Session_5

Julian Maitra

Session 5

La session 5 a pour but d'approfondir les techniques de visualisation de données avec le paquet `ggplot2`, notamment la création des types de graphique suivants :

1. Les nuages de points (angl. *scatter plots*)
2. Les diagrammes à barres (angl. *bar plots*)
3. Les histogrammes (angl. *histograms*)
4. Les boîtes à moustaches (angl. *box plots*)

```
# N'oubliez pas de charger les paquets tidyverse et ggthemes !
# Exécutez le code suivant pour le faire :
library(tidyverse)
library(ggthemes)
```

1. Les nuages de points (angl. *scatter plots*)

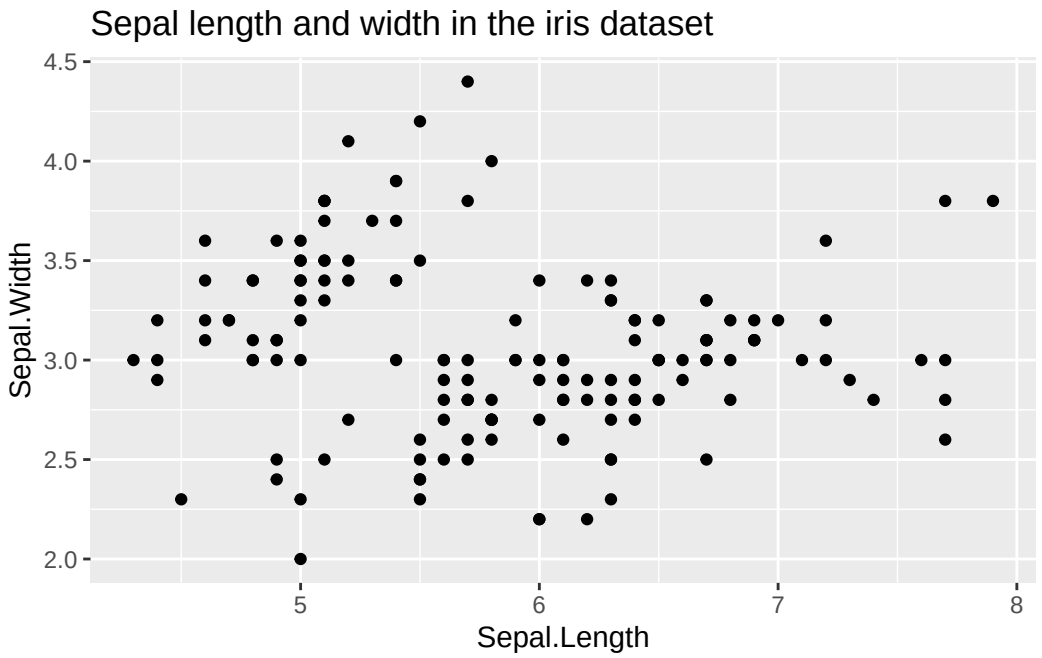
Commençons par le **nuage de points**, que nous avons déjà vu lors de la dernière session, lorsque nous avons visualisé la relation entre les carats et les prix dans le jeu de données `diamonds`.

> **i** À noter
>
> Un **nuage de points** (aussi appelé diagramme de dispersion) est utilisé dans `ggplot2` pour représenter les valeurs de **deux variables numériques différentes** par des points, avec une variable sur l'axe des X et l'autre sur l'axe des Y.
> Ce type de graphique est idéal pour l'analyse exploratoire de données, permettant d'observer les relations entre les variables, de détecter des tendances, des regroupements, des valeurs êxtremes (angl. *outliers*) ou de vérifier des hypothèses de corrélation.
> Dans la fonction `ggplot()`, un scatter plot est créé avec la couche géométrique `geom_point()`.

Créons un nuage de points avec le jeu de données `iris`, en comparant les longueurs et les largeurs des sépales des iris (les variables `Sepal.Length` et `Sepal.Width`).

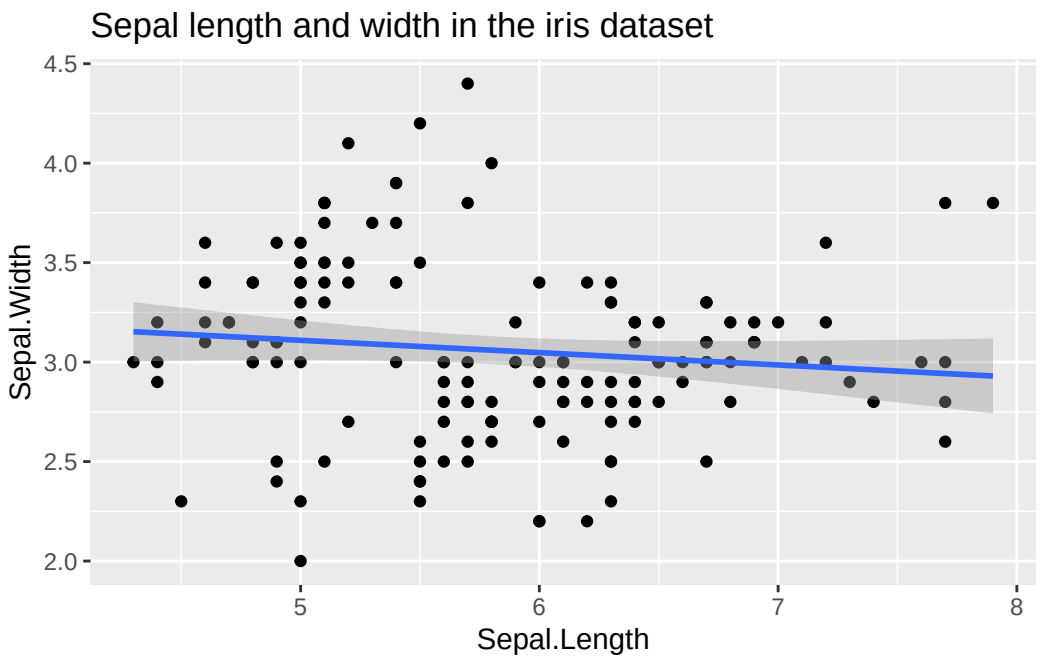
Notez que, dans le code suivant, nous ne créons pas directement un graphique `ggplot` mais nous stockons le graphique comme un objet dans l'environnement, nommé `p1`, que nous pouvons ensuite afficher avec la fonction `print()`. Nous verrons dans un instant pourquoi cette pratique est utile.

```
# nuage de points basique pour comparer la longueur et la largeur des sépales des iris
np1 <- ggplot(data = iris,
              mapping = aes(x=Sepal.Length, y=Sepal.Width)) +
  geom_point() +
  labs(title = "Sepal length and width in the iris dataset")

print(np1)
```



Nous pouvons ajouter des couches à l'objet `np1` sans l'écraser, p. ex. une ligne de régression linéaire avec `geom_smooth(method = lm)` :

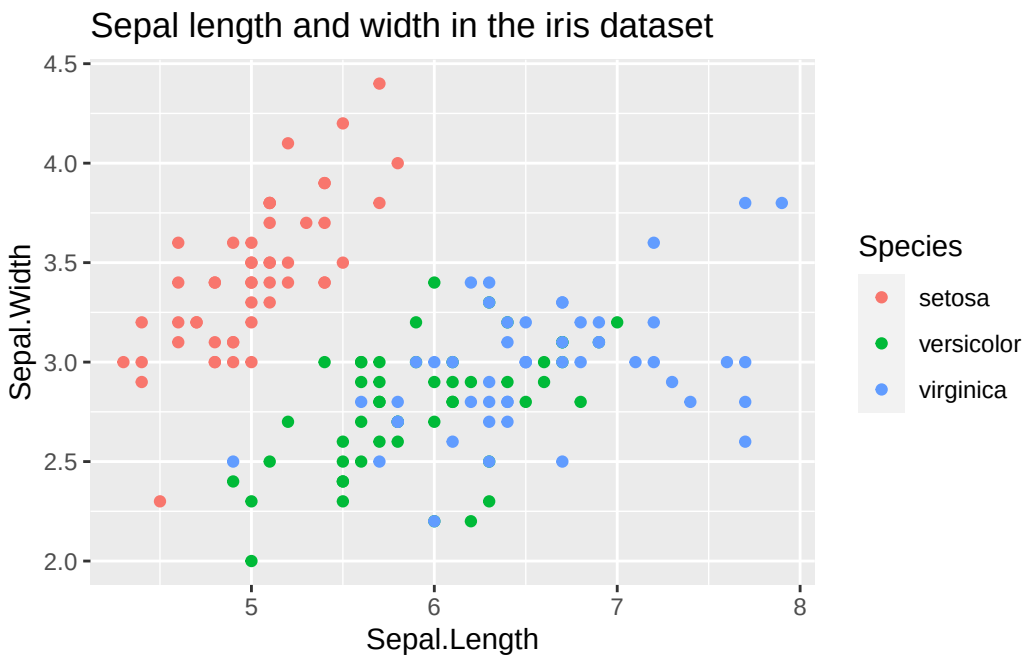
```
np1 + geom_smooth(method = lm)
```



Que nous indique la ligne de régression ? En principe, elle indique une correlation légèrement négative entre les variables `Sepal.Length`et `Sepal.Width` ! (la zone grise autour de la ligne bleue indique une marge d'erreur).

Cela signifie que les iris dont les sépales sont plus longs ont tendance à avoir des sépales un peu moins larges ! Gardez cette constatation à l'esprit pour les étapes de codage suivantes.

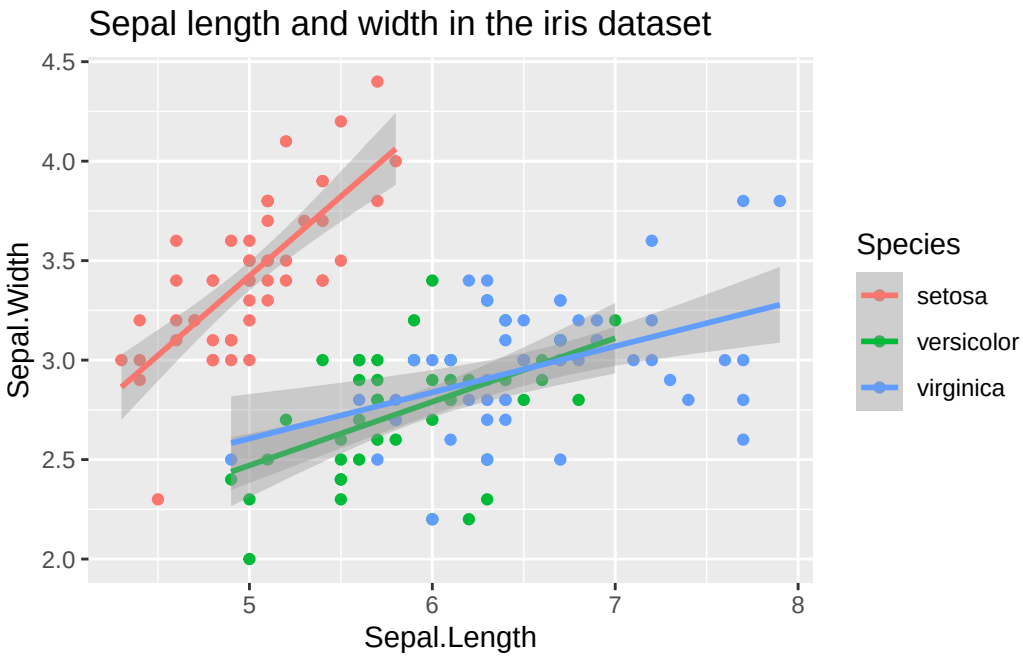
```
# Ecrasons np1 en changeant un peu le code. Découvrez vous-même ce qui a changé dans le code
np1 <- ggplot(iris,
              aes(x=Sepal.Length, y=Sepal.Width, color = Species)) +
  geom_point() +
  labs(title = "Sepal length and width in the iris dataset")

print(np1)
```



Vous pouvez voir dans le graphique que nous pouvons maintenant différencier les trois espèces d'iris. La dispersion des points semble moins aléatoire, ce qui permet de voir des groupes de points appartenant à l'une ou l'autre espèce (angl. *cluster*).

Maintenant, ajoutons à nouveau une couche `geom_smooth()` à l'objet `np1` !

```
np1 + geom_smooth(method = lm)
```



Les trois lignes de régression (une pour chaque type d'iris) montrent maintenant un tout autre résultat : en effet, la corrélation entre `Sepal.Length` et `Sepal.Width` est clairement positive.

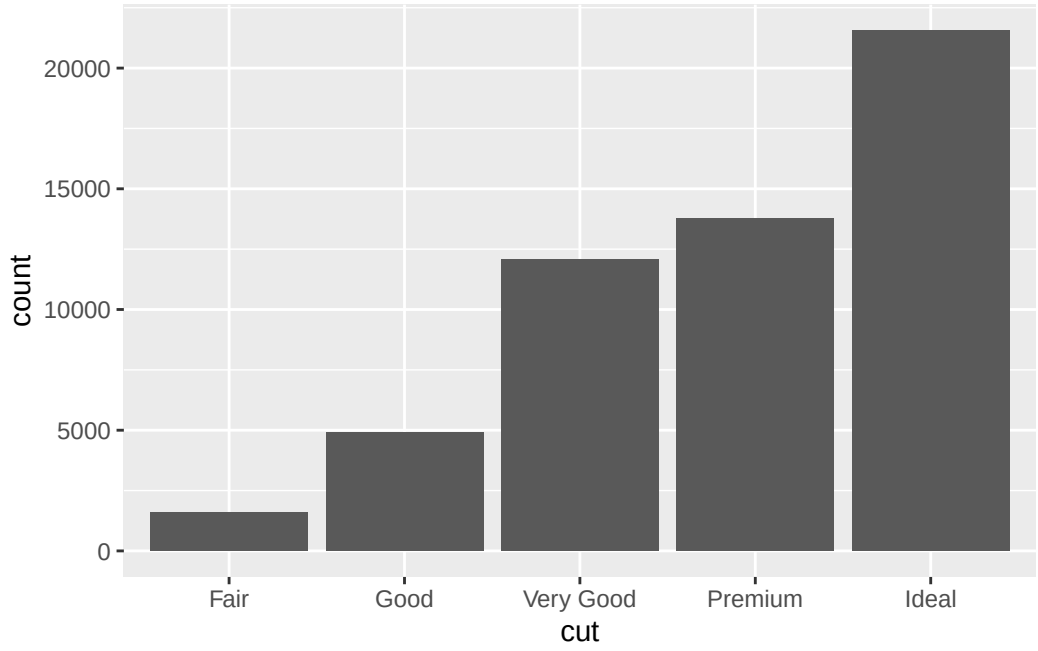
Cette constatation est un exemple d'analyse exploratoire des données, rendue possible par les possibilités de visualisation de `ggplot2`.

2. Les diagrammes à barres

> **i** À noter
>
> Les **diagrammes à barres** dans R avec `ggplot2` sont utilisés pour représenter visuellement des **données catégorielles**, permettant de comparer des quantités entre différentes catégories.
> On les utilise donc souvent pour montrer la fréquence ou la proportion d'observations dans chaque catégorie, ce qui les rend utiles pour analyser des tendances ou des différences au sein de données catégorielles.
> Dans la fonction `ggplot()`, un diagramme à barres est créé avec la couche géométrique `geom_bar()`.

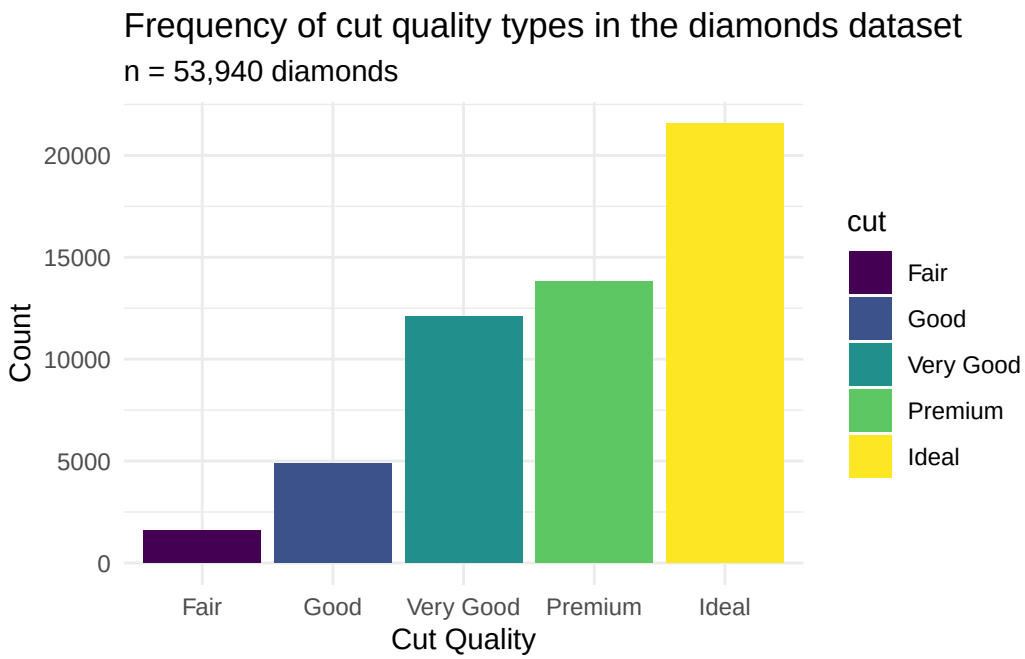
Par exemple, nous pouvons utiliser le code simple suivant pour comparer la fréquence des différentes tailles (angl. `cut`) dans le jeu de données `diamonds`. Cette-fois ci, nous sauvegardons et nommons le graphique `db1`.

```r
db1 <- ggplot(data = diamonds,
       mapping = aes(x = cut)) +
        geom_bar()

print(db1)
```



Améliorons le graphique avec des paramètres et des couches supplémentaires de la fonction `ggplot()` (Notez que nous allons simplement écraser l'objet `db1` créé précédemment avec une nouvelle version de `db1` !) :

```r
library(ggthemes)

db1 <- ggplot(data = diamonds,
       mapping = aes(x = cut, fill = cut)) +
        geom_bar() +
        labs(title = "Frequency of cut quality types in the diamonds dataset",
             subtitle = "n = 53,940 diamonds",
             x = "Cut Quality", y = "Count") +
        theme_minimal()

print(db1)
```



Que nous dit le graphique ? Il est intéressant de constater que dans le jeu de données `diamonds`, le niveau de qualité le plus élevé de la taille, à savoir `ideal`, est également le plus fréquent.

> **Attention**
>
> Dans la fonction `ggplot()`, il y a une différence si vous voulez ajouter de la couleur aux points (qui ont mathématiquement zéro dimension) ou aux surfaces (qui ont deux dimensions), comme dans les diagrammes à barres.
> Si vous voulez ajouter de la couleur à une surface, vous devez utiliser l'argument `fill = "xyz"`. L'argument `color = "xyz"` que nous avons utilisé pour ajouter de la couleur aux nuages de points ne fera qu'ajouter de la couleur à la ligne autour de la surface !

C'est maintenant à vous de jouer !

Reprenez le code que nous avons utilisé pour créer `db1` et modifiez-le pour créer un nouvel objet,`db2`, qui visualise la fréquence des types de qualité de couleur (`color`) dans le jeu de données `diamonds` sous la forme d'un diagramme à barres !

3. Les histogrammes

i À noter

Un **histogramme** est un type de graphique utilisé pour représenter **la distribution** d'**une variable numérique** à travers des barres.
Un "histogramme sépare les valeurs possibles des données en classes ou groupes. Pour chaque groupe, on construit un rectangle dont la base correspond aux valeurs de ce groupe et la hauteur correspond au nombre d'observations dans le groupe.
L'histogramme a une apparence semblable au graphique à barres verticales, mais il n'y a pas d'écart entre les barres.
En règle générale, l'histogramme possède des barres d'une largeur égale" (Source : Statistique Canada, 2021).
L'histogramme est particulièrement utile pour :

- **Analyser la distribution** : Comprendre si les données sont normalement distribuées, asymmétriques (angl. *skewness*), ou si elles présentent un plateau (kurtosis).
- **Détecter les valeurs extrêmes** : Identifier les données qui s'écartent du reste de l'ensemble des données de manière signifiante.
- **Comparer des distributions** : Observer comment différentes sous-populations se comparent les unes aux autres en termes de distribution de données.

Avec `ggplot2` en R, créer un histogramme se fait avec la couche géometrique `geom_histogram()`. `ggplot2` calcule automatiquement la taille des intervalles (angl. *bins*) par défaut, mais cela peut être ajusté manuellement en spécifiant l'argument `binwidth = xyz`dans la fonction `geom_histogram()`, p. ex. `geom_histogram(binwidth = 30)`.

Avant de créer un histogramme avec le code R, introduisons d'abord une nouvelle fonction : `rnorm()`. Elle peut être utilisée pour créer des valeurs aléatoires qui suivent une **loi normale** (angl. *normal distribution*).

Les arguments que nous pouvons entrer dans la fonction sont :

- le nombre `n` de valeurs aléatoires que nous voulons créer
- la moyenne `mean` de la distribution des valeurs aléatoires
- l'écart-type `sd` de la distribution des valeurs aléatoires

Notez que par défaut, la fonction utilise `mean = 0` et `sd = 1`, ce qui correspond à une **loi normale centrée réduite** (angl. *standard normal distribution*).

Utilisez aussi `?rnorm()`pour accéder à plus d'infos.

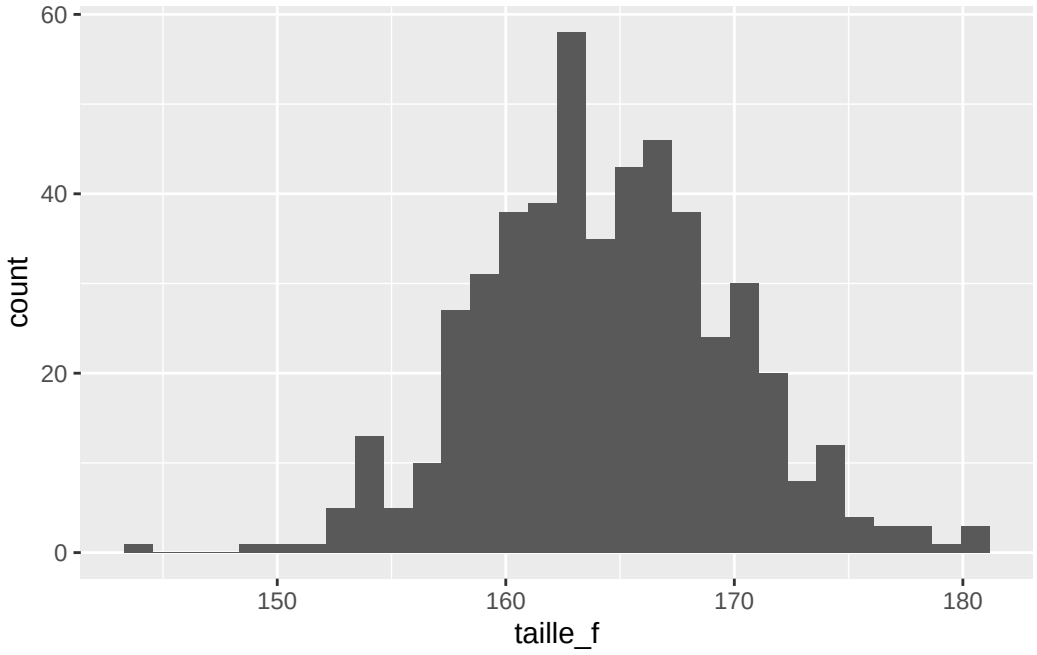
Dans le code R suivant, nous utilisons `rnorm()` pour simuler les distributions des tailles des hommes et des femmes en Suisse.

Selon un article de *24 heures*, en Suisse, les femmes ont une taille moyenne de 164.7 cm tandis que les hommes mesurent 177.4 cm en moyenne. Dans la suite, nous supposons que les écarts-types sont de +- 5.6 cm pour les femmes et de +- 6.1 cm pour les hommes.

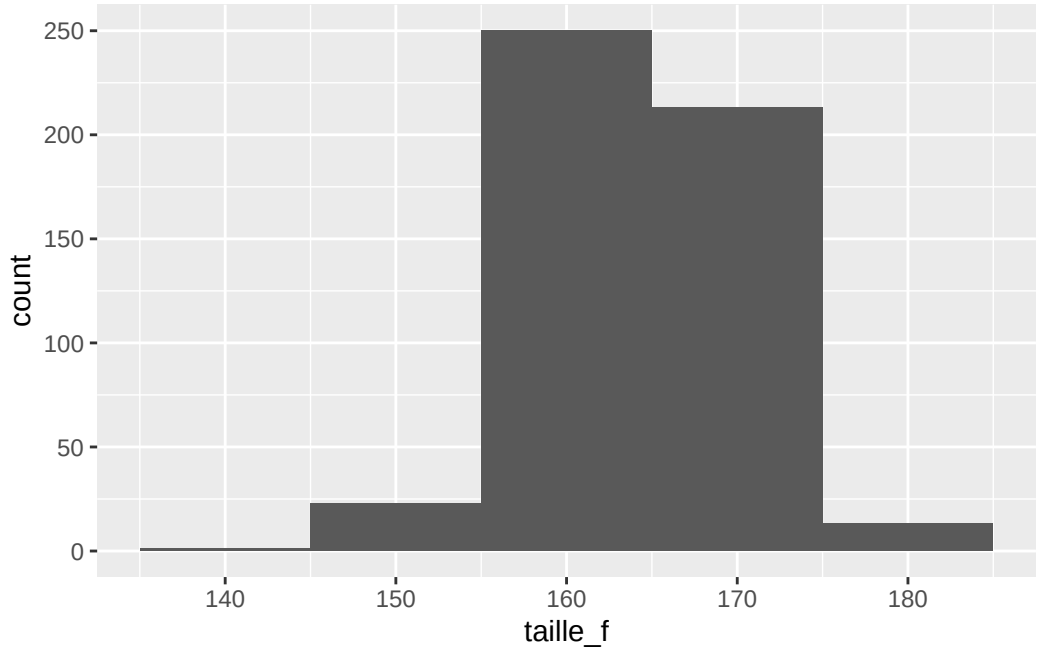
```
# Simulation de tailles de femmes en Suisse
# Créer 500 valeurs aléatoires de taille
# (moyenne = 164.7 et écart-type = 5.6) avec la fonction rnorm().
taille_f <- rnorm(500, mean = 164.7, sd = 5.6)

head(taille_f)
```

```
[1] 158.9821 157.7497 165.7133 162.7058 161.5693 157.9187
```

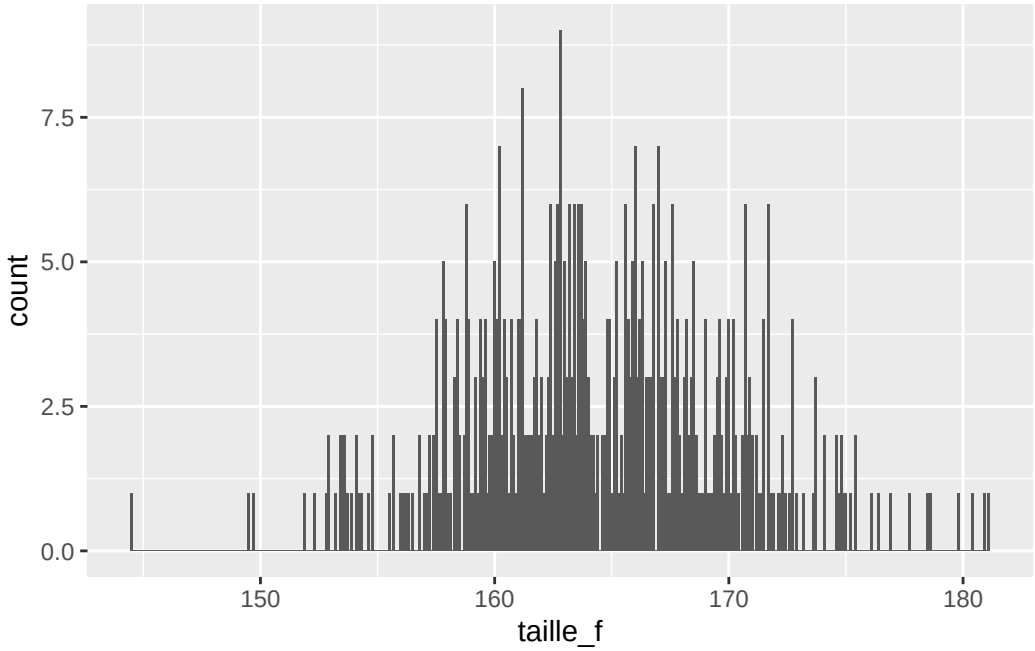
```
# stocker taille_f comme objet de type "data.frame"
taille_f <- as.data.frame(taille_f)

# visualiser taille_f avec un histogramme basique
ggplot(data = taille_f,
       mapping = aes(x=taille_f)) +
  geom_histogram()
```



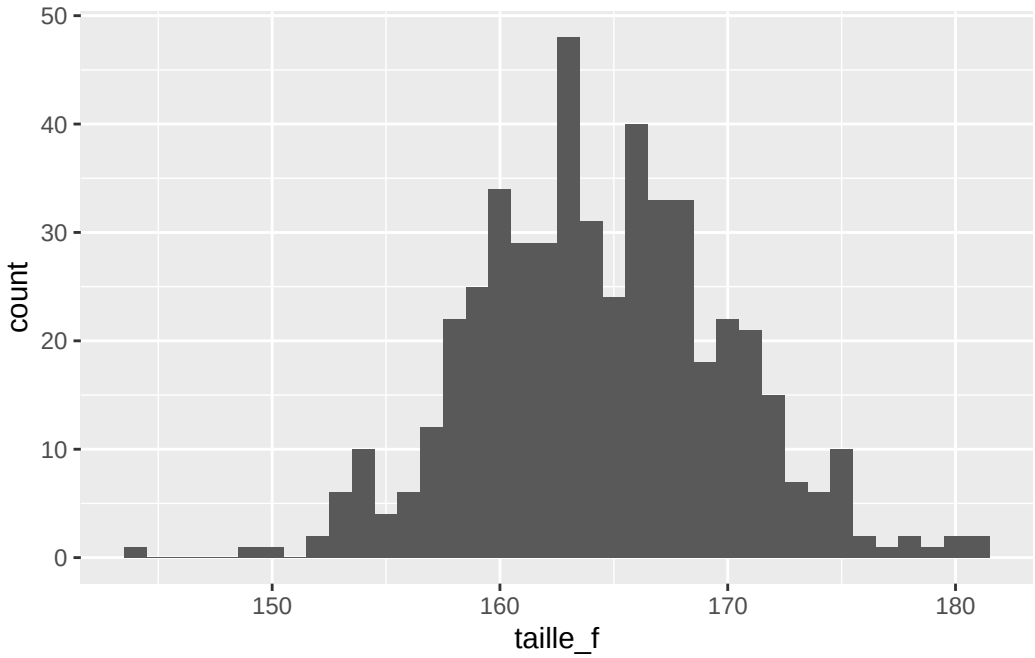
```r
# jouer avec le paramètre "binwidth" de l'histogramme
# qui définit l'intervalle de la base de ses barres
ggplot(data = taille_f,
       mapping = aes(x=taille_f)) +
  geom_histogram(binwidth = 10)
```



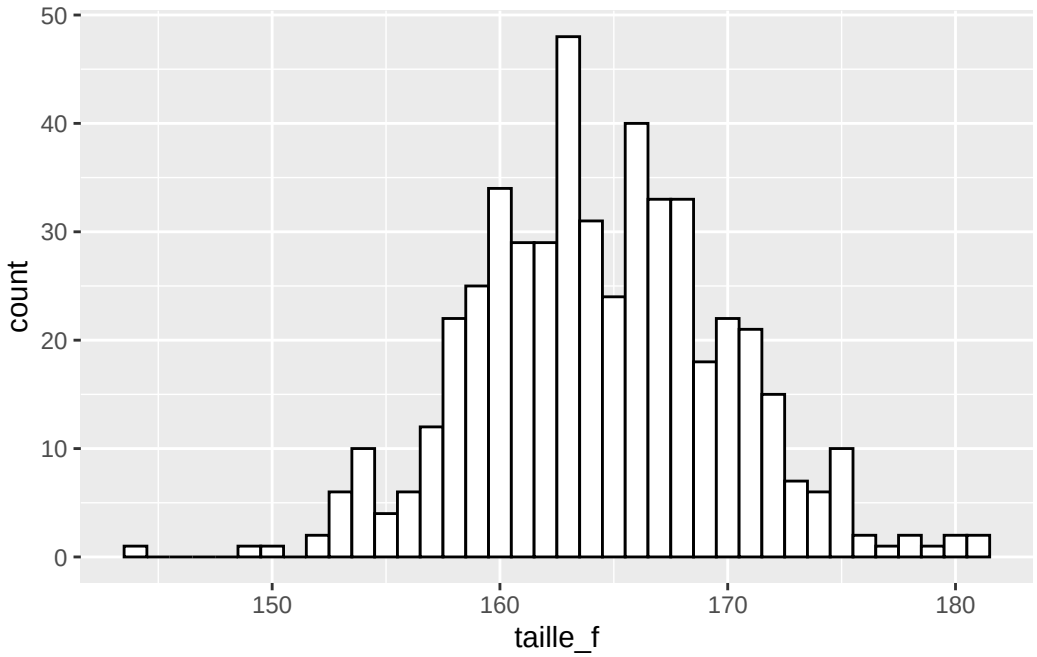

```r
ggplot(data = taille_f,
       mapping = aes(x=taille_f)) +
  geom_histogram(binwidth = 0.1)
```



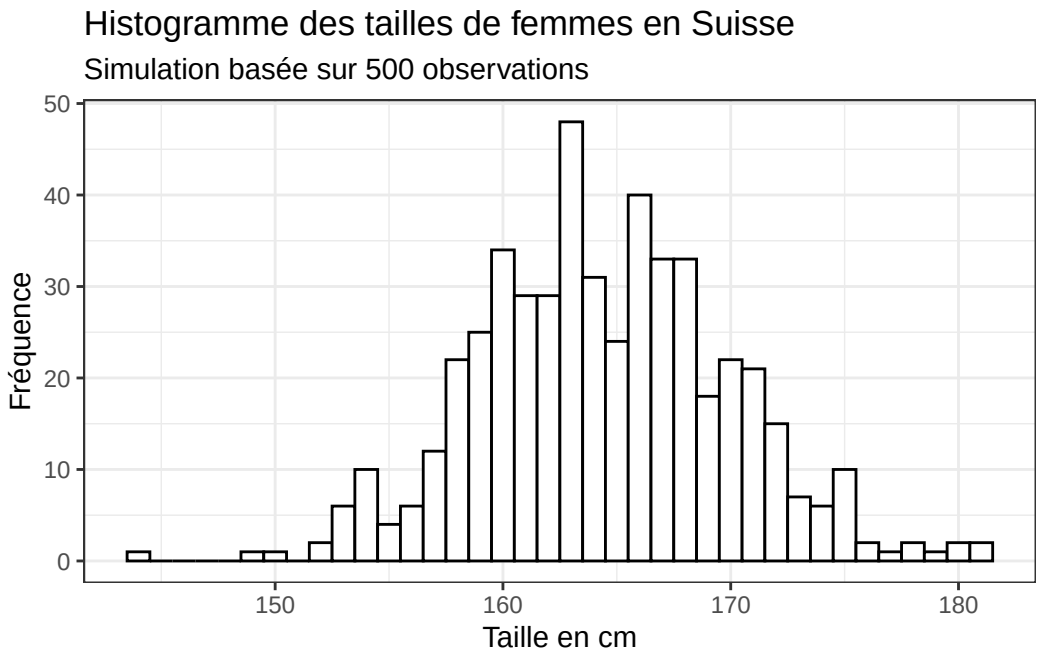

```
ggplot(data = taille_f,
       mapping = aes(x=taille_f)) +
  geom_histogram(binwidth = 1)
```



```r
# stocker l'histogramme comme objet, nommé hist1, et modifier sa couleur
hist1 <- ggplot(data = taille_f,
                mapping = aes(x=taille_f)) +
  geom_histogram(binwidth = 1, color = "black", fill = "white")

print(hist1)
```




```r
# Ajouter des couches supplémentaires pour améliorer la visualisation
# p. ex. des titres
hist1 <- hist1 + labs(title = "Histogramme des tailles de femmes en Suisse",
                      subtitle = "Simulation basée sur 500 observations",
                      x = "Taille en cm",
                      y = "Fréquence") +
        theme_bw()

print(hist1)
```



Astuce

Un élément clé des visualisations de données est la **couleur**. Le document PDF suivant, créé par Ying Wei de l'Université de Columbia à New York, montre et liste les noms des couleurs dans R. Vous pouvez copier-coller le nom d'une couleur et l'utiliser dans votre graphique ggplot2 !

- Colors in R

C'est maintenant à vous de jouer !

Reprenez le code que nous avons utilisé pour créer `taille_f` et `hist1` et modifiez-le pour créer une nouvelle simulation de taille d'hommes, `taille_h`, et un nouveau histogramme qui montre sa distribution, `hist2`. Jouez avec les paramètres `binwidth`et la couleur de votre visualisation. Pour cette tâche, vous pouvez supposer que les hommes en Suisse ont une taille moyenne (`mean`) de 177,4 cm avec un écart-type (`sd`) de 6,1 cm.

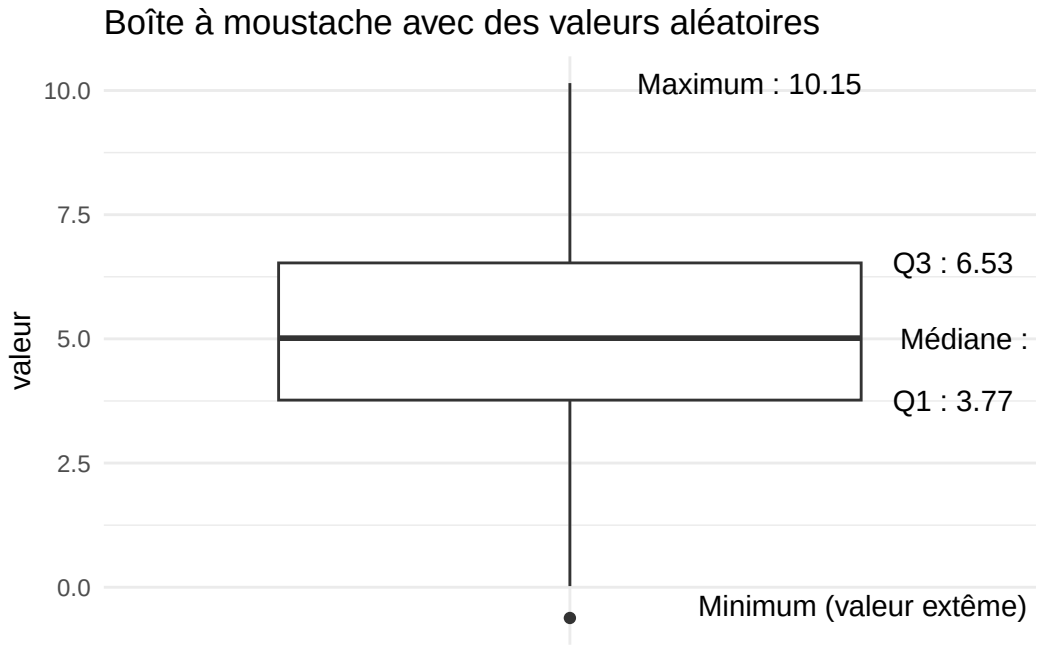
4. Les boîtes à moustaches (angl. *boxplots*)

À noter

Les **boîtes à moustaches**, ou **boxplots**, sont un type de graphique pour représenter la distribution d'un ensemble de données à travers cinq mesures clés :

1. le minimum (début de la "moustache" ou point extrême de minimum)
2. le premier quartile (Q1 ; début de la boîte)
3. la médiane (Q2 ; ligne qui "coupe la boîte en deux")
4. le troisième quartile (Q3 ; fin de la boîte) et
5. le maximum (fin de la moustache ou point extrême de maximum)

Ces diagrammes permettent d'identifier rapidement la médiane, l'étendue interquartile (angl. *interquartile range* = IQR) et les valeurs êxtremes potentielles. Ils sont particulièrement utiles pour comparer les distributions entre plusieurs groupes ou variables.



Le dataset `mtcars` en R, qui contient des données sur les caractéristiques de différents modèles de voitures, peut être utilisé pour illustrer l'utilisation de boxplots.

Par exemple, pour visualiser la distribution de la consommation de carburant (`mpg`) des voitures en fonction du nombre de cylindres (`cyl`), on peut utiliser le code suivant :

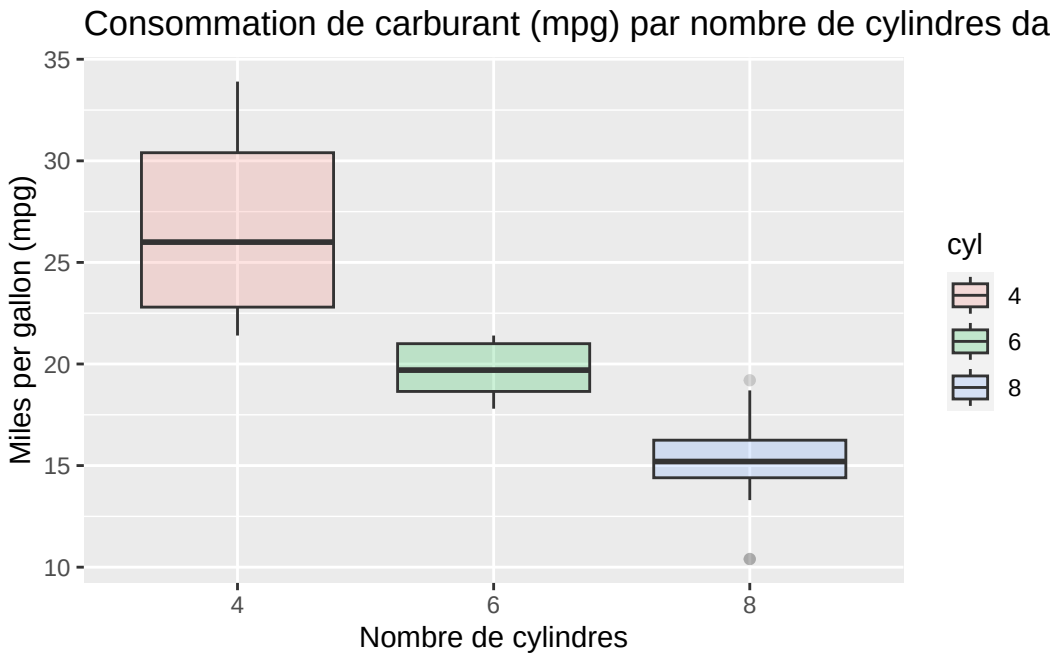
```r
# Avant de créer le boxplot, vérifions d'abord la structure de mtcars :
str(mtcars)
```

```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Nous pouvons constater que la colonne `cyl` (cylindre) est stockée au format numérique.

C'est un problème, car nous voulons comparer la consommation de carburant des voitures à 4, 6 ou 8 cylindres. Nous devons traiter chacune d'entre elles comme un sous-groupe distinct. Comment faire ?

```r
# Transformons donc cyl en facteur
mtcars$cyl <- as.factor(mtcars$cyl)

# Et maintenant nous pouvons construire notre boxplot
box1 <- ggplot(data = mtcars,
          aes(x=cyl, y=mpg, fill = cyl)) +
            geom_boxplot(alpha=0.2) +
              labs(title = "Consommation de carburant (mpg) par nombre de cylindres dans mtca
print(box1)
```



Ce code génère un boxplot, `box1`, pour chaque groupe de cylindres dans le dataset `mtcars`, permettant de comparer facilement la consommation de carburant entre les voitures avec un différents nombres de cylindres.

> **i** À noter
>
> le paramètre `alpha = 0.2` à l'intérieur de `geom_boxplot()` rend les boxplots transparents. En général, `alpha` peut être compris entre 0 (totalement transparent) et 1 (non transparent). C'est un paramètre supplémentaire qui permet de personnaliser votre graphique `ggplot2`.

On voit dans le graphique des boxplots que dans `mtcars`, les voitures avec des moteurs ayant plus de cylindres ont aussi tendance à être moins efficaces en termes de consommation de carburant. Il existe toutefois quelques intersections entre les groupes. C'est-à-dire qu'il y a des voitures à 8 cylindres qui sont relativement efficaces en termes de consommation de carburant, surpassant quelques voitures à 6 cylindres, par exemple.

Devoir pour Session 6 (Problem Set 2)

- **Problem Set 2**
  - Comme le premier Problem Set, le Problem Set 2 est un QCM sur Moodle qui sera accessible pendant 30 minutes du créneau horaire de votre groupe de l'exercice.

  - Assurez-vous que R et RStudio fonctionnent sur votre ordinateur portable et qu'il a suffisamment de batterie. Il est de votre responsabilité de vous en assurer.
  - Le Problem Set 2 concernera principalement le contenu des sessions 4 et 5, à savoir la visualisation de données avec ggplot2.
  - Les bases de R qui ont été traitées dans les premières sessions, comme par exemple la création et l'écrasement d'objets avec l'opérateur d'affectation `<-`, sont toutefois requises, car la matière se fonde sur ces bases. Veuillez combler vous-même les éventuelles lacunes à ce sujet.

- **Annonce : Questionnaire**
  - Nous allons évaluer **un questionnaire sur votre comportement d'utilisation des médias** avec R comme autre élément important du cours.
  - Le questionnaire est **anonyme** (même pour l'enseignant) et peut être rempli en ligne via LimeSurvey jusqu'au **vendredi 19 avril 2024 à 23h59**.
  - Tous les étudiant.es recevront prochainement un lien personnel d'invitation au questionnaire par e-mail (les réponses seront anonymisées et mélangées par le logiciel, l'enseignant ne verra que si les personnes invitées ont répondu au questionnaire avant la date limite).
  - Chaque étudiant(e) qui aura rempli le questionnaire dans les délais recevra un **bonus de 2 points dans le Problem Set 4** (10% des points).