifier à quels individus ou groupes appartiennent les données dont on va changer l'organisation. Dans une base de données classique, il y a toujours une variable présente pour cela. Toutefois, dans le jeu de données `iris`, il n'y a pas une telle variable. Pour pouvoir illustrer l'utilisation de la fonction `pivot_wider()`, nous avons donc ajouté arbitrairement une variable `id` grâce à la fonction `mutate()` pour simuler le fait que les données de `iris` auraient été acquises en référence à des individus bien identifiés, avec à chaque fois une valeur pour les trois modalités de la variable `Species` :

```
iris2 <-
  iris |>
  mutate(id = rep(1:50, times = 3)) |>
  dplyr::select(id, Species,  everything()) |>
  arrange(id, Species) |>
  as_tibble()
iris2
```

```
## # A tibble: 150 x 6
```

```
##       id Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##    <int> <fct>             <dbl>       <dbl>        <dbl>       <dbl>
##  1     1 setosa              5.1         3.5          1.4         0.2
##  2     1 versicolor          7           3.2          4.7         1.4
##  3     1 virginica           6.3         3.3          6           2.5
##  4     2 setosa              4.9         3            1.4         0.2
##  5     2 versicolor          6.4         3.2          4.5         1.5
##  6     2 virginica           5.8         2.7          5.1         1.9
##  7     3 setosa              4.7         3.2          1.3         0.2
##  8     3 versicolor          6.9         3.1          4.9         1.5
##  9     3 virginica           7.1         3            5.9         2.1
## 10     4 setosa              4.6         3.1          1.5         0.2
## # i 140 more rows
```

La fonction `pivot_wider()` permet alors de mettre en colonnes les valeurs des variables sélectionnées pour chacune des trois modalités de la variable `Species`.

```
library(tidyr)
iris3 <-
  iris2 |>
  pivot_wider(
    names_from = Species,
    values_from = Sepal.Length : Petal.Width
    )
iris3
```

```
## # A tibble: 50 x 13
##       id Sepal.Length_setosa Sepal.Length_versicolor
##    <int>               <dbl>                   <dbl>
##  1     1                 5.1                     7
##  2     2                 4.9                     6.4
##  3     3                 4.7                     6.9
##  4     4                 4.6                     5.5
##  5     5                 5                       6.5
##  6     6                 5.4                     5.7
##  7     7                 4.6                     6.3
##  8     8                 5                       4.9
##  9     9                 4.4                     6.6
## 10    10                 4.9                     5.2
## # i 40 more rows
## # i 10 more variables: Sepal.Length_virginica <dbl>,
## #   Sepal.Width_setosa <dbl>, Sepal.Width_versicolor <dbl>,
## #   Sepal.Width_virginica <dbl>, Petal.Length_setosa <dbl>,
## #   Petal.Length_versicolor <dbl>, Petal.Length_virginica <dbl>,
## #   Petal.Width_setosa <dbl>, Petal.Width_versicolor <dbl>,
## #   Petal.Width_virginica <dbl>
```

L'argument `names_from` a permis d'indiquer la variable à partir de laquelle on a dispatché les valeurs en colonnes, et l'argument `values_from` a permis de préciser les variables pour lesquelles on voulait que les valeurs numériques soient dispatchées. L'utilisation des deux-points (`:`) nous a permis de sélectionner toutes les variables allant de `Sepal.Length` à `Petal.Width` dans le jeu de données.

Dans une situation inverse à celle que nous venons de voir, nous pourrions avoir les données représentées en colonnes (comme c'est le cas avec le jeu de données `iris3` crée ci-dessus), alors

que nous les voudrions en lignes (comme c'était le cas avec le jeu de données `iris` à l'origine). La fonction `pivot_longer()` permet de faire ce genre de conversion. Pour observer ce que cette fonction réalise, sélectionnons seulement les colonnes avec les valeurs de sépales en plus de la colonne `id`, tel que montré ci-dessous :

```
iris3 |>
  dplyr::select(id,
                Sepal.Length_setosa,
                Sepal.Length_versicolor,
                Sepal.Length_virginica) %>%
  pivot_longer(cols = c(-id),
               names_to = "Species",
               values_to = "Sepal_len")
```

```
## # A tibble: 150 x 3
##       id Species                Sepal_len
##    <int> <chr>                      <dbl>
##  1     1 Sepal.Length_setosa          5.1
##  2     1 Sepal.Length_versicolor      7
##  3     1 Sepal.Length_virginica       6.3
##  4     2 Sepal.Length_setosa          4.9
##  5     2 Sepal.Length_versicolor      6.4
##  6     2 Sepal.Length_virginica       5.8
##  7     3 Sepal.Length_setosa          4.7
##  8     3 Sepal.Length_versicolor      6.9
##  9     3 Sepal.Length_virginica       7.1
## 10     4 Sepal.Length_setosa          4.6
## # i 140 more rows
```

Dans la fonction `pivot_longer()` ci-dessus, nous avons indiqué à l'aide de l'argument `cols` et de la fonction `c()` avec le signe `-` la colonne que nous ne voulions pas utiliser avec la fonction (c'était plus rapide que d'indiquer dans la fonction `c()` les trois colonnes à utiliser). L'argument `names_to` nous a permis de donner un nom à la variable qualitative qui comporte à présent les modalités associées aux valeurs numériques, et l'argument `values_to` nous a permis de donner un nom à la colonne où se trouvent maintenant les valeurs numériques.

Lorsque l'on veut utiliser la fonction `pivot_longer()` sur plusieurs colonnes qui sont écrites avec la même logique (par exemple, avec un nom composé de deux morceaux séparés par un `_`, avec un morceau pour indiquer la chose mesurée, et un autre morceau pour indiquer la modalité), il convient de procéder comme ci-dessous :

```
iris3 |>
  pivot_longer(
    cols = c(Sepal.Length_setosa:Petal.Width_virginica),
    names_to = c(".value", "Species"),
    names_sep = "_"
  )
```

```
## # A tibble: 150 x 6
##       id Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##    <int> <chr>             <dbl>       <dbl>        <dbl>       <dbl>
##  1     1 setosa              5.1         3.5          1.4         0.2
##  2     1 versicolor          7           3.2          4.7         1.4
```

```
##  3     1 virginica          6.3         3.3         6           2.5
##  4     2 setosa             4.9         3           1.4         0.2
##  5     2 versicolor         6.4         3.2         4.5         1.5
##  6     2 virginica          5.8         2.7         5.1         1.9
##  7     3 setosa             4.7         3.2         1.3         0.2
##  8     3 versicolor         6.9         3.1         4.9         1.5
##  9     3 virginica          7.1         3           5.9         2.1
## 10     4 setosa             4.6         3.1         1.5         0.2
## # i 140 more rows
```

Dans le code montré ci-dessus, l'argument `cols` a permis de sélectionner toutes les colonnes situées à droite de la colonne `id`, l'argument `names_to` a permis d'indiquer qu'il faut garder des colonnes spécifiques nommées avec le premier morceau du nom des variables antérieures (le mot `".value"` permet cela), et le mot `"Species"` a permis de nommer la colonne avec les espèces d'iris qui étaient les seconds morceaux des noms des variables antérieures.

Nous venons de voir plusieurs fonctions qui peuvent être très utiles pour pouvoir facilement préparer sa base de données en vue des futures analyses. Il ne s'agit que d'une vue très superficielle de tout le potentiel de manipulation des données qu'ont ces fonctions. Pour une vue plus approndie des possibilités qu'offrent ces fonctions, la lecture de l'ouvrage *R for Data Science* d'Hadley Wickham et de Garrett Grolemund (2017) sera particulièrement enrichissante. Cet ouvrage est en libre accès ici : https://r4ds.had.co.nz.

2.5 Résumé

- La base de données est un tableau comportant l'ensemble des données avec les observations organisées en lignes et les variables organisées en colonnes.
- Les grands types de variables que l'on peut retrouver dans une base de données sont les variables quantitatives (avec une échelle d'intervalles ou une échelle de ratios) et les variables qualitatives (nominales ou ordinales).
- Avant de débuter un travail d'analyse, il est conseillé d'initialiser un projet (en créant un fichier .Rproj) dans un dossier où se trouvent le ou les fichiers à analyser.
- Pour importer un jeu de données au format .csv, il est possible d'utiliser la fonction `readr::read_csv2()`.
- Pour exporter un jeu de données au format .csv, il est possible d'utiliser la fonction `readr::write_csv2()`.
- Pour mettre un tableau de données au format *data frame*, utiliser la fonction `as.data.frame()`.
- Pour mettre un tableau de données au format *tibble*, utiliser la fonction `tibble::as_tibble()`.
- Pour lister les variables présentes dans un tableau de données, utiliser la fonction `str()`, ou encore la fonction `skimr::skim()`.
- Pour modifier les types des variables, utiliser des fonctions comme `as.numeric()`, `as.factor()`, `as.character()`, `as.Date()`, etc., éventuellement en combinaison avec la fonction `dplyr::across()` si cela s'y prête.
- Pour sélectionner les variables d'un tableau de données, utiliser la fonction `dplyr::select()`.
- Pour renommer les variables d'un tableau de données, utiliser la fonction `dplyr::rename()`, ou encore la fonction `janitor::clean_names()` pour une réécriture automatique des noms des variables.
- Pour créer de nouvelles variables dans un tableau de données, utiliser la fonction `dplyr::mutate()`.
- Pour sélectionner des lignes dans un tableau de données, utiliser la fonction `dplyr::filter()`.
- Pour trier les lignes d'un tableau de données, utiliser la fonction `dplyr::arrange()`.

- Pour résumer les variables d'un tableau de données, utiliser les fonctions `dplyr::group_by()` et `dplyr::summarize()`.
- Pour passer d'un tableau de données au format *long* à un tableau de données au format *wide*, utiliser la fonction `tidyr::pivot_wider()`.
- Pour passer d'un tableau de données au format *wide* à un tableau de données au format *long*, utiliser la fonction `tidyr::pivot_longer()`.
- Pour enchaîner l'application de fonctions, utiliser le symbole `|>` (*pipe*) avec une version de R >= 4.1.0, ou à défaut le symbole `%>%` du package `magrittr`.

partie II

Analyses descriptives

Chapitre 3

Analyses univariées

Réaliser une analyse descriptive univariée signifie que l'on s'intéresse à une seule variable en particulier. L'enjeu est ici de prendre connaissance de la distribution de la variable, c'est-à-dire de la manière selon laquelle se répartissent les observations en fonction des valeurs que prend la variable. De manière complémentaire, l'analyse descriptive univariée vise à prendre connaissance des indices statistiques qui caractérisent la variable, ainsi qu'à déterminer ceux qui seraient les plus pertinents pour la résumer.

Dans cette partie, les notions de **population** et d'**échantillon** vont revenir à plusieurs reprises. La notion de population désigne tous les individus existant qui satisfont à un ou plusieurs critères particuliers (e.g., les adultes de moins de 30 ans). En général, lorsque l'on souhaite étudier un phénomène dans une population cible, il est impossible de prendre en compte tous les individus de la population en question. L'alternative est alors de conduire l'étude sur un échantillon, c'est-à-dire une fraction de la population composée d'individus qui représentent la population étudiée. La distinction entre population et échantillon est importante à faire à plusieurs égards. Si une étude n'a pu être conduite que sur un échantillon, cela implique de mettre en oeuvre des procédures statistiques pour estimer avec plus ou moins d'incertitude le résultat réel concernant la population étudiée, cela à partir du résultat trouvé dans l'échantillon observé. La seule analyse descriptive de l'échantillon ne suffit donc pas en soi à décrire une population. En revanche, lorsque l'étude a pu être conduite sur l'ensemble de la population à étudier (e.g., l'équipe de France dans un sport donné), il n'y a par définition pas lieu de chercher à conduire des procédures statistiques particulières pour estimer le résultat réel pour la population en question. Dans ce chapitre, les procédures d'analyse proposées servent en général à seulement décrire la variable telle qu'elle est donnée à voir à partir des données que l'on a en sa possession. L'objectif n'est donc pas ici de discuter particulièrement des statistiques les plus pertinentes à utiliser lorsqu'il s'agit de chercher à résumer la distribution d'une variable à l'échelle d'une population à partir d'un échantillon initial. Pour le moment, il s'agit d'être en mesure de décrire l'échantillon (ou la population si les données obtenues concernent toute la population) que l'on a sous les yeux.

Dans le cadre de cette partie, nous allons commencer à voir comment produire des graphiques dans RStudio, et par là-même, découvrir progressivement le package `ggplot2`. Le package `ggplot2` n'est pas le plus simple à utiliser lorsque l'on découvre le logiciel R. D'ailleurs, de nombreux manuels portant sur R privilégient les packages et fonctions de base de R lorsqu'il s'agit de montrer comment obtenir des graphiques relativement simples pour analyser ses données. Cependant, les packages et fonctions de base de R sont rapidement limités lorsqu'il s'agit de réaliser des graphiques relativement complexes. Le parti pris ici est donc d'initier dès à présent à l'utilisation du package `ggplot2` pour réaliser des graphiques, même simples, afin de pouvoir être plus rapidement à l'aise dès lors qu'il s'agira par la suite de produire des graphiques relativement élaborés à l'aide de ce package. Cependant, l'ambition n'est pas ici de permettre la maîtrise complète du package `ggplot2`. Pour cela, il vaut mieux se référer à

des documentations spécialisées telles que la seconde édition de l'ouvrage *ggplot2* d'Hadley Wickham (2016), en sachant qu'une troisième édition est en cours de développement et est accessible en ligne ici : https://ggplot2-book.org.

3.1 Variables quantitatives

3.1.1 Visualiser la distribution de la variable

Dans le cadre de l'analyse de variables quantitatives, il est toujours utile de d'abord visualiser graphiquement la distribution des données à l'aide d'un **histogramme**. Un histogramme, c'est un graphique avec des barres dont la largeur représente un intervalle donné de valeurs numériques, et dont la hauteur représente le nombre d'observations associées à une valeur qui est située dans l'intervalle en question. Plus une barre est haute, plus il y a d'observations concernées par l'intervalle de valeurs. Un exemple d'histogramme est montré sur la Figure 3.1.

```
ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_histogram(fill = "white", color = "black")
```

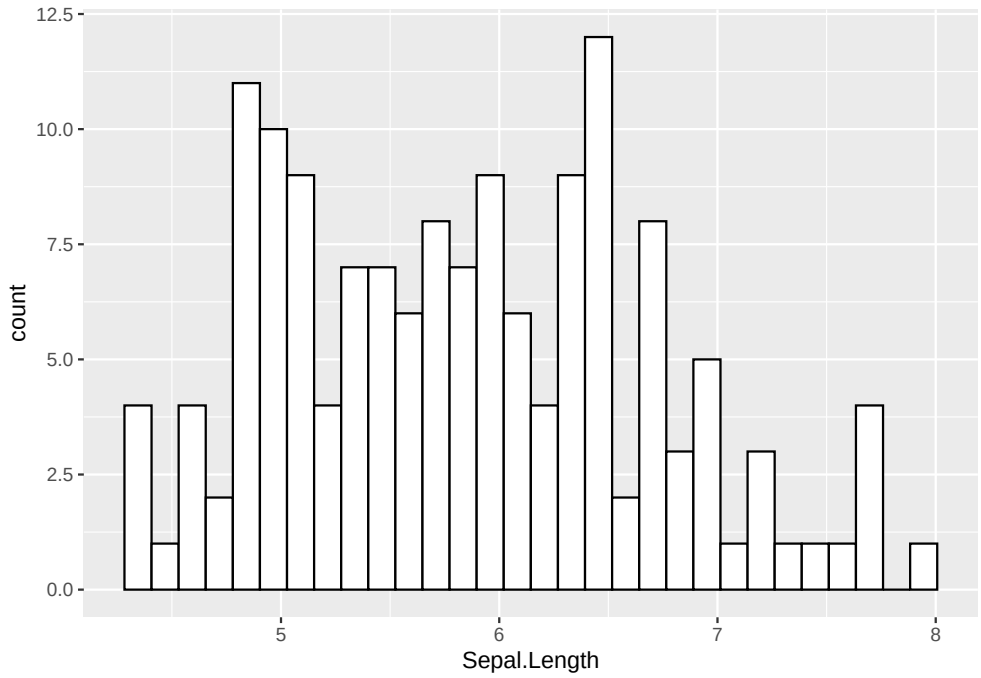

Figure 3.1: Exemple d'histogramme

Pour générer cet histogramme, nous avons utilisé les fonctions `ggplot()` et `geom_histogram()` du package `ggplot2`. La fonction `ggplot()` est nécessaire pour initier le graphique. Si on lance la commande `ggplot()` dans la Console, on peut voir qu'un écran grisé apparaît à droite de l'écran du PC dans la fenêtre **Plots** de RStudio. Cet écran grisé est tel un tableau vierge qui ne demande qu'à être complété grâce à des commandes supplémentaires que l'on doit préciser dans le code. Dans le code montré ci-dessus, on remarque que la fonction `ggplot()` a été configurée à l'aide de deux éléments : l'argument `data`, et la fonction `aes()`. L'argument `data` permet de désigner le jeu de données à partir duquel n'importe quelle autre fonctionnalité du package `ggplot2` sera utilisée si rien d'autre n'est précisé dans le reste du code. Comme on peut le voir,

le jeu de données utilisé ici est `iris`, que nous avons déjà rencontré dans la partie précédente. La fonction `aes()`, elle, permet de désigner les données à partir desquelles les éléments graphiques indiqués par la suite devront être réalisés. Dans le cadre d'une analyse univariée, nous n'avons besoin que d'une seule variable. Celle-ci peut être renseignée à droite de `x =`, et on aura reconnu dans le code ci-dessus le nom d'une variable effectivement présente dans le jeu de données `iris`. Une fois que ces informations sont renseignées, nous ne sommes pas encore en mesure de voir un quelconque graphique. Pour cela, il faut que la fonction `ggplot()` soit accompagnée d'une fonction qui permette d'indiquer quel type de graphique on veut. C'est à cela que sert ici la fonction `geom_histogram()`. On peut noter que l'ajout de cette fonction a été réalisé grâce au signe `+`, en écrivant la fonction *après* ce signe. La fonction `geom_histogram()` aurait pu être écrite directement après le symbole `+`, mais pour des raisons de lisibilité, nous sommes allés à la ligne. (Attention : Aller à la ligne *avant* le signe `+` n'est en revanche pas possible.) De manière intéressante et importante pour la suite, on pourra noter que dans ce cas de figure, nous aurions pu aussi utiliser le symbole *pipe* (`|>`) pour enchaîner la création d'un graphique à la suite de l'écriture du jeu de données comme cela :

```
iris |>
  ggplot(aes(x = Sepal.Length)) +
  geom_histogram(fill = "white", color = "black")

# On remarque ici que l'argument `data = ` dans `ggplot()` a dû être enlevé.
```

Comme l'indique le message qui accompagne le graphique, l'histogramme a été réalisé sur la base de 30 *bins*. Cela signifie que pour faire ce graphique, R a découpé en 30 intervalles égaux l'intervalle allant de la valeur la plus faible de la variable (i.e., le minimum) à la valeur la plus élevée de la variable (i.e., le maximum). Il s'agit de la méthode par défaut utilisée par la fonction `geom_histogram()`. Toutefois, cette méthode par défaut n'est pas vraiment adaptée, comme cela l'est indiqué d'ailleurs dans la documentation d'aide associée à cette fonction. Et puis, lorsqu'il s'agit d'appréhender au mieux la distribution d'une variable avec un histogramme, une bonne pratique est d'observer ce qu'il se passe avec différentes largeurs de *bins*. La largeur d'une *bin* peut être modifiée à l'aide de l'argument `binwidth`. L'unité de la valeur associée à cet argument correspond à l'unité de la variable étudiée (cf. code ci-dessous et Figure 3.2).

```
# Graphique avec binwidth = 0.3
ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_histogram(binwidth  = 0.3,
                 fill = "white",
                 color = "black") +
  ggtitle("binwidth = 0.3")

# Graphique avec binwidth = 0.7
ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_histogram(binwidth  = 0.7,
                 fill = "white",
                 color = "black") +
  ggtitle("binwidth = 0.7")
```

En plus de l'histogramme, une autre manière de prendre connaissance graphiquement de la distribution des données d'une variable quantitative est d'utiliser une **boîte à moustaches**. Pour ce faire, il convient d'utiliser la fonction `geom_boxplot()`, comme montré dans le code ci-dessous. Le résultat de ce code est montré sur la Figure 3.3. On pourra noter ci-dessous l'ajout d'une ligne de code avec une fonction `theme()` dont on ne présentera pas les détails ici ; cette fonction nous sert juste ici à ne pas montrer des chiffres qui auraient été ajoutés par défaut sur l'axe Y de gauche du graphique et qui n'auraient eu aucun intérêt.

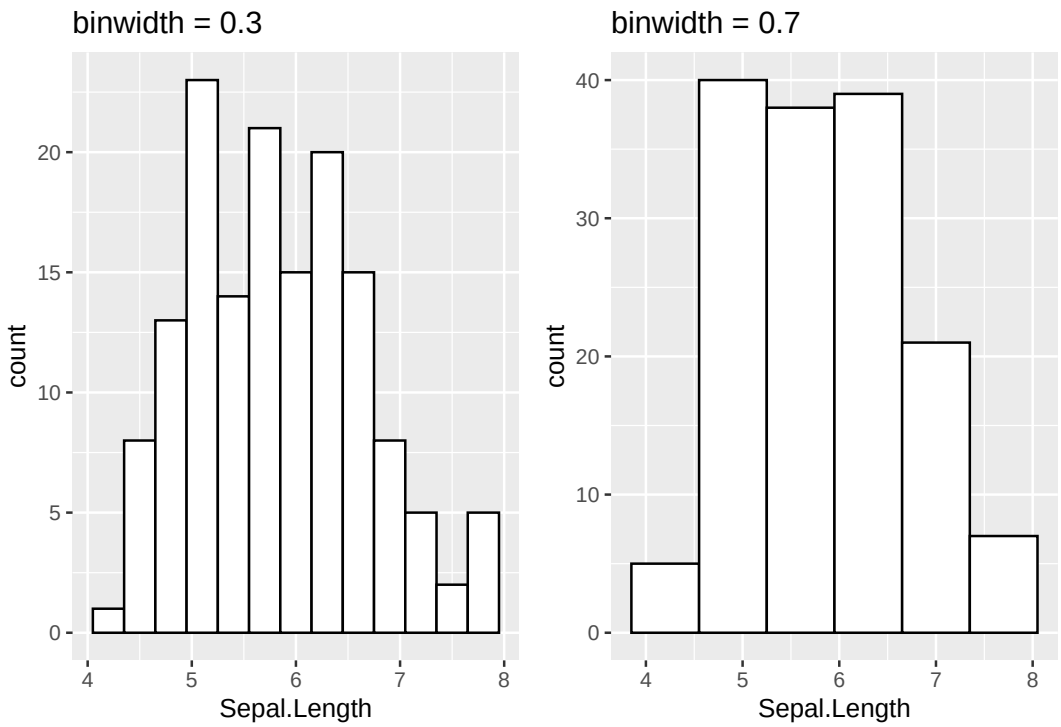

Figure 3.2: Différentes largeurs d'intervalles pour un histogramme

```
ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_boxplot() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

La boîte à moustaches (cf. Figure 3.3) nous livre plusieurs informations. Tout d'abord, ses extrémités nous indiquent ce qu'on appelle le premier quartile (ici représenté par le bord gauche de la boîte) et le troisième quartile (ici représenté par le bord droit de la boîte). Le premier quartile (Q1) désigne la valeur en-dessous de laquelle on retrouve 25 % des observations de la variable (i.e., 25 % des observations sont associées à une valeur plus faible que Q1), alors que le troisième quartile (Q3) représente la valeur en-dessous de laquelle on retrouve 75 % des observations (i.e., 75 % des observations sont associées à une valeur plus faible que Q3). Cela indique alors que sur la Figure 3.3, l'intervalle qui sépare le bord gauche du bord droit de la boîte contient 50 % des observations. La ligne noire à l'intérieur de la boîte blanche désigne la médiane, qui est la valeur pour laquelle on a 50 % des observations qui ont une valeur inférieure à cette valeur repère, et pour laquelle on a 50 % des observations qui ont une valeur supérieure à cette valeur repère. Les lignes noires en-dehors de la boîte sont les moustaches. Dans le cas présent, la moustache de gauche s'étend jusqu'à la valeur minimale de la variable, et la moustache de droite s'étend jusqu'à la valeur maximale de la variable. Si le minimum (ou le maximum) avait été éloigné de la médiane de plus de 1.5 fois la différence entre Q3 et Q1 (qu'on appelle l'**intervalle interquartile**), l'extrémité de la moustache se serait arrêtée à la dernière valeur avant cette limite, et toute valeur ayant dépassé cette limite aurait été représentée par un point. Pour illustrer ce dernier cas de figure, on peut modifier manuellement une valeur de la variable `Sepal.Length` du jeu de données `iris` de telle sorte à ce qu'il y ait une nouvelle valeur qui soit particulièrement éloignée de la boîte. Une telle valeur s'appelle un *outlier* (cf. Figure 3.4).

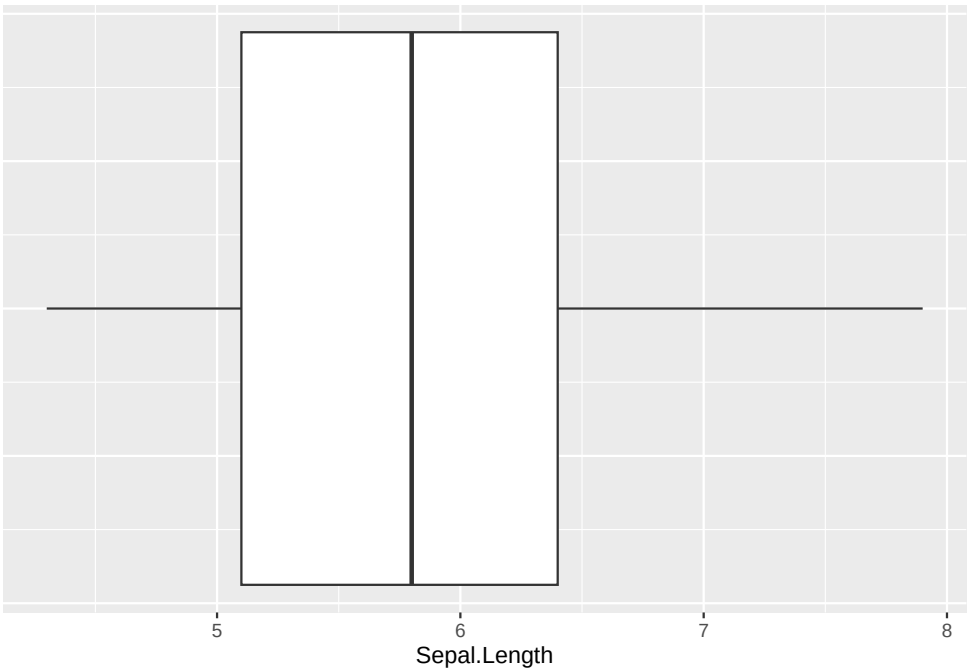

Figure 3.3: Exemple de boîte à moustaches

```
# On modifie ici, pour l'exemple, la valeur de la 3ème observation
# en lui assignant la valeur 12.
iris$Sepal.Length[3] <- 12

# Obtention du graphique
ggplot(data = iris, aes(x = Sepal.Length)) +
  geom_boxplot() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

Une boîte à moustaches a donc notamment l'intérêt de mettre en évidence des valeurs qui apparaissent comme "étranges" par rapport au reste des données. Lorsqu'il semble évident que l'*outlier* est une valeur erronée, ou quand on veut tout simplement vérifier qu'il s'agit d'une erreur ou non, il est intéressant de savoir à quelle observation (i.e., à quel individu dans certains contextes) cette valeur étrange appartient, pour ensuite éventuellement la corriger. Malheureusement, la fonction `geom_boxplot()` ne dispose pas d'argument pour permettre d'identifier facilement à quelle observation appartient cette valeur. Cependant, on peut s'appuyer sur des fonctions créées manuellement dans R pour parvenir à cela. Dans ce cas de figure, le site https://stackoverflow.com est souvent intéressant car riche de solutions. C'est d'ailleurs en provenance de ce site que vient la fonction montrée ci-dessous (voir ici) qui va nous permettre ensuite de savoir, à partir du graphique, à quelle observation correspond cette donnée étrange.

```
is_outlier <- function(x) {
  x < quantile(x, 0.25) - 1.5 * IQR(x) |
    x > quantile(x, 0.75) + 1.5 * IQR(x)
}
```

Voici donc, ci-dessus, à quoi ressemble une fonction à son état brut, avec : le nom de la fonction

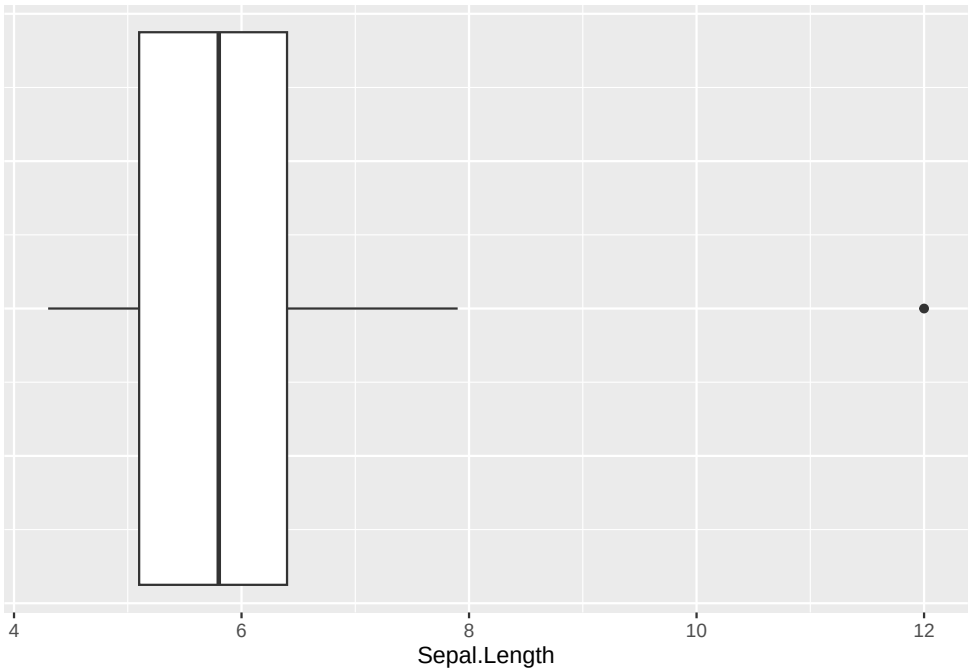

Figure 3.4: Visualisation d'un outlier

à gauche de la flèche d'assignation (`<-`), la commande `function()` qui permet d'amorcer la création de la fonction, et les lignes de code entre les accolades `{ }` qui indiquent les actions que la fonction réalise. La seule chose qu'il faut comprendre à ce stade, c'est que cette fonction, qui va donc s'appeler par la suite `is_outlier()`, a besoin pour fonctionner qu'on lui indique un nom de variable (représenté par la lettre `x` dans le code ci-dessus), et que le résultat de cette fonction sera une nouvelle variable qui contiendra seulement des `TRUE` ou des `FALSE`, en sachant que `TRUE` correspondra au fait que la valeur de la variable étudiée était un *outlier*, et que `FALSE` correspondra au fait que la valeur de la variable étudiée n'était pas un *outlier*. (Notons ici que la définition d'un *outlier* est la même que celle décrite plus haut, à savoir une valeur qui serait éloignée de Q1 ou de Q3 de plus de 1.5 fois l'intervalle interquartile.) Mais regardons concrètement ce que donne cette fonction lorsqu'elle est appliquée à la variable `Sepal.Length` du jeu de données `iris` (NB : La fonction ne marchera que si elle a été activée/créée auparavant) :

```
is_outlier(x = iris$Sepal.Length)
```

```
##   [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [11] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [21] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [31] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [41] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [51] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [71] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [81] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [91] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [101] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [111] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
## [131] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [141] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Quand on regarde bien, on voit que la troisième observation de cette nouvelle variable que l'on vient de créer (seulement de manière temporaire ici car on ne l'a pas assignée à un nom) contient la valeur `TRUE`, ce qui est en accord avec la valeur que nous avons introduite auparavant dans la variable `Sepal.Length`. Le fait d'observer ces valeurs `TRUE` et `FALSE` n'est évidemment pas une stratégie très pratique pour déterminer à quelle observation correspondrait l'*outlier*, et c'est pourquoi l'étape suivante consiste à montrer comment on peut se servir de cette fonction `is_outlier()` pour faire apparaître sur un graphique de boîte à moustaches les observations à qui appartiendraient les valeurs étranges (cf. Figure 3.5).

```
iris |>
  # Ajout d'un numéro id pour les observations
  mutate(id = as.factor(rep(1:50, times = 3)),
         # Création d'une nouvelle variable appelée id_outlier
         id_outlier = ifelse(is_outlier(x = Sepal.Length), id, "")) |>
  ggplot(aes(x = Sepal.Length, y = "")) +
  geom_boxplot() +
  # Ajout des numéros id des outliers
  geom_text(aes(label = id_outlier),  hjust = -1) +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    axis.title.y = element_blank()
  )
```

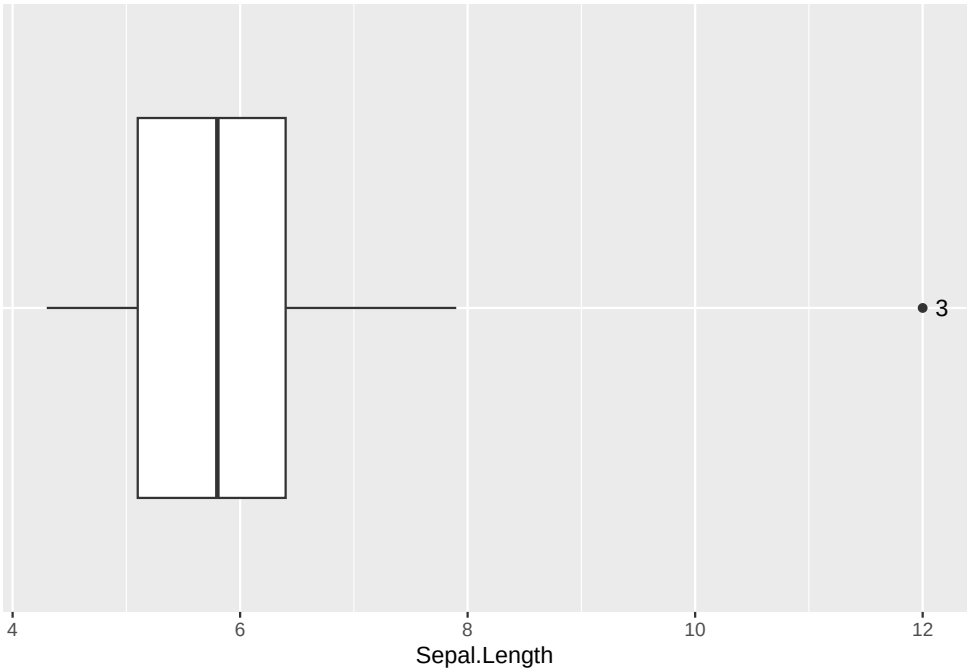

Figure 3.5: Identification d'un outlier

Il y a plusieurs choses à expliquer par rapport au graphique de la Figure 3.5 qui comporte à présent le numéro `id` associé à l'observation pour laquelle nous avions modifié la valeur. Tout

d'abord, il faut noter qu'avant de créer le graphique, nous avons ajouté temporairement au jeu de données, avec la fonction `mutate()`, la variable `id_outlier`. Cette variable a été créée à l'aide de deux fonctions en réalité : la fonction `ifelse()`, et la fonction `is_outlier()` qu'on a présentée succinctement plus haut. Ici, la fonction `ifelse()` a fonctionné comme cela : si la fonction `id_outlier()` renvoyait la valeur `TRUE`, alors on conservait le numéro `id` de la variable `Sepal.Length`, sinon, on ne mettait rien. Cela veut dire que la variable `id_outlier` ne contenait que les numéros `id` pour lesquels la fonction `is_outlier()` avait renvoyé la valeur `TRUE`. Pour visualiser ce qu'il s'est passé, on peut revoir la conséquence du début du code qui a permis de faire le graphique (cf. colonne de droite dans le résultat ci-dessous) :

```
iris |>
  mutate(id = as.factor(rep(1:50, times = 3)),
         id_outlier = ifelse(is_outlier(x = Sepal.Length), id, "")) |>
  dplyr::select(id, Sepal.Length, id_outlier)
```

```
## # A tibble: 150 x 3
##    id    Sepal.Length id_outlier
##    <fct>        <dbl> <chr>
##  1 1              5.1 ""
##  2 2              4.9 ""
##  3 3             12   "3"
##  4 4              4.6 ""
##  5 5              5   ""
##  6 6              5.4 ""
##  7 7              4.6 ""
##  8 8              5   ""
##  9 9              4.4 ""
## 10 10             4.9 ""
## # i 140 more rows
```

Une fois cette procédure réalisée, le reste du code, et notamment la fonction `geom_text()`, a permis d'ajouter des éléments textuels au graphique, en l'occurrence en s'appuyant sur la variable `id_outlier`, tel que configuré avec la fonction `aes()` à l'intérieur de la fonction `geom_text()`. Lorsque la valeur anormale identifiée est effectivement une erreur de saisie dans la base de données, il convient de corriger la valeur avec la fonction d'assignation comme nous l'avons fait précédemment :

```
iris$Sepal.Length[3] <- 4.7
# Le nombre entre crochets désigne la position de l'observation
# dans la variable.
```

On remarque ainsi qu'au-delà de prendre connaissance de la forme de la distribution, passer par ces étapes graphiques permet aussi de s'assurer qu'il n'y a pas eu d'erreur lors de la saisie des données dans la base (du moins, pas d'erreur visible et qui risquerait d'impacter grandement les calculs futurs). **Passer par l'analyse graphique est donc recommandé avant de pouvoir se fier aux résultats numériques que l'on pourrait calculer par la suite, tels que les indices statistiques qui permettent de résumer numériquement une variable.**

À noter que si l'histogramme et la boîte à moustaches sont des graphiques classiques pour étudier la distribution d'une variable quantitative, il est aussi possible avec R de créer ce qu'on appelle un *raincloud plot*, tel qu'illustré sur la Figure 3.6. Un post du blog de Cédric Scherer fournit une description très intéressante et riche de l'intérêt de ce type de graphique. Le *raincloud plot* se compose de trois éléments (cf. figure ci-après) :

- Une aire sous la courbe représentative de la densité de probabilité (*the cloud*), qui représente une estimation de la fréquence d'apparition d'une valeur donnée dans l'échantillon étudié. Cet élément est intéressant pour prendre pleinement conscience de la forme de la distribution. Attention cependant, car les formes des aires et les idées qu'elles donnent des distributions dans la population d'intérêt dépendent beaucoup du nombre d'observations disponibles, et leur utilisation pourrait donc se discuter en présence de peu d'observations.
- Une boîte à moustaches, qui permet de se faire une idée résumée relativement robuste des distributions dans la population d'intérêt (i.e., insensible aux *outliers* et davantage comparable avec d'autres études) ; selon les cas, d'autres statistiques, comme les moyennes et les écarts-types, peuvent être utilisées à la place ou en plus des boîtes à moustaches (Allen et al., 2019).
- Un nuage de points des données individuelles (*the rain*), qui est très intéressant pour avoir une vue précise des valeurs obtenues par les individus, de leurs différences éventuelles, des *outliers* éventuels, voire des *patterns* de distribution éventuellement inattendus ; les données individuelles permettent aussi d'avoir une idée plus claire de la grandeur du nombre d'observations présentes dans le jeu de données (chose non renseignée par les deux éléments précédents).

Parce qu'ils montrent les données individuelles, les *raincloud plots* vont dans le sens du propos de Weissgerber et al. (2015) qui militent pour la disparition des graphiques en forme de simples bâtons de dynamite, lesquels ayant été souvent utilisés par le passé pour montrer des moyennes et écart-types dans les études. Le problème de ces graphiques en forme de bâtons de dynamite est qu'ils peuvent induire en erreur quant à la réelle forme de la distribution et ils limitent les possibilités du lecteur de juger de la pertinence des choix d'analyses inférentielles qui seraient faits par la suite. Les *raincloud plots* peuvent être réalisés relativement facilement à l'aide de la fonction `geom_rain()` du package `ggrain` comme montré ci-dessous.

```
library(ggrain)

ggplot(data = iris, aes(1, Sepal.Length)) +
  geom_rain(
    fill = "grey",
    boxplot.args =  rlang::list2(fill = "white"),
    point.args = rlang::list2(alpha = 0.3, size = 3)
  ) +
  coord_flip(xlim = c(0.9, 1.5)) +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank(),
    axis.title.y = element_blank()
  )
```

Lors de l'analyse de données, différentes formes typiques de distribution peuvent être rencontrées, notamment des formes **gaussiennes**, **asymétriques**, **leptocurtiques**, et **platycurtiques** (Dart & Chatellier, 2003). Ces formes sont illustrées sur la Figure 3.7. Les formes gaussiennes sont observées en présence de variables suivant ce qu'on appelle une **loi normale**. Très souvent, on associe une distribution gaussienne, et donc une loi normale, à une distribution en forme de cloche, bien que l'analogie à la cloche pourrait se discuter. Les formes asymétriques traduisent le fait que la majorité des observations sont concentrées sur une extrémité de l'intervalle des valeurs possibles, et qu'il existe des observations, non majoritaires, avec des valeurs pouvant être très éloignées de la majorité des données, mais seulement d'un seul côté de la distribution. Enfin, les formes leptocurtiques et platycurtiques sont appelées ainsi par comparaison à la forme gaussienne. En présence d'une forme leptocurtique, la distribution

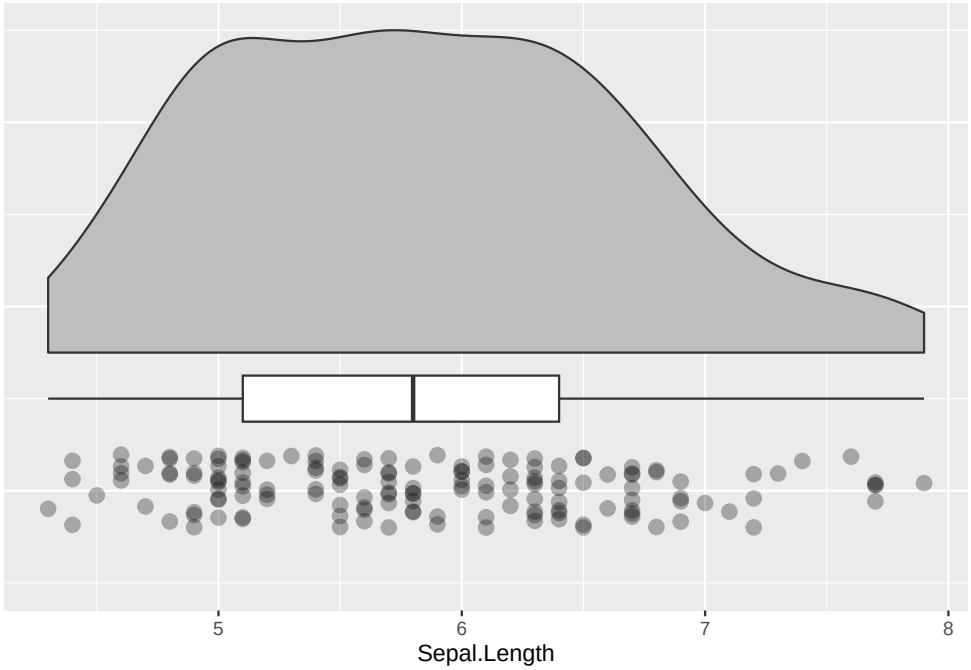

Figure 3.6: Exemple de raincloud plot

s'avère plus pointue, avec des queues (qui sont les extrémités de la distribution) plus longues qu'avec une forme gaussienne. Dans le cadre d'une distribution platycurtique, la distribution est plus aplatie, avec des queues plus courtes qu'avec une forme gaussienne (Dart & Chatellier, 2003). La distribution uniforme est un cas particulier de distribution platycurtique, et c'est cette distribution qui est en réalité montrée sur la Figure 3.7.

La confiance que l'on peut avoir dans des résultats issus d'analyses inférentielles dépend de l'adéquation entre, d'une part, la forme de la distribution de la population d'où vient l'échantillon, et d'autre part, l'indice statistique choisi pour résumer la distribution de l'échantillon, en particulier sa position, pour conduire les analyses inférentielles. Il est donc important de chercher à savoir, graphiquement dans un premier temps, si la distribution de la variable est effectivement gaussienne ou non dans la population d'intérêt, cela à partir de l'échantillon que l'on a sous les yeux. Le fait d'être capable d'identifier les autres formes de distribution peut être aussi important afin de mener des analyses appropriées. Dans les exemples montrés ci-dessus, les distributions ont été créées à partir de 1000 valeurs générées de manière aléatoire de telle sorte à suivre des lois prédéfinies et ainsi illustrer différentes distributions possibles. C'est pour cette raison que les formes de distribution montrées sur la figure ci-dessus sont si nettes. Lorsque l'on travaille dans certains domaines ou contextes, tel qu'avec l'être humain, il peut être compliqué d'obtenir autant de données, et les formes de distribution seront alors plus dures à identifier.

Une fois qu'une première analyse graphique des données a été réalisée, il peut être utile de chercher à résumer de manière numérique la variable. Plusieurs types de statistiques peuvent être utilisés à cet effet : les **indices de position**, les **indices de dispersion**, les **indices d'asymétrie**, et les **indices d'aplatissement**.

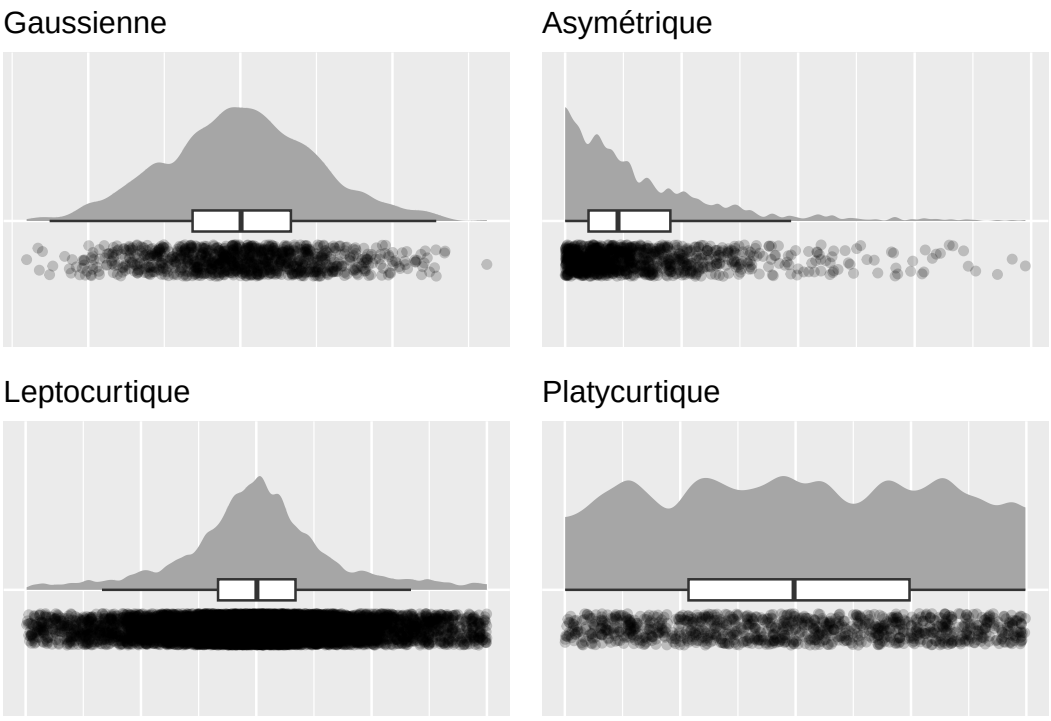

Figure 3.7: Différents types de distributions

3.1.2 Les indices de position

Les indices de position servent à donner un ordre de grandeur de la variable. Autrement dit, ces indices permettent de positionner la variable sur une échelle de valeurs numériques. De plus, ces statistiques peuvent être utilisées pour donner une idée de ce qu'on appelle la **tendance centrale**, c'est-à dire-la valeur typique d'une distribution qui donne une bonne indication de la localisation de la majorité des observations (Rousselet & Wilcox, 2020). Différentes statistiques peuvent être étudiées à cette fin : la **moyenne**, la **médiane**, la **moyenne rognée**, et le **mode**.

La moyenne

Si l'on pose que N est le nombre de valeurs dans une variable (on parle également de **taille** de la variable), que i est la i-ème observation (i-ème position) dans la variable, et que Xi est la valeur associée à la i-ème position, alors le calcul de la moyenne, notée $\overline{X}$, peut être écrit de la manière suivante :

$$\overline{X} = \frac{1}{N} \sum_{i=1}^{N} Xi$$

Cette expression mathématique signifie que la moyenne s'obtient en additionnant ($\sum$) les valeurs allant de la position 1 à la position N de la variable, et en divisant le tout par le nombre total de valeurs N contenues dans la variable. Pour mieux comprendre, prenons par exemple une variable qui ne contiendrait que les cinq premières valeurs de la variable `Sepal.Length` du jeu de données `iris` et qu'on appelle `sample_iris`.

```
sample_iris <- iris$Sepal.Length[1:5]
sample_iris
```

```
## [1] 5.1 4.9 4.7 4.6 5.0
```

La moyenne de la variable `sample_iris` peut alors s'obtenir en divisant la somme des valeurs de la variable par le nombre de valeurs contenues dans la variable, qui est ici de 5 :

```
(5.1 + 4.9 + 4.7 + 4.6 + 5.0) / 5
```

```
## [1] 4.86
```

Évidémment, ce n'est pas très pratique de fonctionner comme cela. Aussi, R permet de calculer directement la moyenne avec la fonction `mean()` :

```
mean(x = sample_iris)
```

```
## [1] 4.86
```

Dans certains cas, il se peut qu'il y ait des valeurs manquantes dans la variable à étudier. Ces valeurs manquantes sont en principe notées **NA**. Introduisons une valeur manquante dans notre variable `sample_iris`, et essayons de calculer la moyenne à nouveau :

```
sample_iris[2] <- NA
sample_iris
```

```
## [1] 5.1  NA 4.7 4.6 5.0
```

```
mean(x = sample_iris)
```

```
## [1] NA
```

Comme nous pouvons le voir ci-dessus, quand il y a une valeur manquante dans la variable, l'utilisation de la fonction `mean()` configurée par défaut renvoie la valeur NA, ce qui signifie que R n'a pas pu calculer de valeur moyenne, ce qui est normal car nous lui avons demandé de le faire en utilisant une valeur inconnue. Dans ce cas là, pour pouvoir faire le calcul de la moyenne seulement à partir des valeurs connues, il faut configurer la fonction pour que les valeurs manquantes ne soient pas considérées pour le calcul. L'argument à configurer dans ce cas là est `na.rm` en lui associant la valeur `TRUE`.

```
mean(x = sample_iris, na.rm = TRUE)
```

```
## [1] 4.85
```

La gestion des valeurs manquantes telle que nous venons de la voir s'effectue de la même manière avec beaucoup de fonctions dans R. Ainsi, il s'agira de fonctionner de la même manière avec la plupart des fonctions de base que nous pourrons rencontrer par la suite et qui seront concernées par ce genre de problème. Par ailleurs, si les exemples ci-dessus ont été réalisés à l'aide d'une variable isolée (i.e., ne faisant pas partie d'un tableau de données), c'est évidemment possible d'utiliser la fonction `mean()` directement à partir d'un tableau de données :

```
mean(x = iris$Sepal.Length)
```

```
## [1] 5.843333
```

La moyenne, c'est en quelque sorte le "centre de gravité" de la variable (Navarro, 2018). L'un de ses intérêts est que son calcul prend en compte toutes les informations contenues dans la variable, ce qui est utile quand on a relativement peu de données (Navarro, 2018). En revanche, un inconvénient est qu'elle est très sensible aux valeurs extrêmes, et en particulier aux valeurs qui seraient particulièrement basses ou particulièrement élevées par rapport à la majorité des valeurs de la variable (il s'agirait ici d'*outliers*), et cela est d'autant plus vrai lorsque la taille de l'échantillon étudié est faible. Dans ce dernier cas, il y a donc un risque assez important que la moyenne ne représente pas bien la tendance centrale, c'est-à-dire la valeur ou la zone de valeurs où sont situées la majorité des observations. Ce risque existe aussi même avec des tailles d'échantillon relativement importantes lorsque la distribution est asymétrique, comme illustré sur la Figure 3.8. Sur cette figure, on peut voir qu'avec une distribution gaussienne, la moyenne correspond parfaitement à la tendance centrale. En revanche, avec une distribution asymétrique à droite, on voit que la moyenne est "tirée vers la droite" par rapport à la tendance centrale sous l'effet des valeurs, certes moins nombreuses, mais d'une grandeur plus importante.

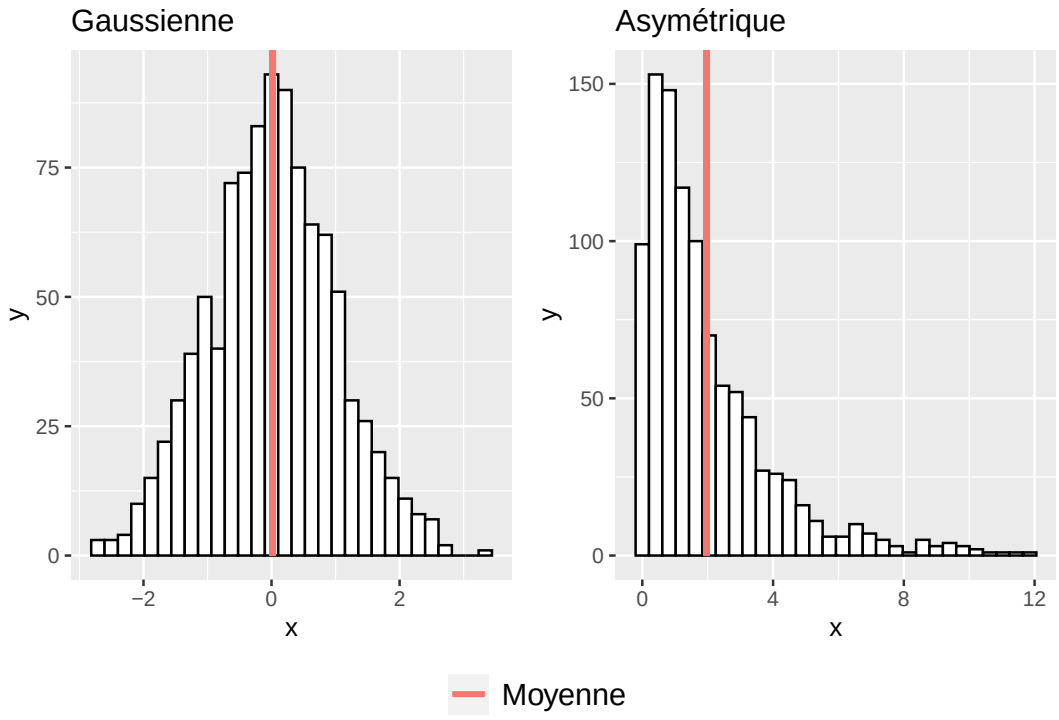

Figure 3.8: Effet de la forme de la distribution sur la position de la moyenne

La médiane

La médiane est le deuxième indice de position que l'on considère régulièrement lorsqu'il s'agit de résumer numériquement une variable quantitative. Pour l'obtenir, il faut d'abord classer les valeurs de la variable selon un ordre croissant. La médiane pour une variable de taille N est alors la valeur correspondant au rang $(N + 1) / 2$. Ainsi, la médiane désigne la valeur qui sépare les valeurs de la variable en deux groupes d'observations de même taille (Chatellier & Durieux, 2003). Dans l'exemple ci-dessous, il y a cinq observations, et donc cinq valeurs, qui ont été triées

par ordre croissant. La médiane est alors la valeur correspondant au rang (5 + 1) / 2 = 3, soit 4.9.

```
##   1   2   3   4   5
## 4.6 4.7 4.9 5.0 5.1
```

Dans le cas où le nombre d'observations contenues dans la variable étudiée serait un nombre pair, la médiane s'obtiendrait différemment. En effet, avec une variable qui contiendrait par exemple six valeurs, la médiane serait associée, selon la méthode expliquée ci-dessus, au rang (6 + 1) / 2 = 3.5, or ce rang n'existe pas. Dans ce cas, la médiane s'obtient en faisant la moyenne des deux nombres du milieu. Par exemple, ci-dessous, la médiane correspond à la moyenne des valeurs de la 3ème et de la 4ème observation, ce qui donne 4.95.

```
##   1   2   3   4   5   6
## 4.6 4.7 4.9 5.0 5.1 5.4
```

Dans R, la médiane d'une variable s'obtient facilement à l'aide de la fonction `median()`.

```
vec <- c(4.6, 4.7, 4.9, 5.0, 5.1, 5.4)
median(x = vec)
```

```
## [1] 4.95
```

Contrairement à la moyenne, la médiane prend donc en compte moins d'informations relatives aux données. Toutefois, en tant que valeur "du milieu", la médiane présente l'intérêt de ne pas être influencée par les valeurs extrêmes. En raison de cela, la médiane est susceptible de mieux refléter la tendance centrale que la moyenne en présence de petits échantillons avec des *outliers*, ou en présence d'une forme de distribution asymétrique. Ce dernier cas est illustré sur la Figure 3.9.

La moyenne rognée

Parfois, il est possible de rencontrer ce qu'on appelle la moyenne rognée. Le principe est ici de calculer la moyenne non pas en prenant en compte toutes les valeurs de la variable, mais en écartant un certain pourcentage des valeurs situées à l'extrémité basse et à l'extrémité haute du classement des valeurs de la variable. Cette procédure consiste à pouvoir calculer une moyenne qui ne serait pas influencée par des *outliers*. Pour pouvoir caculer une moyenne rognée, il faut à nouveau utiliser la fonction `mean()`, en précisant cette fois l'argument `trim` avec la valeur du pourcentage de données que l'on veut rogner aux extrémités de la variable :

```
mean(x = iris$Sepal.Length, trim = 0.05)
```

Dans l'exemple de code précédent, la fonction a été configurée pour rogner 5 % des observations situées à chaque extrémité de la variable (i.e., les observations en-dessous du 5ème percentile, et celles au-dessus du 95ème percentile). Notons que lorsque l'argument `trim` est mis à 0 (ce qui est son paramétrage par défaut), cela consiste à calculer la moyenne normale, et que lorsque l'argument `trim` est mis à 0.50, cela revient à calculer la médiane puisque la fonction supprime alors 50 % des observations de part et d'autre du milieu de la variable.

Le mode

Le mode désigne la valeur qui est la plus fréquemment retrouvée dans une variable. Il n'existe pas de fonction de base dans R pour pouvoir déterminer directement le mode et pour connaître le nombre de fois que le mode apparaît dans la variable. Toutefois, nous pouvons utiliser le package

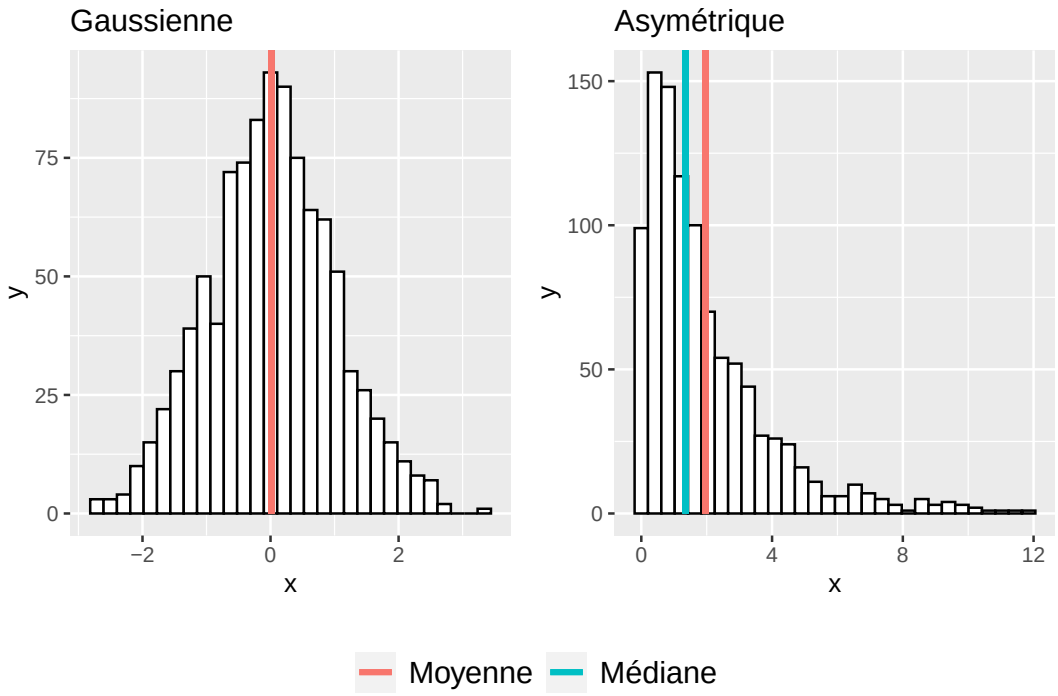

Figure 3.9: Effet de la forme de la distribution sur la position de la médiane

`lsr` crée par Danielle Navarro (2018) pour retrouver ces informations en présence d'une variable quantitative. Une fois le package `lsr` installé puis chargé, nous pouvons utiliser la fonction `modeOf()` pour déterminer le mode, et la fonction `maxFreq()` pour savoir à quelle fréquence revient le mode dans la variable.

```
library(lsr)

# Détermination du mode
modeOf(x = iris$Sepal.Length)
```

```
## [1] 5
```

```
# Détermination de la fréquence du mode
maxFreq(x = iris$Sepal.Length)
```

```
## [1] 10
```

Bien que cela ne soit pas le cas dans l'exemple ci-dessus, il faut comprendre qu'il est à tout à fait possible d'avoir plusieurs modes si plusieurs valeurs reviennent à des fréquences similaires dans la variable. Dans ce cas, la fonction `modeOf()` affichera les différentes valeurs de mode, et la fonction `maxFreq()` continuera de n'afficher qu'une seule valeur de fréquence puisque par définition, le mode désigne la valeur associée à la fréquence d'apparition maximale dans la variable, or il ne peut n'y avoir qu'une seule fréquence maximale… Cela signifie qu'une variable peut contenenir autant de modes que de valeurs si chaque valeur n'est représentée qu'une seule fois dans la variable. Cet inconvénient est probablement l'une des raisons pour lesquelles le mode n'est que très peu utilisé, si ce n'est jamais utilisé, pour décrire la tendance centrale d'une variable quantitative.

3.1.3 Les indices de dispersion

Les indices de dispersion permettent de rendre compte de la manière selon laquelle les observations sont étalées, ou réparties, autour des indices de position. Plusieurs statistiques sont disponibles pour caractériser la dispersion, à savoir : l'**étendue**, l'**écart-type**, et l'**intervalle interquartile**.

L'étendue

L'étendue est la mesure la plus simple de la dispersion des données contenues dans une variable. Elle est exprimée avec la plus petite valeur (minimum) et la plus grande valeur (maximum) observée, ou alors parfois avec la différence entre ces deux valeurs. Par exemple, dans la variable ci-dessous dont les données ont été classées en ordre croissant, le minimum est 4.5, le maximum est 20.2, et l'étendue peut être donnée par l'intervalle [4.5 – 20.2]. Pour obtenir ces différents résultats dans R, il est possible d'utiliser les fonctions `min()`, `max()`, et `range()`. L'amplitude de l'intervalle serait ici de : 20.2-4.5 = 15.7.

```
vec <- c(4.5, 7.8, 10.8, 13.9, 20.2)
min(vec)
```

```
## [1] 4.5
```

```
max(vec)
```

```
## [1] 20.2
```

```
range(vec)
```

```
## [1]  4.5 20.2
```

L'écart-type

L'écart-type est une statistique qui donne une idée de la mesure selon laquelle les valeurs de la variable sont éloignées de la moyenne. Pour calculer l'écart-type, il faut en réalité d'abord calculer la variance σ^2, dont le calcul est le suivant :

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(Xi - \overline{X})^2$$

Cette formule signifie que pour obtenir la variance, il faut d'abord faire la somme des carrés des différences entre chaque valeur et la moyenne de la variable. Cela fait, la variance s'obtient en divisant cette somme de carrés par le nombre N de valeurs de la variable. L'écart-type σ, c'est alors la racine carrée de la variance :

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Xi - \overline{X})^2}$$

Ces calculs sont valides lorsque l'on a en sa possession les données de toute la population que l'on souhaite étudier. Toutefois, lorsque l'on a en sa possession des données issues seulement d'un échantillon de la population, ces calculs biaisent les estimations de la variance et de l'écart-type correspondant à la population étudiée. Cette notion de biais traduit le fait que lorsqu'on répète un grand nombre de fois le calcul de la variance et de l'écart-type à partir, à chaque fois, d'échantillons de population différents, on a en moyenne un décalage entre la valeur de

l'estimation et la réelle valeur de la variance et de l'écart-type de la population. Ce décalage systématique est tel qu'il convient dans ce cas là de diviser la somme des carrés des différences $(Xi - \overline{X})$ par $N - 1$ plutôt que par N (Grenier, 2007). La formule de l'écart-type non biaisé, noté $\hat{\sigma}$, est alors la suivante :

$$\hat{\sigma} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(Xi - \overline{X})^2}$$

L'écart-type est la mesure de dispersion classiquement associée à la moyenne. Si, pour un échantillon, on note une moyenne $\overline{X}$ et un écart-type $\hat{\sigma}$, alors le résumé d'une variable à l'aide de ces statistiques s'écrit comme suit : $\overline{X} \pm \hat{\sigma}$. Lorsque l'écart-type est divisé par la moyenne arithmétique de la variable, on obtient une valeur appelée **coefficient de variation**. Avec le logiciel R, les fonctions pour calculer la variance et l'écart-type non biaisés sont respectivement `var()` et `sd()`.

```
vec <- c(4.5, 7.8, 10.8, 13.9, 20.2)
var(x = vec)
```

```
## [1] 36.153
```

```
sd(x = vec)
```

```
## [1] 6.012736
```

L'intervalle interquartile

L'intervalle interquartile désigne l'étendue entre le premier quartile (Q1) et le troisième quartile (Q3) d'une variable. Comme expliqué auparavant dans le cadre de la boîte à moustaches, Q1 et Q3 désignent respectivement les valeurs en-dessous desquelles 25 % et 75 % des observations de la variable se trouvent (Chatellier & Durieux, 2003). Pour un échantillon de taille N, la procédure classique pour calculer les quartiles est différente selon que le rapport N / 4 est un nombre entier ou non. Lorsque ce rapport n'est pas un nombre entier, Q1 est la valeur correspondant au rang immédiatement supérieur à N / 4. Par exemple, pour la variable ci-dessous, qui a une taille N de 5 valeurs, le rapport N / 4 est égal à 1.25. Q1 est donc la valeur correspondant au rang directement supérieur, c'est-à-dire au rang 2, qui est ici la valeur 7.8.

```
##    1    2    3    4    5
##  4.5  7.8 10.8 13.9 20.2
```

Lorsque le rapport N / 4 est un nombre entier, Q1 correspond à la moyenne des valeurs associées respectivement aux rangs N / 4 et (N / 4) + 1. Par exemple, pour la variable ci-dessous, qui a une taille N de 8 valeurs, le rapport N / 4 est à égal 2. Q1 est donc la moyenne des valeurs correspondant au rang 2 et au rang 3 (i.e., les valeurs 7.8 et 10.8), qui équivaut ici à 9.3.

```
##    1    2    3    4    5    6    7    8
##  4.5  7.8 10.8 13.9 20.2 25.6 37.5 43.9
```

La démarche demeure la même pour déterminer Q3, à ceci près qu'on utilise le nombre $3N$ et non plus le nombre N pour les calculs (Labreuche, 2010). Cette méthode de calcul est en principe à privilégier en présence d'une variable discrète. Si l'on souhaite obtenir les quartiles selon cette méthode avec le logiciel R, il faut utiliser la fonction `quantile()` de la manière suivante :

```
quantile(x = vec, probs = c(0.25, 0.75), type = 2)
```

```
##   25%   75%
##  9.30 31.55
```

On remarque ici que la fonction `quantile()` a plusieurs arguments. L'argument `probs` désigne les quantiles que l'on souhaite obtenir. Le quantile 0.25 correspond à Q1, et le quantile 0.75 correspond à Q3. L'argument `type` permet de configurer le type de calcul à effectuer pour obtenir les valeurs des quantiles recherchés. L'indication du chiffre 2 pour l'argument `type` permet d'obtenir les quantiles selon la méthode de calcul présentée ci-dessus, qui, comme nous l'avons précisé, est dédiée à l'étude d'une variable quantitative discrète. Par défaut, en revanche, la fonction `quantile()` utilise le chiffre 7 pour l'argument `type`, ce qui renvoie à une méthode de calcul des quantiles qui serait davantage pertinente pour étudier des variables quantitatives continues. Comparons les résultats obtenus avec les deux méthodes de calcul :

Tableau 3.1: Comparaison des quantiles obtenus selon différentes configurations de la fonction quantile()

Quantile	Type 2	Type 7
0.25	9.30	10.050
0.75	31.55	28.575

On remarque que les résultats de la fonction `quantile()` sont différents selon la configuration de l'argument `type`. Le choix de la configuration est donc important. Pour comprendre comment R a calculé les valeurs associées aux quantiles 0.25 et 0.75 dans le cadre de la seconde méthode (i.e., avec `type` = 7), regardons le tableau ci-dessous.

Tableau 3.2: Quantiles d'une variable quantitative continue

Rang	Quantile	Valeur
1	0.0000000	4.5
2	0.1428571	7.8
3	0.2857143	10.8
4	0.4285714	13.9
5	0.5714286	20.2
6	0.7142857	25.6
7	0.8571429	37.5
8	1.0000000	43.9

Le tableau montre les données sur lesquelles R s'est appuyé pour déterminer les valeurs des quantiles recherchés (i.e., les quantiles 0.25 et 0.75 pour Q1 et Q3, respectivement). Les données du tableau sont bien celles relatives à notre variable `vec`, dont on peut reconnaître les valeurs dans la colonne de droite du tableau. La colonne "Quantile" montre les fractions (ou portions) de la variable `vec` associées aux valeurs de la variable compte tenu de leurs rangs respectifs. Par exemple, la valeur 25.6, dont le rang est 6, correspond au quantile 0.71 (approximativement).

Cela veut dire que 71 % des observations ont une valeur inférieure ou égale à 25.6. Il faut savoir qu'il existe en réalité plusieurs manières de déterminer la valeur du quantile que représente chaque valeur. Dans le cas présent, le quantile représenté, que l'on va noter q, a été déterminé selon la formule suivante :

$$q = (k - 1)/(N - 1)$$

Dans le calcul ci-dessus, k désigne le rang de la valeur considérée, et N désigne la taille de la variable étudiée (i.e., le nombre total de valeurs). Comme on peut le voir dans le tableau ci-dessus, cette méthode de calcul conduit nécessairement à attribuer le quantile 0 à la valeur de rang 1, et la quantile 1 à la valeur de rang N. Lorsque le nombre de valeurs fait que les quantiles 0.25 et 0.75 n'existent pas, R réalise une interpolation de la valeur correspondant au quantile recherché, cela à partir des quantiles qui existent et qui encadrent le quantile recherché, ainsi qu'à partir des valeurs correspondant à ces quantiles. Dans le cas présent, il s'agit plus précisément d'une interpolation linéaire. Voyons sur la Figure 3.10 en quoi cela consiste.

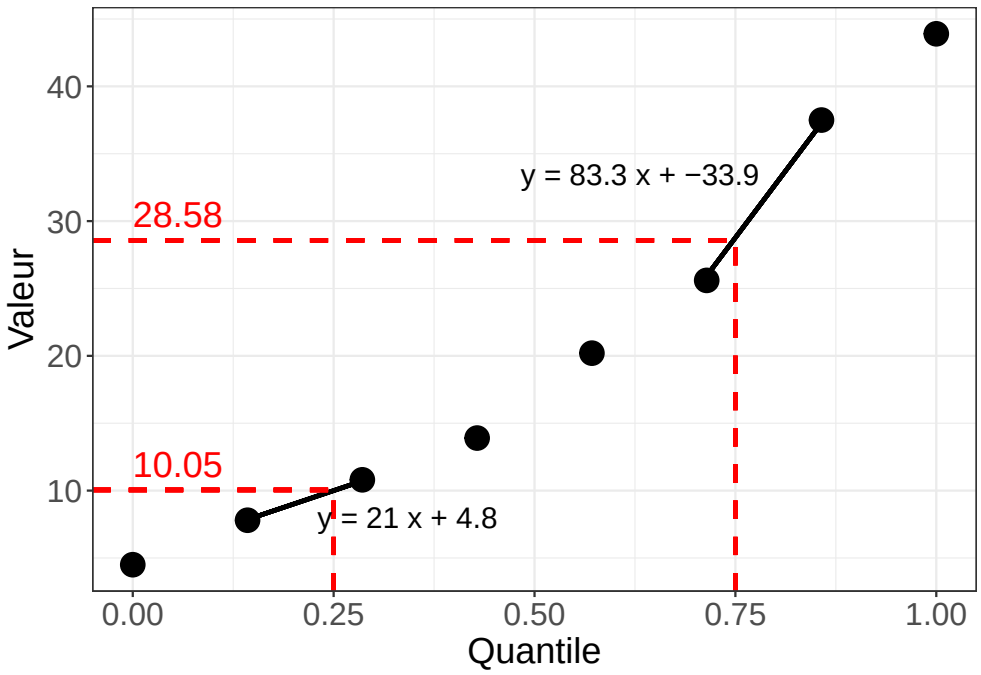

Figure 3.10: Détermination des quartiles Q1 et Q3 avec une variable quantitative continue

La figure représente les valeurs de la variable en fonction des quantiles qui leur correspondent. Les segments de couleur noire montrés sur la figure représentent les droites d'équation utilisées pour le calcul des valeurs correspondant aux quantiles 0.25 (Q1) et 0.75 (Q3), qui ne sont pas représentés initialement dans la variable étudiée. Ces droites d'équation relient les points dont les abscisses sont celles qui encadrent directement les quantiles recherchés. Ainsi, pour trouver la valeur correspondant au quantile 0.25, il a suffi de résoudre l'équation y = 21x + 4.8, en remplaçant x par 0.25. De manière analogue, pour trouver la valeur correspondant au quantile 0.75, il a suffi de résoudre l'équation y = 83.3x - 33.9 en remplaçant x par 0.75. Les solutions de ces équations sont montrées en rouge sur la partie gauche de la figure. On retrouve bien les valeurs associées aux quantiles recherchés et qui avaient été initialement obtenues avec la configuration par défaut de la fonction `quantile()`.

Les quartiles Q1 et Q3 sont les mesures de dispersion classiquement associées à la médiane. Si on note une médiane m et l'intervalle interquartile (Q1 - Q3), alors le résumé d'une variable à l'aide de ces statistiques s'écrit comme suit : m (Q1 - Q3).

3.1.4 Les indices d'asymétrie et d'aplatissement

Le coefficient d'asymétrie (skewness)

Le fait qu'une distribution soit asymétrique désigne le fait que les observations sont réparties de manière inégale de part et d'autre du milieu de la distribution. L'indice statistique qui permet de rendre compte du niveau d'asymétrie est le **coefficient d'asymétrie**, ou *skewness* en anglais. Ce coefficient peut être obtenu à l'aide de la fonction `skewness()` du package `e1071`, qui n'existe pas dans la base de R et qu'il convient d'installer et de charger pour l'utiliser.

```
library(e1071)
skewness(x = iris$Sepal.Width, type = 3)
```

```
## [1] 0.3126147
```

Comme on peut le voir dans l'aide associée à la fonction `skewness()`, il existe en réalité plusieurs méthodes de calcul du coefficient d'asymétrie (noté γ_1 ci-dessous). La méthode de `type` 3, qui est celle configurée par défaut pour cette fonction, consiste à faire le calcul suivant :

$$\gamma_1 = \frac{1}{\hat{\sigma}^3} \frac{\sum_{i=1}^{N} (Xi - \overline{X})^3}{N}$$

Dans ce calcul, $\hat{\sigma}$ désigne l'écart-type non biaisé de la variable, et N désigne la taille de la variable. Avec cette méthode, on obtient un coefficient négatif lorsque la distribution est asymétrique à gauche (longue queue vers la gauche), un coefficient de 0 lorsque la distribution est parfaitement symétrique, et un coefficient positif lorsque la distribution est asymétrique à droite (longue queue vers la droite). Ceci est illustré sur la Figure 3.11.

Pour aller plus loin… Joanes et Gill (1998) ont montré que lorsqu'il s'agit d'estimer le degré d'asymétrie de la distribution relative à la population étudiée, et cela à partir de l'échantillon observé, certaines méthodes de calcul du coefficient d'asymétrie peuvent être plus fiables que d'autres.

Dans le cas où la distribution des valeurs dans la population étudiée suivrait une loi normale, la méthode par défaut présentée ci-dessus serait la plus fiable pour estimer le niveau d'asymétrie lorsque l'échantillon observé est de petite taille ($N < 50$). Cependant, avec des échantillons de grande taille, les méthodes se valeraient.

Dans le cas où la distribution des valeurs de la population étudiée ne suivrait pas une loi normale, et qu'elle s'avèrerait très asymétrique, la méthode de `type` 2 proposée avec la fonction `skewness()` serait la plus fiable, particulièrement en présence d'échantillons de petite taille.

Le coefficient d'applatissement (kurtosis)

Le fait qu'une distribution soit aplatie désigne le fait que la forme de la distribution présente une courbure relativement plate avec des queues de distribution relativement courtes. On parle alors de distribution platycurtique. À l'inverse, lorsque la distribution est pointue avec des queues plus longues, on parle de distribution leptocurtique. L'indice statistique qui permet de rendre compte du degré d'aplatissement est le **coefficient d'aplatissement**, ou *kurtosis* en anglais. Ce coefficient peut être obtenu à l'aide de la fonction `kurtosis()` du package `e1071`.

```
library(e1071)
kurtosis(x = iris$Sepal.Width, type = 3)
```

```
## [1] 0.1387047
```

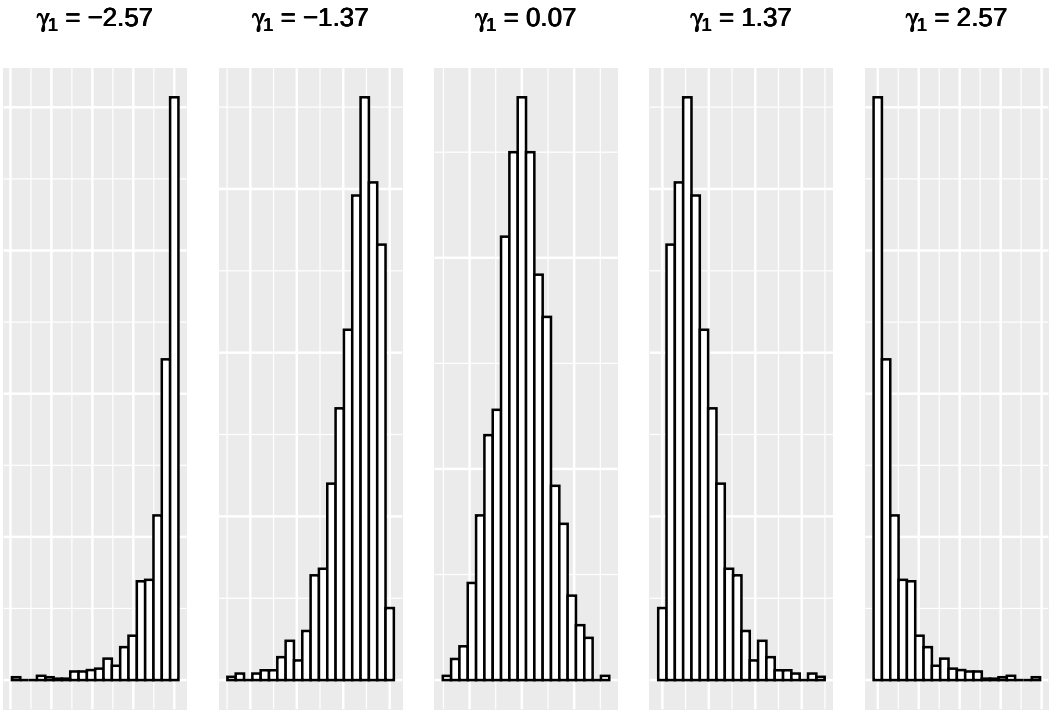

Figure 3.11: Valeur du Skewness selon la forme de la distribution

Comme on peut le voir dans l'aide associée à la fonction `kurtosis()`, il existe en réalité plusieurs méthodes de calcul du coefficient d'aplatissement (noté γ_2 ci-dessous). La méthode de `type` 3, qui est celle configurée par défaut pour cette fonction, consiste à faire le calcul suivant :

$$\gamma_2 = \frac{1}{\hat{\sigma}^4} \frac{\sum_{i=1}^{N} (Xi - \overline{X})^4}{N} - 3$$

Dans ce calcul, $\hat{\sigma}$ désigne l'écart-type non biaisé de la variable, et N désigne la taille de la variable. Avec cette méthode, on obtient un coefficient négatif lorsque la distribution est particulièrement aplatie par rapport à une distribution suivant une loi normale (distribution platycurtique), un coefficient de 0 lorsque la distribution suit une loi normale (distribution mésocurtique), et un coefficient positif lorsque la distribution est particulièrement pointue par rapport à une loi normale (distribution leptocurtique). Ceci est illustré sur la Figure 3.12.

Pour aller plus loin… Comme avec le coefficient d'asymétrie, Joanes et Gill (1998) ont montré que lorsqu'il s'agit d'estimer le degré d'aplatissement de la distribution relative à la population étudiée, et cela à partir de l'échantillon observé, certaines méthodes de calcul du coefficient d'aplatissement peuvent être plus fiables que d'autres.

Dans le cas où la distribution des valeurs dans la population étudiée suivrait une loi normale, la méthode de `type` 1 proposée avec la fonction `kurtosis()` serait la plus fiable pour estimer le niveau d'aplatissement lorsque l'échantillon observé est de petite taille ($N < 50$). Cependant, la méthode par défaut présentée plus haut fournirait des résultats relativement proches de ceux obtenus avec la méthode de `type` 1. De plus, avec des échantillons de grande taille, toutes les méthodes se valeraient.

Dans le cas où la distribution des valeurs de la population étudiée ne suivrait pas une loi normale, et s'avèrerait très asymétrique, la méthode de `type` 2 proposée avec la fonction `kurtosis()` serait la plus fiable, particulièrement en présence d'échantillons de petite taille.

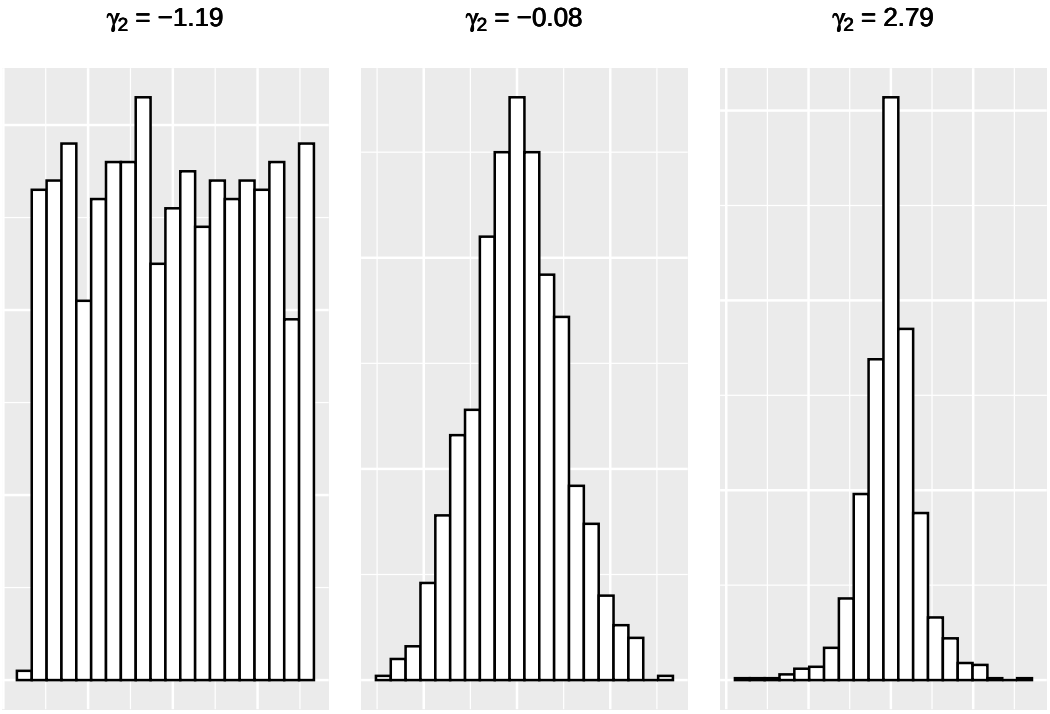

Figure 3.12: Valeur du Kurtosis selon la forme de la distribution

3.1.5 Fonctions pour obtenir un récapitulatif des statistiques descriptives

Il existe plusieurs fonctions pour avoir une vue d'ensemble des statistiques généralement utilisées pour explorer et résumer une variable quantitative. Une fonction particulièrement intéressante est la fonction `describe()` du package `psych`.

```
library(psych)
```

La fonction `describe()` peut être utilisée sur une variable donnée :

```
psych::describe(x = iris$Sepal.Width, quant = c(0.25, 0.75))
```

```
##    vars   n mean   sd median trimmed  mad min max range skew kurtosis
## X1    1 150 3.06 0.44      3    3.04 0.44   2 4.4   2.4 0.31     0.14
##      se Q0.25 Q0.75
## X1 0.04   2.8   3.3
```

La fonction `describe()` peut être aussi utilisée sur un jeu de données entier. (Attention, les résumés numériques fournis pour les variables qualitatives n'auront pas de sens ; les variables qualitatives détectées sont indiquées avec un astérisque dans le tableau de résultats.)

```
psych::describe(x = iris, quant = c(0.25, 0.75))
```

```
##              vars   n mean   sd median trimmed  mad min max range
```

```
## Sepal.Length    1 150 5.84 0.83   5.80    5.81 1.04 4.3 7.9   3.6
## Sepal.Width     2 150 3.06 0.44   3.00    3.04 0.44 2.0 4.4   2.4
## Petal.Length    3 150 3.76 1.77   4.35    3.76 1.85 1.0 6.9   5.9
## Petal.Width     4 150 1.20 0.76   1.30    1.18 1.04 0.1 2.5   2.4
## Species*        5 150 2.00 0.82   2.00    2.00 1.48 1.0 3.0   2.0
##                skew kurtosis   se Q0.25 Q0.75
## Sepal.Length  0.31    -0.61 0.07   5.1   6.4
## Sepal.Width   0.31     0.14 0.04   2.8   3.3
## Petal.Length -0.27    -1.42 0.14   1.6   5.1
## Petal.Width  -0.10    -1.36 0.06   0.3   1.8
## Species*      0.00    -1.52 0.07   1.0   3.0
```

On retrouve la plupart des statistiques que nous avons vues jusqu'à présent, notamment le *skewness* et le *kurtosis* qui ont été ici calculés avec la méthode par défaut du package `e1071`. Pour changer la méthode de calcul de ces deux coefficients, il suffit de modifier l'argument `type` de la fonction, comme dans le cadre de l'utilisation du package `e1071` et des fonctions `skewness()` et `kurtosis()` associées. On notera cependant qu'il ne semble pas possible de modifier la méthode de calcul des quantiles, qui sont ici calculés selon la méthode par défaut configurée telle qu'avec la fonction `quantile()`.

Enfin, il est aussi possible d'obtenir ces récapitulatifs numériques en fonction des modalités d'une variable qualitative du jeu de données, grâce à la fonction `describeBy()` du package `psych`. Dans l'exemple ci-dessous, la variable qualitative est indiquée grâce à l'argument `group`.

```
psych::describeBy(x = iris, quant = c(0.25, 0.75), group = iris$Species)
```

```
##
##  Descriptive statistics by group
## group: setosa
##              vars  n mean   sd median trimmed  mad min max range skew
## Sepal.Length    1 50 5.01 0.35    5.0    5.00 0.30 4.3 5.8   1.5 0.11
## Sepal.Width     2 50 3.43 0.38    3.4    3.42 0.37 2.3 4.4   2.1 0.04
## Petal.Length    3 50 1.46 0.17    1.5    1.46 0.15 1.0 1.9   0.9 0.10
## Petal.Width     4 50 0.25 0.11    0.2    0.24 0.00 0.1 0.6   0.5 1.18
## Species*        5 50 1.00 0.00    1.0    1.00 0.00 1.0 1.0   0.0  NaN
##              kurtosis   se Q0.25 Q0.75
## Sepal.Length    -0.45 0.05   4.8  5.20
## Sepal.Width      0.60 0.05   3.2  3.68
## Petal.Length     0.65 0.02   1.4  1.58
## Petal.Width      1.26 0.01   0.2  0.30
## Species*          NaN 0.00   1.0  1.00
## ------------------------------------------------------
## group: versicolor
##              vars  n mean   sd median trimmed  mad min max range
## Sepal.Length    1 50 5.94 0.52   5.90    5.94 0.52 4.9 7.0   2.1
## Sepal.Width     2 50 2.77 0.31   2.80    2.78 0.30 2.0 3.4   1.4
## Petal.Length    3 50 4.26 0.47   4.35    4.29 0.52 3.0 5.1   2.1
## Petal.Width     4 50 1.33 0.20   1.30    1.32 0.22 1.0 1.8   0.8
## Species*        5 50 2.00 0.00   2.00    2.00 0.00 2.0 2.0   0.0
##               skew kurtosis   se Q0.25 Q0.75
## Sepal.Length  0.10    -0.69 0.07  5.60   6.3
## Sepal.Width  -0.34    -0.55 0.04  2.52   3.0
## Petal.Length -0.57    -0.19 0.07  4.00   4.6
## Petal.Width  -0.03    -0.59 0.03  1.20   1.5
```

Tableau 3.3: Data summary

Name	iris
Number of rows	150
Number of columns	5
Column type frequency:	
factor	1
numeric	4
Group variables	None

```
## Species*       NaN      NaN 0.00  2.00   2.0
## ------------------------------------------------------
## group: virginica
##              vars  n mean   sd median trimmed  mad min max range
## Sepal.Length    1 50 6.59 0.64   6.50    6.57 0.59 4.9 7.9   3.0
## Sepal.Width     2 50 2.97 0.32   3.00    2.96 0.30 2.2 3.8   1.6
## Petal.Length    3 50 5.55 0.55   5.55    5.51 0.67 4.5 6.9   2.4
## Petal.Width     4 50 2.03 0.27   2.00    2.03 0.30 1.4 2.5   1.1
## Species*        5 50 3.00 0.00   3.00    3.00 0.00 3.0 3.0   0.0
##               skew kurtosis   se Q0.25 Q0.75
## Sepal.Length  0.11    -0.20 0.09  6.23  6.90
## Sepal.Width   0.34     0.38 0.05  2.80  3.18
## Petal.Length  0.52    -0.37 0.08  5.10  5.88
## Petal.Width  -0.12    -0.75 0.04  1.80  2.30
## Species*       NaN      NaN 0.00  3.00  3.00
```

Notons aussi l'existence de la fonction `skim()` du package `skimr` pour obtenir un résumé clair des types de variables et des statistiques principales, ainsi que des valeurs manquantes :

```
library(skimr)

# Analyse de l'ensemble des variables
skim(iris)
```

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Species	0	1	FALSE	3	set: 50, ver: 50, vir: 50

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Sepal.Length	0	1	5.84	0.83	4.3	5.1	5.80	6.4	7.9	
Sepal.Width	0	1	3.06	0.44	2.0	2.8	3.00	3.3	4.4	
Petal.Length	0	1	3.76	1.77	1.0	1.6	4.35	5.1	6.9	
Petal.Width	0	1	1.20	0.76	0.1	0.3	1.30	1.8	2.5	

```
# Analyse d'une variable (Sepal.Length) par groupe (Species)
iris |>
  dplyr::select(Species, Sepal.Length) |>
  dplyr::group_by(Species) |>
  skim()
```

Tableau 3.4: Data summary

Name	dplyr::group_by(dplyr::se...
Number of rows	150
Number of columns	2
_____	
Column type frequency:	
numeric	1
_____	
Group variables	Species

Variable type: numeric

skim_variable	Species	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Sepal.Length	setosa	0	1	5.01	0.35	4.3	4.80	5.0	5.2	5.8	
Sepal.Length	versicolor	0	1	5.94	0.52	4.9	5.60	5.9	6.3	7.0	
Sepal.Length	virginica	0	1	6.59	0.64	4.9	6.23	6.5	6.9	7.9	

3.1.6 Quelles statistiques choisir pour résumer une variable quantitative dans un rapport ?

Les statistiques les plus couramment utilisées pour résumer une variable quantitative sont les paramètres de position (moyenne et médiane principalement) en lien avec les paramètres de dispersion correspondants. Aucun paramètre de position ne surpasse les autres dans toutes les situations. Le choix du paramètre de position, en lien avec le paramètre de dispersion associé, dépend de l'objectif de l'analyse.

Lorsqu'il s'agit de simplement décrire une distribution des données obtenues à titre exploratoire, certains auteurs proposent d'utiliser les trois paramètres de position (moyenne, médiane, mode), en sachant que la médiane, d'un point de vue purement descriptif, peut être le paramètre le plus adapté dans de nombreuses situations (Gonzales & Ottenbacher, 2001). Dans le cas où la distribution s'avèrerait plutôt gaussienne, la moyenne et l'écart-type sont intéressants car dans ce cas, on sait que :

1. Approximativement **68.3 %** des observations sont comprises dans l'intervalle [$\overline{X}$ - 1 $\hat{\sigma}$; $\overline{X}$ + 1 $\hat{\sigma}$],
2. Approximativement **95.5 %** des observations sont comprises dans l'intervalle [$\overline{X}$ - 2 $\hat{\sigma}$; $\overline{X}$ + 2 $\hat{\sigma}$],
3. Approximativement **99.7 %** des observations sont comprises dans l'intervalle [$\overline{X}$ - 3 $\hat{\sigma}$; $\overline{X}$ + 3 $\hat{\sigma}$].

Ceci est illustré sur la Figure 3.13.

Lorsqu'il s'agit plus précisément de vouloir renseigner sur la tendance centrale relative à l'échantillon étudié, le choix dépend de la forme de la distribution observée. Lorsque la distribution est gaussienne, la moyenne, la médiane, et le mode, sont similaires et donc se valent. Toutefois, lorsque la distribution est asymétrique et unimodale (i.e., avec un seul pic), le mode reflètera mieux la tendance centrale. De plus, lorsque la distribution est asymétrique, la médiane aura tendance à mieux représenter la tendance centrale que la moyenne (Rousselet & Wilcox, 2020). Notons que dans certains cas où la distribution semble asymétrique, la médiane peut se retrouver malgré tout plus éloignée du mode que la moyenne, comme illustré dans l'exemple emprunté à Gonzales et al. (2001) qui est montré sur la Figure 3.14.

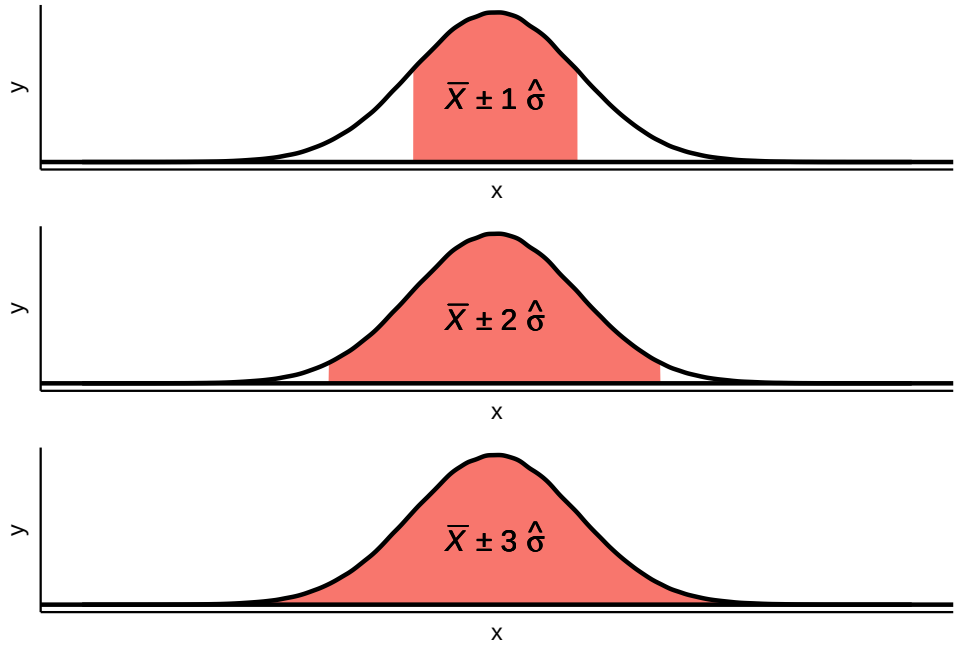

Figure 3.13: Proportions des observations incluses dans différents intervalles liés à la moyenne et à des multiples de l'écart-type

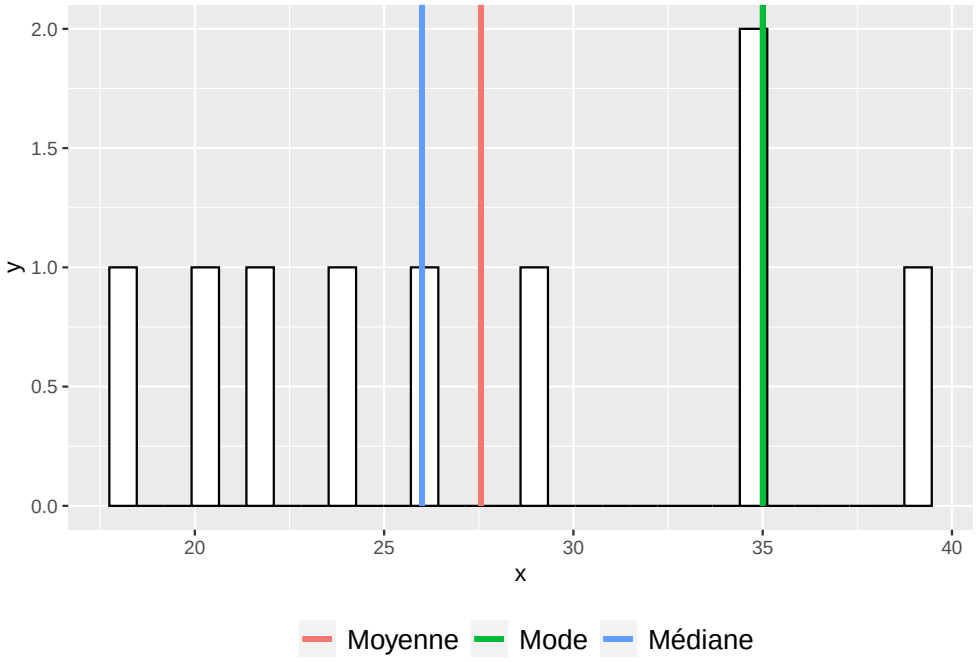

Figure 3.14: Exemples de positions de la moyenne et de la médiane dans le cadre d'une distribution asymétrique

3.2 Variables qualitatives

3.2.1 Visualiser la distribution de la variable

Comme dans le cadre de variables quantitatives, une bonne pratique est de visualiser graphiquement la distribution d'une variable qualitative avant de l'analyser. Ici, il s'agit plus précisément de prendre connaissance des effectifs correspondant aux différentes modalités de la variable. Une manière rapide de procéder pour cela est d'utiliser la fonction `ggplot()` et la fonction `geom_bar()`. Illustrons cela avec le jeu de données `diamonds`, qui contient notamment la variable qualitative `color`.

```
# Aperçu du jeu de données
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  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
##  2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
##  3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
##  4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
##  5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
##  6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
##  7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
## 10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39
## # i 53,930 more rows
```

```
# Visualisation de la distribution de la variable color
diamonds |>
  ggplot(aes(x = color)) +
    geom_bar()
```

Bien que rapide, la manière de procéder avec le code montré ci-avant devient vite limitée lorsque l'on veut enrichir le graphique, tel qu'en ajoutant les valeurs des effectifs au-dessus des barres ou encore en changeant l'ordre de disposition des barres. Pour gagner en capacité de modification du graphique, on peut d'abord passer par une étape intermédiaire consistant à créer un mini-jeu de données où la variable `color` serait déjà résumée à l'aide de la fonction `count` du package `dplyr`, de telle sorte à n'avoir que la valeur de l'effectif en regard de chaque modalité. Ceci est montré dans le code ci-dessous.

```
diamonds |>
  count(color)
```

```
## # A tibble: 7 x 2
##   color     n
##   <ord> <int>
## 1 D      6775
## 2 E      9797
## 3 F      9542
## 4 G     11292
```

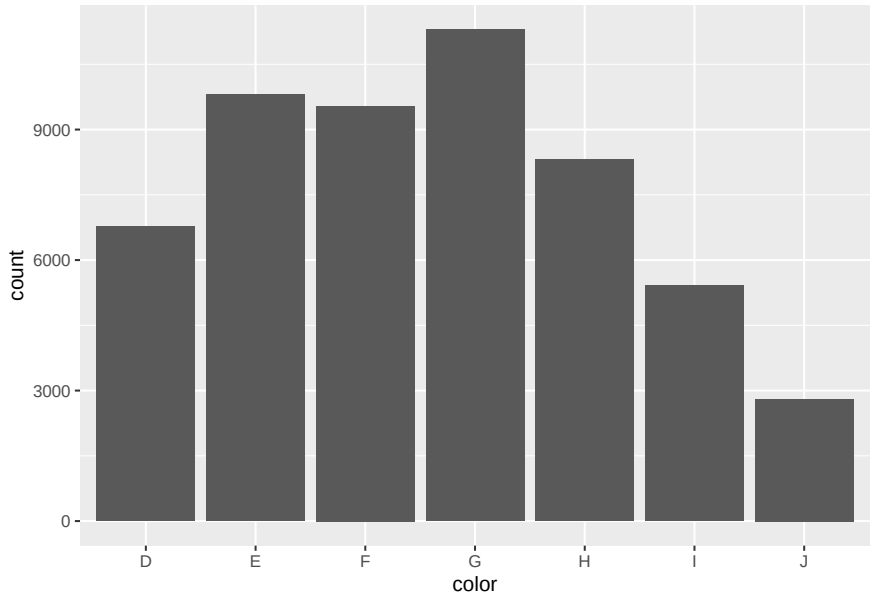

Figure 3.15: Exemple de diagramme en barres

```
## 5 H       8304
## 6 I       5422
## 7 J       2808
```

Si l'on poursuit le code avec les fonctions `ggplot()` et `geom_bar()` (cf. code ci-dessous), on arrive alors au même résultat que précédemment (cf Figure 3.15). On note qu'il a fallu adapter l'argument `stat` de la fonction `geom_bar()` pour que le graphique montre bien en ordonnées la valeur de la variable nouvellement appelée `n`, qui comprend les effectifs de chaque modalité.

```
diamonds |>
  count(color) |>
  ggplot(aes(x = color, y = n)) +
     geom_bar(stat = "identity")
```

Certes, pour le moment, cette seconde procédure n'a fait que rajouter une ligne de code par rapport à la première procédure. Toutefois, en reprenant la logique de la seconde procédure, on peut à présent créer des graphiques en barres plus élaborés relativement facilement, comme montré ci-dessous.

```
# Diagramme en barres avec les effectifs affichés à proximité des barres
A <-
  diamonds |>
  count(color) |>
  ggplot(aes(x = color, y = n)) +
     geom_bar(stat = "identity") +
     geom_text(aes(label = n), size = 2, nudge_y = 700) +
     ggtitle("A")

# Diagramme en barres réorganisées manuellement
B <-
  diamonds |>
```

```
  count(color) |>
  ggplot(aes(x = fct_relevel(color, "J", "I", "H", "G", "F", "E", "D"),
             y = n)) +
     geom_bar(stat = "identity") +
     xlab("color") +
     ggtitle("B")

# Diagramme en barres réorganisées en ordre croissant
C <-
  diamonds |>
  count(color) |>
  ggplot(aes(x = fct_reorder(color, n), y = n)) +
     geom_bar(stat = "identity") +
     xlab("color") +
     ggtitle("C")

# Diagramme en barres réorganisées en ordre décroissant
D <-
  diamonds |>
  count(color) |>
  ggplot(aes(x = fct_reorder(color, -n), y = n)) +
     geom_bar(stat = "identity") +
     xlab("color") +
     ggtitle("D")

# Diagramme en barres pivoté
E <-
  diamonds |>
  count(color) |>
  ggplot(aes(x = fct_reorder(color, -n), y = n)) +
  geom_bar(stat = "identity") +
     coord_flip() +
     ggtitle("E")

# Dotplot pivoté
F <-
  diamonds |>
  count(color) |>
  ggplot(aes(x = fct_reorder(color, -n), y = n)) +
     geom_point() +
     coord_flip() +
     ggtitle("F")
```

Dans le code montré ci-dessus, on remarque que le réagencement manuel des barres (cf. graphique B) a pu être réalisé grâce à la fonction `fct_relevel()` du package `forcats`. Pour les autres diagrammes, la réorganisation des barres en ordre croissant ou décroissant sur la base de la valeur de l'effectif (`n`) a pu se faire grâce à la fonction `fct_reorder()` du package `forcats`. Les noms des arguments de ces fonctions n'ont pas été indiqués pour alléger le code. Le plus important, c'est d'indiquer en premier dans la fonction la variable dont l'ordre d'apparition des modalités doit être réorganisé (il s'agissait de la variable `color` dans cet exemple). En second, il convient d'indiquer la logique de réorganisation de l'apparition des modalités (ce qui a été fait sur la base des valeurs de la variable `n` dans les exemples ci-dessus utilisant la fonction `fct_reorder()`).

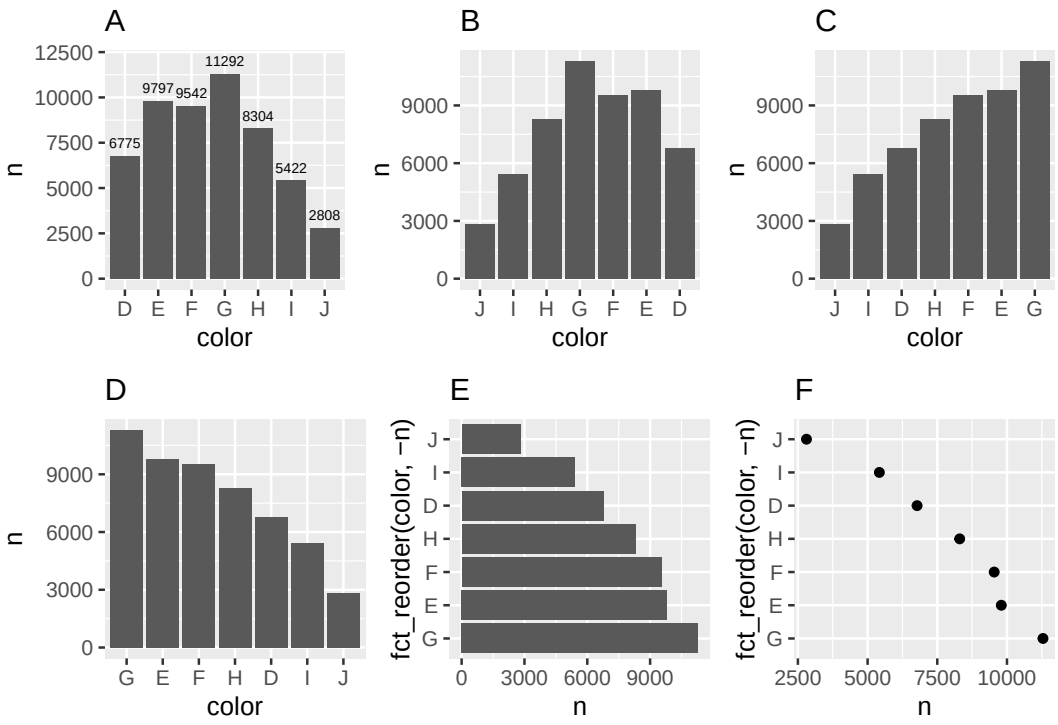

Figure 3.16: Différentes sortes de diagrammes en barres

Toujours dans le code montré ci-dessus, on peut voir aussi, dans la fonction `geom_text()` du graphique A, la présence de l'argument `nudge_y`. Cet argument permet de régler le décalage entre le haut de la barre et le texte. L'unité du chiffre indiqué est celle de la variable montrée en ordonnées.

On observe que la réorganisation des barres selon un ordre croissant ou décroissant clarifie l'information délivrée par le graphique. Toutefois, cette réorganisation est en principe surtout recommandée pour des variables qualitative nominales, c'est-à-dire des variables pour lesquelles il n'existe pas un ordre naturel des modalités. Lorsqu'il existe un ordre naturel des modalités, comme c'est le cas pour des variables qualitatives ordinales, les modalités devraient préférentiellement suivre leur ordre naturel.

En plus des effectifs, il est aussi possible de prendre connaissance de la distribution à l'aide des proportions, c'est-à-dire des ratios entre les effectifs liés aux différentes modalités et l'effectif total. Les proportions peuvent être visualisées avec un diagramme circulaire (i.e., un camembert), avec un diagramme en barres empilées, ou avec des barres disposées côte-à-côte. Les diagrammes circulaires et avec barres empilées mettent en avant le fait que les parties individuelles étudiées font partie d'un même ensemble. Les diagrammes circulaires peuvent être utilisés efficacement à cet effet lorsqu'ils montrent des fractions simples telles qu'un quart, un tiers, ou une moitié. Toutefois, les parties individuelles sont plus facilement comparables lorsqu'on utilise des barres mises côte-à-côte, comme montré ci-avant. Les diagrammes avec barres empilées sont quant à eux difficiles à comprendre lorsqu'il s'agit de n'étudier qu'une seule variable qualitative (Wilke, 2018).

```
# Diagramme en barres empilées montrant les effectifs
A <-
  diamonds |>
  count(color) |>
```

```
  ggplot(aes(x = "", y = n, fill = color)) +
  geom_bar(stat = "identity", na.rm = TRUE) +
  geom_text(aes(label = n),
            size = 2,
            position = position_stack(vjust = 0.5)) +
  ggtitle("A")

# Diagramme en barres empilées montrant les proportions
B <-
  diamonds |>
  count(color) |>
  mutate(prop = n / sum(n) * 100) |>
  ggplot(aes(x = "", y = prop, fill = color)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = paste0(round(prop, digits = 2), " %")),
            size = 2,
            position = position_stack(vjust = 0.5)) +
  ggtitle("B")

# Diagramme circulaire montrant les effectifs
C <-
  diamonds |>
  group_by(color) |>
  summarize(n = n()) |>
  ggplot(aes(x = "", y = n, fill = color)) +
  geom_bar(stat = "identity", position = "stack") +
  coord_polar(theta = "y",
              start = 0,
              direction = -1) +
  geom_text(aes(label = n),
            size = 2,
            position = position_stack(vjust = 0.5)) +
  ggtitle("C")

# Diagramme circulaire montrant les proportions
D <-
  diamonds |>
  count(color) |>
  mutate(prop = n / sum(n) * 100) |>
  ggplot(aes(x = "", y = prop, fill = color)) +
  geom_bar(stat = "identity", position = "stack") +
  coord_polar(theta = "y",
              start = 0,
              direction = -1) +
  geom_text(aes(label = paste0(round(prop, digits = 2), " %")),
            size = 2,
            position = position_stack(vjust = 0.5)) +
  ggtitle("D")
```

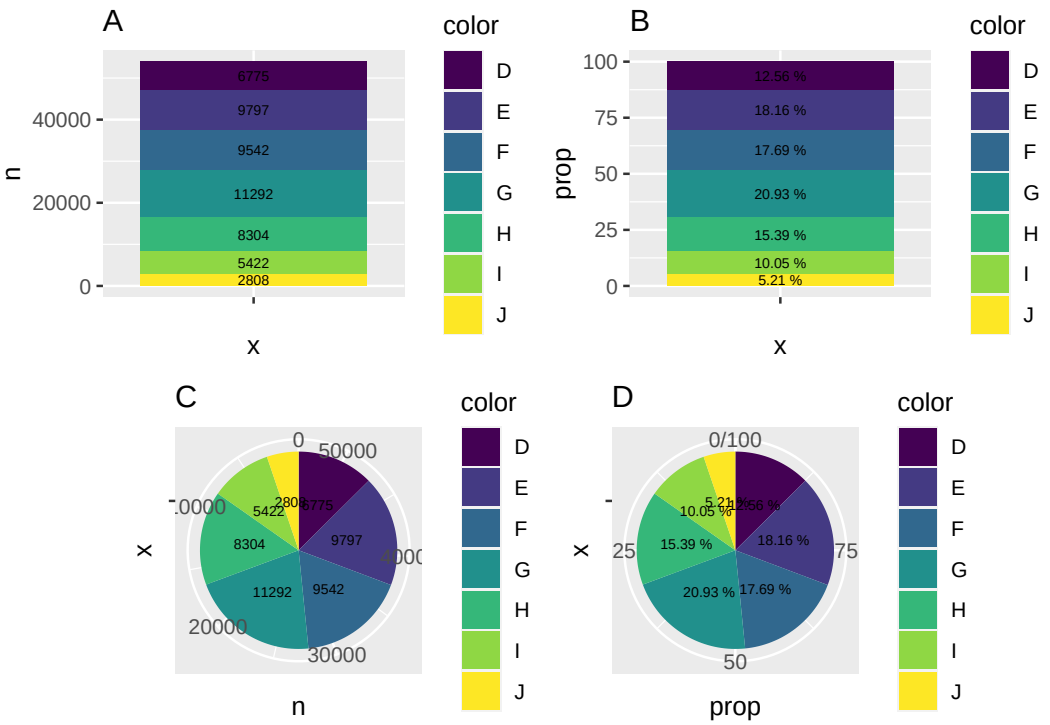

Figure 3.17: Différentes sortes de diagrammes pour représenter des proportions

3.2.2 Déterminer la tendance centrale

Dans le cadre de variables qualitatives, la tendance centrale peut être renseignée à l'aide du mode en présence d'une variable nominale, et à l'aide de la médiane ou du mode en présence d'une variable ordinale (Gonzales & Ottenbacher, 2001). Avec des variables qualitatives, il est possible de prendre connaissance du mode facilement à l'aide d'un tableau de résultats qui récapitule, par ordre décroissant, les effectifs et les proportions associées aux différentes modalités de la variable étudiée. Ce type de tableau peut être obtenu à l'aide de la fonction `freq()` du package `questionr`, qu'il convient d'installer et charger au préalable afin de pouvoir l'utiliser. Dans l'exemple ci-dessous, le mode de la variable `color` dans le jeu de données `diamonds` est la modalité présente sur la première ligne du tableau.

```
library(questionr)
freq(diamonds$color, valid = TRUE, total = TRUE, sort = "dec")
```

```
##             n     %  val%
## G       11292  20.9  20.9
## E        9797  18.2  18.2
## F        9542  17.7  17.7
## H        8304  15.4  15.4
## D        6775  12.6  12.6
## I        5422  10.1  10.1
## J        2808   5.2   5.2
## Total 53940 100.0 100.0
```

3.3 Résumé

- En statistiques, la notion de population désigne l'ensemble des individus existant qui présentent un ou plusieurs critères d'intérêt. Un échantillon est alors une fraction de la population étudiée, composée d'individus qui en principe sont représentatifs de la population étudiée.
- Avec R, les graphiques peuvent être réalisés, entre autres, à l'aide du package `ggplot2` et des fonctions associées.
- Lors de l'analyse d'une variable quantitative, une première étape doit être de visualiser graphiquement la distribution des observations. Cela peut se faire à l'aide :
  - d'un histogramme avec la fonction `ggplot2::geom_histogram()` ;
  - d'une boîte à moustaches avec la fonction `ggplot2::geom_boxplot()` ;
  - ou encore d'un *raincloud plot* avec la fonction `ggrain::geom_rain()`.
- La distribution d'une variable quantitative peut être notamment de forme gaussienne, asymétrique, leptocurtique, ou encore platycurtique.
- Les indices de position disponibles pour résumer une variable quantitative sont la moyenne (`mean()`), la médiane (`median()`), le mode (`lsr::modeOf()`), et la moyenne rognée (`mean(trim = ...)`).
- Les indices de dispersion disponibles pour résumer une variable quantitative sont l'étendue (`min()`, `max()`, `range()`), l'écart-type (`sd()`), et les quartiles (`quantile(probs = c(0.25, 0.75))`).
- L'indice statistique permettant de décrire le niveau d'asymétrie d'une variable quantitative est le coefficient d'asymétrie (`e1071::skewness()`).
- L'indice statistique permettant de décrire le niveau d'aplatissement d'une variable quantitative est le coefficient d'aplatissement (`e1071::kurtosis()`).
- Les fonctions `psych::describe()`, `psych::describeBy()` ou encore `skimr::skim()` permettent de récapituler les indices statistiques généralement étudiés dans le cadre de variables quantitatives.
- Lorsque la distribution d'une variable quantitative est gaussienne, approximativement 68.3 %, 95.5 %, et 99.7 % des observations sont situées dans $\pm$ 1 s, $\pm$ 2 s, et $\pm$ 3 s autour de la moyenne, respectivement (s étant l'écart-type de la variable).
- Lorsque la distribution d'une variable quantitative est asymétrique, la médiane peut être l'indicateur le plus adapté pour décrire la tendance centrale, en particulier en présence de petits échantillons.
- Lors de l'analyse d'une variable qualitative, une première étape doit être de visualiser graphiquement la distribution des effectifs. Cela peut se faire à l'aide d'un diagramme en barres (`ggplot2::geom_bar()`).
- Les variables qualitatives nominales devraient être visualisées avec une organisation des modalités selon un ordre croissant ou décroissant.
- Les variables qualitatives ordinales devraient être visualisées avec une organisation des modalités selon leur ordre naturel.
- Dans le cadre de variables qualitatives nominales, la tendance centrale peut être étudiée à l'aide du mode.

- Dans le cadre de variables qualitatives ordinales, la tendance centrale peut être étudiée à l'aide de la médiane ou du mode.
- La fonction `questionr::freq()` permet de récapituler les effectifs et les proportions relatifs aux modalités d'une variable qualitative.

Chapitre 4

Analyses bivariées

Réaliser une analyse bivariée désigne le fait d'étudier la relation qui peut exister entre deux variables. Dans ce chapitre, nous allons voir les procédures graphiques et calculatoires qui permettent d'étudier et de quantifier le degré de relation pouvant exister entre deux variables dans les cas suivants : entre deux variables quantitatives, entre deux variables qualitatives, et entre une variable quantitative et une variable qualitative. Comme dans le chapitre précédent, l'objectif est ici d'explorer et de décrire les données et leurs relations à l'échelle d'un échantillon, sans pour autant chercher à déterminer l'incertitude qu'il peut exister dans les statistiques calculées en vue de les utiliser pour réaliser une inférence dans la population représentée.

4.1 Relation entre deux variables quantitatives

4.1.1 Étudier graphiquement la relation

Comme dans le cadre d'analyses univariées, une bonne pratique, lorsqu'on étudie une relation bivariée, est de faire un graphique. Avec des variables quantitatives, il s'agit de montrer les valeurs d'une variable en fonction des valeurs de l'autre variable, chose que permet un simple nuage de points. Plusieurs types de relations peuvent alors être rencontrés, ces relations pouvant potentiellement s'apparenter à autant de fonctions mathématiques que l'on connaît. Parmi les plus connues, on a par exemple les relations linéaires, les relations logarithmiques, ou encore les relations quadratiques, qui sont illustrées sur la Figure 4.1.

Avec R, pour obtenir un nuage de points à partir d'un jeu de données, il est possible d'utiliser la fonction `ggplot()` en l'associant à la fonction `geom_point()` du package `ggplot2`, comme dans l'exemple ci-dessous qui utilise le jeu de données `mtcars` (qui est intégré à R de base) et les variables `hp` (*gross horsepower*) et `mpg` (*miles/US gallon*). Dans cet exemple dont le résultat est montré sur la Figure 4.2, on peut voir que la relation semble globalement linéaire négative (voire curvilinéaire négative si l'on donne de l'importance au point isolé à droite du graphique).

```
ggplot(data = mtcars, aes(x = hp, y = mpg)) +
  geom_point()
```

4.1.2 Étudier numériquement la relation

Le coefficient de corrélation de Pearson

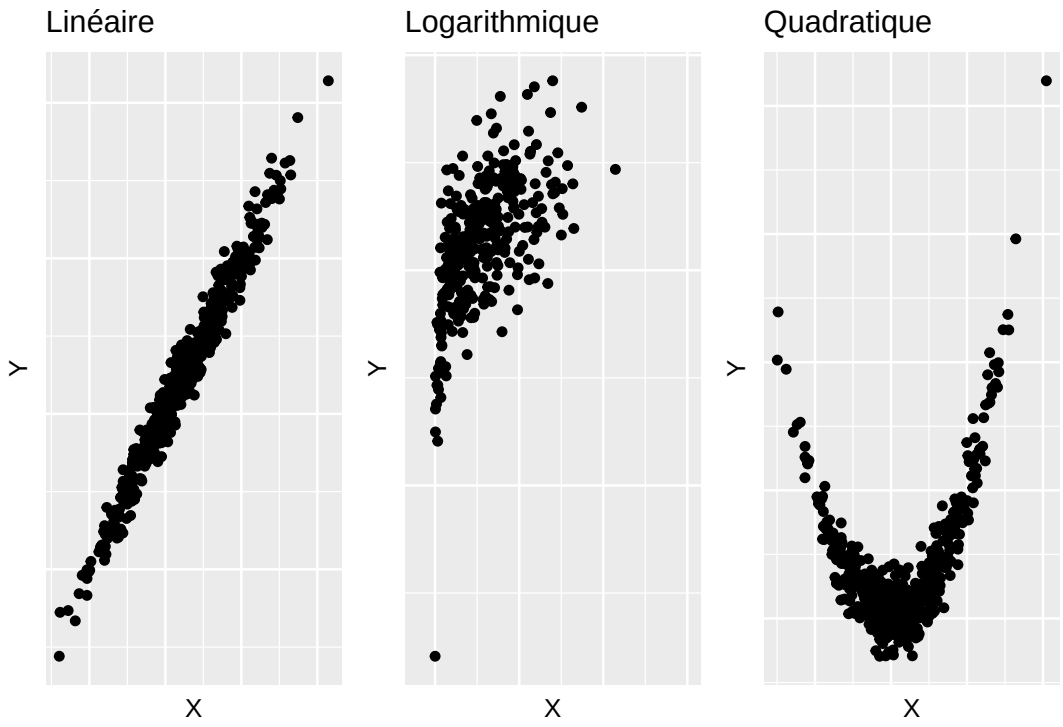

Figure 4.1: Différentes formes de relation entre deux variables quantitatives

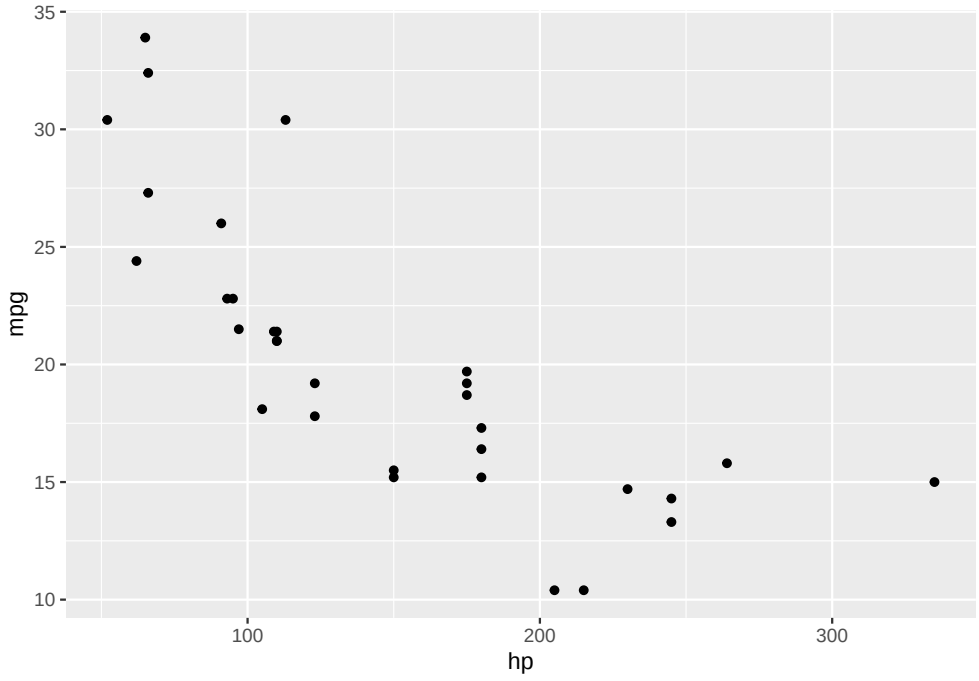

Figure 4.2: Nuage de points montrant les variables hp et mpg du jeu de données mtcars

Lorsque la relation étudiée semble linéaire, l'étude numérique classique consiste à calculer le coefficient de corrélation de Pearson, noté r, dont la valeur vise à renseigner dans quelle mesure le nuage de points représentant le lien entre les deux variables étudiées suit une droite. Avant de se lancer dans le calcul du coefficient de corrélation de Pearson pour étudier la relation entre une variable X et une variable Y, il peut donc être utile de compléter le nuage de points montré sur la Figure 4.2 avec une droite d'équation de type $Y = aX + b$. Cette équation serait la meilleure modélisation possible de la relation linéaire entre X et Y, de telle sorte que parmi l'infinité d'équations qui pourraient lier X à Y, c'est cette équation qui au total donnerait la plus petite erreur lorsque l'on voudrait prédire Y à partir de X. Si X et Y sont liées de manière linéaire, alors le nuage des points relatifs aux deux variables devrait s'étaler le long de cette droite. Pour obtenir cette droite en plus du nuage de points, il est possible d'utiliser la fonction `geom_smooth()` du package `ggplot2`.

```
ggplot(data = mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  geom_smooth(formula = y ~ x, method = "lm", se = FALSE)
```

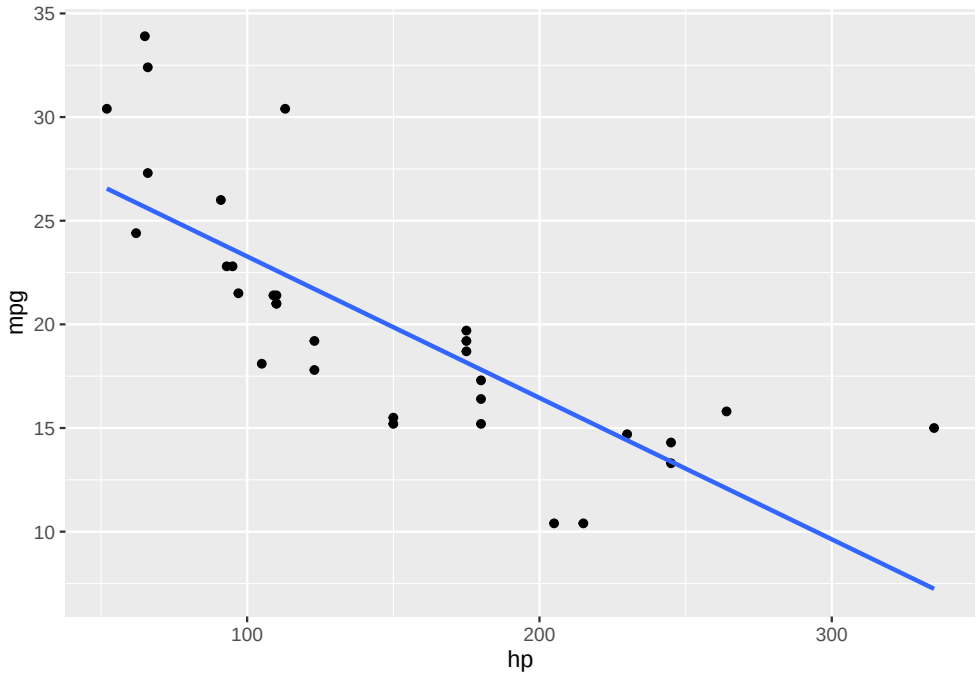

Figure 4.3: Nuage de points avec droite de régression pour les variables hp et mpg du jeu de données mtcars

Dans la fonction `geom_smooth()` qui a été utilisée dans l'exemple ci-dessus, on note que l'argument `formula` pourrait être considéré comme facultatif car il s'agit ici de la configuration par défaut de la fonction. En revanche, l'argument `method` doit être ici configuré avec `"lm"` (pour *linear model*) car ce n'est pas la méthode graphique configurée par défaut dans la fonction. Enfin, l'argument `se` permet de montrer ou non un intervalle de confiance autour de la droite de régression, ce qui n'a pas été activé ici (par défaut, l'argument `se` est configuré pour montrer cet intervalle de confiance). Dans l'exemple montré ci-dessus, la représentation graphique encourage fortement à penser que l'un des types de relations à envisager prioritairement dans l'étude des deux variables est la relation linéaire. Cette information rend pertinente l'utilisation du coefficient de corrélation de Pearson pour une étude numérique de la relation en question.

La valeur du coefficient de corrélation de Pearson peut aller de 1 (suggérant une relation linéaire

positive parfaite) à -1 (suggérant une relation linéaire négative parfaite). Des valeurs proches de 0 suggèreraient une abscence de relation linéaire. La formule du coefficient de corrélation de Pearson (r) pour un échantillon est la suivante :

$$r_{X,Y} = \frac{COV_{X,Y}}{\hat{\sigma}_X \hat{\sigma}_Y} = \frac{\sum_{i=1}^{N}(Xi - \overline{X})(Yi - \overline{Y})}{N-1} \frac{1}{\hat{\sigma}_X \hat{\sigma}_Y},$$

COV désignant la covariance entre les variables X et Y, Xi et Yi les valeurs de X et Y pour une observation i, $\overline{X}$ et $\overline{Y}$ les moyennes respectives des variables X et Y, N le nombre d'observations, et $\hat{\sigma}_X$ et $\hat{\sigma}_Y$ les écarts-types respectifs des variables X et Y. Cette formule indique que le coefficient de corrélation de Pearson s'obtient en divisant la covariance des deux variables étudiées par le produit de leurs écarts-types respectifs.

Le Tableau 4.1 montre les premières étapes du calcul de la covariance pour des couples de variables fictifs $(X1, Y1)$, $(X1, Y2)$, et $(X1, Y3)$. En particulier, la partie droite du tableau (de X1Y1 à X1Y3) montre le calcul du produit $(Xi - \overline{X})(Yi - \overline{Y})$ pour les différents couples de variables et cela pour chaque ligne du jeu de données.

Tableau 4.1: Étape intermédiaire pour le calcul de la covariance entre des variables X1 et Y1, Y2, et Y3

X1	Y1	Y2	Y3	X1Y1	X1Y2	X1Y3
0	0	0	0	36	40.285714	-36
2	2	1	-2	16	22.857143	-16
4	4	15	-4	4	-16.571429	-4
6	6	5	-6	0	0.000000	0
8	8	11	-8	4	8.571429	-4
10	10	3	-10	16	-14.857143	-16
12	12	12	-12	36	31.714286	-36

Ce que ce tableau montre, c'est que plus les deux variables étudiées évolueront de manière consistante dans des sens identiques comme avec $X1$ et $Y1$, ou de manière consistante dans des sens opposés comme avec $X1$ et $Y3$, plus les produits $(Xi - \overline{X})(Yi - \overline{Y})$ donneront respectivement des grands scores positifs ou des grands scores négatifs, et moins les scores $(Xi - \overline{X})(Yi - \overline{Y})$ à additionner pour le calcul de la covariance s'annuleront. En effet, avec une relation relativement linéaire et positive les scores seront plus systématiquement positifs, et avec une relation relativement linéaire et négative les scores seront plus systématiquement négatifs. Toutefois, lorsqu'on aura des variables qui n'évolueront pas de manière consistante dans le même sens ou dans un sens opposé comme avec $X1$ et $Y2$, les scores positifs et négatifs liés aux calculs $(Xi - \overline{X})(Yi - \overline{Y})$ auront tendance à s'annuler et donneront lieu à une somme des scores $(Xi - \overline{X})(Yi - \overline{Y})$ diminuée, et donc à une covariance et à un coefficient de corrélation de Pearson tirés vers 0. Ces différents cas de figure et les calculs $(Xi - \overline{X})(Yi - \overline{Y})$ correspondants sont illustrés sur la Figure 4.4. Sur cette figure, chaque carré correspond au calcul $(Xi - \overline{X})(Yi - \overline{Y})$, le carré étant bleu lorsque le résultat du calcul est positif, et rouge lorsque le résultat est négatif. L'aire d'un carré illustre la grandeur du score issu du calcul. Sur les graphiques de gauche et de droite de la figure, on distingue une relation linéaire parfaite, ce qui maximise les scores à additionner pour le calcul de la covariance, dans le positif pour le graphique de gauche et dans le négatif pour le graphique de droite. Sur le graphique du milieu, on remarque que le manque de relation linéaire donne lieu à des carrés à la fois bleus et rouges,

indiquant que les scores associés aux calculs $(Xi - \overline{X})(Yi - \overline{Y})$ de la covariance s'annulent et diminuent ainsi la valeur finale de la covariance.

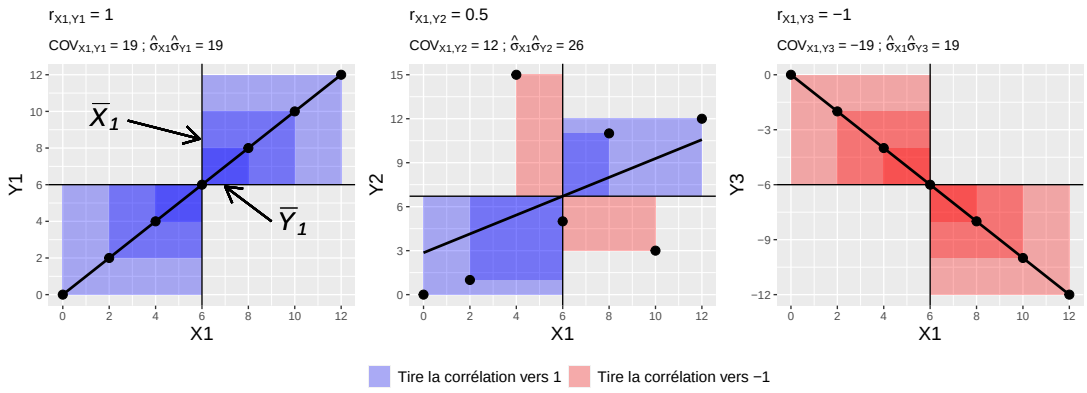

Figure 4.4: Illustration du calcul de la covariance

Dans R, le coefficient de corrélation de Pearson peut être obtenu avec la fonction `cor()`. Dans l'exemple ci-dessous qui reprend les variables du jeu de données `mtcars` utilisées plus haut, on observe un coefficient négatif, relativement proche de -1, suggérant une relation relativement linéaire et négative entre les variables étudiées.

```
cor(x = mtcars$hp, y = mtcars$mpg, method = "pearson")
```

```
## [1] -0.7761684
```

Toutefois, la fonction `cor.test()` sera plus intéressante pour la suite car elle permet de calculer des indices statistiques de probabilité qui seront nécessaires dès lors qu'il s'agira de chercher à inférer la valeur d'une corrélation dans une population d'où l'échantillon étudié provient. La valeur de la corrélation est donnée à la fin de la liste des informations qui apparaissent suite à l'activation de la fonction.

```
cor.test(x = mtcars$hp, y = mtcars$mpg, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  mtcars$hp and mtcars$mpg
## t = -6.7424, df = 30, p-value = 1.788e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.8852686 -0.5860994
## sample estimates:
##       cor
## -0.7761684
```

Sur la base de travaux antérieurs, Hopkins et al. (2009) ont fait une proposition de classification pour qualifier la valeur du coefficient de corrélation qui serait obtenue dans le cadre d'une relation linéaire. Cette proposition est montrée dans le Tableau 4.2 :

Tableau 4.2: Termes caractérisant la taille de l'effet en fonction de la valeur de corrélation obtenue

Petite	Moyenne	Grande	Très grande	Extrêmement grande
0.1	0.3	0.5	0.7	0.9

Pour visualiser le lien que l'on peut faire entre la forme du nuage de points et la valeur du coefficient de corrélation de Pearson que l'on peut obtenir, la page web proposée par Kristoffer Magnusson (https://rpsychologist.com/correlation) peut être particulièrement intéressante. Cette page web donne la possibilité de faire varier manuellement la valeur du coefficient de corrélation de Pearson pour ensuite voir un nuage de points type correspondant à cette valeur. Faites un essai !

À noter que la valeur du coefficient de corrélation de Pearson est dépendante du degré de dispersion des valeurs autour de la tendance centrale (Halperin, 1986). Ceci est illustré sur la Figure 4.5. À gauche de la figure, on observe un nuage de points représentant une relation obtenue entre deux variables quantitatives dans une population complète. La valeur du coefficient de corrélation de Pearson est ici particulièrement élevée. À droite de la figure, on observe exactement les mêmes variables dans la même population que sur le graphique de gauche, mais sur un intervalle dont l'étendue a été manuellement restreinte, diminuant ainsi la variabilité pour les deux variables. On observe alors une diminution de la valeur du coefficient de corrélation de Pearson. Cet exemple doit faire prendre conscience qu'il faut faire attention lorsqu'on cherche à comparer des coefficients de corrélation de Pearson obtenus avec des échantillons différents. En effet, si les variables correspondant respectivement à ces échantillons n'ont pas les mêmes niveaux de variabilité, les valeurs des coefficients de corrélation de Pearson ne seront pas vraiment comparables, en sachant que c'est l'échantillon qui présente la plus grande variabilité qui aura plus de chances de présenter une valeur de coefficient de corrélation de Pearson plus élevée.

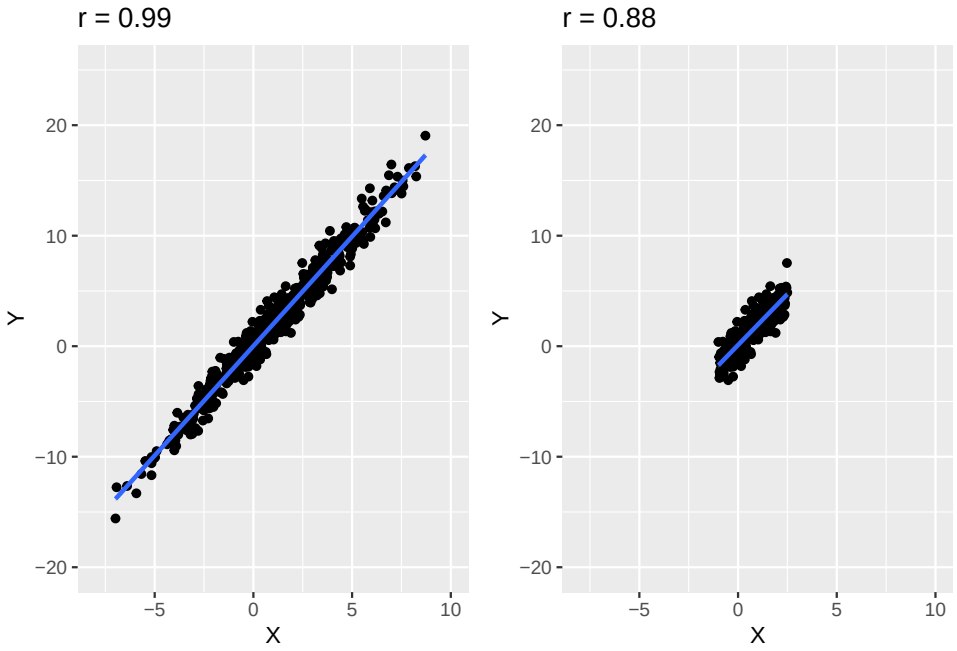

Figure 4.5: Influence de la diminution de l'étendue des variables sur la valeur du coefficient de corrélation de Pearson

Un cas particulier illustrant l'influence du degré de dispersion sur la relation étudiée est montré sur la Figure 4.6. Cet exemple a pour but de montrer que l'étude d'une relation entre deux variables quantitatives peut aboutir à des conclusions radicalement différentes selon que l'on conduit les analyses à l'échelle d'un groupe entier ou que l'on sépare les analyses par groupe de caractéristiques, en particulier lorsque la distribution des données par groupe est radicalement différente de celle obtenue à l'échelle du groupe entier. Sur la gauche de la Figure 4.6, le nuage de points représente la relation entre deux variables à l'échelle de l'ensemble du groupe étudié. La variabilité possible pour les deux variables étudiées (V1 et V2 dans l'exemple) est alors maximale. Toutefois, ces données appartiennent en réalité à des sous-groupes distincts (cf. couleurs sur le graphique de droite de la Figure 4.6). L'analyse par sous-groupe diminue la variabilité à la fois pour V1 et V2, donnant même lieu ici à des relations de sens opposé à celui observé à l'échelle de l'ensemble du groupe ! Cette situation correspond à ce qu'on appelle un paradoxe de Simpson, lequel se présente lorsque le phénomène que l'on peut observer avec une vue globale est annulé voire inversé lors d'une analyse par sous-groupe. Ici, la grande variabilité associée aux données du graphique de gauche de la Figure 4.6 crée une relation artificiellement et donc faussement positive entre V1 et V2. C'est l'analyse par sous-groupe qui permet de révéler la vraie nature de la relation étudiée.

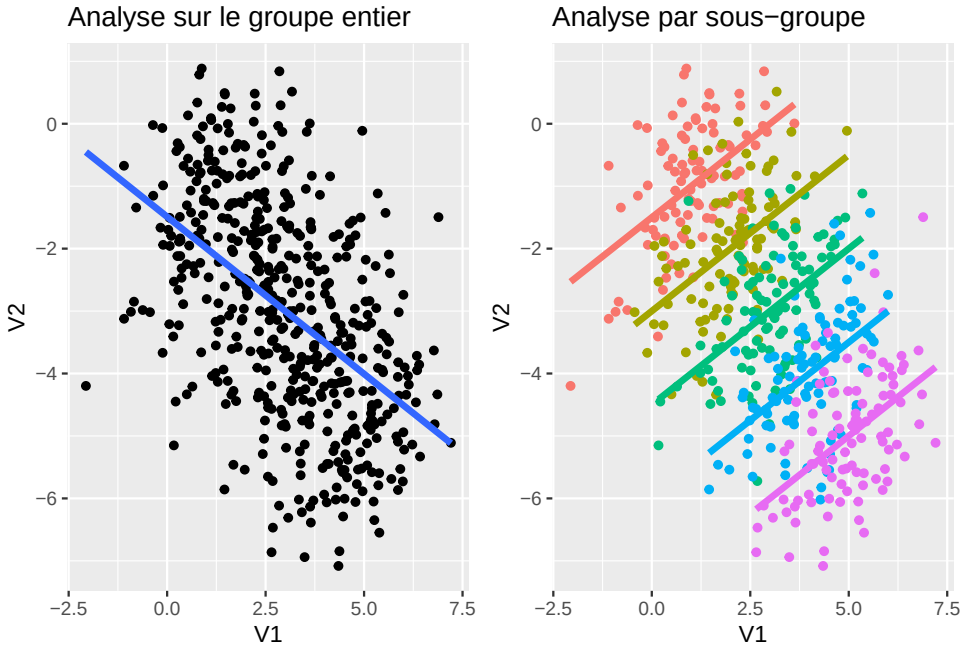

Figure 4.6: Influence du niveau d'analyse (groupe entier vs. sous-groupes) sur la corrélation observée entre deux variables quantitatives

En outre, la valeur du coefficient de corrélation de Pearson peut être influencée par des valeurs extrêmes, comme montré sur la Figure 4.7. Les deux graphiques de cette figure montrent exactement les mêmes données, à ceci près que sur le graphique de droite, on a remplacé en ordonnées une valeur du graphique de gauche pour lui donner la valeur de 10. L'influence de cette simple action sur la valeur du coefficient de corrélation de Pearson est nette. Ceci montre qu'il faut faire attention aux valeurs extrêmes qui pourraient grandement influencer la variabilité des données d'une variable et au final la valeur de corrélation obtenue, notamment en présence d'échantillons de taille relativement faible. Dans le cas où la valeur du coefficient de corrélation de Pearson serait très influencée par une valeur, il pourrait être une bonne pratique de calculer la valeur du coefficient de corrélation de Pearson avec et sans cette valeur afin de pouvoir quantifier son influence sur la relation étudiée (Halperin, 1986). Une alternative pourrait être

aussi d'étudier la relation à l'aide d'autres types de coefficients que celui de Pearson, tels que celui de Spearman, présenté plus bas. Cet exemple doit faire prendre conscience qu'il n'est pas toujours pertinent de calculer le coefficient de corrélation de Pearson. En ce sens, lorsqu'on cherche à inférer la valeur du coefficient de corrélation de Pearson dans la population étudiée, il convient de vérifier certains prérequis, lesquels sont abordés plus loin dans ce livre.

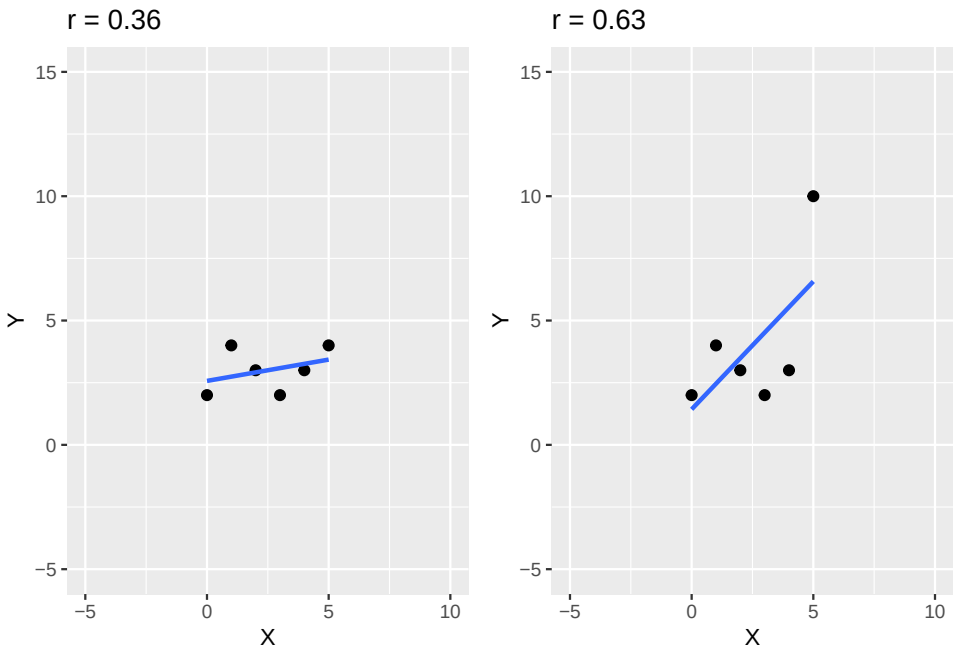

Figure 4.7: Influence d'une valeur extrême sur la valeur du coefficient de corrélation de Pearson en présence d'un petit échantillon

Lorsque la relation étudiée ne semble pas linéaire mais s'apparente assez clairement à d'autres fonctions mathématiques, telles que des relations logarithmiques ou polynomiales, il est possible de transformer une des variables, voire les deux, pour rendre la relation linéaire et à nouveau étudiable à l'aide du coefficient de corrélation de Pearson (Halperin, 1986). Toutefois, il est aussi possible de créer des modèles de régression non linéaires afin de regarder si ces modèles correspondent bien aux données. La détermination et la validation d'un modèle non linéaire qui correspondrait bien aux données confirmerait alors que la relation étudiée a une forme particulière et potentiellement prédictible. Les procédures pour explorer différents modèles de régression (linéaires et non linéaires) sont abordées au chapitre suivant. Enfin, une dernière alternative possible, pour étudier la relation entre deux variables quantitatives dont les distributions ne permettraient pas d'utiliser correctement le coefficient de corrélation de Pearson, serait l'utilisation de coefficients de corrélation basés sur les rangs, tels que le coefficient de corrélation de Spearman.

Le coefficient de corrélation de Spearman

Lorsque le coefficient de corrélation de Pearson ne permet pas de caractériser fiablement le degré de relation linéaire **entre les valeurs** de deux variables (e.g., en présence de valeurs aberrantes au sein d'un échantillon de petite taille), une alternative peut être d'étudier le degré de relation linéaire **entre les rangs** de ces deux variables. Le rang, c'est le classement (ou la position) d'une observation donnée au sein d'une variable en fonction de sa valeur. Dans une variable, les observations avec les valeurs les plus faibles seront associées aux rangs les plus bas alors que les observations avec les valeurs les plus élevées seront associées aux rangs les plus élevés. Une illustration de la notion de rang est proposée dans le Tableau 4.3 pour la variable `hp` du jeu de

données `mtcars`. Dans ce tableau, les lignes ont été ordonnées sur la base des rangs de la variable `hp`. On pourra remarquer que dans le tableau, nous avons ce qu'on appelle des ex-aequos, c'est-à-dire que plusieurs observations peuvent présenter les mêmes valeurs, et donc avoir le même rang.

Tableau 4.3: Rangs de la variable hp du jeu de données mtcars

hp (valeur)	hp (rang)
52	1.0
62	2.0
65	3.0
66	4.5
66	4.5
91	6.0
93	7.0
95	8.0
97	9.0
105	10.0
109	11.0
110	13.0
110	13.0
110	13.0
113	15.0
123	16.5
123	16.5
150	18.5
150	18.5
175	21.0
175	21.0
175	21.0
180	24.0
180	24.0
180	24.0
205	26.0
215	27.0
230	28.0
245	29.5
245	29.5

Tableau 4.3: Rangs de la variable hp du jeu de données mtcars

hp (valeur)	hp (rang)
264	31.0
335	32.0

Le fait d'étudier l'existence d'une relation linéaire entre les rangs et non plus entre les valeurs de deux variables permet de s'affranchir de l'influence possible de valeurs très extrêmes, dans l'une et/ou l'autre variable, sur le calcul final de la corrélation. Pour déterminer alors la valeur de la corrélation, une manière de procéder est d'appliquer la méthode de calcul du coefficient de corrélation de Pearson en utilisant non plus les valeurs des variables, mais les rangs correspondants. Cette methode, c'est celle du calcul du coefficient de corrélation de Spearman (*rho*). Si l'on suit *stricto sensu* cette définition, nous pourrions alors utiliser le code suivant pour avoir le coefficient de corrélation de Spearman :

```
cor(x = rank(mtcars$hp), y = rank(mtcars$mpg), method = "pearson")
```

```
## [1] -0.8946646
```

Toutefois, il existe une manière plus directe d'écrire les choses avec la fonction `cor`, qui contient un argument spécifiquement dédié au calcul du coefficient *rho* de Spearman :

```
cor(x = mtcars$hp, y = mtcars$mpg, method = "spearman")
```

```
## [1] -0.8946646
```

La fonction `cor.test` permet aussi de calculer le coefficient de corrélation de Spearman en fournissant aussi des informations potentiellement intéressantes pour donner une idée de la significativité statistique de l'estimation de *rho* dans la population étudiée.

```
cor.test(x = mtcars$hp, y = mtcars$mpg, method = "spearman")
```

```
##
##  Spearman's rank correlation rho
##
## data:  mtcars$hp and mtcars$mpg
## S = 10337, p-value = 5.086e-12
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## -0.8946646
```

Si l'on veut produire une représentation graphique qui illustre la valeur de *rho*, il pourrait être davantage pertinent de non plus montrer un nuage de points à partir des valeurs des variables mises en lien, mais un nuage de points à partir de leurs rangs respectifs (cf. code ci-dessous et Figure 4.8).

```
mtcars %>%
  mutate(hp_rank = rank(hp), mpg_rank = rank(mpg)) %>%
  ggplot(aes(x = hp_rank, y = mpg_rank)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```

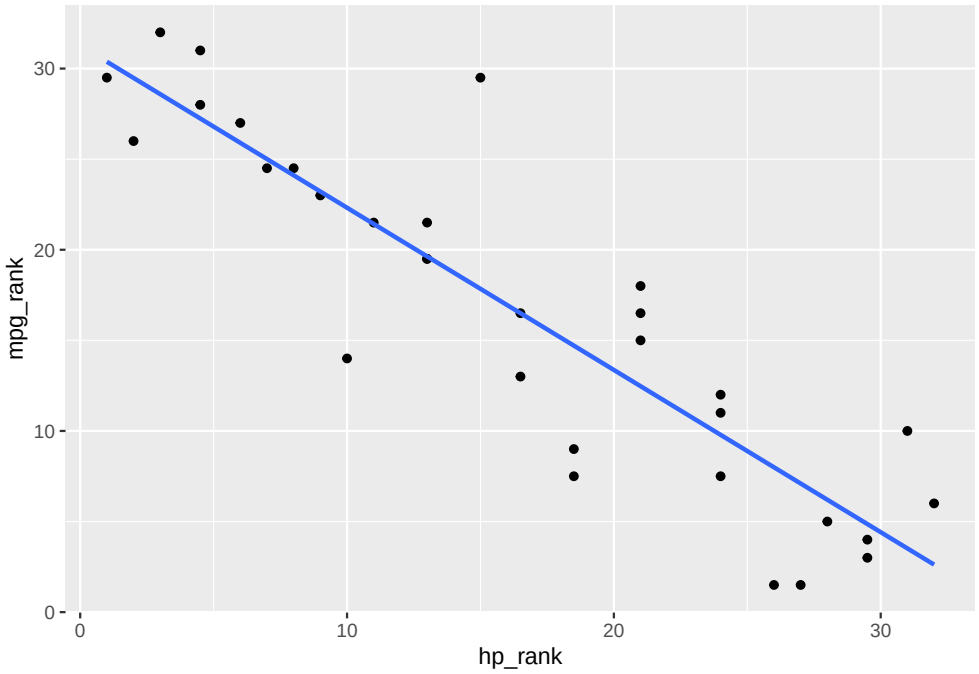

Figure 4.8: Graphique pour le coefficient de corrélation de Spearman

En matière d'interprétation, des valeurs de *rho* positives indiqueront que les deux variables mises en lien tendent à augmenter simultanément, on parlera alors de **relation monotone positive**. Dans le cas inverse, des valeurs négatives indiqueront que les deux variables mises en lien tendent à diminuer simultanément, on parlera alors de relation **monotone négative**. À noter cependant que de par son calcul, la valeur de *rho* ne permet pas de renseigner sur la forme de relation qu'il pourrait y avoir entre les valeurs des deux variables (e.g., linéaire ou curvilinéaire par exemple). Ceci est illustré sur la Figure 4.9. Sur cette figure, le graphique de gauche montre la relation entre les valeurs des variables X et Y, qui est caractérisée par un coefficient de corrélation de Spearman (*rho*) de 1, indiquant donc que la relation est parfaitement monotone positive, sans préjuger de la forme particulière que pourrait présenter la relation. Pour mieux comprendre pourquoi cette valeur de *rho* est de 1, le graphique de droite de la figure montre la relation entre les rangs de ces deux variables X et Y. On voit que la relation entre les rangs est effectivement parfaitement linéaire.

4.2 Relation entre deux variables qualitatives

4.2.1 Étudier graphiquement la relation

Plusieurs types de graphiques peuvent être envisagés lorsqu'il s'agit de visualiser des données relatives au croisement de deux variables qualitatives. Une première approche consiste à utiliser des graphiques avec barres mises côte-à-côte, comme illustré sur la Figure 4.10, qui a été réalisée

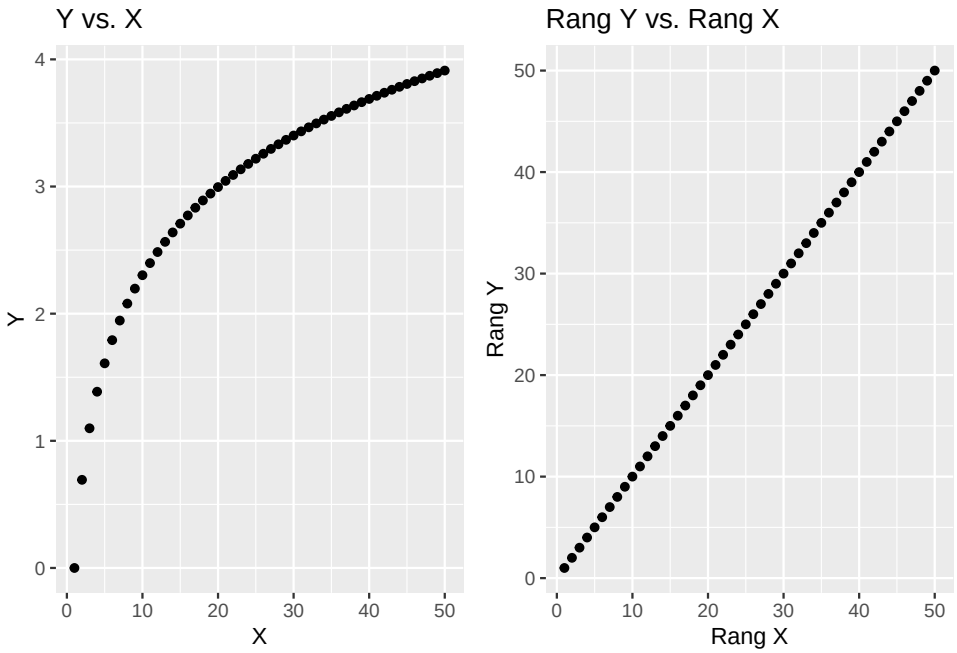

Figure 4.9: Distinction entre la relation observée entre les valeurs (graphique de gauche) et les rangs (graphique de droite) de deux variables

à partir du jeu de données `JointSports`, lequel est utilisable après installation et chargement du package `vcd`. `JointSports` contient des données résumées d'effectifs mis en lien avec les modalités de différentes variables qualitatives, comme on aurait pu l'obtenir avec la fonction `count()` dans les derniers exemples du chapitre précédent. (La différence qu'il y a avec ces précédents exemples est qu'ici, l'effectif est désigné par la variable `Freq`, alors qu'il s'agissait de la variable `n` auparavant.) Pour information, `JointSports` contient les données d'une enquête s'étant intéressée, en 1983 et 1985, aux opinions d'étudiants danois de 16 à 19 ans sur la pratique sportive mixte.

```
library(vcd)

# Reconfiguration de l'ordre des modalités de la variable opinion, et calcul
# des effectifs totaux pour les catégories étudiées

JointSports_new <-
  JointSports %>%
  mutate(opinion = fct_relevel(opinion,
                               "very bad",
                               "bad",
                               "indifferent",
                               "good",
                               "very good"),
         gender = fct_relevel(gender, "Girl", "Boy")) %>%
  group_by(gender, opinion) %>%
  summarize(Freq = sum(Freq))

# Création des graphiques
A <-
```

```
  ggplot(data = JointSports_new, aes(x = gender, y = Freq, fill = opinion)) +
  geom_bar(stat = "identity",  position = "dodge") +
  scale_fill_brewer(palette = "Greens") +
  theme(legend.position = "right") +
  ggtitle("A : Mise en avant de la comparaison des opinions")

B <-
  ggplot(data = JointSports_new, aes(x = opinion, y = Freq, fill = gender)) +
  geom_bar(stat = "identity",  position = "dodge") +
  theme(legend.position = "right") +
  ggtitle("B : Mise en avant de la comparaison des genres")
```

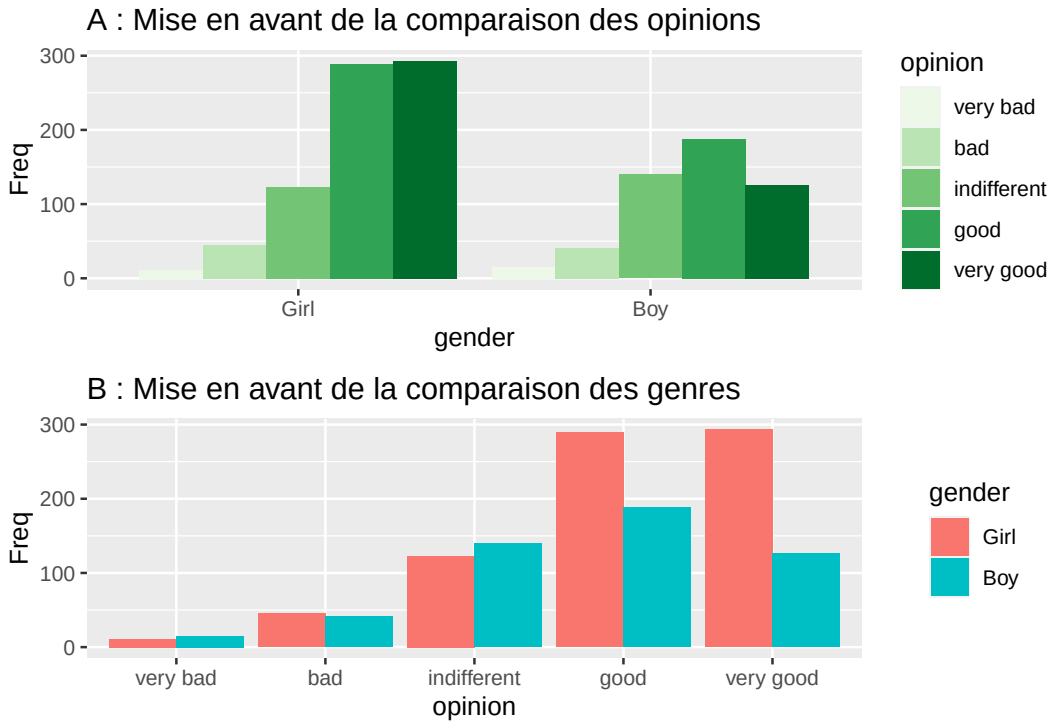

Figure 4.10: Exemples de diagramme en barres mises côte-à-côte

Pour pouvoir réaliser le graphique `A` de la Figure 4.10, il a fallu indiquer dans la fonction `ggplot()`, grâce à l'argument `x =`, la variable dont on voulait voir les modalités en abscisses, et il a fallu renseigner pour les ordonnées, à l'aide de l'argument `y =`, la variable contenant les effectifs correspondants, le tout toujours à l'intérieur de la fonction `aes()`. Étant donné que les données à montrer le long de l'axe des ordonnées sont explicitement indiquées avec `Freq`, il convient d'indiquer à l'intérieur de la fonction `geom_bar()` l'argument `stat = "identity"`, ce qui contraint à déterminer la hauteur des barres en fonction des valeurs de la variable `Freq`. À l'intérieur de la fonction `ggplot()`, plus exactement au niveau de la fonction `aes()`, c'est l'argument `fill = opinion` qui a permis d'indiquer qu'on voulait des couleurs de remplissage différentes selon les modalités de la variable `opinion`. Enfin, c'est grâce à l'argument `position = "dodge"`, à l'intérieur de la fonction `geom_bar()`, que l'on a pu obtenir des barres mises côte-à-côte, et non pas de manière empilée. Une logique similaire a été utilisée pour le graphique `B` en modifiant le code de telle sorte à ce que la distinction de l'information avec des couleurs différentes se fasse avec la variable `gender`, et non plus `opinion`.

Les graphiques `A` et `B` de la Figure 4.10 montrent l'importance de la configuration du graphique en fonction des comparaisons que l'on veut principalement faire, et donc du message que l'on veut prioritairement délivrer. Un principe qui peut guider la conception du graphique est le fait qu'il est plus facile de comparer des barres qui sont mises juste côte-à-côte. Sur la base de ce principe, le graphique `A` de la Figure 4.10 permet de comparer plus facilement les diverses opinions relevées pour les garçons d'un côté et pour les filles de l'autre, alors que le graphique `B` permet de comparer plus facilement les réponses provenant des deux genres et cela pour chaque type d'opinion. Comme indiqué par Wilke (2018), les types de graphiques illustrés avec les graphiques `A` et `B` de la Figure 4.10 peuvent parfois se voir attribuer le reproche que s'il est relativement facile de lire les informations encodées par des positions (cf. ligne de base sur les graphiques), il peut être être difficile de lire les informations encodées par une couleur dont la signification est indiquée en légende, car cela demande un effort mental supplémentaire de garder en tête la signification de la légende lorsqu'on lit le graphique. Pour palier ce problème, qui, selon Wilke (2018), est au final une affaire de goût, on pourrait utiliser la fonction `facet_wrap()` pour créer une figure telle que la Figure 4.11. Cette figure reprend la logique du graphique `A` de la Figure 4.10, avec un besoin de légende pour la variable `opinion` qui n'existe plus car la fonction `facet_wrap()` a permis de montrer les diagrammes en barres pour les deux genres de manière séparée, dans des encarts différents, et avec chacun leur propre axe des abscisses relatif aux modalités de la variable `opinion`.

```
ggplot(data = JointSports_new, aes(x = opinion, y = Freq)) +
  geom_bar(stat = "identity") +
  facet_wrap(. ~ gender)
```

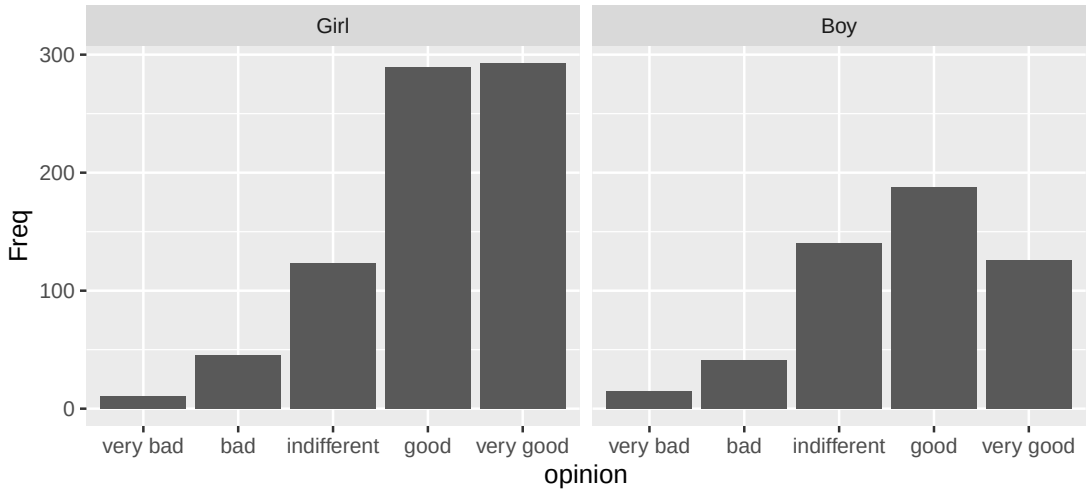

Figure 4.11: Diagrammes en barres côte-à-côte séparés selon une variable qualitative

Dans certains cas, on peut vouloir comparer les effectifs relatifs aux modalités d'une première variable qualitative avec des barres mises côte-à-côte, et n'utiliser la seconde variable qualitative que pour avoir un peu d'éléments de contexte "à l'intérieur" des effectifs affichés pour la première variable qualitative. La Figure 4.12 illustre ce cas de figure où la hauteur des barres sert prioritairement à comparer les effectifs relatifs à diverses opinions, et la coloration des barres sert à fournir une idée de la répartition (hommes / femmes dans l'exemple) dans les réponses, sans pourtant avoir l'ambition de comparer cette répartition facilement d'un type d'opinion à un autre.

```
JointSports_new %>%
  ggplot(aes(x = opinion, y = Freq, fill = gender)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = Freq), size = 3, position = position_stack(vjust = 0.5))
```

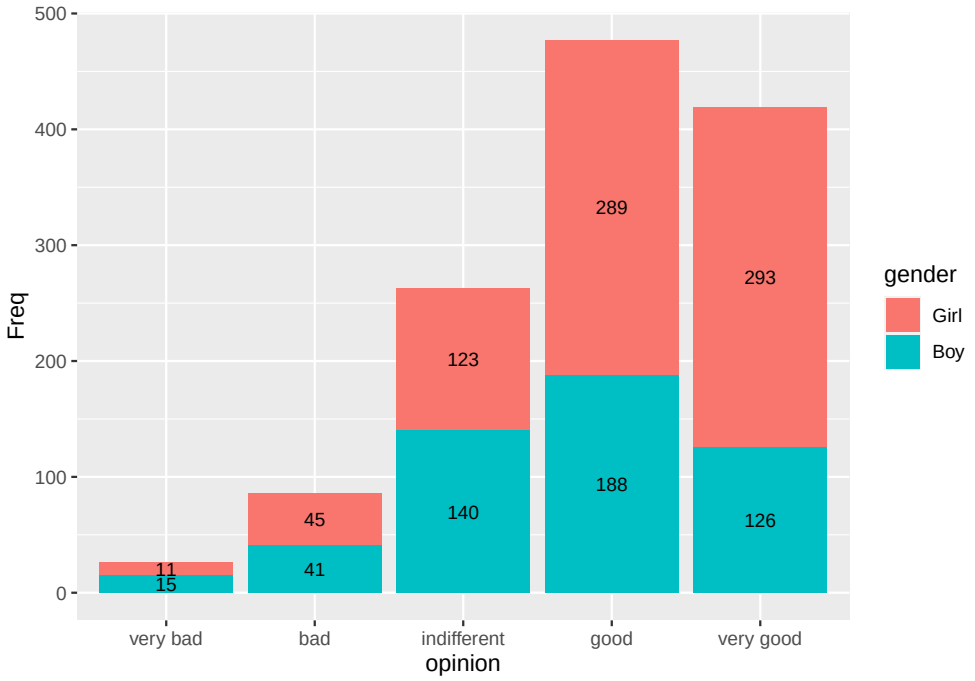

Figure 4.12: Exemple de diagramme en barres empilées

Les graphiques présentés dans cette sous-partie montrent des valeurs d'effectifs, mais selon l'objectif, il pourrait être aussi envisagé d'utiliser ces graphiques pour montrer des proportions. Cela dit, il existe d'autres visualisations possibles des proportions pour visualiser le lien entre deux variables qualitatives. Ces visualisations peuvent être consultées dans l'ouvrage en ligne de Wilke (2018).

4.2.2 Étudier numériquement la relation

Effectifs et proportions

Lorsqu'il s'agit de mener une étude numérique de la relation entre deux variables qualitatives, une première démarche à mettre en oeuvre est de récapituler numériquement les effectifs qui correspondent au croisement des deux variables. Pour cela, la fonction `table()` intégrée à R de base s'avère très pratique. Cependant, cette fonction requiert d'utiliser le jeu de données initial complet (i.e., avec toutes les observations), ce qui n'est pas le cas du jeu de données `JointSports` que nous avons utilisé précédemment, car ce dernier contient des effectifs déjà récapitulés par modalité de variable. Pour pouvoir illustrer le fonctionnement de la fonction `table()` avec les informations du jeu de données `JointSports`, j'ai donc crée un jeu de données complet qui, une fois résumé comme c'est le cas plus haut avec `JointSports`, donnerait les mêmes résultats. Ce nouveau jeu de données se nomme `JointSports_full`.

```
# Création du jeu de données JointSports_full
```

```
id <- rep(1 : sum(JointSports$Freq))
year <- c(rep("1983", 656), rep("1985", 615))
grade <- c(rep("1st", 350), rep("3rd", 306), rep("1st", 354), rep("3rd", 261))
gender <- c(rep("Boy", 134), rep("Girl", 216), rep("Boy", 115),
            rep("Girl", 191),  rep("Boy", 157), rep("Girl", 197),
            rep("Boy", 104), rep("Girl", 157))
opinion <- c(
  rep("very good", 31), rep("good", 51), rep("indifferent", 38),
  rep("bad", 10), rep("very bad", 4),
  rep("very good", 103), rep("good", 67), rep("indifferent", 29),
  rep("bad", 15), rep("very bad", 2),
  rep("very good", 23), rep("good", 39), rep("indifferent", 36),
  rep("bad", 15), rep("very bad", 2),
  rep("very good", 61), rep("good", 72), rep("indifferent", 39),
  rep("bad", 16), rep("very bad", 3),
  rep("very good", 41), rep("good", 67), rep("indifferent", 35),
  rep("bad", 12), rep("very bad", 2),
  rep("very good", 77), rep("good", 80), rep("indifferent", 27),
  rep("bad", 10), rep("very bad", 3),
  rep("very good", 31), rep("good", 31), rep("indifferent", 31),
  rep("bad", 4), rep("very bad", 7),
  rep("very good", 52), rep("good", 70), rep("indifferent", 28),
  rep("bad", 4), rep("very bad", 3)
  )

JointSports_full <-
  data.frame(id = id,
             year = year,
             grade = grade,
             gender = gender,
             opinion = opinion) %>%
  mutate(opinion = fct_relevel(opinion,
                               "very bad",
                               "bad",
                               "indifferent",
                               "good",
                               "very good"))
```

Une fois que l'on a un jeu de données complet sous la main, il est possible de créer ce qu'on appelle un **tableau de contingence**, c'est-à-dire ici un tableau qui récapitule numériquement les effectifs à la croisée des deux variables qui nous intéressent. Pour faire cela, on peut utiliser la fonction `table()` en suivant différentes méthodes montrées ci-dessous. (Le code montré ci-dessous aboutit aux mêmes informations que celles montrées sur la Figure 4.12 ci-dessus.)

```
# 1ère méthode
tab <-
  with(JointSports_full,
       table(opinion, gender))

# 2ème méthode
tab <- table(JointSports_full$opinion, JointSports_full$gender)

# Visualisation du tableau de contingence
tab
```

```
##
##                Boy Girl
##   very bad      15   11
##   bad           41   45
##   indifferent  140  123
##   good         188  289
##   very good    126  293
```

Un tableau de contingence permet donc de comparer des effectifs en fonction de plusieurs modalités et variables à la fois. Le problème, lorsqu'on utilise des effectifs, est que certaines comparaisons peuvent être difficiles à faire lorsque les effectifs totaux liés aux différentes modalités ne sont pas comparables. Par exemple, dans le résultat montré ci-dessus, l'effectif total des filles est de 761 alors que celui des garçons est de 510, ce qui rend difficile la comparaison des garçons et des filles pour les différents types d'opinion recensés dans l'enquête danoise présentée plus haut. C'est pour cela qu'il convient, dans certains cas, de calculer les proportions correspondant à ces différents effectifs. Pour ce faire, on peut :

- Utiliser la fonction `prop.table()`, qui va convertir en proportions les effectifs montrés plus haut en considérant l'effectif total de tout le tableau :

```
round(prop.table(tab) * 100, digits = 2)
```

```
##
##                 Boy  Girl
##   very bad     1.18  0.87
##   bad          3.23  3.54
##   indifferent 11.01  9.68
##   good        14.79 22.74
##   very good    9.91 23.05
```

- Utiliser la fonction `lprop()` du package `questionr`, qui va convertir en proportions les effectifs montrés plus haut en considérant l'effectif total de chaque ligne du tableau :

```
library(questionr)
lprop(tab)
```

```
##
##                Boy   Girl  Total
##   very bad     57.7  42.3 100.0
##   bad          47.7  52.3 100.0
##   indifferent  53.2  46.8 100.0
##   good         39.4  60.6 100.0
##   very good    30.1  69.9 100.0
##   All          40.1  59.9 100.0
```

- Utiliser la fonction `cprop()` du package `questionr`, qui va convertir en proportions les effectifs montrés plus haut en considérant l'effectif total de chaque colonne du tableau :

```
library(questionr)
cprop(tab)
```

```
##
##                Boy   Girl  All
##   very bad      2.9   1.4   2.0
##   bad           8.0   5.9   6.8
##   indifferent  27.5  16.2  20.7
##   good         36.9  38.0  37.5
##   very good    24.7  38.5  33.0
##   Total       100.0 100.0 100.0
```

Il convient de noter que les proportions données par ces différentes fonctions doivent être utilisées selon les comparaisons que l'on veut faire. L'analyse descriptive consiste alors à voir si, tant d'un point de vue graphique que numérique, on observe des différences de scores particulières entre les modalités d'une variable qualitative en fonction des modalités de l'autre variable qualitative. Si l'on considère le dernier tableau de résultats ci-dessus, on peut par exemple observer une très légère tendance à ce que les garçons soient davantage polarisés, par rapport aux filles, sur des opinions négatives vis-à-vis des pratiques sportives mixtes, alors que les filles seraient légèrement plus polarisées que les garçons sur des opinions positives, ce qui n'empêche pas que, pour les deux genres, il y a une polarisation principale sur des opinions neutres à positives.

Si les proportions permettent en principe de mieux comparer les effectifs de sous-groupes (e.g., les opinions par groupe de genre dans l'exemple ci-dessus) lorsque les effectifs des groupes parents sont de tailles différentes (comme c'est le cas pour les groupes de genre dans l'exemple ci-dessus), il convient tout de même de faire attention aux conclusions que l'on tire lorsqu'on s'en tient à une analyse globale, car ces conclusions dépendent de la manière dont les individus des groupes parents sont répartis dans chacun des sous-groupes. Un exemple connu qui permet d'illustrer cette vigilance à avoir lorsqu'on étudie le croisement de variables qualitatives est le cas du taux de réussite des femmes à l'université de Berkeley en 1973, qui apparaissait bien inférieur à celui des hommes lorsqu'on considérait les taux de réussite par genre à l'échelle de l'ensemble de l'université (Bickel et al., 1975), avec 30.3 % de réussite chez les femmes contre 44.5 % de réussite chez les hommes (cf. Figure 4.13).

Toutefois, une analyse par département permettait de voir que les taux de réussite des femmes étaient en réalité supérieurs voire nettement supérieurs à ceux des hommes dans la plupart des départements (cf. Figure 4.14).

Cette situation, qui peut paraître étonnante, illustre à nouveau ce qu'on appelle un paradoxe de Simpson. Dans le cas présent, l'apparent paradoxe s'explique par le fait que, contrairement aux hommes, la majorité des femmes avaient candidaté dans des départements qui étaient très sélectifs, c'est-à-dire où le taux de réussite était faible (il l'était aussi pour les hommes). Très peu de femmes avaient candidaté là où les taux de réussite étaient très élevés (pour les femmes comme pour les hommes). Ceci est illustré sur la Figure 4.15. On voit bien que la majorité des femmes étaient inscrites dans les départements ici montrés avec les lettres allant de C à F, or ces départements étaient associés à des taux de réussite inférieurs à 40 % seulement, que cela soit pour les hommes ou pour les femmes (cf. Figure 4.14).

Autrement dit, les hommes se retrouvaient avec un pourcentage de réussite global bien meilleur que celui des femmes seulement parce que, comparativement aux femmes, les hommes s'étaient en proportion davantage engagés dans les départements où les taux de réussite étaient bien meilleurs.

Au-delà du tableau de contingence, plusieurs outils statistiques sont disponibles pour quantifier le lien entre les modalités de deux variables qualitatives (Jané et al., 2023). On s'intéresse ici plus spécifiquement au coefficient Phi (Φ), au V de Cramer, à la différence de risque, au risque relatif, et au rapport des cotes. À noter que tous ces indices statistiques, excepté le V de Cramer, sont essentiellement faits pour analyser le lien entre deux variables qualitatives dont l'une ne présenterait que deux modalités (on parle alors de variable binaire).

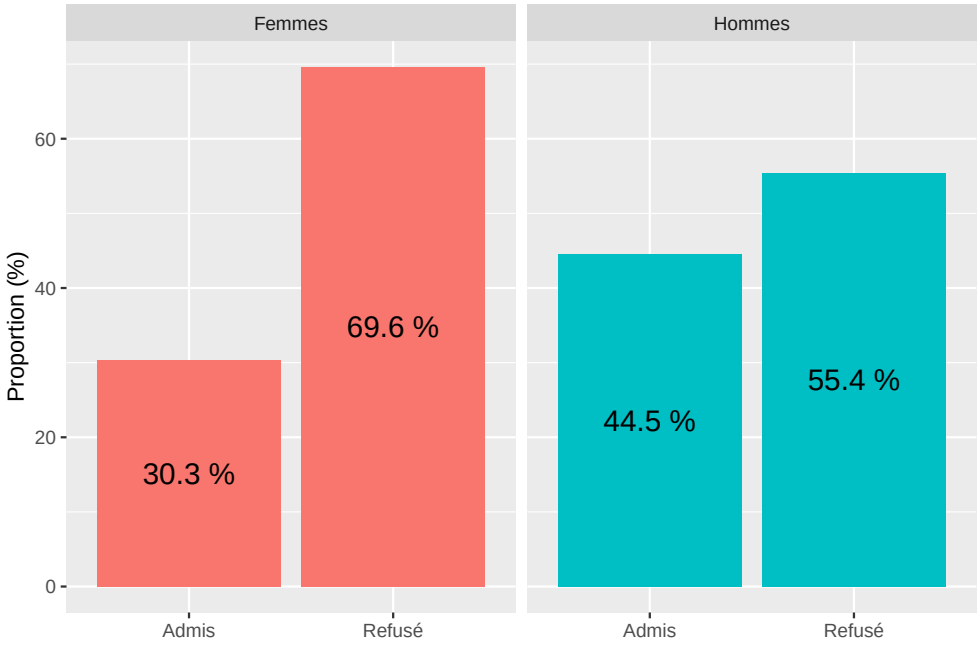

Figure 4.13: Taux de réussite des étudiants femmes et hommes à l'Université de Berkeley en 1973, approche globale (Bickel et al., 1975)

Figure 4.14: Taux de réussite des étudiants femmes et hommes à l'Université de Berkeley en 1973, approche par département (Bickel et al., 1975)

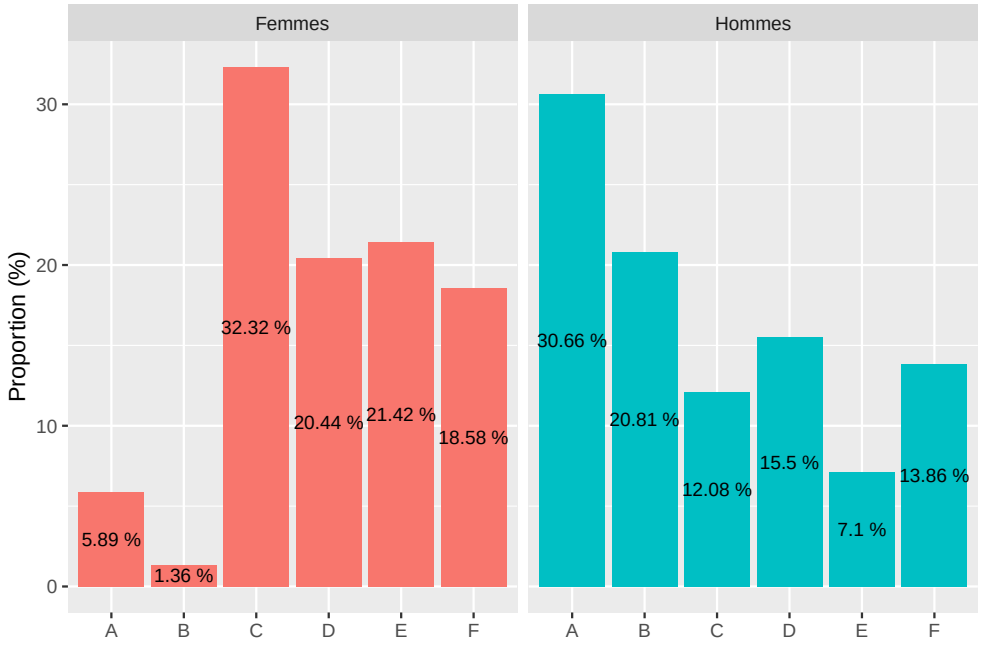

Figure 4.15: Distribution des étudiants femmes et hommes par département à l'Université de Berkeley en 1973 (Bickel et al., 1975)

Coefficient Phi

Le coefficient Φ est une valeur qui peut être comprise entre -1 et 1 (ou entre 0 et 1 selon les manières de l'obtenir) et indique la force de la relation entre deux variables qualitatives binaires. Toutefois, seule la valeur absolue (entre 0 et 1 donc) est à interpréter. Le signe (si jamais différents signes peuvent apparaître selon le calcul utilisé) ne fait que renseigner sur la diagonale du tableau de contingence où les individus sont davantage polarisés. Pour comprendre comment Φ peut être calculé, prenons les tableaux de contingence montrés ci-dessous (Figure 4.16) :

Phi = 0

	X=1	X=2
Y=1	n11	n12
Y=2	n21	n22

Phi = 1

	X=1	X=2
Y=1	n11	n12
Y=2	n21	n22

Phi = −1

	X=1	X=2
Y=1	n11	n12
Y=2	n21	n22

Figure 4.16: Tableaux de contingence schématiques pour comprendre le calcul de Phi

On note sur ces tableaux qu'une valeur absolue de Φ élevée sera attendue dans le cas où il y aura relativement beaucoup d'individus dans une diagonale et relativement peu d'individus dans l'autre diagonale (cf. tableaux du milieu et de droite ci-dessus). Si l'on prend comme base l'un des tableaux de contingence montrés ci-dessus, la formule du coefficient Φ est la suivante (Jané et al., 2023) :

$$\Phi = \frac{(n_{11} \cdot n_{22}) - (n_{21} \cdot n_{12})}{\sqrt{(n_{11} + n_{21}) \cdot (n_{12} + n_{22}) \cdot (n_{11} + n_{12}) \cdot (n_{21} + n_{22})}}$$

Une autre formule pour calculer Φ (donnant systématiquement la valeur absolue du coefficient) pourrait être aussi la suivante (Jané et al., 2023) :

$$\Phi = \sqrt{\frac{\chi^2}{n}},$$

avec χ^2 la statistique "chi-carré" et n le nombre total d'individus. Pour calculer Φ dans R, on peut utiliser la fonction `phi()` du package `effectsize` :

```
tab <- table(mtcars$am, mtcars$vs)
effectsize::phi(tab)
```

```
## Phi (adj.) |       95% CI
## -------------------------
## 0.00       | [0.00, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

V de Cramer

Le V de Cramer est la forme généralisée de la force d'association entre deux variables qualitatives. Il peut donc être calculé à partir d'un tableau de contingence de n'importe quelle taille, prendre des valeurs absolues entre 0 et 1, et est ainsi l'équivalent du coefficient Φ dans le cas d'un tableau de contingence 2 x 2 (2 variables qualitatives x 2 modalités chacune). La formule du V de Cramer est la suivante (Jané et al., 2023) :

$$V = \sqrt{\frac{\chi^2}{n \cdot (k-1)}},$$

avec χ^2 la statistique « chi-carré », n le nombre total d'individus, et k le nombre de modalités de la variable qui a le moins de modalités. Pour calculer le V de Cramer dans R, on peut utiliser la fonction `cramers_v()` du package `effectsize` :

```
tab <- table(JointSports_full$gender, JointSports_full$opinion)
effectsize::cramers_v(tab)
```

```
## Cramer's V (adj.) |       95% CI
## ---------------------------------
## 0.17              | [0.11, 1.00]
##
## - One-sided CIs: upper bound fixed at [1.00].
```

Différence de risque

La différence de risque est une différence de proportions. La notion de risque peut ne pas être très pertinente pour certains contextes, mais elle l'est particulièrement dans un contexte de santé. Pour comprendre le calcul du risque puis de la différence de risque, regardons le tableau ci-dessous (Figure 4.17).

Dans le tableau ci-dessus, chaque case contenant un « n » est un effectif. Si l'on désire connaître le risque d'être malade dans le groupe contrôle (p_C) et dans le groupe traité (p_T), on fait les calculs suivants :

	En bonne santé	Malade
Groupe contrôle	nC0	nC1
Groupe traité	nT0	nT1

Figure 4.17: Tableau de contingence schématique pour comprendre le calcul des risques et des cotes

$$p_C = \frac{n_{C1}}{n_{C0} + n_{C1}}$$

$$p_T = \frac{n_{T1}}{n_{T0} + n_{T1}}$$

La différence de risque d'être malade pour le groupe contrôle par rapport au groupe traité de l'exemple, notée DR, est alors :

$$p_C - p_T$$

Avec R, la différence de risque peut être obtenue avec le package `riskCommunicator`. Il s'agit d'un package qui n'existe pas sur CRAN, mais seulement sur Github. Pour pouvoir l'installer à partir de Github, il faut d'abord installer un package qui permet ce type d'installation, tel que le package `devtools` :

```
install.packages("devtools")
```

Le package `riskCommunicator` peut alors être installé en suivant la procédure ci-dessous :

```
devtools::install_github("jgrembi/riskCommunicator")
```

Pour utiliser le package `riskCommunicator`, il faut bien sûr ensuite le charger :

```
library(riskCommunicator)
```

Pour l'exemple, chargeons le jeu de données `cvdd` associé au package `riskCommunicator`:

```
data(cvdd)
```

Il s'agit d'un jeu de données avec plusieurs variables qualitatives, dont plusieurs sont plus précisément binaires (i.e., qui ne contiennent que deux modalités). Admettons que l'on souhaiterait connaître la différence de risque de mortalité cardiovasculaire (`cvd_dh`) sur une période de suivi donnée, cela en fonction de si l'on a développé une hypertension ou non (`HYPERTEN`). Pour cela, on peut utiliser la fonction `gComp()` du package `riskCommunicator` comme suit :

```
gComp(data = cvdd, X = "HYPERTEN", Y = "cvd_dth", outcome.type = "binary")
```

```
## Formula:
## cvd_dth ~ HYPERTEN
##
## Parameter estimates:
##                              HYPERTEN1_v._HYPERTEN0 Estimate (95% CI)
## Risk Difference                                     0.132 (0.098, 0.159)
## Risk Ratio                                          1.411 (1.275, 1.527)
## Odds Ratio                                          1.753 (1.505, 1.977)
## Number needed to treat/harm                                      7.559
```

Dans le résultat obtenu, la différence de risque est de 0.132, soit 13.2 %. À ce stade, il est fondamental de comprendre que le résultat obtenu dépend de la manière dont les modalités sont ordonnées dans les variables de type facteur utilisées pour faire les calculs. Regardons comment les modalités sont organisées pour les variables qui nous concernent :

```
levels(cvdd$HYPERTEN)
```

```
## [1] "0" "1"
```

```
levels(cvdd$cvd_dth)
```

```
## [1] "0" "1"
```

On remarque qu'à chaque fois, "0" est la première modalité, et "1" est la seconde modalité. C'est important de le savoir pour interpréter le résultat ensuite car la fonction `gComp()` calcule par défaut la différence de risque en faisant le risque de présenter la modalité 2 de la variable Y quand on a la modalité 2 de la variable X (ici le risque de mourir quand on est hypertendu) moins le risque de présenter la modalité 2 de la variable Y quand on a la modalité 1 de la variable X (ici le risque de mourir quand on n'est pas hypertendu). On peut d'ailleurs vérifier le calcul manuellement :

```
# Obtention d'un tableau de contingence avec proportions
tab <- table(cvdd$HYPERTEN, cvdd$cvd_dth)
risks <- questionr::lprop(tab)
colnames(risks) <- c("vivant", "mort", "total")
rownames(risks) <- c("non-hypertendu", "hypertendu", "total")
risks
```

```
##
##                  vivant mort  total
##   non-hypertendu  67.8   32.2 100.0
##   hypertendu      54.6   45.4 100.0
##   total           58.2   41.8 100.0
```

```
# Calcul de la différence de risque de mortalité
# chez les hypertendus vs non-hypertendus (en %)
45.4-32.2
```

```
## [1] 13.2
```

Risque relatif

Le risque relatif, noté RR, reprend les mêmes informations que celles utilisées pour la différence de risque (p_T et p_C). La différence est qu'il ne s'agit plus d'une différence, mais d'un rapport entre les risques présentés par les groupes comparés. Sa formule pour comparer le groupe contrôle au groupe traité dans notre exemple débuté plus haut est la suivante :

$$RR = \frac{p_C}{p_T}$$

Avec R, le risque relatif peut être obtenu avec le package `riskCommunicator` tel que montré pour le calcul de la différence de risque, en faisant toujours bien attention à quel risque est divisé par quel risque selon l'analyse qui est réellement souhaitée. Dans notre exemple débuté plus haut, il s'agit du même calcul que pour la différence de risque sauf qu'il ne s'agit plus d'un signe moins mais d'une division. Les résultats obtenus plus haut avec l'exemple de la différence de risque montrent aussi le risque relatif, qui est tel que le risque de mourir chez les hypertendus est 1.41 fois plus élevé que chez les non-hypertendus.

Rapports des cotes

Le rapport des cotes, noté OR pour *Odds ratio* en anglais, est une statistique dont le calcul peut à nouveau se comprendre à l'aide d'un tableau de contingence. Si l'on reprend le tableau de la Figure 4.17, on peut dire que la cote pour le fait d'être en bonne santé versus être malade dans le groupe contrôle est n_{C0}/n_{C1} et la cote pour le fait d'être en bonne santé versus malade dans le groupe traité est n_{T0}/n_{T1}. Le rapport des cotes pour le groupe contrôle versus le groupe traité dans ce cas-là peut se calculer comme ceci :

$$OR = \frac{n_{C0}/n_{C1}}{n_{T0}/n_{T1}}$$

Le rapport des cotes est toujours 0. Une valeur en-dessous de 1 signifie « moins de chances/risques », une valeur de 1 signifie « chances/risques équivalent(e)s », et une valeur au-dessus de 1 signifie « plus de chances ou de risques ». Le chiffre obtenu renseigne toujours sur un niveau de chances pour le groupe au numérateur par rapport au niveau de chances concernant le groupe au dénominateur (cf. formule ci-dessus). Avec R, le rapport des cotes peut être obtenu avec le package `riskCommunicator` tel que montré pour le calcul de la différence de risque, en faisant cette fois attention à quelle cote est divisée par quelle cote selon l'analyse qui est réellement souhaitée. Dans notre exemple débuté plus haut, le rapport des cotes est de 1.75. Cela signifie ici que la cote pour le fait de mourir (versus le fait de vivre) est 1.75 fois plus élevée quand on est hypertendu que lorsqu'on ne l'est pas. Pour comprendre le calcul, on peut le refaire manuellement :

```
# Obtention d'un tableau de contingence avec effectifs
tab <- table(cvdd$HYPERTEN, cvdd$cvd_dth)
colnames(tab) <- c("vivant", "mort")
rownames(tab) <- c("non-hypertendu", "hypertendu")
tab
```

```
##
##                   vivant mort
##   non-hypertendu    781  371
##   hypertendu       1685 1403
```

```
# Calcul des cotes pour les "chances" de mourir selon le groupe d'appartenance
cote_hypertendus <- 1403/1685
```

```
cote_non_hypertendus <- 371/781

# Calcul du rapport des cotes
OR <- cote_hypertendus / cote_non_hypertendus
OR
```

```
## [1] 1.75281
```

4.3 Relation entre une variable quantitative et une variable qualitative

Lorsqu'on analyse une variable quantitative en fonction d'une variable qualitative, on peut avoir des données quantitatives qui sont non appariées (étude de type *between-subject design* en anglais) ou appariées (étude de type *within-subject design* en anglais). Avoir des données non appariées signifie que les données quantitatives correspondant aux différentes modalités de la variable qualitative étudiée ne sont pas liées. Un exemple simple, pour ce premier cas, peut être l'analyse de la taille des individus en fonction du sexe. Dans ce cas, les données quantitatives de taille pour les personnes de sexe masculin, de sexe féminin, et de sexe indéterminé, proviendront forcément d'individus différents et ne formeront donc pas des paires. S'agissant des cas de données appariées, ils se retrouvent dans les études où plusieurs individus sont évalués plusieurs fois dans des conditions similaires ou différentes et que l'on cherche à comparer. En sciences du sport, un exemple relativement classique est de tester la performance d'endurance (variable quantitative) en ayant pris (condition de test) ou non (condition contrôle) une substance potentiellement ergonénique, la prise de substance ou non étant les modalités d'une même variable qualitative de type condition. Dans ce cas là, tous les individus auront des données dans les deux conditions et ces données seront donc appariées (dépendantes).

Étant donné que les contraintes graphiques et les statistiques à calculer diffèrent selon que l'on est en présence de (i) deux groupes (1 variable quantitative x 1 variable qualitative avec 2 modalités) ou (ii) trois groupes et plus (1 variable quantitative x 1 variable qualitative avec 3 modalités ou plus), les procédures graphiques et calculatoires présentées ci-après sont distinguées selon ces deux grands cas de figure.

4.3.1 Comparaison de deux groupes de données

4.3.1.1 Étudier graphiquement la relation

Lorsque l'on cherche à explorer la relation qu'il peut y avoir entre une variable quantitative et une variable qualitative, il peut être intéressant de visualiser la distribution de la variable quantitative en fonction de chaque modalité de la variable qualitative. Pour faire cela, il a été proposé dans la littérature d'user de graphiques appelés *raincloud plots* qui combinent les avantages de plusieurs techniques graphiques et statistiques et par là même pallient les manques ou défauts inhérents à chacune de ces techniques (Allen et al., 2019). Les blocs de code ci-dessous, et les Figures 4.18 et 4.19, proposent plusieurs options de mise en oeuvre de ce type de graphiques selon la situation d'étude (avec données non appariées et appariées).

Données non appariées

La Figure 4.18, qui est associée au bloc de code ci-dessous, montre un graphique approprié pour des données non appariées.

```
# Configuration du jeu de données pour l'exemple avec données non appariées.
# Il s'agit ici d'obtenir le jeu de données `iris` avec seulement deux modalités
# de la variable `Species`, cela en enlevant la modalité `virginica`.
iris_two_species <-
  iris %>% filter(Species != "virginica")
```

```
ggplot(data = iris_two_species, aes(x = Species, y = Sepal.Length)) +
  geom_rain(point.args = rlang::list2(alpha = 0.3)) +
  stat_summary(
    geom = "errorbar",
    fun.data = "mean_sdl",
    fun.args = list(mult = 1),
    size = 1.1,
    width = 0.06
    ) +
  stat_summary(
    geom = "point",
    fun = "mean",
    size = 3
    )
```

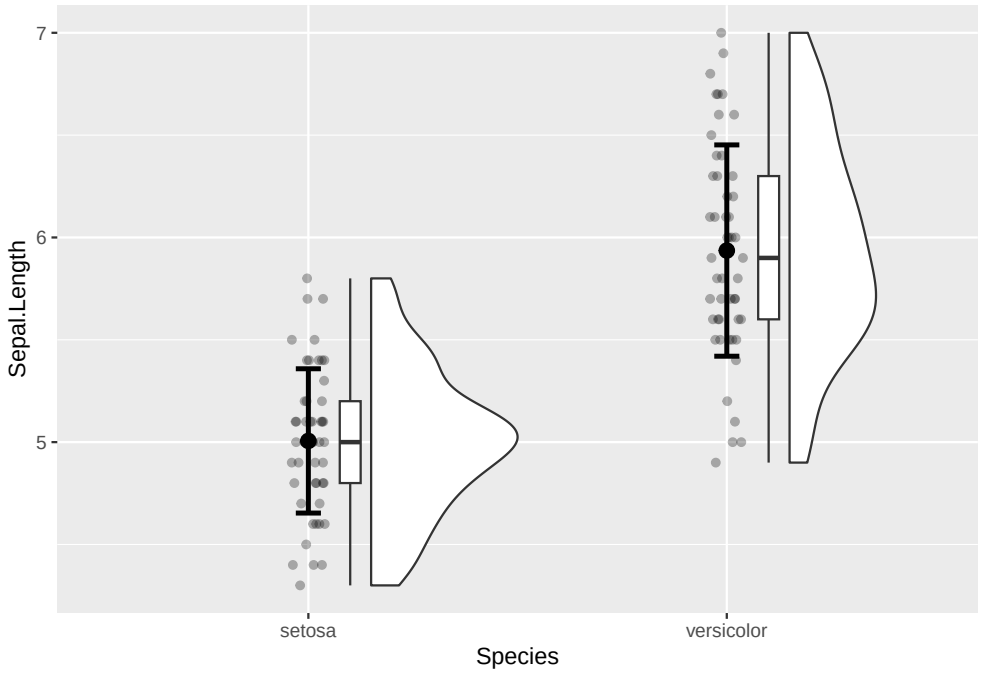

Figure 4.18: Exemple de graphique pour une comparaison de deux groupes de valeurs non-appariés (indépendants)

On note que pour montrer les moyennes et écarts-types sur la Figure 4.18, il a fallu utiliser la fonction `stat_summary()` du package `Hmisc` (qui est chargé automatiquement lors du chargement de `ggplot2` après que `Hmisc` ait été installé). Cette fonction sert à visualiser un résumé statistique. Pour montrer les moyennes, on utilise `fun = "mean"`, alors que pour montrer des écarts-types, on utilise `fun.data = "mean_sdl"`. Pour les écarts-types, il faut ajouter `fun.args = list(mult = 1)` pour bien montrer une barre d'erreur ne correspondant

qu'à un seul multiple de l'écart-type, car par défaut c'est deux fois l'écart-type qui est montré. On peut choisir la forme avec laquelle présenter le résumé statistique grâce à l'argument `geom`.

Données appariées

La Figure 4.19 montre une proposition de graphique dans le cas de l'étude d'une comparaison de deux groupes de données appariées. Pour cela, nous avons installé le package `PairedData` et chargé le jeu de données associé, appelé `Blink`. Pour information, `Blink` contient des données de taux de clignotement des yeux obtenues chez 12 sujets et dans deux conditions différentes : une tâche où il fallait diriger un stylo selon une trajectoire rectiligne (modalité `Straight`), et une tâche où il fallait diriger un stylo selon une trajectoire présentant des oscillations (`Oscillating`). Ce jeu de données a été un peu transformé pour pouvoir passer à l'étape de la conception graphique, comme montré ci-dessous.

```
# Charger le package
library(PairedData)

# Charger le jeu de données
data(Blink)

# Configurer le jeu de données pour l'exemple avec données appariées
Blink2 <-
  Blink %>%
  pivot_longer(cols = c(Straight, Oscillating),
               names_to = "Condition",
               values_to = "Blink_rate") %>%
  mutate(Condition = fct_relevel(Condition, "Straight", "Oscillating"))
```

```
ggplot(data = Blink2, aes(x = Condition, y = Blink_rate)) +
  geom_rain(
    rain.side = "f1x1",
    id.long.var = "Subject",
    point.args = rlang::list2(alpha = 0.3)
  ) +
  stat_summary(
    aes(group = 1),
    fun = "mean",
    geom = "line",
    linewidth = 1
  ) +
  stat_summary(
    geom = "errorbar",
    fun.data = "mean_sdl",
    fun.args = list(mult = 1),
    size = 1.1,
    width = 0.06
    ) +
  stat_summary(
    geom = "point",
    fun = "mean",
    size = 3
    )
```

Le graphiques montré ci-dessus pour données appariées contient des éléments graphiques supplémentaires par rapport à celui proposé pour les données non appariées : des lignes reliant

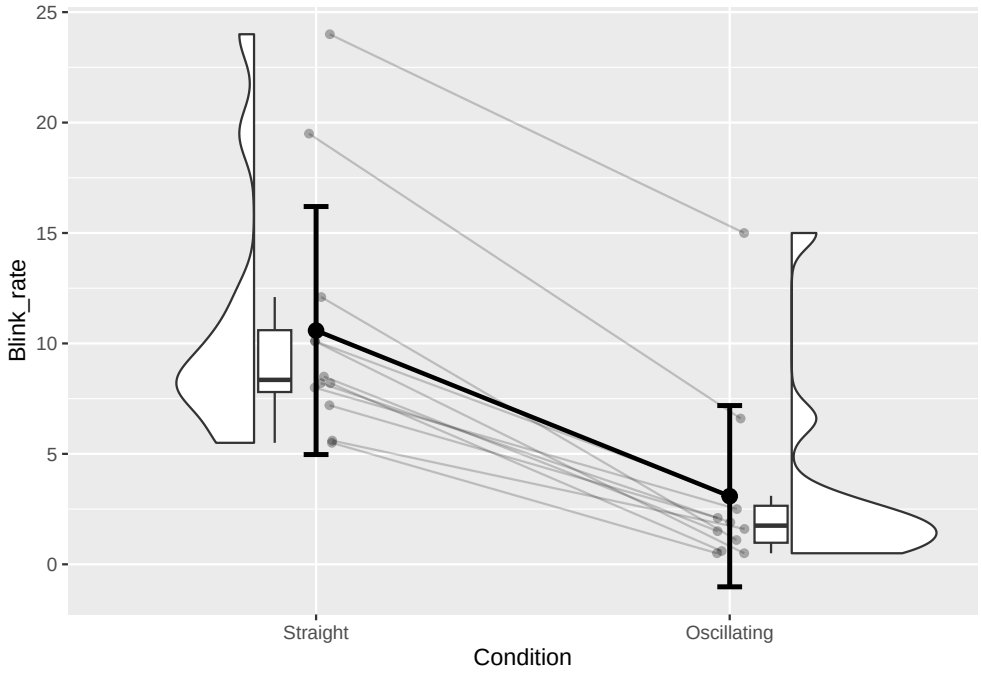

Figure 4.19: Exemple de graphique pour une comparaison de deux groupes de valeurs appariés (dépendants)

les données individuelles, et une ligne reliant les statistiques qui représentent les tendances centrales (ici les moyennes). Ces deux types d'éléments graphiques sont importants pour explicitement montrer qu'il y a un lien entre les deux conditions de mesure (il s'agit des mêmes individus et donc du même groupe), et cela permet de mettre en évidence à la fois les trajectoires individuelles et celle du groupe entre les deux conditions de mesure.

Pour obtenir ces éléments supplémentaires, il a fallu configurer l'argument `id.long.var` de la fonction `geom_rain()` pour indiquer le nom de la variable sur laquelle se baser pour garder l'appariement des données individuelles entre les deux conditions et ainsi permettre un bon tracé des lignes individuelles. La ligne reliant les moyennes a été ajoutée grâce à la fonction`stat_summary()` avec la particularité d'indiquer `1` pour l'argument `group` de la fonction `aes()` associée à la fonction `stat_summary()`. Enfin, l'argument `rain.side` de la fonction `geom_rain()` a été configuré de telle sorte à avoir les données des deux groupes en vis-à-vis, mais un autre choix aurait pu être fait.

4.3.1.2 Étudier numériquement la relation

De manière générale, étudier la relation entre une variable quantitative et une variable qualitative revient souvent à comparer les valeurs que prend la variable quantitative en fonction des modalités de la variable qualitative. L'observation de différences de scores entre les modalités pourrait alors indiquer qu'il y a un lien entre la variable qualitative (qu'on pourrait appeler **variable facteur**) et la variable quantitative (qu'on pourrait appeler **variable réponse**). À noter que la démonstration d'un lien de cause à effet entre la variable quantitative et la variable qualitative ne pourra être effective que si l'on a sciemment fait varier les modalités de la variable qualitative pour en observer la conséquence sur les valeurs de la variable quantitative.

De prime abord, l'analyse qui pourrait être envisagée pour comparer deux groupes serait d'utiliser simplement la différence entre les moyennes des deux groupes. Toutefois, en se

restreignant à cela, il pourrait être difficile de porter un jugement sur la grandeur relative de la différence entre les groupes comparés, qu'on pourrait appeler **taille d'effet**. Il serait également difficile de comparer cette taille d'effet avec celles observées dans d'autres études, en particulier celles traitant de thématiques différentes, car en étant calculée de la sorte, la taille d'effet serait inhérente à la nature des variables investiguées et à la grandeur des valeurs mesurées dans l'étude. Il est donc intéressant, dans ce genre de situations, de standardiser la différence de scores obtenue entre les deux groupes. Dans la littérature, la procédure de standardisation a été très développée pour la comparaison de moyennes. Pour cette raison, les sous-parties suivantes qui traitent de la comparaison de deux groupes présentent essentiellement les calculs pour obtenir une taille d'effet en vue de comparer des moyennes. Ces calculs sont repris de l'article de Lakens (2013). Quelques mots seront toutefois donnés pour les situations pour lesquelles ces calculs risqueraient de ne pas être appropriés.

4.3.2 Cas de deux groupes de données indépendants (données non appariées)

d_s et d_{av} de Cohen

Classiquement, l'indice statistique utilisé pour calculer une différence de moyennes de manière standardisée entre deux groupes de données non appariées, à partir d'échantillons de population, est le d_s de Cohen. Cette statistique se calcule en faisant la différence entre les moyennes des deux groupes à comparer, et en divisant cette différence par l'écart-type combiné des deux groupes. Ce calcul est montré ci-dessous :

$$d_s = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{(N_1-1)\hat{\sigma}_1^2+(N_2-1)\hat{\sigma}_2^2}{N_1+N_2-2}}},$$

avec $\overline{X}_1$ et $\overline{X}_2$ les moyennes des deux groupes comparés, N_1 et N_2 les effectifs respectifs des deux groupes comparés, et $\hat{\sigma}_1^2$ et $\hat{\sigma}_2^2$ les variances respectives des deux groupes comparés. On peut remarquer qu'au dénominateur, la variance d'un groupe donné est multipliée par un coefficient calculé à partir de l'effectif du groupe, cela pour que le poids de la variance d'un groupe dans le calcul final soit ajusté par rapport à la taille du groupe représenté. Dans R, si l'on voulait calculer d_s manuellement, cela donnerait ceci (à partir du jeu de données `iris_two_species`) :

```
# Calcul des paramètres
X1 <- iris_two_species %>% filter(Species == "setosa") %>% pull(Sepal.Length) %>% mean()
X2 <- iris_two_species %>% filter(Species == "versicolor") %>% pull(Sepal.Length) %>% mean()
SD1 <- iris_two_species %>% filter(Species == "setosa") %>% pull(Sepal.Length) %>% sd()
SD2 <- iris_two_species %>% filter(Species == "versicolor") %>% pull(Sepal.Length) %>% sd()
N1 <- iris_two_species %>% filter(Species == "setosa") %>% pull(Sepal.Length) %>% length()
N2 <- iris_two_species %>% filter(Species == "versicolor") %>% pull(Sepal.Length) %>% length()

# Calcul de ds
ds <- (X1 - X2) / sqrt(((N1-1) * SD1^2 + (N2-1) * SD2^2) / (N1+N2-2))
ds
```

```
## [1] -2.104197
```

Heureusement, le d_s de Cohen peut être facilement calculé à l'aide de la fonction `cohens_d()` du package `effectsize`, qui nécessite d'être installé puis chargé avant d'être utilisé :

```
library(effectsize)
cohens_d(
  Sepal.Length ~ Species,
  data = iris_two_species,
  paired = FALSE,
  pooled_sd = TRUE
  )
```

```
## Cohen's d |         95% CI
## -------------------------
## -2.10     | [-2.59, -1.61]
##
## - Estimated using pooled SD.
```

Dans cet exemple, on remarque qu'on a bien cherché à savoir comment les données de la variable `Sepal.Length` pouvaient différer en fonction (`~`) des modalités de la variable `Species`. Si la fonction nous donne un résultat, il faut toutefois bien faire attention au sens du calcul qui a été réalisé. Configurée de la sorte, la fonction `cohens_d()` réalise la différence **modalité 1 - modalité 2**. Il faut donc savoir quelle est la modalité 1 et quelle est la modalité 2 dans la variable `Species` pour ensuite pouvoir interpréter le signe du résultat, qui est négatif ici avec la valeur de -2.10. Pour ce faire, on peut utiliser la fonction `levels()` :

```
levels(iris_two_species$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

L'ordre des modalités affichées nous indique que `setosa` est la modalité 1, et que `versicolor` est la modalité 2. (On remarque par ailleurs que le jeu de données `iris_two_species` contient toujours trois modalités à la suite du filtrage précédent du jeu de données `iris` à l'aide de la fonction `filter()`). Par conséquent, le d_s de Cohen de -2.10 obtenu plus haut indique que la longueur des sépales (`Sepal.Length`) pour l'espèce `setosa` est inférieure à celles des sépales de l'espèce `versicolor`. Cette interprétation est en cohérence avec le graphique réalisé au préalable (cf. Figure 4.18). Si l'on avait voulu avoir le calcul inverse (`versicolor - setosa`), il aurait fallu reconfigurer l'ordre des modalités, par exemple à l'aide de la fonction `fct_relevel()` du package `forcats` comme montré à la fin du chapitre 3.

De manière importante, le calcul de d_s à partir d'un échantillon en vue d'avoir une estimation dans la population suppose que les variances des deux groupes dans la population d'intérêt soient similaires. Si cette supposition n'est pas valide ou est non désirée, alors il peut être préconisé de calculer un autre indicateur appelé d_{av} (cf. https://aaroncaldwell.us/TOSTERpkg/articles/SMD_calcs.html), lequel consiste à diviser la différence de moyennes par un écart-type moyen :

$$d_{av} = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{\hat{\sigma}_1^2 + \hat{\sigma}_2^2}{2}}}$$

En repartant des paramètres calculés précédemment, on peut manuellement obtenir cet indice comme ceci :

```
# Calcul de dav
dav <- (X1 - X2) / sqrt((SD1^2 + SD2^2) / 2)
dav
```

```
## [1] -2.104197
```

On peut obtenir le d_{av} avec le package `effectsize` comme ceci (avec `pooled = FALSE`) :

```
cohens_d(
  Sepal.Length ~ Species,
  data = iris_two_species,
  paired = FALSE,
  pooled_sd = FALSE
  )
```

```
## Cohen's d |         95% CI
## -------------------------
## -2.10     | [-2.60, -1.60]
##
## - Estimated using un-pooled SD.
```

On note que les valeurs de d_s et de d_{av} ne changent pas dans nos exemples car chaque modalité de la variable `Species` est associée au même nombre d'individus, ce qui revient à faire des calculs similaires pour les deux indices statistiques obtenus jusqu'à présent.

Malheureusement, il se trouve que d (d_s ou d_{av}) est indicateur qui est biaisé positivement lorsqu'il s'agit d'estimer la différence de moyennes standardisée à l'échelle de la population d'intérêt. Par « biaisé positivement », il faut comprendre que si une multitude d'études s'intéressait à calculer d et qu'on faisait la moyenne des résultats trouvés, alors on aurait une surestimation de la magnitude de la valeur de d dans la population (il s'agit d'un fait qui peut être démontré avec des simulations sur un ordinateur), cela notamment dans le cas de petits échantillons (N < 20), selon Hedges et Okins (1985) cités par Lakens (2013).

g_s et g_{av} de Hedges

Pour régler le problème de biais que pose le d de Cohen, un autre indicateur statistique a été proposé : le g de Hedges. Il s'agit en réalité du d de Cohen, mais qu'on modifie grâce à l'utilisation d'un facteur de correction. Une approximation de g est montrée ci-dessous (2013) :

$$g = d(1 - \frac{3}{4(N_1 + N_2) - 9}),$$

avec d le d_s ou le d_{av} de Cohen, et N_1 et N_2 les effectifs respectifs des deux groupes comparés. En réalité, le calcul exact du facteur de correction est relativement complexe, et est montré avec l'équation ci-dessous (Kelley, 2005) :

$$g = d\frac{\Gamma(\frac{df}{2})}{\sqrt{\frac{df}{2}}\Gamma(\frac{df-1}{2})}$$

avec Γ la loi Gamma, et df ce qu'on appelle le nombre de degrés de liberté (qui est ici égal au nombre total d'individus moins 2). Dans les faits, les différences entre d et g sont minimes, surtout avec $N > 20$ dans les groupes.

Dans R, le g_s de Hedges lié à un échantillon de population peut être calculé à l'aide de la fonction `hedges_g()` du package `effectsize` :

```
hedges_g(
  Sepal.Length ~ Species,
  data = iris_two_species,
  paired = FALSE,
  pooled_sd = TRUE
  )
```

```
## Hedges' g |         95% CI
## --------------------------
## -2.09     | [-2.57, -1.60]
##
## - Estimated using pooled SD.
```

Ici, la valeur ne change pas beaucoup par rapport à d_s car l'effectif n'est pas si petit que cela ($N_1 + N_2 = 100$ ici, ce qui fait que la correction appliquée au d_s de Cohen est minime).

Le g_{av} de Hedges lié à un échantillon de population peut aussi être calculé à l'aide de la fonction `hedges_g()` du package `effectsize` (avec `pooled = FALSE`) :

```
hedges_g(
  Sepal.Length ~ Species,
  data = iris_two_species,
  paired = FALSE,
  pooled_sd = FALSE
  )
```

```
## Hedges' g |         95% CI
## --------------------------
## -2.09     | [-2.58, -1.58]
##
## - Estimated using un-pooled SD.
```

Par principe, **il pourrait être recommandé de toujours utiliser le g (g_s ou g_{av}) de Hedges** (Lakens, 2013).

Qualification de l'importance de la différence de moyennes standardisée

Une fois que l'on a calculé une taille d'effet, il est toujours intéressant d'essayer de formuler un jugement sur l'importance, l'ampleur de l'effet. En ce sens, des valeurs seuils ont été proposées dans la littérature (Cohen, 1988; in Lakens (2013)). Ces valeurs, valables pour interpréter des tailles d'effet dans le cadre d'une étude de type *between-subject design*, sont montrées dans le tableau ci-dessous.

Tableau 4.4: Termes pour qualifier la taille d'effet dans le cadre d'une comparaison de moyennes associées à des variables indépendantes

Petit	Moyen	Grand
>=0.2	>=0.5	>=0.8

La classification montrée dans ce tableau doit être utilisée avec précaution car en réalité, l'importance d'un effet s'apprécie aussi au regard du contexte dans lequel il s'applique. Les

valeurs du tableau donnent donc des repères généraux mais ne peuvent pas au final servir d'étalon universel (Lakens, 2013).

Il existe également une autre approche pour interpréter une valeur de taille d'effet : l'approche *Common Language explanation* (Lakens, 2013). Cette approche consiste à faire le lien entre la valeur de la taille d'effet et les probabilités de rencontrer des valeurs similaires ou supérieures dans un groupe en comparaison à un autre. Par exemple, lorsqu'on obtient un d_s de Cohen de 0.80, cela peut se traduire par le fait qu'il y a 71.4 % de chances qu'une personne prise au hasard dans le groupe avec la meilleure moyenne ait un score plus élevé qu'une personne qui serait prise au hasard dans le groupe avec la moyenne la plus basse des deux groupes. Kristoffer Magnusson a réalisé une page web qui permet d'utiliser l'approche *Common Language explanation* pour n'importe quelle valeur de taille d'effet dans le cadre d'une étude de type *between-subject design*. Jetez donc un oeil au lien suivant pour plus de détails : https://rpsychologist.com/cohend.

Delta de Glass

Dans certains cas où l'on souhaiterait comparer les scores de deux groupes indépendants pour tester l'effet de deux conditions expérimentales différentes, l'expérimentation en tant que telle peut influencer, au-delà de la moyenne, l'écart-type de la variable réponse dans un des deux groupes. On peut se trouver dans ce genre de situation lorsque l'on compare les données post-programme d'un groupe entraîné ou traité à celles d'un groupe contrôle. En effet, le groupe entraîné/traité peut voir son écart-type changé au terme d'un programme en raison d'une réponse individuelle hétérogène à ce programme, ce qui ne sera pas en principe le cas du groupe contrôle. Dans ce genre de situations, des indices statistiques autres que le d de Cohen ou le g de Hedges mériteraient d'être calculés, tels que le delta (Δ) de Glass (Lakens, 2013) :

$$\Delta = \frac{\overline{X}_1 - \overline{X}_2}{\hat{\sigma}_2}$$

Le calcul du Δ de Glass est dans l'idée le même que celui du d de Cohen, sauf que la différence entre les moyennes des deux groupes n'est pas divisée par un écart-type lié aux deux groupes, mais par l'écart-type d'un seul des deux groupes, qui serait en principe celui qui représenterait la condition contrôle ou la condition de référence. Une pratique souvent recommandée pour comparer dans ce genre de situation les scores post-programme de deux groupes (un groupe entraîné/traité et un groupe contrôle) serait d'utiliser l'écart-type des scores du groupe contrôle obtenu en pré-programme (Lakens, 2013).

Dans R, le Δ de Glass lié à un échantillon de population peut être calculé à l'aide de la fonction `glass_delta()` du package `effectsize`. Attention, l'écart-type utilisé dans le code suivant est celui de la variable quantitative associée à la modalité 2 de la variable qualitative, qui est toujours `versicolor` dans cet exemple (exemple dont le contexte n'est certes pas celui d'un programme dont on cherche l'effet, mais le principe d'utilisation du code reste le même).

```
glass_delta(Sepal.Length ~ Species,
            data = iris_two_species)
```

```
## Glass' delta |         95% CI
## -----------------------------
## -1.80        | [-2.29, -1.31]
```

Notons quand même que l'utilisation du Δ de Glass semble peu courante dans les études cherchant à tester l'effet d'un programme. L'une des difficultés liées à son utilisation est que cela suggère que les groupes à comparer soient identiques en pré-programme pour la variable étudiée. En général, des procédures un peu plus sophistiquées sont utilisées pour déterminer l'effet d'un programme, en particulier des procédures qui permettent de prendre en compte justement les différences qui peuvent exister entre les groupes en pré-programme.

Cas particuliers

Certaines situations peuvent rendre la comparaison de moyennes non pertinente. C'est par exemple le cas lorsque l'un des deux groupes (voire les deux) présentent des données aberrantes ou très extrêmes, en particulier en présence de petits échantillons. L'utilisation des moyennes dans ce cas pourrait ne pas être pertinente car ces moyennes ne refléteraient alors pas correctement les tendances centrales (et donc les groupes associés). Dans ces cas là, mieux vaut observer les médianes de chaque groupe et chercher à voir si les distributions sont en décalage ou non.

4.3.2.1 Cas de deux groupes de données dépendants (données appariées)

d_z et d_{av} de Cohen

Le calcul classique pour obtenir la taille d'effet désignant l'écart de moyennes entre deux groupes de données dépendants est celui du d_z (Lakens, 2013), qui est montré ci-dessous :

$$d_z = \frac{\overline{X}_{diff}}{\hat{\sigma}_{diff}},$$

$\overline{X}_{diff}$ désignant la moyenne des différences relatives à chaque pair de valeurs obtenues dans les deux conditions comparées, et $\hat{\sigma}_{diff}$ désignant l'écart-type de ces différences.

Dans R, on peut calculer manuellement d_z comme ceci (avec le jeu de données `Blink`) :

```
# Calcul des paramètres
diff <- Blink$Straight - Blink$Oscillating
mean_diff <- mean(diff)
sd_diff <- sd(diff)

# Calcul de dz
dz <- mean_diff / sd_diff
dz
```

```
## [1] 2.811462
```

Le d_z peut heureusement être obtenu à nouveau avec la fonction `cohens_d()` du package `effectsize`. Pour illustrer cela, nous allons utiliser le jeu de données `Blink2` crée plus haut pour l'exemple de graphique :

```
cohens_d(
  Pair(Blink_rate[Condition == "Straight"], Blink_rate[Condition == "Oscillating"]) ~ 1,
  data = Blink2
  )
```

```
## Cohen's d |       95% CI
## ------------------------
## 2.81      | [1.51, 4.09]
```

Parce que la valeur de d_z dépend de la corrélation entre les deux groupes de valeurs comparés, certains auteurs préfèrent rapporter un autre indice statistique qui n'a pas cette propriété et qui en ce sens pourrait être davantage comparable au d de Cohen obtenu dans les études avec

groupes indépendants. Parmi les candidats possibles, il y a notamment le d_{av} de Cohen (Lakens, 2013), dont la formule est montrée ci-dessous :

$$d_{av} = \frac{\overline{X}_{diff}}{\sqrt{\frac{\hat{\sigma}_1^2 + \hat{\sigma}_2^2}{2}}}$$

Ce calcul reviendrait donc à diviser la moyenne des différences (qui est égale à la différence de moyennes) par l'écart-type moyen relatif aux deux groupes de données. Manuellement cela donnerait (avec `Blink2`) :

```
# Calcul des paramètres
X1 <- Blink2 %>% filter(Condition == "Straight") %>% pull(Blink_rate) %>% mean()
X2 <- Blink2 %>% filter(Condition == "Oscillating") %>% pull(Blink_rate) %>% mean()
SD1 <- Blink2 %>% filter(Condition == "Straight") %>% pull(Blink_rate) %>% sd()
SD2 <- Blink2 %>% filter(Condition == "Oscillating") %>% pull(Blink_rate) %>% sd()
N1 <- Blink2 %>% filter(Condition == "Straight") %>% pull(Blink_rate) %>% length()
N2 <- Blink2 %>% filter(Condition == "Oscillating") %>% pull(Blink_rate) %>% length()

# Calcul de dav
dav <- (X1 - X2) / sqrt((SD1^2 + SD2^2) / 2)
dav
```

```
## [1] 1.525146
```

Pour obtenir d_{av} avec des données appariées et la fonction `cohens_d()`, il faut procéder comme s'il s'agissait de données non appariées :

```
cohens_d(
  Blink_rate ~ Condition,
  data = Blink2,
  paired = FALSE,
  pooled_sd = FALSE
  )
```

```
## Cohen's d |       95% CI
## ------------------------
## 1.53      | [0.58, 2.44]
##
## - Estimated using un-pooled SD.
```

g_z et g_{av} de Hedges

Comme pour les d de Cohen calculés dans le cadre de groupes indépendants, les d de Cohen obtenus avec groupes dépendants sont positivement biaisés. À nouveau, ces indices statistiques peuvent être corrigés avec le même facteur de correction montré pour le g de Hedges dans la partie sur les groupes dépendants. Toutefois, g_{av} ne serait malgré tout pas complètement non biaisé (Lakens, 2013).

Pour obtenir g_z :

```
hedges_g(
  Pair(Blink_rate[Condition == "Straight"], Blink_rate[Condition == "Oscillating"]) ~ 1,
  data = Blink2
  )
```

```
## Hedges' g |       95% CI
## ------------------------
## 2.61      | [1.41, 3.80]
```

Pour obtenir g_{av} :

```
hedges_g(
  Blink_rate ~ Condition,
  data = Blink2,
  paired = FALSE,
  pooled_sd = FALSE
  )
```

```
## Hedges' g |       95% CI
## ------------------------
## 1.47      | [0.56, 2.35]
##
## - Estimated using un-pooled SD.
```

Cas particuliers

Lorsque la présence de valeurs aberrantes peut fausser le calcul de la taille d'effet que l'on cherche à caractériser dans la population étudiée, une première idée pourrait être ici de faire la différence des médianes des deux groupes. Toutefois, comme le rappellent Weissegerber et al. (2015), cette approche n'est mathématiquement pas correcte pour caractériser l'évolution typique des scores étudiés. La bonne méthode est de calculer la médiane des différences. Le code ci-dessous présente donc le calcul de la médiane des différences, ici à partir du jeu de données `Blink`. La visualisation graphique correspondante pourrait être alors un *raincloud plot* pour une seule variable, celle des différences entre les deux conditions. Le code pour ce type de graphiques a été vu au chapitre précédent.

```
# Calcul de la médiane des différences
 Blink %>%
  mutate(Difference = Straight - Oscillating) %>%
  summarise(median_diff = median(Difference))
```

```
##   median_diff
## 1           7
```

4.3.3 Comparaison de trois groupes de données et plus

…

Chapitre 5

Régressions

5.1 Régression linéaire simple

Il est possible d'investiguer l'existence d'une relation linéaire entre deux variables en modélisant cette relation à l'aide d'une équation de type $Y = aX + b$, et en calculant certaines statistiques qui rendent compte du niveau de correspondance entre le modèle linéaire et les données étudiées. Ces statistiques sont le coefficient de détermination, noté R^2, et l'erreur typique d'estimation, dont on gardera l'acronyme anglais SEE (pour *Standard Error of Estimate*).

5.1.1 Le coefficient de détermination

Le coefficient de détermination, noté R^2, représente la part de variance de la variable Y expliquée par le modèle linéaire. La formule de ce coefficient peut être présentée comme ceci :

$$R^2 = 1 - \frac{Var(\hat{Y} - Y)}{Var(Y)} = 1 - \frac{Var(RES)}{Var(Y)},$$

où $\hat{Y}$ désigne les prédictions faites à partir du modèle, et Y désigne les valeurs réelles que l'on a cherché à prédire à partir du modèle. Le terme $\hat{Y} - Y$ (ou RES) doit se concevoir comme une variable contenant toutes les différences $\hat{Y}i - Yi$ qu'on appelle des **résidus**. Ainsi, le terme $Var(\hat{Y} - Y)$ désigne la variance des résidus (ou encore la variance des erreurs). Au final, le ratio $\frac{Var(\hat{Y}-Y)}{Var(Y)}$ traduit la part de variance non expliquée (non détectée) par le modèle, et le R^2 se calcule en faisant 1 moins ce ratio. (À noter qu'on peut trouver ailleurs d'autres manières de présenter ce coefficient R^2, avec des formules initiales différentes, mais mathématiquement, les méthodes restent équivalentes).

La Figure 5.1 illustre la notion de **résidu** et ce qu'elle représente dans le calcul du R^2. Sur cette figure, les points représentent les valeurs Yi en fonction des valeurs Xi, la ligne bleue représente le modèle de régression linéaire (i.e., toutes les valeurs $\hat{Y}i$ qui seraient prédites à partir du modèle et des valeurs Xi), et les segments rouges représentent les résidus (i.e., les différences qu'on a à chaque fois entre $\hat{Y}i$ et Yi). Pour un modèle donné, plus ces segments rouges seront nombreux et grands, plus cela signifiera que les erreurs de prédiction du modèle sont nombreuses et grandes, que la part de variance non expliquée par le modèle est grande, et que la valeur du R^2 pour ce modèle est éloignée de 1. Ainsi, le coeffient R^2 peut aller de la valeur 0 (signifiant que le modèle n'explique aucune variation de Y), à la valeur de 1 (signifiant que le modèle explique toute les variations de Y). Plus la valeur de R^2 d'un modèle linéaire se rapprochera de 1, plus cela suggérera que la relation étudiée est effectivement linéaire. Le coefficient de détermination R^2

associé à un modèle linéaire est mathématiquement lié au coefficient de corrélation de Pearson (r), r étant la racine carrée du R^2.

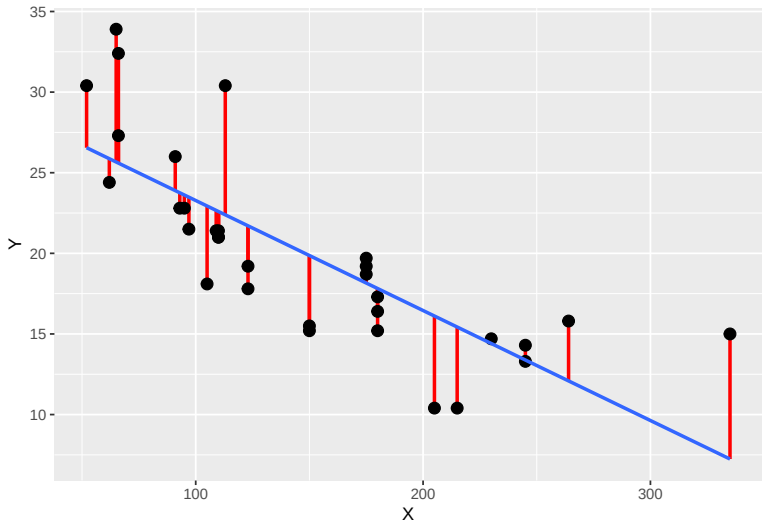

Figure 5.1: Illustration d'un modèle linéaire (en bleu) et de ses résidus (en rouge)

Pour déterminer le R^2 d'un modèle linéaire avec le logiciel R, il faut d'abord créer ce modèle à l'aide de la fonction `lm()`. L'usage simple de cette fonction, tel que montré ci-dessous, permet de prendre connaissance des coefficients du modèle. Dans les résultats issus de l'exemple ci-dessous, l'ordonnée à l'origine est située sous `(Intercept)`, et le coefficient directeur est situé sous le nom de la variable X du modèle, ici `hp`. Dans l'exemple ci-dessous, qui utilise le jeu de données `mtcars`, le modèle nous indique que lorsque `hp` vaudra 0, l'estimation de `mpg` vaudra 30.09886, et que pour chaque augmentation d'unité de `hp`, on aura une diminution de -0.06823 unité de `mpg`.

```
lm(mpg ~ hp, data = mtcars)
```

```
##
## Call:
## lm(formula = mpg ~ hp, data = mtcars)
##
## Coefficients:
## (Intercept)           hp
##    30.09886     -0.06823
```

Pour plus de confort dans l'écriture de la suite du code, il peut être intéressant d'associer le modèle crée avec la fonction `lm()` à un nom. Pour accéder aux différentes informations statistiques résumant le modèle, on peut alors utiliser la fonction `summary()` avec le nom choisi pour le modèle.

```
model <- lm(mpg ~ hp, data = mtcars)
summary(model)
```

```
##
## Call:
## lm(formula = mpg ~ hp, data = mtcars)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7121 -2.1122 -0.8854  1.5819  8.2360
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.09886    1.63392  18.421  < 2e-16 ***
## hp          -0.06823    0.01012  -6.742 1.79e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.863 on 30 degrees of freedom
## Multiple R-squared:  0.6024, Adjusted R-squared:  0.5892
## F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07
```

Dans la liste d'informations données suite à l'activation du code, on retrouve notamment l'ordonnée à l'origine (*Intercept*) et le coefficient directeur (*hp*), et on peut trouver le coefficient R^2 en face de l'écriture *Multiple R-squared*. On peut aussi y voir l'erreur typique d'estimation en face de l'écriture *Residual standard error*.

5.1.2 L'erreur typique d'estimation

L'erreur typique d'estimation, ou SEE, représente l'écart-type des erreurs d'estimation associées à l'utilisation d'un modèle. Son unité est donc celle de la variable Y que l'on a cherché à prédire avec le modèle. La formule suivante permet d'expliquer son calcul à partir de données prélevées sur un échantillon :

$$SEE = \sqrt{\frac{\sum_{i=1}^{N}(RESi - \overline{RES})^2}{N-2}},$$

où $RESi$ désigne le résidu pour une observation donnée, $\overline{RES}$ la moyenne des résidus, et N le nombre d'observations.

5.1.3 Graphique récapitulatif

Il est possible d'extraire l'ordonnée à l'origine et la pente (i.e., le coefficient directeur) du modèle de régression, le coefficient R^2, et la statistique SEE, à partir de la liste d'informations obtenue avec la fonction `summary()`. Le code ci-dessous montre comment faire cela avec l'exemple concernant le jeu de données `mtcars` :

```
# Extraction de l'ordonnée à l'origine
intercept <- summary(model)$coefficients[1]
intercept
```

```
## [1] 30.09886
```

```
# Extraction du coefficient directeur
slope <-  summary(model)$coefficients[2]
slope
```

```
## [1] -0.06822828
```

```
# Extraction du R2
R2 <- summary(model)$r.squared
R2
```

```
## [1] 0.6024373
```

```
# Extraction de SEE
SEE <- summary(model)$sigma
SEE
```

```
## [1] 3.862962
```

Une fois extraites et associées à des noms, ces informations peuvent ensuite être réutilisées avec le package `ggplot2` et la fonction `annotate()` pour compléter le graphique initial avec des informations statistiques (cf. Figure 5.2).

```
ggplot(data = mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  annotate("text", label = bquote(paste(
    "Y = ", .(round(slope, digits = 3)), "X + ",
     .(round(intercept, digits = 3)), " ; ",
     R^2, " = ", .(round(R2, digits = 3)),
     " ; SEE = ", .(round(SEE, digits = 3)))),
           x = 50, y = 35, hjust = 0, size = 5)
```

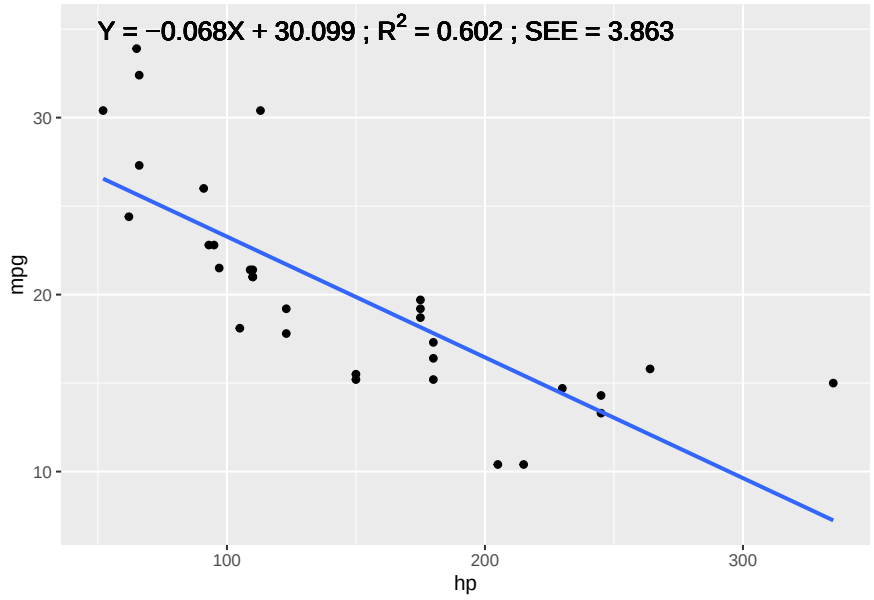

Figure 5.2: Régression linéaire avec les informations correspondantes

La leçon du quartet d'Anscombe

Encore une fois, lorsqu'on étudie un phénomène, ici l'existence d'une relation linéaire, il est important de d'abord faire un graphique montrant les données. Cette première étape graphique est importante car les valeurs numériques qui peuvent être obtenues pour le coefficient R^2 (et donc aussi pour le coefficient de corrélation de Pearson), et la statistique SEE, ne peuvent à elles seules garantir l'aspect linéaire d'une relation. Un exemple qui permet d'illustrer cela est le quartet d'Anscombe (1973). Il s'agit de quatre jeux de données dont les représentations graphiques sont montrées sur la Figure 5.3.

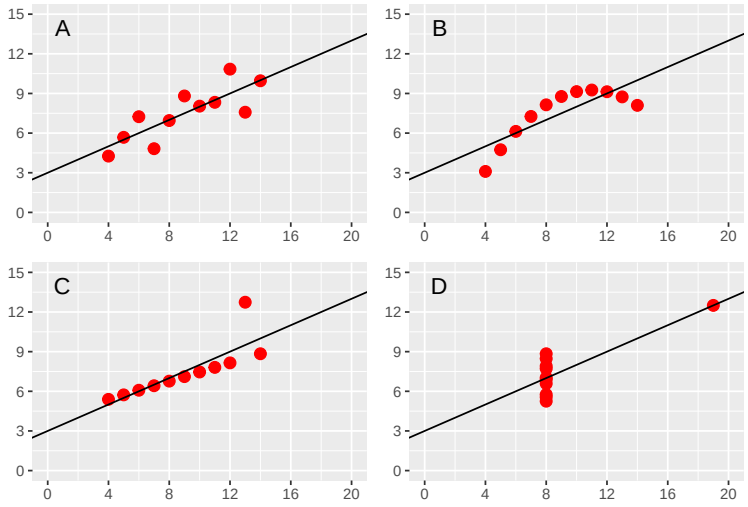

Figure 5.3: Le quartet d'Anscombe

Bien que d'aspects très différents, ces jeux de données montrent pourtant des variables en abscisses qui ont toutes la même moyenne ($\overline{X} = 9$) et le même écart-type ($\hat{\sigma}_X = 3.32$), des variables en ordonnées qui ont elles aussi la même moyenne ($\overline{Y} = 7.5$) et le même écart-type $= (\hat{\sigma}_Y = 2.03)$, et des modèles de régression linéaire qui présentent tous la même équation ($Y = 0.5X + 3$), le même coefficient de détermination ($R^2 = 0.67$) et la même erreur typique d'estimation ($SEE = 1.24$). Pour autant, on observe que seul le premier jeu de données (cf. graphique A de la Figure 5.3) est associé à un modèle linéaire vraiment pertinent. En effet, le graphique B montre bien que la relation n'est pas linéaire mais plutôt quadratique, le graphique C montre que la régression est anormalement influencée par une valeur extrême, et le graphique D montre qu'il n'y a en réalité pas de relation linéaire entre les deux variables et que celle-ci ne semble exister numériquement que grâce à une seule valeur très extrême. Autant le graphique C invite à conserver une analyse de régression linéaire avec éventuellement certains ajustements à réaliser, autant les graphiques B et D indiquent qu'un modèle linéaire n'est pas pertinent en l'état pour caractériser la relation entre les deux variables étudiées.

partie III

Analyses inférentielles

Chapitre 6

Prérequis

6.1 Préambule

Réaliser une inférence statistique consiste à conclure à propos de quelque chose dans une population d'intérêt (cette chose pouvant se traduire par n'importe quel paramètre statistique : une moyenne, une médiane, une proportion, un coefficient de corrélation, une différence de moyennes, un rapport de cotes, etc.), cela à partir de données prélevées dans un échantillon de cette population. Pour comprendre la mécanique des calculs à mettre en oeuvre pour réaliser une inférence statistique, il est nécessaire d'avoir quelques notions en matière de probabilité. Ce chapitre, qui s'inspire largement du chapitre "Introduction to probability" de l'ouvrage de Danielle Navarro (2018), vise à présenter brièvement ces notions.

6.2 Lois de probabilité

6.2.1 Notion de loi de probabilité

Les procédures de calcul pour réaliser une inférence statistique requièrent d'utiliser des lois mathématiques que l'on doit configurer pour déterminer théoriquement les probabilités de rencontrer telle ou telle valeur d'un paramètre statistique donné lorsqu'on étudie un échantillon provenant de la population d'intérêt. Cette démarche implique de comprendre que si un phénomène existe (ou pas) à l'échelle d'une population (chose quantifiable à l'aide de la valeur d'un paramètre statistique donné), l'étude d'un échantillon provenant de cette population ne donnera pas forcément la même valeur pour le paramètre statistique considéré. L'enjeu est alors de pouvoir déterminer la probabilité qui était celle d'obtenir telle ou telle valeur du paramètre statistique considéré avec son échantillon dans le cas où le phénomène étudié existerait (ou pas), cela pour envisager une conclusion quant à la réelle existence ou non du phénomène étudié dans la population d'intérêt.

Ces lois mathématiques utilisées pour faire des inférences statistiques, qu'on appelle aussi des lois de probabilité, donnent directement les valeurs de probabilité d'obtenir telle ou telle valeur du paramètre statistique considéré dans les cas de variables qualitatives. Dans les cas de variables quantitatives, les lois de probabilité ne donnent pas directement les probabilités. En effet, dans ces situations, les lois donnent les **densités de probabilité**. Ceci est lié au fait que dans ce type de situations, les probabilités concernent le fait d'avoir des valeurs appartenant à des intervalles donnés, et non pas d'avoir une valeur précise. Cela implique que pour obtenir la probabilité de rencontrer une valeur appartenant à un intervalle donné, il ne faut pas prendre la valeur directement donnée par la loi de probabilité, mais l'intégrale (i.e., l'aire sous la courbe

de densité de probabilité) correspondant à l'intervalle de valeurs considéré. Ce qui suit vise à décrire et illustrer des lois de probabilité typiquement utilisées dans les procédures de calcul servant à réaliser des inférences statistiques.

6.2.2 La loi binomiale

La loi binomiale est une loi mathématique qui concerne les situations où seulement deux résultats sont possibles, comme par exemple "succès" et "échec", "pile" et "face", "0" et "1", etc. Un exemple classique où l'on est en présence d'une variable binomiale est celui où l'on demande à N personnes de lancer une fois une pièce non truquée, avec par conséquent "pile" et "face" comme seuls résultats possibles et autant de chances de tomber sur "pile" que sur "face" à chaque lancer ($\theta = 0.5$, soit une chance sur deux). Une fois les essais de toutes les personnes terminés, la distribution qui en résulte contient alors la proportion de personnes qui ont obtenu "pile" et la proportion de personnes qui n'ont pas obtenu "pile" (donc "face"). Lorsque le nombre N d'essais ou de participants et la probabilité d'avoir tel ou tel résultat (θ) sont connus, la loi binomiale, qu'on note $X \sim B(\theta, N)$, permet de connaître la probabilité $P(X)$ d'avoir X succès (e.g., X fois "pile") sur les N essais ou participants. La formule de cette loi est montrée ci-dessous :

$$P(X|\theta, N) = \frac{N!}{X!(N-X)!}\theta^X(1-\theta)^{N-X}$$

La Figure 6.1 illustre une loi binomiale (avec $N = 100$ et $\theta = 0.5$) qui pourrait s'appliquer au cas d'un lancer de pièce non truquée. Cette figure montre que même si une pièce est non truquée ($\theta = 0.5$), il est tout à fait possible qu'il y ait plus de personnes à obtenir "pile" plutôt que "face" et inversement sur un échantillon de 100 personnes. Cependant, ce que montre aussi la figure, c'est que si la pièce est non truquée, les chances restent les plus fortes pour l'obtention de 50 % de personnes avec "pile" et avec "face" pour un lancer de pièce.

Plusieurs fonctions R peuvent être utilisées pour obtenir des informations liées aux probabilités données par une loi binomiale. Ces fonctions sont montrées ci-dessous.

```
# Fonction pour déterminer la probabilité d'obtenir précisément x succès
# (ici 50) à partir de N essais/personnes (ici 100) et d'une probabilité
# de succès donnée (ici 0.5)
dbinom(x = 50, size = 100, prob = 0.5)
```

```
## [1] 0.07958924
```

```
# Fonction pour déterminer la probabilité d'obtenir un nombre de succès
# inférieur ou égal à q (ici 50) à partir de N essais/personnes (ici 100)
# et d'une probabilité de succès donnée (ici 0.5)
pbinom(q = 50, size = 100, prob = 0.5)
```

```
## [1] 0.5397946
```

```
# Fonction pour déterminer la valeur pour laquelle il y a une probabilité p
# d'obtenir une valeur inférieure ou égale à la valeur définie à partir de N
# essais/personnes (ici 100) et d'une probabilité de succès donnée (ici 0.5)
qbinom(p = 0.6, size = 100, prob = 0.5)
```

```
## [1] 51
```

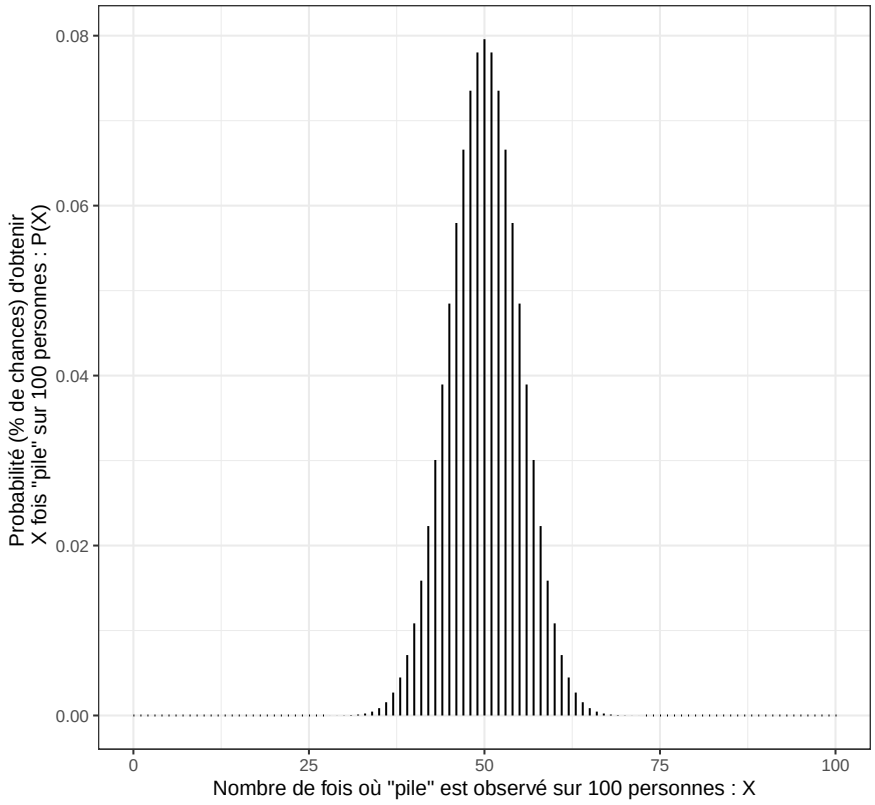

Figure 6.1: Illustration d'une loi binomiale avec la situation de 100 personnes lançant une pièce non truquée

6.2.3 La loi normale et lois apparentées

La **loi normale**, ou encore **loi gaussienne**, qu'on note $X \sim N(\mu, \sigma)$, avec μ la moyenne de la variable, et σ l'écart-type de la variable, est une loi qui permet de déterminer la probabilité de rencontrer une valeur dans un intervalle donné en lien avec la formule montrée ci-dessous :

$$p(X|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(X-\mu)^2}{2\sigma^2}),$$

$p(X|\mu, \sigma)$ étant la densité de probabilité correspondant à la valeur X. Il s'agit bien ici de la densité de probabilité, et non pas de la valeur de la probabilité d'obtenir une valeur précise. Pour mieux comprendre les valeurs que donne cette loi mathématique, regardons la Figure 6.2 (graphique de gauche). Sur cette figure, la courbe noire représente les valeurs données par la fonction définissant la loi normale avec $\mu = 1$, et $\sigma = 1$. Ces valeurs en réalité n'ont pas vraiment d'intérêt en soi. Par contre, elle permettent de délimiter une aire (en rouge) entre elles et l'axe horizontal, la valeur de cette aire étant pour le coup la probabilité d'obtenir une valeur incluse dans l'intervalle de valeurs relatif à l'aire sous la courbe considérée. Ainsi, l'aire sous l'ensemble de la courbe représentant la densité de probabilité est associée à une probabilité de 1 (il y a par définition, lors d'un tirage au sort, 100 % de chances de rencontrer une valeur située entre le minimum et le maximum de la variable modélisée à l'aide de cette loi). Dans la même veine, le graphique de droite de la Figure 6.2 montre une aire sous la courbe (en rouge) dont la valeur est la probabilité d'obtenir une valeur comprise en 2 et 3 lorsqu'on tire au sort une observation en provenance de la population représentée par une loi normale de moyenne 1 et d'écart-type 1, la probabilité étant ici de 14 %.

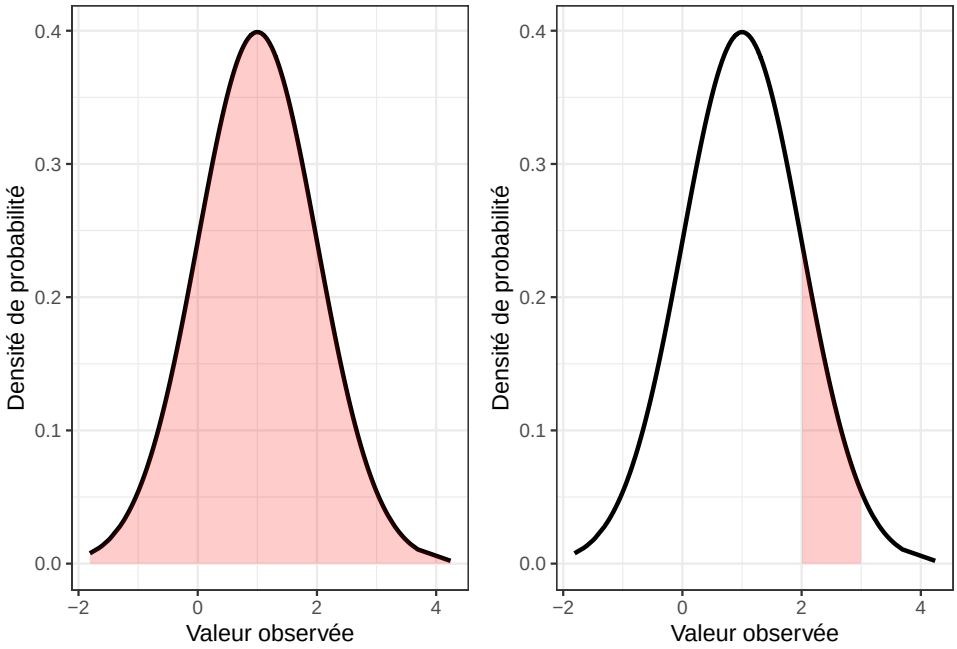

Figure 6.2: Densité de probabilité d'une loi normale

Plusieurs fonctions R peuvent être utilisées pour obtenir des informations liées à la distribution d'une loi normale donnée. Ces fonctions sont montrées ci-dessous.

```r
# Fonction pour déterminer la densité de probabilité correspondant à la valeur x
dnorm(x = 1, mean = 1, sd = 1)
```

```
## [1] 0.3989423
```

```r
# Fonction pour déterminer la probabilité d'obtenir une valeur inférieure ou
# égale à q
pnorm(q = 2, mean = 1, sd = 1)
```

```
## [1] 0.8413447
```

```r
# Fonction pour déterminer la valeur pour laquelle il y a une probabilité p
# d'obtenir une valeur inférieure ou égale à la valeur définie
qnorm(p = 0.7, mean = 1, sd = 1)
```

```
## [1] 1.524401
```

La loi normale peut être mise en lien avec d'autres grandes lois, telles que :

- La loi Chi-carré (χ^2) : lorsque l'on prend les valeurs de plusieurs distributions normales standards (avec des moyennes de 0 et des écarts-types de 1), qu'on les met au carré, puis qu'on les additionne, on obtient une variable suivant une loi χ^2 à k degrés de liberté (cf. Figure 6.3, graphique A), k étant le nombre de variables que l'on a mises au carré. Comme on peut le voir sur la Figure 6.3, la distribution χ^2 est plutôt asymétrique, avec des valeurs toujours supérieures à 0.
- La loi t : les distributions relatives à des lois t ressemblent aux distributions relatives à des lois normales mais avec des queues de distribution plus épaisses (cf. Figure 6.3, graphique B). Une distribution t peut être obtenue en divisant les valeurs d'une distribution χ^2 par le nombre de degrés de liberté k, puis en prenant leurs racines carrées, et enfin en divisant les valeurs d'une loi normale par la variable obtenue. On obtient alors une distribution t à k degrés de liberté (cf. Figure 6.3, graphique C).
- La loi F : la distribution d'une loi F ressemble à celle d'une loi χ^2. Une distribution F sert à comparer deux distributions χ^2.

6.3 Loi des grands nombres, distribution d'échantillonnage de la moyenne, et théorème de la limite centrale

La loi des grands nombres décrit un principe de probabilité permettant de comprendre bon nombre de phénomènes statistiques, notamment lorsque l'on s'intéresse à la moyenne d'un échantillon. Cette loi implique par exemple le fait que la moyenne d'un échantillon pris au hasard dans une population tend à être plus proche de la moyenne de la population à mesure que la taille de l'échantillon étudié est grande. Cette loi est illustrée sur la Figure 6.4. Sur cette figure, les distributions représentent des échantillons crées de manière aléatoire à partir d'une population ayant pour moyenne 0 et pour écart-type 400. Le trait vertical rouge représente la moyenne de la population d'origine alors que le trait en pointillés noirs montre la moyenne de l'échantillon qui a été obtenue. Lorsqu'on regarde chaque colonne de la figure du haut vers le bas et qu'on compare les colonnes entre elles, on se rend compte effectivement que plus la taille N de

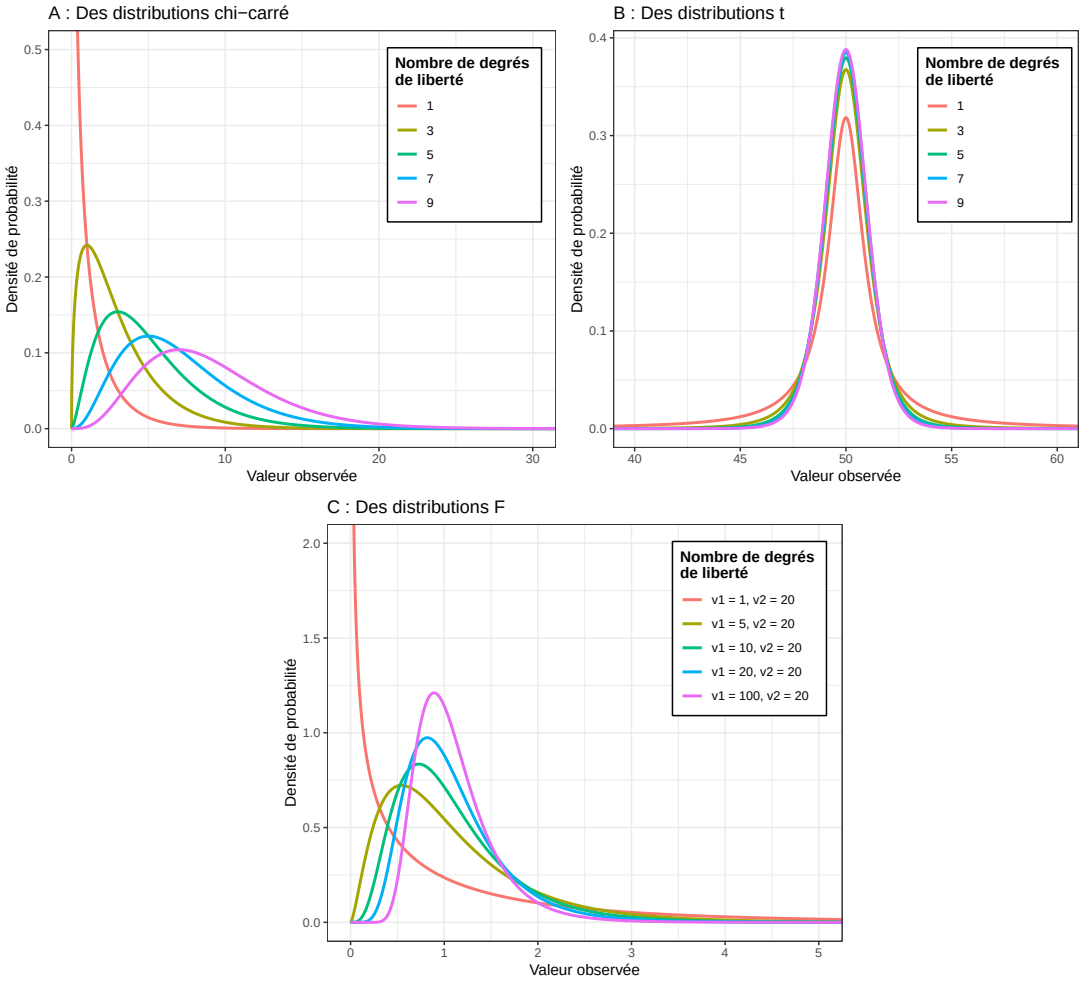

Figure 6.3: Densité de probabilité de lois chi-carré, t, et F

l'échantillon tiré de la population est grande, plus le trait noir se retrouve en général proche du trait rouge, cela traduisant le fait qu'on a de meilleures chances que la moyenne de l'échantillon étudié soit plus proche de la moyenne de la population lorsque l'échantillon est de grande taille. Une question qui pourrait alors se poser est le nombre de personnes ou d'individus que doit contenir l'échantillon pour obtenir un résultat avec une marge d'erreur jugée acceptable.

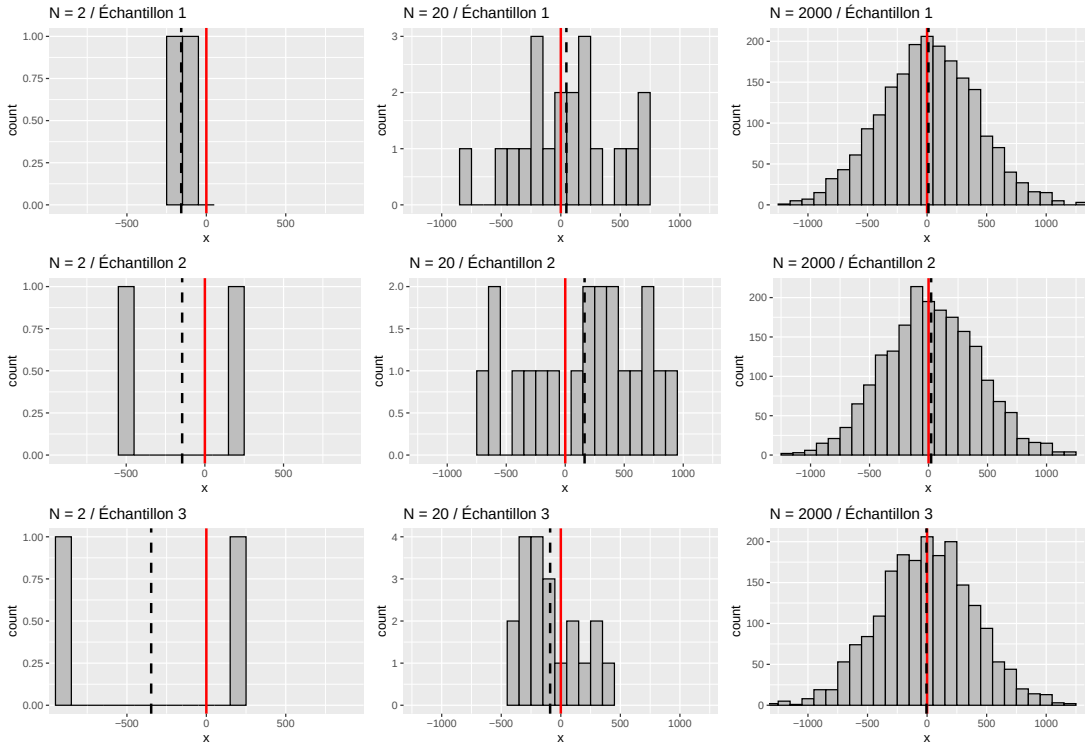

Figure 6.4: Illustration de la loi des grands nombres avec la moyenne d'un échantillon. Les distributions ont été obtenues à partir de N valeurs obtenues aléatoirement à partir d'une population de moyenne 0 et d'écart-type 400. Trait rouge = moyenne de la population d'origine ; trait noir = moyenne de l'échantillon

La Figure 6.4 montre certes qu'on a plus de chances d'avoir une moyenne d'échantillon proche de la moyenne de la population avec un grand N, mais elle esquisse aussi, avec les traits en pointillés noirs, le fait qu'avec un grand N, la variabilité des valeurs que peuvent prendre les moyennes de plusieurs échantillons diminue. Pour s'en assurer, on peut chercher à voir les valeurs de moyennes que l'on obtiendrait si l'on étudiait un grand nombre d'échantillons (e.g., 10 000) de même taille, autrement dit la distribution d'échantillonnage de la moyenne pour une valeur de N donnée. La Figure 6.5 donne une vision de ce que serait une telle distribution pour différentes valeurs de N dans ce cas là.

La Figure 6.5 illustre plusieurs principes qui relèvent du **théorème de la limite centrale**, à savoir : la moyenne d'une distribution d'échantillonnage de la moyenne tend à être la même moyenne que celle de la population d'origine ; et l'écart-type de la distribution d'échantillonnage de la moyenne (*SEM*, pour *Standard Error of the Mean* en anglais), devient plus faible à mesure que N grandit. Un troisième principe est illustré sur la Figure 6.6. Pour réaliser cette figure, la même démarche que pour la Figure 6.5 a été suivie, si ce n'est qu'auparavant, la population d'origine suivait systématiquement une loi normale. Dans le cas de la Figure 6.6, la population suit à l'origine une loi chi-carré, elle est donc asymétrique. Malgré tout, on voit que dès que N est suffisamment grand, la distribution d'échantillonnage de la moyenne suit une loi normale.

Si l'on formalise les choses d'un point de vue plus mathématique, le théorème de la limite

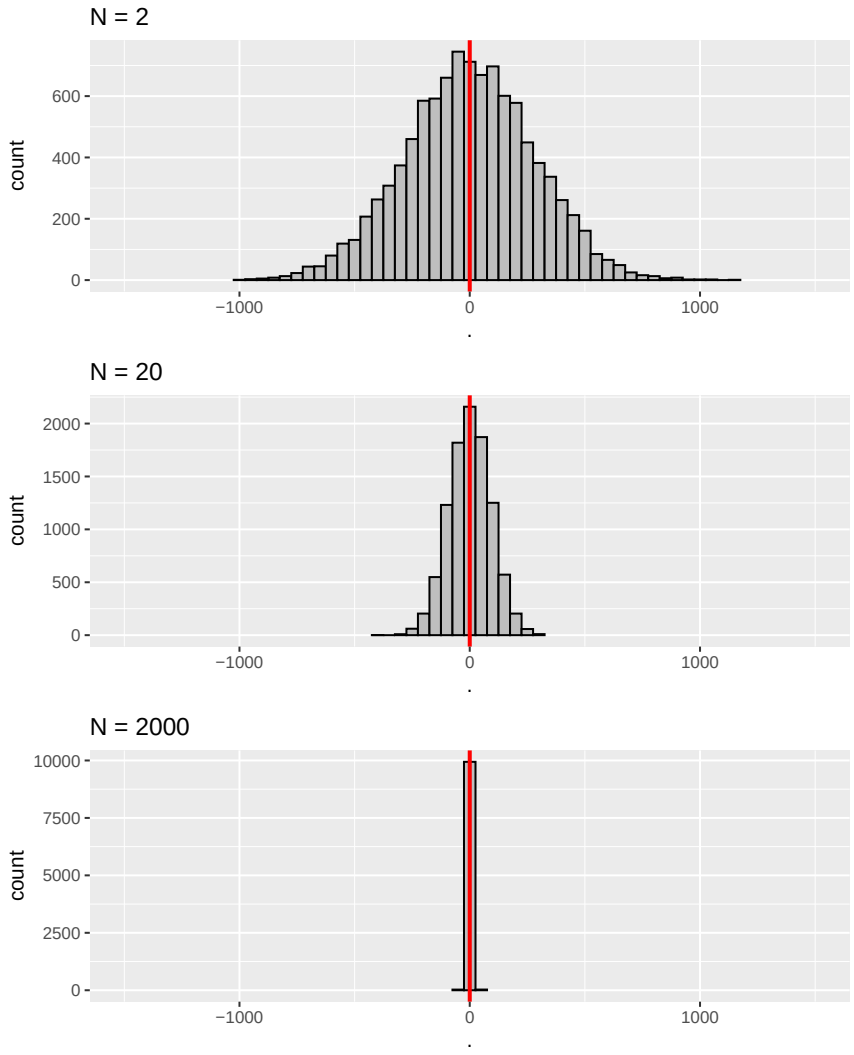

Figure 6.5: Illustration du théorème de la limite centrale appliqué à une moyenne d'échantillon. Les distributions des moyennes montrées ici ont été obtenues avec 10 000 échantillons de N observations obtenues aléatoirement à partir d'une population ayant pour moyenne 0 et écart-type 400. Trait rouge = moyenne de la population d'origine

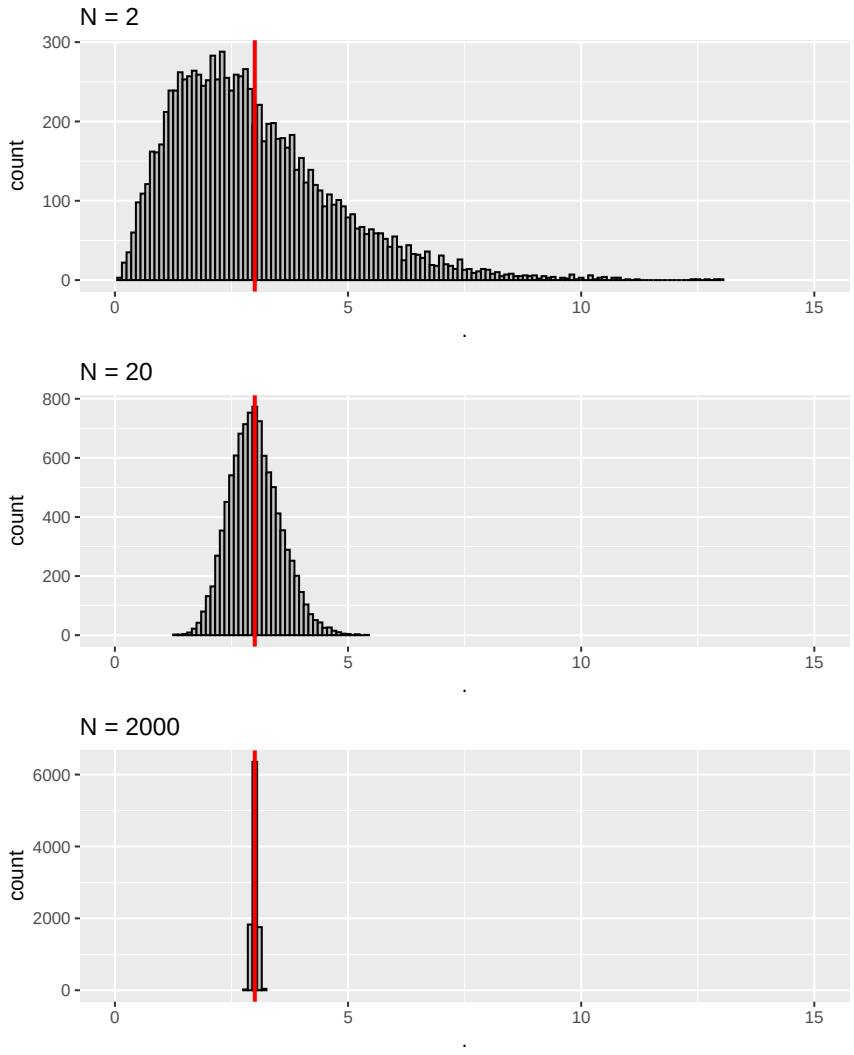

Figure 6.6: Illustration du théorème de la limite centrale appliqué à une moyenne d'échantillon. Les distributions des moyennes montrées ici ont été obtenues avec 10 000 échantillons de N observations obtenues aléatoirement à partir d'une population suivant une loi chi-carré avec 3 degrés de liberté. Trait rouge = moyenne de la population d'origine

centrale nous dit que si une population a une moyenne μ et un écart-type σ, alors la moyenne de la distribution d'échantillonnage a aussi μ comme moyenne, et l'écart-type de la distribution d'échantillonnage de la moyenne (i.e., l'erreur standard, *SEM*) vaut :

$$SEM = \frac{\sigma}{\sqrt{N}}.$$

La formule montre bien que pour une population présentant un écart-type donné, plus les échantillons étudiés seront de petite taille (N), plus grande sera la variabilité des moyennes provenant des différents échantillons. D'un point de vue plus pratique, cela veut dire notamment que dans une méta-analyse où l'on chercherait à estimer par exemple la moyenne de l'effet d'une intervention dans une population donnée, on aura beau avoir plusieurs d'études conduites sur le sujet, si elles sont de trop petites tailles, il y a de bonnes chances pour que les valeurs des effets trouvés divergent substantiellement et ne permettent donc pas d'avoir une vue précise, fiable, de la valeur de l'effet concernant la population globale.

6.4 Résumé

- Les lois de probabilité, la loi des grands nombres, la distribution d'échantillonnage d'une statistique, ou encore le théorème de la limite centrale, sont des outils mathématiques qui permettent de réaliser et de mieux comprendre les inférences statistiques.
- Une loi de probabilité renseigne sur les chances, pour une statistique donnée, d'obtenir une valeur précise (e.g., un nombre de succès dans le cas d'une binomiale) ou d'obtenir une valeur tombant dans un rang de valeurs donné (e.g., s'agissant d'une moyenne dans le cas d'une loi normale).
- Des exemples de lois de probabilité couramment utilisées sont la loi binomiale, la loi normale, la loi χ^2, la loi t, et la loi F.
- Selon la loi des grands nombres, plus la taille d'un échantillon est grande, plus il y a de chances que la moyenne de cet échantillon soit relativement proche de la moyenne de la population d'origine.
- Selon le théorème de la limite centrale, la moyenne de la distribution d'échantillonnage de la moyenne est la même valeur que la moyenne de la population d'origine, et l'écart-type de cette distribution (SEM) est tel que : $SEM = \frac{\sigma}{N}$. Par conséquent, plus N est grand, plus SEM est petite.

Chapitre 7

Tests statistiques pour des variables qualitatives

7.1 Test du χ^2 d'adéquation (ou encore dit de conformité ou d'ajustement)

7.1.1 Calculs sous-jacents

Le test du χ^2 d'adéquation consiste à comparer la distribution d'une variable qualitative observée avec une distribution théorique. Pour mieux voir en quoi cela consiste, prenons un exemple repris de Danielle Navarro (2018) où l'on chercherait à savoir si, lorsqu'on demande à des personnes de choisir "au hasard" mentalement une carte parmi un ensemble de cartes étalées devant soi, le choix se fait vraiment de manière entièrement aléatoire. Pour étudier cela, on demande à 200 personnes de réaliser l'expérimentation et on note le type de carte qui a été retenu : coeur, trèfle, carreau, ou pique. Le jeu de données est crée avec le code ci-dessous :

```
# Creation du jeu de données
trefles <- rep("trèfles", 35)
carreau <- rep("carreau", 51)
coeur <- rep("coeur", 64)
pique <- rep("pique", 50)

cartes <- data.frame(
  id = seq(1, 200, 1),
  choix = c(trefles, carreau, coeur, pique)
  )
```

Dans ce cadre, l'idée que les personnes choisiraient leur carte purement au hasard (idée ou hypothèse qu'on va noter H_0) se traduit par le fait que chaque type de carte aurait la même probabilité d'être tiré à chaque fois, soit une chance sur quatre (0.25). Le vecteur P qui résumerait les probabilités, pour les différents types de carte, d'être choisi à chaque choix d'une personne serait alors le suivant :

$$H_0 : P = (0.25, 0.25, 0.25, 0.25).$$

Étant donné qu'il y a 200 personnes (N) dans notre exemple, la distribution théorique E des

fréquences de tirage de chaque type de carte qui correspondrait à l'hypothèse H_0 serait telle que $E = N \cdot P$, soit :

$$E = (50, 50, 50, 50).$$

On peut aussi mettre cela sous forme de vecteur avec R :

```
expected <- c("carreau" = 50, "coeur" = 50, "pique" = 50, "trefles" = 50)
expected
```

```
## carreau   coeur   pique trefles
##      50      50      50      50
```

Maintenant que l'on connaît la distribution théorique correspondant à notre exemple, voyons qu'elle est la distribution réelle obtenue suite à notre expérimentation. Pour l'obtenir, on peut résumer la variable `choix` du jeu de données `cartes` crée juste précédemment, cela grâce à la fonction `table()` :

```
# Vue numérique de la distribution observée
observed <- table(cartes$choix)
observed
```

```
##
## carreau   coeur   pique trèfles
##      51      64      50      35
```

```
# Vue graphique de la distribution observée
cartes %>%
  group_by(choix) %>%
  count(choix) %>%
  ggplot(aes(x = choix, y = n)) +
  geom_bar(stat = "identity")
```

La distribution O des fréquences observées pourrait s'écrire comme suit :

$$O = (51, 64, 50, 35)$$

À présent, il convient de calculer une statistique qui résumerait l'écart qu'il peut y avoir entre la distribution théorique, et celle observée. Cette statistique, c'est X^2 ou *GOF* (pour *Goodness Of Fit*), tel que :

$$X^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i},$$

k étant le nombre total de modalités de la variable (ici les types de cartes), O_i désignant la fréquence pour la i-ème modalité de la variable observée, et E_i désignant la fréquence pour la i-ème modalité de la variable théorique. On peut calculer X^2 manuellement :

```
sum((observed - expected)^2 / expected)
```

```
## [1] 8.44
```

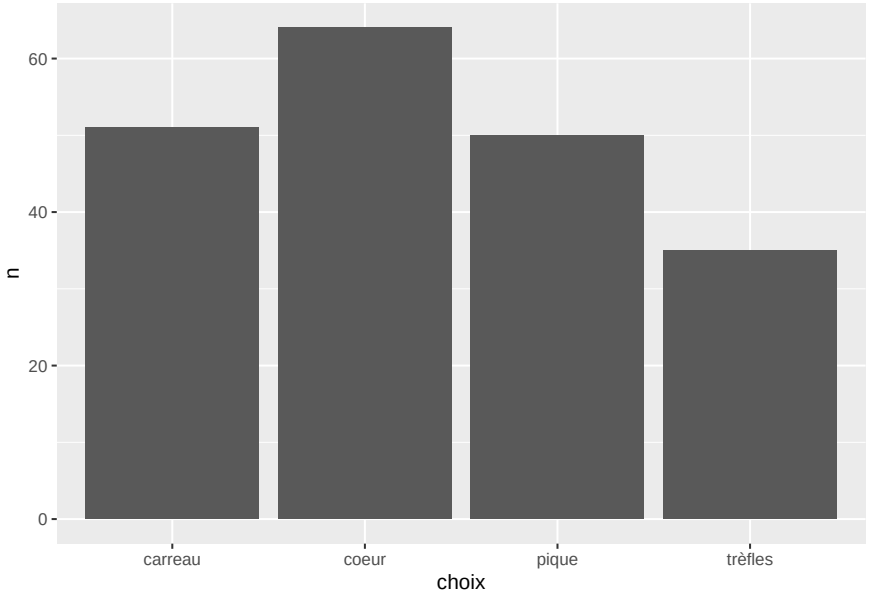

Figure 7.1: Visualisation des fréquences dans le jeu de données 'cartes'

Nous avons à présent le score qui résume l'écart obtenu entre la distribution observée et la distribution théorique de notre variable qualitative. Il nous reste alors à connaître la distribution d'échantillonnage de la statistique X^2 dans le cas où H_0 serait vraie. Cela nous permettra de connaître la probabilité de rencontrer une valeur de X^2 au moins aussi grande que 8.44 dans le cas où H_0 serait vraie, et de juger ainsi si cette probabilité supporte ou non H_0... Trêve de suspens, la distribution d'échantillonnage de X^2 suit une loi χ^2 à $k-1$ degrés de liberté (k étant le nombre de types d'évènement possibles, ici les types de cartes tirées). Comment peut-on expliquer cela? Tout d'abord, reprenons le calcul de X^2, et plus précisément le terme O_i. Admettons qu'il ne concerne qu'un seul type de cartes (e.g., les cœurs). Ce terme a sa propre distribution de probabilité en posant que H_0 soit vraie, et cette distribution est celle d'une loi binomiale (avec $\theta = 0.25$ et $N = 200$ dans notre exemple), puisque pour chaque sujet interrogé, il était possible que ce type de carte soit choisi, ou non. Or, il s'avère que lorsque N est relativement grand et que θ n'est ni trop proche de 0, ni trop proche de 1, la distribution se rapproche d'une loi normale (Navarro, 2018). En allant vite, pour un type de cartes donné, on peut alors considérer que l'expression $\frac{(O_i - E_i)}{E_i}$ a une distribution d'échantillonnage qui s'apparente à une loi normale standard, et c'est le cas à chaque fois que cette expression doit être reprise pour chaque type de carte. Si on met au carré chacune de ces variables, puis qu'on les additionne (comme décrit plus haut pour le calcul de X^2), on se retrouve dans la situation dans laquelle nous obtenons une loi χ^2 (cf. chapitre Prérequis). La subtilité ici, par rapport au chapitre Prérequis, est que le nombre de degrés de liberté n'est pas k mais $k-1$. Le nombre de degrés de liberté désigne le nombre de quantités distinctes à décrire (ici 4 quantités car les données sont regroupées en 4 modalités) moins le nombre de contraintes (ici 1 contrainte liée au groupe de sujets qui a un nombre fini). On a donc bien 3 degrés de liberté ici car il nous suffit de connaître le score de fréquence de 3 modalités pour connaître le score de fréquence de la 4ème modalité qu'il nous manquerait pour une taille d'échantillon donnée.

La Figure 7.2 montre la loi χ^2 relative à 3 degrés de liberté et donc les probabilités de rencontrer tel ou tel intervalle de valeurs de X^2 dans le cas où notre H_0 serait vraie. La zone hachurée représente la probabilité qui était d'obtenir une valeur de X^2 au moins aussi grande que 8.44 dans l'hypothèse où H_0 serait vraie. Cette probabilité P est de 3.8 %. En principe, lors d'un test statistique, il y a aussi ce qu'on appelle une "région critique", c'est-à-dire l'intervalle dans lequel la valeur de la statistique de test, ici X^2, devrait se trouver pour qu'on considère que nos

résultats ne supportent pas H_0 et donc qu'on ne retienne pas cette hypothèse. Classiquement (voire par défaut, à tort), cette région critique est définie à l'aide du seuil $P <= 0.05$. Dans le cas présent, cela se traduirait par le fait que notre statistique X^2 tomberait dans l'intervalle pour lequel il y avait une probabilité inférieure ou égale à 5 % de tomber dans le cas où H_0 serait vraie. Dans notre exemple, cela se traduirait par le fait d'obtenir une valeur de X^2 supérieure ou égale à environ 7.81. La région critique correspondant à cette zone est montrée en rouge sur la Figure 7.2. En suivant cette démarche, dans le cas présent H_0 serait donc bien rejetée puisque notre valeur X^2 de 8.44 tomberait dans la région critique. Le test serait alors dit "significatif".

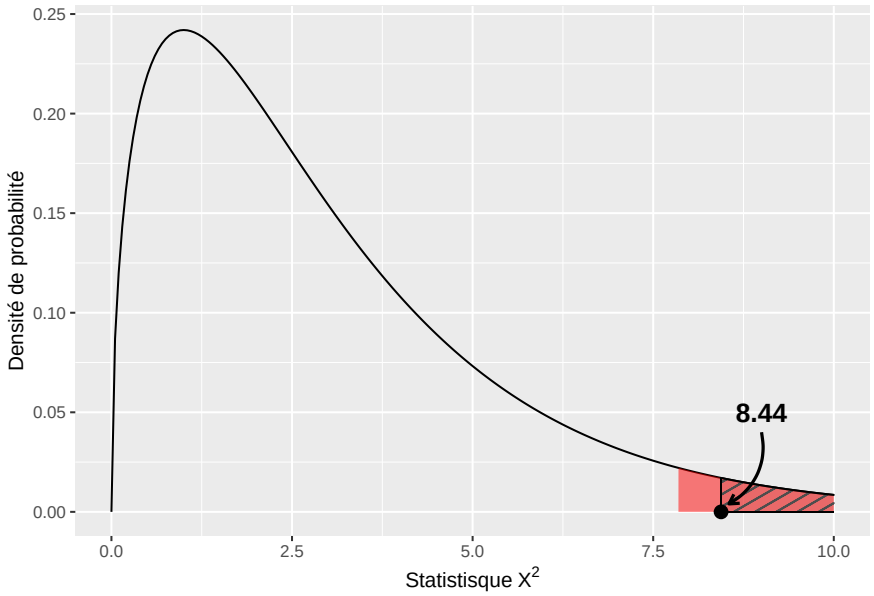

Figure 7.2: Distribution d'échantillonnage de la statistique X^2 sous $H0$

7.1.2 Application avec R

Nous venons de voir la mécanique des calculs qu'il y a derrière le test χ^2 d'adéquation. Heureusement, il existe des fonctions dans R pour faire cela automatiquement, comme la fonction `chisq.test()` :

```
observed <- table(cartes$choix)
expected <- c(0.25, 0.25, 0.25, 0.25)
chisq.test(x = observed, p = expected)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  observed
## X-squared = 8.44, df = 3, p-value = 0.03774
```

Une seconde fonction possible est la fonction `goodnessOfFitTest()` du package `lsr`. Elle est intéressante en raison du fait qu'elle propose un récapitulatif plus détaillé de la configuration du test qui a été faite. De plus, pas besoin de passer par la fonction `table()`, mais il faut que la variable étudiée soit en format `factor` :

```
lsr::goodnessOfFitTest(as.factor(cartes$choix), p = c(0.25, 0.25, 0.25, 0.25))
```

```
##
##      Chi-square test against specified probabilities
##
## Data variable:   as.factor(cartes$choix)
##
## Hypotheses:
##    null:        true probabilities are as specified
##    alternative: true probabilities differ from those specified
##
## Descriptives:
##         observed freq. expected freq. specified prob.
## carreau             51             50            0.25
## coeur               64             50            0.25
## pique               50             50            0.25
## trèfles             35             50            0.25
##
## Test results:
##    X-squared statistic:  8.44
##    degrees of freedom:  3
##    p-value:  0.038
```

Pour reporter les résultats, nous pourrions écrire les choses comme cela (Navarro, 2018) : Le résultat du test était significatif ($\chi^2(3) = 8.44$, $P = 0.04$). Ainsi, on considère que nos données ne supportent pas suffisamment l'hypothèse initiale selon laquelle il y aurait une probabilité identique pour chaque type de carte d'être tiré par une personne. Le choix d'une carte par une personne ne serait donc pas réellement aléatoire.

Il convient de noter que les probabilités espérées (théoriques) peuvent être définies à volonté selon la question de recherche. Par exemple, on aurait pu tester l'hypothèse H_0 selon laquelle les personnes préfèrent à 80 % les cœurs, 10 % les trèfles, 5 % les piques, et 5 % les carreaux. Le tout est de bien configurer le test pour que les fréquences observées pour chaque modalité (ici les types de cartes) soient bien mises en correspondance avec les fréquences théoriques (ces dernières étant configurées à l'aide de l'argument `p` dans l'exemple de code ci-dessous). La fonction `lsr::goodnessOfFitTest()` a donc un fort intérêt ici pour bien vérifier, dans les résultats affichés, qu'on a bien attribué les probabilités théoriques aux bonnes modalités) :

```
lsr::goodnessOfFitTest(as.factor(cartes$choix), p = c(0.05, 0.8, 0.05, 0.1))
```

```
##
##      Chi-square test against specified probabilities
##
## Data variable:   as.factor(cartes$choix)
##
## Hypotheses:
##    null:        true probabilities are as specified
##    alternative: true probabilities differ from those specified
##
## Descriptives:
##         observed freq. expected freq. specified prob.
## carreau             51              10            0.05
## coeur               64             160            0.80
```

```
## pique               50             10            0.05
## trèfles             35             20            0.10
##
## Test results:
##    X-squared statistic:  396.95
##    degrees of freedom:  3
##    p-value:  <.001
```

7.2 Test du χ^2 d'indépendance (ou d'association)

```
library(catdata)
data("encephalitis")
encephalitis
```

```
##    year country count
## 1     1       1     1
## 2     1       2     2
## 3     2       1     0
## 4     2       2     1
## 5     3       1     1
## 6     3       2     2
## 7     4       1     2
## 8     4       2     5
## 9     5       1     2
## 10    5       2     4
## 11    6       1     3
## 12    7       1     8
## 13    8       1     5
## 14    8       2     6
## 15    9       1    13
## 16    9       2     7
## 17   10       1    12
## 18   10       2     7
## 19   11       1     6
## 20   11       2     7
## 21   12       1    13
## 22   12       2     3
## 23   13       1    10
## 24   13       2     4
## 25   14       1    12
## 26   14       2     2
```

(en cours…)

7.3 Résumé

- Le test du χ^2 d'adéquation consiste à comparer une distribution observée à une distribution théorique qui représenterait l'hypothèse à tester.
- La statistique de test du χ^2 d'adéquation est X^2 et suit une distribution χ^2.

- La fonction R de base pour réaliser le test du χ^2 d'adéquation est `chisq.test()`.
- Une fonction R très aidante pour réaliser correctement le test du χ^2 d'adéquation est `lsr::goodnessOfFitTest()`.

Références

Allen, M., Poggiali, D., Whitaker, K., Marshall, T. R., & Kievit, R. A. (2019). Raincloud plots: A multi-platform tool for robust data visualization. *Wellcome Open Res*, *4*, 63. https://doi.org/10.12688/wellcomeopenres.15191.1

Anscombe, F. J. (1973). Graphs in Statistical Analysis. *Am Stat*, *27*(1), 17-21.

Bickel, P. J., Hammel, E. A., & O'Connell J, W. (1975). Sex Bias in Graduate Admissions: Data from Berkeley. *Science*, *187*(4175), 398-404. https://doi.org/10.1126/science.187.4175.398

Chatellier, G., & Durieux, P. (2003). Moyenne, Médiane, et Leurs Indices de Dispersion : Quand Les Utiliser et Comment Les Présenter Dans Un Article Scientifique ? *Rev Mal Respir*, *20*(3), 421-424. https://doi.org/RMR-06-2003-20-3-0761-8425-101019-ART17

Dart, T., & Chatellier, G. (2003). Comment Décrire La Distribution d'une Variable ? *Rev Mal Respir*, *20*(6), 946-951. https://doi.org/RMR-12-2003-20-6-0761-8425-101019-ART19

Gonzales, V. A., & Ottenbacher, K. J. (2001). Measures of central tendency in rehabilitation research: What do they mean? *Am J Phys Med Rehabil*, *80*(2), 141-6. https://doi.org/10.1097/00002060-200102000-00014

Grenier, E. (2007). Quelle Est La « Bonne » Formule de l'écart-Type ? *Revue MODULAD*, *37*, 102-105.

Halperin, S. (1986). Spurious correlations–causes and cures. *Psychoneuroendocrinology*, *11*(1), 3-13. https://doi.org/10.1016/0306-4530(86)90028-4

Hopkins, W. G., Marshall, S. W., Batterham, A. M., & Hanin, J. (2009). Progressive Statistics for Studies in Sports Medicine and Exercise Science. *Med Sci Sports Exerc*, *41*(1), 3-13. https://doi.org/10.1249/MSS.0b013e31818cb278

Jané, M. B., Xiao, Q., Yeung, S. K., Dunleavy, D. J., Röseler, L., Elsherif, M., Cousineau, D., Caldwell, A. R., Johnson, B. T., & Feldman, G. (2023). *Effect Sizes and Confidence Intervals Guide*. https://matthewbjane.com/effect-sizes-and-confidence-intervals-guide/

Joanes, D. N., & Gill, C. A. (1998). Comparing Measures of Sample Skewness and Kurtosis. *The Statistician*, *47*, 183-189. https://doi.org/10.1111/1467-9884.00122

Kelley, K. (2005). The Effects of nonnormal distributions on confidence Intervals around the standardized mean difference: Bootstrap and parametric confidence intervals. *Educ Psychol Meas*, *65*(1), 51-69. https://doi.org/10.1177/0013164404264850

Labreuche, J. (2010). Les Différents Types de Variables, Leurs Représentations Graphiques et Paramètres Descriptifs. *Sang Thrombose Vaisseaux*, *22*(10), 536-543. https://doi.org/10.1684/stv.2010.0541

Lakens, D. (2013). Calculating and Reporting Effect Sizes to Facilitate Cumulative Science: A Practical Primer for t-Tests and ANOVAs. *Front. Psychol.*, *4*. https://doi.org/10.3389/fpsyg.2013.00863

Navarro, D. (2018). *Learning Statistics with R*. UNSW Computational Cognitive Science.

Rousselet, G. A., & Wilcox, R. R. (2020). Reaction Times and Other Skewed Distributions: Problems with the Mean and the Median. *Meta-Psychology*, *4*, 1-39. https://doi.org/10.15626/MP.2019.1630

Weissgerber, T. L., Milic, N. M., Winham, S. J., & Garovic, V. D. (2015). Beyond Bar and Line Graphs: Time for a New Data Presentation Paradigm. *PLoS Biol*, *13*(4), e1002128. https://doi.org/10.1371/journal.pbio.1002128

Wickham, H. (2016). *Ggplot2* (2e éd.). Springer-Verlag.

Wickham, H., & Grolemund, G. (2017). *R for Data Science*. O'Reilly.

Wilke, C. O. (2018). *Fundamentals of Data Visualization*. O'Reilly Media, Inc. Retrieved from https://clauswilke.com/dataviz.