Les statistiques avec R

Marc-André Désautels

2024-06-09

Table des matières

Préface

Ceci est un livre Quarto.

Pour en apprendre davantage sur Quarto, visitez https://quarto.org/docs/books.

partie I

Les probabilités et la combinatoire

1 La combinatoire

1.1 La factorielle

Pour calculer la factorielle d'un nombre en `R`, il faut utiliser la commande `factorial`. Par exemple, si nous voulons calculer 6!:

```
factorial(6)
#> [1] 720
```

1.2 Les combinaisons

Pour calculer le nombre de combinaisons lorsque nous choisissons k objets parmi n (**sans** ordre), c'est-à-dire C_k^n, nous utilisons la commande `choose(n,k)`. Par exemple, si nous voulons calculer le nombre de combinaisons possibles au loto 6-49, C_6^{49}, nous avons:

```
choose(49,6)
#> [1] 13983816
```

1.3 Les arrangements

Pour calculer le nombre d'arrangements lorsque nous choisissons k objets parmi n (**avec** ordre), c'est-à-dire A_k^n, nous utilisons les commandes `choose(n,k)` et `factorial`. En effet, nous savons que:

$$A_k^n = C_k^n \cdot k! \tag{1.1}$$

et donc on peut calculer un arrangement en effectuant `choose(n,k)*factorial(k)`. Si nous voulons calculer le nombre de comités de 5 personnes nous pouvons former en choisissant parmi 12 personnes, A_5^{12}, nous avons:

```r
choose(12,5)*factorial(5)
#> [1] 95040
```

2 Les lois de probabilités

Pour être en mesure d'utiliser les lois de probabilités en langage `R`, il faut charger le paquetage `stats`.

```
library(stats)
library(ggplot2)
```

Chaque distribution en `R` possède quatre fonctions qui lui sont associées. Premièrement, la fonction possède un *nom racine*, par exemple le *nom racine* pour la distribution *binomiale* est `binom`. Cette racine est précédée par une de ces quatre lettre:

- `p` pour *probabilité*, qui représente la fonction de répartition
- `q` pour *quantile*, l'inverse de la fonction de répartition
- `d` pour *densité*, la fonction de densité de la distribution
- `r` pour *random*, une variable aléatoire suivant la distribution spécifiée.

Pour la loi binomiale par exemple, ces fonctions sont `pbinom`, `qbinom`, `dbinom` et `rbinom`.

2.1 Les lois de probabilités discrètes

2.1.1 La loi binomiale

Le *nom racine* pour la loi binomiale est `binom`.

Soit X: le nombre de succès en n essais et $X \sim B(n, p)$. Voici la façon de calculer des probabilités pour la loi binomiale à l'aide de `R`:

Probabilités	Commande `R`
$P(X = k)$	`dbinom(k, n, p)`
$P(i \leq X \leq j)$	`sum(dbinom(i:j, n, p))`
$P(X \leq k)$	`pbinom(k, n, p)`
$P(X > k)$	`1-pbinom(k, n, p)`

Soit X la variable aléatoire comptant le nombre de face 2 que nous obtenons en lançant un dé à quatre reprises. Nous avons que $X \sim B(4, \frac{1}{6})$. Si nous voulons calculer $P(X = 3)$, nous aurons:

```
dbinom(3,4,1/6)
#> [1] 0.0154321
```

Nous avons donc une probabilité de 1.5432099% d'obtenir 3 fois la face deux en lançant un dé à quatres reprises.

2.1.2 La loi de Poisson

Le *nom racine* pour la loi de Poisson est `pois`.

Soit X: le nombre d'événements dans un intervalle fixé et $X \sim Po(\lambda)$. Voici la façon de calculer des probabilités pour la loi de Poisson à l'aide de `R`:

Probabilités	Commande `R`
$P(X = k)$	`dpois(k, lambda)`
$P(i \leq X \leq j)$	`sum(dpois(i:j, lambda))`
$P(X \leq k)$	`ppois(k, lambda)`
$P(X > k)$	`1-ppois(k, lambda)`

Soit X le nombre d'erreurs dans une page. Si une page contient en moyenne une demie erreur alors $X \sim Po(1/2)$. Si nous voulons calculer $P(X = 2)$, nous aurons:

```
dpois(2, 1/2)
#> [1] 0.07581633
```

Nous avons donc une probabilité de 7.5816332% d'obtenir deux erreurs sur une page.

2.1.3 La loi géométrique

Le *nom racine* pour la loi géométrique est `geom`.

Soit X: le nombre d'échecs avant d'obtenir un succès et $X \sim G(p)$. Voici la façon de calculer des probabilités pour la loi géométrique à l'aide de `R`:

Probabilités	Commande `R`
$P(X = k)$	`dgeom(k, p)`
$P(i \leq X \leq j)$	`sum(dgeom(i:j, p))`
$P(X \leq k)$	`pgeom(k, p)`
$P(X > k)$	`1-pgeom(k, p)`

Soit X le nombre d'échecs avant d'avoir un premier succès. Si la probabilité de succès est $\frac{1}{5}$ alors $X \sim G(1/5)$. Si nous voulons calculer $P(X = 6)$, nous aurons:

```
dgeom(6, 1/5)
#> [1] 0.0524288
```

Nous avons donc une probabilité de 5.24288% d'obtenir 6 échecs avant un premier succès.

> Remarque : Pour la loi géométrique, on rencontre parfois cette définition : la probabilité p'(k) est la probabilité, lors d'une succession d'épreuves de Bernoulli indépendantes, d'obtenir k échecs avant un succès. On remarque qu'il ne s'agit que d'un décalage de la précédente loi géométrique. Si X suit la loi p, alors $X + 1$ suit la loi p'.

2.1.4 La loi hypergéométrique

Le *nom racine* pour la loi hypergéométrique est `hyper`.

On tire sans remise n objets d'un ensemble de N objets dont A possèdent une caractéristique particulière (et les autres $B = N - A$ ne la possèdent pas). Soit X le nombre d'objets de l'échantillon qui possèdent la caractéristique. Nous avons que $X \sim H(N, A, n)$.

Voici la façon de calculer des probabilités pour la loi hypergéométrique à l'aide de `R`:

Probabilités	Commande `R`
$P(X = k)$	`dhyper(k, A, B, n)`
$P(i \leq X \leq j)$	`sum(dhyper(i:j, A, B, n))`
$P(X \leq k)$	`phyper(k, A, B, n)`
$P(X > k)$	`1-phyper(k, A, B, n)`

Soit X le nombre de boules blanches de l'échantillon de taille 4. Si l'urne contient 5 boules blanches et 8 boules noires, nous avons $X \sim H(13, 5, 4)$. Si nous voulons calculer $P(X = 2)$, nous aurons:

```r
dhyper(2, 5, 8, 4)
#> [1] 0.3916084
```

Nous avons donc une probabilité de 39.1608392% de piger 2 boules blanches dans un échantillon de taille 4.

2.2 Les lois de probabilités continues

2.2.1 La loi normale

Le *nom racine* pour la loi normale est `norm`.

Si X suit une loi normale de moyenne μ et de variance σ^2, nous avons $X \sim N(\mu, \sigma^2)$.

Voici la façon de calculer des probabilités pour la loi normale à l'aide de `R`:

Probabilités	Commande `R`
$P(i \leq X \leq j)$	`pnorm(j, mu, sigma)-pnorm(i, mu, sigma)`
$P(X \leq k)$	`pnorm(k, mu, sigma)`
$P(X > k)$	`1-pnorm(k, mu, sigma)`

Soit $X \sim N(3, 25)$ une variable aléatoire suivant une loi normale de moyenne 3 et de variance 25. Si nous voulons calculer la probabilité $P(1.25 < X < 3.6)$ en `R`, nous pouvons utiliser la commande suivante:

```r
pnorm(3.6, 3, 5) - pnorm(1.25, 3, 5)
#> [1] 0.1845891
```

La probabilité que notre variable aléatoire se trouve entre 1.25 et 3.6 est donc 18.4589077 %.

2.2.2 La loi de Student

Le *nom racine* pour la loi de Student est `t`.

Si X suit une loi de Student à ν degrés de liberté, nous avons $X \sim T_\nu$.

Voici la façon de calculer des probabilités pour la loi de Student à l'aide de `R`:

Probabilités	Commande `R`
$P(i \leq X \leq j)$	`pt(j, nu)-pt(i, nu)`
$P(X \leq k)$	`pt(k, nu)`
$P(X > k)$	`1-pt(k, nu)`

Soit $X \sim T_5$ une variable aléatoire suivant une loi de Student à 5 degrés de liberté. Si nous voulons calculer la probabilité $P(X > 3)$ en `R`, nous pouvons utiliser la commande suivante:

```
1 - pt(3, 5)
#> [1] 0.01504962
```

La probabilité que notre variable aléatoire soit plus grande que 3 est donc 1.5049624 %.

partie II

Les statistiques descriptives

3 Les tableaux

Une fois les données d'un sondage recueillies, il est plus aisé d'analyser ces données si elles sont classées dans un tableau.

Nous utiliserons l'extension `nycflights13` avec les bases de données `planes`, `weather` et `flights` pour montrer la création de tableaux en `R`.

```
library(nycflights13)
data(planes)
data(weather)
data(flights)
```

3.1 Tableau de fréquences à une variable

3.1.1 Les variables qualitatives

Le tableau de fréquences que nous utiliserons est le suivant:

Titre		
Nom de la variable (*Modalités*)	**Nombre d'unités statistiques** (*Fréquences absolues*)	**Pourcentage d'unités statistiques** (%) (*Fréquences relatives*)
Total	n	100%

> Important : Le titre doit toujours être indiqué lors de la construction d'un tableau de fréquence.

Lorsque les données se trouvent dans une `tibble` dans `R`, il est possible d'utiliser la commande `freq` de la librairie `questionr` pour afficher le tableau de fréquences. La commande `freq` prend comme argument la variable dont vous voulez produire le tableau de fréquences. Pour obtenir une sortie adéquate, il faut ajouter trois options à la commande:

- `cum = FALSE`; permet de ne pas afficher les pourcentages cumulés
- `valid = FALSE`; permet de ne pas afficher les données manquantes

- `total = TRUE`; permet d'afficher le total

Dans la base de données `nycflights13::planes`, nous allons afficher la variable `engine`, qui correspond au type de moteur de l'avion.

```
freq(planes$engine,cum = FALSE,valid = FALSE,total = TRUE)
#>                   n     %
#> 4 Cycle           2   0.1
#> Reciprocating    28   0.8
#> Turbo-fan      2750  82.8
#> Turbo-jet       535  16.1
#> Turbo-prop        2   0.1
#> Turbo-shaft       5   0.2
#> Total          3322 100.0
```

3.1.2 Les variables quantitatives discrètes

Le tableau de fréquences que nous utiliserons est le suivant :

Titre			
Nom de la variable (*Valeurs*)	**Nombre d'unités statistiques** (*Fréquences absolues*)	**Pourcentage d'unités statistiques** (%) (*Fréquences relatives*)	**Pourcentage cumulé** (*Fréquences relatives cumulées*)
Total	n	100%	

Le pourcentage cumulé permet de déterminer le pourcentage des répondants qui ont indiqué la valeur correspondante, ou une plus petite. Il sert à donner une meilleure vue d'ensemble.

> Si pour la valeur x_i de la variable A la pourcentage cumulé est de b %, ceci signifie que b % des valeurs de la variable A sont plus petites ou égales à x_i.

La commande `freq` prend comme argument la variable dont vous voulez produire le tableau de fréquences. Pour obtenir une sortie adéquate, il faut ajouter trois options à la commande:

- `cum = TRUE`; permet d'afficher les pourcentages cumulés
- `valid = FALSE`; permet de ne pas afficher les données manquantes
- `total = TRUE`; permet d'afficher le total

Dans la base de données `nycflights13::planes`, nous allons afficher la variable `engines`, qui correspond au nombre de moteurs de l'avion.

```
freq(planes$engines,cum = TRUE,valid = FALSE,total = TRUE)
#>           n     %  %cum
#> 1        27   0.8   0.8
#> 2      3288  99.0  99.8
#> 3         3   0.1  99.9
#> 4         4   0.1 100.0
#> Total  3322 100.0 100.0
```

Dans la base de données `nycflights13::planes`, nous allons afficher la variable `seats`, qui correspond au nombre de sièges de l'avion.

```
freq(planes$seats,cum = TRUE,valid = FALSE,total = TRUE)
#>           n     %  %cum
#> 2        16   0.5   0.5
#> 4         5   0.2   0.6
#> 5         2   0.1   0.7
#> 6         3   0.1   0.8
#> 7         2   0.1   0.8
#> 8         5   0.2   1.0
#> 9         1   0.0   1.0
#> 10        1   0.0   1.1
#> 11        2   0.1   1.1
#> 12        1   0.0   1.1
#> 14        1   0.0   1.2
#> 16        1   0.0   1.2
#> 20       80   2.4   3.6
#> 22        2   0.1   3.7
#> 55      390  11.7  15.4
#> 80       83   2.5  17.9
#> 95      123   3.7  21.6
#> 100     102   3.1  24.7
#> 102       1   0.0  24.7
#> 128       1   0.0  24.7
#> 139       8   0.2  25.0
#> 140     411  12.4  37.4
#> 142     158   4.8  42.1
#> 145      57   1.7  43.8
#> 147       3   0.1  43.9
#> 149     452  13.6  57.5
#> 172      81   2.4  60.0
#> 178     283   8.5  68.5
#> 179     134   4.0  72.5
```

```
#> 182    159   4.8  77.3
#> 189     73   2.2  79.5
#> 191     87   2.6  82.1
#> 199     43   1.3  83.4
#> 200    256   7.7  91.1
#> 222     13   0.4  91.5
#> 255     16   0.5  92.0
#> 260      4   0.1  92.1
#> 269      1   0.0  92.1
#> 275     25   0.8  92.9
#> 290      6   0.2  93.1
#> 292     16   0.5  93.6
#> 300     17   0.5  94.1
#> 330    114   3.4  97.5
#> 375      1   0.0  97.5
#> 377     14   0.4  98.0
#> 379     55   1.7  99.6
#> 400     12   0.4 100.0
#> 450      1   0.0 100.0
#> Total 3322 100.0 100.0
```

Comme nous pouvons le constater, le tableau est très grand car la variable `seats` possède 48 valeurs différentes.

3.1.3 Les variables quantitatives continues

Le tableau de fréquences que nous utiliserons est le suivant :

Titre			
Nom de la variable (*Classes*)	**Nombre d'unités statistiques** (*Fréquences absolues*)	**Pourcentage d'unités statistiques** (%) (*Fréquences relatives*)	**Pourcentage cumulé** (*Fréquences relatives cumulées*)
Total	n	100%	

Pour être en mesure de briser une variable en classes, il faut utiliser la commande `cut`.

Les options de `cut` sont:

- `include.lowest = TRUE` qui permet d'avoir un intervalle fermé à droite et ouvert à gauche;

- `breaks` qui permet d'indiquer à quel endroit on doit créer les classes;
- `seq(from = A, to = B, by = C)` permet de créer un vecteur comportant les valeurs de `A` jusqu'à `B` en faisant des bonds de `C`.

Pour simplifier le code, nous créons en premier lieu une variable `air_time_rec` avec les classes et nous l'affichons ensuite avec `freq`. Remarquons que nous avons ajouté l'option `valid = TRUE` car certaines valeurs sont manquantes. Rappelons que les données manquantes sont représentées par `NA` en `R`. Deux colonnes sont ajoutées:

- `val%`: le pourcentage en omettant les valeurs manquantes
- `val%cum`: le pourcentage cumulé en omettant les valeurs manquantes

Nous obtenons donc:

```
air_time_rec <- cut(flights$air_time,
                    right=FALSE,
                    breaks=seq(from = 0, to = 700, by = 100))
freq(air_time_rec,cum = TRUE,total = TRUE,valid = TRUE)
#>                n     %  val%  %cum val%cum
#> [0,100)   105687  31.4  32.3  31.4    32.3
#> [100,200) 146527  43.5  44.8  74.9    77.0
#> [200,300)  31036   9.2   9.5  84.1    86.5
#> [300,400)  43347  12.9  13.2  97.0    99.8
#> [400,500)     48   0.0   0.0  97.0    99.8
#> [500,600)    132   0.0   0.0  97.0    99.8
#> [600,700)    569   0.2   0.2  97.2   100.0
#> NA          9430   2.8    NA 100.0      NA
#> Total     336776 100.0 100.0 100.0   100.0
```

3.2 Tableau de fréquences à deux variables

Faire une analyse bivariée, c'est étudier la relation entre deux variables : sont-elles liées ? les valeurs de l'une influencent-elles les valeurs de l'autre ? ou sont-elles au contraire indépendantes ?

À noter qu'on va parler ici d'influence ou de lien, mais pas de relation de cause à effet. Les outils présentés permettent de visualiser ou de déterminer une relation, mais la mise en évidence de liens de causalité proprement dit est nettement plus complexe : il faut en effet vérifier que c'est bien telle variable qui influence telle autre et pas l'inverse, qu'il n'y a pas de "variable cachée", etc.

Là encore, le type d'analyse ou de visualisation est déterminé par la nature qualitative ou quantitative des deux variables.

3.2.1 Croisement de deux variables qualitatives

On continue à travailler avec le jeu de données tiré de l'enquête *Histoire de vie* inclus dans l'extension `questionr`. On commence donc par charger l'extension, le jeu de données, et à le renommer en un nom plus court pour gagner un peu de temps de saisie au clavier.

```
library(questionr)
data(hdv2003)
d <- hdv2003
```

Quand on veut croiser deux variables qualitatives, on fait un *tableau croisé*. Comme pour un tri à plat ceci s'obtient avec la fonction `table` de R, mais à laquelle on passe cette fois deux variables en argument. Par exemple, si on veut croiser la catégorie socio-professionnelle et le sexe des enquêtés :

```
table(d$qualif, d$sexe)
#>
#>                            Homme Femme
#>   Ouvrier specialise          96   107
#>   Ouvrier qualifie           229    63
#>   Technicien                  66    20
#>   Profession intermediaire    88    72
#>   Cadre                      145   115
#>   Employe                     96   498
#>   Autre                       21    37
```

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

Pour calculer les pourcentages ligne :

```
tab <- table(d$qualif, d$sexe)
lprop(tab)
#>
#>                            Homme Femme Total
#>   Ouvrier specialise        47.3  52.7 100.0
#>   Ouvrier qualifie          78.4  21.6 100.0
#>   Technicien                76.7  23.3 100.0
#>   Profession intermediaire  55.0  45.0 100.0
#>   Cadre                     55.8  44.2 100.0
#>   Employe                   16.2  83.8 100.0
```

```
#>   Autre                     36.2  63.8 100.0
#>   All                       44.8  55.2 100.0
```

Et pour les pourcentages colonne :

```
cprop(tab)
#>
#>                           Homme Femme All
#>   Ouvrier specialise       13.0  11.7  12.3
#>   Ouvrier qualifie         30.9   6.9  17.7
#>   Technicien                8.9   2.2   5.2
#>   Profession intermediaire 11.9   7.9   9.7
#>   Cadre                    19.6  12.6  15.7
#>   Employe                  13.0  54.6  35.9
#>   Autre                     2.8   4.1   3.5
#>   Total                   100.0 100.0 100.0
```

4 Les graphiques

Pour produire un graphique, nous utiliserons l'extension `ggplot2` qui est chargée avec le coeur de la librairie `tidyverse`. La grammaire graphique de `ggplot2` peut être décrite de la façon suivante:

> **A statistical graphic is a `mapping` of `data` variables to `aesthetic` attributes of `geometric` objects.**

Plus spécifiquement, nous pouvons briser un graphique en trois composantes essentielles:

1. `data`: la base de données contenant les variables que nous désirons visualiser.
2. `geom`: l'objet géométrique en question. Ceci réfère au type d'objet que nous pouvons observer dans notre graphique. Par exemple, des points, des lignes, des barres, etc.
3. `aes`: les attributs esthétiques (aesthetics) de l'objet géométrique que nous affichons dans notre graphique. Par exemple, la position x/y, la couleur, la forme, la taille. Chaque attribut peut être associé à une variable dans notre base de données.

Une des particularités de `ggplot2` est qu'elle part du principe que les données relatives à un graphique sont stockées dans un tableau de données (*data frame*, *tibble* ou autre).

Dans ce qui suit on utilisera le jeu de données issu du recensement de la population de 2018 inclus dans l'extension `questionr` (résultats partiels concernant les communes de plus de 2000 habitants de France métropolitaine). On charge ces données et on en extrait les données de 5 départements (l'utilisation de la fonction `filter` sera expliquée **?@sec-filter**) :

```
library(questionr)
data(rp2018)

rp <- filter(
    rp2018,
    departement %in% c("Oise", "Rhône", "Hauts-de-Seine", "Lozère", "Bouches-du-Rhône")
)
```

4.1 Initialisation

Un graphique `ggplot2` s'initialise à l'aide de la fonction `ggplot()`. Les données représentées graphiquement sont toujours issues d'un tableau de données (*data frame* ou *tibble*), qu'on passe en argument `data` à la fonction :

```
ggplot(data = rp)
## Ou, équivalent
ggplot(rp)
```

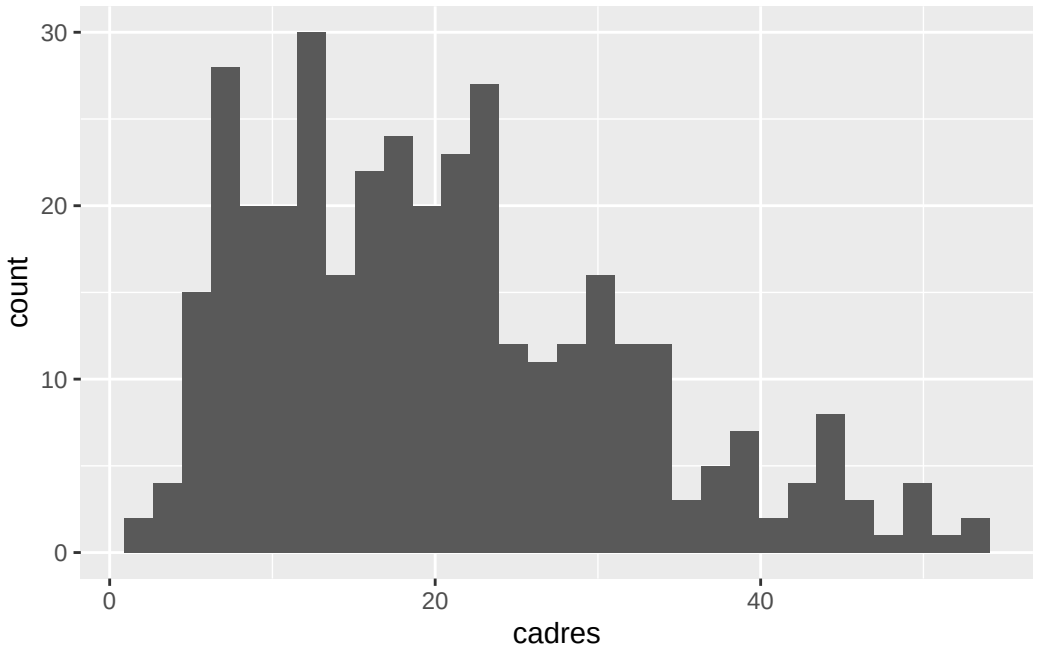
On a défini la source de données, il faut maintenant ajouter des éléments de représentation graphique. Ces éléments sont appelés des `geom`, et on les ajoute à l'objet graphique de base avec l'opérateur `+`.

Un des `geom` les plus simples est `geom_histogram`. On peut l'ajouter de la manière suivante :

```
ggplot(rp) +
    geom_histogram()
```

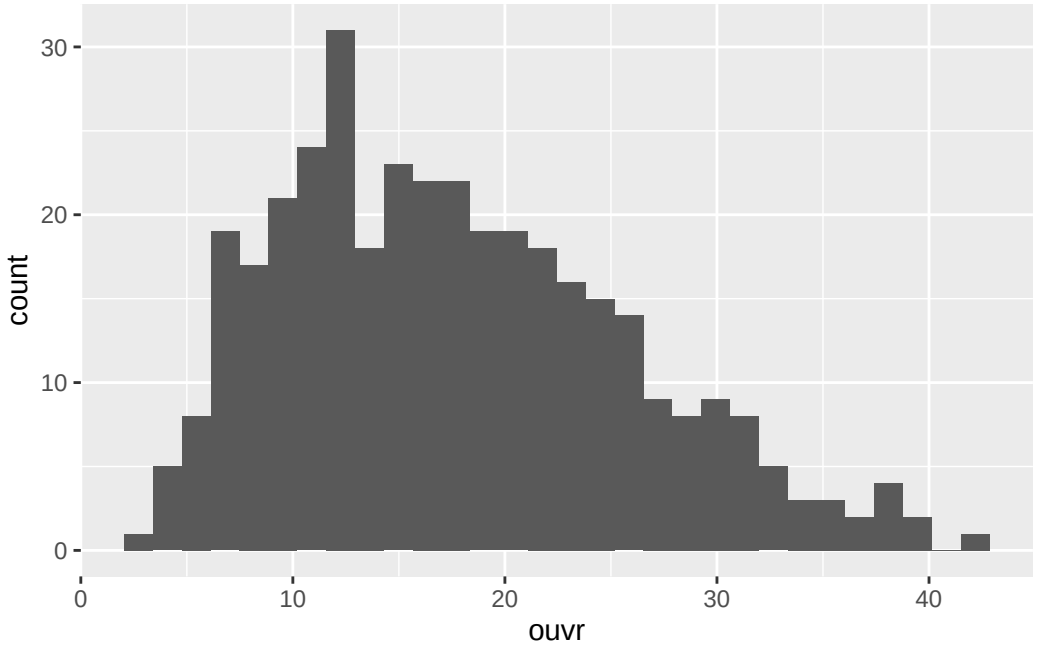
Reste à indiquer quelle donnée nous voulons représenter sous forme d'histogramme. Cela se fait à l'aide d'arguments passés via la fonction `aes()`. Ici nous avons un paramètre à renseigner, `x`, qui indique la variable à représenter sur l'axe des x (l'axe horizontal). Ainsi, si on souhaite représenter la distribution des communes du jeu de données selon le pourcentage de cadres dans leur population active (variable `cadres`), on pourra faire :

```
ggplot(rp) +
    geom_histogram(aes(x = cadres))
```



Si on veut représenter une autre variable, il suffit de changer la valeur de `x` :

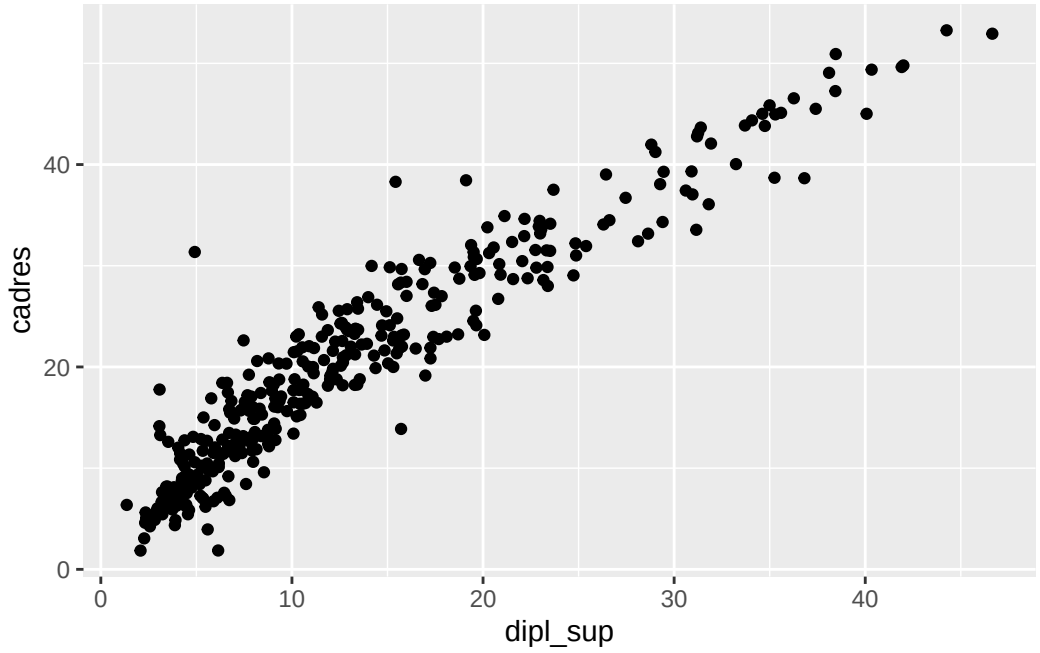
```
ggplot(rp) +
    geom_histogram(aes(x = ouvr))
```



> **Note**
>
> Quand on spécifie une variable, inutile d'indiquer le nom du tableau de données sous la forme `rp$ouvr`, car `ggplot2` recherche automatiquement la variable dans le tableau de données indiqué avec le paramètre `data`. On peut donc se contenter de `ouvr`.

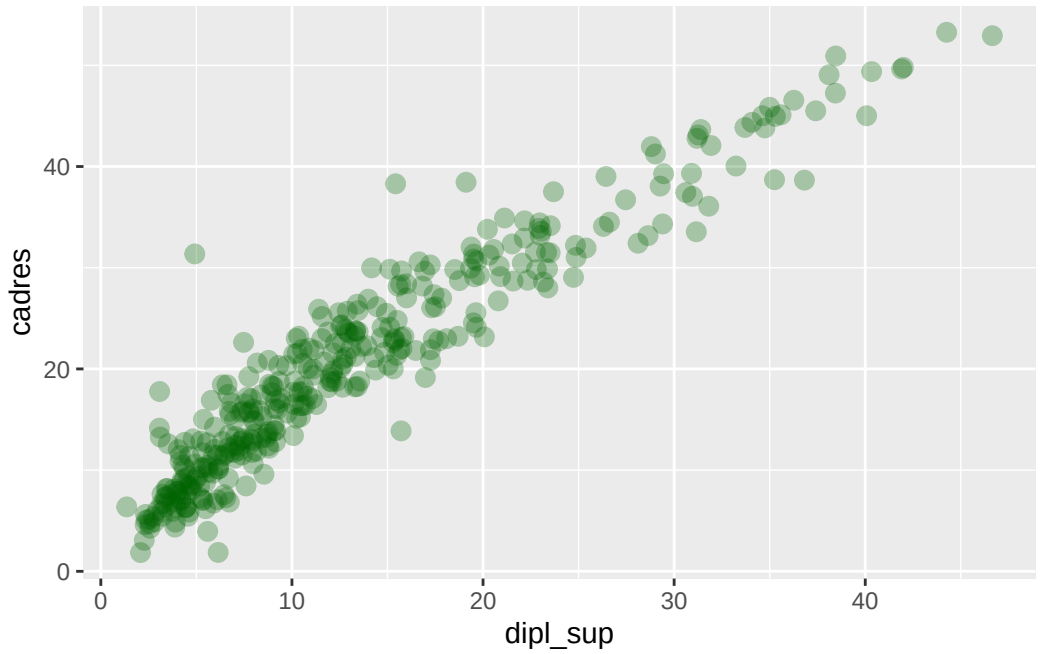
Certains `geom` prennent plusieurs paramètres. Ainsi, si on veut représenter un nuage de points, on peut le faire en ajoutant un `geom_point`. On doit alors indiquer à la fois la position en `x` (la variable sur l'axe horizontal) et en `y` (la variable sur l'axe vertical) de ces points, il faut donc passer ces deux arguments à `aes()` :

```
ggplot(rp) +
    geom_point(aes(x = dipl_sup, y = cadres))
```



On peut modifier certains attributs graphiques d'un `geom` en lui passant des arguments supplémentaires. Par exemple, pour un nuage de points, on peut modifier la couleur des points avec l'argument `color`, leur taille avec l'argument `size`, et leur transparence avec l'argument `alpha` :

```
ggplot(rp) +
    geom_point(
        aes(x = dipl_sup, y = cadres),
        color = "darkgreen", size = 3, alpha = 0.3
    )
```

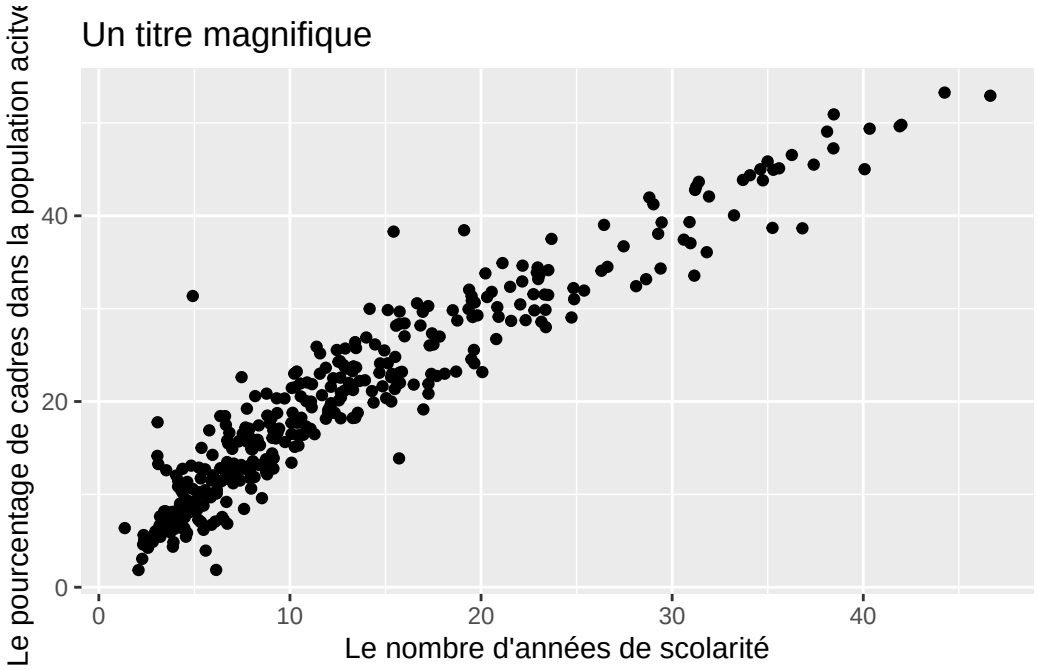


On notera que dans ce cas les arguments sont dans la fonction `geom` mais à l'extérieur du `aes()`. Plus d'explications sur ce point dans quelques instants.

4.2 Les titres

Pour ajouter un titre à votre graphique et pour ajouter des titres à vos axes `x` et `y`, nous utilisons la commande `labs()`.

```
ggplot(rp) +
    geom_point(
        aes(x = dipl_sup, y = cadres)
    ) +
    labs(
        title = "Un titre magnifique",
        x = "Le nombre d'années de scolarité",
        y = "Le pourcentage de cadres dans la population acitve"
    )
```



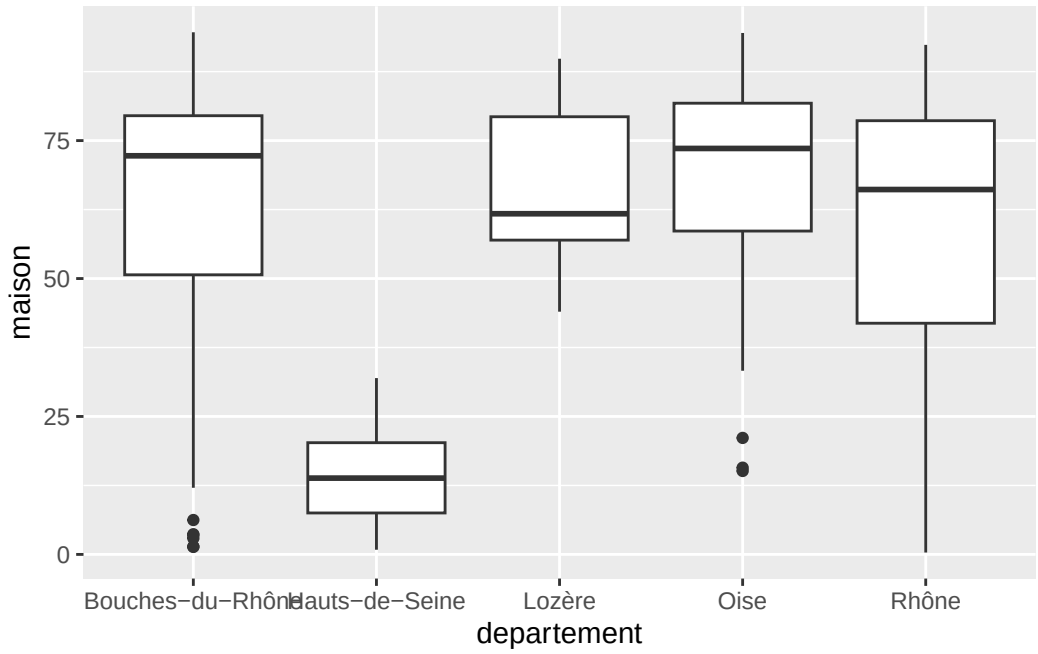
4.3 Exemples de `geom`

Il existe un grand nombre de `geom`, décrits en détail dans la documentation officielle. Outre les `geom_histogram` et `geom_point` que l'on vient de voir, on pourra noter les `geom` suivants.

4.3.1 `geom_boxplot`

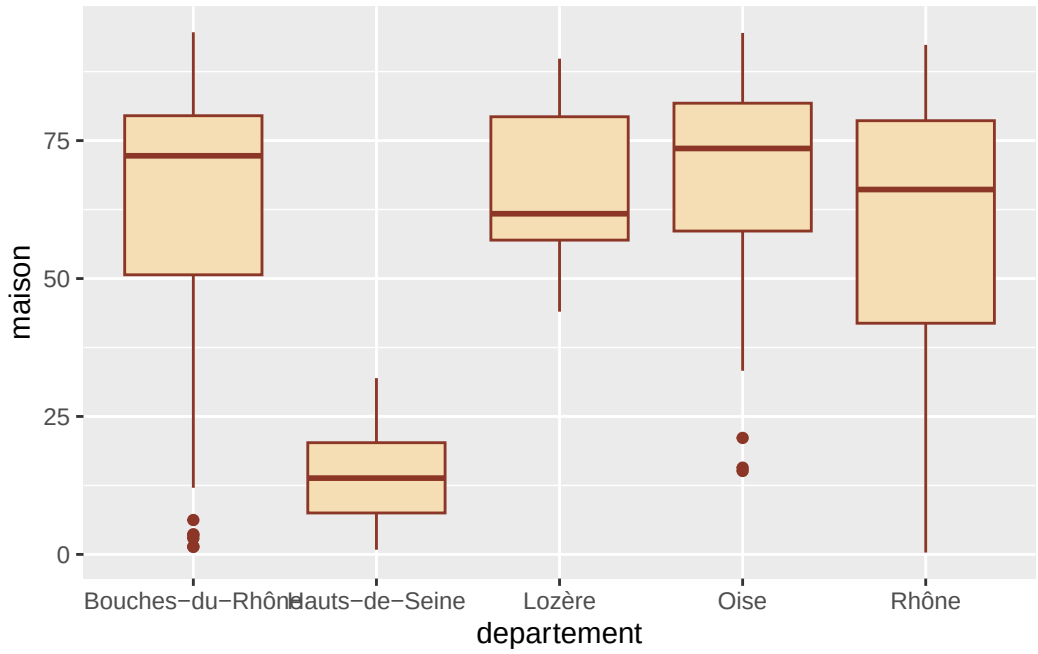
`geom_boxplot` permet de représenter des boîtes à moustaches. On lui passe en `y` la variable numérique dont on veut étudier la répartition, et en `x` la variable qualitative contenant les classes qu'on souhaite comparer. Ainsi, si on veut comparer la répartition du pourcentage de maisons en fonction du département de la commune, on pourra faire :

```
ggplot(rp) +
    geom_boxplot(aes(x = departement, y = maison))
```



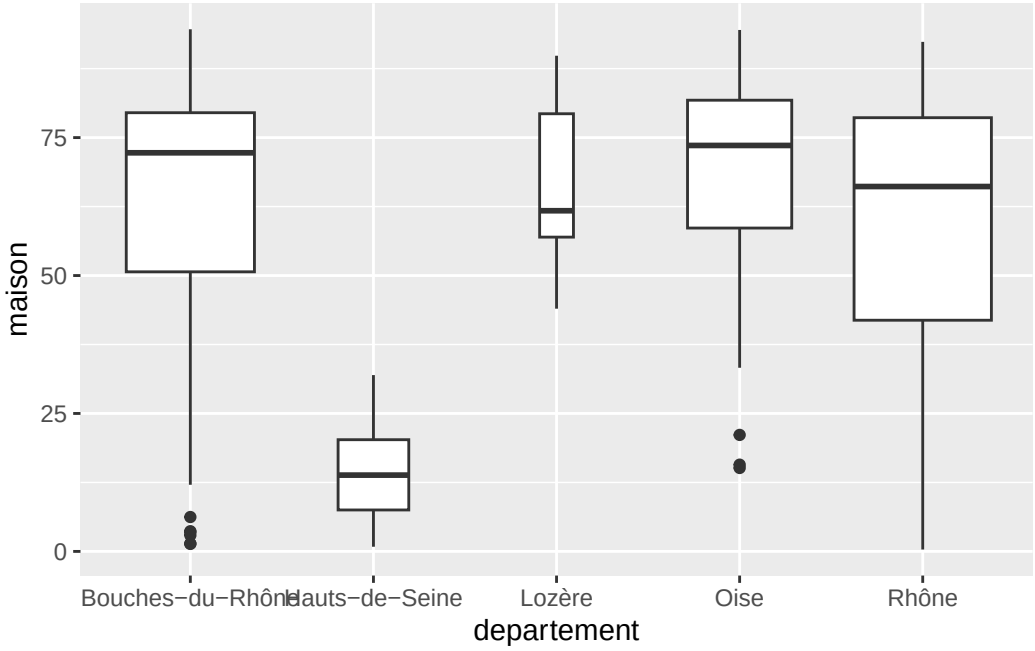
On peut personnaliser la présentation avec différents argument supplémentaires comme `fill` ou `color` :

```
ggplot(rp) +
    geom_boxplot(
        aes(x = departement, y = maison),
        fill = "wheat", color = "tomato4"
    )
```



Un autre argument utile, **varwidth**, permet de faire varier la largeur des boîtes en fonction des effectifs de la classe (donc, ici, en fonction du nombre de communes de chaque département) :

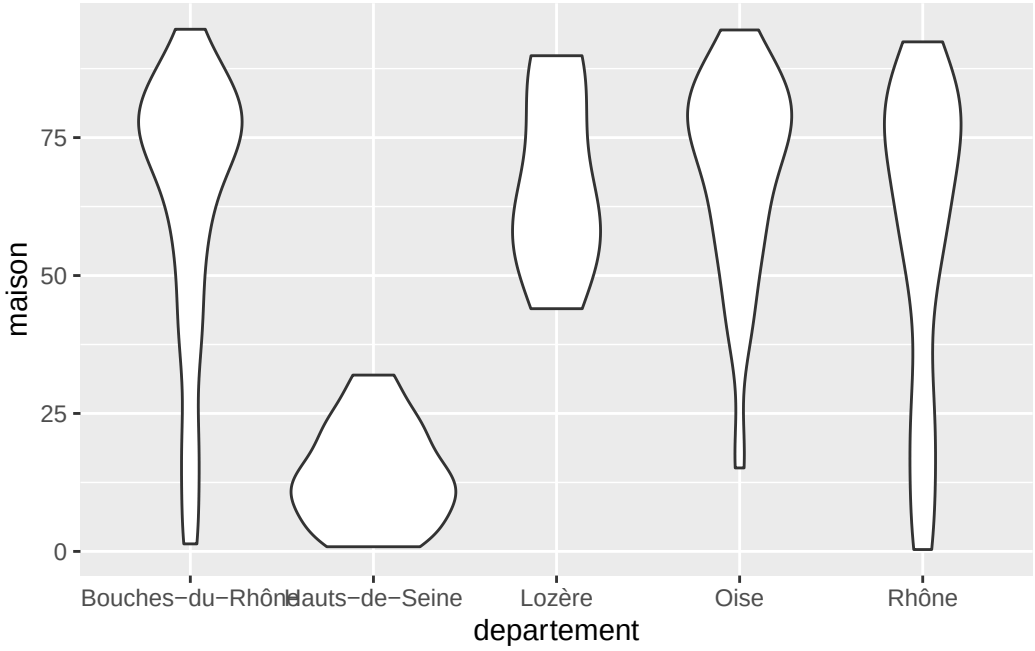
```
ggplot(rp) +
    geom_boxplot(
        aes(x = departement, y = maison),
        varwidth = TRUE)
```



4.3.2 `geom_violin`

`geom_violin` est très semblable à `geom_boxplot`, mais utilise des graphes en violon à la place des boîtes à moustache.

```
ggplot(rp) +
    geom_violin(aes(x = departement, y = maison))
```



Les graphes en violon peuvent donner une lecture plus fine des différences de distribution selon les classes. Comme pour les graphiques de densité, on peut faire varier le niveau de "détail" de la représentation en utilisant l'argument `bw` (bande passante).

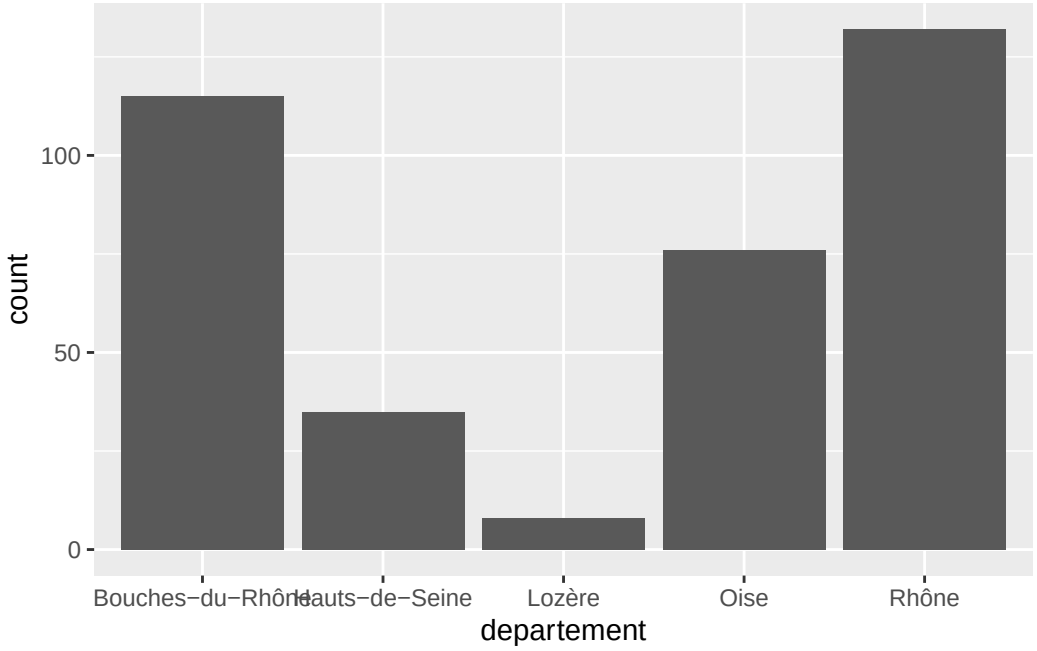
```
ggplot(rp) +
    geom_violin(
        aes(x = departement, y = maison),
        bw = 2
    )
```

4.3.3 `geom_bar` et `geom_col`

`geom_bar` permet de produire un graphique en bâtons (*barplot*). On lui passe en `x` la variable qualitative dont on souhaite représenter l'effectif de chaque modalité.

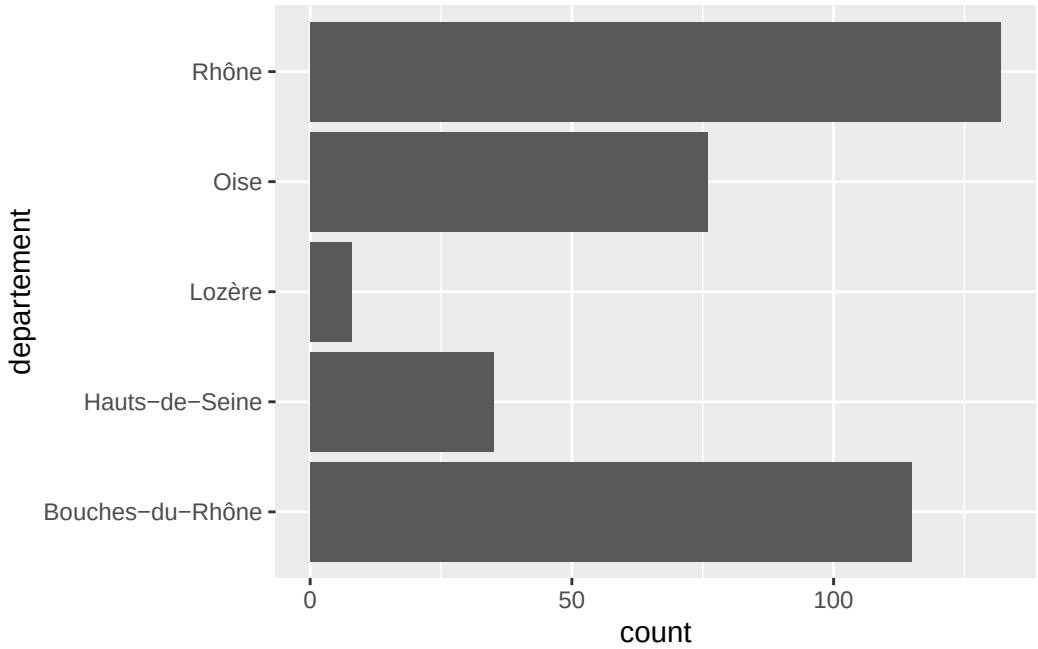
Par exemple, si on veut afficher le nombre de communes de notre jeu de données pour chaque département :

```
ggplot(rp) +
    geom_bar(aes(x = departement))
```



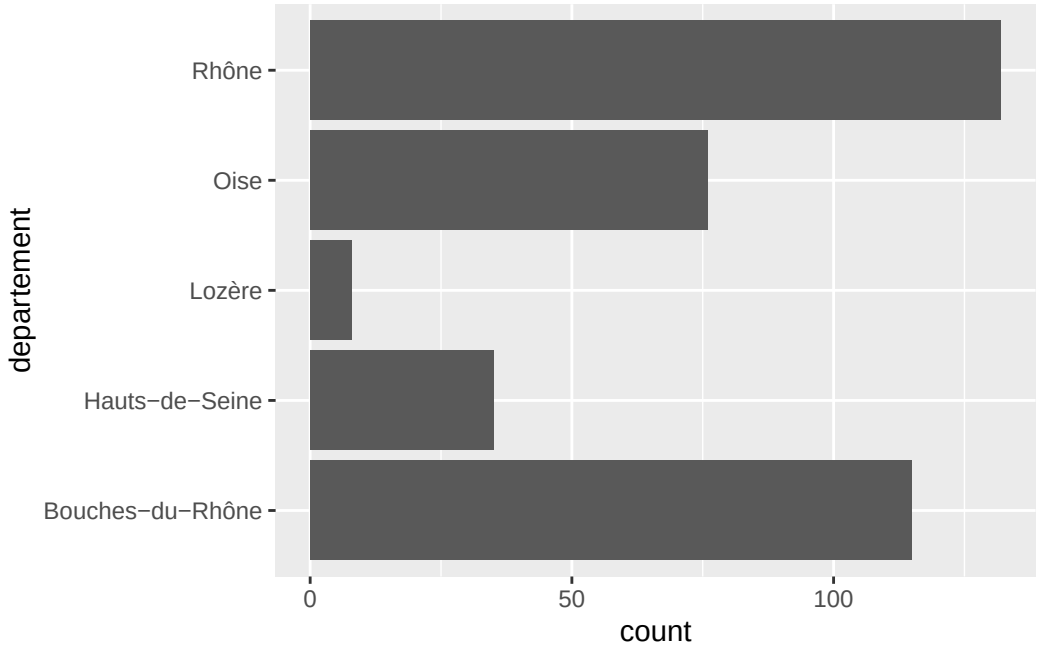
Si on préfère avoir un graphique en barres horizontales, il suffit de passer la variable comme attribut `y` plutôt que `x`.

```
ggplot(rp) +
    geom_bar(aes(y = departement))
```



Une autre possibilité est d'utiliser `coord_flip()`, qui permet d'intervertir l'axe horizontal et l'axe vertical.

```
ggplot(rp) +
    geom_bar(aes(x = departement)) +
    coord_flip()
```

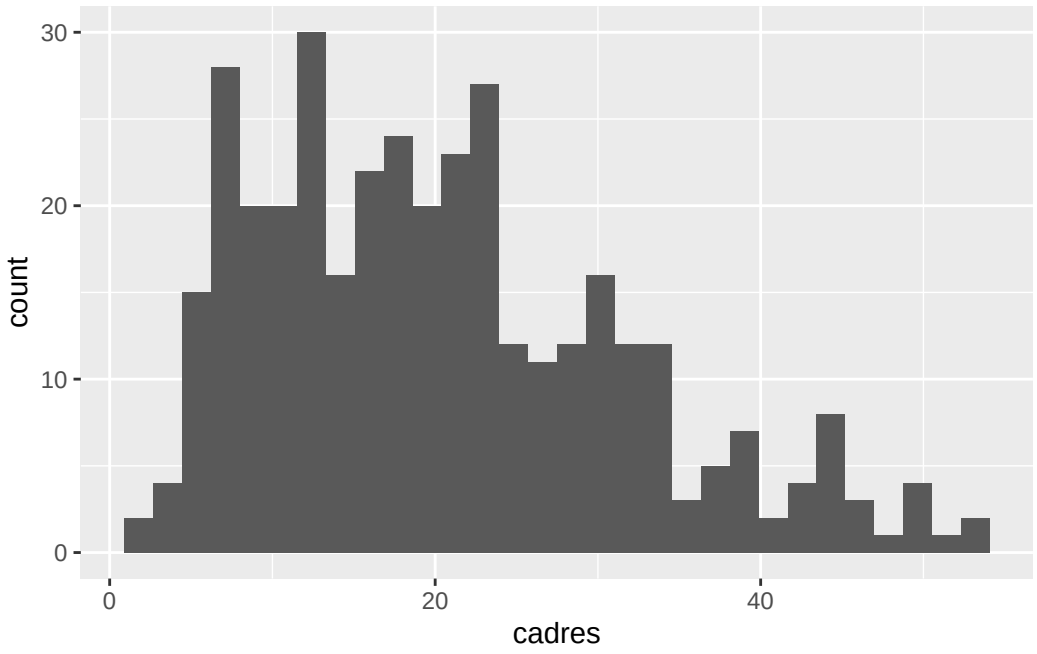


À noter que `coord_flip()` peut s'appliquer à n'importe quel graphique `ggplot2`.

4.3.4 `geom_histogram`

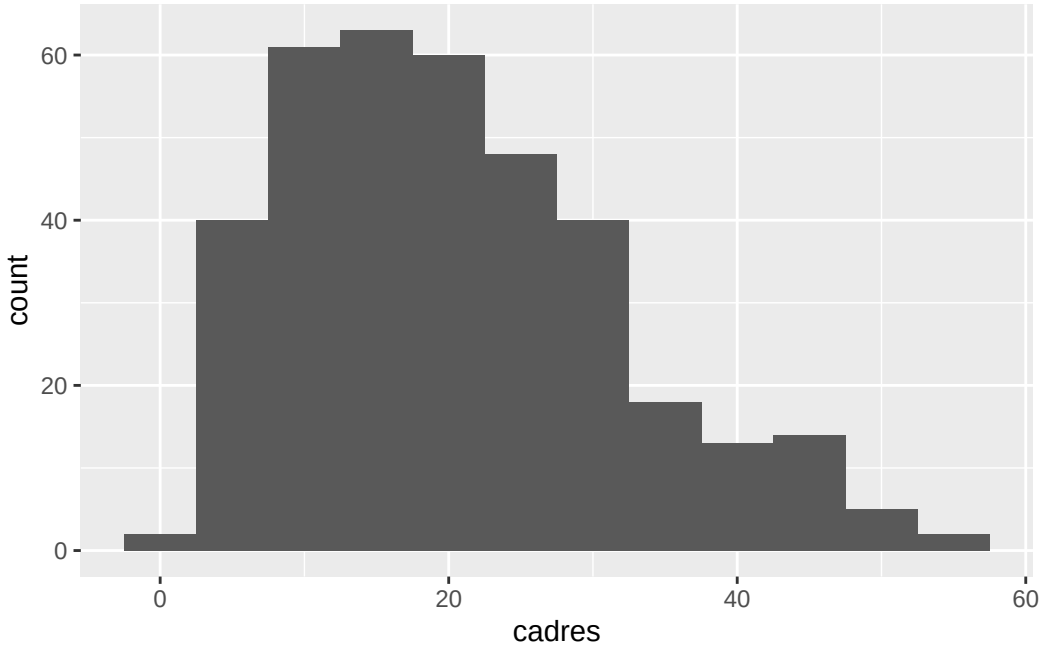
`geom_histogram` permet de représenter des histogrammes. On lui passe en `x` la variable quantitative dont on souhiate étudier la répartition.

```
ggplot(rp) +
    geom_histogram(aes(x = cadres))
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



On peut utiliser différents arguments, comme par exemple `binwidth` pour spécifier la largeur des rectangles de notre histogramme.

```
ggplot(rp) +
    geom_histogram(aes(x = cadres), binwidth = 5)
```

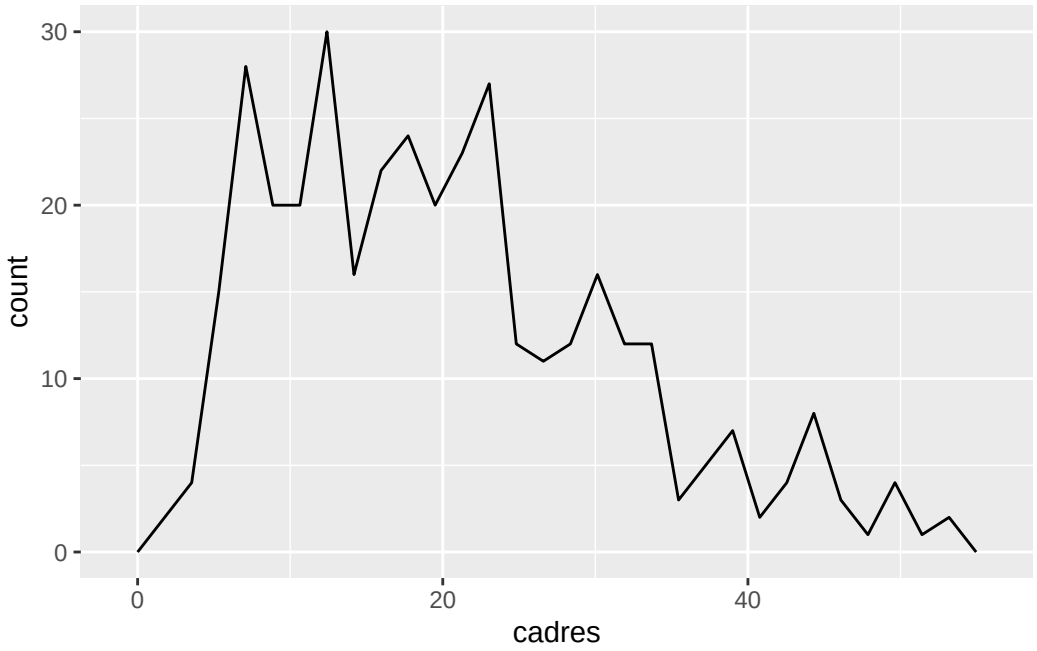


4.3.5 `geom_freqpoly`

`geom_freqpoly` permet d'afficher le polygone de fréquences d'une variable numérique. Son usage est similaire à celui de `geom_histogram`.

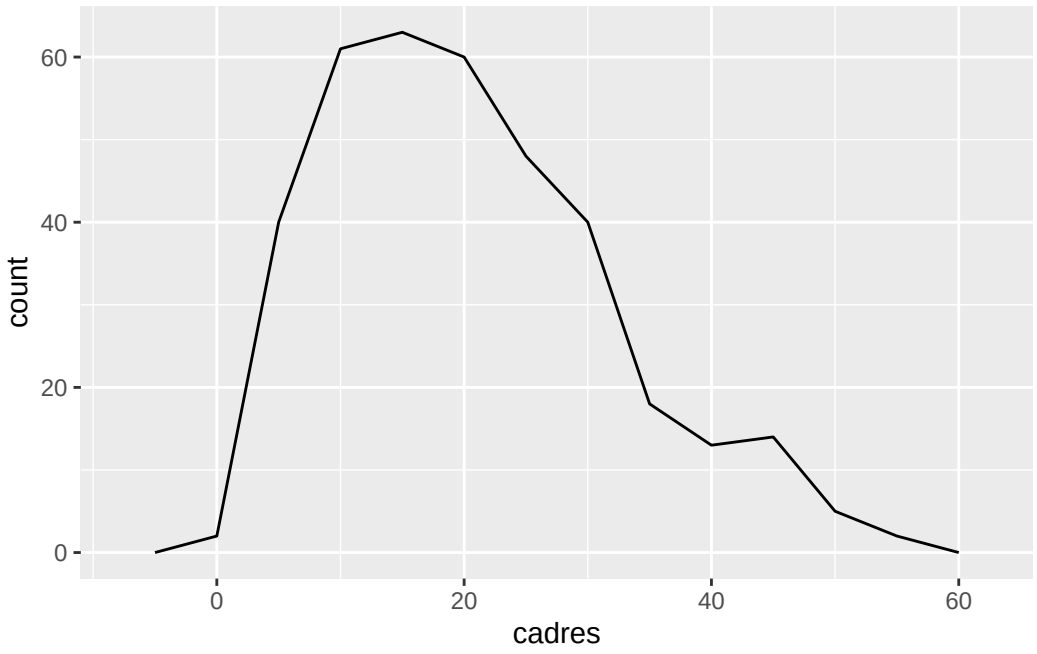
Ainsi, si on veut afficher le polygone de fréquences de la part des cadres dans les communes de notre jeu de données :

```
ggplot(rp) +
    geom_freqpoly(aes(x = cadres))
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



On peut utiliser différents arguments pour ajuster le calcul de l'estimation de densité, parmi lesquels `kernel` et `bw` (voir la page d'aide de la fonction `density` pour plus de détails). `bw` (abbréviation de *bandwidth*, bande passante) permet de régler la "finesse" de l'estimation de densité, un peu comme le choix du nombre de classes dans un histogramme :

```
ggplot(rp) +
    geom_freqpoly(aes(x = cadres), binwidth = 5)
```



4.3.6 `geom_line`

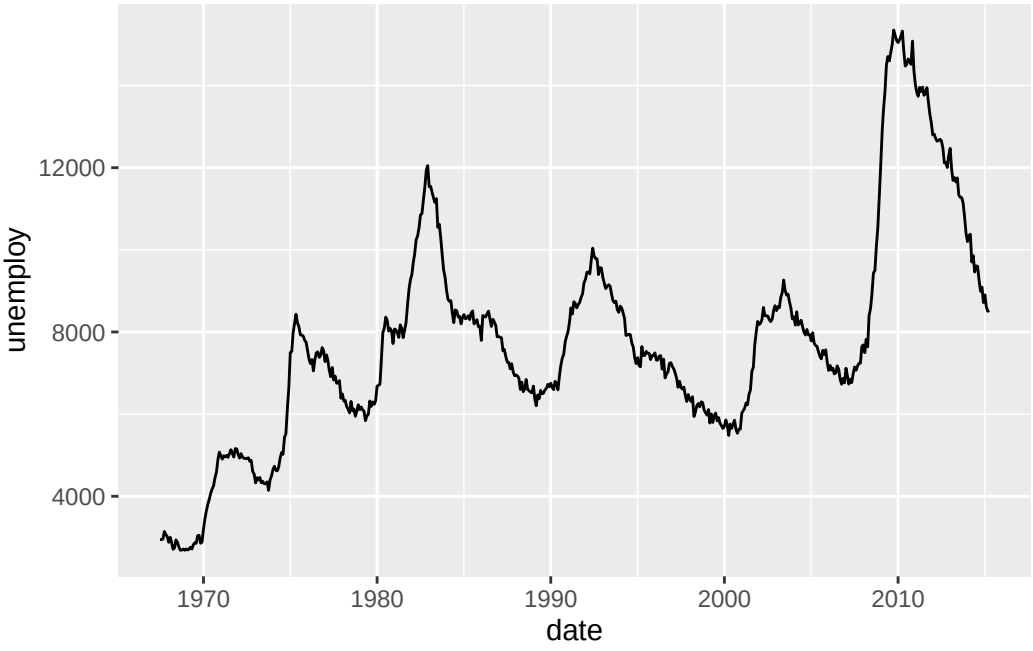
`geom_line` trace des lignes connectant les différentes observations entre elles. Il est notamment utilisé pour la représentation de séries temporelles. On passe à `geom_line` deux paramètres : `x` et `y`. Les observations sont alors connectées selon l'ordre des valeurs passées en `x`.

Comme il n'y a pas de données adaptées pour ce type de représentation dans notre jeu de données d'exemple, on va utiliser ici le jeu de données `economics` inclus dans `ggplot2` et représenter l'évolution du taux de chômage aux États-Unis (variable `unemploy`) dans le temps (variable `date`) :

```
data("economics")
economics
#> # A tibble: 574 x 6
#>    date         pce    pop psavert uempmed unemploy
#>    <date>     <dbl>  <dbl>   <dbl>   <dbl>    <dbl>
#>  1 1967-07-01  507. 198712    12.6     4.5     2944
#>  2 1967-08-01  510. 198911    12.6     4.7     2945
#>  3 1967-09-01  516. 199113    11.9     4.6     2958
#>  4 1967-10-01  512. 199311    12.9     4.9     3143
#>  5 1967-11-01  517. 199498    12.8     4.7     3066
#>  6 1967-12-01  525. 199657    11.8     4.8     3018
#>  7 1968-01-01  531. 199808    11.7     5.1     2878
```

```
#>  8 1968-02-01  534. 199920    12.3     4.5     3001
#>  9 1968-03-01  544. 200056    11.7     4.1     2877
#> 10 1968-04-01  544  200208    12.3     4.6     2709
#> # i 564 more rows
```

```
ggplot(economics) +
    geom_line(aes(x = date, y = unemploy))
```



4.4 Mappages

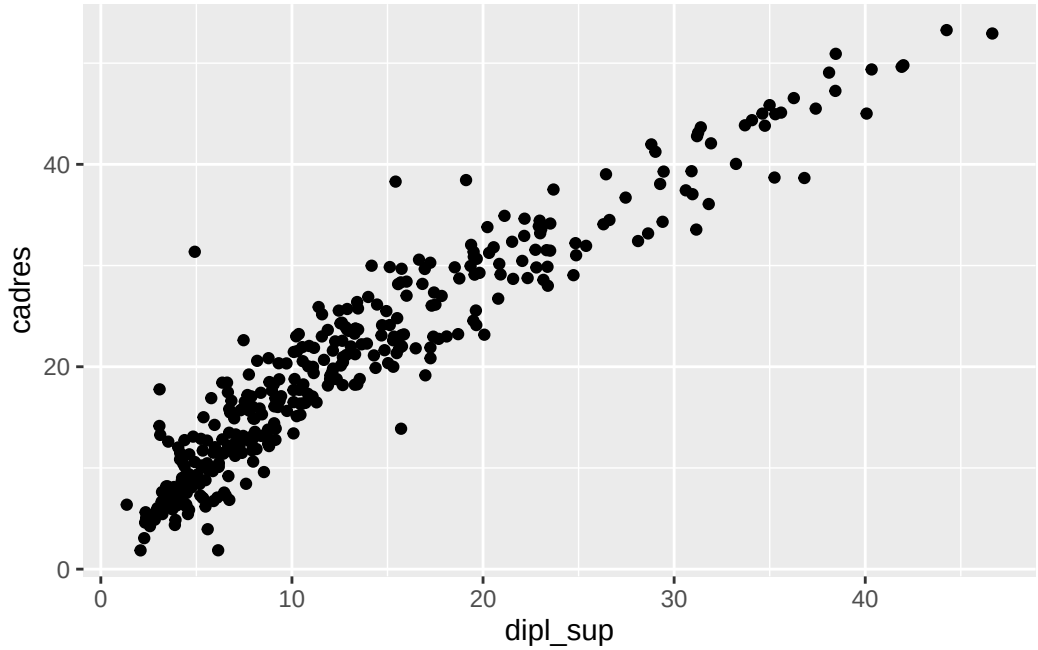
Un *mappage*, dans `ggplot2`, est une mise en relation entre un **attribut graphique** du `geom` (position, couleur, taille…) et une **variable** du tableau de données.

Ces mappages sont passés aux différents `geom` via la fonction `aes()` (abbréviation d'*aesthetic*).

4.4.1 Exemples de mappages

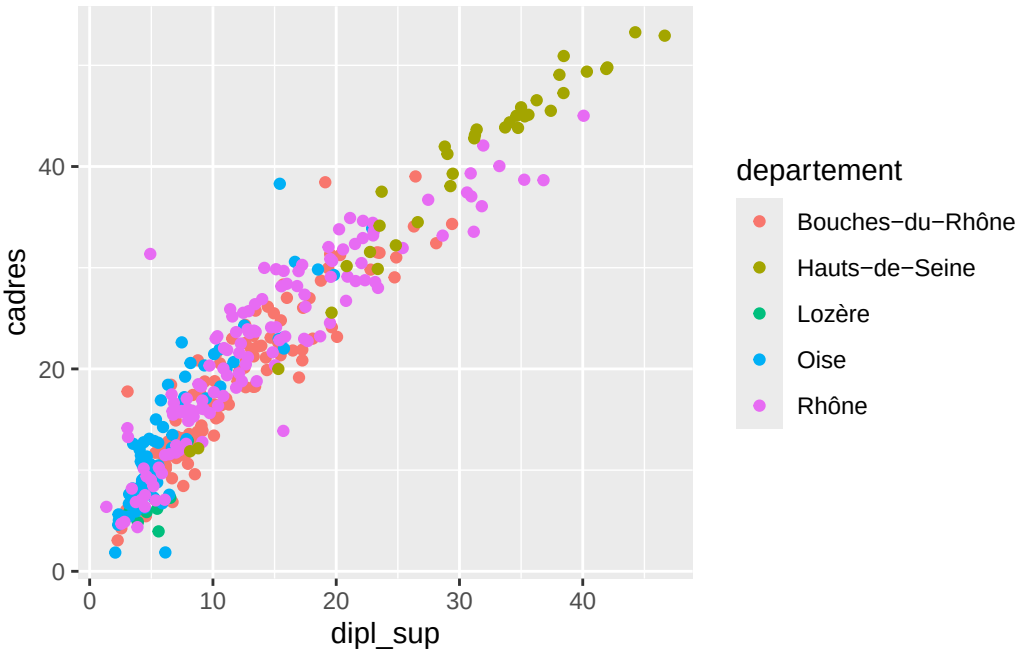
On a déjà vu les mappages `x` et `y` pour un nuage de points. Ceux-ci signifient que la position d'un point donné horizontalement (`x`) et verticalement (`y`) dépend de la valeur des variables passées comme arguments `x` et `y` dans `aes()`.

```
ggplot(rp) +
    geom_point(
        aes(x = dipl_sup, y = cadres)
    )
```



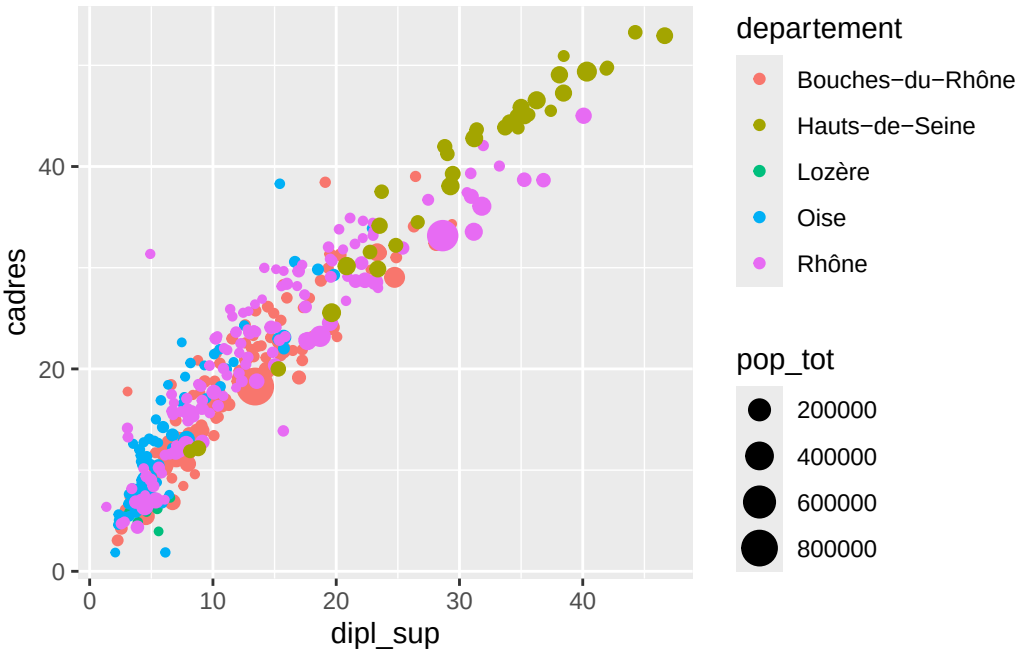
Mais on peut ajouter d'autres mappages. Par exemple, `color` permet de faire varier la couleur des points automatiquement en fonction des valeurs d'une troisième variable. Ainsi, on peut vouloir colorer les points selon le département de la commune correspondante.

```
ggplot(rp) +
    geom_point(
        aes(x = dipl_sup, y = cadres, color = departement)
    )
```



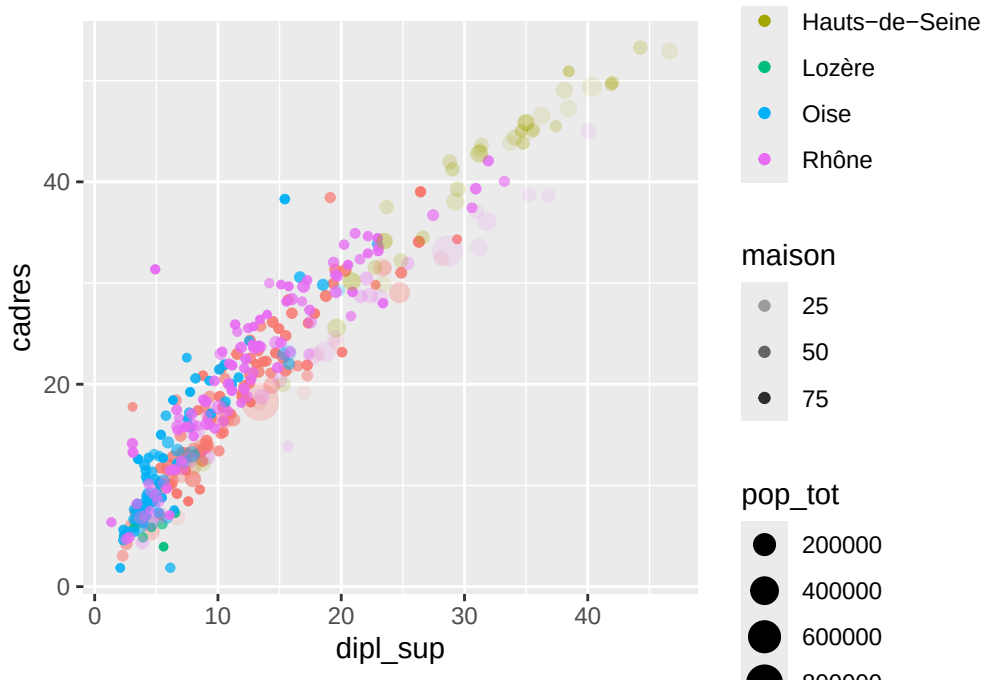
On peut aussi faire varier la taille des points avec `size`. Ici, la taille dépend de la population totale de la commune :

```
ggplot(rp) +
    geom_point(
        aes(x = dipl_sup, y = cadres, color = departement, size = pop_tot)
    )
```



On peut même associer la transparence des points à une variable avec `alpha` :

```
ggplot(rp) +
    geom_point(
        aes(x = dipl_sup, y = cadres, color = departement, size = pop_tot, alpha = maison)
    )
```

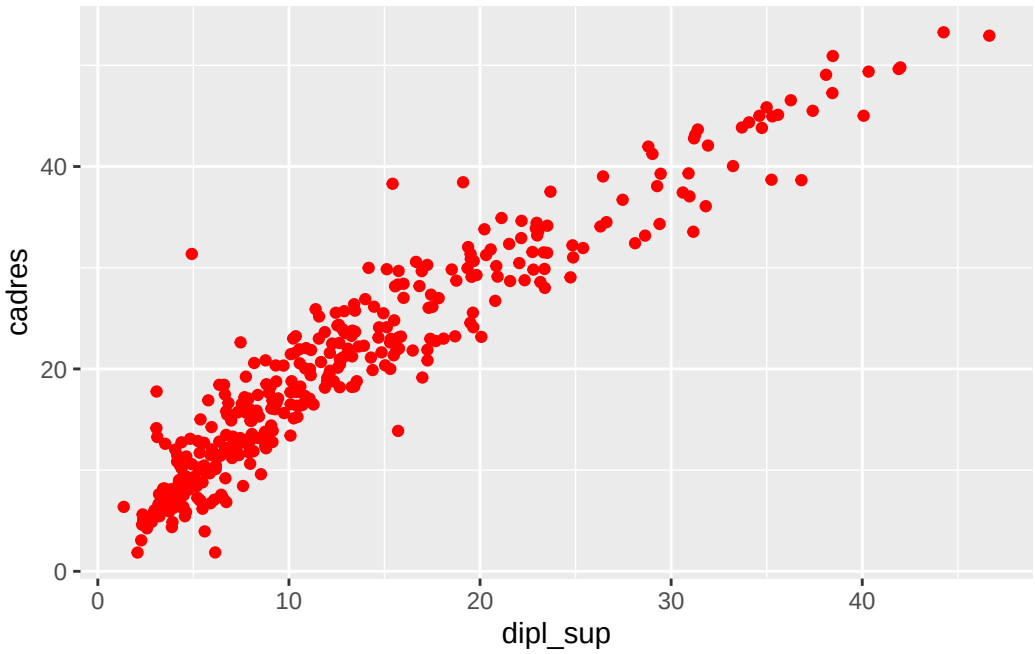


Chaque `geom` possède sa propre liste de mappages.

4.4.2 `aes()` or not `aes()` ?

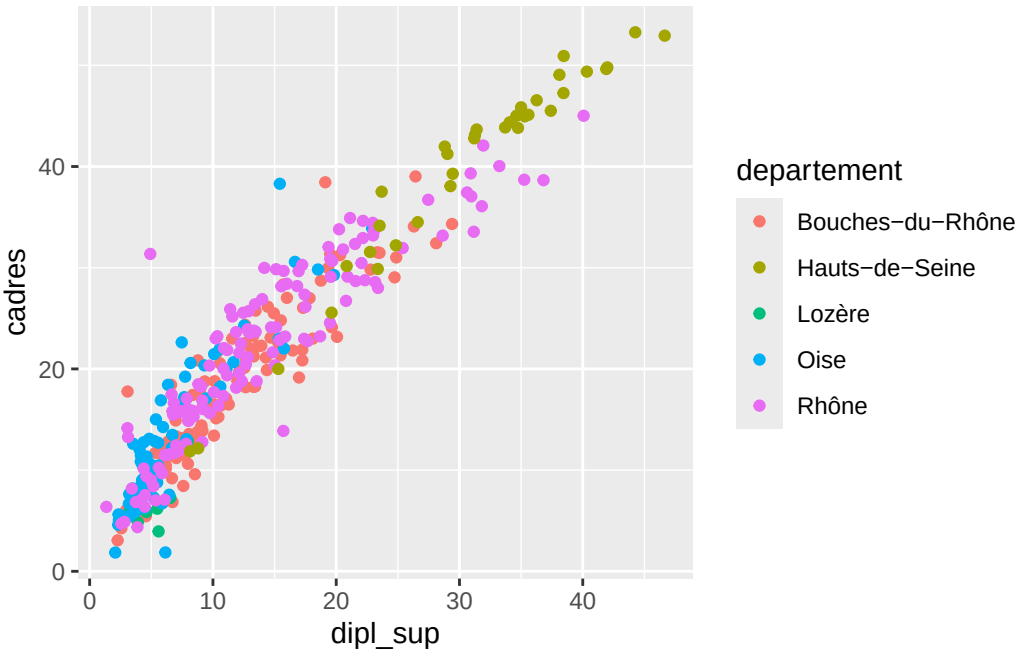
Comme on l'a déjà vu, parfois on souhaite changer un attribut sans le relier à une variable : c'est le cas par exemple si on veut représenter tous les points en rouge. Dans ce cas on utilise toujours l'attribut `color`, mais comme il ne s'agit pas d'un mappage, on le définit **à l'extérieur** de la fonction `aes()`.

```
ggplot(rp) +
    geom_point(
        aes(x = dipl_sup, y = cadres),
        color = "red"
    )
```



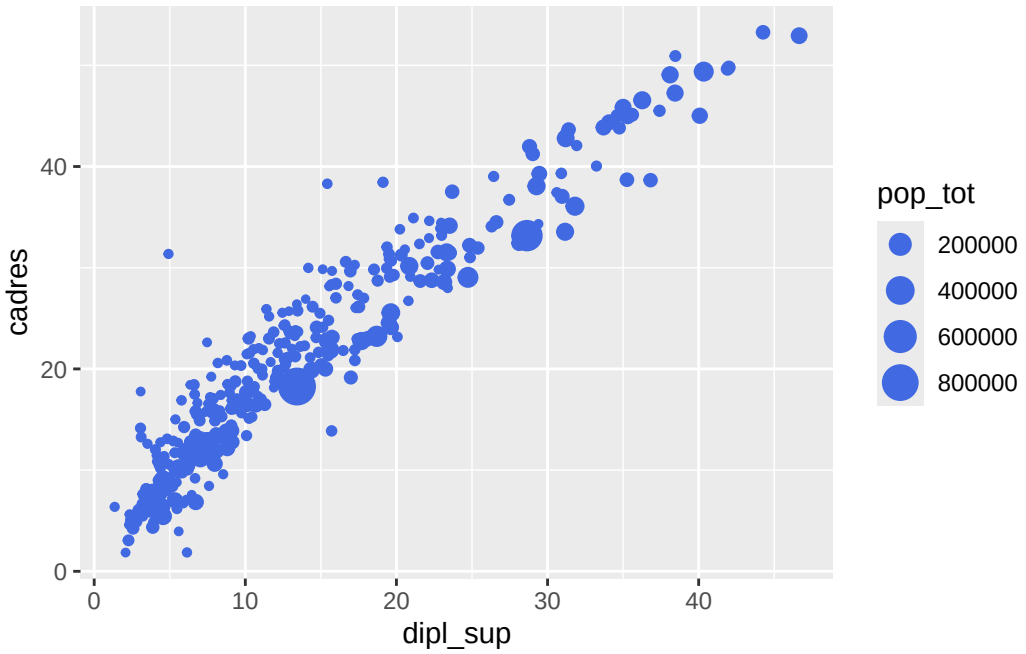
Par contre, si on veut faire varier la couleur en fonction des valeurs prises par une variable, on réalise un mappage, et on doit donc placer l'attribut `color` **à l'intérieur** de `aes()`.

```
ggplot(rp) +
    geom_point(
        aes(x = dipl_sup, y = cadres, color = departement)
    )
```



On peut mélanger attributs liés à une variable (mappage, donc dans `aes()`) et attributs constants (donc à l'extérieur). Dans l'exemple suivant, la taille varie en fonction de la variable `pop_tot`, mais la couleur est constante pour tous les points.

```
ggplot(rp) +
    geom_point(
        aes(x = dipl_sup, y = cadres, size = pop_tot),
        color = "royalblue"
    )
```



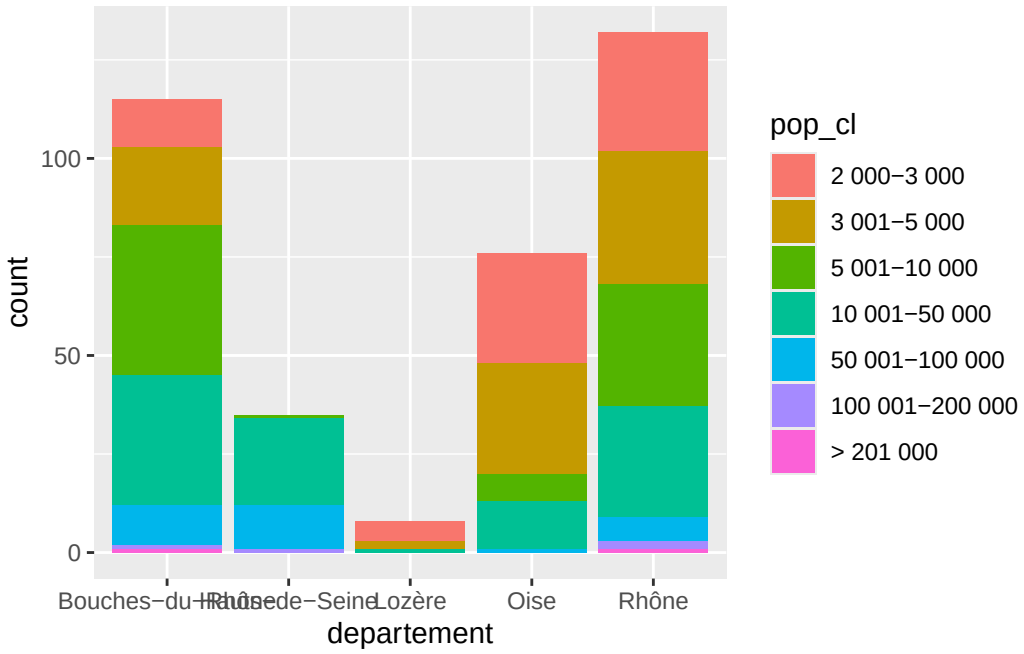
Avertissement

La règle est donc simple mais très importante :
Si on établit un lien entre les valeurs d'une variable et un attribut graphique, on définit un mappage, et on le déclare dans `aes()`. Sinon, on modifie l'attribut de la même manière pour tous les points, et on le définit en-dehors de la fonction `aes()`.

4.4.3 `geom_bar` et `position`

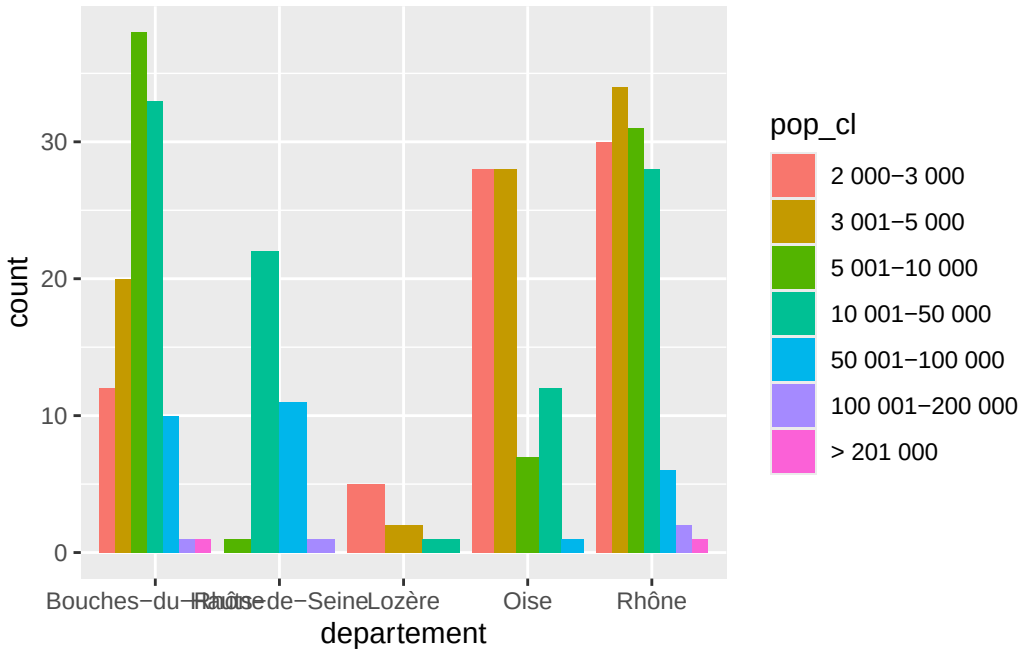
Un des mappages possibles de `geom_bar` est l'attribut `fill`, qui permet de tracer des barres de couleur différentes selon les modalités d'une deuxième variable :

```
ggplot(rp) +
    geom_bar(aes(x = departement, fill = pop_cl))
```



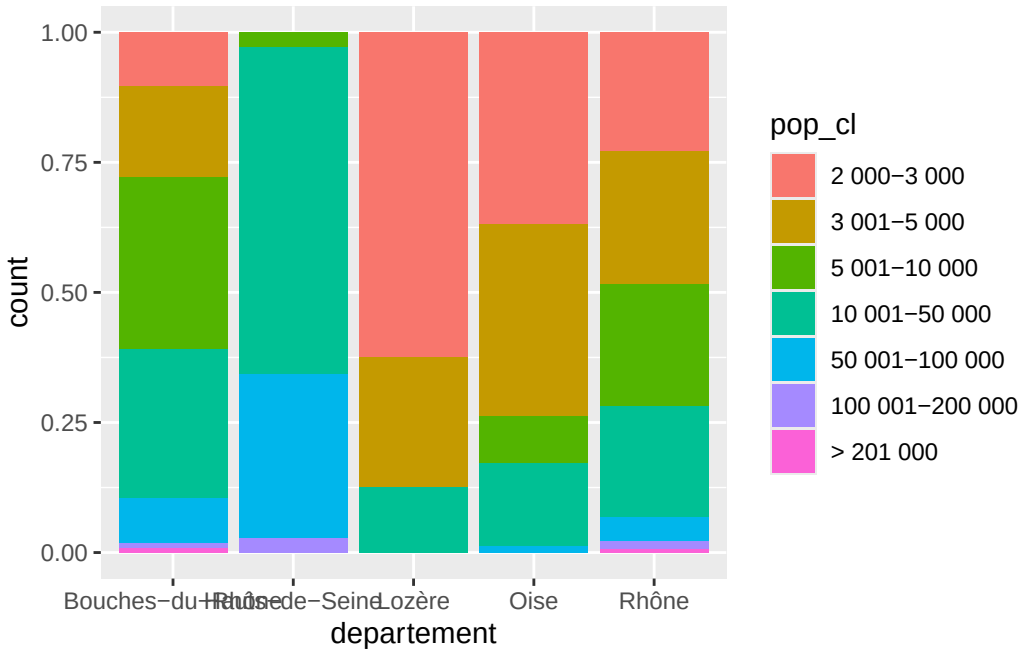
L'attribut `position` de `geom_bar` permet d'indiquer comment les différentes barres doivent être positionnées. Par défaut l'argument vaut `position = "stack"` et elles sont donc "empilées". Mais on peut préciser `position = "dodge"` pour les mettre côte à côte.

```
ggplot(rp) +
    geom_bar(
        aes(x = departement, fill = pop_cl),
        position = "dodge"
    )
```



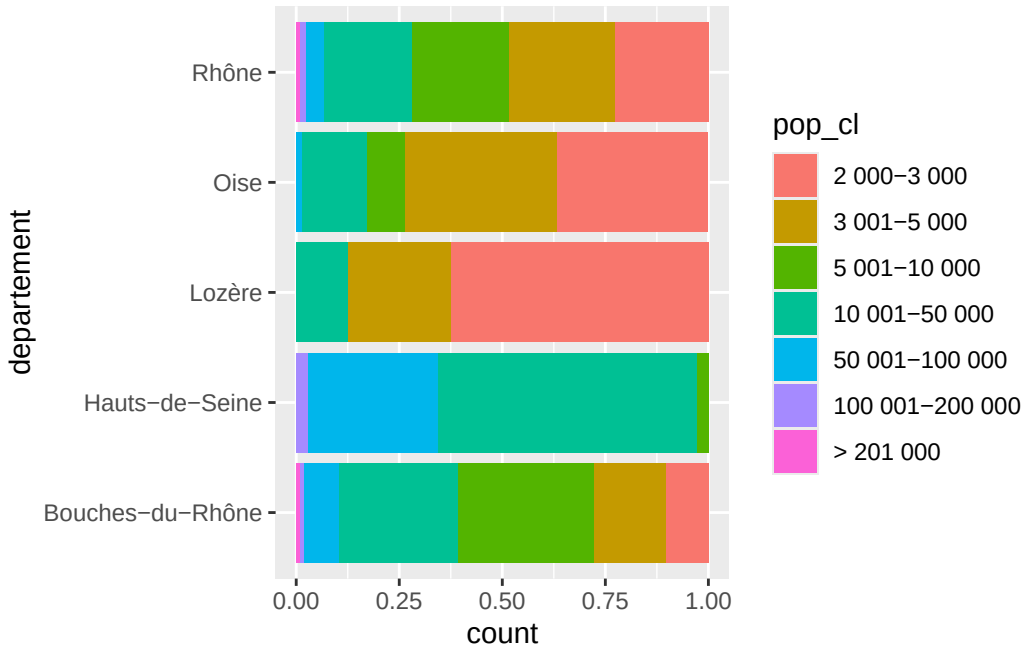
Ou encore `position = "fill"` pour représenter non plus des effectifs, mais des proportions.

```
ggplot(rp) +
    geom_bar(
        aes(x = departement, fill = pop_cl),
        position = "fill"
    )
```



Là encore, on peut utiliser `coord_flip()` si on souhaite une visualisation avec des barres horizontales.

```
ggplot(rp) +
    geom_bar(
        aes(x = departement, fill = pop_cl),
        position = "fill"
    ) +
    coord_flip()
```

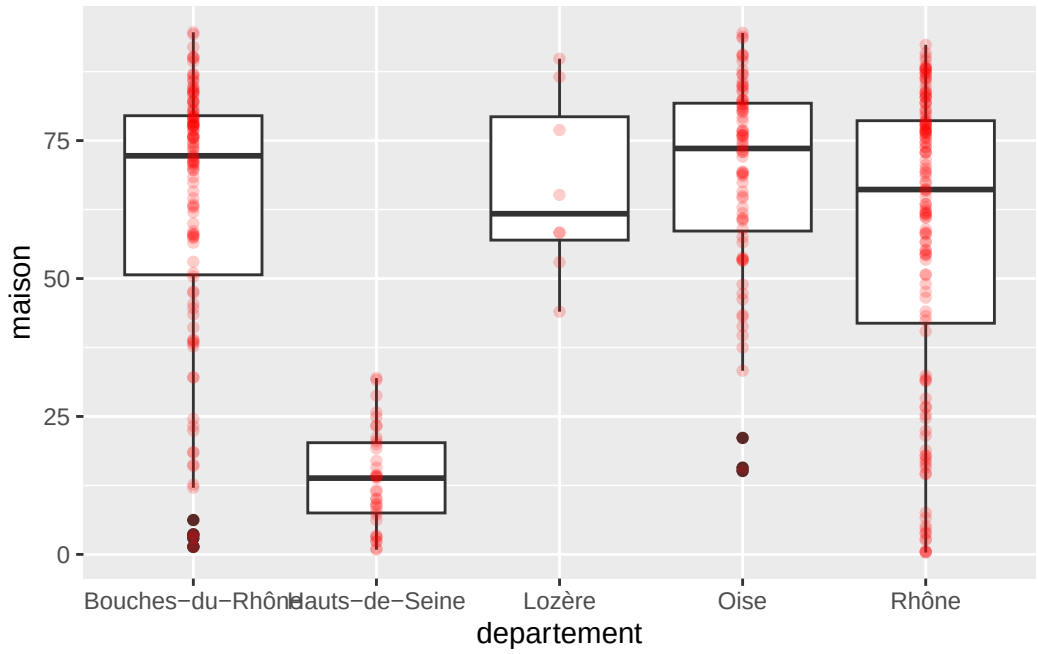


4.5 Représentation de plusieurs `geom`

On peut représenter plusieurs `geom` simultanément sur un même graphique, il suffit de les ajouter à tour de rôle avec l'opérateur `+`.

Par exemple, on peut superposer la position des points au-dessus d'un boxplot. On va pour cela ajouter un `geom_point` après avoir ajouté notre `geom_boxplot`.

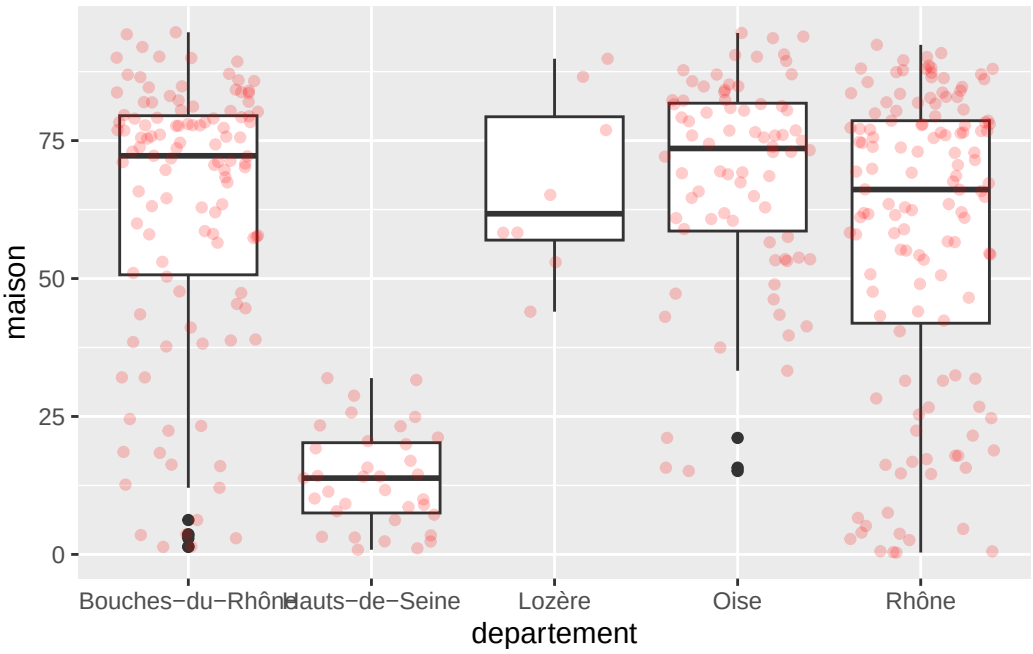
```
ggplot(rp) +
    geom_boxplot(aes(x = departement, y = maison)) +
    geom_point(
        aes(x = departement, y = maison),
        col = "red", alpha = 0.2
    )
```



> **Note**
>
> Quand une commande `ggplot2` devient longue, il peut être plus lisible de la répartir sur plusieurs lignes. Dans ce cas, il faut penser à placer l'opérateur `+` en fin de ligne, afin que R comprenne que la commande n'est pas complète et qu'il prenne en compte la suite.

Pour un résultat un peu plus lisible, on peut remplacer `geom_point` par `geom_jitter`, qui disperse les points horizontalement et facilite leur visualisation.

```
ggplot(rp) +
    geom_boxplot(aes(x = departement, y = maison)) +
    geom_jitter(
        aes(x = departement, y = maison),
        col = "red", alpha = 0.2
    )
```

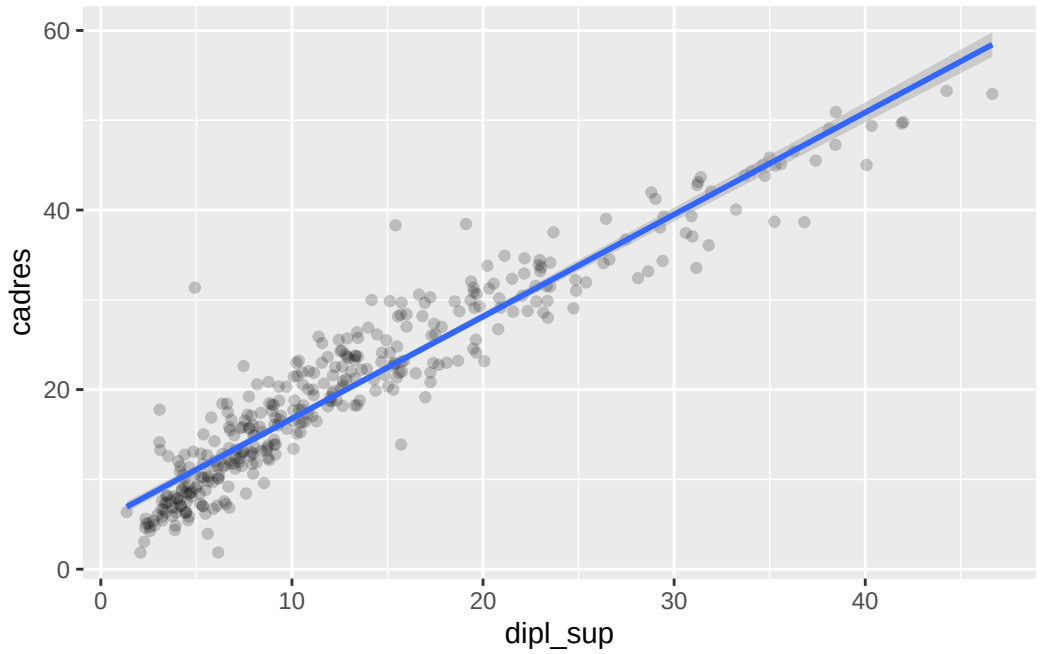


Pour simplifier un peu le code, plutôt que de déclarer les mappages dans chaque `geom`, on peut les déclarer dans l'appel à `ggplot()`. Ils seront automatiquement "hérités" par les `geom` ajoutés (sauf s'ils redéfinissent les mêmes mappages).

```
ggplot(rp, aes(x = departement, y = maison)) +
    geom_boxplot() +
    geom_jitter(color = "red", alpha = 0.2)
```

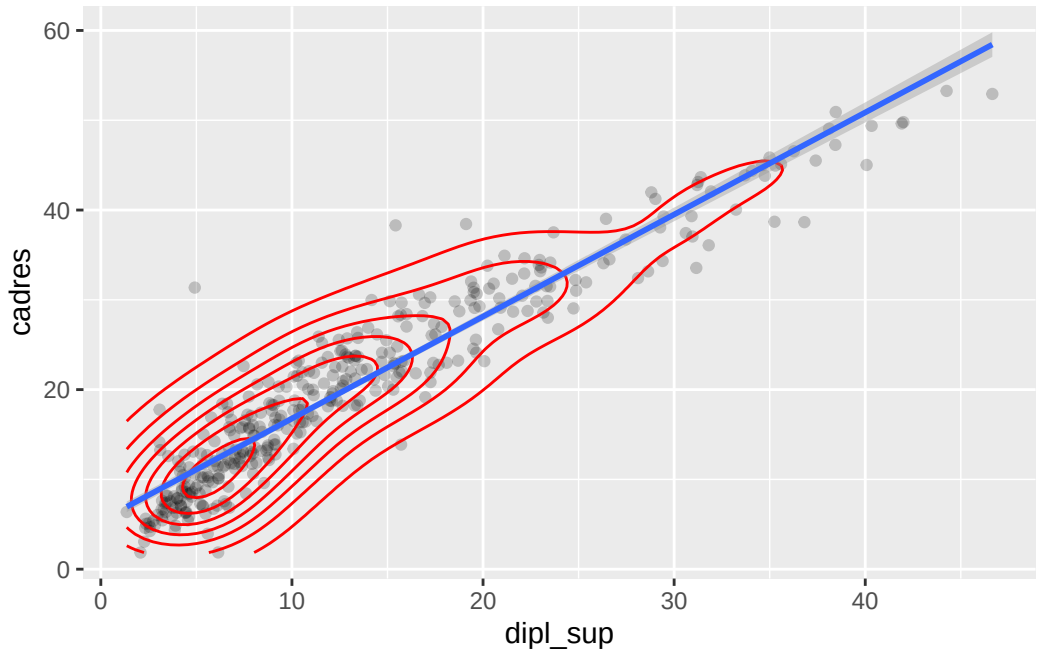
Autre exemple, on peut vouloir ajouter à un nuage de points une ligne de régression linéaire à l'aide de `geom_smooth` :

```
ggplot(rp, aes(x = dipl_sup, y = cadres)) +
    geom_point(alpha = 0.2) +
    geom_smooth(method = "lm")
#> `geom_smooth()` using formula = 'y ~ x'
```



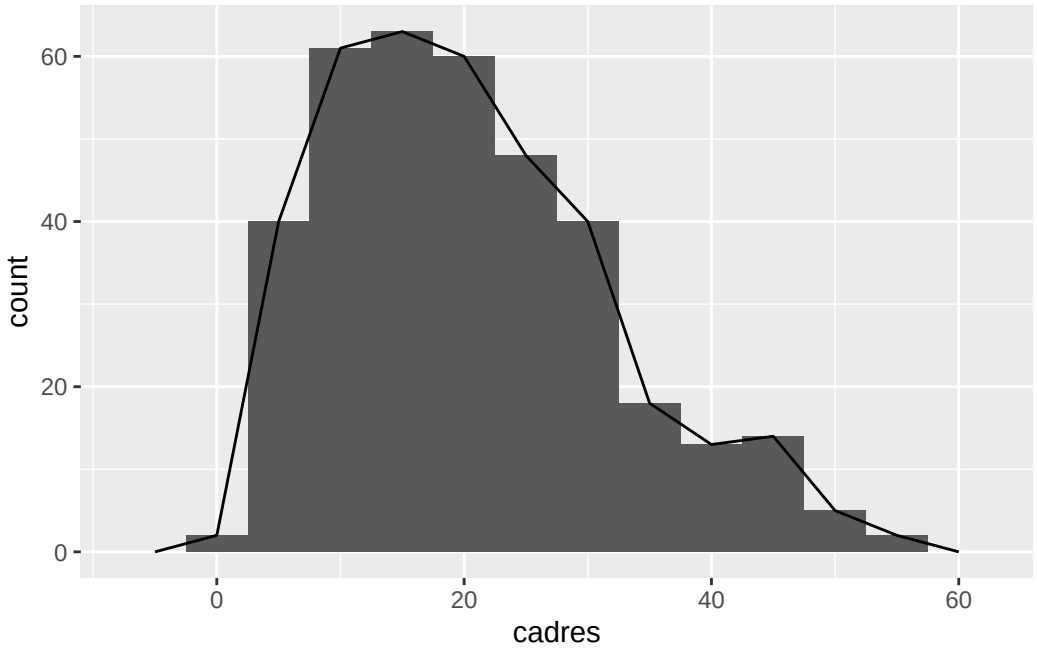
Et on peut même superposer une troisième visualisation de la répartition des points dans l'espace avec `geom_density2d` :

```
ggplot(rp, aes(x = dipl_sup, y = cadres)) +
    geom_point(alpha = 0.2) +
    geom_density2d(color = "red") +
    geom_smooth(method = "lm")
#> `geom_smooth()` using formula = 'y ~ x'
```



On peut enfin superposer l'histogramme ainsi que le polygone de fréquences.

```
ggplot(rp) +
    geom_histogram(aes(x = cadres), binwidth = 5) +
    geom_freqpoly(aes(x = cadres), binwidth = 5)
```

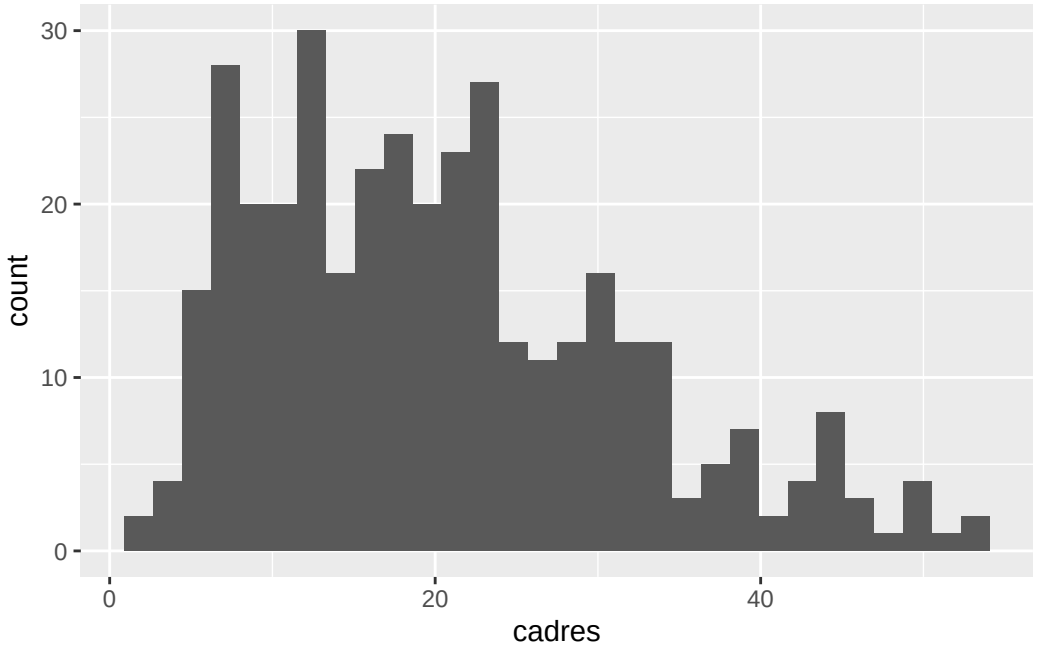


4.6 Faceting

Le *faceting* permet d'effectuer plusieurs fois le même graphique selon les valeurs d'une ou plusieurs variables qualitatives.

Par exemple, on a vu qu'on peut représenter l'histogramme du pourcentage de cadres dans nos communes avec le code suivant :

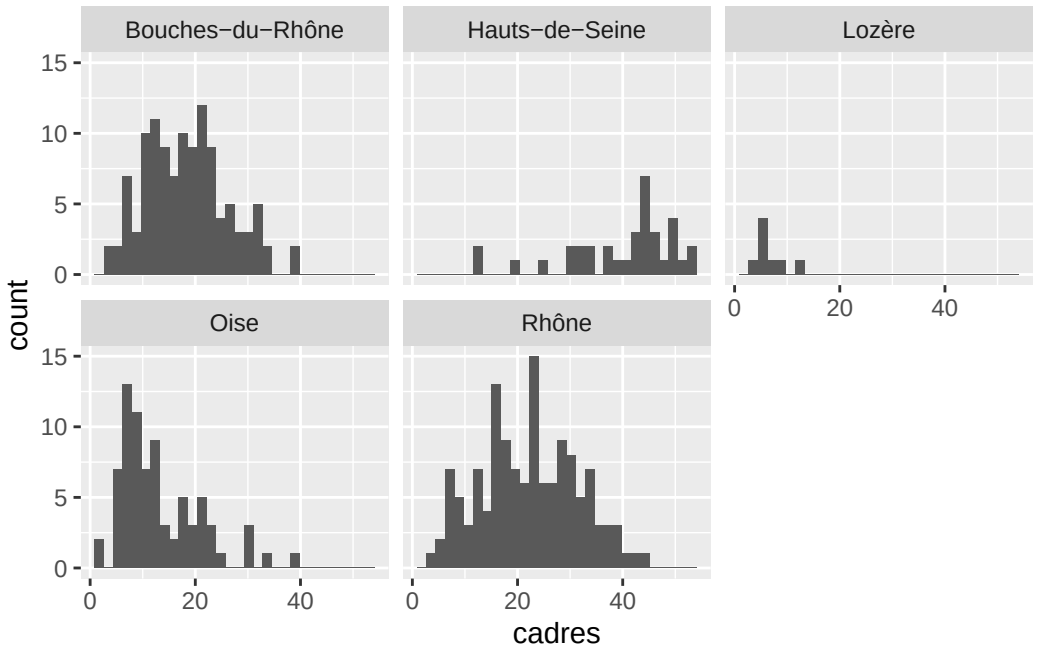
```
ggplot(data = rp) +
    geom_histogram(aes(x = cadres))
```



On souhaite comparer cette distribution de la part des cadres selon le département, et donc faire un histogramme pour chacun de ces départements. C'est ce que permettent les fonctions `facet_wrap` et `facet_grid`.

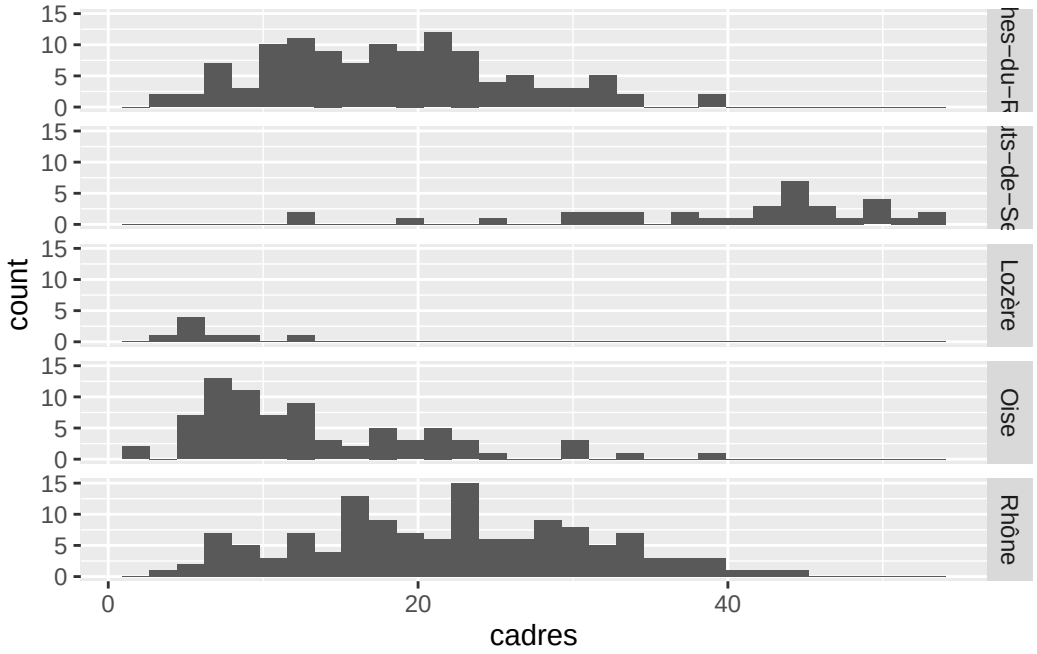
`facet_wrap` prend un paramètre de la forme `vars(variable)`, où `variable` est le nom de la variable en fonction de laquelle on souhaite faire les différents graphiques. Ceux-ci sont alors affichés les uns à côté des autres et répartis automatiquement dans la page.

```
ggplot(data = rp) +
    geom_histogram(aes(x = cadres)) +
    facet_wrap(vars(departement))
```



Pour `facet_grid`, les graphiques sont disposés selon une grille. La fonction prend alors deux arguments, `rows` et `cols`, auxquels on passe les variables à afficher en ligne ou en colonne via la fonction `vars()`.

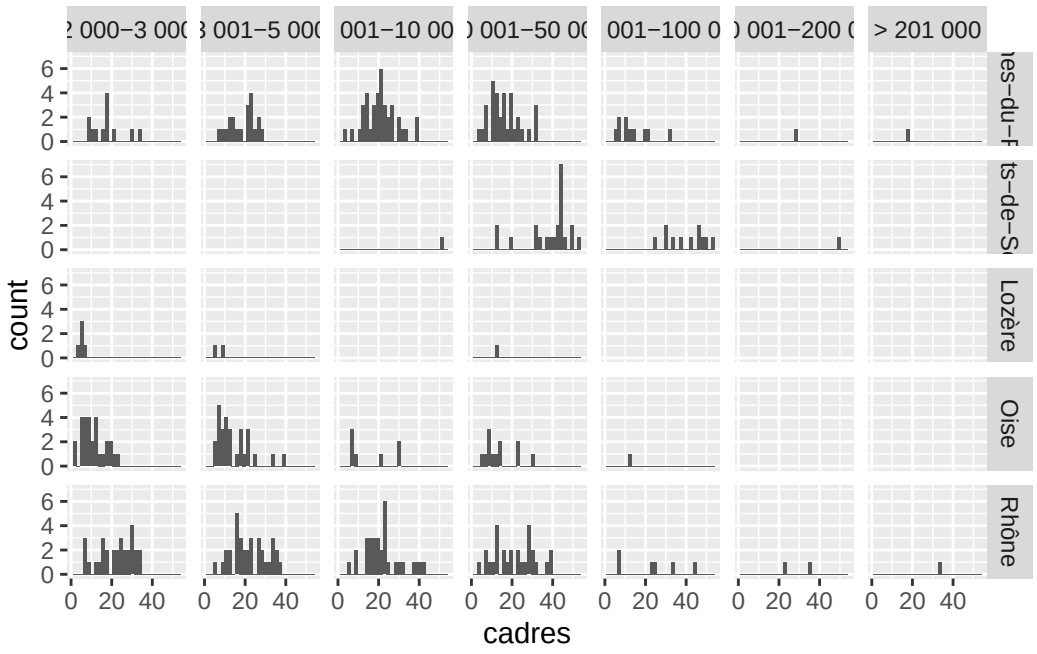
```
ggplot(data = rp) +
    geom_histogram(aes(x = cadres)) +
    facet_grid(rows = vars(departement))
```



Un des intérêts du faceting dans `ggplot2` est que tous les graphiques générés ont les mêmes échelles, ce qui permet une comparaison directe.

Enfin, notons qu'on peut même faire du faceting sur plusieurs variables à la fois. On peut par exemple faire des histogrammes de la répartition de la part des cadres pour chaque croisement des variables `departement` et `pop_cl` :

```
ggplot(data = rp) +
    geom_histogram(aes(x = cadres)) +
    facet_grid(
        rows = vars(departement), cols = vars(pop_cl)
    )
```



L'histogramme en haut à gauche représente la répartition du pourcentage de cadres parmi les communes de 2000 à 3000 habitants dans les Bouches-du-Rhône, etc.

4.7 Ressources

La documentation officielle (en anglais) de `ggplot2` est très complète et accessible en ligne.

Une "antisèche" (en anglais) résumant en deux pages l'ensemble des fonctions et arguments et disponible soit directement depuis RStudio (menu *Help* > *Cheatsheets* > *Data visualization with ggplot2*) ou en ligne.

Les parties Data visualisation et Graphics for communication de l'ouvrage en ligne *R for data science*, de Hadley Wickham, sont une très bonne introduction à `ggplot2`.

Plusieurs ouvrages, toujours en anglais, abordent en détail l'utilisation de `ggplot2`, en particulier ggplot2: Elegant Graphics for Data Analysis, toujours de Hadley Wickham, et le R Graphics Cookbook de Winston Chang.

Le site associé à ce dernier ouvrage comporte aussi pas mal d'exemples et d'informations intéressantes.

Enfin, si `ggplot2` présente déjà un très grand nombre de fonctionnalités, il existe aussi un système d'extensions permettant d'ajouter des `geom`, des thèmes, etc. Le site ggplot2 extensions est une très bonne ressource pour les parcourir et les découvrir, notamment grâce à sa galerie.

5 Les mesures

Les mesures de tendance centrale permettent de déterminer où se situe le "centre" des données. Les trois mesures de tendance centrale sont le mode, la moyenne et la médiane.

5.1 Les mesures de tendance centrale

5.1.1 Le mode

Le mode est la **modalité**, **valeur** ou **classe** possédant la plus grande fréquence. En d'autres mots, c'est la donnée la plus fréquente.

Puisque le mode se préoccupe seulement de la donnée la plus fréquente, il n'est pas influencé par les valeurs extrêmes.

Lorsque le mode est une classe, il est appelé **classe modale**.

Le mode est noté **Mo**.

Le langage `R` ne possède pas de fonction permettant de calculer le mode. La façon la plus simple de le calculer est d'utiliser la fonction `table` de `R`.

Par exemple, si nous voulons connaître le mode de la variable `marital` de la base de données `gss_cat`:

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

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

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

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

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

Dans le cas d'une variable quantitative continue, pour calculer le mode, il faut commencer par séparer les données en classes. Nous utiliserons les mêmes classes utilisées à la section:

```
carat_class = cut(diamonds$carat,
                  breaks = seq(from = 0, to = 6, by = 1),
                  right = FALSE)
table(carat_class)
#> carat_class
#> [0,1) [1,2) [2,3) [3,4) [4,5) [5,6)
#> 34880 16906  2114    34     5     1
```

La classe modale est donc la classe *[0,1)* avec une fréquence de 34880.

5.1.2 La médiane

La médiane, notée **Md**, est la valeur qui sépare une série de données classée en ordre croissant en deux parties égales.

La médiane étant la valeur du milieu, elle est la valeur où le pourcentage cumulé atteint 50%.

Puisque la médiane se préoccupe seulement de déterminer où se situe le centre des données, elle n'est pas influencée par les valeurs extrêmes. Elle est donc une mesure de tendance centrale plus fiable que la moyenne.

> Important : La médiane n'est définie que pour les variables quantitatives. En effet, si vous tentez d'utiliser la médiane pour des données autres que numériques, `R` vous donnera un message d'erreur.

La fonction `median` permet de calculer la médiane en langage `R`.

Par exemple, pour calculer la médiane de la variable `carat` de la base de données `diamonds`, nous avons:

```
median(diamonds$carat)
#> [1] 0.7
```

Ceci signifie que 50% des diamants ont une valeur en carat inférieure ou égale à 0.7 et que 50% des diamants ont une valeur en carat supérieure ou égale à 0.7.

Nous pouvons aussi obtenir que la médiane de la variable `price` de la base de données `diamonds` est donnée par:

```
median(diamonds$price)
#> [1] 2401
```

5.1.3 La moyenne

La moyenne est la valeur qui pourrait remplacer chacune des données d'une série pour que leur somme demeure identique. Intuitivement, elle représente le centre d'équilibre d'une série de données. La somme des distances qui sépare les données plus petites que la moyenne devrait être la même que la somme des distances qui sépare les données plus grandes.

> Important : La moyenne n'est définie que pour les variables quantitatives. En effet, si vous tentez d'utiliser la moyenne pour des données autres que numériques, `R` vous donnera un message d'erreur.

La fonction `mean` permet de calculer la moyenne en langage `R`.

Par exemple, pour calculer la moyenne de la variable `carat` de la base de données `diamonds`, nous avons:

```
mean(diamonds$carat)
#> [1] 0.7979397
```

Nous pouvons aussi obtenir que la moyenne de la variable `price` de la base de données `diamonds` est donnée par:

```
mean(diamonds$price)
#> [1] 3932.8
```

5.2 Les mesures de dispersion

Les mesures de tendance centrale (mode, moyenne et médiane) ne permettent pas de déterminer si une série de données est principalement située autour de son centre, ou si au contraire elle est très dispersée.

Les mesures de dispersion, elles, permettent de déterminer si une série de données est centralisée autour de sa moyenne, ou si elle est au contraire très dispersée.

Les mesures de dispersion sont l'étendue, la variance, l'écart-type et le coefficient de variation.

5.2.1 L'étendue

La première mesure de dispersion, l'étendue, est la différence entre la valeur maximale et la valeur minimale.

L'étendue ne tenant compte que du maximum et du minimum, elle est grandement influencée par les valeurs extrêmes. Elle est donc une mesure de dispersion peu fiable.

La fonction `range` permet de calculer l'étendue d'une variable en langage `R`.

Par exemple, pour calculer l'étendue de la variable `carat` de la base de données `diamonds`, nous avons:

```
range(diamonds$carat)
#> [1] 0.20 5.01
```

Nous pouvons donc calculer l'étendue de la variable `carat` en soustrayant les deux valeurs obtenues par la fonction `range`, c'est-à-dire que l'étendue est 5.01-0.2 = 4.81.

5.2.2 La variance

La variance sert principalement à calculer l'écart-type, la mesure de dispersion la plus connue.

> Attention : Les unités de la variance sont des unités^2.

La fonction `var` permet de calculer la variance d'une variable en langage `R`.

Par exemple, pour calculer la variance de la variable `carat` de la base de données `diamonds`, nous avons:

```
var(diamonds$carat)
#> [1] 0.2246867
```

Ceci signifie que la variance de la variable `carat` est 0.2246867 carat^2.

5.2.3 L'écart-type

L'écart-type est la mesure de dispersion la plus couramment utilisée. Il peut être vu comme la « moyenne » des écarts entre les données et la moyenne.

Puisque l'écart-type tient compte de chacune des données, il est une mesure de dispersion beaucoup plus fiable que l'étendue.

Il est défini comme la racine carrée de la variance.

La fonction `sd` permet de calculer l' 'écart-type d'une variable en langage `R`.

Par exemple, pour calculer l'écart-type de la variable `carat` de la base de données `diamonds`, nous avons:

```
sd(diamonds$carat)
#> [1] 0.4740112
```

Ceci signifie que l'écart-type de la variable `carat` est 0.4740112 carat.

5.2.4 Le coefficient de variation

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

$$C.V. = \frac{\text{ecart-type}}{\text{moyenne}} \times 100\% \tag{5.1}$$

Si le coefficient est inférieur à 15%, les données sont dites **homogènes**. Cela veut dire que les données sont situées près les unes des autres.

Dans le cas contraire, les données sont dites **hétérogènes**. Cela veut dire que les données sont très dispersées.

> Important : Le coefficient de variation ne possède pas d'unité, outre le symbole de pourcentage.

Il n'existe pas de fonctions en `R` permettant de calculer directement le coefficient de variation. Par contre, nous pouvons utiliser en conjonction les fonctions `sd` et `mean` pour le calculer.

Par exemple, pour calculer le coefficient de variation de la variable `carat` de la base de données `diamonds`, nous avons:

```
sd(diamonds$carat)/mean(diamonds$carat)*100
#> [1] 59.40439
```

Le C.V. de la variable `carat` est donc 59.4043906 %, ce qui signifie que les données sont hétérogènes, car le coefficient de variation est plus grand que 15%.

5.3 Les mesures de position

Les mesures de position permettent de situer une donnée par rapport aux autres. Les différentes mesures de position sont la cote Z, les quantiles et les rangs.

Tout comme les mesures de dispersion, celles-ci ne sont définies que pour une variable quantitative.

5.3.1 La cote z

Cette mesure de position se base sur la moyenne et l'écart-type.

La cote Z d'une donnée x est calculée comme suit :

$$Z = \frac{x - \text{moyenne}}{\text{ecart-type}} \tag{5.2}$$

Important : La cote z ne possède pas d'unités.

Une cote Z peut être positive, négative ou nulle.

Cote Z	Interprétation
Z>0	donnée supérieure à la moyenne
Z<0	donnée inférieure à la moyenne
Z=0	donnée égale à la moyenne

Il n'existe pas de fonctions en `R` permettant de calculer directement la cote Z. Par contre, nous pouvons utiliser en conjonction les fonctions `sd` et `mean` pour la calculer.

Par exemple, si nous voulons calculer la cote Z d'un diamant de 3 carats, nous avons:

```
(3-mean(diamonds$carat))/sd(diamonds$carat)
#> [1] 4.645587
```

5.3.2 Les quantiles

Un quantile est une donnée qui correspond à un certain pourcentage cumulé.

Parmi les quantiles, on distingue les quartiles, les quintiles, les déciles et les centiles.

- Les quartiles Q_1, Q_2 et Q_3, séparent les données en quatre parties égales. Environ 25% des données sont inférieures ou égales à Q_1. Environ 50% des données sont inférieures ou égales à Q_2. Environ 75% des données sont inférieures ou égales à Q_3.
- Les quintiles V_1, V_2, V_3 et V_4, séparent les données en cinq parties égales. Environ 20% des données sont inférieures ou égales à V_1. Environ 40% des données sont inférieures ou égales à V_2. Etc.
- Les déciles D_1, D_2, ..., D_8 et D_9, séparent les données en dix parties égales. Environ 10% des données sont inférieures ou égales à D_1. Environ 20% des données sont inférieures ou égales à D_2. Etc.
- Les centiles C_1, C_2, ..., C_{98} et C_{99}, séparent les données en cent parties égales. Environ 1% des données sont inférieures ou égales à C_1. Environ 2% des données sont inférieures ou égales à C_2. Etc.

  Il est utile de noter que certains quantiles se recoupent.

La fonction `quantile` permet de calculer n'importe quel quantile d'une variable en langage `R`. Il suffit d'indiquer la variable étudiée ainsi que le pourcentage du quantile voulu.

Par exemple, si nous voulons calculer D_1 pour la variable `carat`, nous allons utiliser la fonction `quantile` avec une probabilité de 0,1.

```
quantile(diamonds$carat, 0.1)
#>  10%
#> 0.31
```

Ceci implique que 10% des diamants ont une valeur en carat inférieure ou égale à 0.31 carat.

Nous pouvons calculer le troisième quartile Q_3 de la variable `price` en utilisant la fonction `quantile` avec une probabilité de 0,75.

```
quantile(diamonds$price, 0.75)
#>     75%
#> 5324.25
```

Ceci implique que 75% des diamants ont un prix en dollars inférieur ou égal à 5324.25 $.

5.3.3 La commande `summary`

La commande `summary` produit un sommaire contenant six mesures importantes:

1. `Min` : le minimum de la variable
2. `1st Qu.`: Le premier quartile, Q_1, de la variable
3. `Median` : La médiane de la variable
4. `Mean` : La moyenne de la variable
5. `3rd Qu.` : Le troisième quartile, Q_3, de la variable
6. `Max` : Le maximum de la variable

Nous pouvons donc produire le sommaire de la variable `price` de la base de données `diamonds` de la façon suivante:

```
summary(diamonds$price)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>     326     950    2401    3933    5324   18823
```

5.3.4 Le rang centile

Un rang centile représente le pourcentage cumulé, *exprimé en nombre entier*, qui correspond à une certaine donnée. Nous déterminerons les rangs centiles pour les variables continues seulement.

Les rangs centiles sont donc exactement l'inverse des centiles.

Il n'existe pas de fonctions dans `R` permettant de trouver directement le rang centile, mais il est facile d'utiliser la fonction `mean` pour le trouver.

Par exemple, si nous voulons trouver le rang centile d'un diamant qui coûte 500$, il suffit d'utiliser la commande suivante. La commande calcule la moyenne de toutes les valeurs en dollars des diamants coûtant 500$ ou moins.

```
floor(mean(diamonds$price<=500)*100)
#> [1] 3
```

Ceci signifie que pour un diamant de 500$, il y a 3 % des diamants qui ont une valeur égale ou inférieure.

partie III

L'estimation et les tests d'hypothèses

6 L'estimation de paramètres

Nous allons utiliser la librairie `infer` pour faire de l'estimation de paramètres, ainsi que la base de données `gss`, présente dans la librairie.

```
library(infer)
data(gss)
```

6.1 L'intervalle de confiance sur une moyenne

Pour trouver un intervalle de confiance sur une moyenne, nous utilisons la fonction `t_test`.

Les quatres arguments nécessaires sont:

- `x`: la base de données à utiliser, sous forme de *tibble*.
- `response`: la variable quantitative dont on veut connaître l'intervalle de confiance pour la moyenne.
- `alternative`: pour un intervalle de confiance, on utilise toujours la valeur `two-sided`.
- `conf_level`: un niveau de confiance entre 0 et 1.

Par exemple, si on veut trouver un intervalle de confiance à 95% pour la moyenne de la variable `age`, nous utilisons:

```
t_test( x = gss,
        response = age,
        alternative = "two-sided",
        conf_level = 0.95)
#> # A tibble: 1 x 7
#>   statistic  t_df   p_value alternative estimate lower_ci upper_ci
#>       <dbl> <dbl>     <dbl> <chr>          <dbl>    <dbl>    <dbl>
#> 1      67.6   499 2.42e-253 two.sided       40.3     39.1     41.4
```

La borne inférieure de l'intervalle de confiance est donnée par la variable `lower_ci` et la borne supérieure par la variable `upper_ci`. Dans notre test, nous avons donc un intervalle de confiance entre 39.095674 et 41.436326.

6.2 L'intervalle de confiance sur une proportion

Pour trouver un intervalle de confiance sur une proportion, nous utilisons la fonction `prop_test`.

Les cinq arguments nécessaires sont:

- `x`: la base de données à utiliser, sous forme de *tibble*.
- `response`: la variable quantitative dont on veut connaître l'intervalle de confiance pour la proportion.
- `success`: la modalité de la variable que nous considérons comme un succès.
- `alternative`: pour un intervalle de confiance, on utilise toujours la valeur `two-sided`.
- `conf_level`: un niveau de confiance entre 0 et 1.

Par exemple, si on veut trouver un intervalle de confiance à 95% pour la proportion de `female` de la variable `age`, nous utilisons:

```
prop_test(  x = gss,
            response = sex,
            success = "female",
            alternative = "two-sided",
            conf_level = 0.95)
#> No `p` argument was hypothesized, so the test will assume a null hypothesis `p
#> = .5`.
#> # A tibble: 1 x 6
#>   statistic chisq_df p_value alternative lower_ci upper_ci
#>       <dbl>    <int>   <dbl> <chr>          <dbl>    <dbl>
#> 1      1.25        1   0.264 two.sided      0.430    0.519
```

```
#> No `p` argument was hypothesized, so the test will assume a null hypothesis `p
#> = .5`.
```

La borne inférieure de l'intervalle de confiance est donnée par la variable `lower_ci` et la borne supérieure par la variable `upper_ci`. Dans notre test, nous avons donc un intervalle de confiance entre 0.4296103 et 0.5187952.

7 Les tests d'hypothèses

Nous allons utiliser la librairie `infer` pour faire des tests d'hypothèses, ainsi que la base de données `gss`, présente dans la librairie.

```
library(infer)
data(gss)
```

7.1 Les tests d'hypothèses sur une moyenne

Pour effectuer un test d'hypothèses sur une moyenne, nous utilisons la fonction `t_test`.

Les quatres arguments nécessaires sont:

- `x`: la base de données à utiliser, sous forme de *tibble*.
- `response`: la variable dont on veut tester l'hypothèse.
- `mu`: la valeur de la moyenne à l'hypothèse H_0.
- `alternative`:
  - `less`: pour un test unilatéral à gauche
  - `two-sided`: pour un test bilatéral
  - `greater`: pour un test unilatéral à droite

Par exemple, si on veut tester si la moyenne de la variable `age` est plus grande que 39 ans, à un niveau de confiance de 98%, nous utilisons:

```
t_test( x = gss,
        response = age,
        mu = 39,
        alternative = "greater"
)
#> # A tibble: 1 x 7
#>   statistic  t_df p_value alternative estimate lower_ci upper_ci
#>       <dbl> <dbl>   <dbl> <chr>          <dbl>    <dbl>    <dbl>
#> 1      2.13   499  0.0170 greater         40.3     39.3      Inf
```

La variable `p_value` nous permet de conserver ou de rejetter H_0. En effet, dans notre cas, la valeur est de 0.0170242 qui est plus petite que le risque d'erreur de 2% (associé au niveau de confiance de 98%). On rejette donc H_0 et on accepte H_1, l'âge moyen est plus grand que 39 ans.

7.2 Les tests d'hypothèses sur une proportion

Pour effectuer un test d'hypothèses sur une proportion, nous utilisons la fonction `prop_test`.

Les quatres arguments nécessaires sont:

- `x`: la base de données à utiliser, sous forme de *tibble*.
- `response`: la variable dont on veut connaître l'intervalle de confiance pour la proportion.
- `success`: la modalité de la variable que nous considérons comme un succès.
- `alternative`:
  - `less`: pour un test unilatéral à gauche
  - `two-sided`: pour un test bilatéral
  - `greater`: pour un test unilatéral à droite

Par exemple, si on veut tester si la proportion de `female` de la variable `sex` est plus petite que 51%, à un niveau de confiance de 99%, nous utilisons:

```
prop_test(  x = gss,
            response = sex,
            success = "female",
            alternative = "less",
            p = 0.51)
#> # A tibble: 1 x 4
#>   statistic chisq_df p_value alternative
#>       <dbl>    <int>   <dbl> <chr>
#> 1      2.45        1  0.0587 less
```

La variable `p_value` nous permet de conserver ou de rejetter H_0. En effet, dans notre cas, la valeur est de 0.0587257 qui est plus grande que le risque d'erreur de 1% (associé au niveau de confiance de 99%). On conserve donc H_0, la proportion de femmes ne semble pas être plus petite que 51%.

7.3 Les tests d'hypothèses sur une différence de moyennes