ATELIER DATA ANALYSIS / Congrès MQDS

Marie VAUGOYEAU

07/06/2023

Contents

Introduction à R

Fonctionnement de R

`R` est un langage open-source de programmation orienté objet, développé par et pour les statisticien.ne.s.
Il fonctionne grâce à des fonctions (toujours suivis de parenthèses dans ce support comme `plot()`) paramétrables grâce à des arguments. Les fonctions sont réunis dans des packages (ici écrit entre `{}`, *par exemple* `{tidyverse}`).
Certains packages sont téléchargés de base dans `R`, les autres sont à installer depuis le `CRAN` ou depuis d'autres sources.
Pour pouvoir utiliser les fonctions contenues dans un package il faut le charger dans l'environnement en utilisant `library(nom_du_package)`.

RStudio

Pour bien travailler avec R, il est indispensable d'utiliser RStudio comme IDE.
Pour installer RStudio, il faut commencer par installer R, puis RStudio.

Le `cheatsheet` de RStudio en tant qu'IDE est téléchargeable via les ressources de posit.

Attention : Pour toutes les personnes travaillant sous **Windows**, il faut aussi installer `Rtools`.

```
install.packages("utils")
install.packages("installr")
installr::install.Rtools()
```

RMarkdown

La base

La syntaxe et la structure des rapports générés avec `Rmarkdown` permettent de faciliter la **maintenance** et la **mise à jour** des codes réalisés. En effet, `Rmarkdown` est conçu pour permettre l'**automatisation** et la **réutilisation de code**. Un bon développement fait gagner du temps par la suite.

Le `Rmarkdown` est un type de fichier qui permet d'organiser le code (R mais aussi Python, SQL...) **sous forme de blocs** avec du **texte** et des **sorties** (graphiques, tableaux...).
Il est aussi possible d'intégrer des images, des liens vers des fichiers extérieurs ou des pages web : `![](adresse_image)` ou `[texte_a_afficher](adresse_page_web)`.

Le `cheatsheet` de `Rmarkdown` est téléchargeable via les ressources de RStudio.

Syntaxe en `Rmarkdown`

Le fichier rédigé en `Rmarkdown` est différent de la sortie finale (`.docx`, `.html`, `.pdf`...) qui est "tricotée" (`knit` en anglais). Il faut donc utiliser un codage spécifique pour mettre en forme le texte.

Les titres sont caractérisés par des `#` en fonction du niveau : `#Titre de niveau 1`, `##Titre de niveau 2`...

La mise en forme du texte se fait avec les étoiles ou les " :

- Une étoile `*` avant et après la partie à valoriser permet de mettre en *italique* (codé `*italique*`)
- Deux `**` avant et après mettent en **gras** (codé `**gras**`)
- Un **double espace** permet de passer à la ligne. Il faut donc toujours mettre deux espaces à la fin de chaque ligne. Sans ces espaces, toutes les lignes sont collées les unes à la suite des autres.
- Pour éditer un `format code` (sans qu'il se lance) pour présenter les packages, les fonctions ou les objets utilisés (comme fait dans ce document), il faut encadrer d'impostrophes

Le format `Rmarkdown` fait la part belle au texte alors que le code doit être spécifié. Le code apparaît soit dans des `chunks` (morceaux en anglais) qui commencent et terminent avec 3 signes accents graves (ou impostrophe) qu'on peut générer avec `Ctrl+ Alt + I`, soit mis dans le texte sur une ligne entouré d'impostrophes.
Dans les deux cas, il faut toujours écrire au début le langage utilisé (donc r).
Par exemple : La largeur moyenne des sépales du jeu de données iris est 5.84.

Le `{tidyverse}`

Le `{tidyverse}` s'appelait encore le `hadleyverse` il y a quelques année, c'est-à-dire `l'univers de Hadley` pour Hadley Wickham son génial créateur.
Le but de Hadley est de rendre l'analyse données plus facile, plus rapide et surtout **plus fun** et je trouve que cela transparaît dans ses packages !
Le `{tidyverse}` c'est l'ensemble des packages open-source développé par Hadley et son équipe (Hadley travaille maintenant pour RStudio en plus de plusieurs universités) qui partagent la même philosophie, la même structure de données (le fameux format `tidy`) et la même syntaxe.

Les packages concernés :

_ `ggplot2` : Visulisation des données
_ `dplyr` : Manipulation des données (filtrer, trier,...) à ne pas confondre avec `tidyr` qui manipule le format du jeu de données. Présenté le 7 février sur twitch.
_ `tidyr` : Modification du format du jeu de données pour en faire un jeu de donnée `tidy`. Présenté le 7 février sur twitch.
_ `readr` : Lecture rapide de fichiers de données format `csv` et autres. **Attention** : format `xslx` non pris

en charge, il faut utiliser le package `readxl` qui fait partie du `tidyverse` au sens large mais qui n'est pas attaché par défaut quand on fait `library(tidyverse)`
_ `purrr` : Permet le remplacement d'un grand nombre de boucles. Présenté le 2 mai sur twitch.
_ `tibble` : Format des données `tidy`
_ `stringr` : Manipulation des chaînes de caractères. Présenté le 21 mars sur twitch.
_ `forcats` : Manipulation des variables facteurs `factors`. Présenté le 21 mars sur twitch.
_ `lubrdate` : Manipulation des dates. *Nouveau dans le* ***{tidyverse}***

Oubli : Import et export de fichiers

Lors de la séance de mercredi 7 juin j'ai oublié de parler de l'import et l'export des fichiers !
Contrairement à ce qui se fait classiquement je vais enregistrer des fichiers dans le sous-dossier `data` pour les réimporter afin que c'est lignes de codes soit réutilisables par tou.te.s.

Création d'un dossier `data`

Possible en cliquant sur `New Folder` ou grâce à la fonciton `dir.create()` du package `{base}`.

```
dir.create("data", showWarnings = FALSE)
```

Enregistrer un fichier

Fichier `.csv` Création un fichier `.csv` à partir du jeu de données `mtcars` grâce à la fonction `write.csv()` du package `{utils}`.

```
write.csv(
  mtcars,
  "data/jdd_voiture.csv"
  )
```

Fichier `.xlsx` Création un fichier `.xlsx` à partir du jeu de données `women` grâce à la fonction `write.xlsx()` du package `{openxlsx}`.
Il n'y a pas de fonction présente de base pour importer un fichier `.xlsx`.

```
library(openxlsx)

write.xlsx(
  women,
  "data/jdd_femme.xlsx"
  )
```

Web-scrapping Il est possible de récupérer facilement des données sur une page internet grâce au package `{rvest}`.
Par exemple : Importer le titre des lettres de la contesse de Ségur à partir de Wikisource grâce aux fonctions du package `{rvest}`.

```
library(rvest)
(page <- read_html("https://fr.m.wikisource.org/wiki/Lettres_de_la_comtesse_de_S%C3%A9gur/Texte_entier")
  html_nodes("h1") %>%
  html_text())
```

```
## [1] "Lettres de la comtesse de Ségur/Texte entier"
```

Import d'un fichier

Lecture d'un `.csv` Import du fichier `.csv` créé grâce à la fonction `read.csv()` du package `{utils}`.
Le nom des lignes est stocké par défaut dans la première colonne `X`.

```
jdd_voiture <- read.csv(
  file = "data/jdd_voiture.csv",
  row.names = "X"
)
```

Lecture d'un `.xlsx` Import du fichier `.xlsx` à partir du jeu de données `women` grâce à la fonction `read.xlsx()` du package `{openxlsx}`.

```
jdd_femme <- read.xlsx(
  "data/jdd_femme.xlsx"
)
```

Analyse descriptive

Utilisation du jeu de données `penguins` du package `{palmerpenguins}` qui recense les caractéristiques des pingouins de l'archipel de Palmer. Plus d'informations sur ce jeu de données dans la page d'aide `help(penguins)`.

visualisation rapide des données

Avec la fonction très généraliste `plot()` chargée de base dans l'environnement.

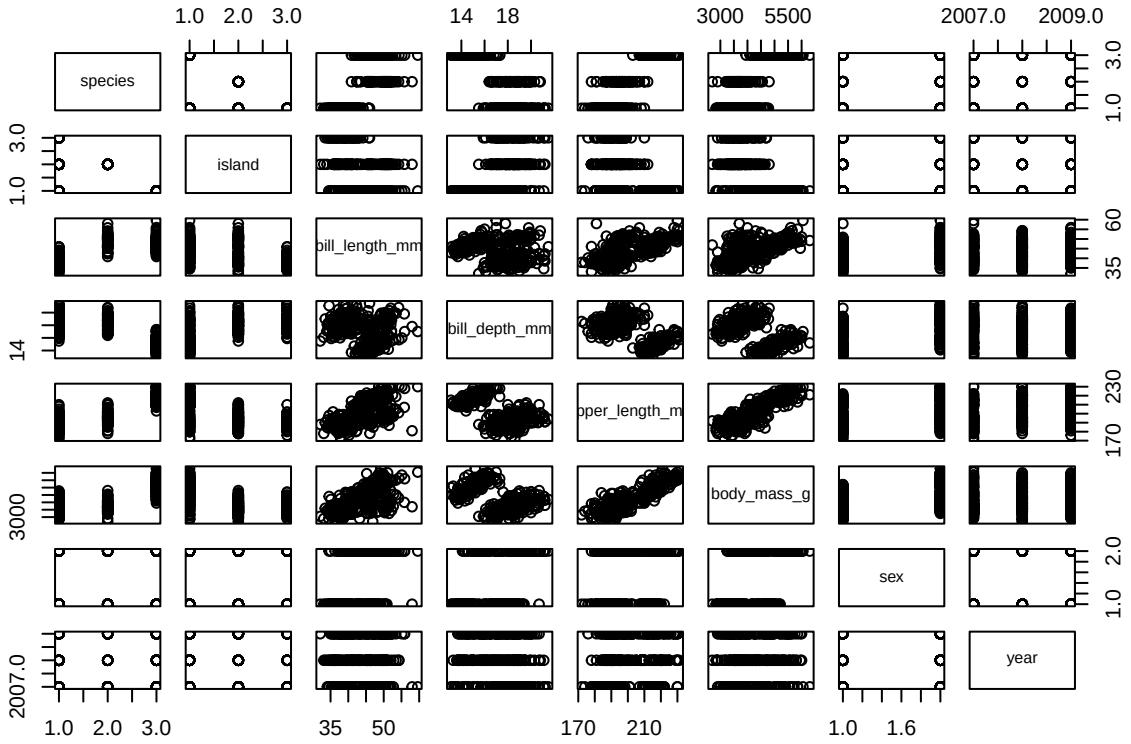
```
# nécessité d'importer le package pour utiliser le jeu de données
library(palmerpenguins)

penguins
```

```
## # A tibble: 344 x 8
##    species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##    <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
##  1 Adelie  Torgersen           39.1          18.7               181        3750
##  2 Adelie  Torgersen           39.5          17.4               186        3800
##  3 Adelie  Torgersen           40.3          18                 195        3250
##  4 Adelie  Torgersen           NA            NA                  NA          NA
##  5 Adelie  Torgersen           36.7          19.3               193        3450
##  6 Adelie  Torgersen           39.3          20.6               190        3650
##  7 Adelie  Torgersen           38.9          17.8               181        3625
##  8 Adelie  Torgersen           39.2          19.6               195        4675
##  9 Adelie  Torgersen           34.1          18.1               193        3475
## 10 Adelie  Torgersen           42            20.2               190        4250
## # i 334 more rows
## # i 2 more variables: sex <fct>, year <int>
```

```
View(penguins)

plot(penguins)
```



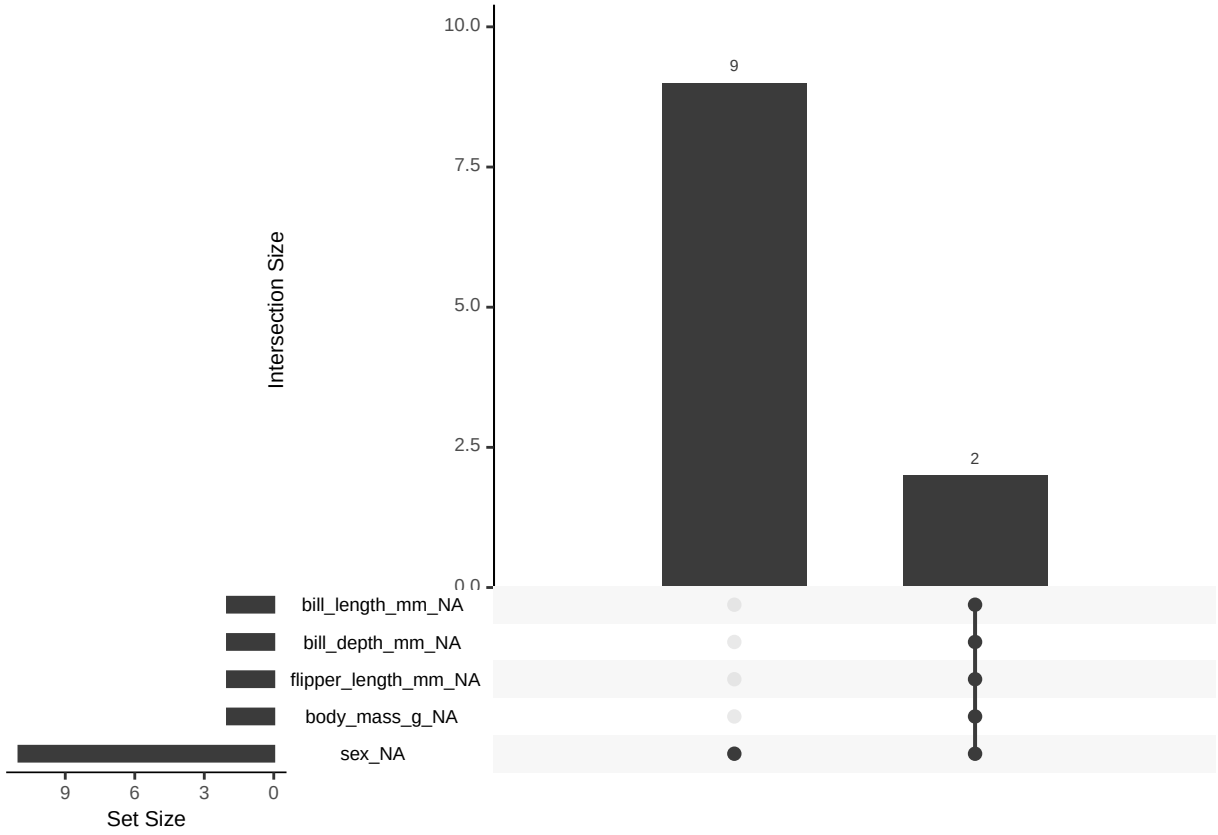
Pour avoir un appercu des données il est interressant d'utiliser la fonction `summary()` présent dans le package {base}.

```
summary(penguins)
```

```
##       species          island    bill_length_mm  bill_depth_mm
##  Adelie   :152   Biscoe   :168   Min.   :32.10   Min.   :13.10
##  Chinstrap: 68   Dream    :124   1st Qu.:39.23   1st Qu.:15.60
##  Gentoo   :124   Torgersen: 52   Median :44.45   Median :17.30
##                                  Mean   :43.92   Mean   :17.15
##                                  3rd Qu.:48.50   3rd Qu.:18.70
##                                  Max.   :59.60   Max.   :21.50
##                                  NA's   :2       NA's   :2
##  flipper_length_mm  body_mass_g       sex           year
##  Min.   :172.0     Min.   :2700   female:165   Min.   :2007
##  1st Qu.:190.0     1st Qu.:3550   male  :168   1st Qu.:2007
##  Median :197.0     Median :4050   NA's  : 11   Median :2008
##  Mean   :200.9     Mean   :4202                Mean   :2008
##  3rd Qu.:213.0     3rd Qu.:4750                3rd Qu.:2009
##  Max.   :231.0     Max.   :6300                Max.   :2009
##  NA's   :2         NA's   :2
```

Il y a des valeurs manquantes, il faut donc les visualiser.
{naniar} est un package très performant pour travailler sur les données manquantes.

```
naniar::gg_miss_upset(penguins)
```



Pour visualiser différement le tableau de données, il est possible d'utiliser la fonction `glimpse()` du `{tidyverse}`. Plus d'information sur le `{tidyverse}` dans le paragraphe ci-dessus Le `{tidyverse}`.

```
library(tidyverse)

glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ sex               <fct> male, female, female, NA, female, male, female, male~
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

```
pingouins <- penguins
# le jeu de données est chargé dans l'environnement et visible en haut à droite
```

Attention le chargement de certain package remplace des fonctions déjà chargées par celles chargée en dernière.
Par exemple, le chargement du package `{tidyverse}` ou `{dplyr}` remplace la fonction `filter()` du package `{stat}` par la sienne.

Analyse univariée

Variable qualitative

Il y a trois variables qualitatives ici : `species`, `island` et `sex`.
Toutes les trois sont finis -> donc on peut réaliser directement des tableaux de contingence.

```
# fonction `table()` du package `{base}`
table(pingouins$species)
```

Tableau de contingence

```
##
##    Adelie Chinstrap    Gentoo
##       152        68       124
```

```
table(pingouins$island)
```

```
##
##    Biscoe     Dream Torgersen
##       168       124        52
```

```
table(pingouins$sex)
```

```
##
## female   male
##    165    168
```

```
# ne permet pas de voir les NA !

count(pingouins, sex)
```

```
## # A tibble: 3 x 2
##   sex        n
##   <fct>  <int>
## 1 female   165
## 2 male     168
## 3 <NA>      11
```

Petit apparté sur le pipe Le **pipe** est une syntaxe qui permet d'enchaîner les opérations sur un même objet.
Plus d'information dans le guide-R de Joseph Larmarange et cet article de blog de Lise Vaudor (c'est sur le pipe de `{magrittr}` et non le pipe natif présenté ici mais le fonctionnement est le même).

```
# sans pipe
mean(sqrt(c(1:10)*2))
```

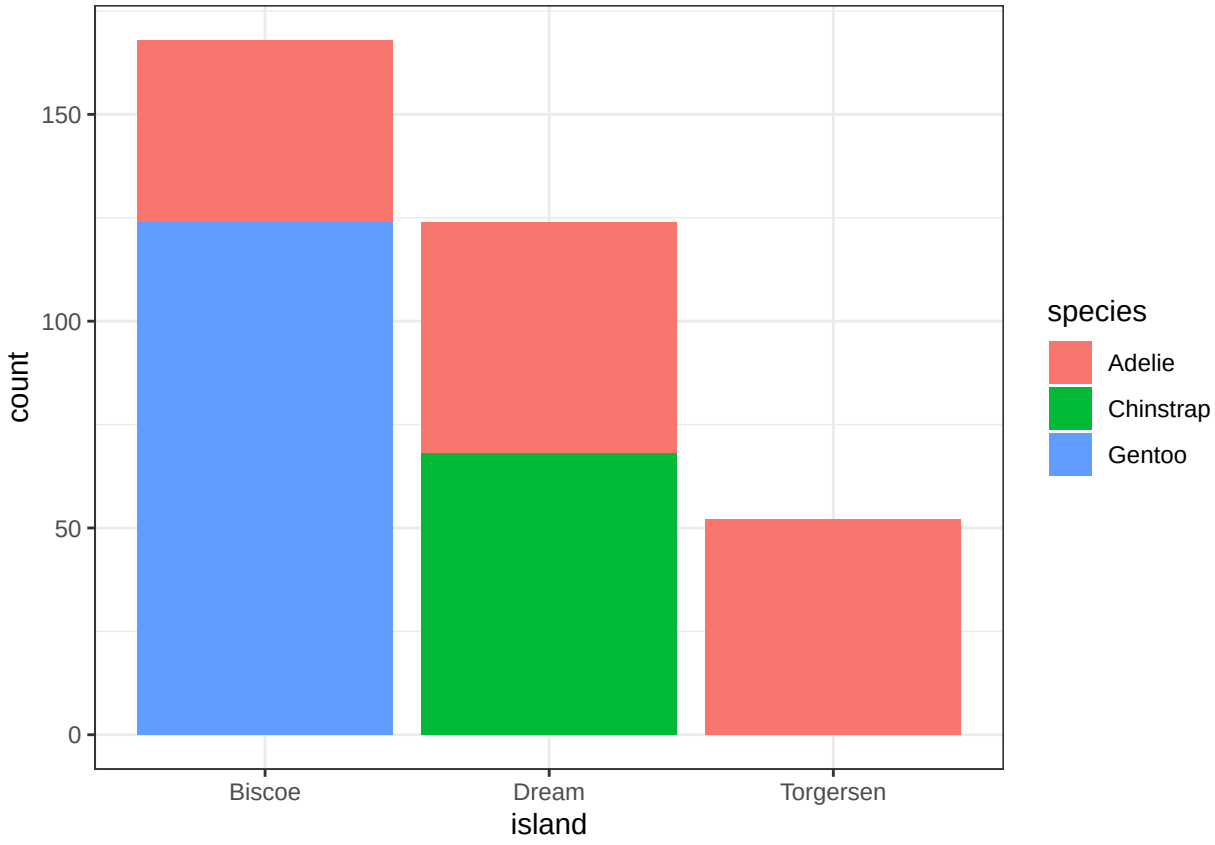
```
## [1] 3.177494
```

```
# avec pipe
(c(1:10)*2) |>
  sqrt() |>
  mean()
```

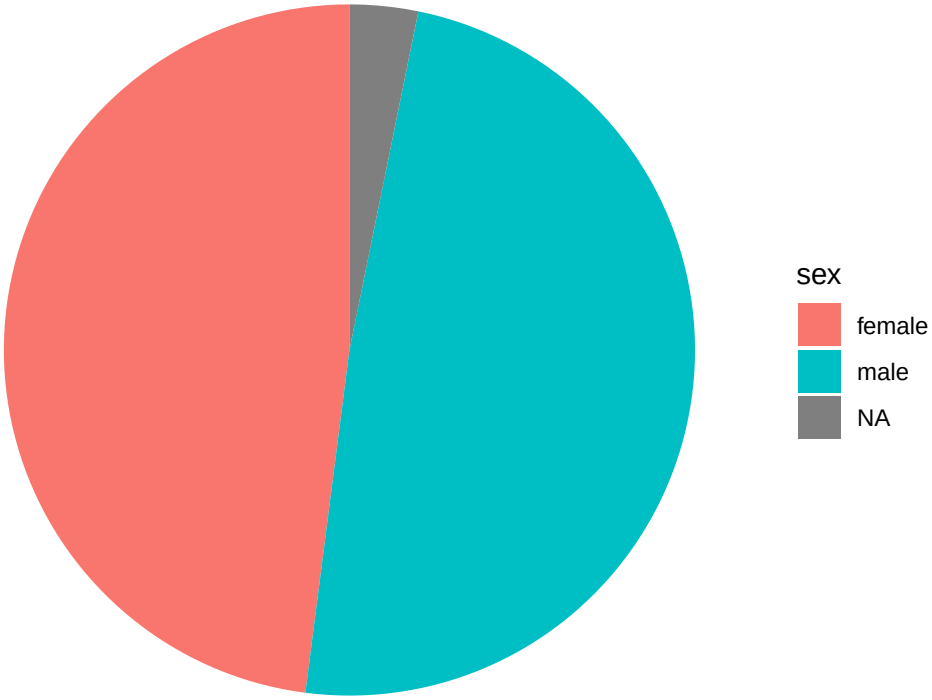
```
## [1] 3.177494
```

Représentation graphique Ressource conseillée pour la réalisation de graphiques : From Data to Viz.

```r
# diagramme en barre
pingouins |>
  # filter(species == "Adelie") |>
  ggplot() +
  aes(x = island, fill = species) +
  geom_bar() +
  theme_bw()
```

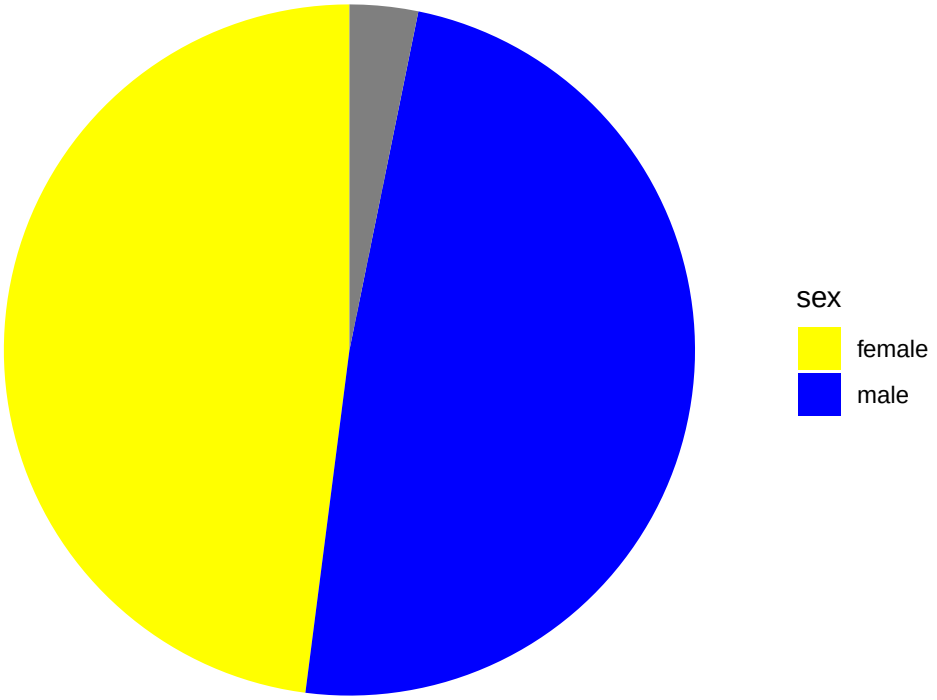



```r
# diagramme circulaire
pingouins |>
  count(sex) |>
  ggplot() +
  aes(x = "", y = n, fill = sex) +
  geom_bar(stat = "identity") +
  coord_polar("y") +
  theme_void()
```




```
couleur <- c("female" = "yellow", "male" = "blue")

pingouins |>
  count(sex) |>
  ggplot() +
  aes(x = "", y = n, fill = sex) +
  geom_bar(stat = "identity") +
  coord_polar("y") +
  scale_fill_manual(values = couleur) +
  theme_void()
```

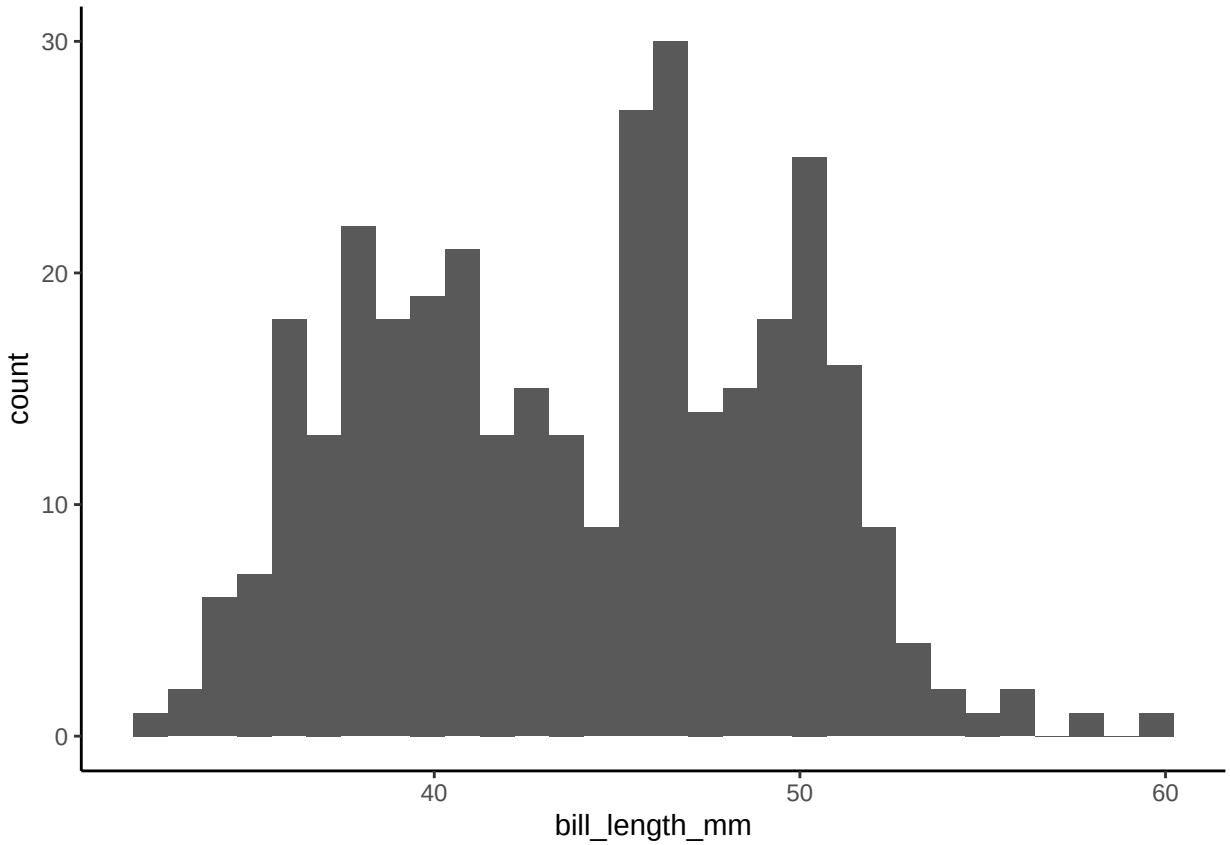


Variable quantitative

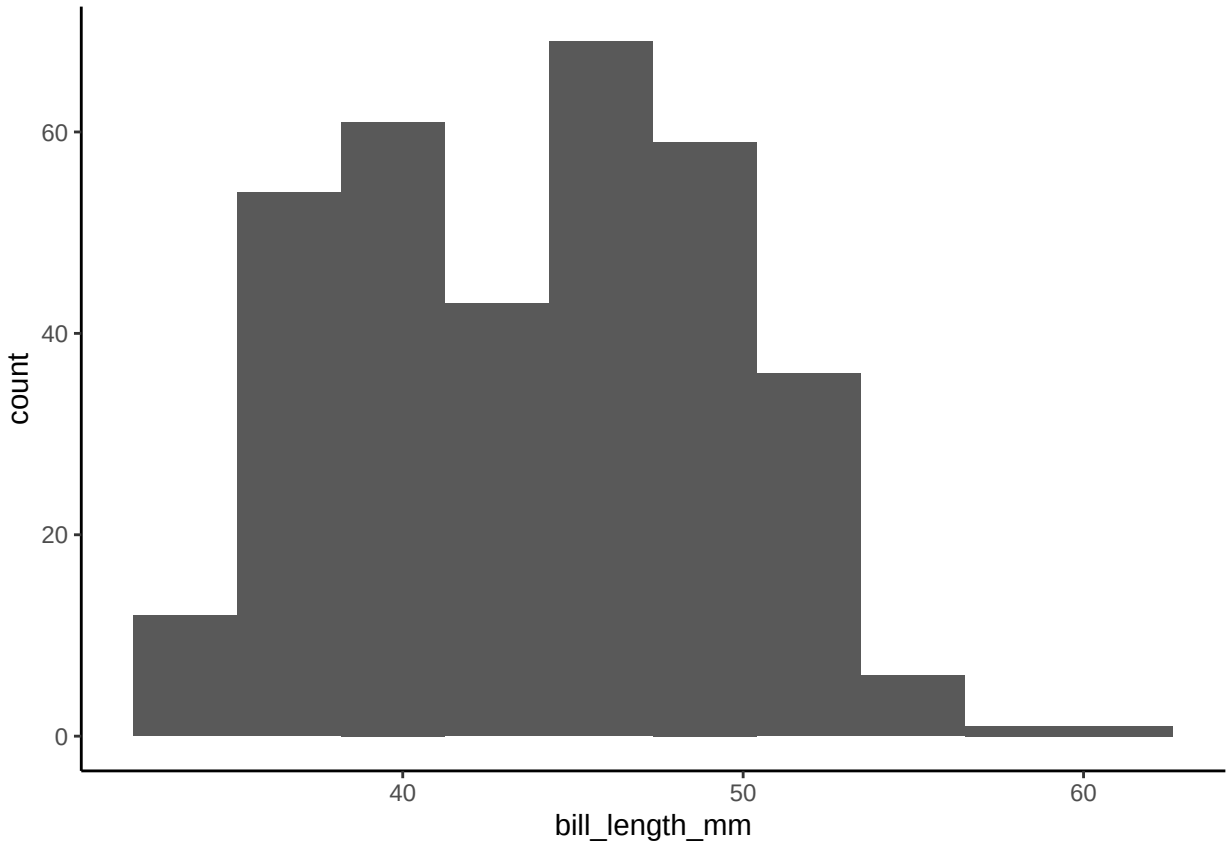
Il y a 5 variables quantitatives, il est possible d'étudier leurs dispersion grâce aux histogrammes ou de calculer les mesures de cette dispersion.

Histogramme **Attention** au nombre d'intervalles représenté.

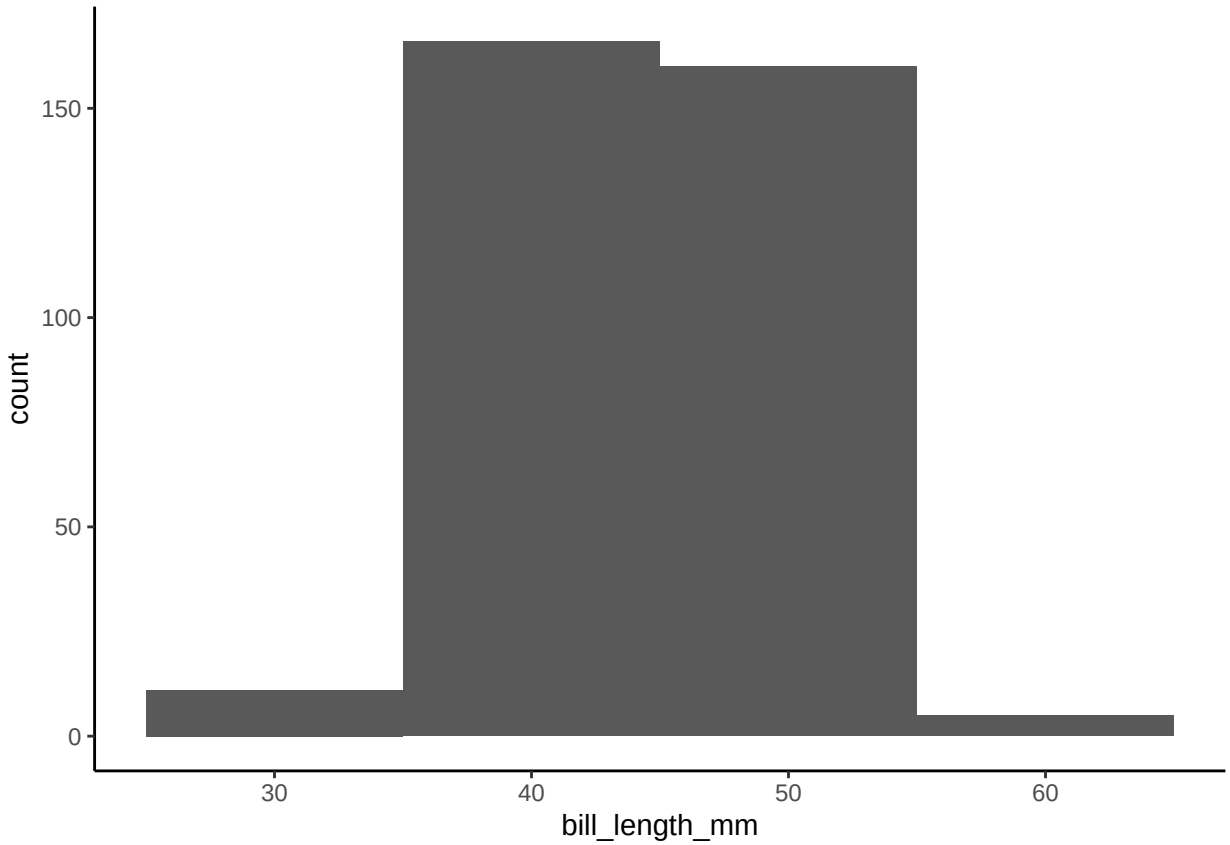
```
pingouins |>
  ggplot() +
  aes(x = bill_length_mm) +
  geom_histogram() +
  theme_classic()
```




```
# changement nombre d'intervalles (10)
pingouins |>
  ggplot() +
  aes(x = bill_length_mm) +
  geom_histogram(bins = 10) +
  theme_classic()
```




```
# largeur de la barre
pingouins |>
  ggplot() +
  aes(x = bill_length_mm) +
  geom_histogram(binwidth = 10) +
  theme_classic()
```




```
summary(penguins)
```

Calcul des mesures de dispersion

```
##       species          island    bill_length_mm  bill_depth_mm
##  Adelie   :152   Biscoe   :168   Min.   :32.10   Min.   :13.10
##  Chinstrap: 68   Dream    :124   1st Qu.:39.23   1st Qu.:15.60
##  Gentoo   :124   Torgersen: 52   Median :44.45   Median :17.30
##                                  Mean   :43.92   Mean   :17.15
##                                  3rd Qu.:48.50   3rd Qu.:18.70
##                                  Max.   :59.60   Max.   :21.50
##                                  NA's   :2       NA's   :2
##  flipper_length_mm  body_mass_g       sex           year
##  Min.   :172.0     Min.   :2700   female:165   Min.   :2007
##  1st Qu.:190.0     1st Qu.:3550   male  :168   1st Qu.:2007
##  Median :197.0     Median :4050   NA's  : 11   Median :2008
##  Mean   :200.9     Mean   :4202                Mean   :2008
##  3rd Qu.:213.0     3rd Qu.:4750                3rd Qu.:2009
##  Max.   :231.0     Max.   :6300                Max.   :2009
##  NA's   :2         NA's   :2
```

```
mean(pingouins$bill_depth_mm)
```

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

```r
# NA car présence de valeur manquantes

mean(pingouins$bill_depth_mm, na.rm = TRUE)
```

```
## [1] 17.15117
```

```r
max(pingouins$bill_length_mm, na.rm = TRUE)
```

```
## [1] 59.6
```

```r
# création d'un jeu de données sans valeurs manquantes (suppression des lignes avec NA)
pingouins <-
  penguins |>
  drop_na()

median(pingouins$flipper_length_mm)
```

```
## [1] 197
```

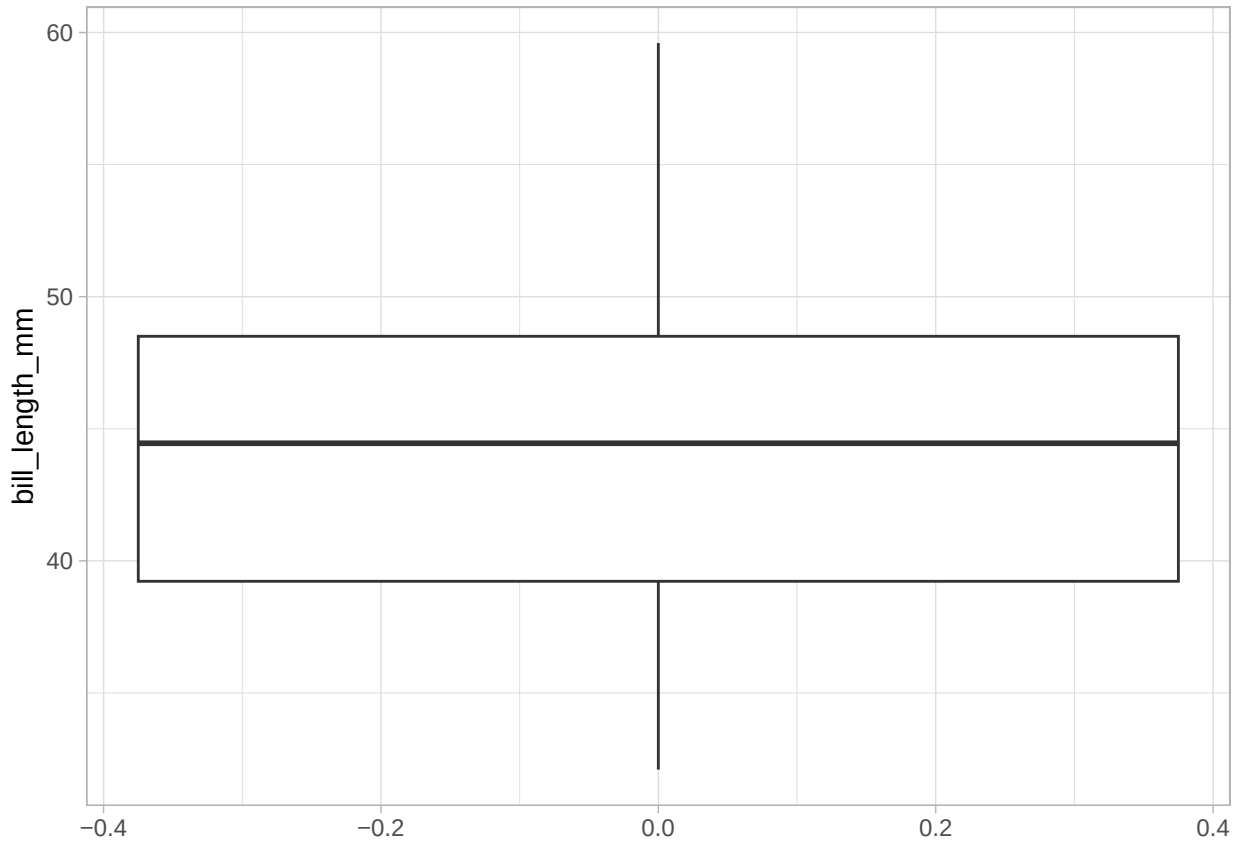
```r
pingouins |>
  summarise(
    across(
      .cols = where(is.numeric),
      .fns = list(
        moyenne = ~ mean(.x),
        minimum = ~ min(.x),
        maximum = ~ max(.x)
      ),
      .names = "{col}_{fn}"
    )
  ) |>
  pivot_longer(everything())
```

```
## # A tibble: 15 x 2
##    name                       value
##    <chr>                      <dbl>
##  1 bill_length_mm_moyenne      44.0
##  2 bill_length_mm_minimum      32.1
##  3 bill_length_mm_maximum      59.6
##  4 bill_depth_mm_moyenne       17.2
##  5 bill_depth_mm_minimum       13.1
##  6 bill_depth_mm_maximum       21.5
##  7 flipper_length_mm_moyenne  201.
##  8 flipper_length_mm_minimum  172
##  9 flipper_length_mm_maximum  231
## 10 body_mass_g_moyenne       4207.
## 11 body_mass_g_minimum       2700
## 12 body_mass_g_maximum       6300
## 13 year_moyenne              2008.
## 14 year_minimum              2007
## 15 year_maximum              2009
```

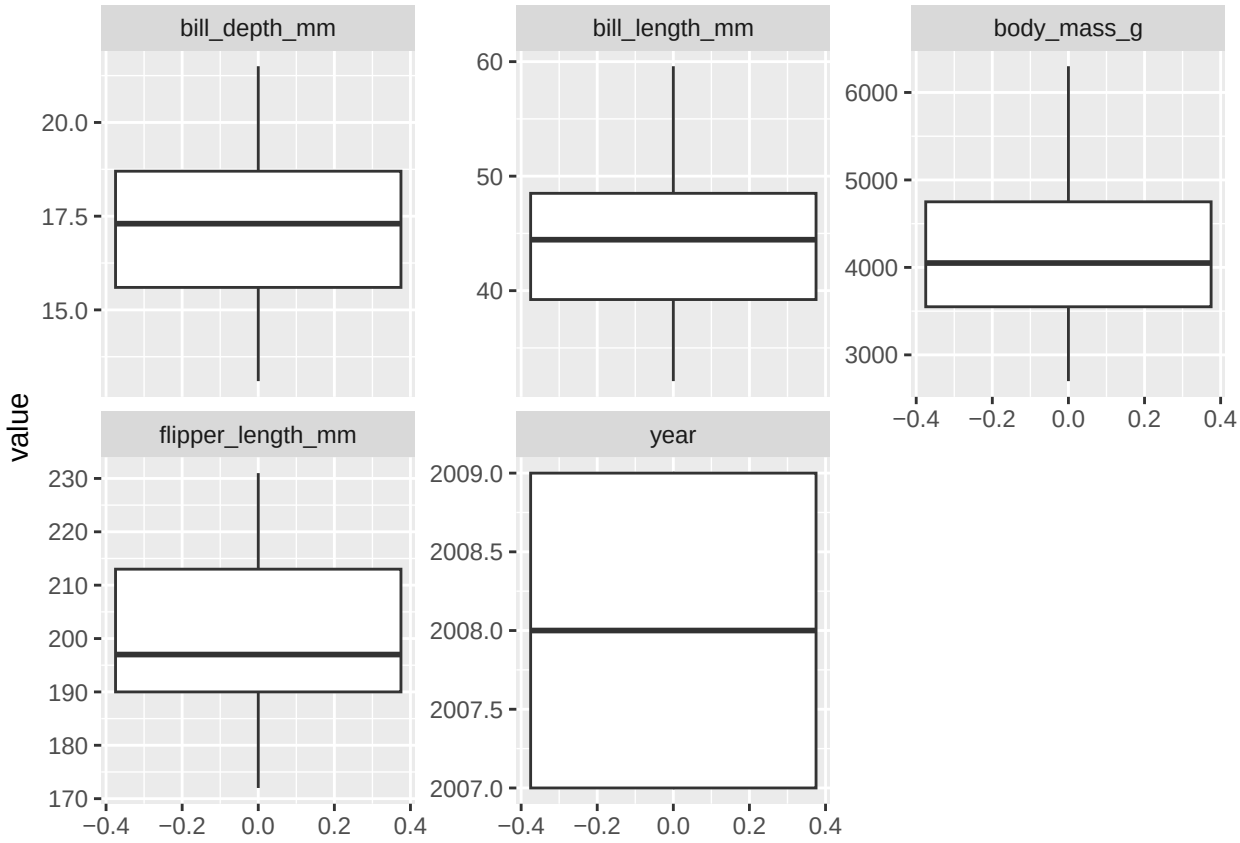
Boîte à moustaches Graphique généralisant les données de dispersion.

```r
penguins |>
  ggplot() +
```

```
  aes(y = bill_length_mm) +
  geom_boxplot() +
  theme_light()
```




```
# sur toutes les colonnes numériques
penguins |>
  pivot_longer(
    cols = where(is.numeric)
  ) |>
  ggplot() +
  aes(y = value) +
  facet_wrap(~ name, scales = "free_y") +
  geom_boxplot() +
  theme_grey()
```

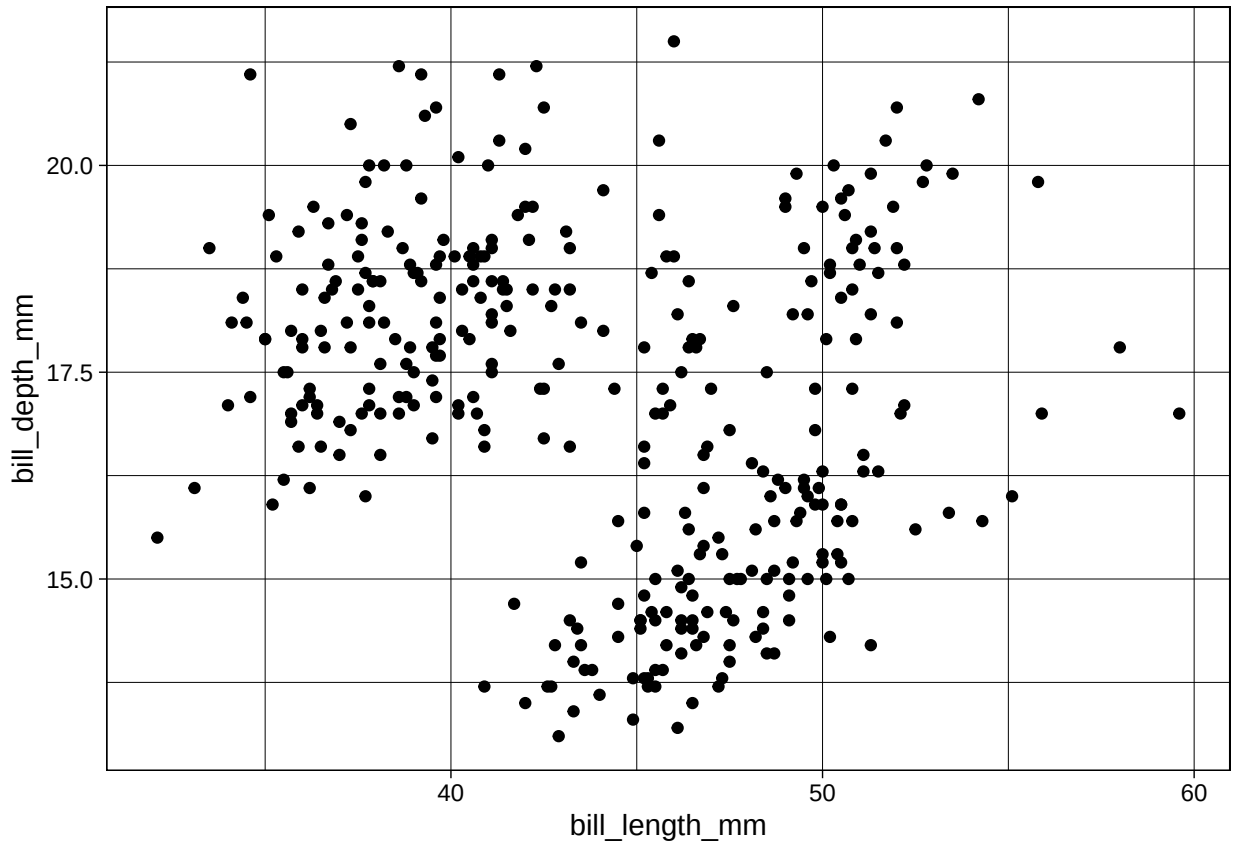


Analyse bivariée

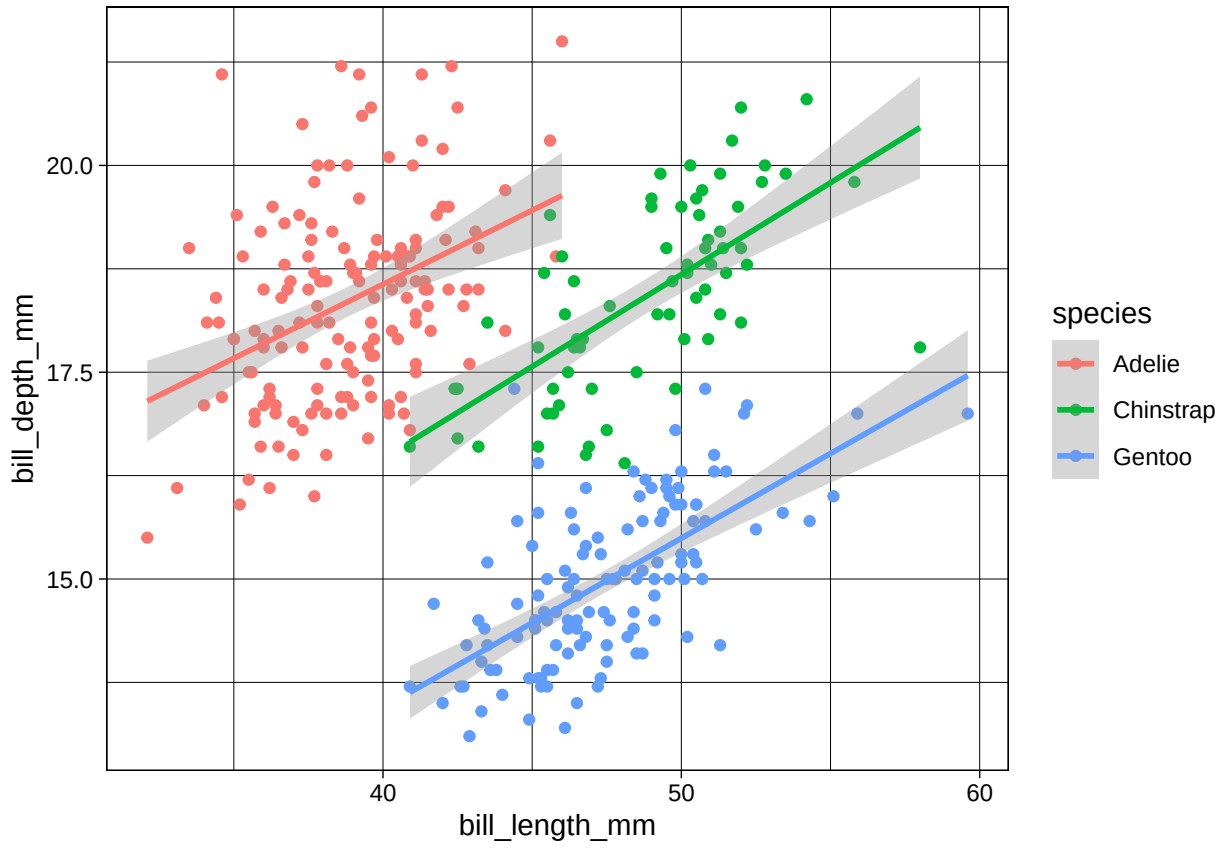
Deux variables quantitatives

Nuage de points Il faut **toujours commencer** par la représentation graphique !

```
penguins |>
  ggplot() +
  aes(x = bill_length_mm, y = bill_depth_mm) +
  geom_point()+
  theme_linedraw()
```




```
# l'influence du sex
penguins |>
  ggplot() +
  aes(x = bill_length_mm, y = bill_depth_mm, color = species) +
  geom_point()+
  geom_smooth(method ="lm") +
  theme_linedraw()
```



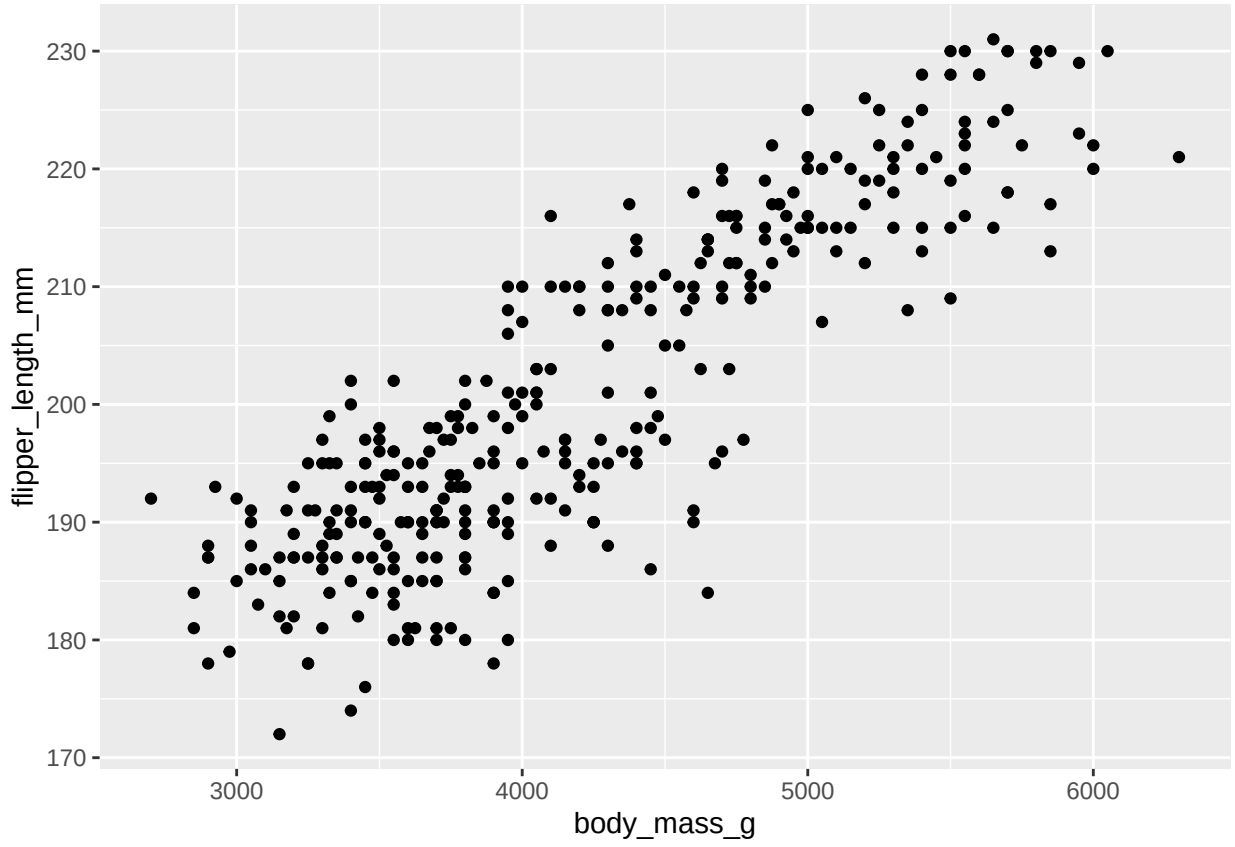

```
modele_lineaire <- lm(
  bill_depth_mm ~ bill_length_mm,
  data = penguins |> filter(species == "Adelie")
  )
summary(modele_lineaire)
```

Réalisation d'une régression linéaire

```
##
## Call:
## lm(formula = bill_depth_mm ~ bill_length_mm, data = filter(penguins,
##     species == "Adelie"))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1512 -0.8012 -0.0698  0.5766  3.5032
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    11.40912    1.33893   8.521 1.61e-14 ***
## bill_length_mm  0.17883    0.03444   5.193 6.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.123 on 149 degrees of freedom
```

```
##   (1 observation effacée parce que manquante)
## Multiple R-squared:  0.1533, Adjusted R-squared:  0.1476
## F-statistic: 26.97 on 1 and 149 DF,  p-value: 6.674e-07
```

```
# autre exemple
penguins |>
  ggplot() +
  aes(y = flipper_length_mm, x = body_mass_g) +
  geom_point()
```

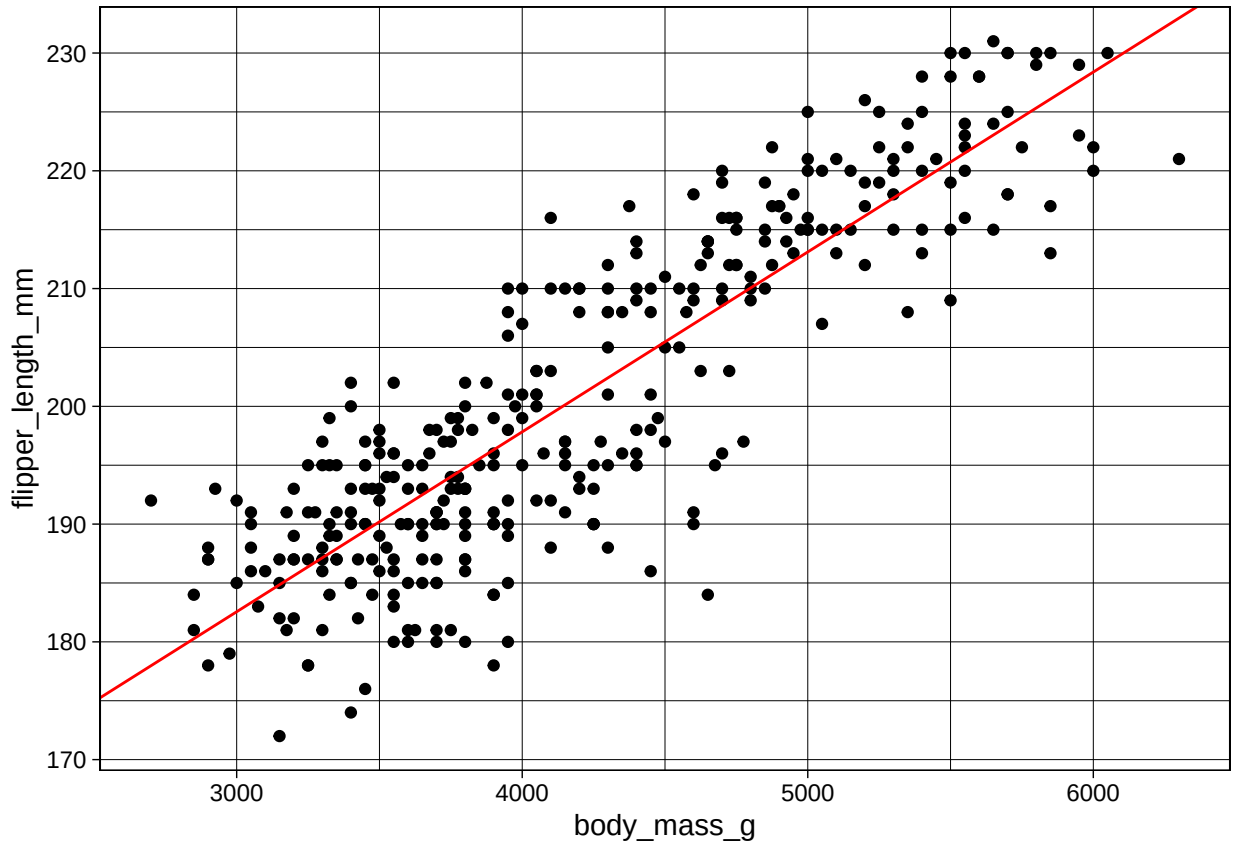



```
modele_lineaire_nageoire <-
  lm(
    flipper_length_mm ~ body_mass_g,
    data = penguins
    )
summary(modele_lineaire_nageoire)
```

```
##
## Call:
## lm(formula = flipper_length_mm ~ body_mass_g, data = penguins)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.7626  -4.9138   0.9891   5.1166  16.6392
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

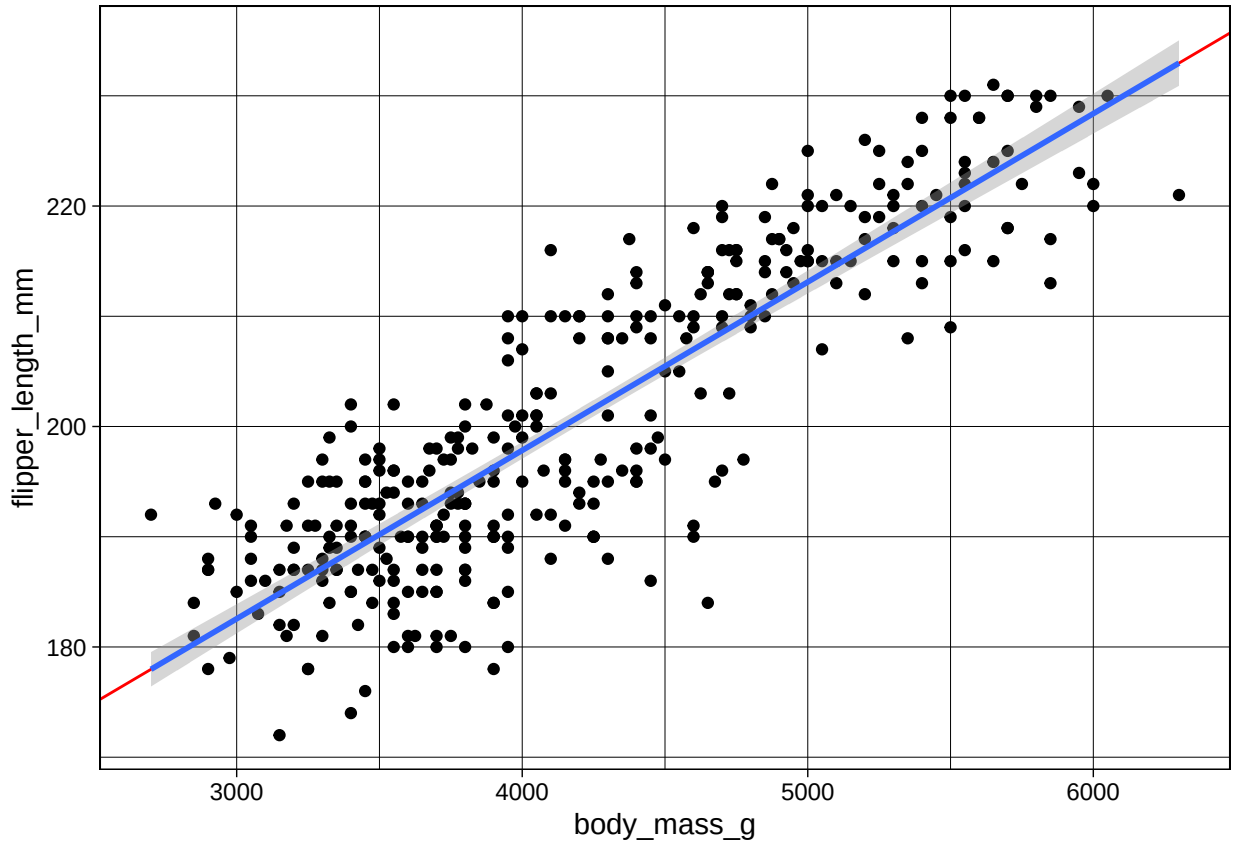
```
## (Intercept) 1.367e+02  1.997e+00   68.47   <2e-16 ***
## body_mass_g 1.528e-02  4.668e-04   32.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.913 on 340 degrees of freedom
##   (2 observations effacées parce que manquantes)
## Multiple R-squared:  0.759,  Adjusted R-squared:  0.7583
## F-statistic:  1071 on 1 and 340 DF,  p-value: < 2.2e-16
```

```r
# Représentation graphique
penguins |>
  ggplot() +
  aes(y = flipper_length_mm, x = body_mass_g) +
  geom_point() +
  geom_abline(
    aes(
      intercept = modele_lineaire_nageoire$coefficients[1],
      slope = modele_lineaire_nageoire$coefficients[2]),
    colour = "red"
    ) +
  theme_linedraw()
```




```r
# ou automatiquement
penguins |>
  ggplot() +
  aes(y = flipper_length_mm, x = body_mass_g) +
```

```
  geom_point() +  geom_abline(
    aes(
      intercept = modele_lineaire_nageoire$coefficients[1],
      slope = modele_lineaire_nageoire$coefficients[2]),
    colour = "red"
    ) +
  geom_smooth(method = "lm") +
  theme_linedraw()
```



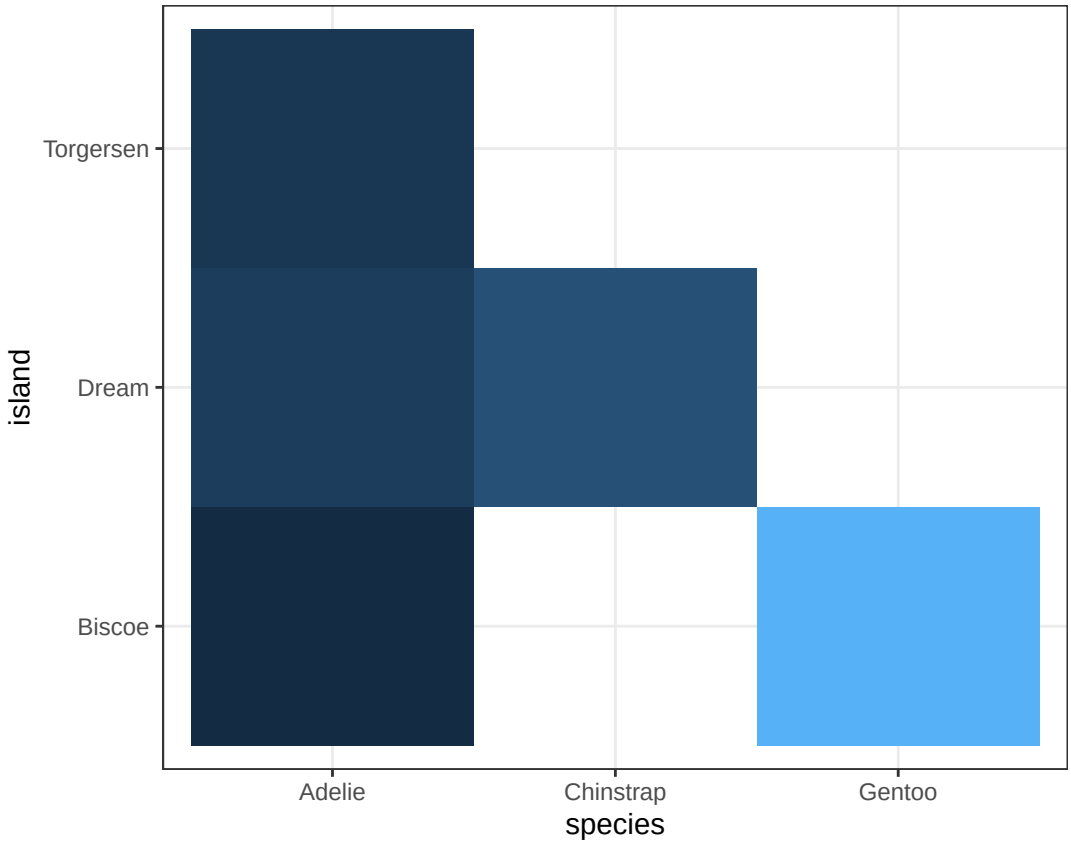
Deux variables qualitatives

```
penguins |>
  count(species, island)
```

Tableau de contingence

```
## # A tibble: 5 x 3
##   species   island        n
##   <fct>     <fct>     <int>
## 1 Adelie    Biscoe       44
## 2 Adelie    Dream        56
## 3 Adelie    Torgersen    52
## 4 Chinstrap Dream        68
## 5 Gentoo    Biscoe      124
```

```r
penguins |>
  count(species, island) |>
  ggplot() +
  aes(x = species, y = island, fill = n) +
  geom_tile() +
  theme_bw()
```



Carte des points chauds

```r
# diagramme de Venn
library(VennDiagram)
venn.diagram(
  x = map(
      .x = penguins$island |> levels(),
      .f = ~ penguins |>
        filter(island == .x) |>
        select(species) |>
        unlist()
    ),
  category.names = penguins$island |> levels(),
  filename = "img/diagramme_venn.png",
  output = TRUE,
  imagetype="png"
)
```

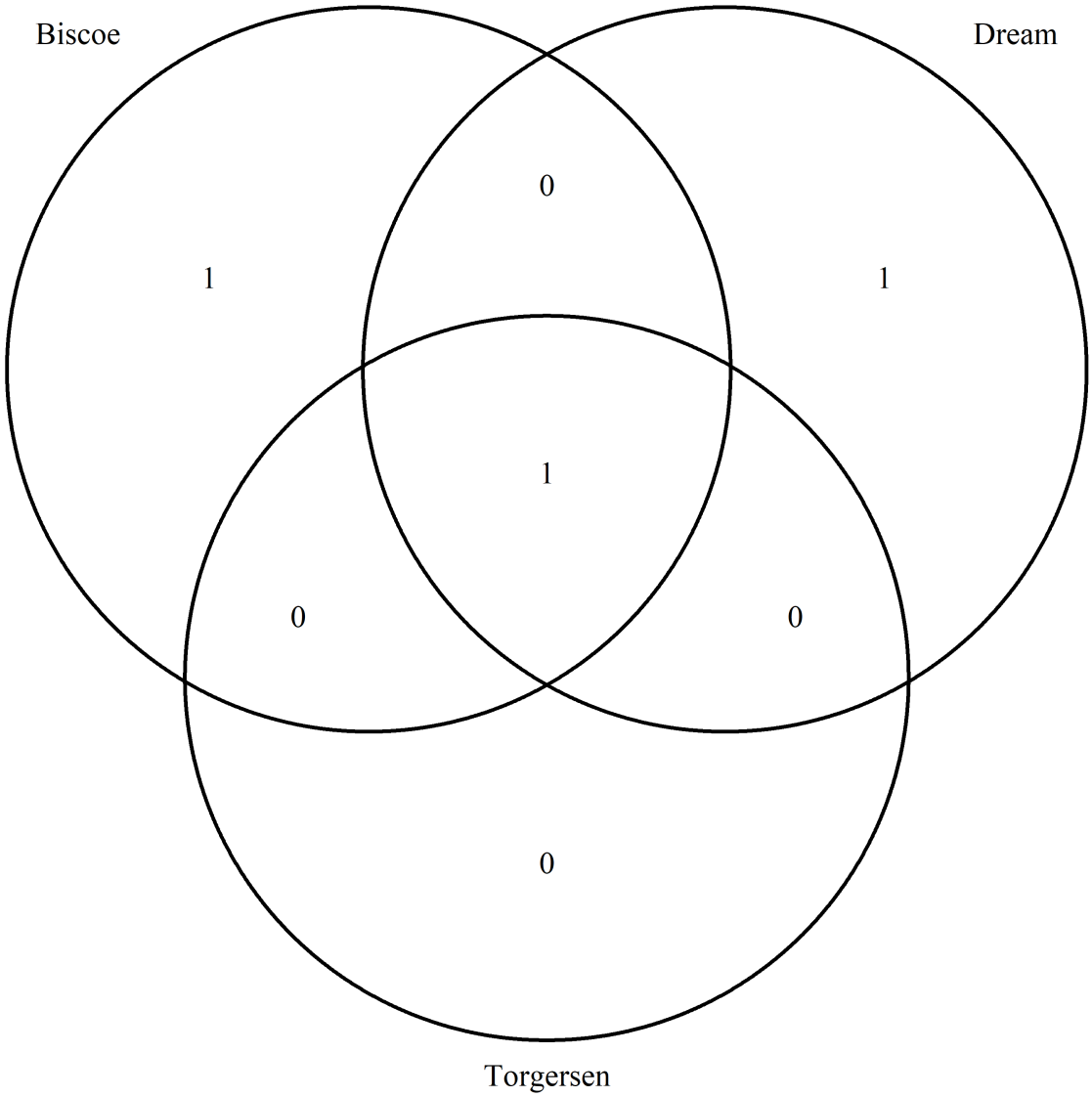
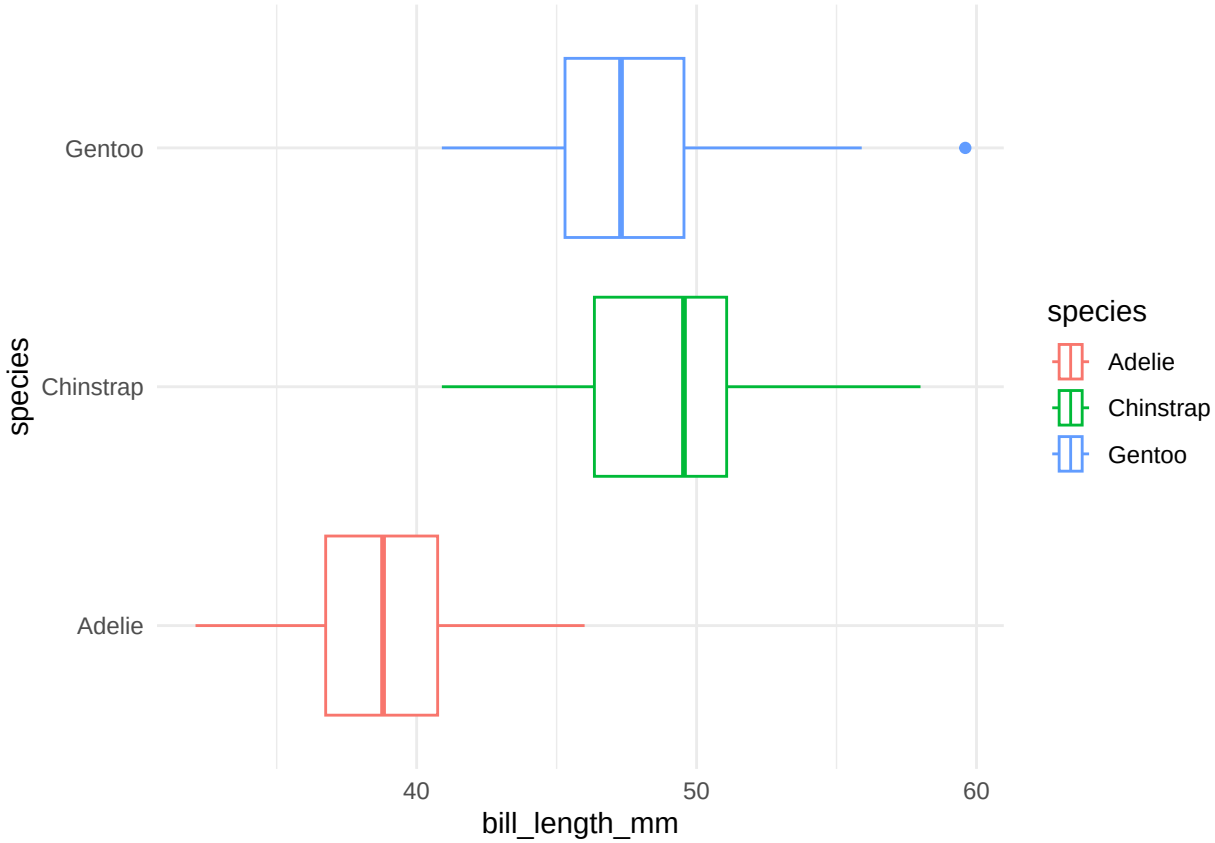


Diagramme de Venn
Le diagramme de Venn montre qu'une seule espèce est présente sur les trois îles, les deux autres espèces sont chacune présente sur une seule île.
Attention : Impossible de faire une effet croisé de l'île et de l'espèce !

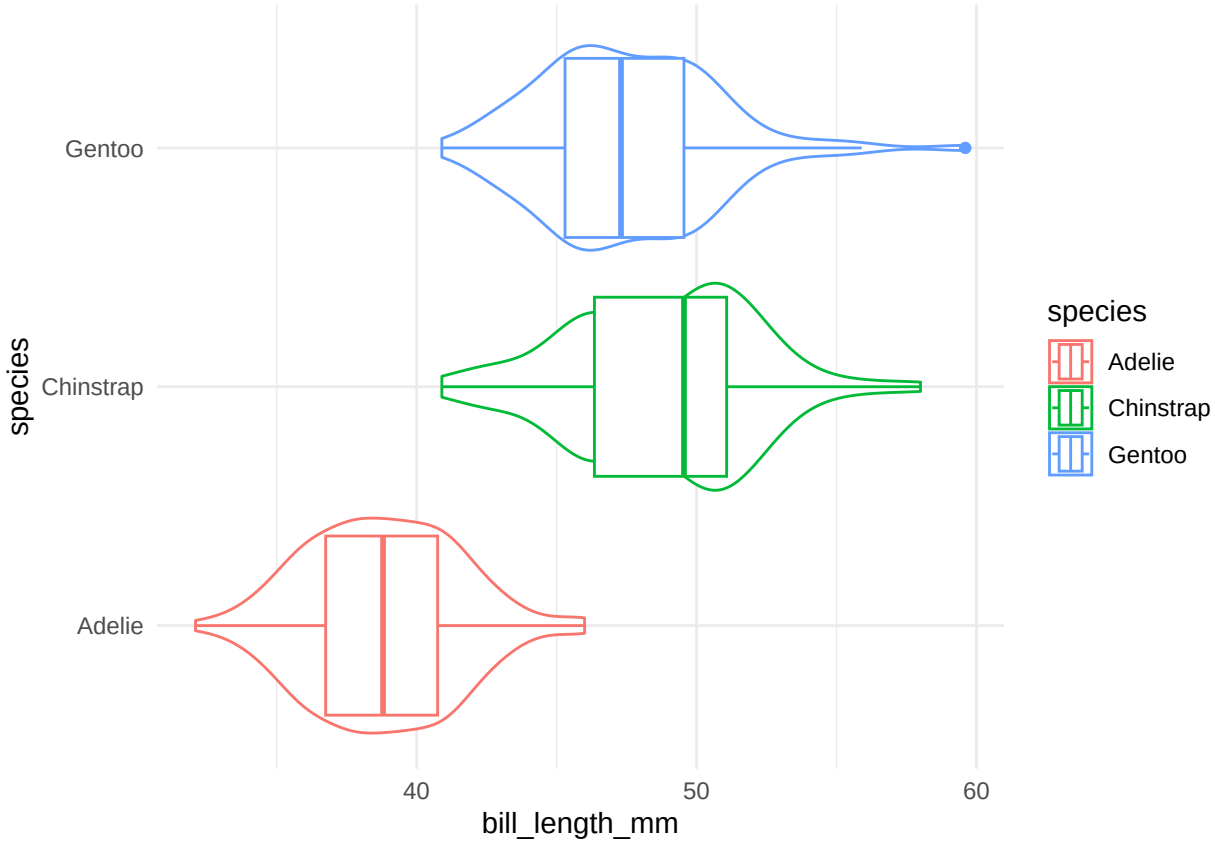
Une variable qualitative et une quantitative

Revient à chercher si au moins un des groupes est différent.
La représentation graphique se fait comme précédemment en ajoutant de la couleur.

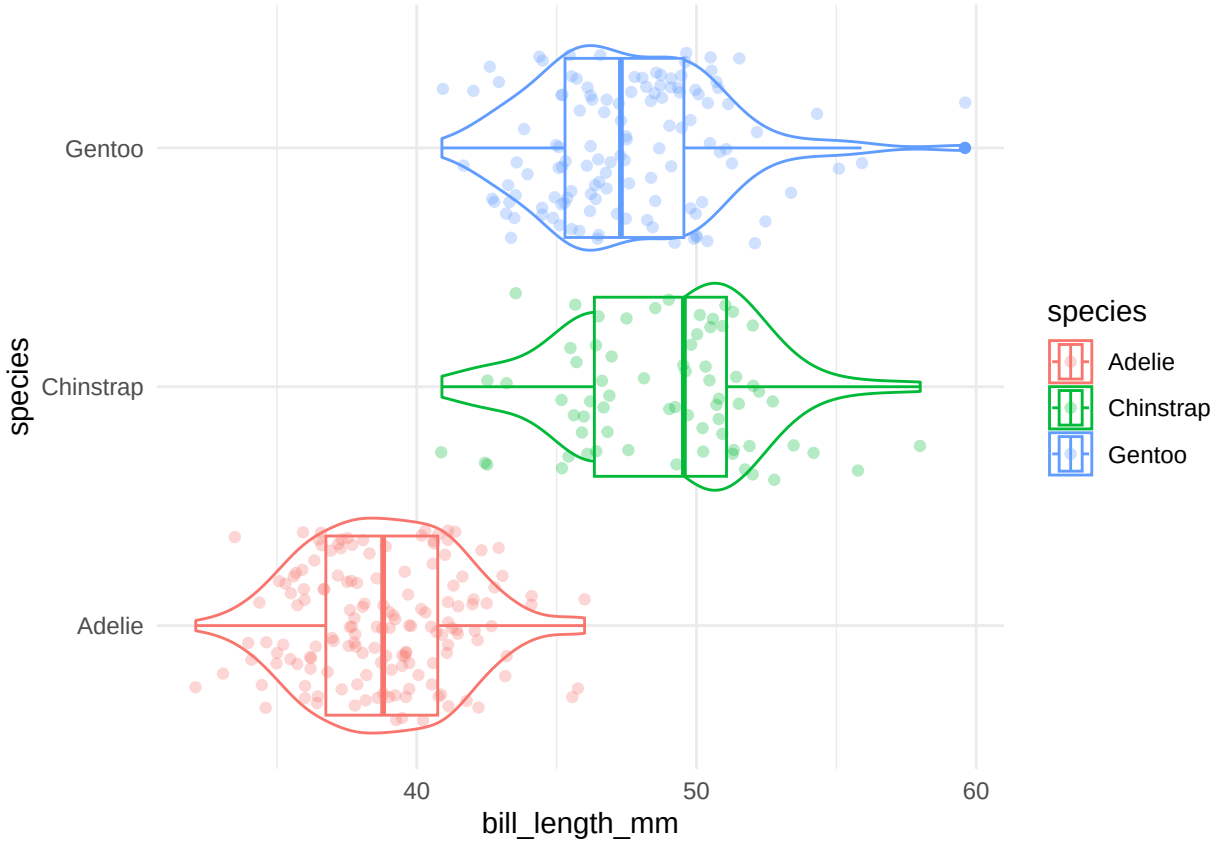
```
penguins |>
  ggplot() +
  aes(x = bill_length_mm, y = species, color = species) +
  geom_boxplot() +
  theme_minimal()
```




```
# attention à la dispersion des points, possible de rajouter un graphique violon
penguins |>
  ggplot() +
  aes(x = bill_length_mm, y = species, color = species) +
  geom_violin() +
  geom_boxplot() +
  theme_minimal()
```




```
# rajout des points en transparance
penguins |>
  ggplot() +
  aes(x = bill_length_mm, y = species, color = species) +
  geom_violin() +
  geom_boxplot() +
  geom_jitter(alpha = 0.3) +
  theme_minimal()
```



ANOVA Pour l'ANOVA permet de déterminer les différences entre groupes.

```
shapiro.test((penguins |> filter(species == "Adelie"))$bill_length_mm)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  (filter(penguins, species == "Adelie"))$bill_length_mm
## W = 0.99336, p-value = 0.7166
```

```
# vérification de l'égalité des variances
map(
  .x = penguins$species |> levels(),
  .f = ~ var.test(
    bill_length_mm ~ species,
    data = penguins |>  filter(species != .x))
  ) |>
  set_names(penguins$species |> levels())
```

```
## $Adelie
##
##  F test to compare two variances
##
## data:  bill_length_mm by species
## F = 1.174, num df = 67, denom df = 122, p-value = 0.4411
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
```

```
##  0.778208 1.818704
## sample estimates:
## ratio of variances
##           1.174017
##
##
## $Chinstrap
##
##  F test to compare two variances
##
## data:  bill_length_mm by species
## F = 0.74688, num df = 150, denom df = 122, p-value = 0.08894
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.5303445 1.0455238
## sample estimates:
## ratio of variances
##          0.7468774
##
##
## $Gentoo
##
##  F test to compare two variances
##
## data:  bill_length_mm by species
## F = 0.63617, num df = 150, denom df = 67, p-value = 0.02424
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.4153566 0.9432215
## sample estimates:
## ratio of variances
##          0.6361726
```

```
# anova
anova <- aov(bill_length_mm ~species, data = penguins)

summary(anova)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## species       2   7194    3597   410.6 <2e-16 ***
## Residuals   339   2970       9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 2 observations effacées parce que manquantes
```

```
# réalisation post-hoc
library(multcomp)
## attention la fonction `select()` du package `{dplyr}` qui permet de sélectionner une ou plusieurs co
summary(
  glht(anova, linfct = mcp(species = "Tukey"))
)
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
```

```
##
##
## Fit: aov(formula = bill_length_mm ~ species, data = penguins)
##
## Linear Hypotheses:
##                          Estimate Std. Error t value Pr(>|t|)
## Chinstrap - Adelie == 0  10.0424     0.4323  23.232  < 0.001 ***
## Gentoo - Adelie == 0      8.7135     0.3595  24.237  < 0.001 ***
## Gentoo - Chinstrap == 0  -1.3289     0.4473  -2.971  0.00871 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
anova$residuals |> shapiro.test()
```

```
##
##  Shapiro-Wilk normality test
##
## data:  anova$residuals
## W = 0.98903, p-value = 0.01131
```

```
# non paramétrique
kw <- kruskal.test(bill_length_mm ~species, data = penguins)
kw
```

Test de Kruskal Wallis (ANOVA non paramétrique)

```
##
##  Kruskal-Wallis rank sum test
##
## data:  bill_length_mm by species
## Kruskal-Wallis chi-squared = 244.14, df = 2, p-value < 2.2e-16
```

```
# post_hoc

summary(
  PMCMRplus::kwAllPairsNemenyiTest(
    data = penguins,
    bill_length_mm ~species
  )
)
```

```
##                          q value Pr(>|q|)
## Chinstrap - Adelie == 0  18.036  < 2e-16 ***
## Gentoo - Adelie == 0     18.576  < 2e-16 ***
## Gentoo - Chinstrap == 0   2.500  0.18063
```

Et les données manquantes ?

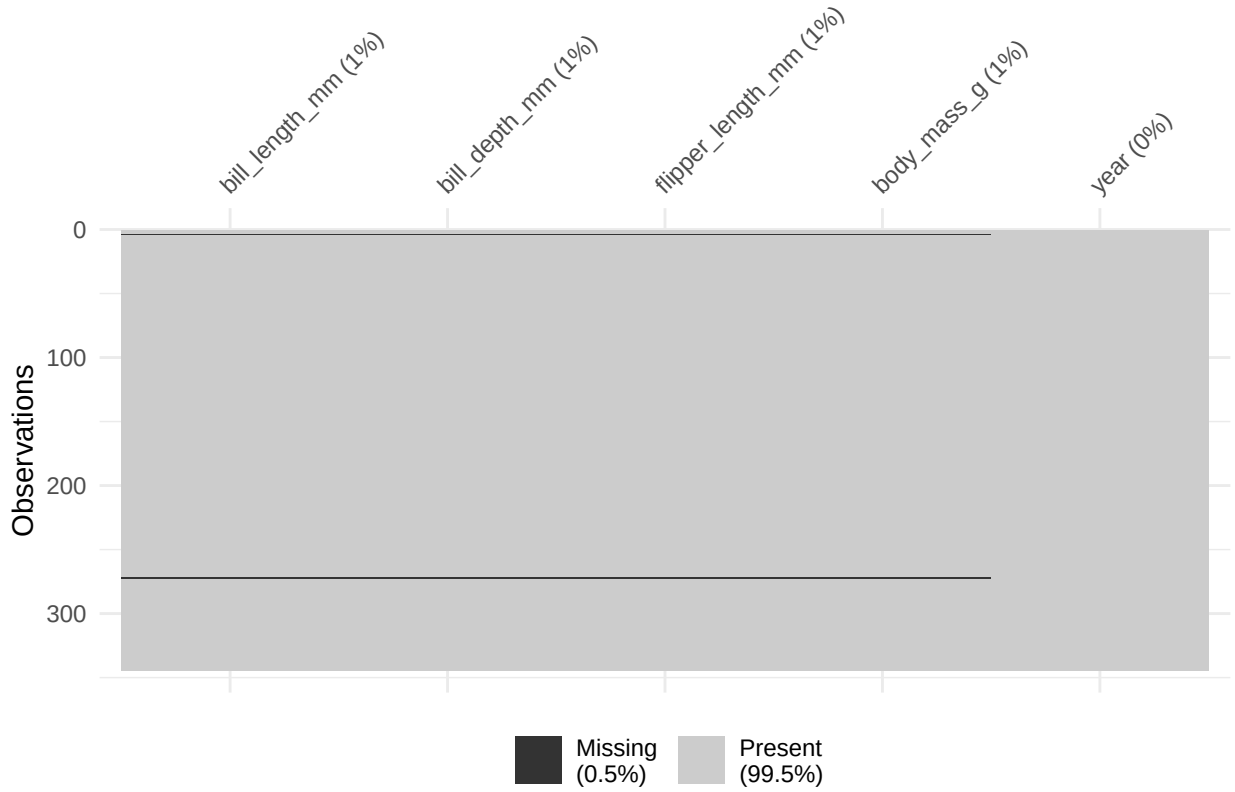
Il est possible de faire des imputations des données manquante via une analyse en composantes factorielles (ACP : uniquement quanti, AFM : quanti et quali, ACM : uniquement quali) avec le package `{missMDA}`.

```
library(missMDA)

# choix d'une ACP pour ne retrouver que les données des deux pingouins non mesurés
```

```
pingouin_num <-
  penguins |>
  dplyr::select(where(is.numeric))

visdat::vis_miss(pingouin_num)
```




```
test_donnees_manquantes <-
  MIPCA(
    pingouin_num,
    ncp = estim_ncpPCA(pingouin_num)$ncp
    )

test_donnees_manquantes$res.imputePCA |> summary()
```

```
##  bill_length_mm  bill_depth_mm   flipper_length_mm  body_mass_g
##  Min.   :32.10   Min.   :13.10   Min.   :172.0     Min.   :2700
##  1st Qu.:39.27   1st Qu.:15.60   1st Qu.:190.0     1st Qu.:3550
##  Median :44.45   Median :17.30   Median :197.0     Median :4050
##  Mean   :43.92   Mean   :17.15   Mean   :200.9     Mean   :4202
##  3rd Qu.:48.50   3rd Qu.:18.70   3rd Qu.:213.0     3rd Qu.:4750
##  Max.   :59.60   Max.   :21.50   Max.   :231.0     Max.   :6300
##       year
##  Min.   :2007
##  1st Qu.:2007
##  Median :2008
##  Mean   :2008
```

```
##  3rd Qu.:2009
##  Max.   :2009
```

En savoir un peu plus sur moi

Bonjour,

Je suis Marie Vaugoyeau et je suis disponible pour des **missions en freelance** d'**accompagnement à la formation** à R et à l'analyse de données et/ou en **programmation** (reprise de scripts, bonnes pratiques de codage, développement de package).
Ayant un **bagage recherche en écologie**, j'ai accompagné plusieurs chercheuses en biologie dans leurs analyses de données mais je suis ouverte à d'autres domaines.

Vous pouvez retrouver mes offres ici.

En plus de mes missions de consulting je diffuse mes savoirs en R et analyse de données sur plusieurs plateformes :

- J'ai écrit un **livre** aux éditions ENI
- Tous les mois je fais un **live sur Twitch** pour parler d'un package de R, d'une analyse
- Je rédige une **newsletter** de manière irrégulière pour parler de mes **inspirations** et transmettre **des trucs et astuces sur R**. Pour s'y inscrire, c'est par là. J'ai aussi un **blog**, en PLS en ce moment, qu'il faut que je reprenne.

Pour en savoir encore un peu plus sur moi, il y a LinkedIn et pour retrouver tous ces liens et plus encore, c'est ici

N'hésitez pas à me contacter sur marie.vaugoyeau@gmail.com !

Bonne journée

Marie