Formation R

Mathieu Ferry

2025-06-03

Table of contents

Préface

Ce petit guide synthétise les éléments d'apprentissage proposés lors de la Formation R du 10 et 11 juin 2025 organisée par la Graduate School de Sciences sociales de l'Université Paris-Saclay.

Les différents tutoriels visent à appréhender la console RStudio, le langage R (et le "tidyverse") à travers quelques fonctions et outils considérés comme les plus pertinents pour qui manipule des données statistiques en sciences sociales.

Support de formation à R, ce guide n'est pas une formation aux statistiques et on suppose que les notions statistiques de base sont connues pour appréhender cette formation. Il n'a pas non plus vocation à se substituer aux nombreux et excellents guides existants, dont on peut ici mentionner quelques uns :

- guide-R
- Introduction à R et au tidyverse
- analyse-R
- utilitR
- R for data science

Part I

À la découverte de R

1 Introduction

1.1 Installer R et RStudio

On commencera d'abord par installer R suivant la configuration de sa machine, à partir des liens disponibles sur le Comprenhensive R Archive Network (CRAN).

On installera ensuite RStudio en se rendant sur cette page. RStudio est un "Integrated Development Environment (IDE)", c'est à dire qu'il s'agit d'un logiciel offrant une interface conviviale pour travailler avec le langage de programmation R. Notons qu'il est possible d'utiliser d'autres logiciels d'interface, mais RStudio est sans doute le plus utilisé actuellement.

Si pour une raison ou une autre, il n'est pas possible d'installer R / RStudio sur sa machine, on peut, temporairement (parce qu'on ne recommande pas de mobiliser un serveur privé pour stocker des données sensibles), utiliser R / RStudio en ligne en se créant un compte sur le "Posit cloud".

Une fois qu'on a installé R et RStudio, on peut directement ouvrir RStudio qui va afficher une jolie console (à l'exception qu'ici, j'ai déjà créé un script R en cliquant sur la petite flèche blanche sur fond vert que j'ai enregistré dans sous le nom 1Intro.R, ce que vous devez faire également).

Exercice

- Ouvrir RStudio et créer un créer un script vide.
- Enregistrer ce script dans un dossier (par exemple "Formation R") en le nommant par exemple 1Intro.R.

1.2 R, RStudio et le tidyverse

On pourra se reporter à la petite présentation de R, RStudio du tidyverse réalisée par Julien Barnier. Rappelons ici que R est un langage de programmation développé depuis les années 1990, dérivé du langage S, et dont on retrouve certaines proximités avec le langage C. Il est libre et gratuit.

Parmi les multiples forces de R, on notera notamment sa communauté d'utilisateur · ices qui peuvent aussi jouer le rôle de développeur · euses. R est ainsi doté de nombreuses extensions sous la forme de "fonctions", stockées dans des "packages" (ou bibliothèques), qui facilitent grandement son utilisation. La liste des packages disponibles sur le CRAN (mais on peut aussi mobiliser des packages qui ne sont pas sur le CRAN) est disponible ici : https://cran.r-project.org/web/packages/.

Avec plus de 22 000 packages, on s'y perd assez vite. On propose tout au long de ce guide ici d'utiliser quelques packages considérés comme facilitant la vie pour réaliser des statistiques pour les sciences sociales. Là encore, il ne s'agit pas d'être exhaustif, ni d'imposer quoi que ce soit. La particularité de R est qu'il est souvent possible de faire la même chose en mobilisant des opérations / fonctions / packages différents. À chacun de trouver ses petits "trucs" !

Le package tidyverse comprend en fait une suite de packages conçus pour fonctionner ensemble et qui facilitent la manipulation, le recodage et la production de graphiques par rapport au language "base R". On va ici s'appuyer dessus pour une partie de nos manipulations.

> Exercice
>
> Installer le package tidyverse, puis le charger dans la session R à l'aide des deux lignes de code suivantes. On peut les copier coller dans son script puis les faire tourner (voir la section suivante si on ne sait pas comment faire tourner des lignes de code).

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

> **i** Note
>
> On a besoin d'installer les packages qu'une seule fois, en revanche, il faut charger son package utilisé à chaque nouvelle ouverture d'une session R. Ainsi, si vous refermez RStudio puis le réouvrez, pas besoin de refaire tourner install.packages("tidyverse"), mais pour utiliser les fonctions de cette extension, il faut quand même appeler tidyverse en faisant tourner library(tidyverse).
> L'appel des packages pertinents se fait généralement systématiquement au début du script.

1.3 RStudio comme outil de travail

De base, la console se présente sous la forme de quatre panneaux :

- Le premier en haut à gauche correspond au fichier du script R, qui pour l'instant est vide (ou presque vide, si vous avez déjà copié-collé les lignes de code pour installer et

ouvrir tidyverse). C'est ici que nous écrirons nos lignes de commande que nous ferons tourner, soit en cliquant sur Run, soit en utilisant le raccourci clavier Cmd + Entrée (ou Contrôle + Entrée).

- Le second en bas à gauche correspond à la console, pour l'instant, elle nous informe juste qu'elle a chargé R dans la version la plus à jour trouvée sur mon ordinateur (4.4.3). Dans la console apparaitront les commandes que nous avons fait tourné et les messages renvoyés par R à cette occasion.
- Le troisième en haut à droite correspond à plusieurs onglets. Le plus important correspond à l'Environment, qui nous indique tous les objets créés dans la session dans laquelle nous nous trouvons (des dataframes, des listes, des vecteurs….).
- Le quatrième en bas à droite comprend également plusieurs onglet, dont Files - qui permet de naviguer dans l'arborescence de son ordinateur -, Plots - où s'afficheront nos magnifiques graphiques, Packages - qui indique les packages (les librairies) installées sur notre ordinateur et celles qui sont chargées dans la session actuelle (elles sont cochées), Help - où s'affiche les informations sur une fonction quand on tape dans la console ?nom_de_la_function, Viewer - où apparaitront nos jolis tableaux.

Exercice

Vérifier que le package tidyverse est bien installé et chargé dans la session R.

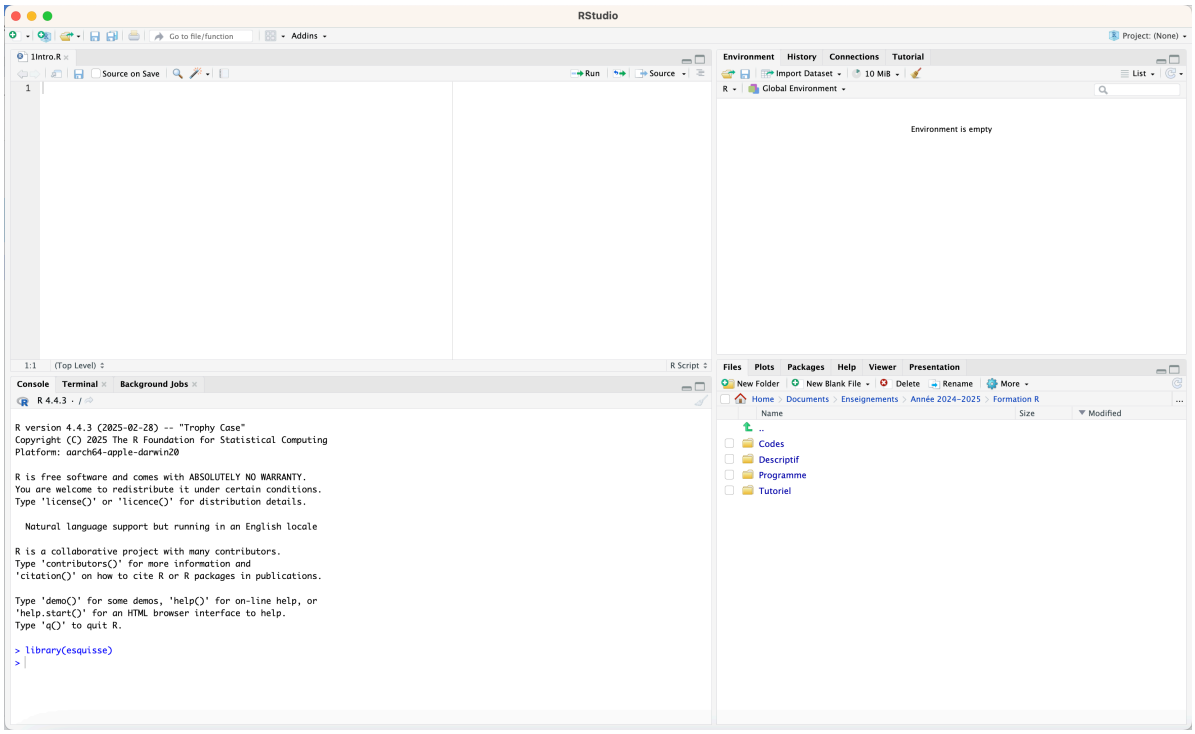


Figure 1.1: La console de RStudio

Caution

- Au moment de quitter RStudio, il est possible qu'une fenêtre apparaisse demandant si on veut "Save the workspace image…". Et en fait ce n'est pas une bonne idée de cliquer sur Save !
- Car cela enregistre l'ensemble de l'espace de travail (la session) qui sera réouverte automatiquement en réouvrant RStudio, et on peut se retrouver avec des objets non pertinents pour une session ultérieure (et qui peuvent se retrouver en conflit les uns avec les autres).
- On poura suivre Julien Barnier qui propose de désactiver l'apparition de cette fenêtre pop-up : https://juba.github.io/tidyverse/05-organiser.html#d%C3%A9sactiver-la-sauvegarde-de-lespace-de-travail.
- Si on n'a pas cliqué sur Save, une réouverture de RStudio réinitialise la session R. Pour être sur que c'est bien le cas (vider le global environment, les packages chargés, réinitialise la mémoire…), on pourra écrire dans la console :

```
.rs.restartR()
```

1.4 Charger des données dans R

Pour faire des stats, il nous faut des données. Commençons par les charger dans notre espace de travail. Pour ce faire, trois solutions :

- On peut dans le panneau en bas à droite naviguer dans l'arborescence des dossiers pour trouver le fichier adéquat, cliquer dessus et une fenêtre apparait, permettant de formater le fichier avant de le charger dans l'espace de travail.
- On peut cliquer sur Import Dataset dans l'onglet en haut à droite, puis cliquer sur Browse et naviguer dans le Finder ou l'Explorateur.
- On peut directement avoir recours à des lignes de code mises dans son script pour charger son fichier.

La deuxième solution est préférable, dans un souci de reproductibilité de son code. Mais la première ou la seconde ont l'avantage d'être plus facile quand on n'est pas familier du code, d'autant que ces solutions "clic-souris" ont l'avantage de proposer les lignes de code correspondantes qu'on peut recopier dans son script pour la prochaine session de travail !

Ici, nous allons mobiliser une base de données appelée "Salaires.csv" qui porte sur les niveaux de salaire de plusieurs centaines d'Américains en 1991 (dont la provenance n'est pas bien établie, mais ce n'est pas grave, ce sont juste des données pour l'exemple).

Exercice

Charger la base de données "Salaires.csv", en navigant dans l'arborescence. Avant de cliquer sur importer, veiller à vérifier dans le "Data Preview" que les données sont lues correctement, i.e. modifier le Delimiter si nécessaire (ici, des points-virgules). Copier aussi le code permettant de charger les données. Après avoir cliqué sur "Import", coller le code dans le script "1Intro.R".

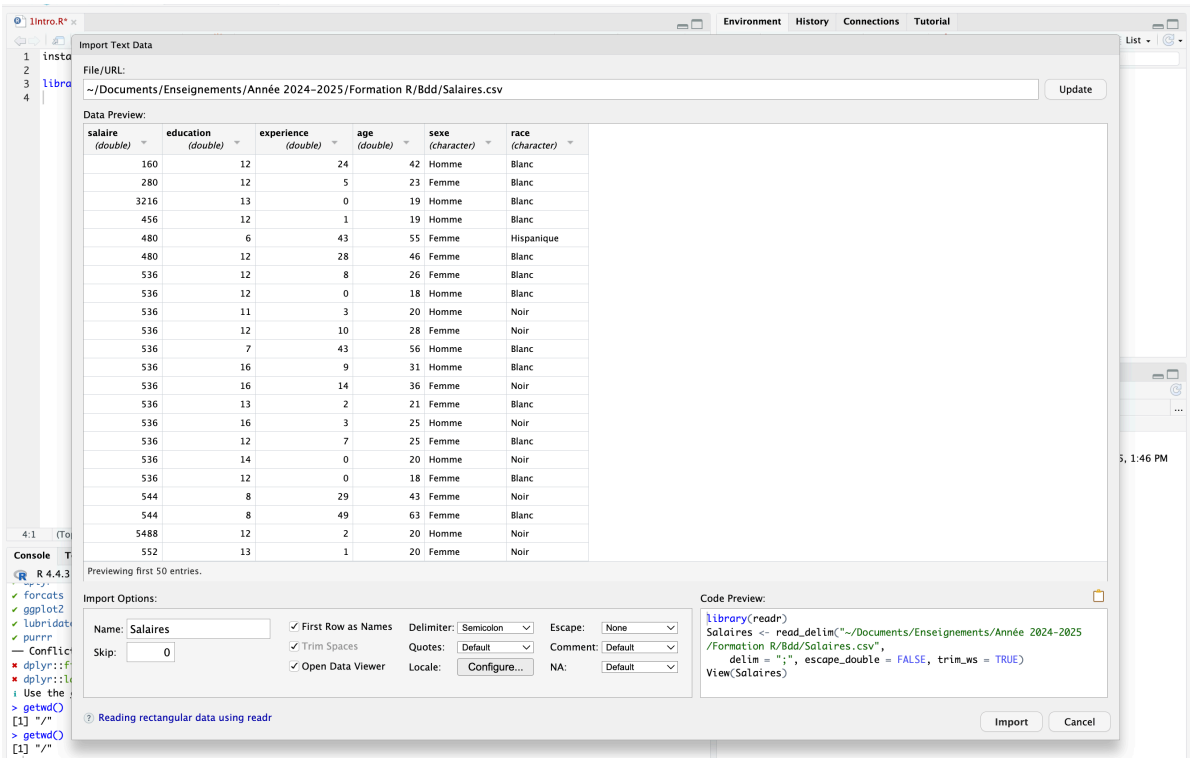


Figure 1.2: Chargement des données avec la solution “clic-bouton”

Un mot sur les trois lignes de code que nous avons copié :

```
library(readr)
Salaires <- read_delim("~/Documents/Enseignements/Année 2024-2025/Formation R/Bdd/Salaires.cs
                       delim = ";", escape_double = FALSE, trim_ws = TRUE)
View(Salaires)
```

- La première (library(readr)) n’est pas totalement utile car elle charge le package readr qui permet d’utiliser la fonction read_delim. Or, le package readr s’est en fait déjà chargé quand nous avons chargé l’extension tidyverse.
- La troisième View(Salaires) peut également être enlevée, c’est une fonction qui permet de pré-visualiser la base de données dans un nouvel onglet (la garder peut être embêtant si on fait tourner une série de lignes de codes d’un coup).

La deuxième ligne de code appelle la fonction read_delim, avec l’argument du chemin complet où se trouve l’objet (ce chemin est a priori différent sur votre machine…), un argument spécifiant la manière dont les colonnes du fichier csv sont délimitées (par des points-virgules), l’argument escape_double=FALSE porte sur la manière de lire les colonnes (dans le cas où il

y aurait eu des doubles " ", mais ça n'a aucune incidence ici), et l'argument trim_ws=TRUE enlève automatiquement les espaces vides dans les colonnes avant et après chaque chaine de caractères.

Le résultat de la fonction read_delim est sotcké (grâce à l'opérateur "<-") dans l'objet Salaires. Ici, on a appelé la base dans R du même nom que celui du fichier csv, mais rien n'empêche de l'appeler par un autre nom, bdd, ou bidule, ou us, etc.

Si tout s'est bien passé, la base apparait maintenant dans l'environnement (onglet en haut à droite).

Noter qu'on aurait pu écrire cette ligne en deux temps :

```
setwd("~/Documents/Enseignements/Année 2024-2025/Formation R/Bdd")
Salaires <- read_delim("Salaires.csv",
                       delim = ";", escape_double = FALSE, trim_ws = TRUE)
```

Dans cette version, on indique d'abord à R quelle est le chemin où se trouve le fichier à lire avec la fonction setwd() qui signifie "set the working directory", et dans un deuxième temps, on charge la base de données. Cette solution est avantageuse si on veut charger plusieurs bases de données en même temps dans son environnement qui sont stockées dans le même dossier local. Car oui, il est possible de charger plusieurs bases de données en même temps !

Si on choisit de charger sa base de données directement par les lignes de code, on peut récupérer le chemin dans son Finder ou son Explorateur de fichiers en cliquant-droit sur le fichier csv, puis Propriétés ou Lire les informations.

Noter que pour vérifier vers quel dossier R pointe actuellement, on peut utiliser la fonction getwd().

1.5 Décrire sa base de données

Formellement, comme nous utilisons l'extension tidyverse, notre base de données est maintenant un "tibble" (un format un peu plus intelligent, sinon par défaut dans R, les base de données sont des data.frame). Lors du chargement, la console nous dit que la base contient 534 lignes, 6 colonnes, de différents types (character pour le sexe et la race, dbl ou double pour education, experience, age, et num ou number pour salaire). Il n'y a pas vraiment de différence qui nous préoccupe entre double et number (ce sont des variables quantitatives).

```
> Salaires <- read_delim("Salaires.csv",
+                        delim = ";", escape_double = FALSE, trim_ws = TRUE)
Rows: 534 Columns: 6
── Column specification ────────────────────────────────────────────────────────
Delimiter: ";"
chr (2): sexe, race
dbl (3): education, experience, age
num (1): salaire

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Figure 1.3: Sortie de la console R lors du chargement de la base de données

Plusieurs manières de mieux décrire sa base de données peuvent être utiles :

- On peut écrire le nom de la base de données Salaires dans la console, ce qui va afficher les 10 premières lignes de la base de données
- On peut aussi écrire str(Salaires) pour appréhender la structure du jeu de données
- On peut aussi écrire summary(Salaires) pour avoir un résumé des variables quantitatives du jeu de données

1.6 Quelques fonctions de manipulation de ses données

1.6.1 Accéder à une variable

Le jeu de données est donc composé de 6 variables. Pour accéder à une variable, par exemple la variable salaire, on peut taper :

```
Salaires$salaire
```

Le signe $ indique qu'on vient s'intéresser à la variable salaire contenue dans la base Salaires. Cette commande nous sort un vecteur (vector) des valeurs de la variable.

Dans le tidyverse, on appelle les objets grâce à des pipes (des tuyaux), qui prennent deux opérateurs différents (et largement équivalents) : %>% et |>. On cherchera à privilégier le second.

Par exemple, pour calculer la moyenne des salaires on écrira :

```
Salaires |> summarise(mean(salaire))
```

summarise est la fonction qui permet de résumer la base de données suivant la fonction indiquée. Notons qu'on aurait pu écrire :

```
Salaires |> summarise(Moyenne=mean(salaire))
```

Pour travailler sur le logarithme des salaires plutôt que sur les salaires, on pourra créer une nouvelle variable dans la base Salaires comme ceci :

```
Salaires <- Salaires |> mutate(salaire_log=log(salaire))
```

mutate est la fonction qui permet de transformer les variables de sa base de données. Attention à bien assigner les changements de sa base de données dans un objet (ici, le même à savoir la base de données Salaires).

1.6.2 Sélectionner des colonnes

Pour créer une base de données nommée Sal2 où on ne sélectionne que les variables salaire, sexe et âge, on écrira :

```
Sal2 <- Salaires |> select(salaire,sexe,age)
```

select permet de sélectionner des colonnes.

On peut aussi choisir d'enlever des colonnes, par exemple, si on veut toutes les variables de Salaires sauf age et sexe :

```
Sal3 <- Salaires |> select(-c(age,sexe))
```

1.6.3 Sélectionner des lignes

Pour créer une base de données à partir de Salaires nommée Salh où on ne travaille que sur les hommes qui ont un salaire supérieur à la moyenne des salaires dans l'échantillon étudié, on écrira :

```
Salh <- Salaires |> filter(sexe=="Homme" & salaire > mean(salaire))
```

filter permet de sélectionner les lignes de la base de données suivant une certaine condition. L'opérateur de l'esperluette & signifie "et". Si on veut utiliser l'opérateur logique ou, on écrit : |.

Si on veut sélectionner les individus de race Hispanique et Noir :

```
Salmin<- Salaires |> filter(race %in% c("Hispanique","Noir"))
```

1.6.4 Renommer une variable

Si on veut renommer dans Salaires la variable sexe et l'appeler genre :

```
Salaires <- Salaires |> rename(genre=sexe)
```

Dans la fonction rename, on place le nouveau nom, le signe égal, puis l'ancien nom de la variable.

1.7 Sauvegarder ses données

Pour sauvegarder ses données, plusieurs solutions.

1.7.1 Les formats natifs R

- On peut sauver ses données en format .RData. C'est le plus simple, pas de prise de tête et l'avantage c'est qu'on est sur de conserver le format des variables, les labels (on en reparle plus tard), etc.

```
save(Salaires,file="Salaires.RData")
```

On n'oubliera pas avant d'utiliser cette fonction d'utiliser setwd("chemin du dossier où on enregistre son fichier") pour indiquer où enregistrer son fichier (où à tout le moins de vérifier que R pointe bien vers le dossier que l'on souhaite grâce à la commande getwd()).

À noter que le format RData permet tout à fait d'enregistrer plusieurs objets R dans un même fichier :

```
save(Salaires,Salmin,file="Sal.RData")
```

Pour réouvrir un fichier .RData, il suffit d'écrire :

```
load("Sal.RData")
```

Le défaut du format .RData est que comme il y a potentiellement plusieurs objets dans le fichier, qu'on ne sait pas comment ils s'appellent et qu'on ne peut pas directement assigner un nom de fichier par défaut à l'ouverture dans la console R, et bien cette méthode peut parfois être risquée (imaginons qu'on ait un fichier Sal.RData qui contienne Salaires et Salmin, mais que dans notre environnement nous ayons déjà un fichier Salmin, en ouvrant Sal.RData, on va écraser le fichier existant).

Pour cette raison, des bases de données peuvent être enregistrées individuellement en utilisant le format .rds avec des fonctions spécifiques :

```
saveRDS(Salaires, "Salaires.rds")
```

En ouvrant un fichier .rds, on contrôle explicitement le nom qu'on va lui attribuer dans l'environnement R :

```
Bidule <- readRDS("Salaires.rds")
```

1.7.2 Le format csv

- Toutefois, on a parfois besoin d'exporter ses données en format csv par exemple. On utilisera alors des fonctions du package readr, write_csv (pour utiliser des séparateurs de colonne avec virgule et décimales avec points) ou write_csv2 (séparateur colonne avec point-virgule avec décimales à virgule).

Le format csv avec séparateur virgules est le standard international des csv anglais, qui est formaté facilement quand son Excel est configuré en anglais :

```
write_csv(Salaires, "Salaires.csv")
```

Le format csv avec séparateur points-virgules est le format csv "français", et permet directement d'ouvrir de manière adéquate son excel dans son Excel configuré en français :

```
write_csv2(Salaires, "Salaires.csv")
```

1.7.3 Les autres formats

- Si on souhaite enregistrer son fichier directement comme un fichier excel, c'est également possible. Au moins deux packages permettent de le faire, writexl et openxlsx. Si on installe openxlsx (install.packages("openxlsx")) :

```r
library(openxlsx)
write.xlsx(Salaires, "Salaires.xlsx")
```

Si on souhaite lire un fichier excel, on peut utiliser read.xlsx du même package, ici l'argument sheet=NULL indique qu'on lit tous les onglets du fichier excel s'il y en a plusieurs :

```r
read.xlsx("Salaires.xlsx", sheet = NULL)
```

- Parfois, on travaille avec des collègues qui travaillent sur d'autres logiciels propriétaires payants (quelle idée !). En ce cas, on aura recours au package haven, qui permet de sauvegarder des données en format SAS, SPSS ou Stata :

```r
library(haven)
write_dta(Salaires, "Salaires.dta")
write_sas(Salaires, "Salaires.sas7bdat")
```

Attention, en format SAS, se pose la question de la conversion des étiquettes de valeurs qui n'est pas aussi simple que pour d'autres logiciels.

1.8 Exercice

Exercice

1. Calculer la moyenne du nombre d'années d'expérience.
2. Calculer la moyenne de l'âge.
3. Calculer la moyenne du nombre d'années d'expérience pour les individus qui ont un âge supérieur à l'âge moyen et les individus qui ont un âge inférieur à l'âge moyen.
4. Enregistrer en format rds une base construite à partir de Salaires avec les conditions suivantes :
   - Elle contient seulement les colonnes race et salaire
   - Elle contient les individus qui ont réalisé plus de 10 années d'études
   - La colonne race est renommée "race_ethnicity"

Part II

Manipuler et analyser ses données

2 Statistiques univariées et recodages des variables

Dans ce chapitre, nous allons nous pencher sur la description univariée des variables, aussi bien des variables numériques (quantitatives) que des variables catégorielles (qualitatives) et sur leur recodage.

Nous allons avoir besoin de trois packages qui n'ont pas encore été chargés (et que vous n'avez peut-être pas encore installé).

Dans le code ci-dessous, voici une proposition de code pour vérifier l'installation des packages nécessaires à ce chapitre et les charger dans l'environnement.

Nous utilisons les mêmes données que dans l'introduction. Notons que nous utilisons ici des données non pondérées. Pour utiliser des données pondérées, on pourra se reporter au chapitre dédié (Approfondissements).

Exercice

- Créer un script vide.
- Enregistrer ce script dans un dossier (par exemple "Formation R") en le nommant par exemple 2Statunis.R.
- Charger la base de données des salaires et ajouter les commandes ci-dessous pour installer / charger les packages.

```
# On liste les packages dont on a besoin dans un vecteur nommé load.lib.
load.lib <- c("tidyverse","questionr","esquisse","kableExtra")

install.lib <- load.lib[!load.lib %in% installed.packages()] # On regarde les paquets qui n

for (lib in install.lib) install.packages(lib,dependencies=TRUE) # On installe ceux-ci

sapply(load.lib,require,character=TRUE) # Et on charge tous les paquets nécessaires
```

2.1 Décrire une variable quantitative

2.1.1 Résumer les variables par quelques statistiques

Dans R, les fonctions de base permettent de sortir les nombres de Tukey pour une variable quantitative :

- Minimum
- Premier quartile (Q1)
- Médiane (Q2)
- Troisième quartile (Q3)
- Maximum

Pour obtenir ces statistiques descriptives on pourra écrire :

```
summary(Salaires$salaire)
```

ou :

```
fivenum(Salaires$salaire)
```

Quel serait l'équivalent dans le tidyverse, ce qui nous serait bien utile, notamment si nous souhaitons filtrer les lignes avant d'obtenir ces statistiques descriptives ?

Une solution serait de recourir aux fonctions permettant de générer chacune des statistiques, que nous connaissons déjà pour la moyenne :

```
Salaires |>
  summarise(
    min    = min(salaire, na.rm = TRUE),
    q1     = quantile(salaire, 0.25, na.rm = TRUE),
    median = median(salaire, na.rm = TRUE),
    q3     = quantile(salaire, 0.75, na.rm = TRUE),
    max    = max(salaire, na.rm = TRUE)
  )
```

On peut aussi utiliser la fonction fivenum, à ce moment là au lieu d'utiliser summarise, il faut utiliser reframe (car la sortie de fivenum renvoie plusieurs nombres) :

```r
Salaires |>
  reframe(
    stat = c("min", "Q1", "médiane", "Q3", "max"),
    salaire = fivenum(salaire)
  )
```

Avec la même logique, on peut alors sortir ces statistiques en filtrant sur les hommes et en s'intéressant à la fois à la variable du salaire et de l'expérience par exemple :

```r
Salaires |>
  filter(genre=="Homme") |> #ou sexe=="
  reframe(
    stat = c("min", "Q1", "médiane", "Q3", "max"),
    salaire = fivenum(salaire),
    experience = fivenum(experience)
  )
```

On a fait une si jolie sortie qu'on souhaiterait la sortir sous forme d'un tableau, mais voilà le copier-coller de la console n'est pas très joli…

Pas de panique, il existe plein de packages dans R pour faire des sorties de jolis tableaux. On se propose ici d'utiliser kableExtra :

```r
sumH<-Salaires |>
  filter(genre=="Homme") |> #ou sexe=="
  reframe(
    stat = c("min", "Q1", "médiane", "Q3", "max"),
    salaire = fivenum(salaire),
    experience = fivenum(experience)
  )

sumH |> kbl() |> kable_classic(full_width = F)
```

La fonction kbl transforme l'objet sumH en un tableau de format latex/html qui est affiché dans le "Viewer" (en bas à droite) et la fonction kable_classic est une des fonctions de formatage par défaut que j'ai choisi. On peut customiser à l'infini ce type de tableau (y compris en ajoutant titre, légende…), le principal intérêt étant surtout de pouvoir le copier-coller de manière propre dans son rapport/article/thèse.

stat	salaire	experience
min	160	0
Q1	1000	8
médiane	1600	14
Q3	8896	23
max	42064	55

Figure 2.1: Tableau créé avec kableExtra et visible dans le Viewer

2.1.2 Résumer avec les quantiles

On peut aussi vouloir résumer sa variable avec des quantiles, qui sont des seuils qui permettent de séparer une distribution en parties égales. Par exemple, si on souhaite obtenir les déciles d'une distribution :

```
quantile(Salaires$salaire, probs = seq(0.1, 0.9, by = 0.1))
```

ou :

```
Salaires |>
  reframe(
    stat = paste0("D", 1:9),
    salaire = quantile(salaire, probs = seq(0.1, 0.9, by = 0.1))
  )
```

Exercice

1. Résumer la distribution de l'âge avec des quartiles.
2. Réaliser un tableau dans le Viewer de cette sortie.

2.1.3 Résumer une variable quantitative avec un graphique

Tout cela est bien beau et semble nous suggérer que le salaire est une variable très asymétrique avec une queue de distribution très étalée sur la droite. Peut-on visualiser la distribution ?

Oui, on peut faire un petit histogramme de base, en écrivant :

```r
hist(Salaires$salaire)
```

Si on veut faire un graphique plus joli, on utilisera le package ggplot2 (chargé avec l'extension tidyverse). Comme nous nous familiarisons avec R, on peut utiliser le package esquisse qui propose une solution clic-bouton pour réaliser son graphique. Il faut exécuter la ligne suivante :

```r
esquisser(Salaires)
```

Cela devrait charger une interface avec en haut les différentes variables du jeu de données qu'il va falloir glisser dans les cases correspondantes.

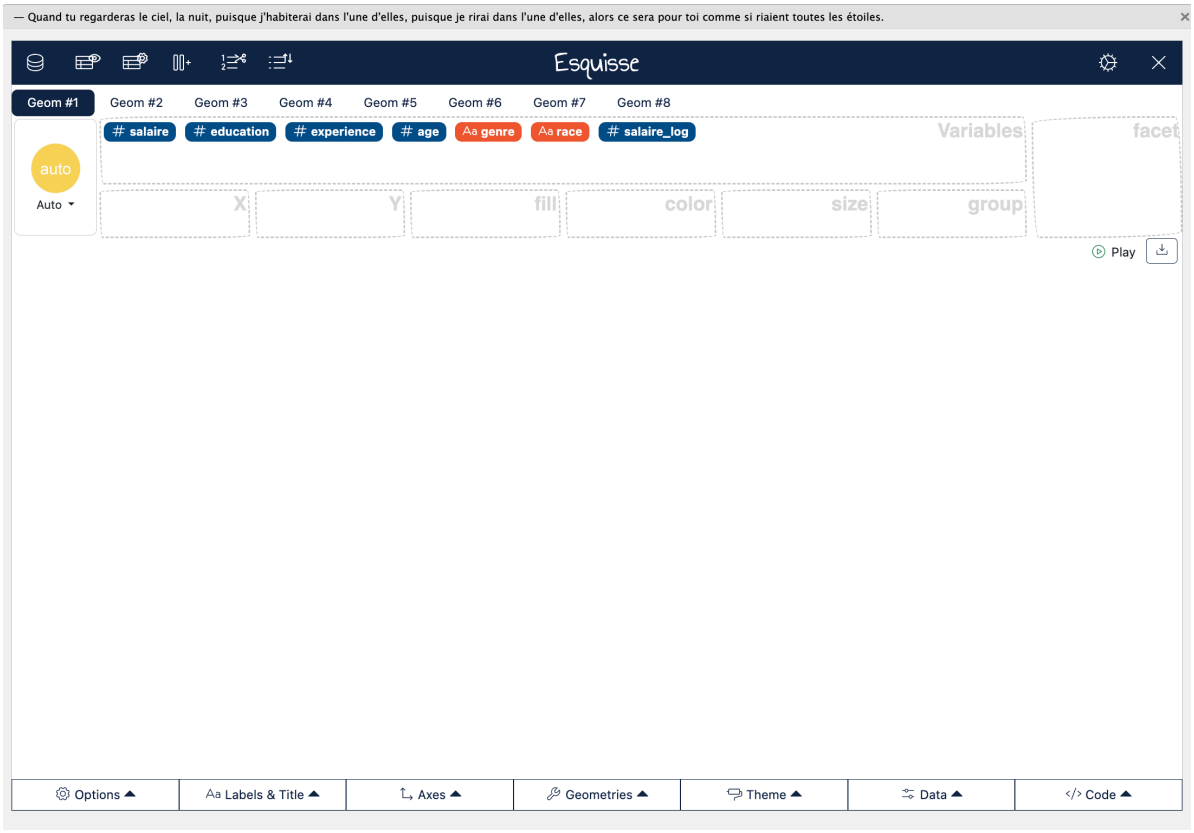


Figure 2.2: Interface d'esquisse

En faisant glisser la variable salaire dans la case X, un histogramme se crée automatiquement. On peut modifier cette représentation et par exemple préférer une distribution avec la fonction de densité. L'ensemble du graphique est "customizable" avec les options dessous.

Enfin, on peut enregistrer son graphique mais aussi copier-coller le code permettant de créer le même graphique sans avoir à réouvrir Esquisse !

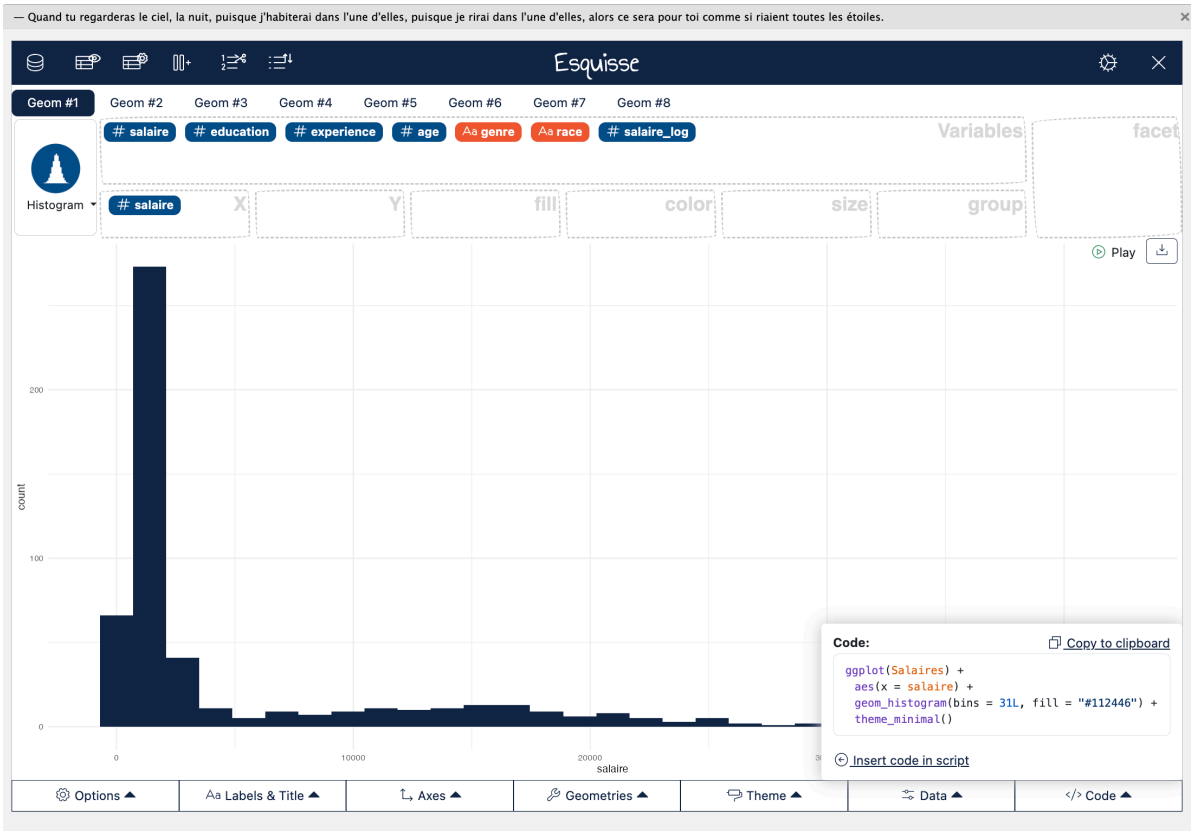


Figure 2.3: Création d'un histogramme avec Esquisse

Quelques mots sur le code permettant de faire cet histogramme :

```
#|eval: false
ggplot(Salaires) +
  aes(x = salaire) +
  geom_histogram(bins = 31L, fill = "#112446") +
  theme_minimal()
```

- ggplot dit à R qu'on crée un graphique sur les données de la base salaire,
- les fonctions au sein du ggplot sont séparées par des +,
- la fonction aes définit les variables mises en jeu dans le graphique (ici le salaire en x),

- la fonction “geom_…” définit la représentation graphique souhaitée avec les paramètres automatiquement choisi (la couleur et le nombre de bins : plus il est élevé, plus les barres sont fines),
- et theme_minimal assigne un “theme” de fond pour le graphique.

Pour une présentation un peu plus avancée de ggplot et de son fonctionnement, on pourra s’appuyer sur une présentation et les ressources proposées par Julien Barnier.

2.2 Décrire une variable qualitative

Décrire une variable quantitative c’est bien beau, mais nous avons souvent pas mal de variables qualitatives dans nos jeux de données. Comment la décrire ?

La fonction freq du package permet de sortir le nombre d’observations, la proportion dans l’échantillon et la proportion cumulée :

```
#|eval: false
freq(Salaires$sexe)
```

Il est aussi possible d’utiliser cette fonction dans le tidyverse en filtrant sur un sous-échantillon de sa base :

```
#|eval: false

Salaires |> filter(race=="Blanc") |> freqtable(genre) |> freq()
```

> Exercice
>
> Avec Esquisse, on peut aussi réaliser un diagramme à barres non empilées d’une variable qualitative. Reproduire le graphique ci-dessous :

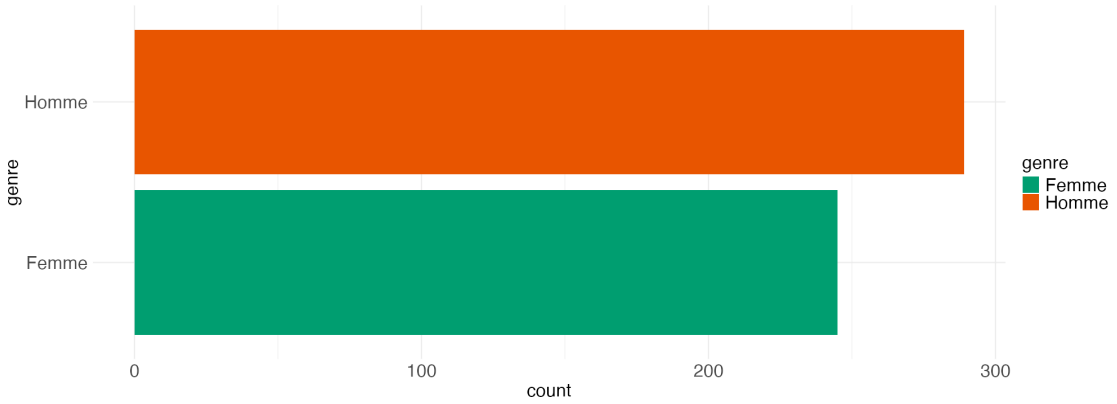


Figure 2.4: Distribution de la variable sexe

Si on veut ajouter le nombre d'individus par modalités sur le graphique on pourra ajouter cette ligne de code :

```
ggplot(Salaires) +
  aes(x = genre, fill = genre) +
  geom_bar() +
  geom_text(stat = "count", aes(label = after_stat(count)), hjust = 1.1, size = 6) +  #Ligne
  scale_fill_brewer(palette = "Dark2", direction = 1) +
  coord_flip() +
  theme_minimal() +
  theme(
    axis.title.y = element_text(size = 18L),
    axis.title.x = element_text(size = 18L),
    axis.text.y = element_text(size = 18L),
    axis.text.x = element_text(size = 18L),
    legend.text = element_text(size = 18L),
    legend.title = element_text(size = 18L)
  )
```

2.3 Recoder une variable quanti en une variable quali

Pour recoder une variable quantitative en une variable qualitative, rien de plus simple. Dans R, nous pouvons utiliser un add-ins de Julien Barnier dans son package questionr, icut. Pour le lancer, on peut cliquer sur "Addins" dans RStudio et cliquer sur "Numeric range dividing". Ou alors, on peut simplement écrire dans la console :

```
icut()
```

Cela devrait ouvrir la console. On peut alors choisir la base au sein de laquelle on veut recoder une variable, choisir la variable à recoder et indiquer le nom de la variable recodée (par défaut, ancien nom de la variable où est ajouté "_rec"). Cet add-ins s'appuie sur la fonction cut() de R.

Différentes options de recodage sont proposées, soit de manière manuelle (on a repéré des valeurs saillantes, ou on connait des valeurs saillantes d'une variable, par exemple un niveau de pauvreté…), en utilisant un algorithme de reclassification (Jenks que les géographes aiment bien, ou d'autres algorithmes), ou avec les quantiles, ou avec classes d'intervalles égaux dans la distribution (equal width).

Il ne faut pas confondre les deux derniers :

- les quantiles sont des seuils qui permettent de séparer l'échantillon en n parties égales
- les classes d'intervalles égaux coupent la distribution (ici les valeurs du salaire) en tranches égales. Si la distribution n'est pas unifome, classes de quantiles et classes d'intervalles seront très différents (et c'est le cas ici pour le salaire qui a une distribution très asymétrique).

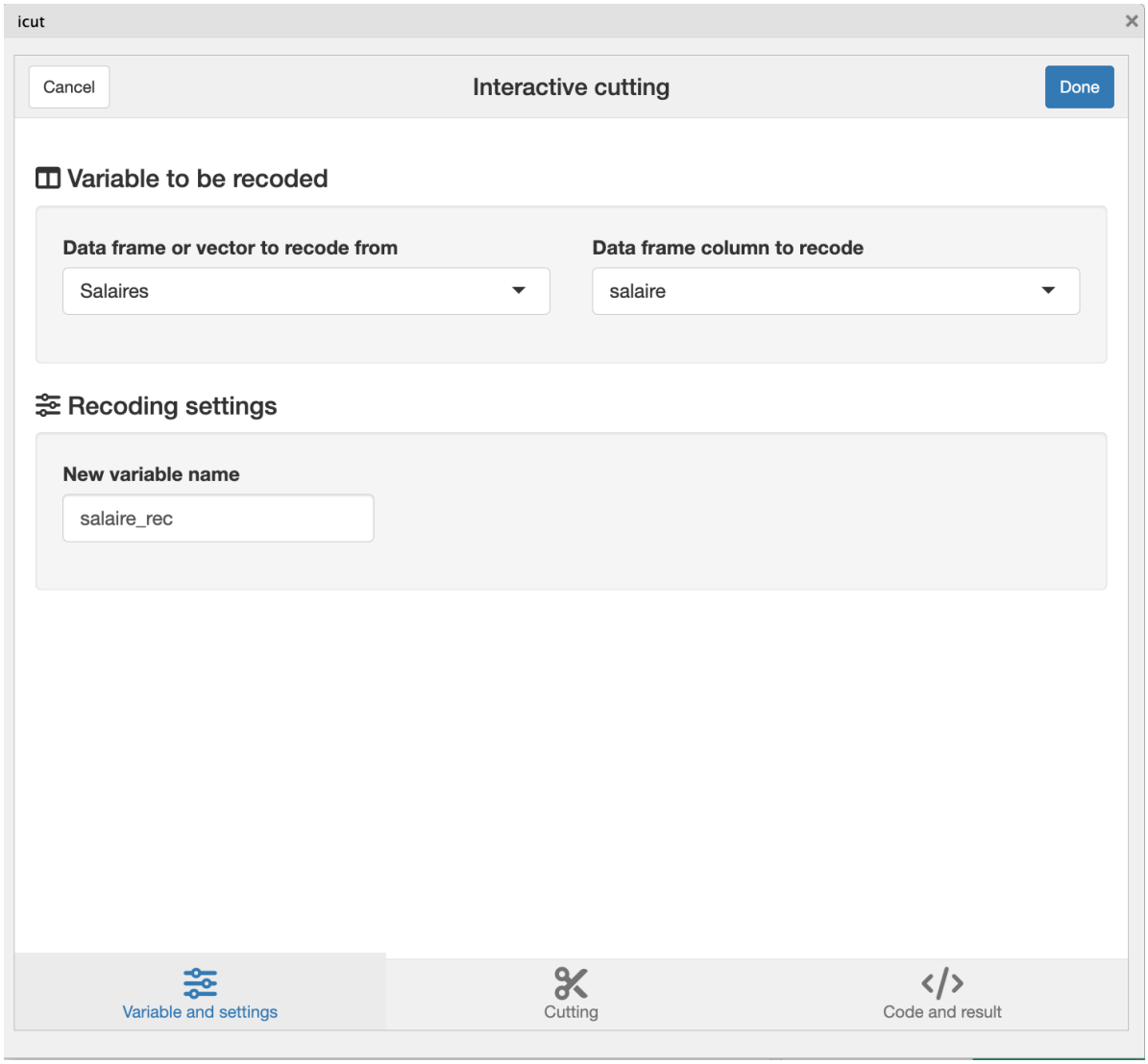


Figure 2.5: Interface de icut

Dans la pratique, j'aime bien les quantiles, qui sont des seuils qui permettent de séparer une distribution ordonnée en parties égales. On crée alors des classes ou des tranches ou des groupes de quantiles, par exemple :

- Avec des quartiles
- Avec des quintiles
- Avec des déciles
- Avec des centiles

Probablement par abus de langage, on a parfois tendance à dire simplement qu'on a recodé notre variable en quantile (par exemple en déciles) et qu'on a créé des déciles (et non des classes de déciles). C'est pratique, mais il ne faut pas oublier que les quantiles sont avant tout des seuils dans une distribution !

Ici, recodons la variable salaires en salaires_rec en utilisant les quintiles. Il faut choisir :

- Cutting method : Quantile
- Breaks number : 5
- Breaks : rien à modifier, les quantiles ont été calculés automatiquement
- Right-closed intervals : par défaut, la fonction cut propose des intervalles fermés à gauche et ouverts à droite. Suivant les cas (et c'est le cas ici pour bien équilibrer les groupes), on peut choisir des intervalles fermés à droite, donc cocher l'option.
- Include extreme : on veut s'assurer qu'aucune valeur extrême n'est omise dans la catégorisation, donc on coche.
- De même avec Append extreme values if necessary
- Label digits : le nombre de décimales retenues pour définir les seuils (notre variable salaire n'a pas de décimale donc ce n'est pas important ici).

Le troisième onglet propose le code, un tableau de distribution de la variable recodée et un diagramme à barres pour visualiser la distribution des différentes modalités (c'est en regardant ce graphique que j'ai décidé de fermer les intervalles à droite pour légèrement rééquilibrer les modalités qui doivent l'être vu qu'on a choisi des quintiles).

On peut alors cliquer sur Done. Attention, la variable n'a pas été créée mais dans la console, le code permettant de la créer est proposé, il suffit de le copier-coller dans son code et de le faire tourner (on peut aussi le modifier directement soi-même si on veut) :

```
## Cutting Salaires$salaire into Salaires$salaire_rec
Salaires$salaire_rec <- cut(Salaires$salaire,
                        include.lowest = TRUE,
                        right = TRUE,
                        dig.lab = 4,
                        breaks = c(160, 800, 1200, 1920, 11008, 42064)
)
```

Exercice

À l'aide des fonctions vues précédemment, proposer un code permettant de vérifier la distribution de la variable salaire_rec. Vérifier en particulier qu'il n'y a pas eu de NA dans la création de la variable et que les classes de quintiles sont à peu près égaux (on ne

s'attend pas forcément à avoir des classes parfaitement égales si plusieurs individus ont le même salaire au niveau des valeurs de seuils).

2.4 Recoder les modalités d'une variables quali

L'onglet icut ne permet pas d'ajouter des labels aux classes créées.

On peut modifier la fonction cut pour indiquer des noms grâce à l'argument "labels" :

```
Salaires$salaire_quint <- cut(Salaires$salaire,
                          include.lowest = TRUE,
                          right = TRUE,
                          dig.lab = 4,
                          breaks = c(160, 800, 1200, 1920, 11008, 42064),
                          labels=c("Très faible","Faible","Moyen","Elevé","Très élevé")
)
```

Une autre solution est d'utiliser un autre add-ins du package de questionr, "Levels recoding" (disponible dans l'onglet Addins) qu'on peut appeler grâce à la fonction :

```
irec()
```

On peut choisir de recoder ses variables en format *character* ou en format *factor*. Dans la pratique, dès lors qu'on a une variable avec une liste fermée de catégories, on travaille avec des *factor* (notamment parce qu'on peut ordonner les catégories). On privilégiera le format *character* pour des variables de type textuel, des phrases, des paragraphes, etc (dans la pratique, plusieurs fonctions de R transforment automatiquement une variable *character* en *factor* avant un traitement statistique).

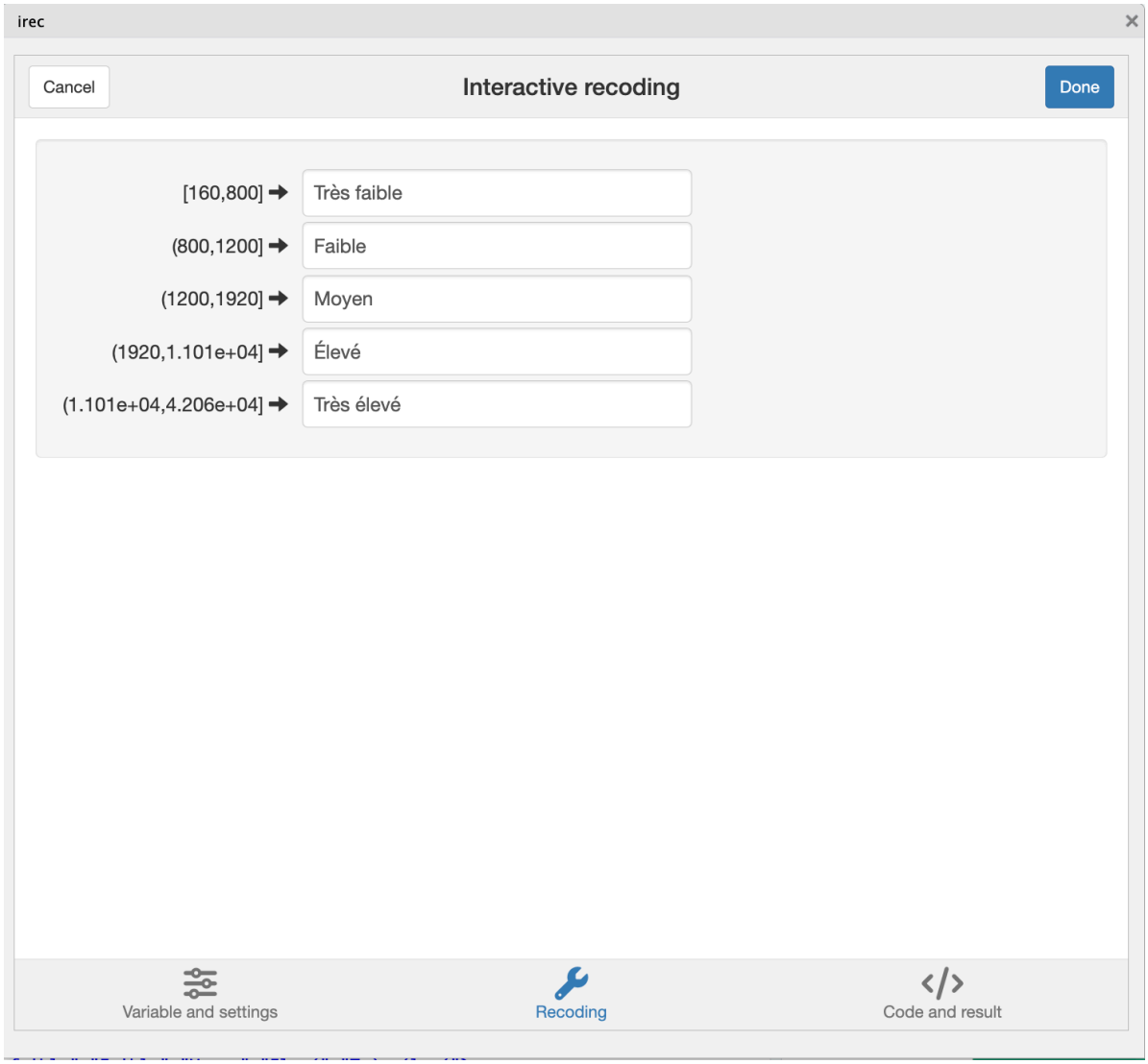


Figure 2.6: Interface de recodage d'irec

Là encore, le troisième onglet permet de récupérer le code R correspondant et de vérifier qu'aucune modalité de la variable recodée n'est oubliée (elle serait transformée en NA). À noter qu'il faut explicitement copier le code dans le script et le faire tourner pour que les changements s'appliquent à la variable.

Exercice

1. Recoder la variable race en une variable “maj” qui distinguent la catégorie raciale “majoritaire” (les Blancs) et les catégories “minoritaires” (les Hispaniques et les Noirs).
2. Réaliser une sortie dans le Viewer d’un tableau de la distribution de cette nouvelle variable.

2.5 Modifier l’ordre des modalités

Un dernier add-ins bien utile est celui permettant le réordonnancement des catégories d’une variable de type *factor*.

Par défaut, les modalités sont souvent ordonnées par ordre alphabétique.

Dans la pratique, c’est rarement l’ordre que nous souhaitons privilégier !

L’onglet “Levels ordering” permet alors de réordonner ses catégories.

```
iorder()
```

Exercice

Réordonner les catégories de la variable sexe.

2.6 Exercices

On propose de répondre aux questions suivantes à l’aide des outils que nous avons vu dans ce chapitre :

Exercice

1. La médiane des salaires des individus blancs est-elle plus ou moins élevée que celle des individus noirs et hispaniques ?
2. Quelle est la répartition des femmes et des hommes dans l’échantillon ? Parmi elles et eux, quelle proportion appartient au décile supérieur des salaires (les 10 % les plus élevés) ? Ces différences suggèrent-elles un effet du genre sur la probabilité d’accéder aux salaires les plus élevés ?
3. Créer une variable jeune qui vaut “oui” si l’individu a moins de 30 ans (et “non”

sinon). Quelle est la proportion des jeunes dans l'échantillon ?

4. Quelle est la répartition des jeunes chez les femmes ? Chez les hommes ?
5. Les jeunes ont-ils un niveau plus ou moins élevé d'expérience que le reste de la population ?

3 Statistiques bivariées et graphiques

Dans ce chapitre, nous abordons maintenant les statistiques bivariées. Nous utilisons la même base et les mêmes packages que nous avons déjà installés dans le chapitre sur les statistiques univariées, avec en plus un package GGally qui permet de réaliser des diagrammes de nuages de points sur plusieurs variables (à voir plus bas).

Exercice

- Créer un script vide.
- Enregistrer ce script dans un dossier (par exemple "Formation R") en le nommant par exemple 3Statbis.R.
- Charger la base de données des salaires et ajouter les commandes ci-dessous pour installer / charger les packages.

```
# On liste les packages dont on a besoin dans un vecteur nommé load.lib.
load.lib <- c("tidyverse","questionr","esquisse","kableExtra","GGally")
install.lib <- load.lib[!load.lib %in% installed.packages()] # On regarde les paquets qui n
for (lib in install.lib) install.packages(lib,dependencies=TRUE) # On installe ceux-ci
sapply(load.lib,require,character=TRUE) # Et on charge tous les paquets
```

3.1 L'association entre deux variables quantitatives et le nuage de points

3.1.1 Le nuage de points et la corrélation de Pearson

Pour étudier la relation entre deux variables quantitatives, rien ne vaut le nuage de points (le scatterplot en anglais) que nous pouvons réaliser grâce au package esquisse :

```
esquisser(Salaires)
```

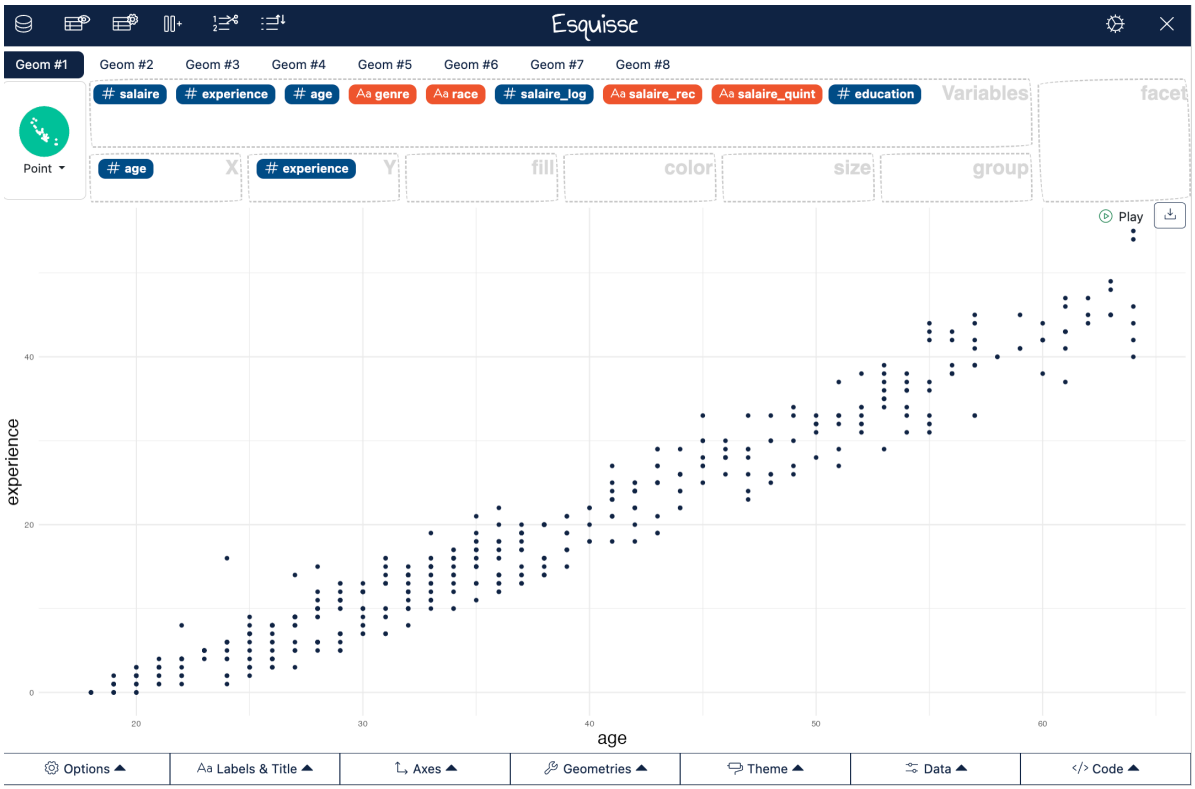


Figure 3.1: Nuage de points entre l'âge et l'expérience dans esquisse

Le nuage de points entre l'âge et l'expérience suggère une relation positive (les points sont aligné sur une droite ascendante) et élevée (les points sont quasi-parfaitement alignés).

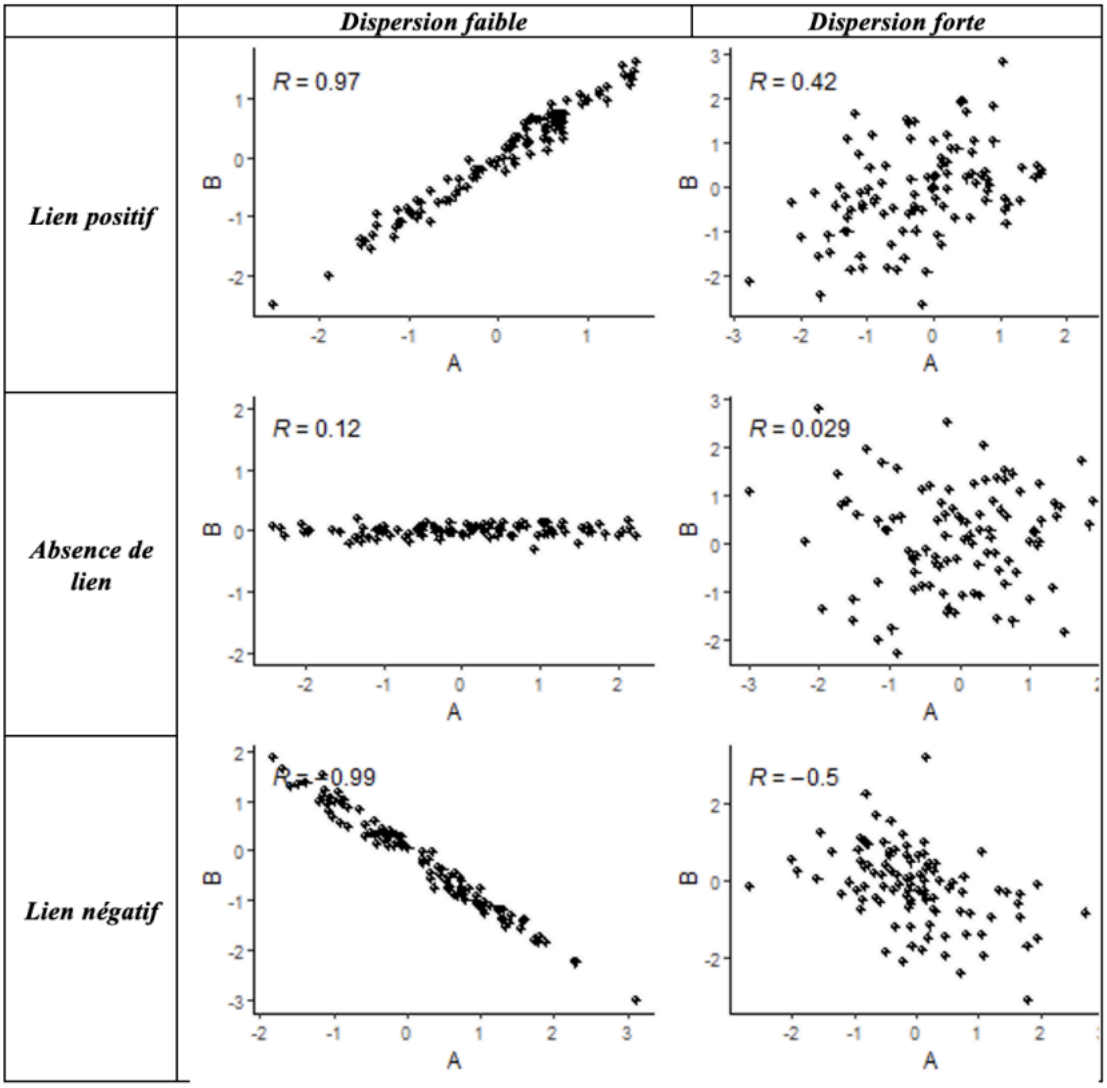
Pour avoir un indicateur de la relation entre les deux variables, on peut calculer l'indice de corrélation.

```
cor(Salaires$age,Salaires$experience)
```

La corrélation est ici de 0,98, ce qui est proche de 1, ce qui suggère une forte corrélation positive entre les deux variables. Rien d'étonnant, l'expérience vient avec l'age.

Le graphique ci-dessous présente des simulations de nuages de points et les indices de corrélations associés en distinguant le lien positif ou négatif et la dispersion plus ou moins forte des données autour d'une droite indiquant une relation linéaire entre les deux variables.

Quand est-ce qu'une relation est considérée comme forte ? Il n'y a pas de règle absolue, mais on peut se dire *par convention* qu'une corrélation en valeur absolue supérieure à 0,5 indique une forte corrélation.



3.1.2 Corrélation de Pearson et de Spearman ?

Par défaut, la fonction cor calcule une corrélation dite “de Pearson”, utile pour décrire des relations linéaires monotones entre des variables quantitatives, et lorsque les courbes des distributions des variables ont à peu près une “forme normale”, c’est-à-dire symétrique et en cloche. L’indice de corrélation de Pearson est assez sensible aux points aberrants, les “outliers”.

Tip

Mais en fait, c'est quoi une distribution normale ?
C'est une distribution qui suit la "loi normale" dont la courbe s'appelle aussi courbe de Gauss ou courbe en cloche, elle est symétrique par rapport à sa moyenne.
Cette distribution particulière est rarement présente dans un jeu de données sauf pour certaines variables particulières (la taille, le poids dans une population).
Cette distribution a beaucoup de propriétés statistiques particulières qui la rende intéressante comme point de départ d'une description statistique.
On retrouve aussi cette loi statistique quand on cherche à faire de l'inférence statistique (établir des résultats pas seulement à l'échelle de son échantillon mais pour l'ensemble de la population dont il est représentatif).

On peut résumer quelques configurations idéal-typiques entre deux variables quantitatives avec la figure ci-dessous :

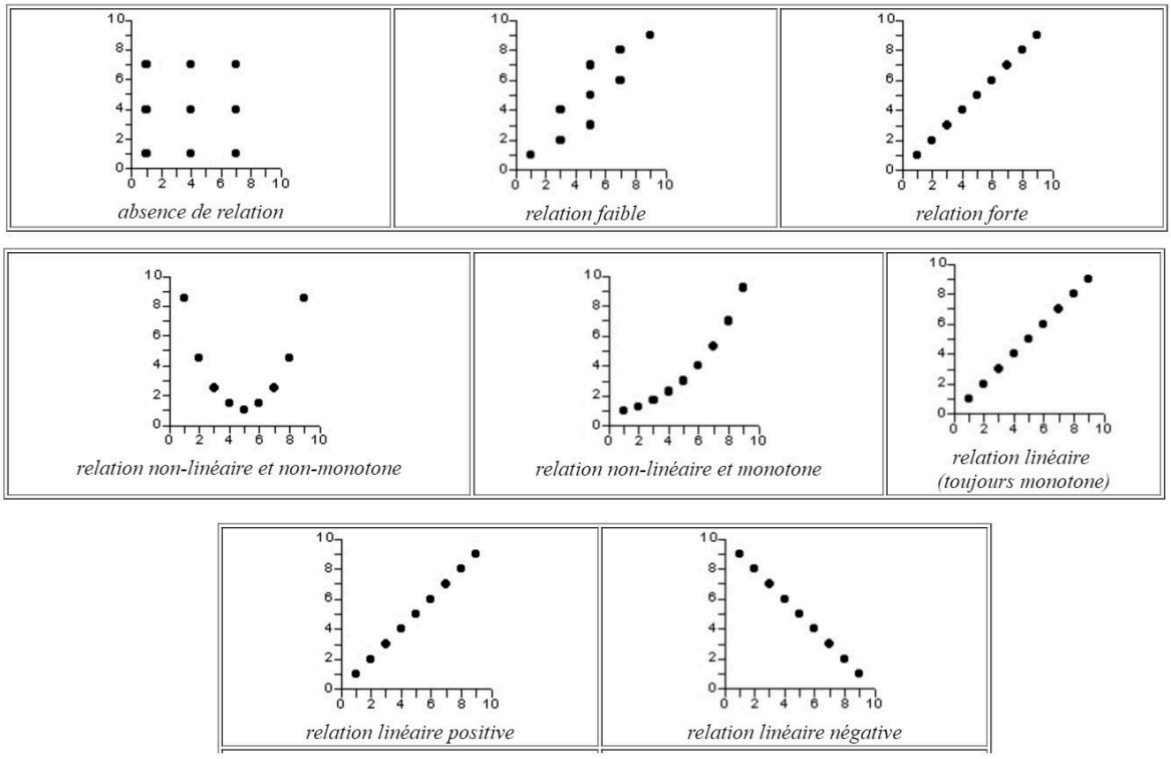


Figure 3.2: Type de relations entre deux variables quantitatives

Lorsque ces conditions ne sont pas bien remplies, on peut réfléchir à calculer une corrélation dite "de Spearman", aussi appelée "corrélation des rangs". Le principe est que la corrélation

est calculée sur les rangs des valeurs des variables plutôt que sur leur valeur elles-mêmes. Ce peut être par exemple assez utile pour travailler sur la relation entre l'âge et le salaire, dans la mesure où le salaire a une distribution très asymétrique (on pourrait aussi transformer le salaire en log pour que la distribution soit moins dissymétrique).

```
cor(Salaires$age, Salaires$salaire,
    method = c("pearson"))
cor(Salaires$age, Salaires$salaire,
    method = c("spearman"))
```

Tip

On travaille ici sur des données qui n'ont pas de valeur manquante (des NA), sinon il aurait fallu rajouter l'argument use="complete.obs" dans la fonction corrélation.

Exercice

Étudier la corrélation entre l'expérience et le salaire.

3.1.3 La corrélation sur des sous-groupes

On peut aussi calculer des corrélations par groupe, par exemple étudier l'association entre l'âge et l'expérience suivant le sexe.

```
Salaires |>
  group_by(genre) |>
  summarise(
    cor_pearson = cor(age, salaire, method = "pearson", use = "complete.obs"),
    cor_spearman = cor(age, salaire, method = "spearman", use = "complete.obs")
  )
```

On pourra chercher à interpréter pourquoi l'âge a un effet plus fort sur le salaire chez les hommes que chez les femmes (carrières plus linéaires pour les hommes etc).

Dans esquisse, on peut utiliser l'onglet color et/ou facet pour créer deux nuages de points séparés pour les hommes et les femmes. À noter que la forme du nuage suggère que quel que soit l'âge, le salaire reste très faible et ne varie pas en fonction de l'âge (travail à temps partiel ?).

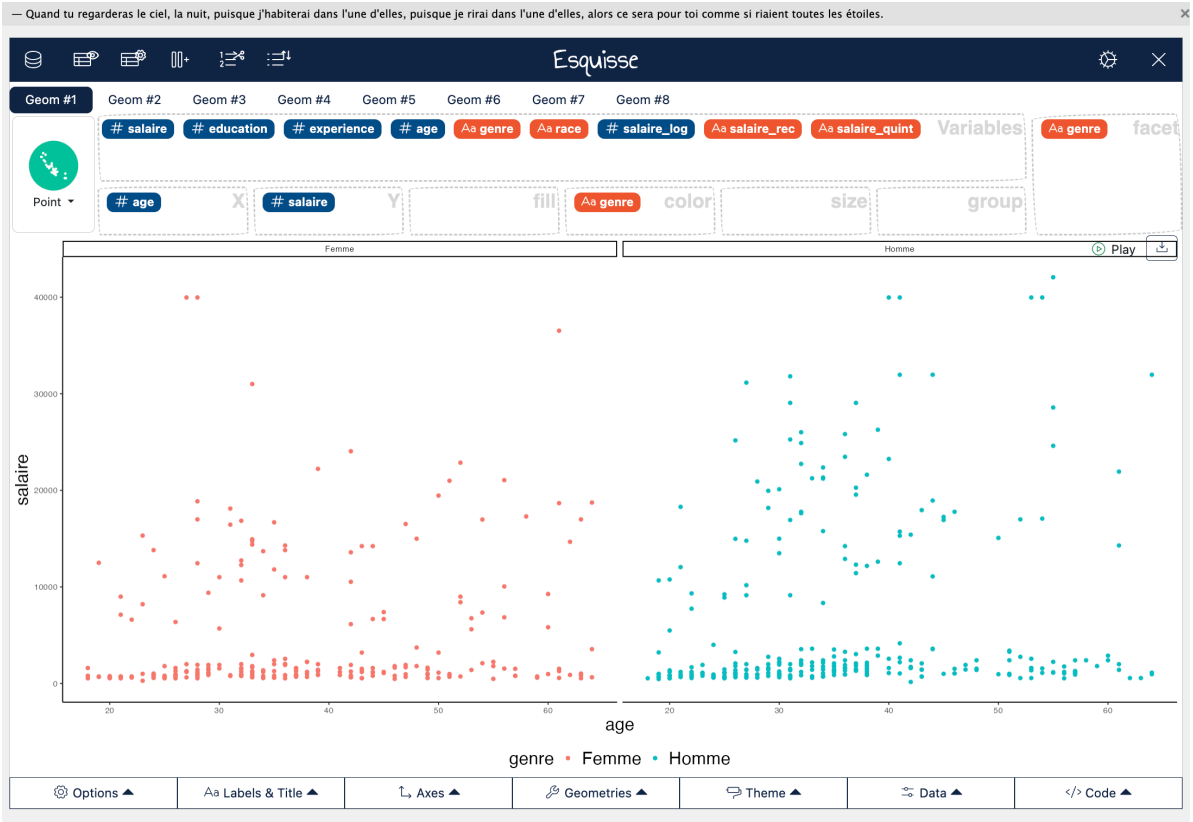


Figure 3.3: Nuages de points avec facet dans esquisse

Exercice

Étudier la corrélation entre l'expérience et le salaire suivant la catégorie raciale.

3.1.4 Matrice des corrélations et matrice des nuages de points

On peut enfin vouloir créer une matrice des corrélations, par exemple de toutes les variables quantitatives de notre jeu de données :

```
Salaires |>
  select(where(is.numeric))  |>     # Sélectionne toutes les variables numériques
  cor(use = "complete.obs", method = "pearson") |> # Matrice de corrélation
  round(2) # arrondit les décimales à deux
```

Cette matrice de corrélation peut aussi être représentée avec des nuages de points grâce au package GGally que nous avons chargé dans ce chapitre :

```
Salaires |>
  select(where(is.numeric)) |>
  ggcorr(label = TRUE, label_round = 2)
```

Ou :

```
Salaires |>
  select(where(is.numeric)) |>
  ggpairs()
```

3.2 L'association entre une variable quali et une variable quanti et la boite à moustaches

3.2.1 Statistiques univariées par groupe

Dans le chapitre sur les statistiques univariées, nous avons abordé les cinq nombres de Tukey et les quantiles pour résumer une distribution d'une variable quantitative.

Grâce aux lignes de codes mises en place, nous avons juste besoin d'ajouter une autre fonction pour réaliser ces statistiques en fonction de différentes modalités d'une variable, à savoir, group_by :

```
Salaires |>
  group_by(genre) |> # ou sexe
  reframe(
    stat = c("min", "Q1", "médiane", "Q3", "max"),
    salaire = fivenum(salaire)
  )
```

Si on veut obtenir deux colonnes séparées pour les Hommes et les Femmes, il suffit d'ajouter une fonction pivot_wider, qui redéploie le tableau, en prenant comme nouveaux noms de colonnes les modalités la variable indiquée par l'argument names_from (ici genre, ou sexe) et les valeurs assignées sont celles de la colonne précédemment nommée salaire :

```
Salaires |>
  group_by(genre) |> # ou sexe
  reframe(
    stat = c("min", "Q1", "médiane", "Q3", "max"),
    salaire = fivenum(salaire)
  ) |>
  pivot_wider(names_from=genre,values_from=salaire)
```

À ce propos, nous venons de réaliser notre première transformation d'un tableau de données dans R ! Pas si compliqué, non ? Pour quelques outils de transformation si besoin, voir les cheatsheet de R...

> Exercice
>
> Créer un tableau des quantiles de salaire suivant la race. Dans le tableau final, les colonnes correspondront aux différentes catégories raciales. Faire une sortie de ce tableau dans le Viewer.

3.2.2 Résumer par la moyenne et l'écart-type

Il est assez courant dans les articles de décrire ses variables quantitatives en indiquant la moyenne (indicateur de centralité) et l'écart-type (indicateur de dispersion, correspondant à la racine carrée de la variance).

```
Salaires |>
  group_by(genre) |>
  summarise(
    mean= mean(age,na.rm=T),
    sd= sd(age,na.rm=T),
)
```

genre	mean	sd
Femme	38	12
Homme	36	11

Figure 3.4: Moyenne et écart-type de l'âge pour les hommes et les femmes

Pourquoi ces deux indicateurs sont-ils utilisés ? D'abord, on peut dire ici avec ce tableau qu'en moyenne les femmes sont un peu plus âgées que les hommes et qu'il y a aussi une plus forte variabilité de l'âge chez les femmes que chez les hommes (sd pour écart-type). Par ailleurs, *si on suppose que l'âge a un distribution normale dans l'échantillon* (condition rarement remplie mais passons), on peut dire que :

- 68 % des femmes ont un âge compris entre [38-12 ; 38 + 12]=[26 ; 50] et 68 % des hommes ont un âge compris entre [25 ; 47],
- 95 % des femmes ont un âge compris entre [38-**2***12 ; 38 + **2***12]=[14 ; 62] et 95 % des hommes ont un âge compris entre [14; 24],

- 99 % des femmes ont un âge compris entre [38-**3***12 ; 38 + **3***12], etc…

Bref, tout cela donnerait des intervalles tout à fait cohérents si les distributions des variables étaient normales, ce qui est rarement le cas (enfin, pas toujours en tout cas), et l'intérêt de cette approximation est de donner corps à ce que veut dire l'écart-type comme indicateur de dispersion d'une variable.

Exercice

Calculer la moyenne et l'écart-type du salaire suivant la catégorie raciale. Commenter.

Tip

Faut-il résumer une variable par sa moyenne ou sa médiane ?
Par son écart-type ou par des quantiles ?
La première option est très courante (notamment dans les revues anglophones) mais pas forcément la plus judicieuse…
Quand on travaille sur des distributions asymétriques (donc non normales) !
Car :

- la moyenne est sensible aux valeurs extrêmes d'une distribution
- l'idée de dispersion des données par l'écart-type ne sera pas non plus très fiable sur une telle distribution…

3.2.3 La boîte à moustaches : un peu d'explication

Revenons à des statistiques plus amusantes avec John Tukey qui a proposé dans les années 1950 le graphique de la boîte à moustache pour résumer d'un coup d'oeil la distribution d'une variable quantitative. En anglais, on dit boxplot ou box-and-whisker plot (le deuxième est autrement plus amusant, comme le nom en français !).

L'idée est de représenter graphiquement certaines valeurs clefs de la distribution d'une variable quantitative, de manière à visualiser en un clin d'oeil la position centrale et la dispersion des données d'une distribution. Ce type de graphique a surtout de l'intérêt quand on compare des distributions d'une même variable quantitative entre plusieurs modalités d'une variable qualitative (le sexe, les catégories raciales, etc).

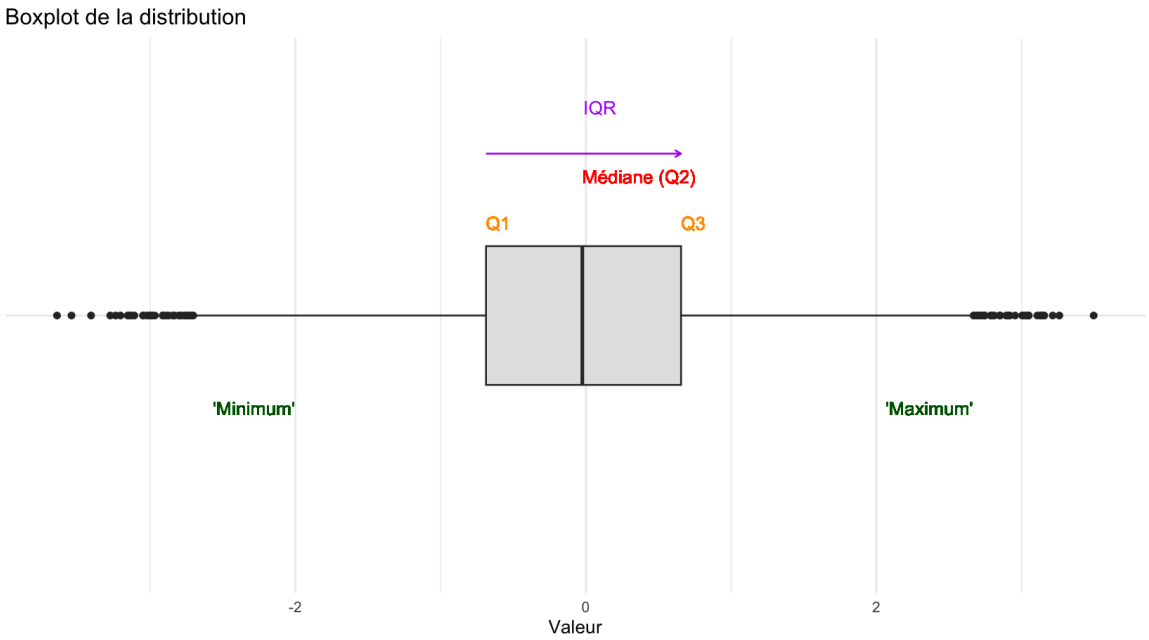


Figure 3.5: Une boîte à moustaches

Détail sur la boîte à moustaches

Il y a plusieurs variantes de la boîte à moustaches, mais le principe c'est qu'il y a toujours une boite et des moustaches sur les côtés. Les valeurs des extrêmités de la boîte et des moustaches, dans la version originale et celle obtenue dans R par défaut sont les valeurs suivantes :

- L'extrêmité gauche de la boîte (en rouge ici) correspond au Q1, c'est-à-dire au premier quartile (autrement dit, 25 % des individus sont situés en-dessous de cette valeur).

```
quantile(Salaires$salaire,probs=0.25)
```

- L'extrêmité droite de la boîte correspond au Q3, le troisième quartile (autrement dit, 25 % des individus sont situés *au-dessus* de cette valeur).

```
quantile(Salaires$salaire,probs=0.75)
```

- L'écart entre les deux limites de la boîte correspond à l'étendue inter-quartile (interquartile range), autrement dit 50 % des individus de la distribution prennent une valeur comprise au sein de cette boîte.

```r
quantile(Salaires$salaire,probs=0.75)-quantile(Salaires$salaire,probs=0.25)
#ou :
IQR(Salaires$salaire)
```

- La barre au milieu de la distribution correspond à la médiane de la distribution.
- L'extrêmité basse ou à gauche de la moustache correspond au premier quartile moins 1,5 fois l'écart inter-quartile ('minimum'=Q1-1.5*IQR).

```r
quantile(Salaires$salaire,probs=0.25)-1.5*IQR(Salaires$salaire)
```

- L'extrêmité haute ou à droite de la moustache correspond au troisième quartile plus 1,5 fois l'écart inter-quartile ('maximum'=Q3+1.5*IQR).

```r
quantile(Salaires$salaire,probs=0.75)+1.5*IQR(Salaires$salaire)
```

Si on comprend relativement facilement l'intérêt des extrêmités de la boîte comme indicateurs de distribution, pourquoi diable cette définition des extrêmités des moustaches ? Pour le comprendre, il faut encore revenir à la densité de distribution d'une loi normale :

- Par définition, il y a 50% des observations de la distribution entre le Q1 et le Q3
- Dans le cas d'une distribution normale (symétrique par rapport à la moyenne, qui est alors aussi égale à la médiane et au mode), environ 25% des observations sont comprises entre l'extrêmité de la moustache inférieure et le le Q1, de même à droite du Q3 jusqu'à la limite de la moustache supérieure.
- Ce qui est en dehors des extrêmités, ce sont alors des points aberrants (qui représentent dans le cas d'une loi normale exactement 0,7% des observations, ce qu'on pourrait montrer mathématiquement, mais on l'a ici simulé sur une distribution normale avec 10 000 observations et on obtient 0,6% en additionnant à gauche et à droite, bon c'est presque ça !).

En somme, les moustaches nous indiquent l'étendue des valeurs non aberrantes de la distribution.

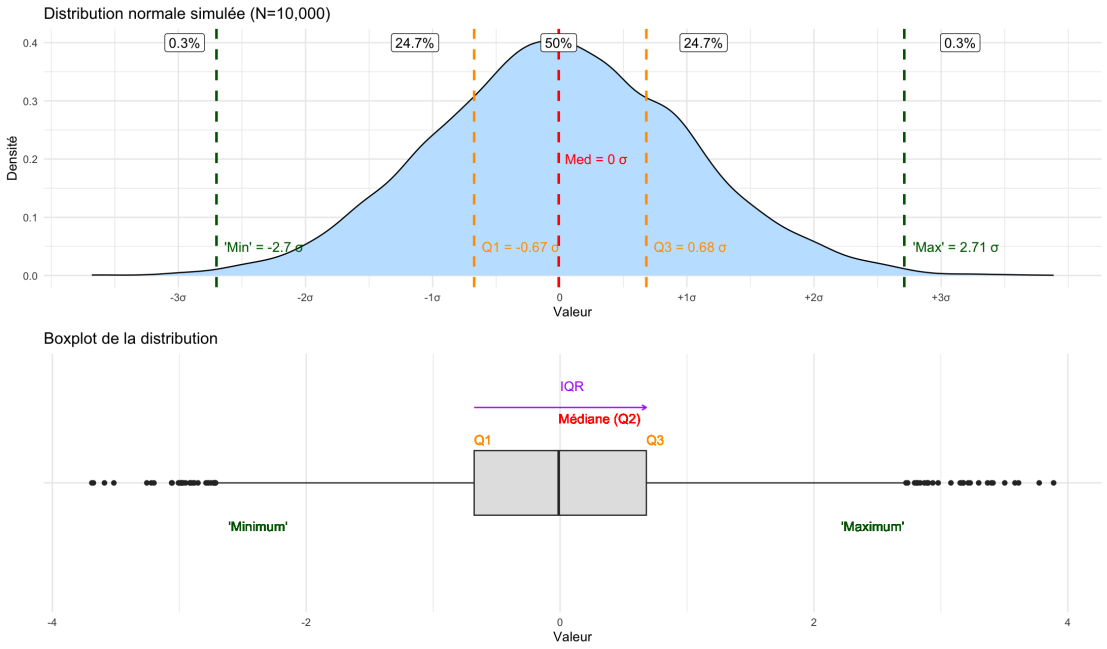
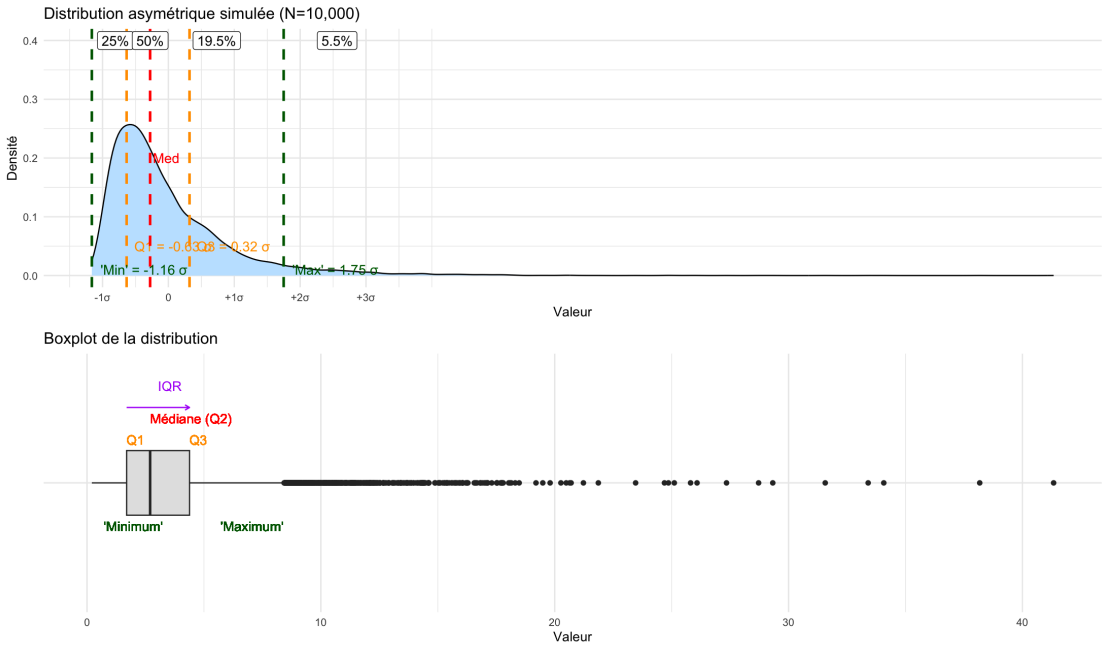


Figure 3.6: Densité et boîte à moustaches d'une distribution normale

À noter : si la distribution statistique décrite n'est pas normale, alors il n'est pas dit que les outliers représentent seulement 0,7%, comme on le voit aisément avec cette simulation d'une distribution assez asymétrique, où les outliers représentent 5,5% des observations.



Cela reste faible et montre bien l'intérêt de la boîte à moustaches : elle nous montre l'étendue de quasi toutes les observations. Le décalage observé sur ce second graphique de la boîte et de la médiane (qui n'est plus centré) nous montre bien que la courbe est asymétrique.

3.2.4 La boîte à moustaches : un peu de pratique

Pour réaliser des boîtes à moustaches, rien de plus simple, nous pouvons de nouveau avoir recours à esquisse :

```
esquisser(Salaires)
```

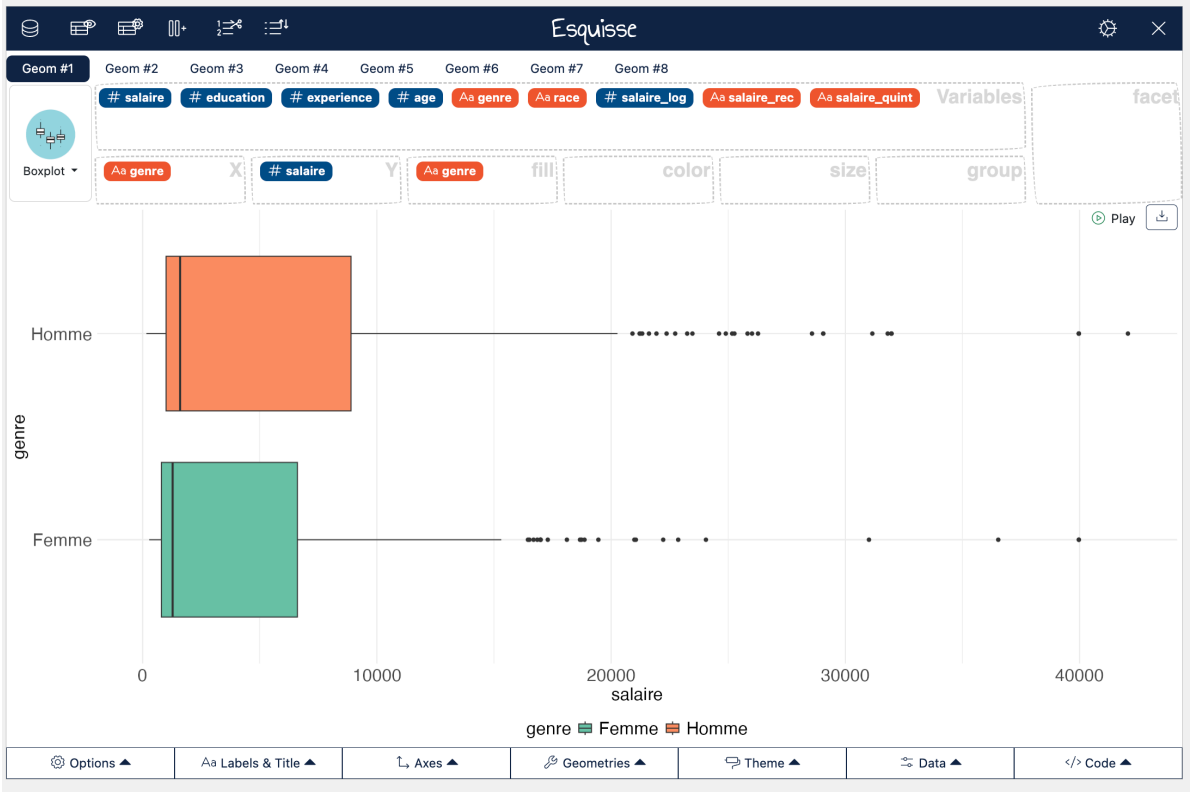


Figure 3.7: Boîte à moustaches du salaires entre les hommes et les femmes

On voit bien qu'aussi bien chez les hommes que les femmes, la distribution du salaire est très asymétrique... Beaucoup de monde qui gagne peu et une minorité qui gagne pas mal, les outliers sont tous à des niveaux stratosphériques par rapport au reste de la distribution...

Mais on voit aussi que la médiane du salaire des hommes et plus élevée que celle des femmes et que l'étendue des valeurs de salaire est plus forte chez les hommes que chez les femmes.

L'asymétrie de la distribution justifie souvent sa transformation logarithmique ce qui la rend plus "symétrique" (on fait log(Salaires$salaire), ce qu'on peut faire directement dans Esquisse en choisissant log dans Y-axis transform, et fait alors surtout apparaitre que les femmes gagnent moins que les hommes.

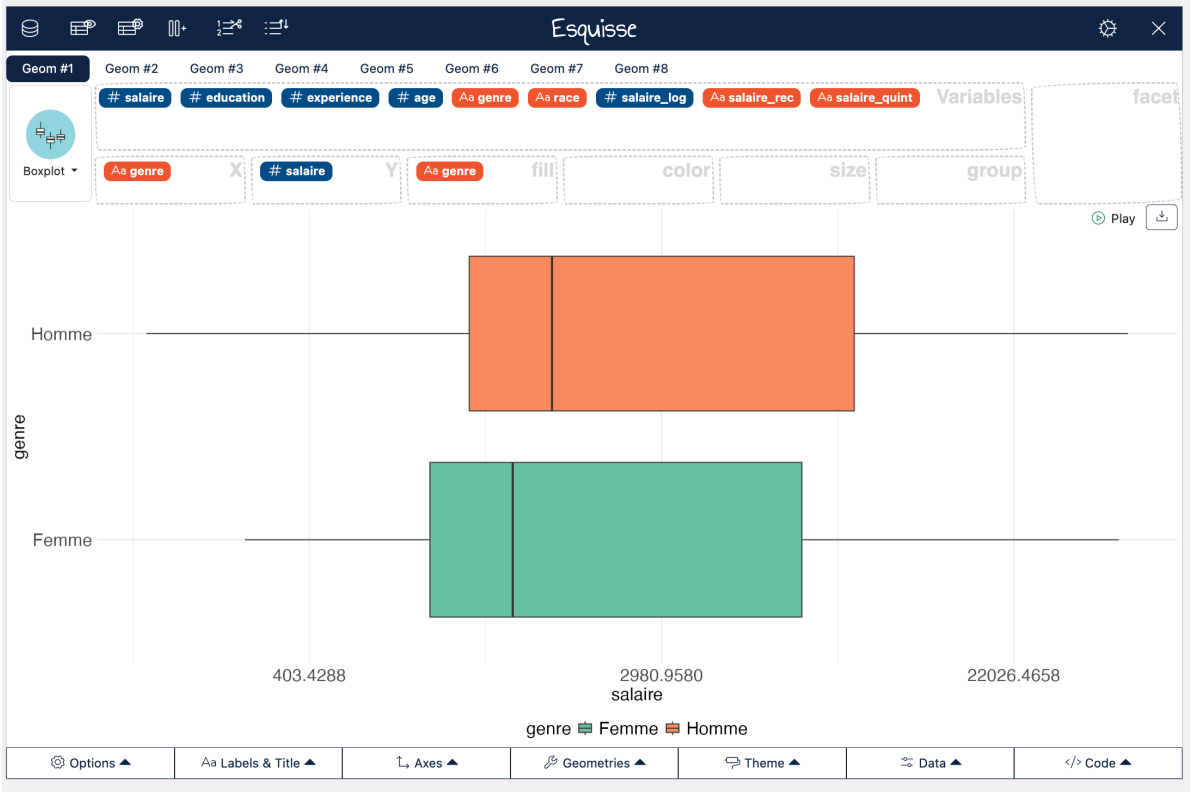


Figure 3.8: Boîtes à moustaches sur le logarithme du salaire

Exercice

Créer la boîte à moustaches de l'âge entre les hommes et les femmes. Que peut-on dire par rapport à la section précédente ?
Créer la boîte à moustache de l'âge suivant les catégories raciales. Commenter.

3.3 L'association entre deux variables qualis et le tableau croisé

3.3.1 Sur quelques conventions du tableau croisé

Nous en arrivons au tableau croisé pour étudier les associations entre deux variables qualitatives.

D'abord, rappelons quelques *conventions.* On s'emmêle facilement les pinceaux quand on fait des tableaux croisés, et on peut garder à l'esprit quelques petites choses en tête :

- Quelle est ma *variable-réponse* ou ma *variable dépendente,* c'est à dire la variable dont on suppose qu'elle dépend d'un autre facteur ?
- Quelle est ma *variable indépendante*, c'est à dire la variable dont on suppose qu'elle a un effet sur la *variable dépendante* ?

On peut par exemple ici supposer que le niveau de salaire (codé de manière catégorielle) dépend du sexe, mais aussi de la catégorie raciale, du niveau d'éducation, etc. On pourrait aussi supposer que le niveau de diplôme dépend de la catégorie raciale, etc.

Pour visualiser si un facteur affecte une variable dépendante, on peut réaliser un tableau croisé dans lequel :

- On mettra la variable dépendante "en colonnes"
- On met la variable indépendante "en lignes"
- On calcule des pourcentages en ligne (on a besoin de normaliser les effectifs des lignes pour les comparer entre elles !)
- On comparera alors les lignes d'une même colonne entre elles.

Bien sur, ça c'est de la théorie. Il y a plein de cas où en fait on est davantage intéressé par les pourcentages en colonnes, voire les pourcentages totaux, on veut inverser les lignes et les colonnes… Toutefois, avoir ces petites conventions en tête permet de s'y retrouver quand on est un peu perdu dans ses traitements statistiques.

3.3.2 Mise en pratique du tableau

Nous retrouvons les fonctions du package questionr. D'abord nous avons vu précédemment comment catégoriser le salaire en quintiles, ce que nous pouvons reproduire, je ne mets ci-dessous que les lignes de code correspondantes :

```
icut()
Salaires$salaire_rec <- cut(Salaires$salaire,
                            include.lowest = TRUE,
                            right = TRUE,
                            dig.lab = 4,
                            breaks = c(160, 800, 1200, 1920, 11008, 42064)
)
irec()
## Recoding Salaires$salaire_rec into Salaires$salaire_quint
Salaires$salaire_quint <- Salaires$salaire_rec |>
  fct_recode(
    "Très faible" = "[160,800]",
```

```
    "Faible" = "(800,1200]",
    "Moyen" = "(1200,1920]",
    "Élevé" = "(1920,1.101e+04]",
    "Très élevé" = "(1.101e+04,4.206e+04]"
  )
```

Pour créer un tableau croisé des effectifs, on pourra écrire :

```
Salaires |> freqtable(race,salaire_quint)
```

La variable en ligne (ici, race) est la variable indépendante, c'est la première à être écrire, tandis que la variable en colonne (ici, salaire_quint) est la variable dépendante.

Pour obtenir des pourcentages en ligne, il suffit d'écrire :

```
Salaires |> freqtable(race,salaire_quint) |> rprop()
```

Dans ce tableau, on peut aussi ajouter les effectifs totaux des lignes et par exemple ne pas mettre de décimales après la virgule :

```
Salaires |> freqtable(race,salaire_quint) |> rprop(n=T,digit=0)
```

Bien sur, il est possible de réaliser un tableau des pourcentages en colonne :

```
Salaires |> freqtable(race,salaire_quint) |> cprop(n=T,digit=0)
```

Ou des pourcentages totaux (qu'on appelle aussi pourcentages conjoints) :

```
Salaires |> freqtable(race,salaire_quint) |> prop(n=T,digit=0)
```

Exercice

1. Étudier le lien entre le sexe et le niveau de salaire.
2. Réaliser un tableau dans le Viewer.

3.3.3 Le tableau et son graphique

On peut aussi réaliser un diagramme à barres empilées ou adjacentes de l'association entre la catégorie raciale et le niveau de salaire :

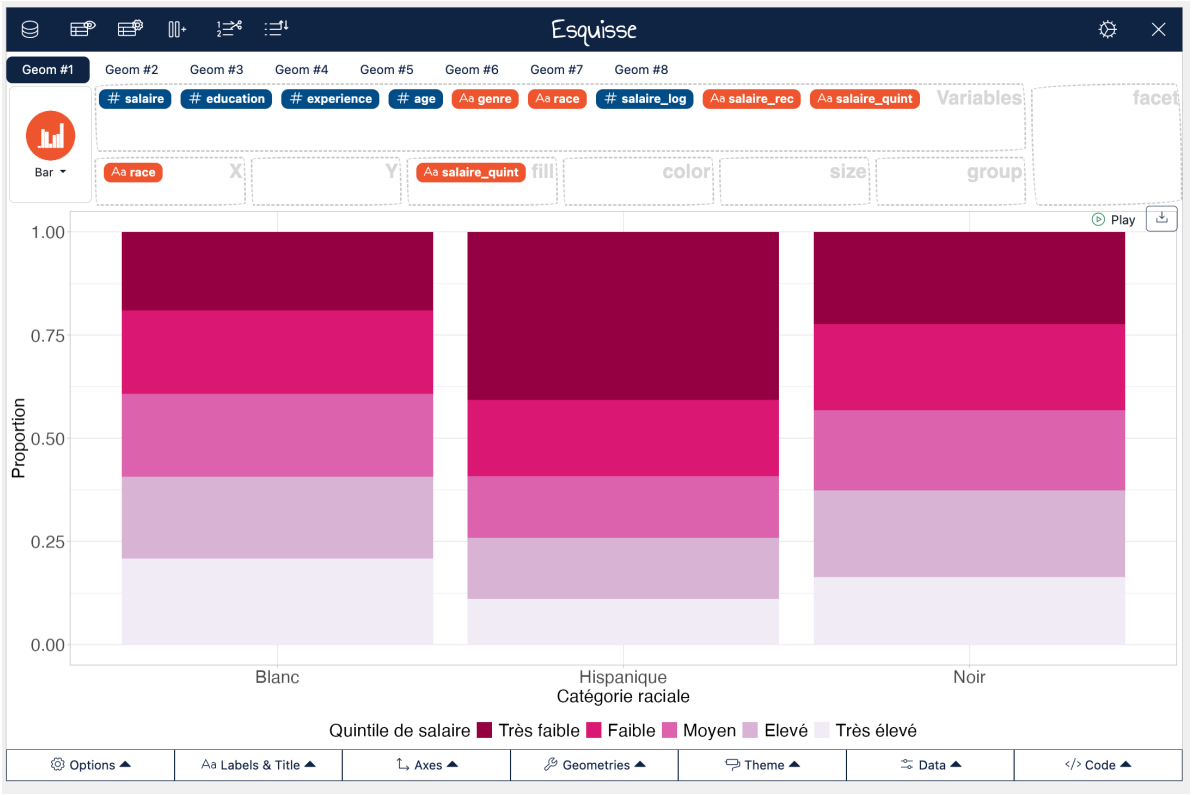


Figure 3.9: Graphique à barres avec Esquisse

On pourrait améliorer ce graphique de deux manières :

- En inversant l'ordre des étiquettes empilées (Très élevé en haut, très faible en bas)
- En mettant l'axe des Y en pourcentages (attention, il faut avoir installer le package scales)

```
ggplot(Salaires) +
  aes(x = race, fill = fct_rev(salaire_quint)) +  # Inverser l'ordre d'empilement
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) + # Axe des Y en % avec package scale
  scale_fill_brewer(palette = "PuRd", direction = -1,
                    guide = guide_legend(reverse = TRUE)) +  # Légende dans le bon ordre
```

```
  labs(
    x = "Catégorie raciale",
    y = "Pourcentage",
    fill = "Quintile de salaire"
  ) +
  theme_light() +
  theme(
    legend.position = "bottom",
    axis.title.y = element_text(size = 18L),
    axis.title.x = element_text(size = 18L),
    axis.text.y = element_text(size = 18L),
    axis.text.x = element_text(size = 18L),
    legend.text = element_text(size = 18L),
    legend.title = element_text(size = 18L)
  )
```

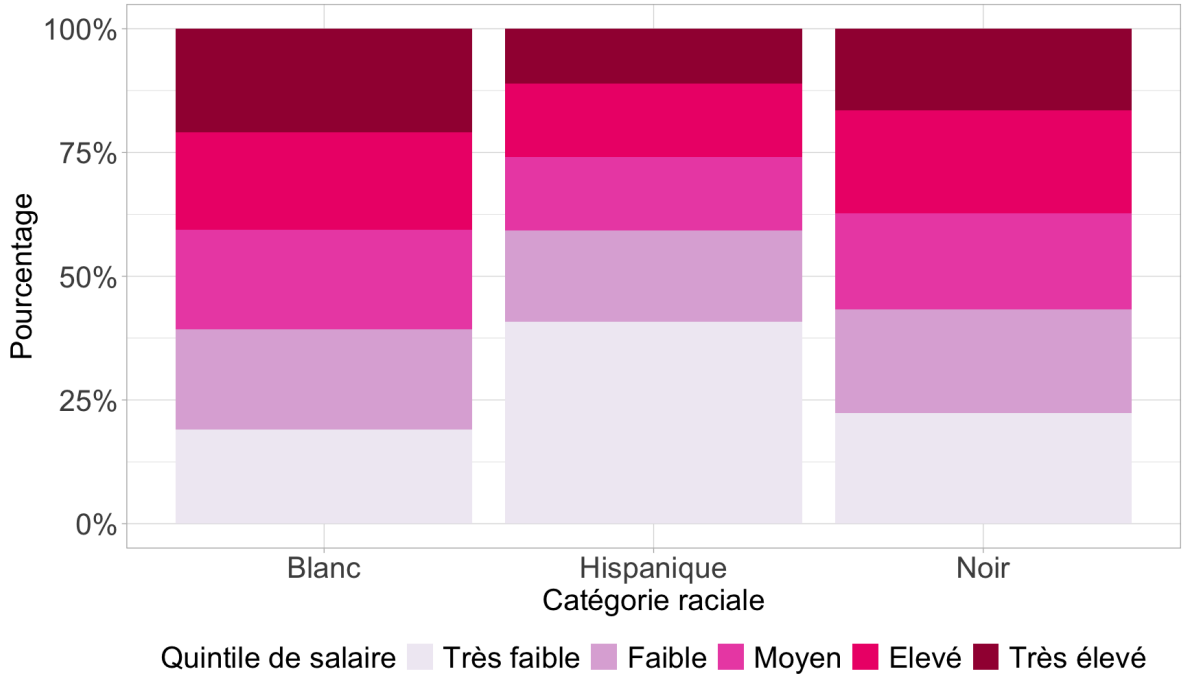


Figure 3.10: Graphique à barres empilées amélioré

Bonus 1

On peut vouloir plutôt réaliser un diagramme à barres adjacentes plutôt que empilées. Dans ce cas, il est plus simple de repartir du tableau des pourcentages en ligne :

```r
#On crée le tableau croisé avec % en ligne
tab <- Salaires |>
  freqtable(race, salaire_quint) |>
  rprop(total=F) #On enlève les % totaux

# Il faut transformer ce tableau en format tidy pour le graphique
df_tab <- as.data.frame.matrix(tab) |> #transformation en une matrice de format data.frame
  rownames_to_column("race") |> #Les lignes étaient des "row.names" auxquelles on assigne u
  pivot_longer(
    cols = -race,
    names_to = "salaire_quint",
    values_to = "pct"
  ) #On transforme le tableau de telle sorte que tous les % soient dans la même colonne et
df_tab$salaire_quint <- df_tab$salaire_quint |>
  fct_relevel(
    "Très faible","Faible","Moyen","Elevé","Très élevé"
  )

ggplot(df_tab, aes(x = salaire_quint, y = pct / 100, fill = race)) +
  geom_bar(stat = "identity",position = position_dodge2(width = 0.9,preserve="single"))+
  geom_text(aes(label = paste0(round(pct,0), "%")),
            position = position_dodge2(width = 0.9,preserve="single"),
            vjust=-.3,
            size = 5) +
  scale_y_continuous(lim=c(0,.45),labels = scales::percent_format(accuracy = 1)) +
  scale_fill_brewer(palette = "Dark2", direction = -1,
                    guide = guide_legend(reverse = T)) +
  labs(
    x = "Quintile de salaire",
    y = "Pourcentage",
    fill = "Catégorie raciale"
  ) +
  theme_light() +
  theme(
    legend.position = "bottom",
    axis.title.y = element_text(size = 18L),
    axis.title.x = element_text(size = 18L),
    axis.text.y = element_text(size = 18L),
    axis.text.x = element_text(size = 18L),
    legend.text = element_text(size = 18L),
    legend.title = element_text(size = 18L)
  )
```

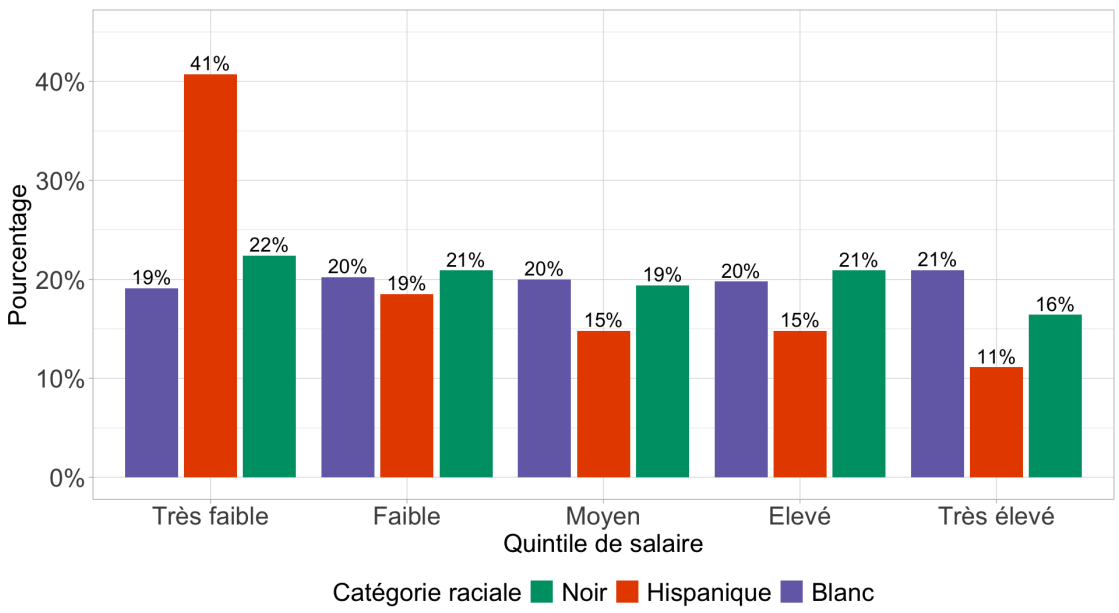


Figure 3.11: Diagramme à barres adjacentes

Bonus 2

On pourrait aussi vouloir créer un diagramme à barres avec les étiquettes de valeurs de pourcentage sur le graphique. Dans ce cas, il est plus simple de repartir du tableau des pourcentages en ligne :

```r
#On crée le tableau croisé avec % en ligne
tab <- Salaires |>
  freqtable(race, salaire_quint) |>
  rprop()

# Il faut transformer ce tableau en format tidy pour le graphique
df_tab <- as.data.frame.matrix(tab) |> #transformation en une matrice de format data.frame
  rownames_to_column("race") |> #Les lignes étaient des "row.names" auxquelles on assigne u
  select(-Total) |> #On enlève les lignes de total
  pivot_longer(
    cols = -race,
    names_to = "salaire_quint",
    values_to = "pct"
  ) #On transforme le tableau de telle sorte que tous les % soient dans la même colonne et
df_tab$salaire_quint <- df_tab$salaire_quint |>
  fct_relevel(
    "Très faible","Faible","Moyen","Elevé","Très élevé"
  )

# Graphique
ggplot(df_tab, aes(x = race, y = pct / 100, fill = fct_rev(salaire_quint))) +
  geom_bar(stat = "identity") +
  geom_label(aes(label = paste0(round(pct,0), "%")),
             position = position_stack(vjust = 0.5),
             size = 5) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  scale_fill_brewer(palette = "PuRd", direction = -1,
                    guide = guide_legend(reverse = T)) +
  labs(
    x = "Catégorie raciale",
    y = "Pourcentage",
    fill = "Quintile de salaire"
  ) +
  theme_light() +
  theme(
    legend.position = "bottom",
    axis.title.y = element_text(size = 18L),
    axis.title.x = element_text(size = 18L),
    axis.text.y = element_text(size = 18L),
    axis.text.x = element_text(size = 18L),
    legend.text = element_text(size = 18L),
    legend.title = element_text(size = 18L)
  )
```

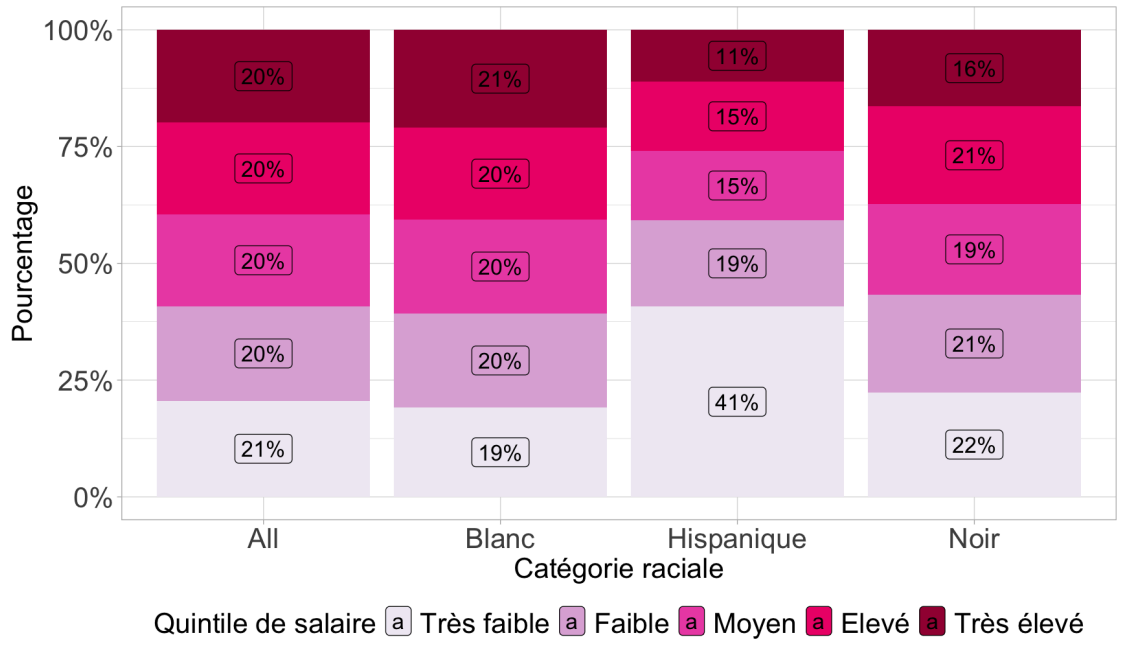


Figure 3.12: Diagramme à barres empilées avec étiquettes

Exercice

Étudier le lien entre sexe et niveau de salaire à l'aide d'un diagramme à barres empilées ou adjacentes.

3.3.4 La force de l'association

Pour étudier l'intensité de l'association entre deux variables qualitatives, l'indice le plus évident est le V de Cramer qui prend une valeur entre 0 (pas d'association) et 1 (association parfaite).

À partir de 0,2 ou 0,3, on pourra considérer qu'il y a une association notable.

On le calcule ainsi :

```
tab <- Salaires |>
  freqtable(race, salaire_quint)
cramer.v(tab)
```

Exercice

L'intensité de l'association est-elle plus forte pour entre le salaire et le sexe ou le salaire et la catégorie raciale ?

3.3.5 Un mot sur le test du khi-deux

Difficile de ne pas évoquer pour finir le test du chi-deux, sans toutefois trop s'y attarder. Précisons que Julien Barnier a écrit un récapitalutif très exhaustif sur ce qu'est et n'est pas le test du khi-deux (ou chi-deux ou -deux ou chi-squared…).

À quoi sert le test du khi-deux :

- Déterminer la probabilité que les lignes et les colonnes du tableau croisé sont indépendantes
- Évaluer si la répartition des effectifs dans une table de contingence est significativement différente de la table calculée sous l'hypothèse d'indépendance des deux variables croisées

La distribution du salaire est-elle indépendante de celle de la catégorie raciale des individus ? Y-a-t-il une association entre race et salaire ?

Explications sur le test du khi-deux

On va poser un test statistique, avec une hypothèse H0, ou hypothèse nulle qui est que :
La distribution des quintiles de salaire est indépendante de la catégorie raciale.
On cherche à savoir à quoi ressemblerait notre tableau croisé si les deux variables étaient effectivement indépendantes l'une de l'autre et quelle est la probabilité (la p-valeur) pour que les deux variables soient effectivement indépendantes l'une de l'autre, modulo nos fluctuations d'échantillonnage.
En pratique, les variables sont indépendantes si :

- Les pourcentages lignes du tableau croisé sont les mêmes pour toutes les lignes
- Les pourcentages colonnes du tableau croisé sont les mêmes pour toutes les colonnes

Dans notre exemple, lignes et colonnes ne semblent pas très indépendantes.
Mais dans un échantillon issu d'une enquête, il est rare que les variables croisées soient parfaitement indépendantes car les données du tableau sont dépendantes de l'échantillon interrogé et tout échantillon est soumis à des biais (qui, si l'échantillon a été construit de manière aléatoire, sont dus *au hasard,* donc des *fluctuations d'échantillonage*).
Le test du khi-deux permet de savoir à partir de quel seuil on peut estimer que les variations observées par rapport à la situation d'indépendante sont dues au hasard et à partir de quand elles sont dues à un lien entre les variables.

À partir du tableau des effectifs, on peut calculer les effectifs théoriques, *si les deux variables étaient indépendantes*, grâce aux marges des lignes et des colonnes :
Effectif théorique d'une cellule = (total ligne x total colonne) / total global
Ce qu'on peut calculer manuellement à partir des marges (les totaux) qu'on peut obtenir par exemple en faisant ainsi :

```
Salaires |>
  freqtable(race, salaire_quint) |>
  prop(n=T)
```

L'effectif théorique de la cellule Blanc x Très faible est donc 90,6 car :

```
110*440/534
```

Heureusement R dispose de la fonction chisq.test() qui permet notamment d'afficher les effectifs théoriques :

```
tab<-Salaires |>
  freqtable(race, salaire_quint)
chi<-chisq.test(tab)
chi$expected
```

Pour mémoire les effectifs observés sont :

```
chi$observed
```

Alors à quel point les effectifs théoriques divergent des effectifs observés ?
Pour ce faire, on calcule des "khi-deux partiels" définis comme la distance standardisée entre les effectifs théoriques et observés :

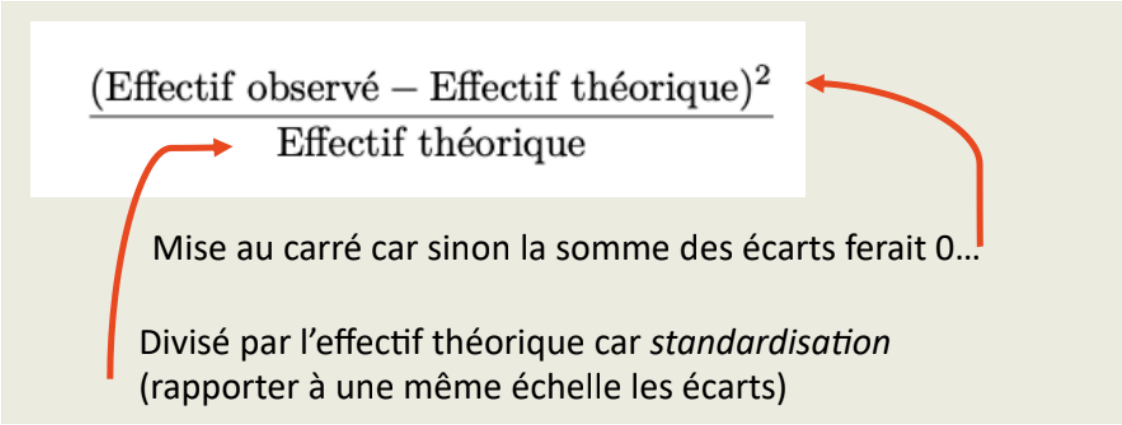


Figure 3.13: Définition des khi-deux partiels

Ces khi-deux partiels peuvent être calculés comme ceci :

```
(chi$expected-chi$observed)^2/chi$expected
```

	Très faible	Faible	Moyen	Elevé	Très élevé
Blanc	0.486	0.000	0.025	0.003	0.249
Hispanique	5.317	0.039	0.323	0.323	1.039
Noir	0.104	0.015	0.002	0.052	0.398

Figure 3.14: Khi-deux partiels

Ici, on voit déjà que les écarts à l'indépendance sont les plus élevés pour la cellule HispaniquexTrès faible, ce que nous avions repéré avec une surreprésentation de cette catégorie raciale parmi ceux qui gagnent de faibles salaires.
On calcule ensuite la valeur du khi-deux du tableau, qui correspond à la somme des khi-deux partiels :

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

Qu'on obtient aussi ici :

```
chi$statistic
```

Est-ce que cette valeur est faible ou élevée ?
On va pouvoir comparer cette statistique à la "loi du khi-deux", une distribution statistique qui nous donne les valeurs théoriques du khi-deux d'un tableau sous condition d'indépendance.
Cette loi dépend d'un paramètre : le nombre de degrés de libertés. Quel est le nombre de degrés de libertés ici ?
Il dépend grosso modo de la taille du tableau :
(Nombre de lignes - 1) x (Nombre de colonnes - 1)
Ici, nous avons 3 lignes et 5 colonnes, donc le degré de liberté (ddl) est égal à 8.
On le vérifie ici :

```
chi$parameter
```

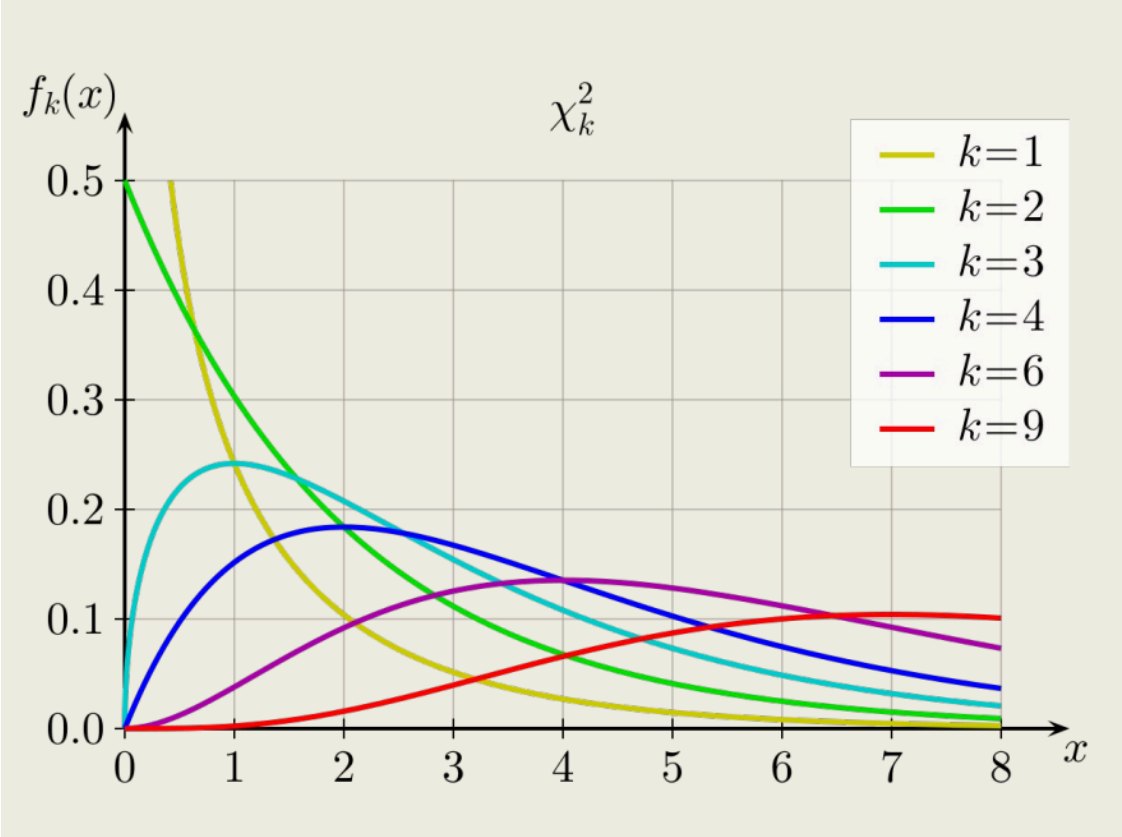


Figure 3.15: Loi du khi-deux suivant le degré de libertés (k)

On va enfin calculer une probabilité (une p-valeur), qui correspond à l'aire sous la courbe théorique à droite de la valeur du khi-deux observée, qui ici vaut 0,397, comme indiqué sur la figure.
En effet, rappelons-nous, la p-valeur c'est la probabilité que le khi-2 théorique soit supérieur ou égal au khi-2 observé sous condition d'indépendance.
Ici, on voit que si les deux variables sont indépendantes, il y a une probabilité de presque 40 % pour que par hasard on obtienne une valeur du khi-2 au moins aussi élevée que celle qu'on a observé (8,374).
Autrement dit, notre valeur du khi-deux est assez plausible / compatible avec le fait que les deux variables soient indépendantes !
Cette p-valeur nous empêche de rejeter l'hypothèse d'indépendance entre catégorie racial et salaire.
En sciences sociales, on retient généralement le seuil d'une p-valeur de 0,05 en dessous de laquelle on estime qu'on peut rejeter l'hypothèse d'indépendance.
Pour ce faire, il aurait fallu ici obtenir un khi-2 observé au moins égal à 15,507 (on en

est loin !).

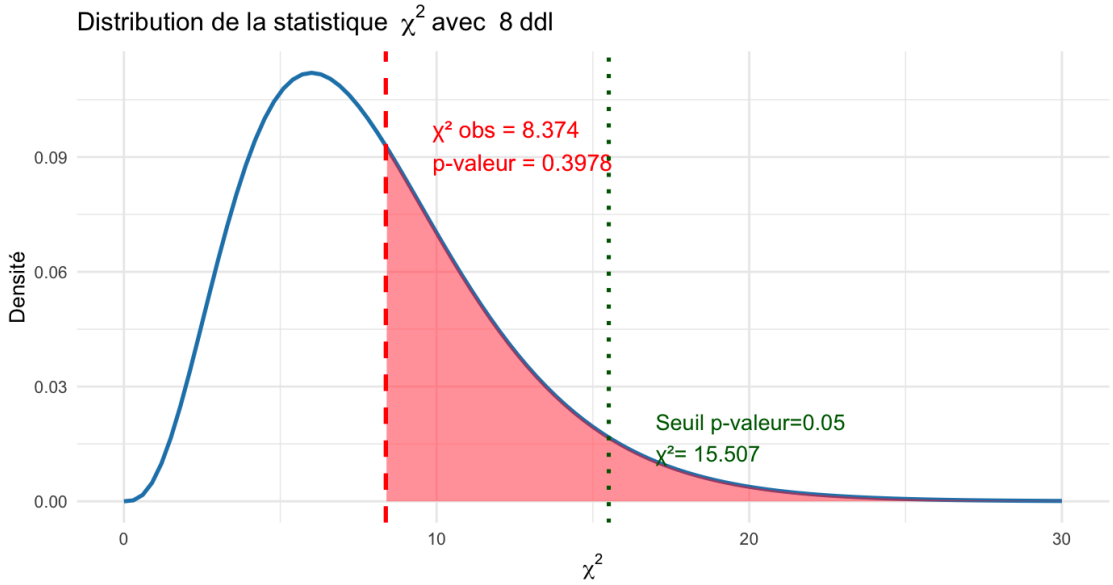


Figure 3.16: Distribution du khi-2 et p-valeur

Alors que conclure ? On ne rejette pas H0, l'hypothèse d'indépendance.
Mais pouvons-nous affirmer que le salaire ne dépend pas de la catégorie raciale ? Eh bien, pas vraiment non plus. En gros, on n'a pas rejeté l'idée que les deux variables sont indépendantes, et les variations observées peuvent être dues à des fluctuations d'échantillonnage… ou pas.
Et si la p-valeur avait été inférieure à 0,05 ? On aurait pu rejeter H0 l'hypothèse d'indépendance et on aurait considéré qu'un lien existe entre les deux variables.
On peut vérifier la p-valeur calculée comme ceci :

```
chi$p.value
```

On pose l'hypothèse nulle (H0) que les deux variables sont indépendantes. On fixe un seuil de significativité pour la p-valeur égal à 0,05, en dessous de laquelle on rejette l'hypothèse d'indépendance et à ce moment là on considère qu'un lien existe entre les variables :

```
tab<-Salaires |>
  freqtable(race, salaire_quint)
chi<-chisq.test(tab)
chi
```

La p-valeur est supérieure à 0,05, donc on ne peut pas rejeter H0 et on ne peut pas conclure

qu'un lien existe entre les deux variables.

> **Exercice**
>
> Tester l'hypothèse d'un lien entre le sexe et les quintiles de salaire.

3.4 Exercices

> **Exercice**
>
> 1. Créer une variable haut_revenu qui vaut "oui" si un individu appartient aux 10 % les mieux rémunérés (salaire 9e décile) et "non" sinon. Calculer la proportion d'individus ayant un haut revenu pour le genre et la race, calculer les V de Cramer et réaliser des tests du khi-deux. Réaliser des diagrammes à barres. Qui sont les plus présents en haut de l'échelle salariale ?
> 2. Recoder la variable âge en quatre groupes :
>
> - Moins de 30 ans
> - 30-44 ans
> - 45-59 ans
> - 60 ans et plus
>
> 3. Pour chaque groupe d'âge, calculer la médiane et les quartiles du salaire et de l'expérience. Représenter ces distributions avec des boxplots. Observe-t-on un effet d'âge linéaire ?
> 4. Décrire à l'aide d'un tableau croisé la distribution de l'âge en fonction des hauts revenus. Calculer le V de Cramer et réaliser un test du khi-deux. Réaliser un diagramme à barres. Qui sont les plus présents en haut de l'échelle salariale ?

Part III

Approfondissements

4 Les données de sondage, des données pondérées

Nous avons jusqu'ici travaillé en supposant que nos données sont issues d'un échantillon qui ne présentent pas de poids de pondération.

Or, c'est rarement le cas quand on travaille sur des données d'échantillon issue d'un protocole d'échantillonnage aléatoire (ou même par quotas, ou même aujourd'hui suivant d'autres méthodes…), pour lesquelles les poids permettent d'assurer la représentativité de l'échantillon à l'ensemble de la population (pour aller vite, dans le cas le plus courant, les poids corrigent des biais dans la collecte des données car la mise en oeuvre du plan de collecte n'a pas permis d'interroger tous les enquêté · es pressenti · es et le poids permet de corriger ces biais).

Ces données sont très courantes en sciences sociales, même si on est aussi amené à travailler sur des populations entières, des données administratives, des données issues du web, etc, pour lesquelles il n'y a pas forcément de variable de poids d'échantillonnage.

Ici, nous mobilisons une version allégée de la base de données "Histoire de vie 2003" de l'Insee, stockée dans le package questionr, pour illustrer l'usage des pondérations dans R.

Exercice

- Créer un script vide.
- Enregistrer ce script dans un dossier (par exemple "Formation R") en le nommant par exemple 4DonneesPond.R.
- Charger les packages mobilisés dans le chapitre précédents (tidyverse, questionr, kableExtra a minima). Installer et charger également le package Hmisc qui contient quelques fonctions utiles pour pondérer des statistiques.
- Charger aussi ggplot2 (esquisse n'est pas utile ici).
- Charger la base de données Histoire de vie grâce à la commande data("hdv2003").
- Nous allons également installer le package descriptio créé par Nicolas Robette, mais au moment d'écrire ce chapitre, la version la plus à jour n'est pas encore sur le CRAN, mais disponible sur son repository (>=1.4.2, sur le CRAN : 1.4.1). Pour installer et charger cette version, on pourra faire tourner les lignes de code suivantes :

```
if (!require(devtools)){
  install.packages('devtools')
  library(devtools)
}
detach("package:descriptio", unload = TRUE, character.only = TRUE)
remove.packages("descriptio")
install_git("https://framagit.org/nicolas-robette/descriptio",force=T)
library(descriptio)
```

4.1 R et les données pondérées

Dans R, il y a plusieurs stratégies pour gérer les données issues d'échantillon :

- Dans une stratégie de statistique inférentielle et notamment à partir du moment où on souhaite pondérer des modèles de régression, on passera par le package survey et son extension srvyr qui permettent de gérer les poids d'échantillonnage, mais aussi la structure de l'échantillon (les strates, les grappes, etc.). On se reportera aux sections spécifiques du guide-R de Joseph Larmarange qui présente l'utilisation de ces packages.
  - Le principe est de définir un objet "design" qui décrit le plan d'échantillonnage à partir duquel on travaille
  - Ces packages ne sont pas présentés ici car dans la pratique on n'a pas toujours forcément besoin de définir précisément le plan d'échantillonnage (ça a de l'importance pour les intervalles de confiance, en tenant compte des variables de strates / grappes pour autant qu'elles soient présentes dans le jeu de données, on obtient des estimations généralement plus précises).
- On peut aussi tout à fait utiliser des fonctions de R sans passer par ces deux packages et dans un premier temps cela facilite l'intégration entre manipulations de données et descriptions statistiques tel que nous l'avons réalisé jusqu'à présent. C'est ce qui est présenté dans ce chapitre.

Décrivons rapidement le jeu de données hdv2003 :

```
str(hdv2003)
```

Le fichier contient 20 variables, dont id qui est un identifiant des observations et poids qui est la variable de pondération (occup est le statut par rapport à l'emploi, qualif est la PCS, clso est le sentiment d'appartenance à une classe sociale).

Exercice

Quelles sont les variables quantitatives du jeu de données (à part id et poids) ?

4.2 Les données pondérées avec Hmisc & questionr

4.2.1 Résumer une variable quantitative avec les nombres de Tukey

Pour obtenir les 5 nombres de Tukey d'une variable quantitative, il faut tenir compte des poids pour calculer le Q1, la médiane et le Q3, ce qu'on peut faire avec la variable wtd.quantile du package Hmisc. Ici, on étudie la distribution du nombre d'heures de visionnage de la télévision en fonction du sexe.

```
hdv2003 |>
  group_by(sexe) |>
  summarise(
    min    = min(heures.tv, na.rm = TRUE),
    q1     = wtd.quantile(heures.tv,weights=poids, probs=0.25, na.rm = TRUE),
    median = wtd.quantile(heures.tv, weights=poids,probs=.5, na.rm = TRUE),
    q3     = wtd.quantile(heures.tv,weights=poids,probs= 0.75, na.rm = TRUE),
    max    = max(heures.tv, na.rm = TRUE)
  )
```

L'argument weights tient bien compte du poids de pondération.

Si on veut inverser les lignes et les colonnes on pourra avoir recours à cette petite transformation :

```
hdv2003 |>
  group_by(sexe) |>
  summarise(
    min    = min(heures.tv, na.rm = TRUE),
    q1     = wtd.quantile(heures.tv,weights=poids, probs=0.25, na.rm = TRUE),
    median = wtd.quantile(heures.tv, weights=poids,probs=.5, na.rm = TRUE),
    q3     = wtd.quantile(heures.tv,weights=poids,probs= 0.75, na.rm = TRUE),
    max    = max(heures.tv, na.rm = TRUE)
  ) |>
  pivot_longer(min:max) |> pivot_wider(names_from=sexe,values_from=value) |>
  kbl() |> kable_classic_2(full_width=F)
```

4.2.2 L'histogramme pondéré

Pour obtenir un histogramme en tenant compte de la pondération, il faut ajouter l'argument weight dans l'aes.

```
ggplot(hdv2003, aes(x = age,y = ..density.., weight = poids)) +
  geom_histogram(binwidth = 5, fill = "grey", color = "white") +
  labs(title = "Histogramme pondéré de l'âge", x = "Âge", y = "Densité")+
  theme_classic()
```

Si on préfère les effectifs pondérés :

```
ggplot(hdv2003, aes(x = age, weight = poids)) +
  geom_histogram(binwidth = 5, fill = "grey", color = "white") +
  labs(title = "Histogramme pondéré de l'âge", x = "Âge", y = "Effectifs pondérés")+
  theme_classic()
```

4.2.3 La boîte à moustaches pondérée

On peut manuellement calculer les seuils du boxplot et les indiquer à ggplot :

```
box_stats <- hdv2003 %>%
  group_by(sexe) %>%
  summarise(
    ymin  = wtd.quantile(age, poids, probs = 0.25)-1.5*(wtd.quantile(age, poids, probs = 0.5)
    lower = wtd.quantile(age, poids, probs = 0.25),
    middle= wtd.quantile(age, poids, probs = 0.50),
    upper = wtd.quantile(age, poids, probs = 0.75),
    ymax  = wtd.quantile(age, poids, probs = 0.75)+1.5*(wtd.quantile(age, poids, probs = 0.5)
  )
ggplot(box_stats, aes(x = sexe,color=sexe)) +
  geom_boxplot(
    aes(
      ymin = ymin,
      lower = lower,
      middle = middle,
      upper = upper,
      ymax = ymax
    ),
    stat = "identity"
  ) +
```

```
  theme_classic()
#Note 1 : par défaut dans geom_boxplot, on peut aussi ajouter l'argument weight.
#Note 2 : Le graphique serait légèrement différent au niveau des moustaches car si Q1-1.5*IQ
```

4.2.4 Résumer avec la moyenne et l'écart-type

Si on veut plutôt résumer une variable avec la moyenne et l'écart-type, on peut écrire, pour étudier l'âge en fonction du sexe :

```
hdv2003 |>
  group_by(sexe) |>
  summarise(
    mean=wtd.mean(age,weights=poids,na.rm=T),
    sd=sqrt(wtd.var(age,weights=poids,na.rm=T))
  ) |>
  pivot_longer(mean:sd) |> pivot_wider(names_from=sexe,values_from=value) |>
  kbl(digits=1) |> kable_classic_2(full_width=F)
```

4.2.5 Recoder une variable quantitative avec les quantiles pondérés et regarder sa distribution

Si on veut découper une variable quantitative en tranches de quantiles, il faut tenir compte de la pondération encore une fois. L'interface icut() de questionr ne permet pas de prendre en compte les quantiles pondérés. Ainsi, il faudra avoir recours à un recodage avec le code directement :

```
hdv2003$age_rec <- cut(hdv2003$age,
                       include.lowest = TRUE,
                       right = FALSE,
                       dig.lab = 4,
                       breaks = wtd.quantile(hdv2003$age,weights=hdv2003$poids,probs=c(0,.25
)
```

On pourra ensuite vérifier la distribution des tranches d'âge avec les fonctions de questionr :

```
hdv2003 |> freqtable(age_rec,weights=poids) |> freq()
```

La fonction freqtable() accepte bien une variable de pondération. À noter que les N des effectifs n'ont ici pas beaucoup de sens : ils correspondent à des effectifs pondérés.

4.2.6 La corrélation en tenant compte de la pondération

Le package descriptio de Nicolas Robette propose une fonction weighted.cor() qui permet de tenir compte de la pondération grâce à l'argument weights :

```
weighted.cor(hdv2003$age,hdv2003$heures.tv,weights=hdv2003$poids,na.rm=T)
```

Pour calculer la corrélation pondérée sur des sous-groupes (en gardant le coefficient et la p-valeur) :

```
hdv2003 |>
 group_by(sexe) |>
  reframe(
    cor = weighted.cor(age, heures.tv, weights = poids,na.rm=T)
  )
```

Pour une matrice des corrélations :

```
hdv2003 |>
  select(age,freres.soeurs,heures.tv) |>
  weighted.cor2(weights=hdv2003$poids,na.rm=T) |>
  round(2)
```

4.2.7 Le tableau croisé et le diagramme à barres pondéré

Pour réaliser un tableau croisé sur données pondérées, rien de plus simple, il suffit d'ajouter l'argument weights à freqtable :

```
hdv2003 |> freqtable(nivetud,cinema,weights=poids) |> rprop()
```

Si on veut ignorer la modalité NA :

```
hdv2003 |> freqtable(nivetud,cinema,weights=poids,na.rm = T) |> rprop()
```

> **Bonus**
>
> Pour créer un diagramme à barres de la proportion de "Oui" suivant le niveau d'études :

```r
#D'abord, stocker le tableau croisé des % en ligne dans l'objet tab
tab <- hdv2003 |>
  freqtable(nivetud, cinema, weights = poids, na.rm = TRUE) |>
  rprop(total=F) #ici on a indiqué qu'on ne souhaitait pas le total

#On transforme notre tab en un objet tidy pour le ggplot:
df_tab <- as.data.frame.matrix(tab) |> #cela permet de transformer la matrice en data frame
  rownames_to_column("nivetud") |> #on remet le niveau d'études comme un nom de colonnes
  mutate(nivetud=factor(nivetud,levels=levels(hdv2003$nivetud))) #cette manipulation permet

ggplot(df_tab, aes(x = nivetud, y = Oui)) + #les % sont stockés dans Oui.
  geom_bar(stat = "identity") +
  geom_text(aes(label = sprintf("%0.f%%", Oui)),#On ajoute les labels au dessus de chaque b
            hjust = -0.1, size = 3.5) +
  ylim(c(0,67))+ #On s'assure que les labels de % seront bien sur le graphique
  coord_flip()+ #On inverse l'axe des x et des y.
  #On crée des jolis labels pour les axes
  labs(
    x = "Niveau d'études",
    y = "Proportion (%)",
    title = "Sortie au cinéma selon le niveau d'études"
  ) +
  theme_classic()
```

Sur un tableau croisé où il y a plus de deux colonnes, les commandes pour obtenir le tableau croisé sont les mêmes :

```r
hdv2003 |>
  freqtable(age_rec, occup, weights = poids, na.rm = TRUE) |>
  rprop()
```

Pour construire un graphique avec ggplot, c'est un peu plus compliqué :

```r
tab<-hdv2003 |>
  freqtable(age_rec, occup, weights = poids, na.rm = TRUE) |>
  rprop(total=F)
df_tab <- as.data.frame.matrix(tab) |>
  rownames_to_column("age_rec") |>
  #il faut remettre tous les pourcentages dans une seule colonne pour ggplot
  pivot_longer(-age_rec,names_to="occup",values_to="prop") |>
  mutate(age_rec=factor(age_rec,levels=levels(hdv2003$age_rec)),
         occup=factor(occup,levels=levels(hdv2003$occup))
  )

ggplot(df_tab, aes(x = age_rec, y = prop, fill=occup)) +
  geom_bar(stat = "identity",position="stack") +
  #on ajoute des labels qui ne seront montrés que s'ils sont supérieurs à 5%
  geom_text(
    aes(label = ifelse(prop > 5, sprintf("%0.0f%%", prop), "")),
    position = position_stack(vjust = 0.5),
    color = "white",
    size = 3
  ) +
  #On a choisi une palette de couleurs
  scale_fill_brewer(palette="Dark2")+
  coord_flip()+
  labs(
    x = "Âge",
    y = "Proportion (%)",
    fill="Statut d'occupation"
  ) +
  theme_classic()+
  theme(legend.position="bottom")
```

Pour le diagramme à barres adjacents :

```
ggplot(df_tab, aes(x = occup, y = prop, fill=age_rec)) +
  geom_bar(stat = "identity",position=position_dodge(width = 0.9)) +
  geom_text(
    aes(label = sprintf("%0.0f%%", prop)),
    position = position_dodge(width = 0.9),
    vjust=-.5,
    color = "black",
    size = 3
  ) +
  scale_fill_brewer(palette="Oranges")+
  labs(
    x = "Statut d'occupation",
    y = "Proportion (%)",
    fill="Âge"
  ) +
  theme_classic()+
  theme(legend.position="bottom")
```

4.2.8 Le test du chi-deux et le V de Cramer avec pondération

Le test du chi-deux est sensible à la taille de l'échantillon. Or si on crée simplement un tableau des effectifs pondérés, on "grossit" artificiellement la taille de l'échantillon en utilisant chisq.test.

On pourra avoir recours à la fonction assoc.twocat() dans le package descriptio qui rassemble de nombreuses mesures d'association entre deux variables catégorielles.

- À noter que le test du chi-2 est ici réalisé à partir d'un "test de permutation" (plutôt qu'un test fréquentiste, il faut donc choisir un nombre de permutations avec l'argument nperm, ici 100 mais choisir plutôt 1000).
- À noter également que dans la fonction il est conseillé d'utiliser des poids "normalisés" à la taille de l'échantillon (c'est-à-dire dont la somme est égale au nombre d'individus enquêtés) pour ne pas distordre le calcul du chi-2.

```
hdv2003$poidsn<- length(hdv2003$poids) * hdv2003$poids / sum(hdv2003$poids)
sum(hdv2003$poidsn)
tri<-assoc.twocat(hdv2003$age_rec,hdv2003$occup, weights=hdv2003$poidsn,nperm=100)
tri$global$chi.squared
tri$global$permutation.pvalue
```

Pour obtenir le V de Cramer :

```r
weighted.cramer(hdv2003$age_rec, hdv2003$occup, weights = hdv2003$poids)
#Ou simplement :
tri$global$cramer.v
```

4.3 Exercices

Exercice

Pour tous les exercices suivants, utiliser la pondération du jeu de données.

1. Créer une nouvelle variable de diplôme en rassemblant de manière logique les modalités de nivetud. Vérifier la distribution de cette nouvelle variable en la croisant avec nivetud.
2. Calculer les 5 nombres de Tukey et réaliser des boîtes à moustaches sur heures.tv selon cette nouvelle variable du niveau d'études. Observe-t-on des différences notables ?
3. Recoder la variable heures.tv en 3 modalités : la première comprend les individus qui ne regardent jamais la tv, la seconde ceux qui la regardent moins que la médiane (parmi les individus qui la regardent) et la troisième ceux qui la regardent plus que la médiane (parmi les individus qui la regardent). Nommer les modalités construites : Rien, Faible, Elevé.
4. À partir de cette variable, étudier la fréquence de visionnage de la télévision suivant le sexe, l'âge recodé en quartiles et le niveau de diplôme. Réaliser des tests de significativité Si vous le souhaitez, réaliser des diagrammes à barres empilées.

5 Les relations multivariées entre variables

Nous avons jusqu'ici analysé des relations bivariées (deux variables prises deux à deux). Bien souvent, on est amené à s'intéresser à des relations multivariées.

On peut alors être tenté d'avoir recours, lorsqu'on a une *variable dépendante* et plusieurs *variables indépendantes*, à des modèles de régression. On trouvera dans le guide-R de Joseph Larmarange quelques pistes pour mettre en place ce type de modèle et le raisonnement "toutes choses égales par ailleurs".

Toutefois, il est aussi possible de mettre en place des relations multivariées sans ce type de modélisation.

Dans ce chapitre, nous mobilisons les mêmes packages et les mêmes données que dans le chapitre précédent des données pondérées.

Exercice

- Créer un script vide.
- Enregistrer ce script dans un dossier (par exemple "Formation R") en le nommant par exemple 5statmultis.R.
- Charger les packages mobilisés dans le chapitre précédents (tidyverse, questionr, kableExtra a minima).
- Installer et charger également les packages Hmisc, FactoMineR, explor qui contiennent quelques fonctions utiles pour pondérer des statistiques et réaliser des analyses multivariées.
- Pour le package descriptio (de Nicolas Robette), on veillera à installer la dernière version sur son repository grâce aux commandes présentées dans le chapitre précédent.
- Charger la base de données Histoire de vie grâce à la commande data("hdv2003").

5.1 Le tidyverse pour le multivarié sur variable quantitatives

Les outils du tidyverse permettent assez facilement d'analyser des relations multivariées lorsqu'on s'intéresse à une variable quantitative en fonction de plusieurs variables qualitatives.

Si on veut la distribution de l'âge en fonction de l'intensité de la pratique religieuse et du sexe (en réalité, on suppose ici que l'âge et le sexe influencent l'intensité de la pratique…), on pourra écrire :

```
hdv2003 |>
  group_by(sexe,relig) |>
  summarise(
    mean=wtd.mean(age,weights=poids,na.rm=T),
    sd=sqrt(wtd.var(age,weights=poids,na.rm=T))
```

Si on veut en faire un joli tableau avec kableExtra :

```
tab <- hdv2003 |>
group_by(sexe, relig) |>
summarise(
  mean = wtd.mean(age, weights = poids, na.rm = TRUE),
  sd = sqrt(wtd.var(age, weights = poids, na.rm = TRUE)),
  .groups = "drop" #on enlève l'option de grouping présente dans l'objet créé
) |>
# Commandes pour présenter une cellule rassemblant moyenne et sd : moyenne (sd)
mutate(mean_sd = sprintf("%.1f (%.1f)", mean, sd)) |>
select(sexe, relig, mean_sd)

# Tidy le tableau
tab_wide <- tab |>
  pivot_wider(names_from = sexe, values_from = mean_sd)

#Joli tableau
tab_wide |>
  kbl(col.names = c("Religions", "Homme","Femme"), align = "lcc") |>
  kable_classic_2(full_width = FALSE)
```

Imaginons qu'on veuille les 5 nombres de Tukey :

```r
tab<-hdv2003 |>
  group_by(sexe,relig) |>
  summarise(
    min    = min(age, na.rm = TRUE),
    q1     = wtd.quantile(age,weights=poids, probs=0.25, na.rm = TRUE),
    median = wtd.quantile(age, weights=poids,probs=.5, na.rm = TRUE),
    q3     = wtd.quantile(age,weights=poids,probs= 0.75, na.rm = TRUE),
    max    = max(age, na.rm = TRUE),
    .groups="drop"
  )

tab |>
  select(relig,min:max) |>
  kbl(digits=1,
      col.names = c("Religion", "Min", "Q1", "Médiane", "Q3", "Max"),
      align = c("l", rep("c", 5))
      ) |>
  kable_classic_2(full_width = FALSE) |>
  pack_rows("Homme", 1, 6) %>%
  pack_rows("Femme", 7, 12)
```

Religion	Min	Q1	Médiane	Q3	Max
Homme					
Pratiquant regulier	18	27	51	66	89
Pratiquant occasionnel	18	33	48	61	92
Appartenance sans pratique	18	31	47	61	91
Ni croyance ni appartenance	18	28	36	49	87
Rejet	18	26	34	47	83
NSP ou NVPR	27	33	41	62	77
Femme					
Pratiquant regulier	18	40	60	72	90
Pratiquant occasionnel	18	36	51	62	97
Appartenance sans pratique	19	31	44	57	90
Ni croyance ni appartenance	18	26	35	46	96
Rejet	20	29	39	52	78
NSP ou NVPR	30	38	48	59	83

Figure 5.1: Résumé de la variable d'âge en fonction du sexe et de la pratique religieuse

Si on veut créer des boîtes à moustaches, on peut utiliser les "facet" de ggplot ou ajouter un argument "color" :

```
ggplot(hdv2003, aes(x = fct_rev(relig),y=age,weight=poids)) +
  geom_boxplot() +
  facet_wrap(~sexe)+
  labs(x="Pratique religieuse",y="Âge")+
  coord_flip()+
  theme_classic()
ggplot(hdv2003, aes(x = fct_rev(relig),y=age,weight=poids,color=sexe)) +
  geom_boxplot() +
  coord_flip()+
  labs(x="Pratique religieuse",y="Âge",color="Sexe")+
  scale_color_brewer(palette="Set1")+
  theme_classic()+
  theme(legend.position="bottom")
#Ici fct_rev sert à inverser l'ordre des modalités de la variable relig
```

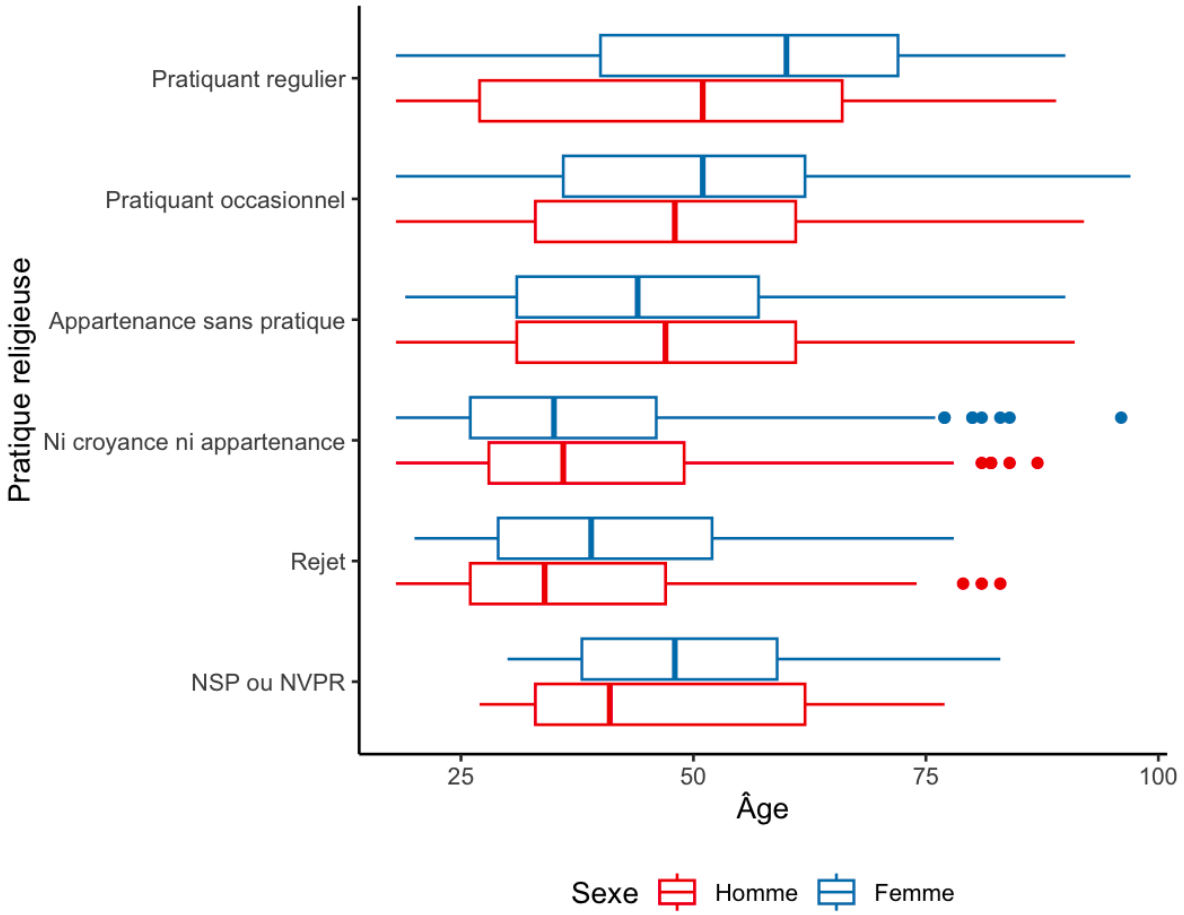


Figure 5.2: Boîtes à moustaches de l'âge en fonction du sexe et de la pratique religieuse

5.2 Le tableau croisé multi-dimensions

Nous avons précédemment étudié la pratique d'aller au cinéma en fonction du niveau de diplôme. L'effet du diplôme varie-t-il suivant le sexe ?

On peut à nouveau réaliser un tableau croisé avec pourcentages en lignes, cette fois en mobilisant group_by. Attention, pour obtenir un tableau croisé du cinema en fonction du diplôme pour les hommes d'une part et les femmes d'autre part, le plus simple est d'écrire ceci :

```
hdv2003 |> group_by(nivetud,cinema) |> freqtable(sexe,weights=poids,na.rm = T) |> rprop()
```

Pour faire un joli tableau :

```
tab<-hdv2003 |> group_by(nivetud,cinema) |> freqtable(sexe,weights=poids,na.rm = T) |> rprop
df_tab<-as.data.frame(tab) |>
  pivot_wider(names_from=cinema,values_from=Freq)
df_tab |>
  select(nivetud,Non,Oui,Total) |>
  kbl(digits=0,
      col.names = c("Niveau d'études","Oui","Non","Total")) |>
  kable_classic_2(full_width = FALSE) |>
  pack_rows("Homme", 1, 9) %>%
  pack_rows("Femme", 10, 18)
```

Cette syntaxe n'est, avouons-le, pas des plus intuitives. On pourra si on le souhaite préférer s'appuyer sur le package descriptio et sur la fonction assoc.twocat.by() :

```
tri3<-assoc.twocat.by(hdv2003$nivetud, hdv2003$sport, hdv2003$sexe,na.rm=T)
tri3$tables$rprop

#Pour en faire un joli tableau :
t <- as.data.frame(tri3$tables$rprop)

# Construction du tableau
t |>
  kbl(col.names = NULL,digits = 0) |>
  add_header_above(c("Diplôme" = 1, "Oui" = 1, "Non" = 1, "Total" = 1,
                     "Oui" = 1, "Non" = 1, "Total" = 1)) |>
  add_header_above(c(" " = 1, "Homme" = 3, "Femme" = 3)) |>
  kable_classic_2(full_width = FALSE)
```

Pour faire un diagramme à barres :

```
tab<-hdv2003 |> group_by(nivetud,cinema) |> freqtable(sexe,weights=poids,na.rm = T) |> rprop
df_tab<-as.data.frame(tab)
ggplot(df_tab,aes(x=nivetud,y=Freq,fill=cinema))+
  geom_bar(stat = "identity")+
  facet_wrap(~sexe)+
  labs(y="Proportion (%)",x="Diplôme",fill="Est allé au cinéma")+
  coord_flip()+
  theme_classic()+
  theme(legend.position="bottom")
```

5.3 L'analyse géométrique des données : le cas de l'ACM

Lorsqu'on cherche à étudier de multiples associations entre variables, on peut mobiliser l'analyse géométrique des données, et notamment l'Analyse des Correspondances Multiples (ACM).

On présente ici rapidement comment mettre en oeuvre une ACM avec le package FactoMineR.

5.3.1 Réaliser une ACM avec FactoMineR

On sélectionne d'abord les variables de fréquence de pratique culturelle qui sont déjà codées sous forme catégorielle et deux variables qualitative supplémentaire, le sexe et la PCS (variable qualif).

```
d_acm<- hdv2003 |>
  select(hard.rock,lecture.bd,peche.chasse,cuisine,bricol,cinema,sport,sexe,qualif)
```

On utilise ensuite la fonction MCA pour lancer une ACM :

```
acm<-MCA(d_acm,quali.sup=c(8,9),row.w=hdv2003$poids)
```

L'argument quali.sup permet d'indiquer les colonnes des variables supplémentaires de l'analyse. Par défaut, les autres sont considérées comme des variables actives.

On a veillé à ajouter l'argument row.w qui permet de pondérer les individus dans l'analyse. Cette analyse produit trois graphiques présentés en bas à droite :

- le carré des liaisons sur le premier plan factoriel,
- le nuage des individus sur le premier plan factoriel et
- le nuage des catégories des variables actives et supplémentaires sur le premier plan factoriel.

Il suffit de cliquer sur la flèche gauche pour avoir les différents graphiques.

5.3.2 Explorer les résultats d'une ACM

On peut avoir recours à l'interface explor du package explor de Julien Barnier pour visualiser les résultats de l'ACM :

```
explor(acm)
```

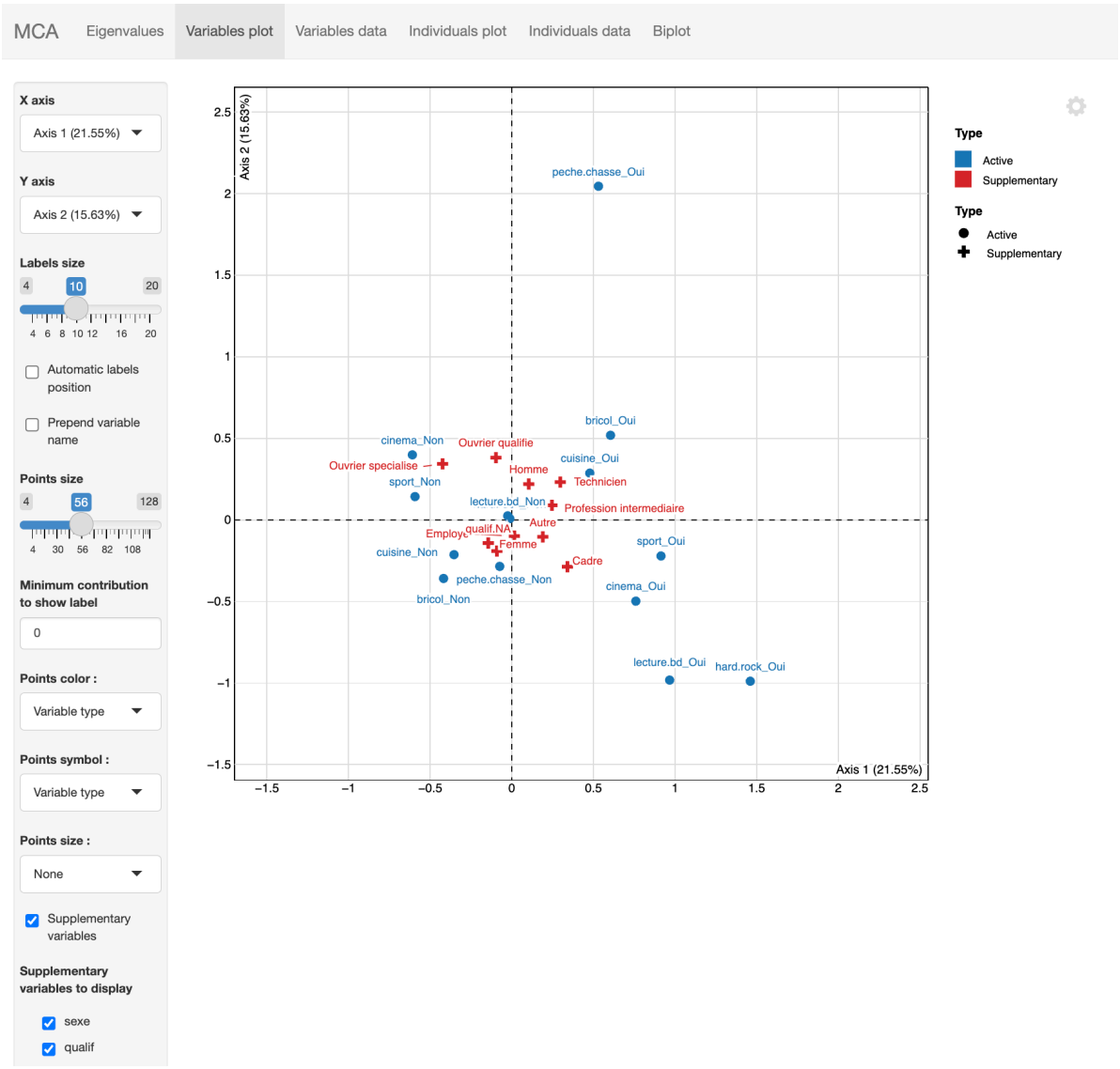


Figure 5.3: Résultats de l'ACM dans explor

5.4 Exercices

Exercice

1. Procédons d'abord à quelques recodages :
    - Recoder la variable diplôme comme dans le chapitre précédent
    - Recoder la variable heures.tv comme dans le chapitre précédent
    - Recoder une variable classe qui permette d'avoir les PCS pour les actifs et le statut d'occupation "inactif" pour les autres
2. Quel est l'effet du diplôme sur la fréquentation du cinéma ? Cet effet varie-t-il suivant l'âge ?
3. Quelle est la répartition de la nouvelle variable classe suivant l'âge pour les hommes et pour les femmes ? Les associations sont-elles significatives ?
4. Existe-t-il un gradient social dans l'intensité du visionnage de la télévision ? Ce gradient social varie-t-il suivant l'âge ?
5. Parmi les individus actifs, quels sont les classes qui ont un sentiment d'appartenance le plus élevé (clso) ? Cet effet varie-t-il suivant l'âge ?
6. Réaliser une ACM en ajoutant parmi les variables actives l'intensité du visionnage de la télévision (variable catégorielle). Parmi les variables supplémentaires, ajouter l'âge recodé, la classe et le diplôme recodé. Dans cette ACM, on se concentrera sur les individus actifs. Faire attention à bien inclure la variable de pondération. Interpréter.

6 Organisation du code et manipulations avancées

Ce chapitre vise surtout à synthétiser quelques trucs et astuces pour se faciliter la vie dans l'utilisation de R.

6.1 La notion de "projet" dans RStudio

Tout au long de ces chapitres, nous avons créé différents scripts R rassemblés dans un dossier "Formation R". Une bonne pratique est de créer un "projet" pointant vers le dossier des scripts qui ont une cohérence ensemble (un projet de recherche par exemple), les données afférentes, etc.

Ce dossier projet peut être créé en cliquant en haut à droite sur "Project: (None)", puis "New Project" et ici "Existing Directory" (on sélectionnera alors le dossier Formation R.

Ce dossier peut contenir plusieurs sous-dossiers :

- Script
- Data
- Output (là où éventuellement on rassemble ses graphiques et autres jolis tableaux)

L'intérêt de fonctionner par projet est que RStudio va ensuite automatiquement pointer vers le dossier racine (Formation R) quand le projet est ouvert et on accèdera plus facilement à ses fichiers dans le Files et en écrivant le chemin d'accès dans ses scripts.

6.2 Organiser son code et ses fichiers

Il est assez courant de commencer un projet et d'y revenir des mois voire des années plus tard.

Que faire quand on a tout oublié ?

Si on a bien structuré son code, ce qu'on peut faire en ajoutant des sections, dans son script, on s'y retrouve plus facilement :

```
# Ceci marque une section ------------------------------
```

On peut aussi commenter allégrement ses différentes opérations pour s'y retrouver :

```
#Le dièse permet d'initier un commentaire
# Quand on doit faire un long commentaire,
# on peut écrire sans dièse, puis surligner le tout,
# et taper Cmd (ou Ctrl)+Maj+C et le bloc sera mis en
# commentaire !
```

Il est aussi assez courant de créer différents scripts pour différents types d'opérations (mais ça dépend de ses habitudes), par exemple :

- Un script ou des script(s) où je réalise tous les recodages nécessaires à l'analyse et j'enregistre ma base avec les nouvelles variables (en format R, rds plutôt).
- Un ou des script(s) où je réalise toutes les opérations d'analyse.

On pourra si on le souhaite créer différents fichiers suivant l'avancement du projet et indiquer la date dans le titre. Ainsi, le fichier de recodage pourra s'appeler "20250602Recodages.R" (la date est mise dans le format AAAAMMJJ pour être sur que les fichiers soient présentés du plus ancien au plus récent dans le dossier où ils sont stockés.

6.3 Obtenir de l'aide

Listons ici quelques manières d'avoir de l'aide quand le script buggue :

- Le premier réflexe est de chercher comment s'utilise une fonction dans R :

```
?mean #le point d'interrogation ouvre la page de description de la fonction
```

- On peut aussi taper son problème dans son moteur de recherche, qui renvoie souvent vers le forum Stackoverflow (mais pas que) où le problème auquel on fait face a été discuté (généralement, en anglais)
- On peut consulter les guides de formation à R mentionnés sur la page de présentation de ce guide.
- …

6.4 La jointure de différents fichiers statistiques

Nous n'avons pas ici parlé de la jointure de différents fichiers statistiques, opération pourtant courante. Cette page de utilitR est très complète pour mener ces opérations sans trop se prendre la tête : https://book.utilitr.org/03_Fiches_thematiques/Fiche_joindre_donnees.html.

6.5 Les étiquettes de variables et de valeurs

Enfin, nous avons ici travaillé à partir de variables qui n'ont pas d'étiquettes et dont les modalités des variables catégorielles n'ont pas de labels. C'est pourtant possible (voire courant, si on charge un fichier Stata dans R) et plutôt pratique :

- On a ainsi des étiquettes déjà prêtes pour les tableaux / graphiques
- On va bien plus vite dans le recodage et surtout on évite les erreurs (recoder 1 en 3 va plus vite et comprend moins de risque d'erreur que recoder "Cadres et professions scientifiques" en "Classes supérieures").

On se reportera aux Chapitres 11 et 12 de guide-R qui introduisent à ces concepts et leur usage.

6.6 Exercices

> Exercice
>
> Créer un Projet pointant sur le dossier de la Formation R. Commenter et créer des sections dans l'un de ses scripts. Pas facile ? D'où l'intérêt de le faire au fur et à mesure :)

Part IV

Analyse statistique textuelle

7 Créer un corpus à partir d'Europresse

Dans ce chapitre, nous allons construire un corpus d'articles de presse à partir du moteur de recherche Europresse.

On peut suivre les instructions présentées dans les slides ci-dessous.

À noter que cette méthode est inspirée d'un billet du carnet Hypotheses "QUANTI" écrit par Corentin Roquebert. On peut donc tout à fait réaliser la mise en forme de la base de données des articles directement dans R (modulo le fait que le code proposé n'est pas totalement à jour vue l'évolution des packages utilisés).

8 Quelques outils textométriques dans R

Ce chapitre propose une brève introduction à l'analyse d'un corpus de textes, ici constitué a priori d'articles collectés sur Europresse (voir chapitre précédent).

Si on n'a pas de corpus, on peut utiliser un corpus utilisé pour l'exemple ici portant sur l'utilisation du terme "wokisme" dans les médias de la presse nationale.

À noter que la plupart des opérations présentées ici peuvent aussi être réaliser avec une application "clic-bouton" appelée Mendak et disponible ici ou ici (préférer a priori le deuxième lien : pour comprendre comment fonctionne Mendak - et l'analyse statistique textuelle surtout -, on pourra se reporter à ce tutoriel).

Un tutoriel de la manipulation des données textuelles dans R est également présenté en anglais ici.

Exercice

- Créer un script vide.
- Enregistrer ce script dans un dossier (par exemple "Formation R") en le nommant par exemple 7textometrie.R.
- Installer/Charger les packages suivants : tidyverse, questionr, kableExtra, esquisse, FactoMineR, explor, factoextra, ggrepel, rainette, quanteda, quanteda.textstats, quanteda.textplots.
- Charger la base de données dans un objet nommé textes (voir chapitre 1 - Introduction pour ce faire).

```
# On liste les packages dont on a besoin dans un vecteur nommé load.lib.
load.lib <- c("openxlsx","tidyverse", "questionr", "kableExtra", "esquisse", "FactoMineR",
install.lib <- load.lib[!load.lib %in% installed.packages()] # On regarde les paquets qui n
for (lib in install.lib) install.packages(lib,dependencies=TRUE) # On installe ceux-ci
sapply(load.lib,require,character=TRUE) # Et on charge tous les paquets
```

8.1 L'exploration du corpus

Exercice

- Commencer par décrire la base de données.
- Puis, réaliser des tris à plat des variables NomSource, TypeSource, PaysSource, PeriodiciteSource, LangueSource.
- Si certains textes du corpus ne sont pas français, créer un textes2 où on retire les textes correspondants.

Pour étudier la date de publication des articles, on dispose de la variable Date : "03 mai 2024", "26 avril 2023"... soit un format JJ mois AAAA, ce qu'on peut indiquer à R en utilisant le format "Date" :

```
#ATTENTION, pour que la fonction suivante fonctionne il va peut-être falloir préciser qu'on e
#- sur mac ou linux
Sys.setlocale("LC_TIME", "fr_FR.UTF-8")
#- sur windows
Sys.setlocale("LC_TIME", "French_France.1252")

# Conversion des dates en format Date
textes$Date_propre <- as.Date(textes$Date, format = "%d %B %Y")

#Si on veut par exemple extraire juste l'année :
# Extraire l'année
textes$Annee <- format(textes$Date_propre, "%Y")

#Une autre alternative pour extraire l'année à partir de Date est d'utiliser str_extract avec
textes$Annee <- str_extract(textes$Date, "[0-9]{4}$")
# "[0-9]{4}$" = extrais 4 chiffres entre 0 et 9 en fin de chaine.
```

Exercice

Avec esquisse, étudier la distribution de parution des articles suivant la date de parution. La distribution est-elle différente suivant le titre de presse ?

```
esquisser(textes)
ggplot(textes) +
  aes(x = Date_propre) +
  geom_histogram(bins = 30L, fill = "#000000") +
  labs(x = "Période", y = "Nombre d'articles") +
```

```
  theme_light() +
  theme(
    axis.title.y = element_text(size = 18L),
    axis.title.x = element_text(size = 18L),
    axis.text.y = element_text(size = 18L),
    axis.text.x = element_text(size = 14L),
    legend.text = element_text(size = 18L),
    legend.title = element_text(size = 18L),
    strip.text = element_text(size = 18L) #Augmenter taille des labels des facets
  ) +
  facet_wrap(vars(NomSource))
```

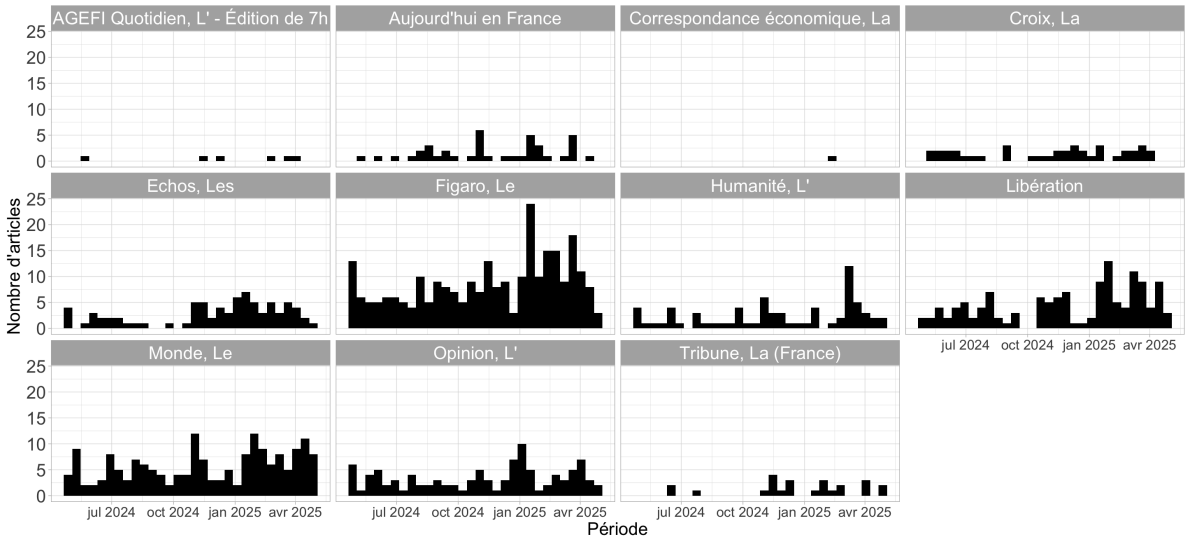


Figure 8.1: Distribution des articles suivant la période par titre de presse

On va créer une variable de comptage du nombre de mots dans le corps de textes des articles. La première étape est de transformer la colonne correspondante en format “corpus” avec le package quanteda :

```
#On s'assure que tous les types d'apostrophes sont bien remplacés par des espaces,
#Car quanteda le gère assez mal
textes$Textec = str_replace_all(textes$Texte, "['’‘` ɔc ˘'’']", " ")
corpus <- corpus(textes$Textec)
```

Le corpus est un ensemble de textes — ici, les 938 corps de textes des articles.

À partir de cet objet, on peut créer des " tokens ", c'est-à-dire les mots contenus dans les textes. Ce processus s'accompagne d'un léger nettoyage en vue de l'analyse :

- On supprime la ponctuation (points, virgules, etc.) ainsi que les symboles (par exemple : * $ €…) afin d'éviter de les considérer comme des mots.
- On supprime aussi les nombres écrits en chiffres (et non en toutes lettres). Un autre choix aurait pu être fait, mais il me semble raisonnable de les exclure ici.

```
tok <- tokens(corpus, remove_punct = TRUE,
              remove_symbols=TRUE,
              remove_numbers =TRUE)
```

On peut ainsi créer une variable du nombre de mots par corps d'articles dans la base :

```
textes$nmots <- ntoken(tok)
```

> Exercice
>
> La distribution du nombre de mots varie-t-elle suivant le titre de presse ?

8.2 Le nettoyage des données

On retire un certain nombre de "mots vides " (stop words) de l'analyse. Il s'agit de mots ou expressions très fréquents en français, qui apportent peu d'information sur le contenu des textes. Leur suppression permet de faire ressortir les éléments lexicaux réellement caractéristiques.

On utilise pour cela une liste proposée par Gilles Bastin qu'on va télécharger (ça permet de voir qu'il est possible de charger directement un fichier avec un url dans R !) :

```
stopwords_url <- "https://raw.githubusercontent.com/gillesbastin/french_stopwords/main/frenc
stop_fr <- read_csv2(stopwords_url)
```

On peut aussi enlever les mots liés à notre recherche sur le wokisme. En effet, l'idée n'est pas de se concentrer sur la présence du terme « wokisme » lui-même ou de ses variantes, mais plutôt sur ce qui est dit *à propos* du wokisme — c'est-à-dire les jugements, qualificatifs, oppositions, cadrages ou champs sémantiques associés

```
mot_woke<-c("wokisme","woke","wokiste")
tok <- tokens_remove(tok,c(stop_fr$token,mot_woke))
```

On crée ensuite une “matrice de mots” (une document-feature matrix) :

```
#Create document-feature matrix (keep in lower case)
dtm <- dfm(tok, tolower = TRUE)
dtm
```

La matrice comprend en ligne les textes et en colonnes les mots du corpus. Chaque cellule indique le nombre de fois qu'un mot apparait dans le texte correspondant.

Arbitrairement, ce peut être une bonne idée de ne pas garder les mots trop rares (mettons au moins ceux qui ne sont présents qu'une seule fois). Ici, je garde même ceux qui sont présents plus de 5 fois au total sur l'ensemble du corpus.

```
dtm <- dfm_trim(dtm, min_docfreq = 5)
```

8.3 Le nuage de mots

Après tout ce travail de préparation, on peut passer à une première analyse exploratoire ! Commençons simplement par afficher un nuage de mots dans lequel la taille des mots est proportionnelle à leur fréquence dans le corpus.

```
textplot_wordcloud(dtm, random_order = F, rotation = 0.25,min_size =1,max_words = 100,
                   color = RColorBrewer::brewer.pal(8, "Dark2"))
```

Figure 8.2: Nuage de mots du corpus nettoyé

8.4 L'Analyse Factorielle des Correspondances

La matrice de mots se prête assez bien à une Analyse Factorielle des Correspondances (AFC), en considérant la matrice comme un tableau de contingence.

Il s'agit ici de mener une analyse exploratoire des principales associations récurrentes dans le corpus entre les mots, et si ces associations font ressortir des champs thématiques spécifiques, eux-mêmes associés à des journaux, des dates, des journalistes…

Pour simplifier encore plus et faire en sorte que l'analyse ne prenne pas trop de temps à tourner, je ne garde ici que les mots qui sont présents au moins 20 fois dans le corpus (3605 mots tout de même, et autant de colonnes dans l'analyse, et ensuite de points sur un graphique !).

```
#On ne garde que les termes mentionnés au moins 20 x ... !
dtm_reduc <- dfm_trim(dtm, min_docfreq = 20)
#On convertit la matrice en un data frame
dtm_pour_afc <- convert(dtm_reduc, to = "data.frame")
#On enlève la colonne doc_id, convertie en rownames
rownames(dtm_pour_afc) <- dtm_pour_afc$doc_id
dtm_pour_afc$doc_id <- NULL
#On ajoute trois colonnes de métadonnées sur les textes
dtm_pour_afc<-cbind(NomSource=textes$NomSource,Annee=textes$Annee,dtm_pour_afc)
ca<-CA(dtm_pour_afc,graph=F,quali.sup=1:2)
```

On pourra éventuellement examiner l'éboulis des valeurs propres des axes, dénotant ainsi quelle part de l'information de la matrice est exprimée sur chacun des axes (on pourra n'analyser que les premiers axes, avant une coupure dans le diagramme à barres des valeurs propres).

```
fviz_eig(ca,ncp=20)
```

Pour visualiser les mots sur le premier plan factoriel, il va être difficile de tous les projeter. Il est possible de ne privilégier que les 100 ou 200 mots qui contribuent le plus à ce plan factoriel :

```
fviz_ca_col(ca,select.col=list(contrib=200),axes = c(1, 2), repel = TRUE)
```

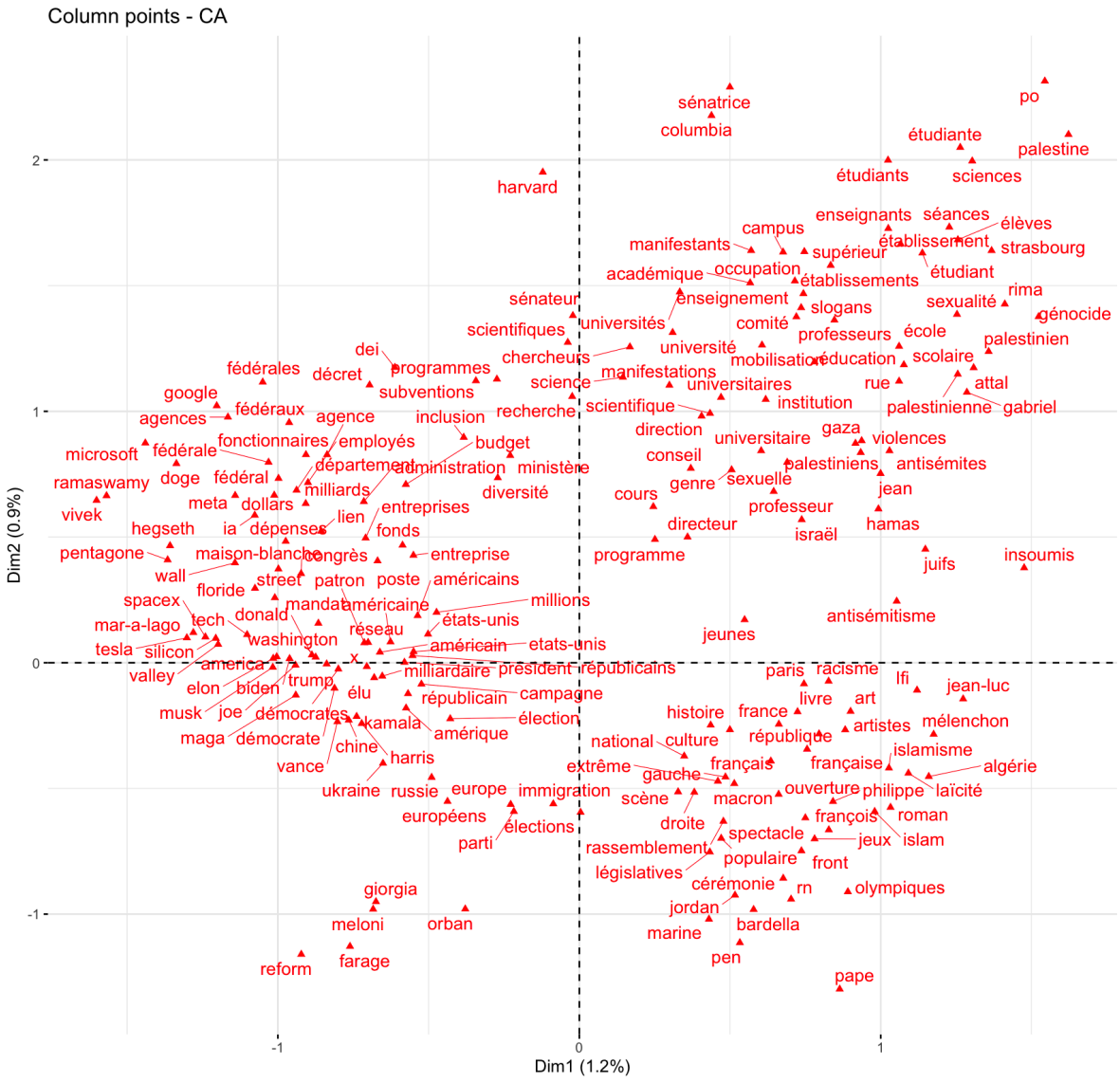


Figure 8.3: 200 mots les plus contributeurs sur le premier plan factoriel

Clairement, le premier axe (horizontal) oppose des articles portant à gauche sur les États-Unis (et l'international) et à droite la France. Le second axe distingue surtout les articles sur la France entre en haut des articles sur l'institution scolaire / universitaire et en bas des articles sur le champ politique à proprement parler (cérémonie renvoie en fait aux jeux olympiques…).

Ces champs thématiques sont-ils associés à des journaux / années ? Malheureusement il n'y pas de fonction clef en main pour représenter les variables catégorielles supplémentaires dans le cadre du AFC avec factoextra…

```r
quali_coord <- as.data.frame(ca$quali.sup$coord)
quali_coord$mod <- rownames(quali_coord)
quali_coord$mod <- str_remove(quali_coord$mod, "^[^\\.]+\\.")


ggplot(quali_coord, aes(x = `Dim 1`, y = `Dim 2`, label = mod)) +
  geom_vline(xintercept = 0, linetype = "dashed")+
  geom_hline(yintercept = 0, linetype = "dashed")+
  geom_point(size = .5) +
  geom_text_repel(size = 4) +
  theme_minimal() +
  labs(title = "Modalités des variables qualitatives supplémentaires",
       x = "Dimension 1", y = "Dimension 2")
```

On repère quand même que les articles sur les Etats-Unis sont plus associés à la presse économique, contre la presse généraliste et les titres de presse connotés dans le champ politique associés à des articles sur la France.

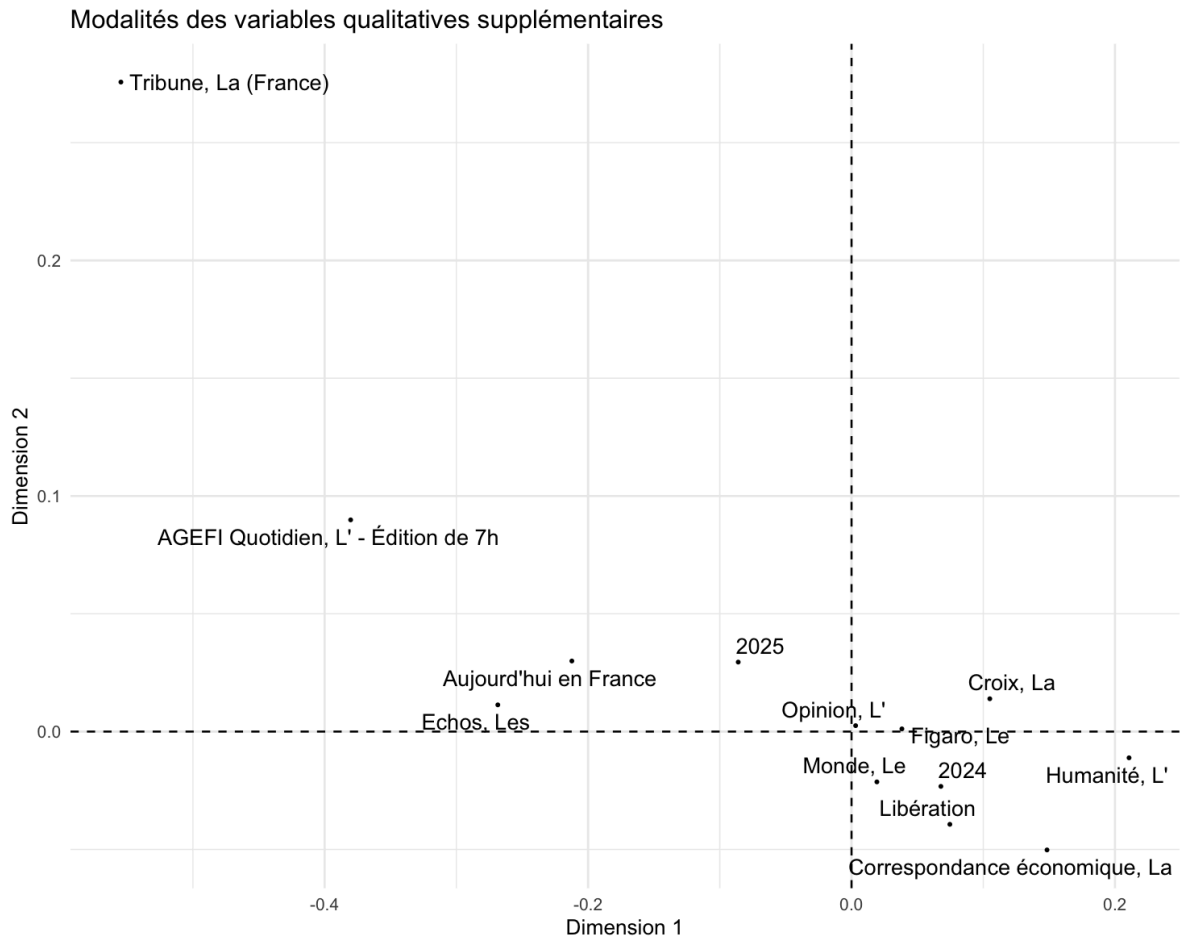


Figure 8.4: Titre et année sur le premier plan factoriel

8.5 La classification de Max Reinert

On mobilise ici le package rainette créé par Julien Barnier qui reprend la méthode de classification de Max Reinert présentée dans Alceste et dans Iramuteq.

À partir de la matrice document-terme, on utilise la distance du khi-deux pour mesurer les similarités et différences entre les descriptions de familles.

La méthode repose sur un algorithme de classification hiérarchique descendante, que l'on peut décrire ainsi :

- On commence par diviser les documents en deux groupes, de manière à ce qu'ils soient aussi différents que possible. Cette différence repose sur le fait que certains mots (tokens)

sont fréquemment mentionnés dans un groupe et rarement dans l'autre, et inversement.

- Parmi les deux groupes obtenus, on identifie le plus grand, que l'on divise à nouveau en deux sous-groupes.
- Parmi les trois groupes ainsi créés, on divise à nouveau le plus grand, ce qui donne quatre groupes.
- Et ainsi de suite…

Une description plus complète et rigoureuse de l'algorithme est disponible ici.

Cette étape est réalisée en exécutant la commande R suivante (ici, on demande à l'algorithme de créer jusqu'à dix groupes de documents) :

```
res <- rainette(dtm, k = 10)
```

Combien de groupes ou de clusters de documents faut-il conserver dans la partition finale ?

Il s'agit ici d'un outil exploratoire, donc la meilleure approche consiste à examiner d'abord la partition en deux groupes, puis celle en trois groupes, etc.

À chaque étape, on interprète la nouvelle distinction qui émerge avec l'ajout d'un cluster.

- On conserve autant de groupes qu'on est capable d'interpréter.
- L'objectif est de comprendre les distinctions dans le corpus : trop de clusters nuisent à la lisibilité de l'analyse.
- En pratique, au-delà de 8 à 10 clusters, la classification devient souvent peu informative, car il devient difficile de saisir d'un coup d'œil les différences entre les groupes.

L'exploration concrète des clusters de documents et des mots (tokens) qui les structurent est réalisée grâce à une application Shiny très pratique :

```
rainette_explor(res, dtm, corpus)
```

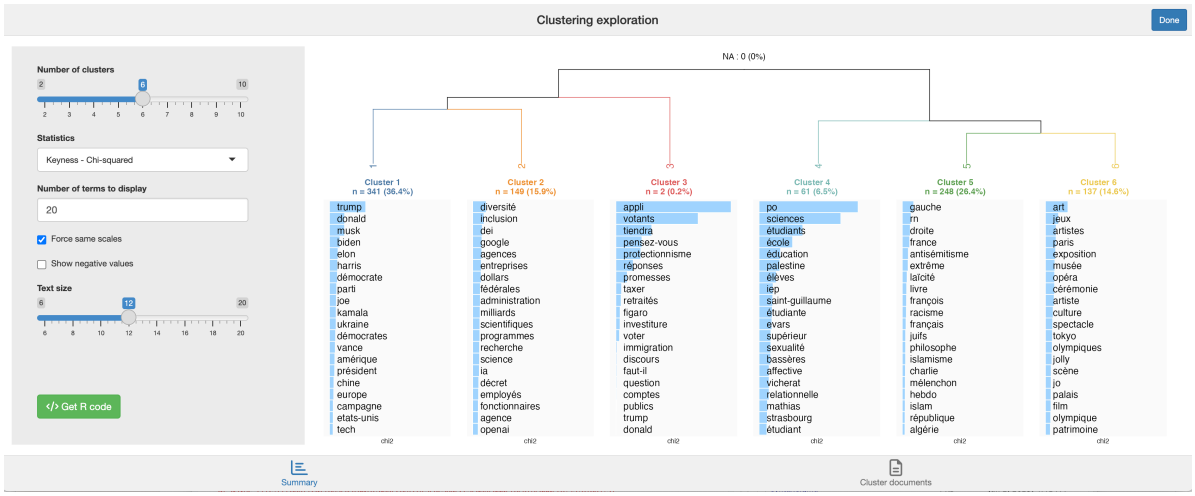


Figure 8.5: Interface de l'explorateur de classification de rainette

Chaque cluster est caractérisé par ses tokens les plus spécifiques, c'est-à-dire, en simplifiant, les mots qui sont souvent mentionnés dans un cluster mais peu présents dans les autres.

Comme il peut être difficile d'interpréter directement la spécificité de chaque cluster, il est utile de revenir aux descriptions textuelles pour voir comment ces termes spécifiques sont utilisés dans les phrases.

Dans le coin inférieur droit du panneau de l'application Shiny, cliquer sur "Documents du cluster" pour explorer ces occurrences.

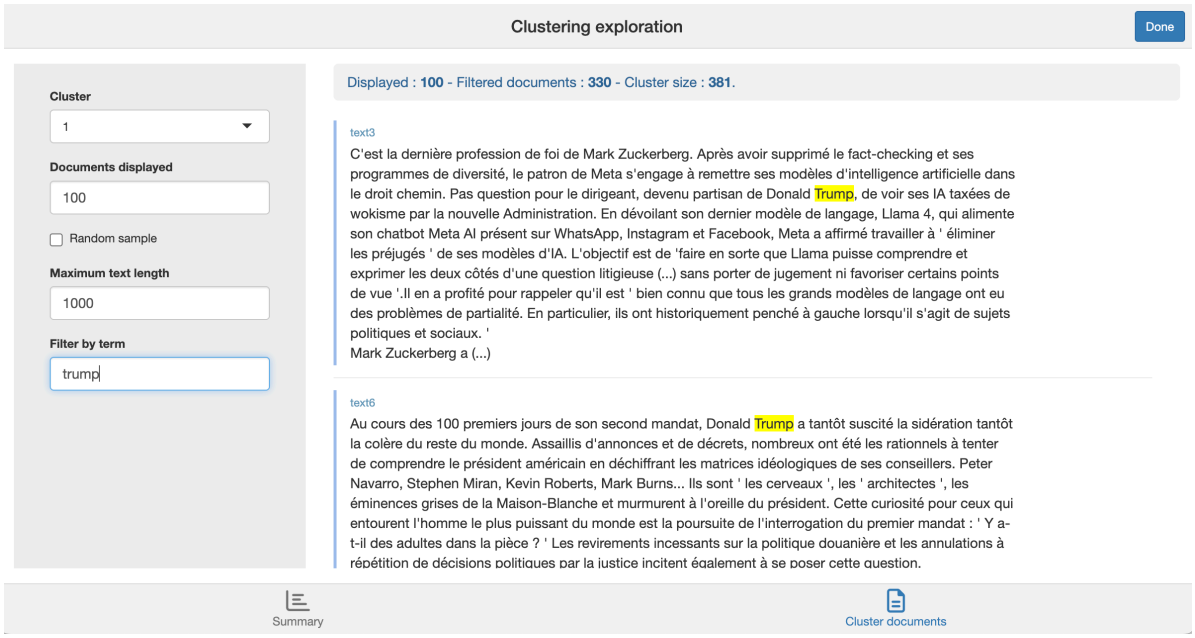


Figure 8.6: Recherche des termes dans les documents

Les clusters obtenus sont-ils associés à certains journaux / période de parution ?

Pour le savoir ajoutons une variable de cluster à la base de données :

```
textes$cluster <- cutree(res, k = 6) #Ou remplacer 6 par le nombre de clusters finalement re
```

Exercice

1. Renommer les clusters avec des labels explicites.
2. Proposer des traitements statistiques permettant d'analyser le lien entre les champs thématiques et les caractéristiques du corpus.

Note : normalement la méthode Reinert de classification s'applique à des segments de texte courts, et non à des textes longs. Il faudrait donc plutôt découper les textes en segments pour les classer. Voir la présentation de Julien Barnier pour ce faire !