RÉPUBLIQUE DU SÉNÉGAL

Un peuple- un But- une Foi

****

Agence Nationale de la Statistique et de la Démographie

ANSD

Agence Nationale de la Statistique et de la Démographie

Ecole Nationale de la Statistique et de l'Analyse Economique Pierre Ndiaye

ensae

Traitement statistique avec le logiciel R

Visualisation des données avec la bibliothèque ggplot2

Rédigé par :

ZAONGO INOUSSA

Élève ingénieur statisticien économiste

Sous la supervision de :

M. HADY DIALLO

Enseignant à l'ENSAE

30-04-2023

Avant-propos

Créée en 2008, l'École Nationale de la Statistique et de l'Analyse Économique (ENSAE) est une grande école de statistique à caractère sous-régional située à Dakar (capitale du Sénégal). Elle constitue une direction de l'Agence Nationale de la Statistique et de la Démographie (ANSD), qui est la structure principale du Système Statistique Nationale du Sénégal. L'ENSAE forme avec l'Institut Sous-régional de Statistique et d'Économie Appliquée (ISSEA-Yaoundé), l'École Nationale Supérieure de Statistique et d'Économie Appliquée (ENSEA-Abidjan) et l'École Nationale d'Économie Appliquée et de Management (ENEAM-Cotonou) le réseau des grandes écoles de statistiques africaines. Ces quatre écoles sont sous la coordination du Centre d'Appui aux Écoles de Statistiques Africaines (CAPESA) basé en France. L'ENSAE a pour vocation de former et de mettre à la disposition de tous les pays africains des statisticiens et des économistes qualifiés. Ainsi, elle propose des formations qui se déroulent dans trois cycles à savoir les Analystes Statisticiens (AS), les Ingénieurs des Travaux Statistiques (ITS) et les Ingénieurs Statisticiens Economistes (ISE). Les élèves Ingénieurs Statisticiens Economistes qui suivent une formation de ISE cycle long (05 ans), se doivent de faire le module logiciel R pour apprendre à utiliser et à traiter les données statistiques avec ce logiciel et à synthétiser les données à travers des graphiques afin de faciliter leur comprehension .
C'est dans cette optique que s'inscrit notre exposé dont le thème est : VISUALISATION DES DONNEES AVEC LA BIBLIOTHEQUE GGPLOT2 dont le but est de présenter comment créer des graphiques de manière simple,efficace et esthétique avec le logiciel R.

Remerciements

Nos remerciements à l'administration, aux enseignants, à tous ceux qui luttent pour l'avancée de la Statistique et de l'informatique, à nos camarades de l'ENSAE-SENEGAL, ainsi qu'à tous ceux qui ont contribué de près ou de loin à la réalisation de ce document.

source:BM

Table des matières

Introduction

R est un langage de programmation pour l'analyse de données,la modélisation des données et la visualisation graphique. R peut être utilise comme un langage oriente objet tout comme un environnement statistique dans lequel des listes d'instructions peuvent être exécutées en séquence sans l'intervention de l'utilisateur.

Le package ggplot2 a été publié pour la première fois en 2006 sous le nom de ggplot. Il fut amélioré de façon importante et renommé ggplot2 en 2007. Son créateur est Hadley Wickham, qui est aussi derrière plusieurs des packages du tidyverse, duquel ggplot2 fait partie. Le package est maintenant développé par toute une équipe, dont des employés de RStudio. Il implémente la grammaire graphique présentée dans : « Wilkinson, L. (2005). The grammar of graphics, 2e édition. Springer ».

Le package ggplot2 a été conçu en ayant comme objectif la simplicité d'utilisation et la qualité des graphiques produits. Il reprend les forces suivantes des systèmes graphiques R précédents :

- système de base : création de graphiques par **couches** (ajouts séquentiels d'éléments);
- package lattice : **représentations multivariées simples**.

tout en apportant les améliorations suivantes :

- **esthétique** par défaut pensée de façon à transmettre plus efficacement les informations contenues dans le graphique;
- **automatisation de certaines configurations** graphiques, notamment les légendes;
- **ajout de transformations statistiques** communes (p. ex. courbes de lissage, barres d'erreur) facilité.

Dans le package ggplot2, tout a été repensé pour être plus simple d'utilisation et surtout pour que le graphique produit transmette plus efficacement l'information qu'il contient.

L'utilisation du package ggplot2 est présentée ici de façon brève, mais avec tout de même assez de détails pour démarrer un apprentissage de cet outil aux possibilités vastes.

1 Généralités

1.1 Installation du package ggplot2

Comme tout package,il faut commencer par l'importer.pour cela il y'a plusieurs solutions:

- Utiliser l'onglet package sous R studio puis le sous onglet install.Une fois installé vous devez charger la library ggplot2 comme ci dessous.

```
library(ggplot2)
```

- installer et charger le package tidyverse car ggplot2 y est inclu.
- Par code

```
install.packages("gglot2") #installé le package ggplot2

library(ggplot2) #charger la library ggplot2
```

1.2 Présentation des données utilisées pour les exemples.

Comme dans les autres notes sur les graphiques en R, les données quakes auxquelles deux facteurs sont ajoutés sont utilisées dans les exemples de cette fiche.En effet,la base de données **quakes** est un ensemble de données intégré dans le logiciel R qui fournit des informations sur 1000 tremblements de terre enregistrés près de l'île de Fidji.Elle contient des informations sur les tremblements de terre qui ont été enregistrés entre 1964 et 1974 sur la planète Terre.Cette base compte 1000 observations pour 7 variables.Les principales variables sont:

- lat : la latitude géographique de l'emplacement du séisme
- long : la longitude géographique de l'emplacement du séisme
- depth : la profondeur du séisme en kilomètres
- mag : la magnitude du séisme sur l'échelle de Richter
- stations : le nombre de stations sismiques qui ont enregistré le séisme.
- tsunami : un indicateur binaire (0 ou 1) qui indique si un tsunami a été déclenché par le séisme
- mag_region categorisation du magnitude du séisme
- region :indique les regions sismique en fonction de la longitude.

```
data("quakes")
quakes$mag_catego <- factor(floor(quakes$mag))
quakes$region <- factor(
  ifelse(quakes$long < 175, yes = "Ouest", no = "Est"),
  levels = c("Ouest", "Est")
)
str(quakes)
```

```
## 'data.frame':    1000 obs. of  7 variables:
##  $ lat       : num  -20.4 -20.6 -26 -18 -20.4 ...
##  $ long      : num  182 181 184 182 182 ...
##  $ depth     : int  562 650 42 626 649 195 82 194 211 622 ...
##  $ mag       : num  4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
##  $ stations  : int  41 15 43 19 11 12 43 15 35 19 ...
##  $ mag_catego: Factor w/ 3 levels "4","5","6": 1 1 2 1 1 1 1 1 1 1 ...
##  $ region    : Factor w/ 2 levels "Ouest","Est": 2 2 2 2 2 2 1 2 2 2 ...
```

2 Principes de base de ggplot2

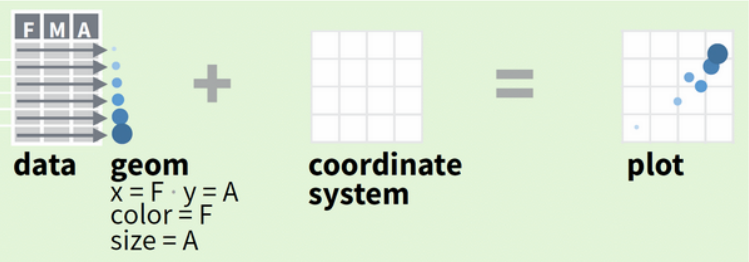
2.1 Grammaire graphique

La grammaire graphique est une approche systématique pour construire des graphiques, qui repose sur la composition de couches de graphiques élémentaires. Le principe de base derrière la grammaire graphique (d'où le gg dans ggplot) est qu'un graphique statistique est une représentation de données, dans un système de coordonnées spécifique, divisée en éléments de base :

- éléments géométriques (geoms) : points, lignes, barres, etc.; propriétés visuelles (aesthetics) des éléments géométriques : axes, couleurs, formes, tailles, etc.
- transformations statistiques, si désiré : courbe de régression ou de lissage, région d'erreur, etc.

Un graphique est spécifié en associant des variables, provenant des données, à des propriétés visuelles des éléments géométriques du graphique. Ces principes sont représentés comme suit sur la feuille de triche officielle du package.

Figure 1: Principe



source: https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf

2.2 Principales fonctions et gabarit de code

Les principales fonctions du package ggplot2 sont les suivantes :

- ggplot [1] : initialisation d'un objet de classe ggplot;
- qplot [2] : initialisation rapide (q pour quick) d'un objet ggplot;
  - – : opérateur pour l'ajout de couches ou la modification de configurations dans un objet ggplot;
- fonctions de type geom_* (p. ex. geom_point, geom_boxplot, geom_bar, etc.) : spécification de couches à ajouter à un graphique;
- aes : création d'un mapping, soit une association entre des propriétés visuelles et des variables;
- ggsave : enregistrement d'un graphique.

Notons que l'utilisation de la fonction qplot est plus intuitive que celle de ggplot pour des gens familiers avec plot. La fonction qplot n'offre cependant pas toutes les possibilités de ggplot. Elle ne sera pas couverte dans ce document.

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

ou encore la version « tout sur la même ligne » :

```
ggplot(data = <DATA>) + <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

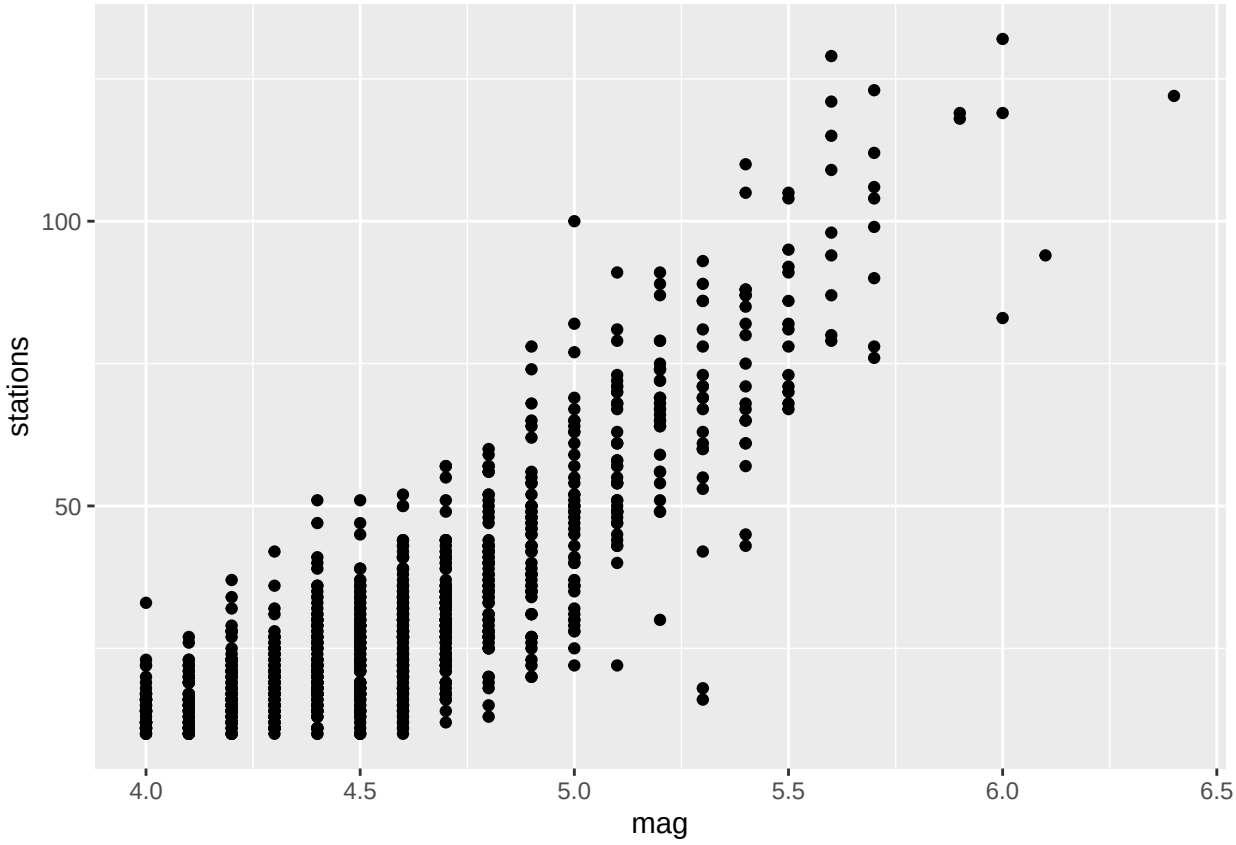
Dans ce gabarit, il faut remplacer les éléments entre < et > comme suit :

[1] grammar of graphics plot
[2] quick plot:qui permet de créer des graphiques rapidement et facilement en utilisant une syntaxe simplifiée

- **DATA** : jeu de données, stocké dans un data frame (ou un tibble), dans lequel les variables catégoriques doivent être des facteurs (dont les libellés des niveaux ont avantage à être informatifs, car ils apparaîtront dans le graphique);
- **GEOM_FUNCTION** : le nom d'une fonction de type geom_* pour ajouter une couche au graphique;
- **MAPPINGS** : arguments à fournir à la fonction aes (les arguments acceptés varient un peu selon la).

Voici un premier exemple simple. Nous allons utiliser la base quakes pour representer un nuage de point de la magnetude du séisme en fonction du nombre de stations sismiques qui ont enregistré le séisme.

```
ggplot(data = quakes) + geom_point(mapping = aes(x = mag, y = stations))
```



2.2.1 Quelques fonctions de type geom

NB: ***La plupart de ces fonctions cachent des transformations statistiques (p. ex. geom_bar et geom_histogram calculent des fréquences, geom_boxplot calcule des quantiles, geom_density estime une densité, etc.)***

2.2.2 Quelques autres fonctions communes

Voici quelques autres fonctions communes de ggplot2 servant à ajouter des couches ou modifier des configurations dans un objet ggplot initialisé :

- fonctions labs, ggtitle, xlab, ylab : ajouter un titre, modifier les noms d'axes; fonctions de type coord_* : modifier des configurations reliées au système de coordonnées;
- fonctions de type facet_* : créer des grilles de sous-graphiques chacun des sous-graphiques est conditionnel à la valeur de facteurs(s), il représente donc seulement le sous-ensemble des observations ayant une modalité particulière pour ce(s) facteur(s);

Figure 2: fonction geom

Fonction	Type de graphique	Élément(s) ajouté(s)
`geom_point`	diagramme de dispersion	points selon des coordonnées
`geom_line`	diagramme en lignes	segments de droites reliant des points
`geom_bar`	diagramme à barres	barres disjointes, de hauteurs spécifiées ou calculées = fréquences des niveaux d'un facteur
`geom_histogram`	histogramme	barres collées, de hauteurs calculées = fréquences d'observations d'une variable numérique tombant dans des intervalles joints (*bin*)
`geom_boxplot`	diagramme en boîte	*boxplots*
`geom_density`	courbe de densité à noyau	courbe de la densité estimée par noyau (*kernel density*)
`geom_qq`	diagramme quantile-quantile théorique	points pour les couples de quantiles empiriques et théoriques
⋮	il en existe plusieurs autres	voir http://ggplot2.tidyverse.org/reference/

Source : http://seananderson.ca/ggplot2-FISH554/

- fonctions de type scale_* : modifier les échelles de certaines propriétés visuelles (p. ex. couleurs, formes, tailles, etc.)
- fonctions de type theme_* : modifier des configurations reliées à l'apparence du graphique;
- fonctions de type stat_* : ajouts d'éléments tirés d'un calcul mathématique ou statistique.

2.3 Objet de classe ggplot

Un graphique produit en ggplot2 est en fait un objet de classe ggplot. La fonction ggplot initialise un objet de cette classe.

```
mon_graph <- ggplot(data = quakes)
```

Nous pouvons ajouter des couches à un objet ggplot avec l'opérateur +

```
mon_graph <- mon_graph + geom_point(mapping = aes(x = mag, y = stations))
```

Il est possible d'observer le contenu de l'objet.

```
str(mon_graph)  # non évalué, car la sortie est longue
```

```
summary(mon_graph)
```

```
## data: lat, long, depth, mag, stations, mag_catego, region [1000x7]
## faceting: <ggproto object: Class FacetNull, Facet, gg>
##     compute_layout: function
##     draw_back: function
##     draw_front: function
##     draw_labels: function
##     draw_panels: function
##     finish_data: function
##     init_scales: function
##     map_data: function
##     params: list
##     setup_data: function
##     setup_params: function
```

```
##     shrink: TRUE
##     train_scales: function
##     vars: function
##     super:  <ggproto object: Class FacetNull, Facet, gg>
## -----------------------------------
## mapping: x = ~mag, y = ~stations
## geom_point: na.rm = FALSE
## stat_identity: na.rm = FALSE
## position_identity
```

C'est en affichant un objet de classe ggplot qu'un graphique est produit

La fonction ggsave permet quant à elle d'enregistrer un graphique ggpot2.

```
ggsave(file = "ExempleGraphique_ggplot2.pdf", plot = mon_graph)
```

```
## Saving 6.5 x 4.5 in image
```

NB: ***Le format du fichier à créer (p. ex. PDF, PNG, EPS, etc.) est déduit de l'extension du fichier si l'argument device n'est pas spécifié. Aussi, si aucune valeur n'est fourni à l'argument plot, c'est le dernier graphique ggpot2 produit qui est enregistré. La dimension du graphique enregistré peut être contrôlée à l'aide des arguments width, height et units.***

2.4 Démarche de création d'un graphique ggplot2

Voici une suggestion de démarche à suivre lors de la création d'une graphique ggplot2.

1-Planifier le travail - Répondre aux questions suivantes :

a-Je veux représenter quelles variables, de quel jeu de données ?

b-Je veux créer quel type de graphique ?

c-Quelle fonction de type geom_* me permettra de produire les éléments géométriques de ce graphique ?

d-Cette fonction accepte quelles propriétés visuelles ?

e.Je veux associer les variables concernées à quelles propriétés visuelles ?

2-Écrire et soumettre la première version du code de création du graphique.

3-Modifier le code de création du graphique pour ajuster la mise en forme (titre, nom d'axes, autres annotations, palette de couleur, etc.) selon mes besoins. - travail itératif : cycle « modification code » →

« jugement du graphique créé » répété jusqu'à l'obtention d'un résultat satisfaisant.

La planification effectuée avant de produire le premier exemple de graphique ggplot pourrait être représentée comme suit.

3 Quelques exemples

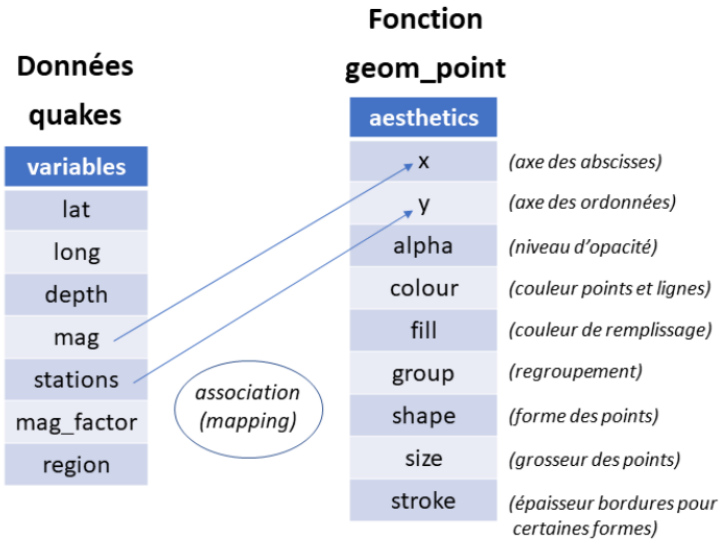
3.1 Exemples de configurations graphiques simples

3.1.1 Ajout d'un titre et de noms d'axes - fonction labs

Ajoutons un titre et des noms d'axes à note premier graphique ggplot.

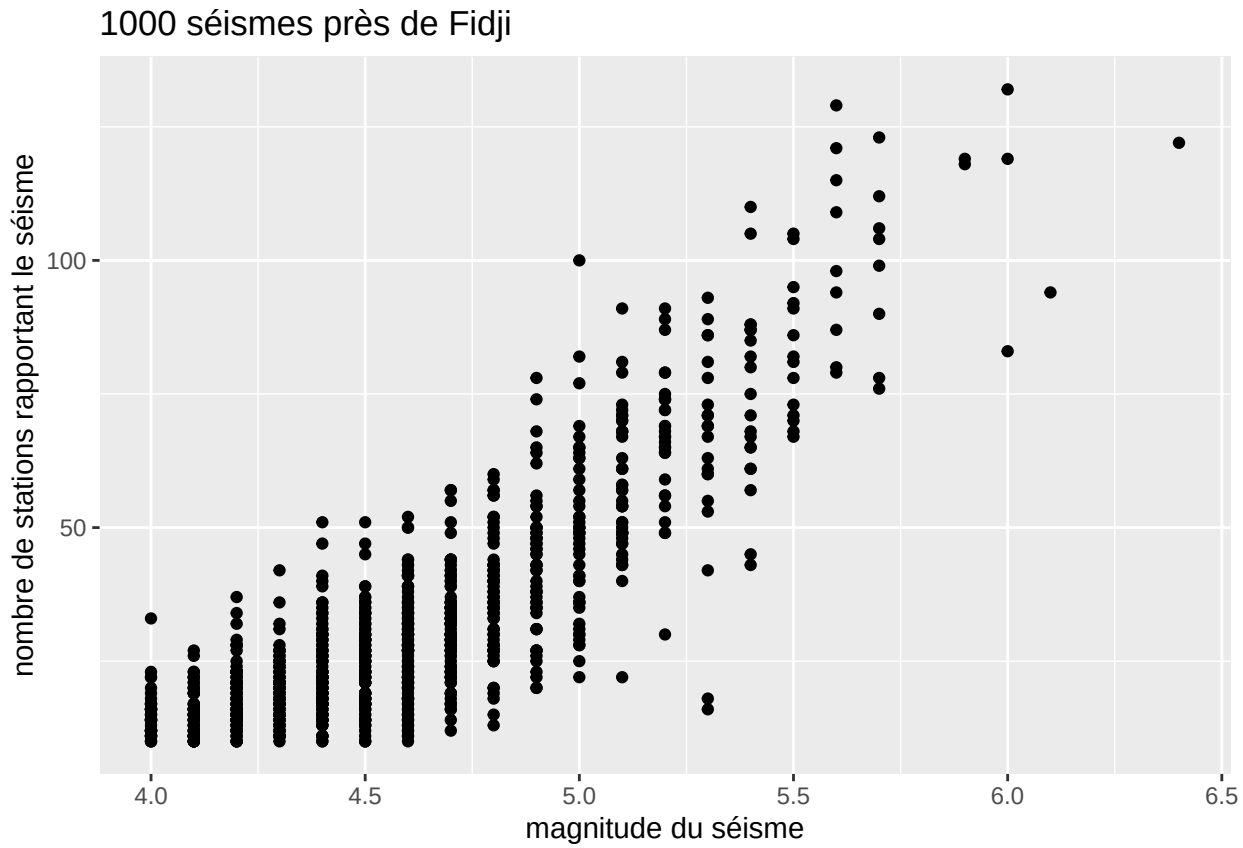
```
ggplot(data = quakes) +
  geom_point(mapping = aes(x = mag, y = stations)) +
  labs(
    title = "1000 séismes près de Fidji",          # ou ggtitle("...") +
    x = "magnitude du séisme",                     # ou xlab("...") +
```

Figure 3: schema



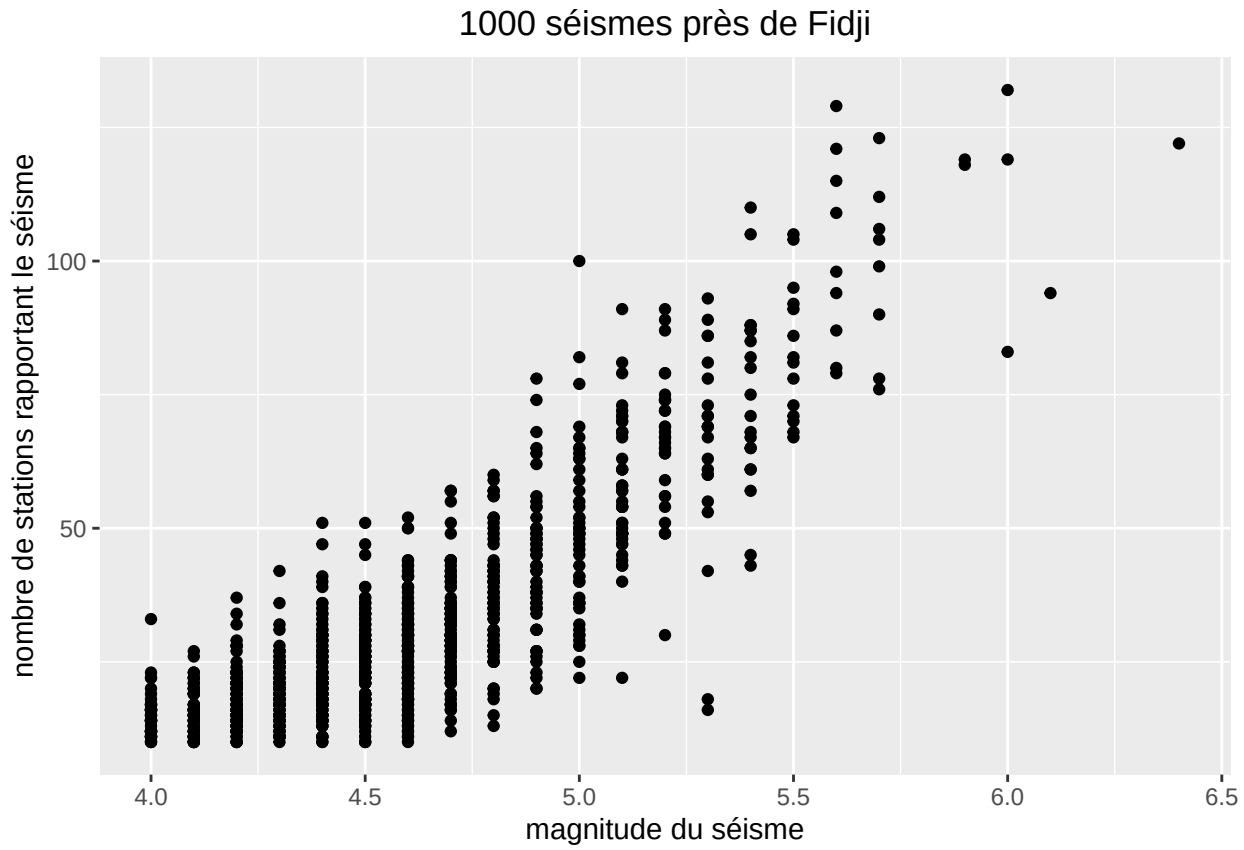
source: https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf

```
    y = "nombre de stations rapportant le séisme"  # ou ylab("...")
)
```



Vous avez envie de centrer le titre du graphique ? Voici comment faire

```r
ggplot(data = quakes) +
  geom_point(mapping = aes(x = mag, y = stations)) +
  labs(
    title = "1000 séismes près de Fidji",
    x = "magnitude du séisme",
    y = "nombre de stations rapportant le séisme"
  ) +
  theme(plot.title = element_text(hjust = 0.5))  # permet de centrer le titre
```

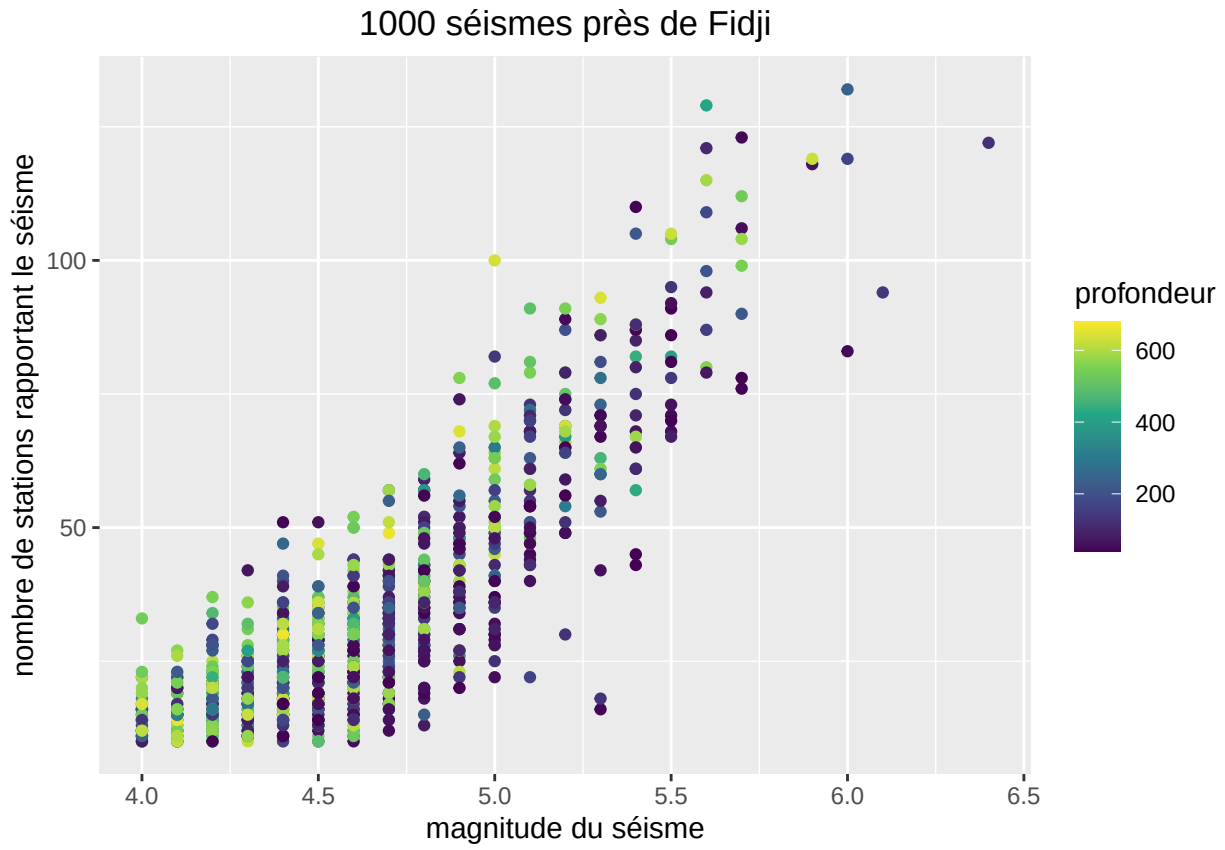


3.1.2 Ajout d'une variable associée à une propriété visuelle autre qu'un axe.

Le graphique précédent représente deux variables. Ajoutons une troisième variable au graphique, qui fera varier la couleur des points. Si la palette de couleur utilisée par défaut ne nous plaît pas, nous pouvons la changer.

```r
scatterplot <- ggplot(data = quakes) +
  geom_point(mapping = aes(
    x = mag,
    y = stations,
    colour = depth  # permet de faire varier la couleur des points en fonction de depth
  )) +
  labs(
    title = "1000 séismes près de Fidji",
    x = "magnitude du séisme",
    y = "nombre de stations rapportant le séisme",
    colour = "profondeur"  # permet de modifier le titre de la légende
```

```
  ) +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_colour_viridis_c()       # permet d'utiliser la palette de couleur viridis

scatterplot
```



3.1.3 Échelles de couleurs - fonctions de type scale-colour-* et scale-fill-*

Deux propriétés visuelles (aesthetics) permettent de spécifier des couleurs :

- colour = couleur de points et de lignes,
- fill = couleur de remplissage.

Les fonctions permettant de contrôler les palettes de couleurs utilisées pour ses propriétés visuelles ont toutes un nom débutant par scale_* suivi du nom de la propriété visuelle concernée et de _*. Le nom de la fonction peut se terminer par :

- viridis_d ou viridis_c : utiliser une palette de couleur offerte dans le package R viridisLite, soit discrète (_d) ou continue (_c);
- brewer ou distiller : utiliser une palette de couleur de ColorBrewer;
- grey : utiliser un dégradé de gris,
- manual : utiliser une palette discrète spécifiée manuellement;
- gradient, gradient2 ou gradientn : utiliser dégradé continu spécifié manuellement; etc.

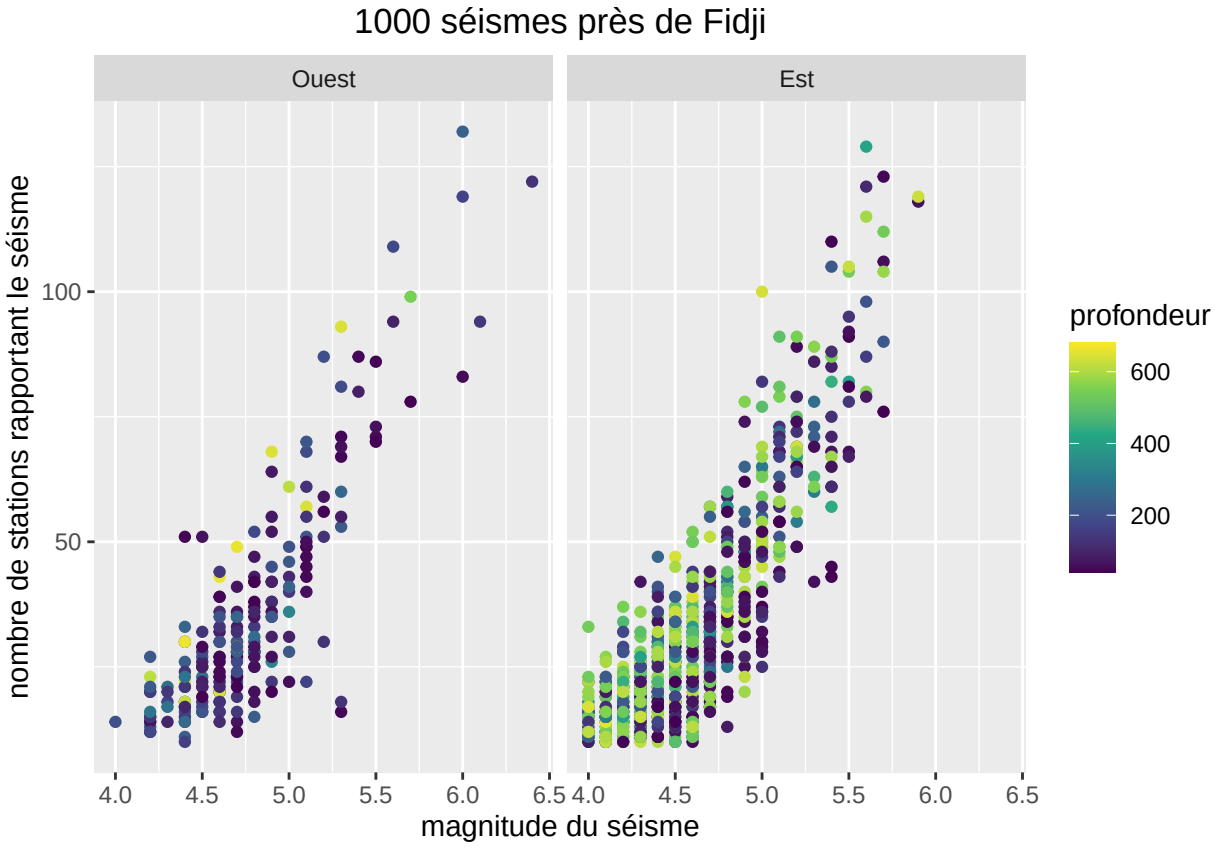
Une de ces fonctions a été utilisée dans l'exemple précédent : la fonction scale_colour_viridis_c.

3.1.4 Ajout de variables par l'intermédiaire d'une grille de sous-graphiques (facets)

- Graphiques côte à côte selon les niveaux d'une variable catégorique

Ajoutons maintenant une quatrième variable au graphique, celle-ci catégorique, en créant des graphiques disjoints selon le niveau de la variable.

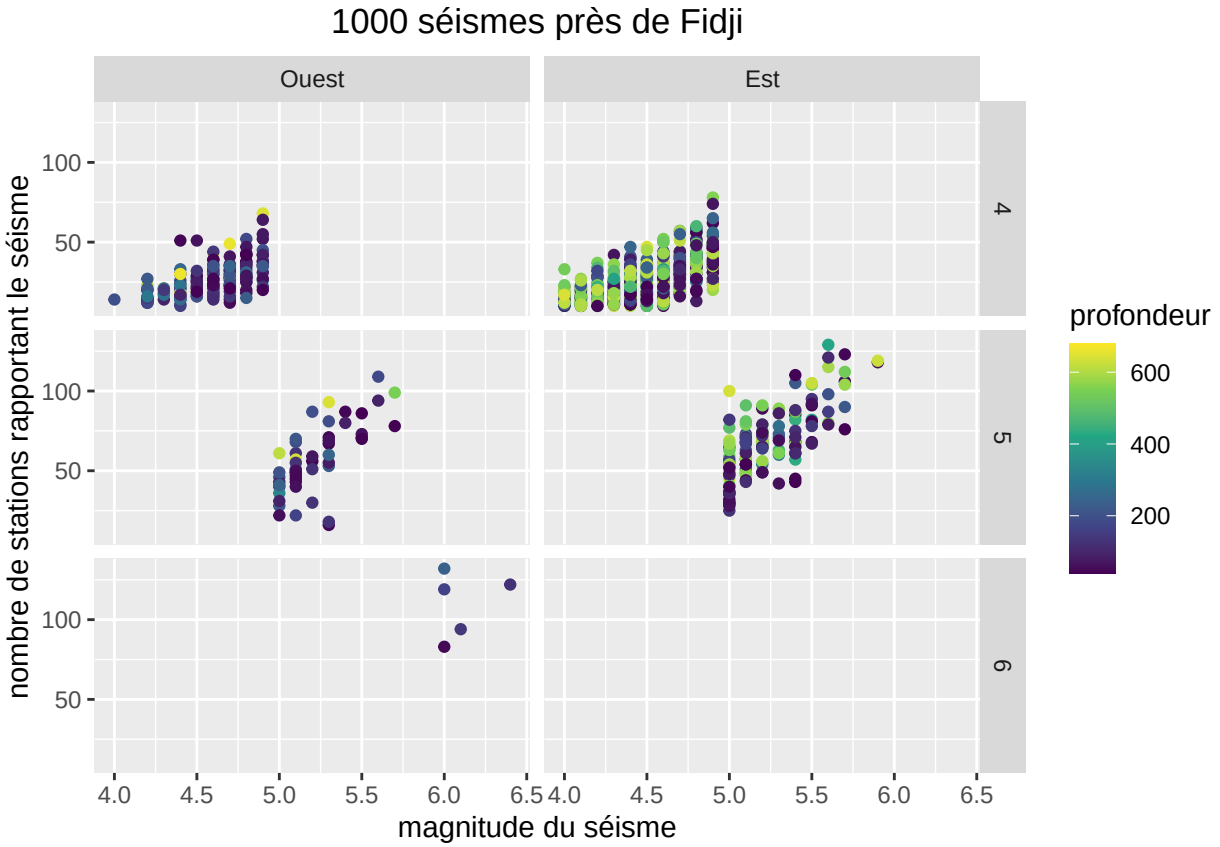
```
scatterplot +
  facet_wrap(facets = ~ region)   # permet de créer un sous-graphique par niveau de la variable region
```



- Grilles de graphiques selon les niveaux de deux variables catégoriques

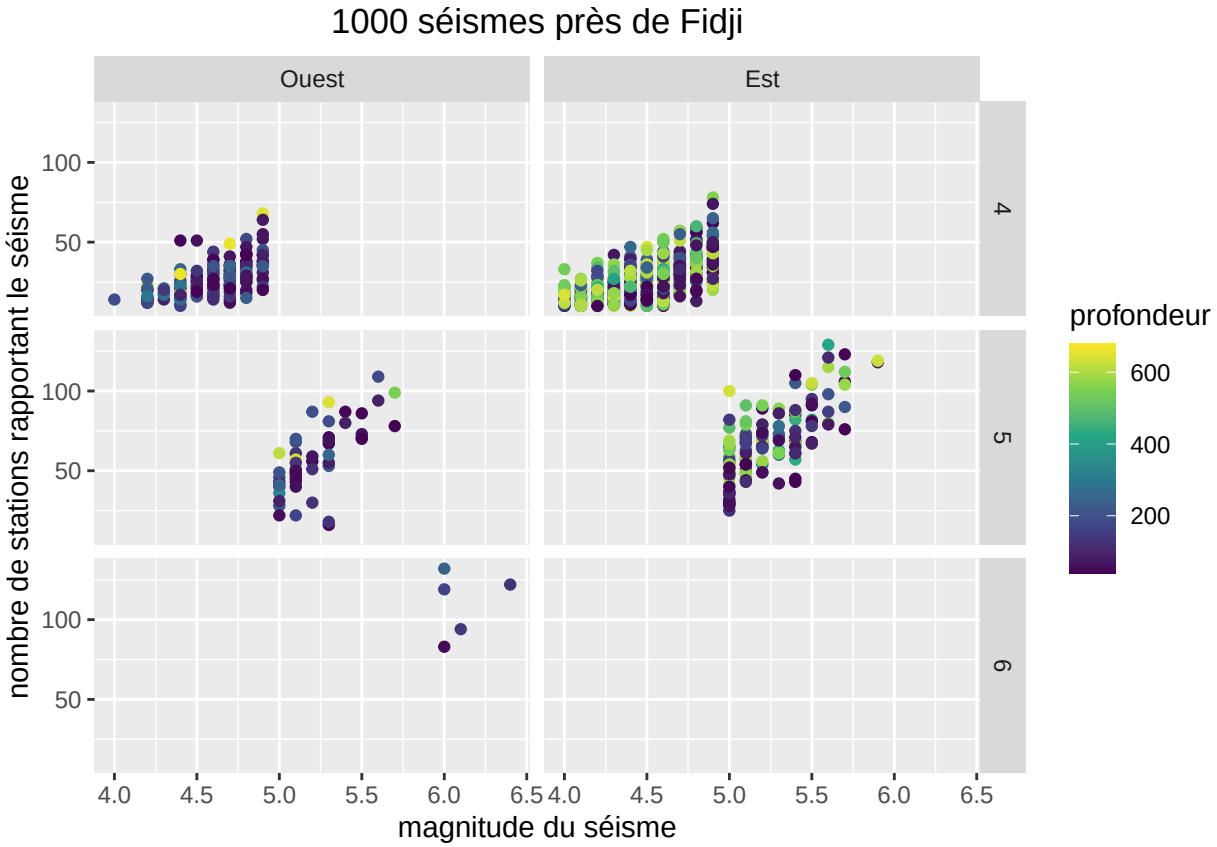
Nous pourrions même facilement ajouter une cinquième variable en créant une grille à deux dimensions de sous-graphiques côte à côte.

```
scatterplot +
  facet_grid(mag_catego ~ region)  # ou facet_grid(rows = vars(mag_catego), cols = vars(region))
```



Remarque : Si une variable qui n'a pas été stockée sous forme de facteur doit être traitée comme une variable catégorique dans un graphique ggplot, il suffit d'encadrer son nom d'un appel à la fonction factor directement dans le code de création du graphique, comme dans l'exemple suivant.

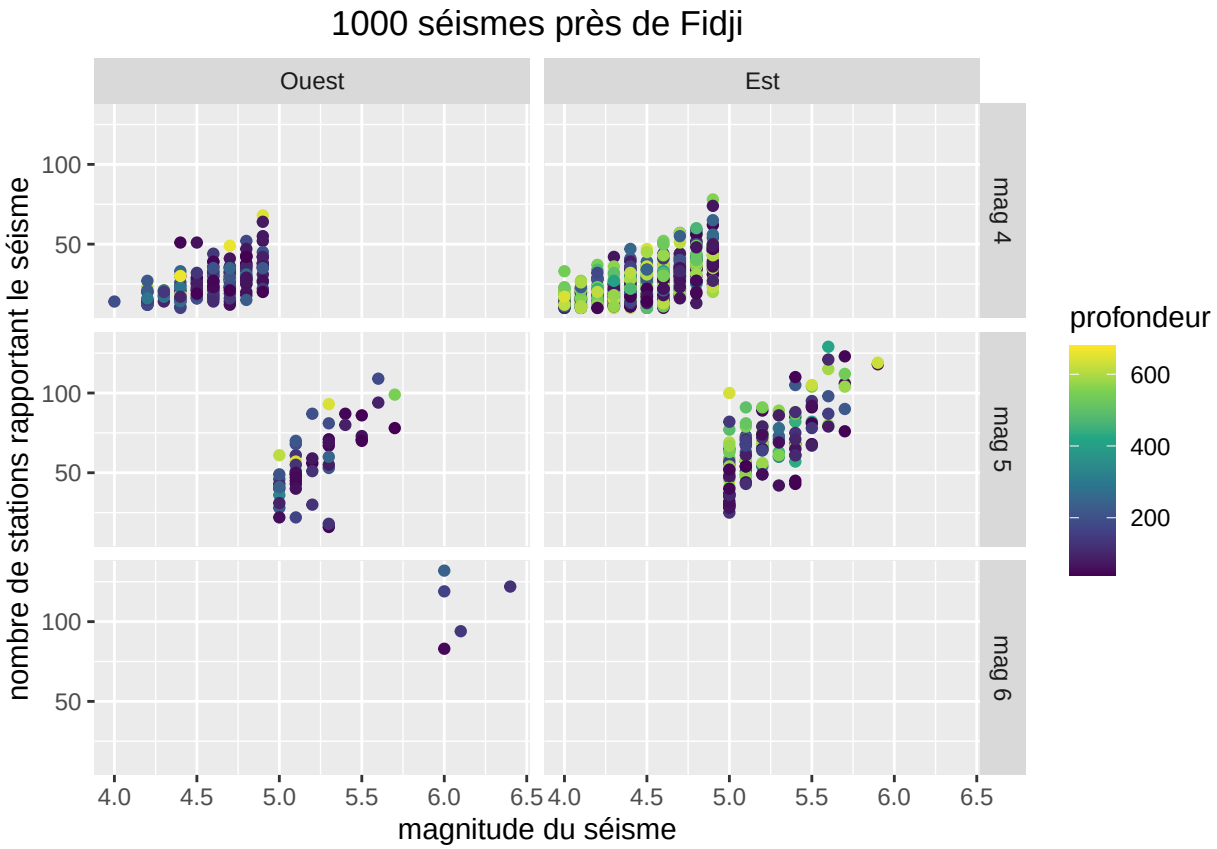
```
scatterplot +
  facet_grid(mag_catego ~ factor(region))
```



- Modification des libellés des niveaux d'une variable catégorique.

Les libellés des niveaux d'un facteur peuvent être modifiés ainsi.

```
# Création d'un vecteur faisant office de « lookup table », contenant l'association entre
# - les niveaux d'un facteur (nom des éléments dans le vecteur) et
# - leurs étiquettes à afficher dans le graphique (éléments du vecteur)
mag_catego_labels <- c(
  "4" = "mag 4",
  "5" = "mag 5",
  "6" = "mag 6"
)
# Production du graphique
scatterplot +
  facet_grid(
    mag_catego ~ region,
    labeller = labeller(mag_catego = mag_catego_labels)
  )
```



3.2 Exemples de graphiques de différents types

Voici quelques exemples faisant intervenir divers types de graphiques, pour illustrer des possibilités de ggplot2.

Dans la suite du travail nous allons utiliser non seulement la base quakes utiliser precedemment mais aussi la base issu du recensement de la population de 2018 en France 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

```
library(ggplot2)
library(questionr)
```

```
##
## Attachement du package : 'questionr'
```

```
## L'objet suivant est masqué depuis 'package:psych':
##
##     describe
```

```
## L'objet suivant est masqué depuis 'package:lessR':
##
##     prop
```

```
data(rp2018)

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

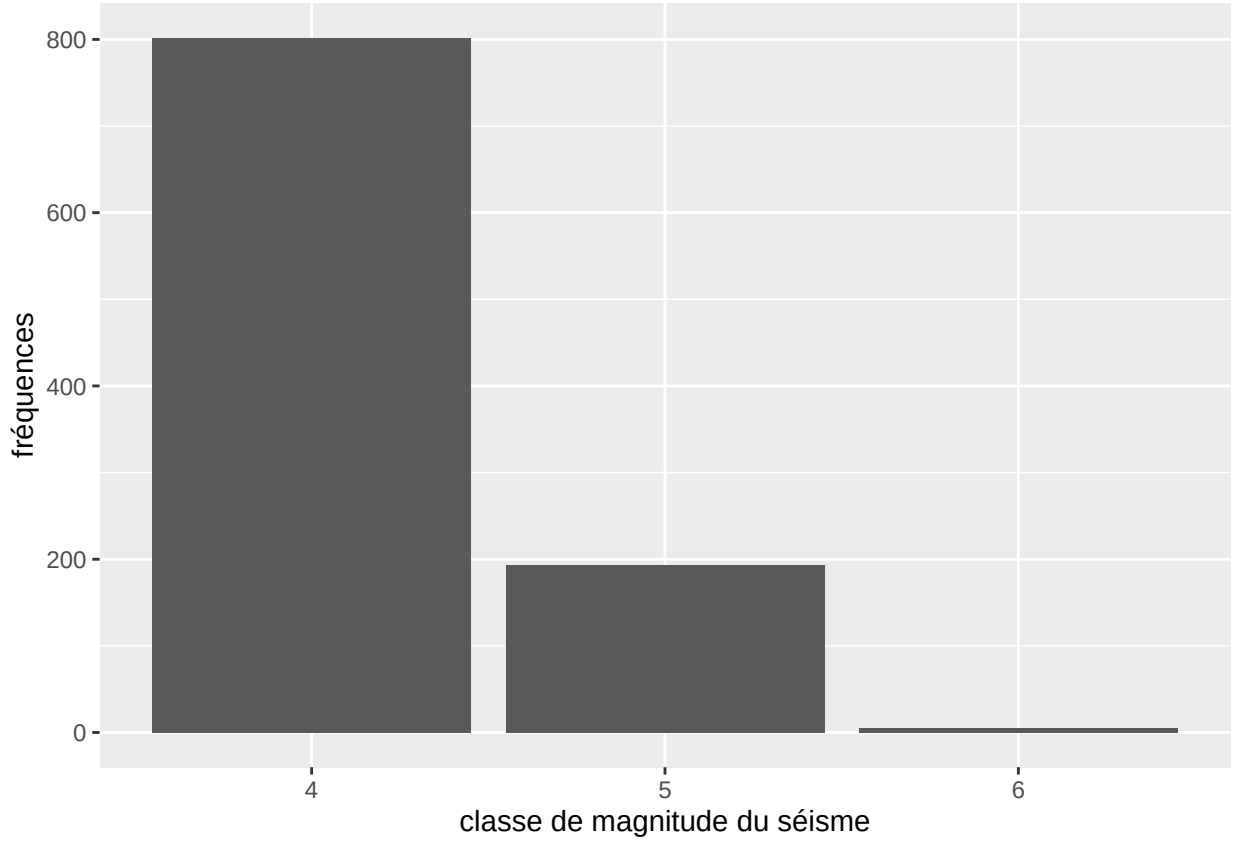
```
)
```

3.2.1 Diagramme en barres: fonctions geom-bar et geom-col

La fonction *geom-bar()* permet de produire un graphique en barre (barplot). On lui passe en x la variable qualitative dont on souhaite représenter l'effectif de chaque modalité.

En effet,la fonction *geom-bar()* calcule les fréquences des niveaux de facteurs et produit des diagrammes à barres. Avec cette fonction, pas besoin d'utiliser *table()* ou une aute fonction similaire pour calculer les fréquences.

```
ggplot(data = quakes) +
  geom_bar(mapping = aes(x = mag_catego)) +
  labs(
    x = "classe de magnitude du séisme",
    y = "fréquences"
  )
```

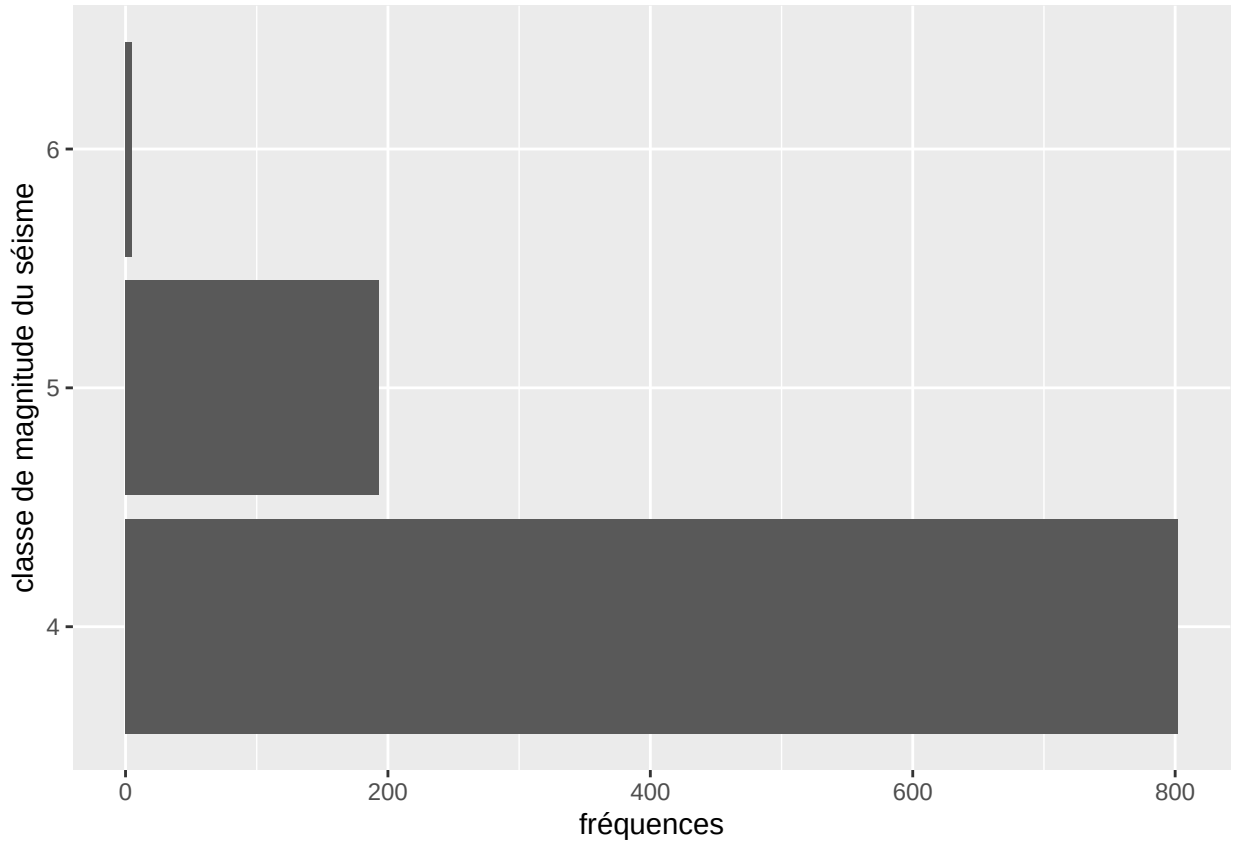


Si nous avons déjà en main les fréquences, il faut utiliser la fonction geom_bar avec l'argument stat = "identity", ou encore la fonction geom_col, comme suit.

```
quakes_mag <- as.data.frame(xtabs(~ mag_catego, data = quakes))
quakes_mag
```

```
##   mag_catego Freq
## 1          4  802
## 2          5  193
## 3          6    5
```

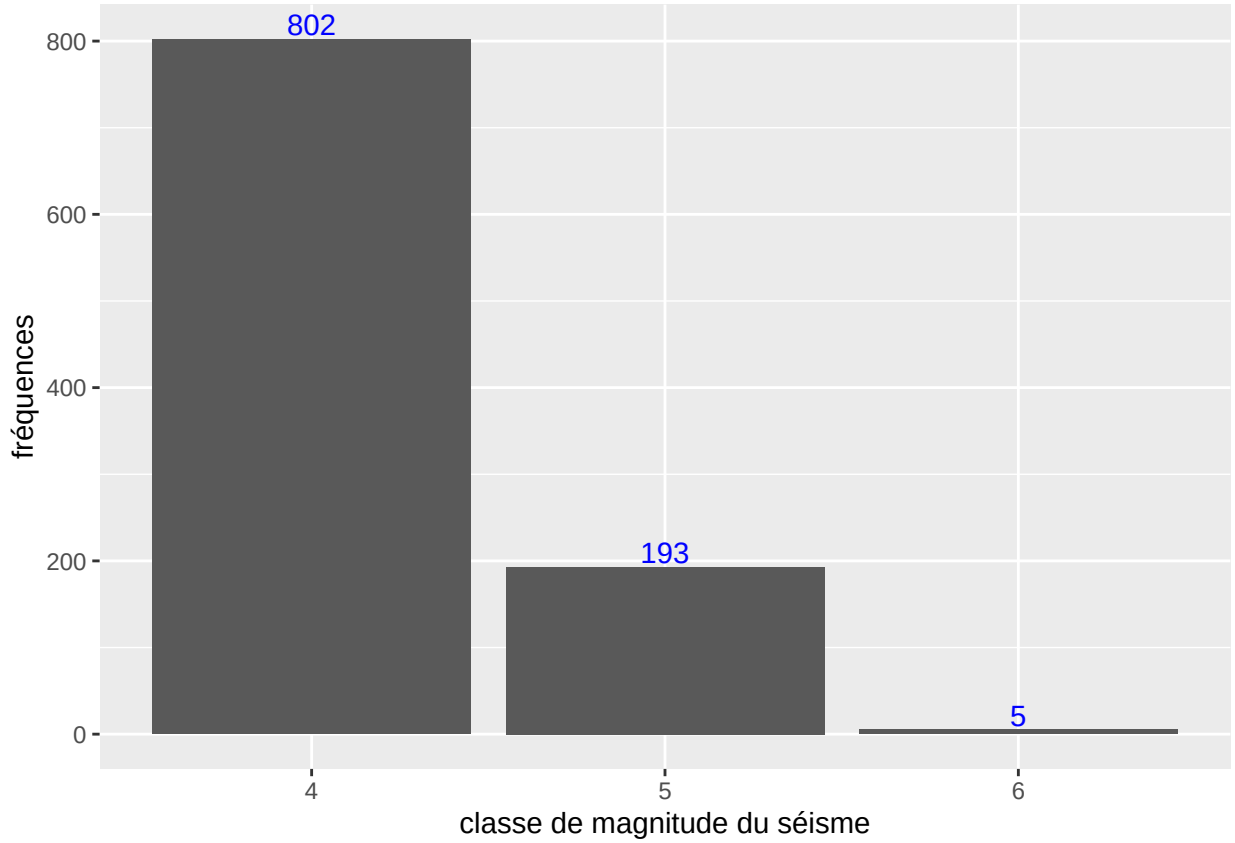
```r
ggplot(data = quakes_mag) +
  geom_col(mapping = aes(x = mag_catego, y = Freq)) +  # aesthetic y ajoutée
  coord_flip() +                                       # permet d'inverser les axes
  labs(
    x = "classe de magnitude du séisme",
    y = "fréquences"
  )
```



Un appel à la fonction coord_flip inverse les deux axes et permet d'obtenir des barres horizontales plutôt que verticales.

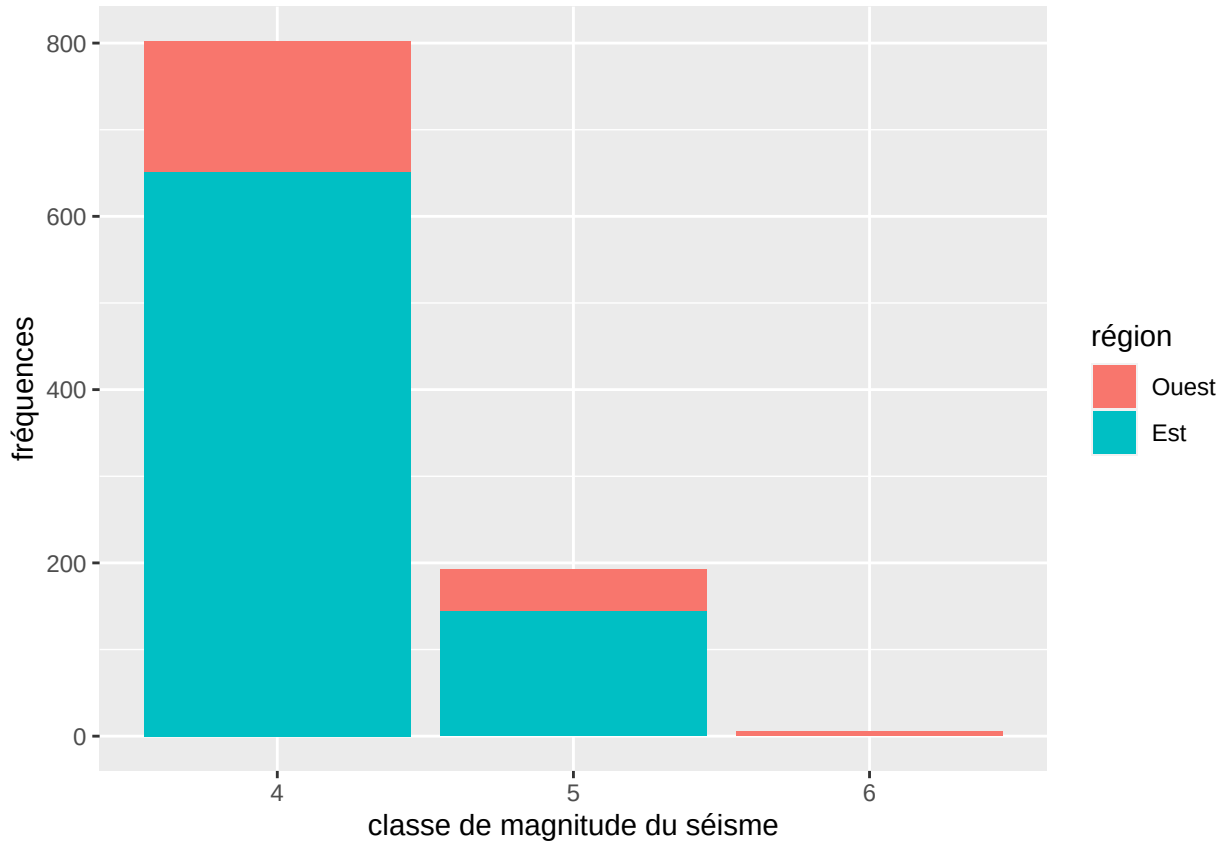
La fonction geom_text permet d'ajouter des annotations textuelles dans le graphique.Nous allons approfondir cette notion plus tard.

```r
ggplot(data = quakes, mapping = aes(x = mag_catego)) + # aesthetics communs ici
  geom_bar() +
  geom_text(
    mapping = aes(label = after_stat(count)),          # texte ajouté = fréquences
    colour = "blue",
    stat = "count",                                    # calcul des fréquences demandé ici
    vjust = -0.2                                       # ajustement vertical
  ) +
  labs(
    x = "classe de magnitude du séisme",
    y = "fréquences"
  )
```



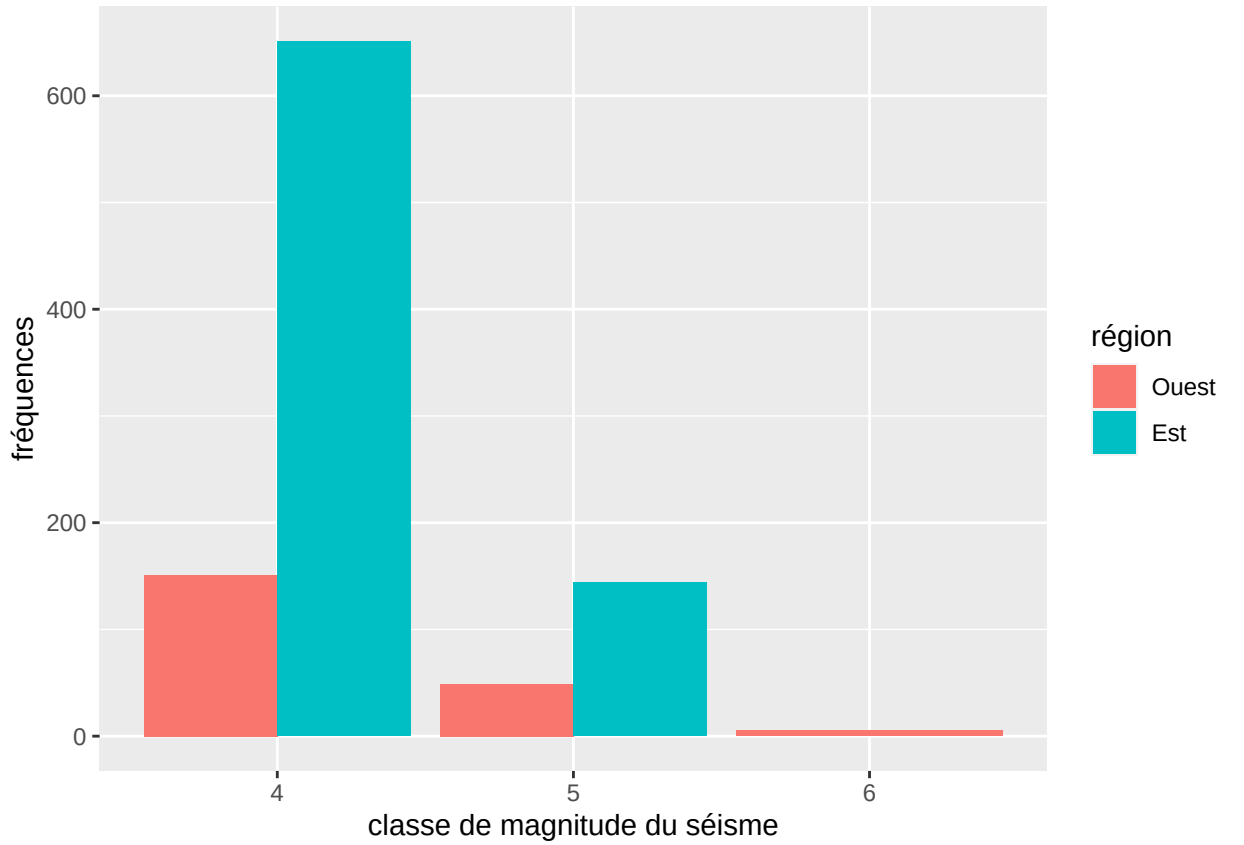
Pour représenter des fréquences croisées, on peut ajouter une variable catégorique au graphique via l'argument fill.

```
ggplot(data = quakes) +
  geom_bar(mapping = aes(x = mag_catego, fill = region)) +  # aesthetic fill ajoutée
  labs(
    x = "classe de magnitude du séisme",
    y = "fréquences",
    fill = "région"
  )
```



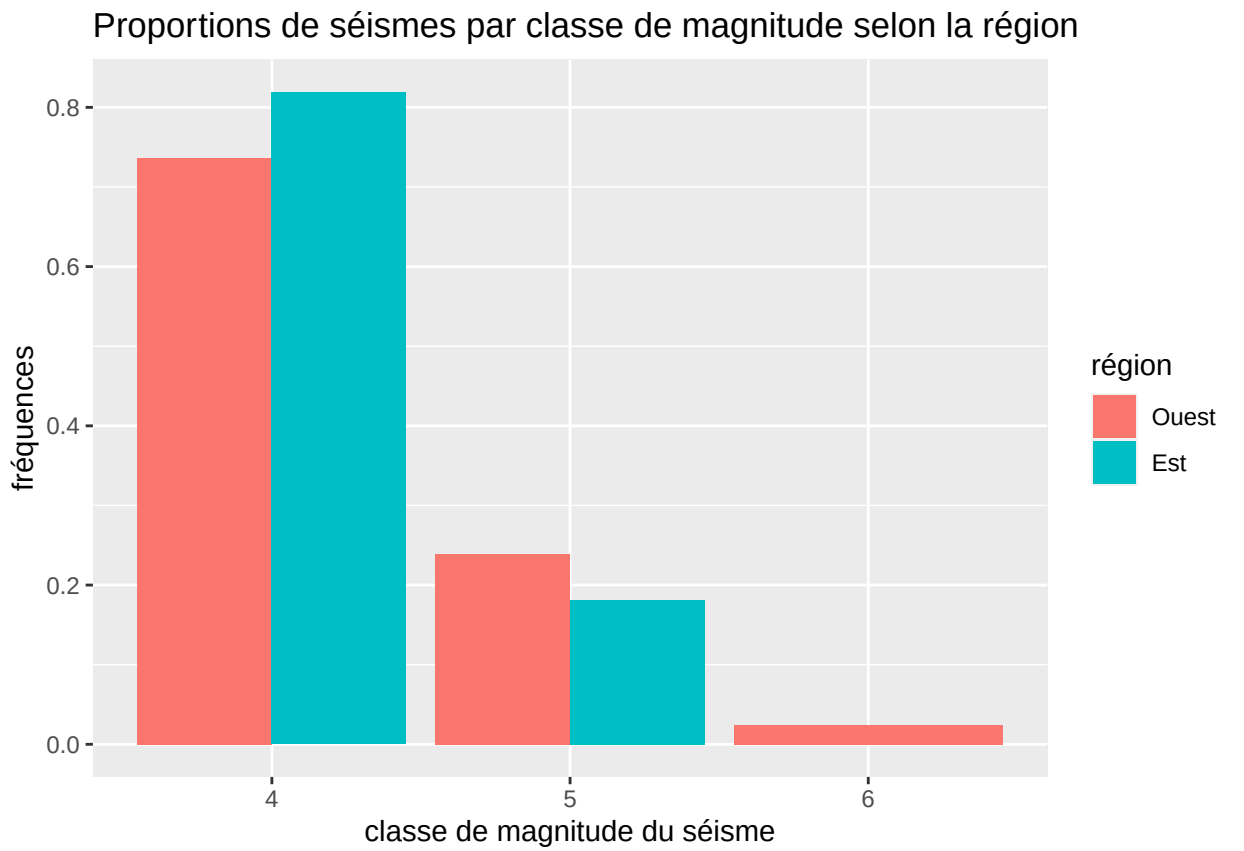
Pour avoir des bâtons groupés plutôt qu'empilés, il faut modifier la valeur de l'argument position.

```
ggplot(data = quakes) +
  geom_bar(
    mapping = aes(x = mag_catego, fill = region),
    position = "dodge"                              # position des bâtons modifiée
  ) +
  labs(
    x = "classe de magnitude du séisme",
    y = "fréquences",
    fill = "région"
  )
```



Afin de présenter des fréquences relatives plutôt que brutes.

```
ggplot(data = quakes) +
  geom_bar(
    mapping = aes(
      x = mag_catego,
      y = after_stat(prop),  # permet le calcul de fréquences relatives (proportions)
      group = region,        # pour avoir les fréq. relatives de mag conditionnelles à region
      fill = region),
    position = "dodge"
  ) +
  labs(
    title = "Proportions de séismes par classe de magnitude selon la région",
    x = "classe de magnitude du séisme",
    y = "fréquences",
    fill = "région"
  )
```

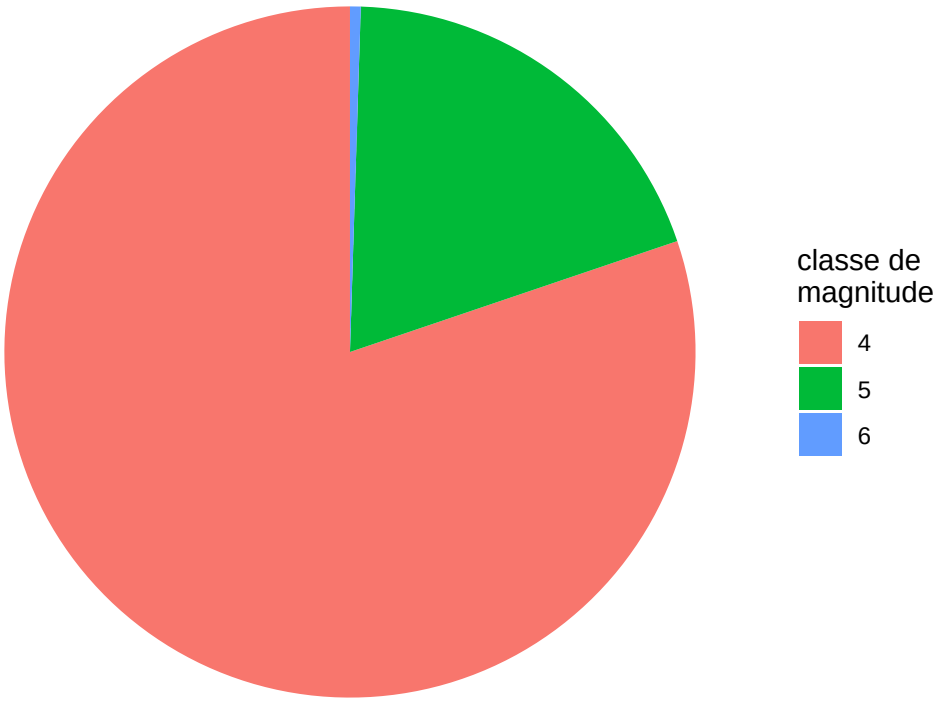


3.2.2 Diagramme en secteurs: coordonnées polaires avec coord-polar

Les auteurs de ggplot2 n'offrent pas de fonction conviviale pour la création de diagrammes en secteurs, probablement parce qu'ils ne recommandent pas leur utilisation. Malgré tout, tenter de tracer un diagramme en secteurs avec ggplot2 aide à comprendre davantage les possibilités du package.

Pour produire un diagramme en secteurs avec ggplot2, il faut d'abord produire un diagramme à barres empilées, puis demander l'utilisation d'un système de coordonnées polaires par un appel à la fonction coord_polar.

```
ggplot(data = quakes) +
  geom_bar(mapping = aes(x = 1, fill = mag_catego)) +
  coord_polar(theta = "y") +                              # coordonnées polaires
  theme_void() +                                          # thème vide
  labs(fill = "classe de\nmagnitude")
```



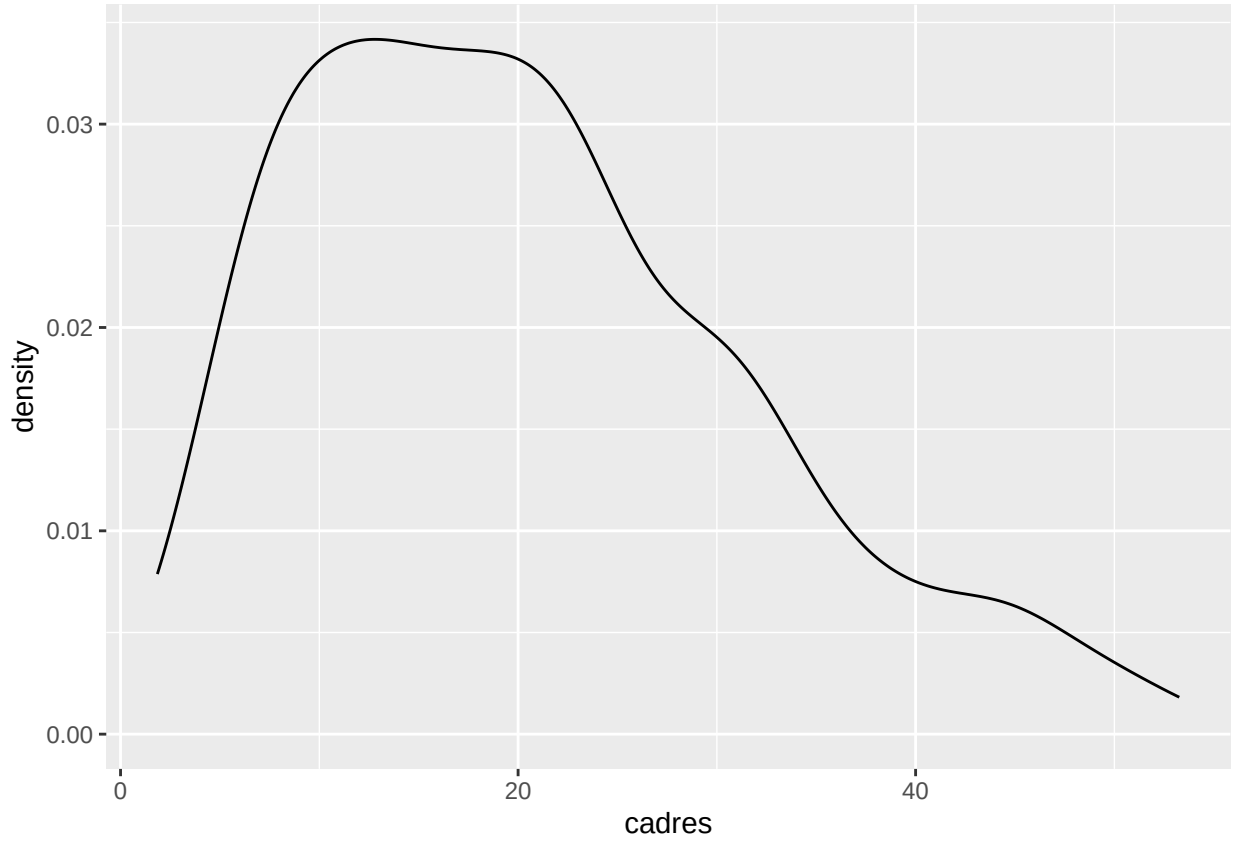
3.2.3 Courbes de densité :fonction geom-density

- Courbes de densité à noyau unique

geom_density permet d'afficher l'estimation de densité d'une variable numérique. Son usage est similaire à celui de geom_histogram.

Ainsi, si on veut afficher la densité de la répartition de la part des cadres dans les communes de notre jeu de données :

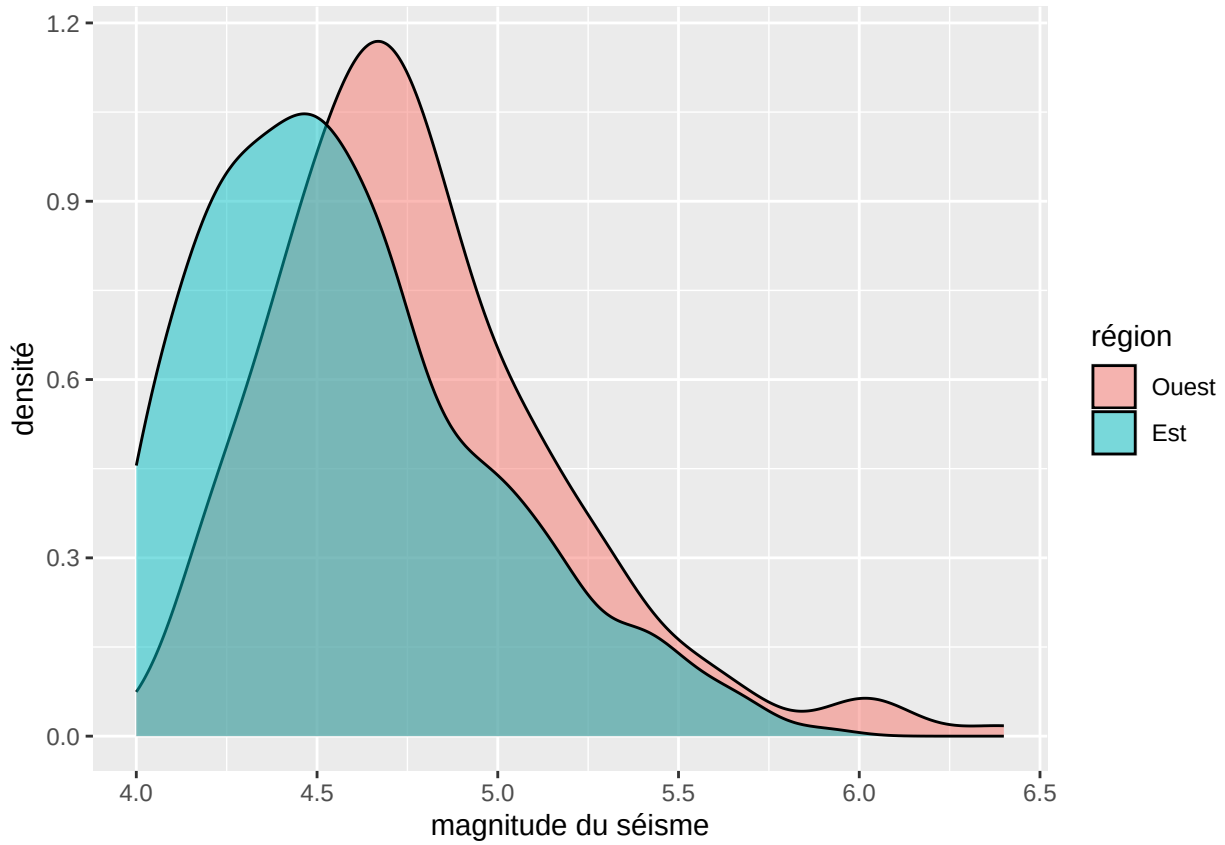
```
ggplot(rp) + geom_density(aes(x = cadres))
```



- Courbes de densité à noyau superposées

Il est facile avec ggplot2 de superposer des histogrammes ou des courbes de densités à noyau (comme ci-dessous).

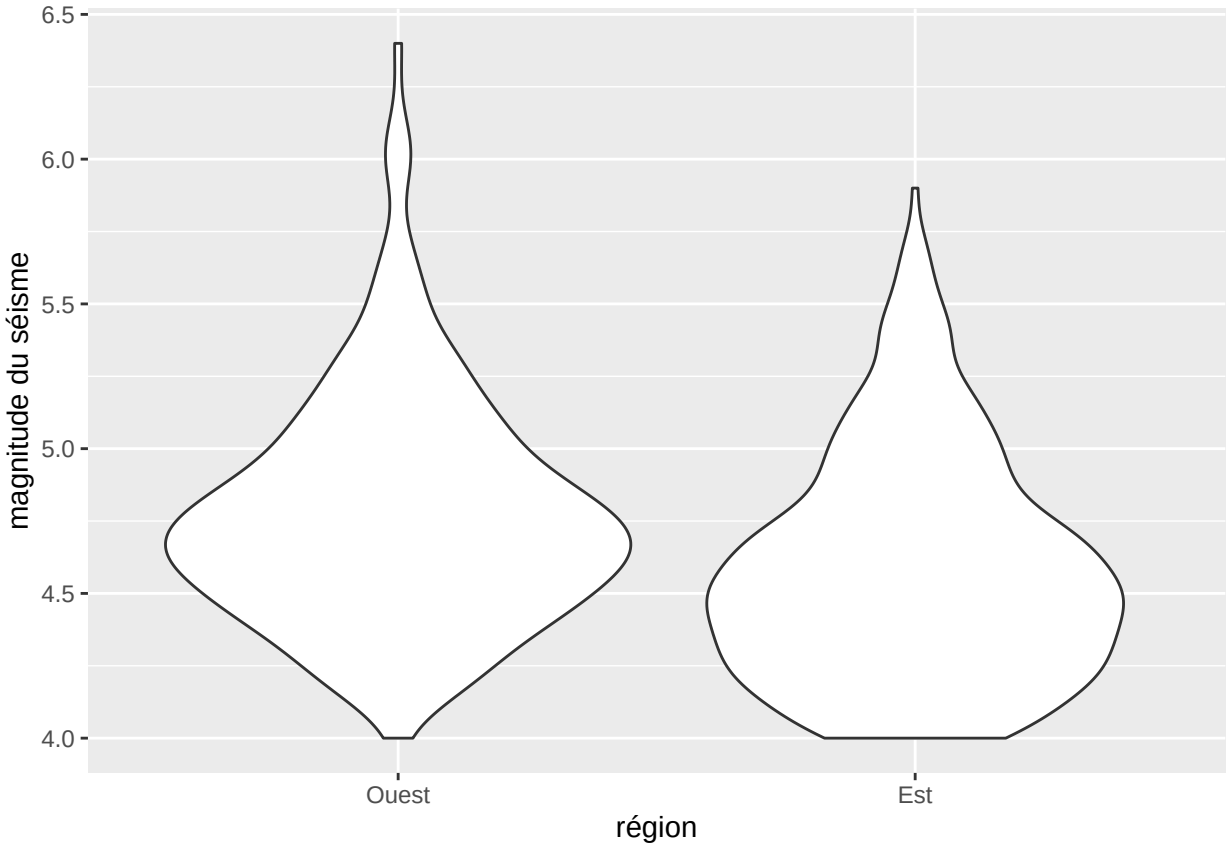
```
ggplot(data = quakes) +
  geom_density(
    mapping = aes(x = mag, fill = region),
    alpha = 0.5  # niveau d'opacité (0 = transparent, 1 = complètement opaque)
  ) +
  labs(
    x = "magnitude du séisme",
    y = "densité",
    fill = "région"
  )
```



3.2.4 Diagrammes en violons : fonction geom-violin

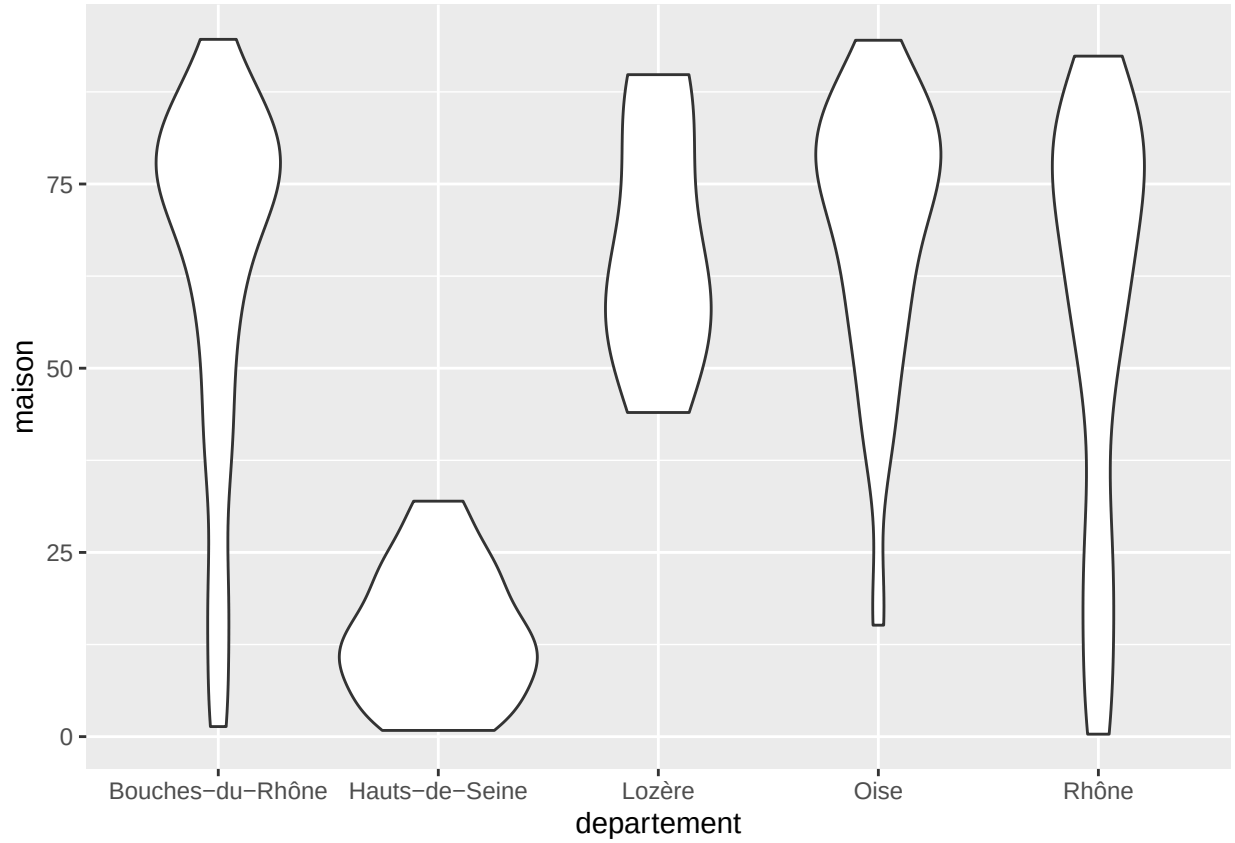
Un autre outil, dérivé des courbes de densités à noyau, permettant d'explorer l'association potentielle entre une variable numérique et d'une variable catégorique est le diagramme en violon (violin plot).

```
violinplots <- ggplot(data = quakes) +
  geom_violin(mapping = aes(x = region, y = mag)) +
  labs(
    x = "région",
    y = "magnitude du séisme"
  )

violinplots
```



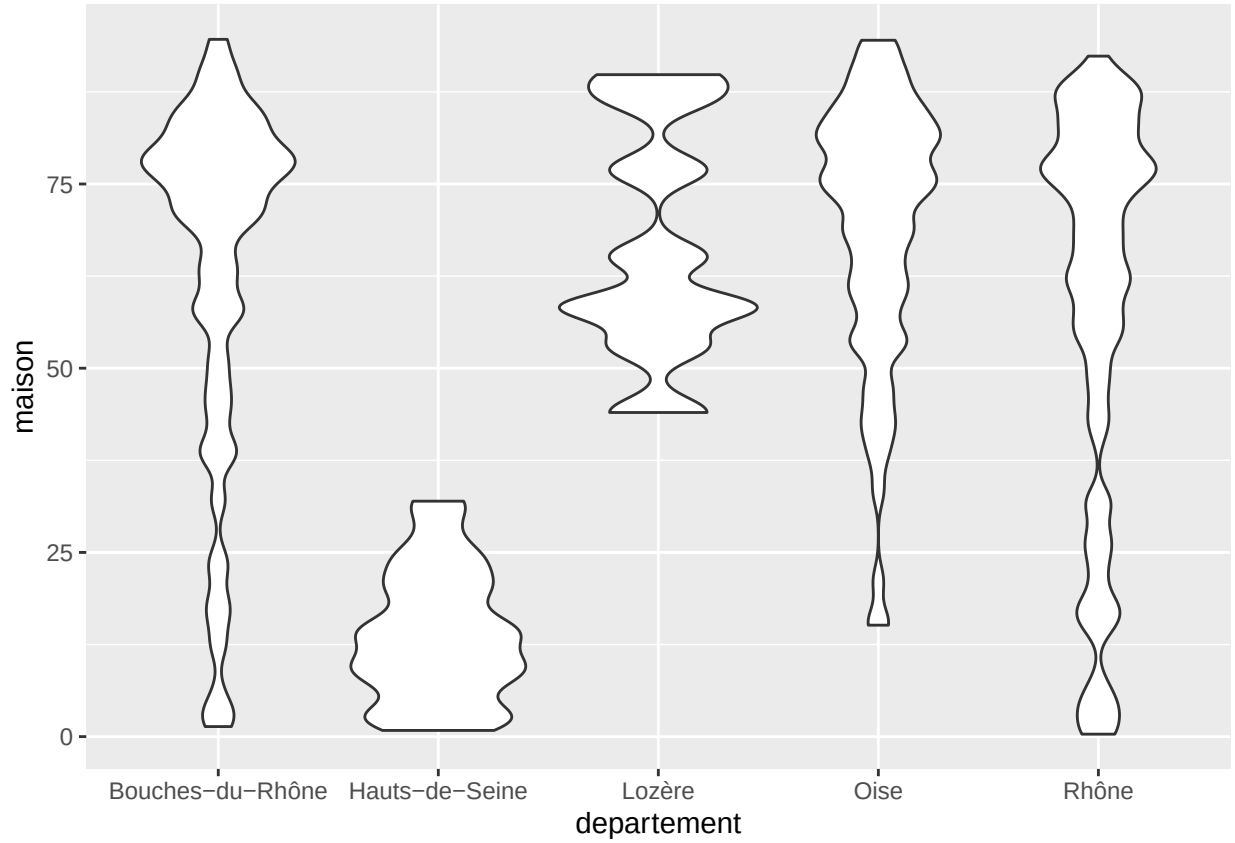
Il faut noter aussi que geom_violin est très semblable à geom_boxplot, mais utilise des graphes en violon à la place des boîtes à moustache.sommes toutes on peut dire que les graphiques geom-violon sont dérivés des courbes de densité et des boites à moustache.Le diagramme en violon est en fait un mélange entre les courbes de densité à noyau et les diagrammes en boîtes (boxplots).on peut alors supperposer plusieurs graphiques geom_violon comme les boxplots aussi.

```
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).

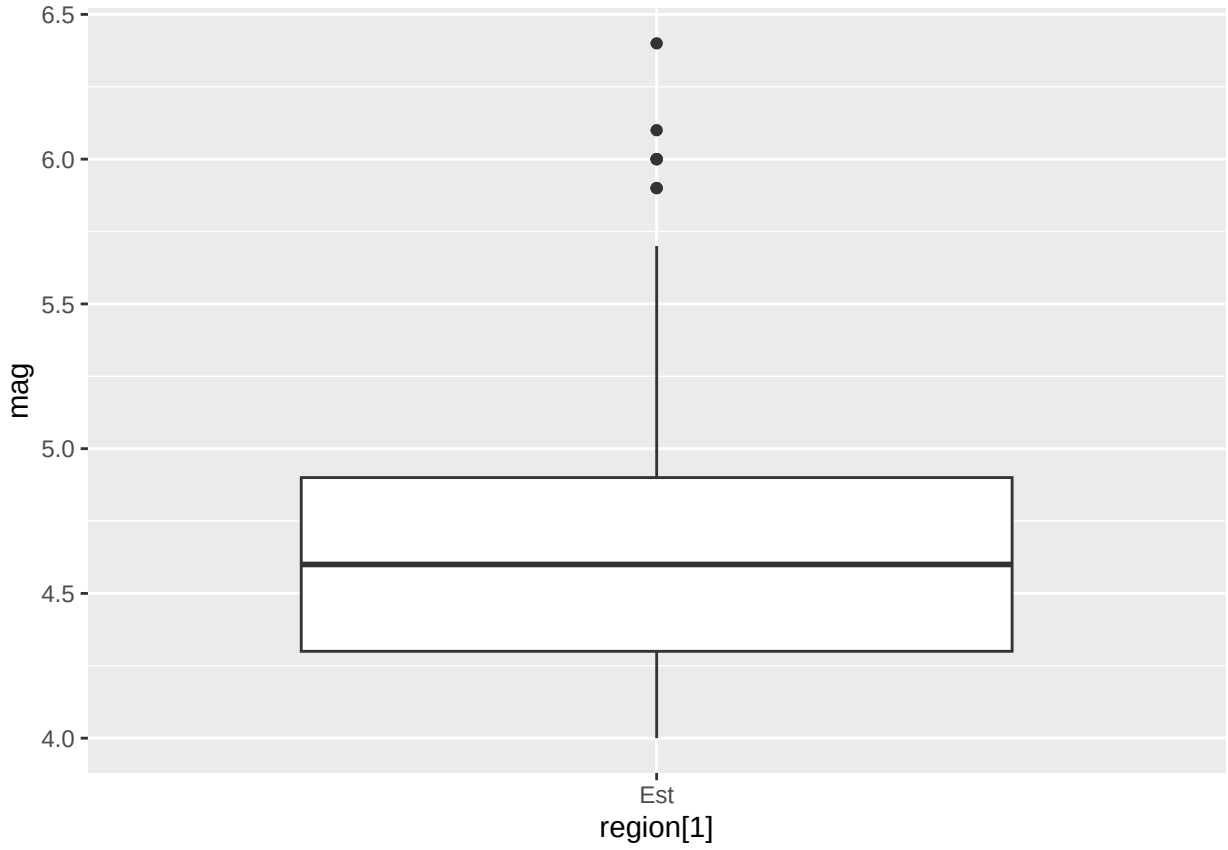
```
ggplot(rp) +
  geom_violin(
    aes(x = departement, y = maison),
    bw = 2
  )
```



3.2.5 Diagrammes en boîtes : fonction geom-boxplot

- diagramme en boite unique

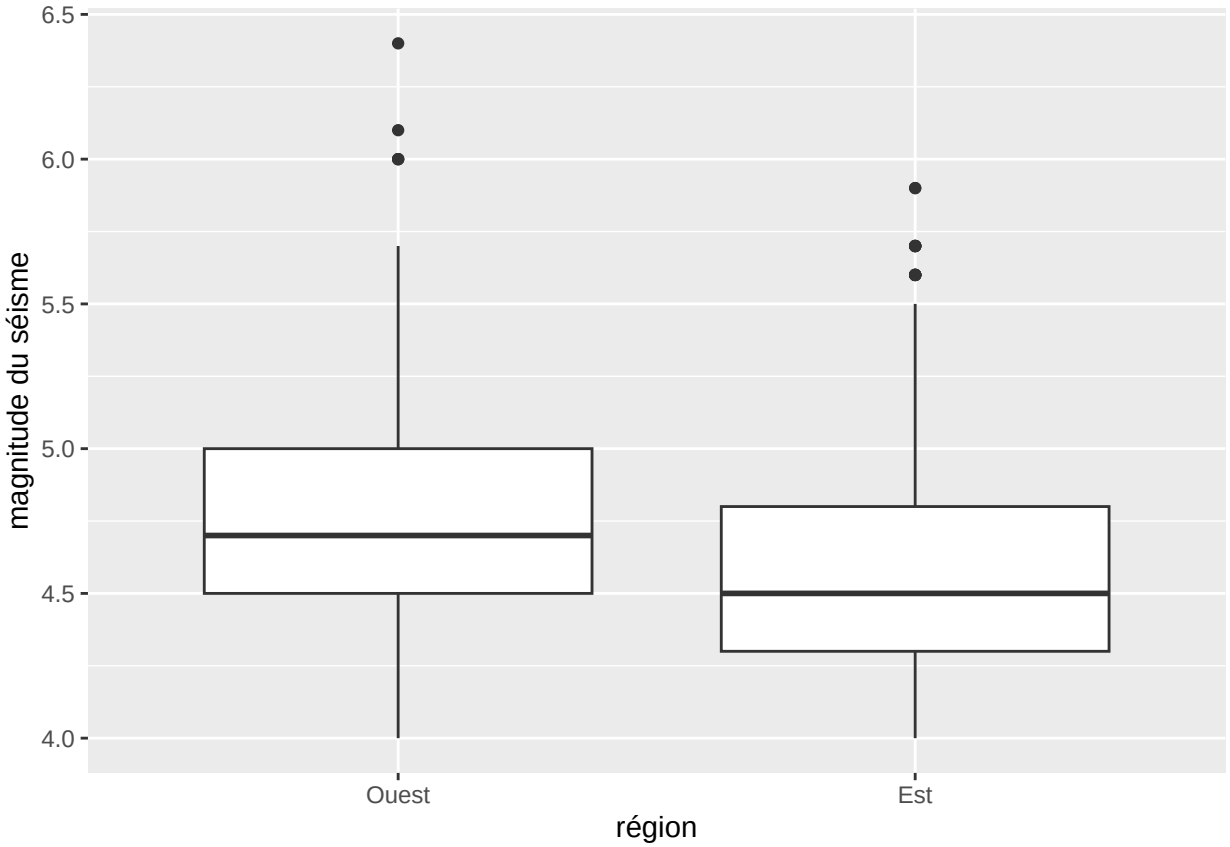
```
ggplot(quakes, aes(x = region[1], y =mag)) +
  geom_boxplot()
```



- diagramme en boite juxtaposés.

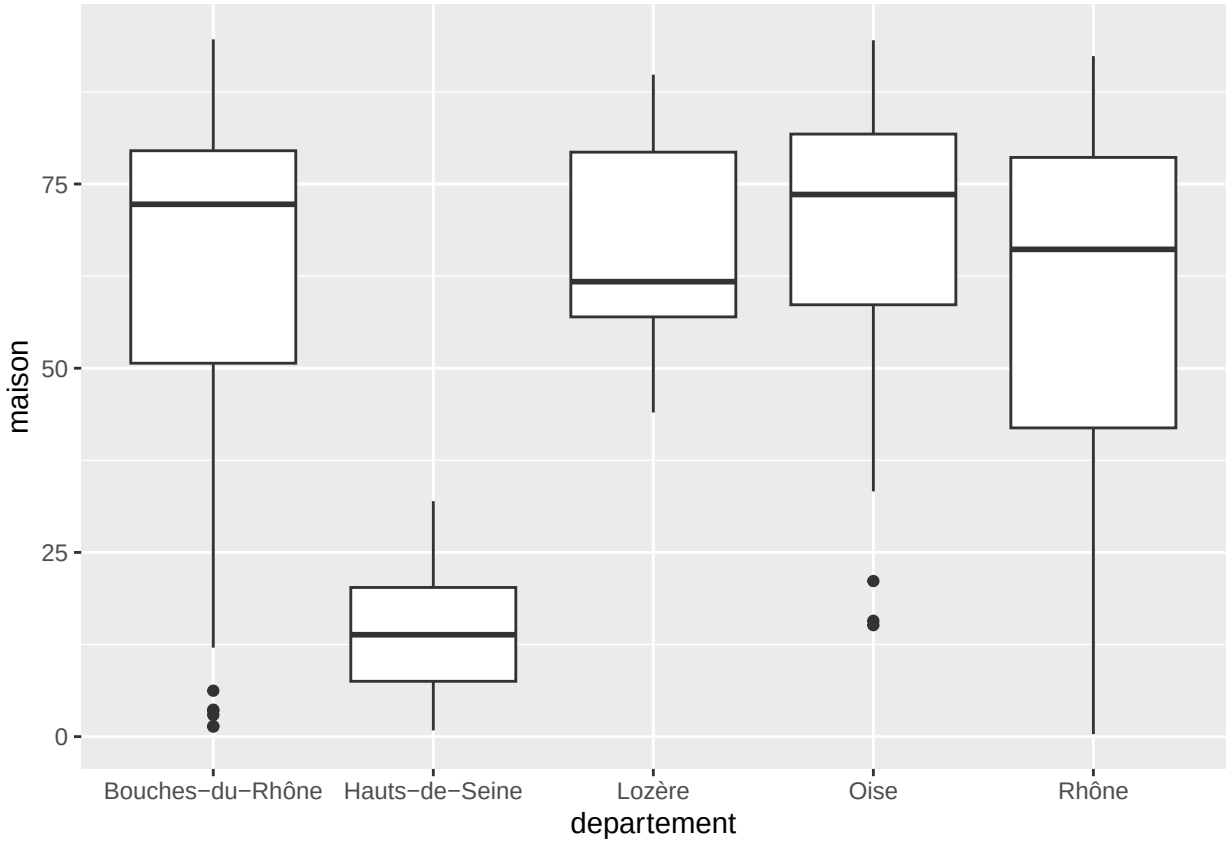
Il est possible de faire des boites à moustaches justaposé en fonction des besoins .Ici nous allons voir different reprentation .

```
boxplots <- ggplot(data = quakes) +
  geom_boxplot(mapping = aes(x = region, y = mag)) +
  labs(
    x = "région",
    y = "magnitude du séisme"
  )

boxplots
```



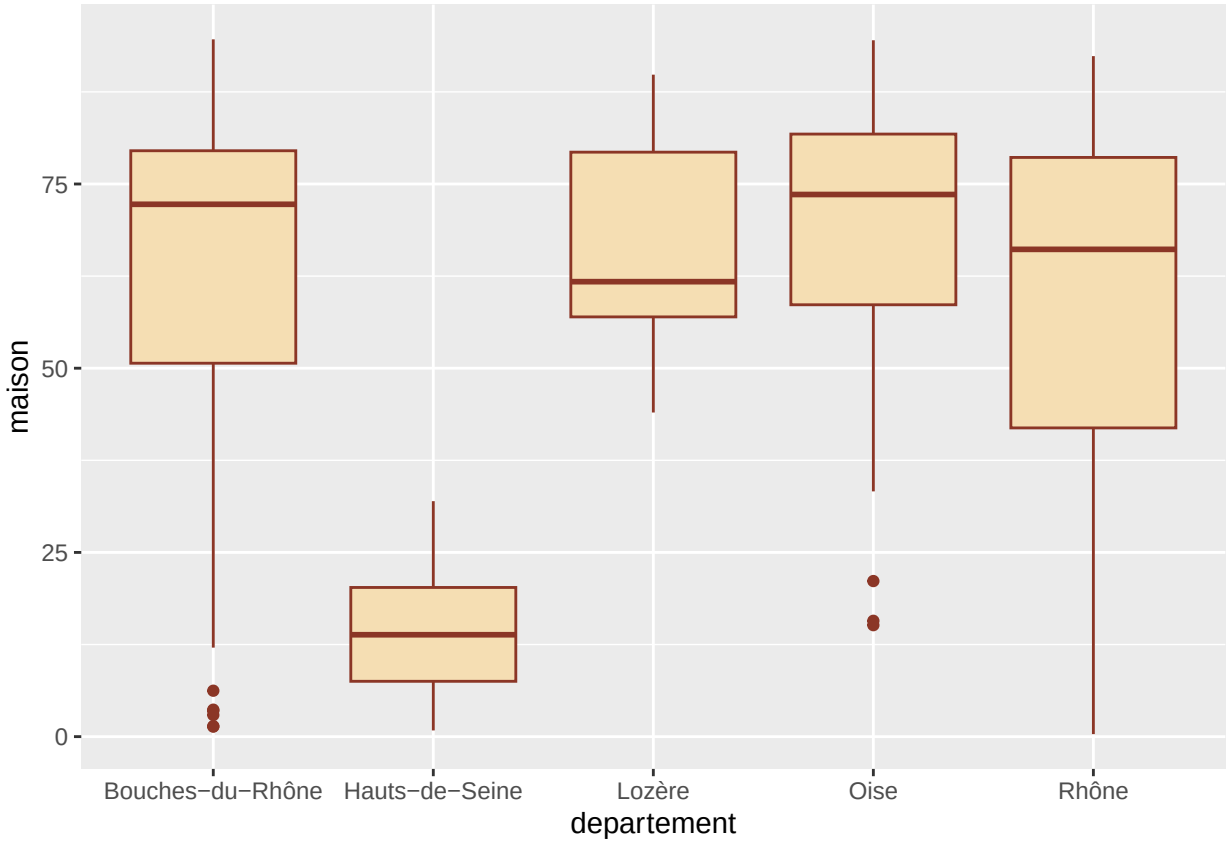
Cette fois on souhaite comparer plusieurs boites a moutache donc nous allons utiliser les données rp . Ainsi, si on veut comparer la répartition du pourcentage de maisons en fonction du département de la commune, on pourra faire :

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



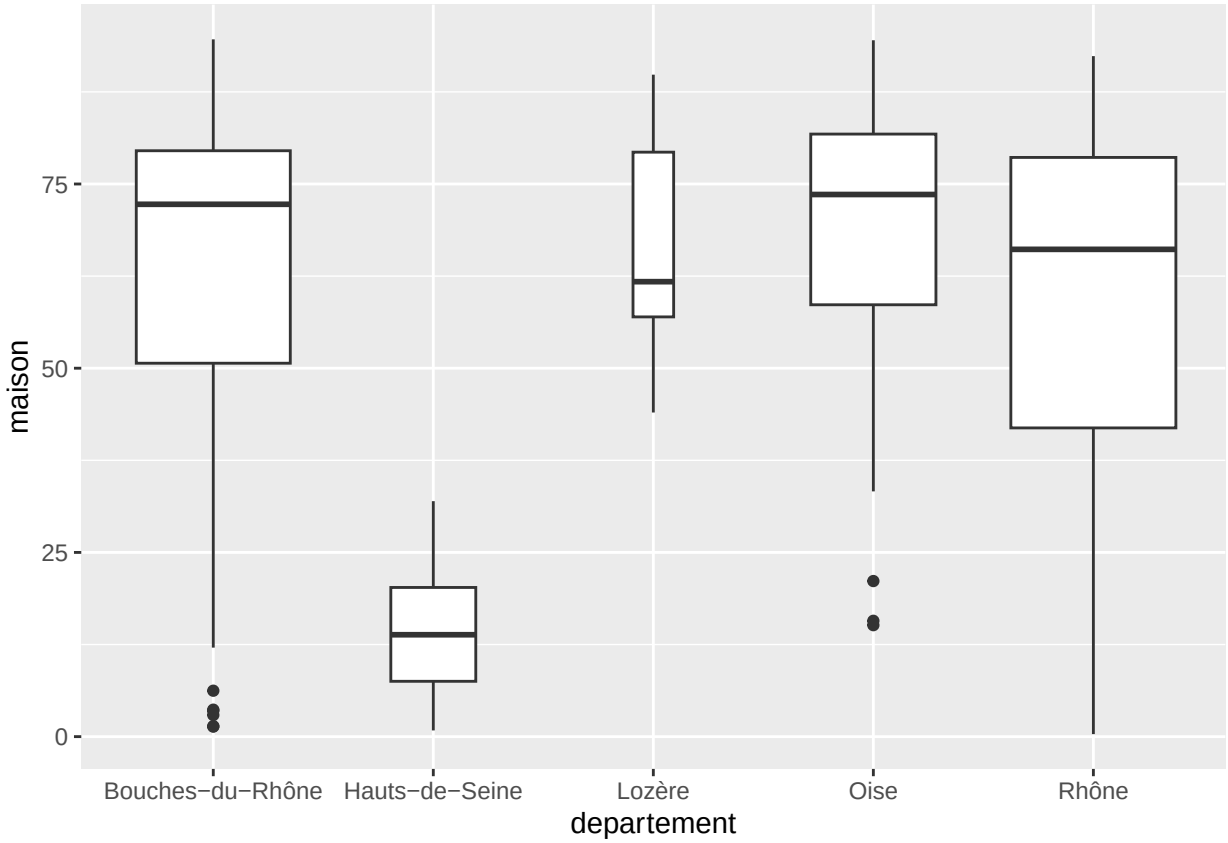
On peut personnaliser la présentation avec différents argument supplémentaires comme fill ou color

```
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)
```

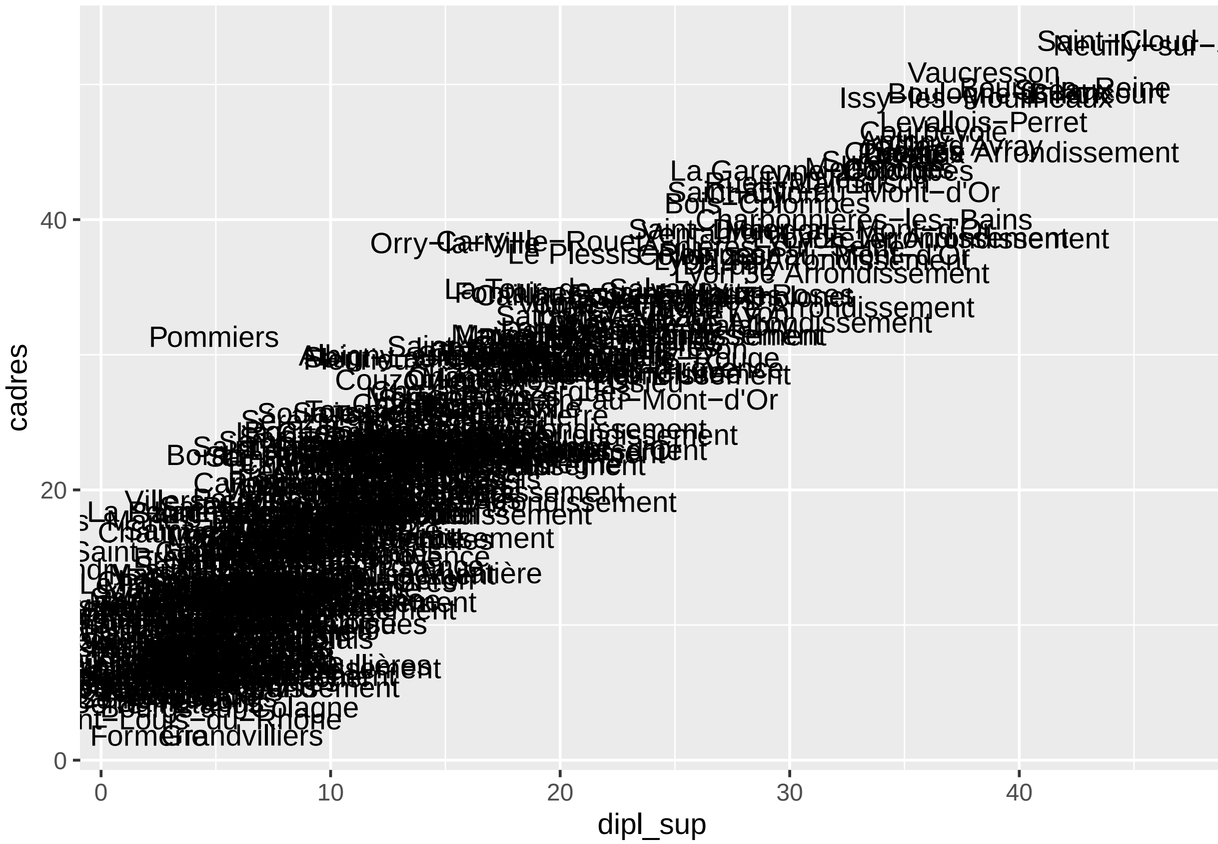


3.2.6 La fonction geom-text

geom_text permet d'afficher des étiquettes de texte. On doit lui fournir trois paramètres dans aes : x et y pour la position des étiquettes, et label pour leur texte.

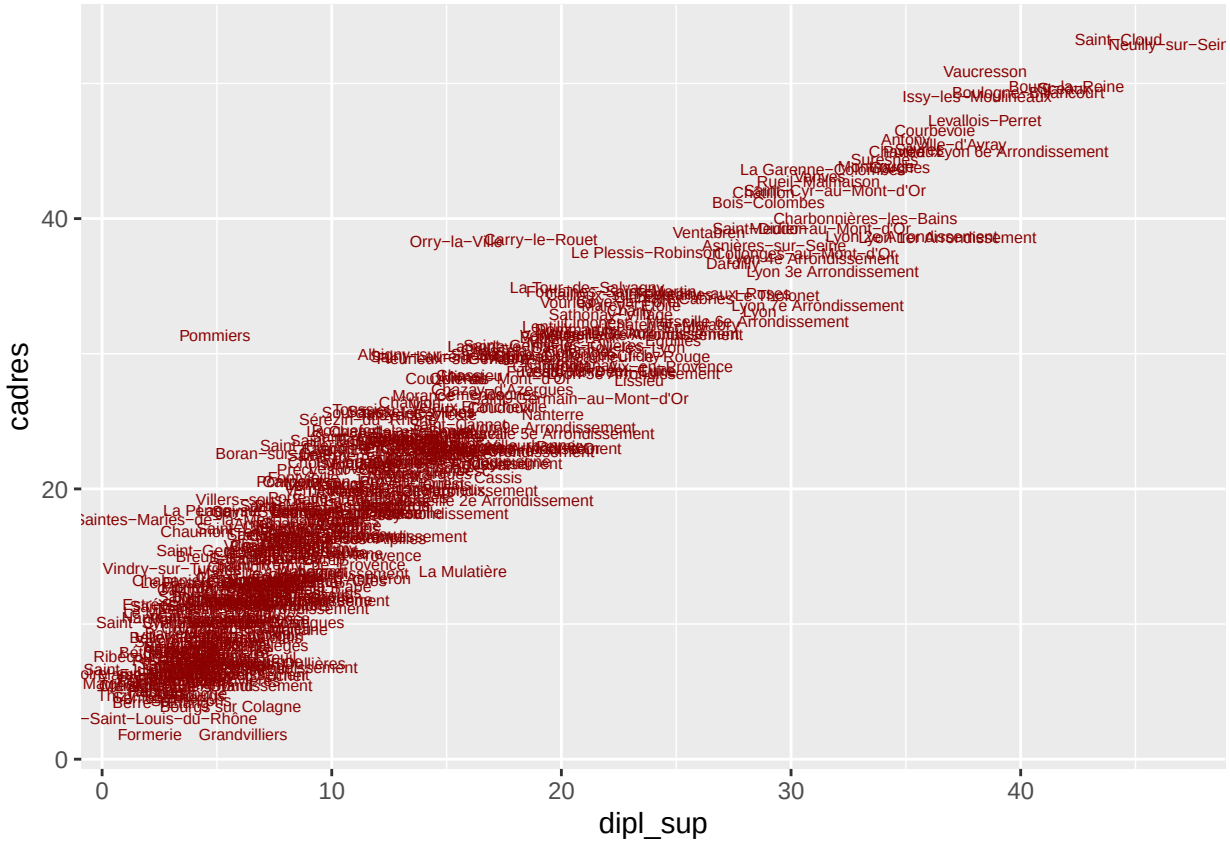
Par exemple, si on souhaite représenter le nuage croisant la part des diplômés du supérieur et la part de cadres, mais en affichant le nom de la commune (variable commune) plutôt qu'un simple point, on peut faire :

```
data(rp)
ggplot(rp) +
  geom_text(
    aes(x = dipl_sup, y = cadres, label = commune)
  )
```



On peut personnaliser l'apparence et la position du texte avec des arguments comme size, color, etc.

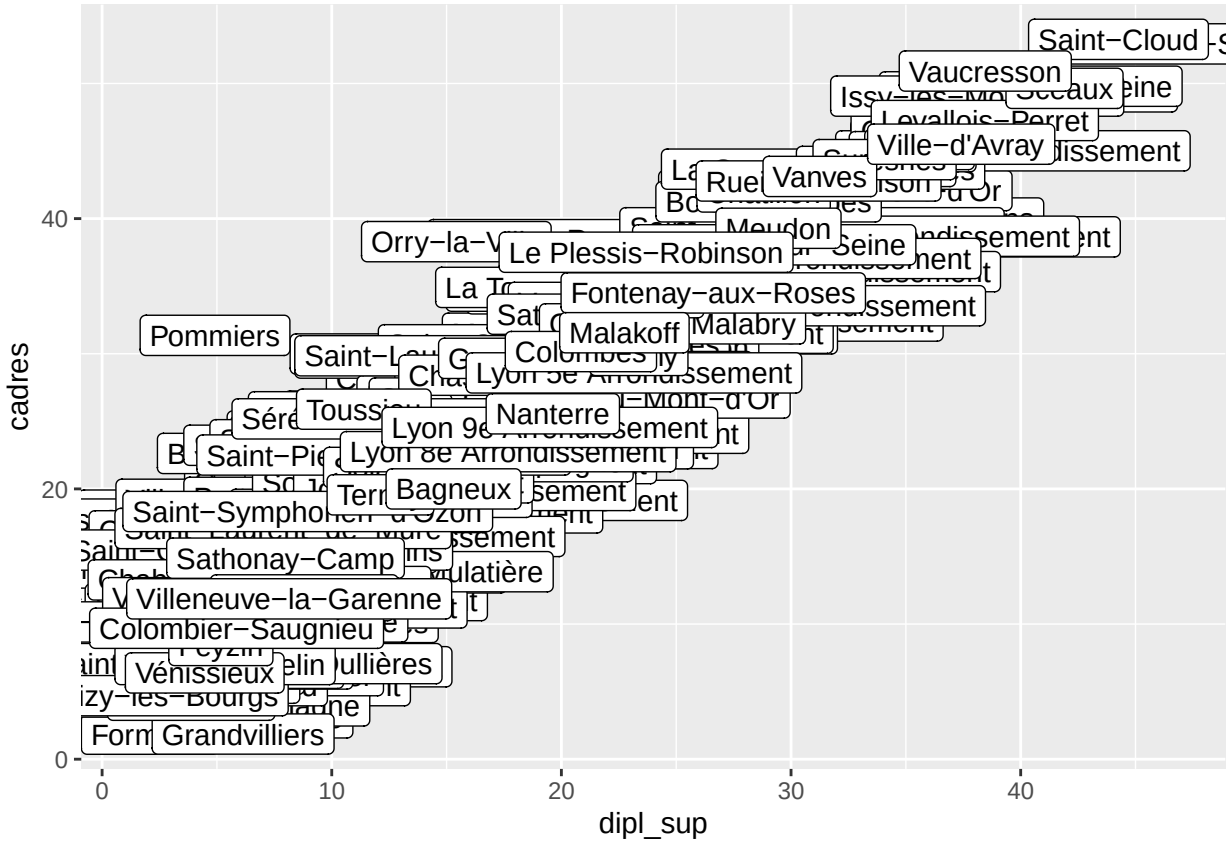
```
ggplot(rp) +
  geom_text(
    aes(x = dipl_sup, y = cadres, label = commune),
    color = "darkred", size = 2
  )
```



3.2.7 La fonction geom-label

geom_label est identique à geom_text, mais avec une présentation un peu différente.

```
library(ggplot2)
ggplot(rp) + geom_label(aes(x = dipl_sup, y = cadres, label = commune))
```



3.2.8 La fonction 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.

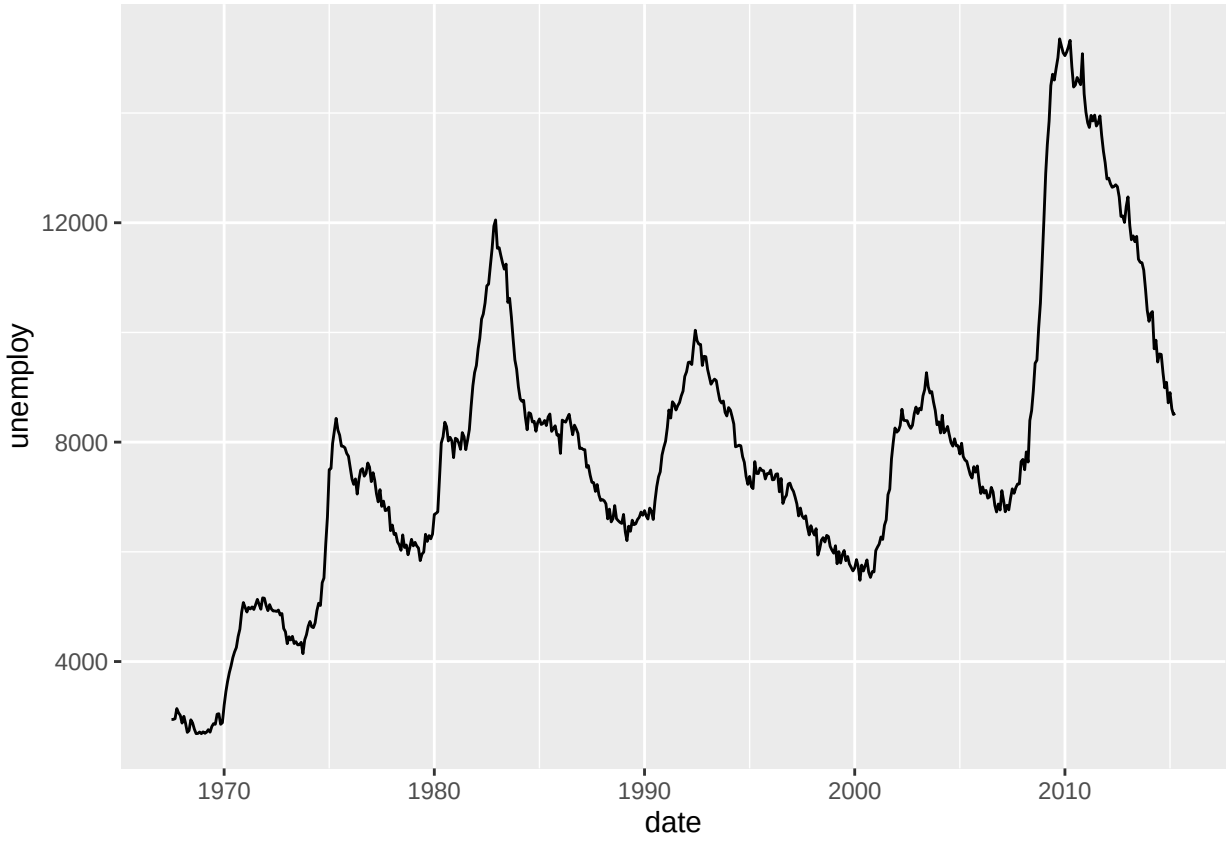
Comme il n'y a pas de données adaptées pour ce type de représentation dans notre jeu de données d'exemple, on va utiliser ici le jeu de données economics inclus dans ggplot2 et représenter l'évolution du taux de chômage aux États-Unis (variable unemploy) dans le temps (variable date) :

```
data("economics")
str(economics)
```

```
## spc_tbl_ [574 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ date    : Date[1:574], format: "1967-07-01" "1967-08-01" ...
##  $ pce     : num [1:574] 507 510 516 512 517 ...
##  $ pop     : num [1:574] 198712 198911 199113 199311 199498 ...
##  $ psavert : num [1:574] 12.6 12.6 11.9 12.9 12.8 11.8 11.7 12.3 11.7 12.3 ...
##  $ uempmed : num [1:574] 4.5 4.7 4.6 4.9 4.7 4.8 5.1 4.5 4.1 4.6 ...
##  $ unemploy: num [1:574] 2944 2945 2958 3143 3066 ...
```

representation

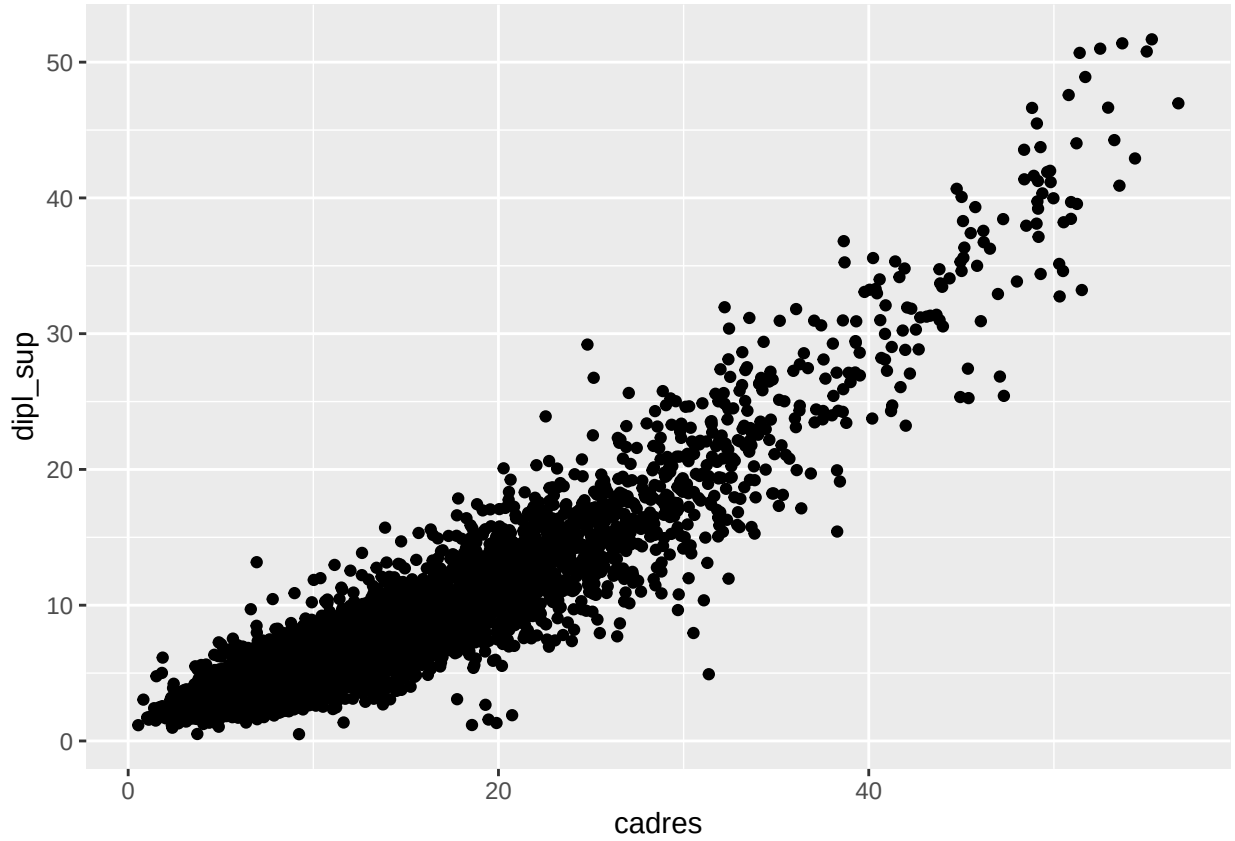
```
ggplot(economics) + geom_line(aes(x = date, y = unemploy))
```



3.2.9 La fonction geom-hex et geom-bin2d

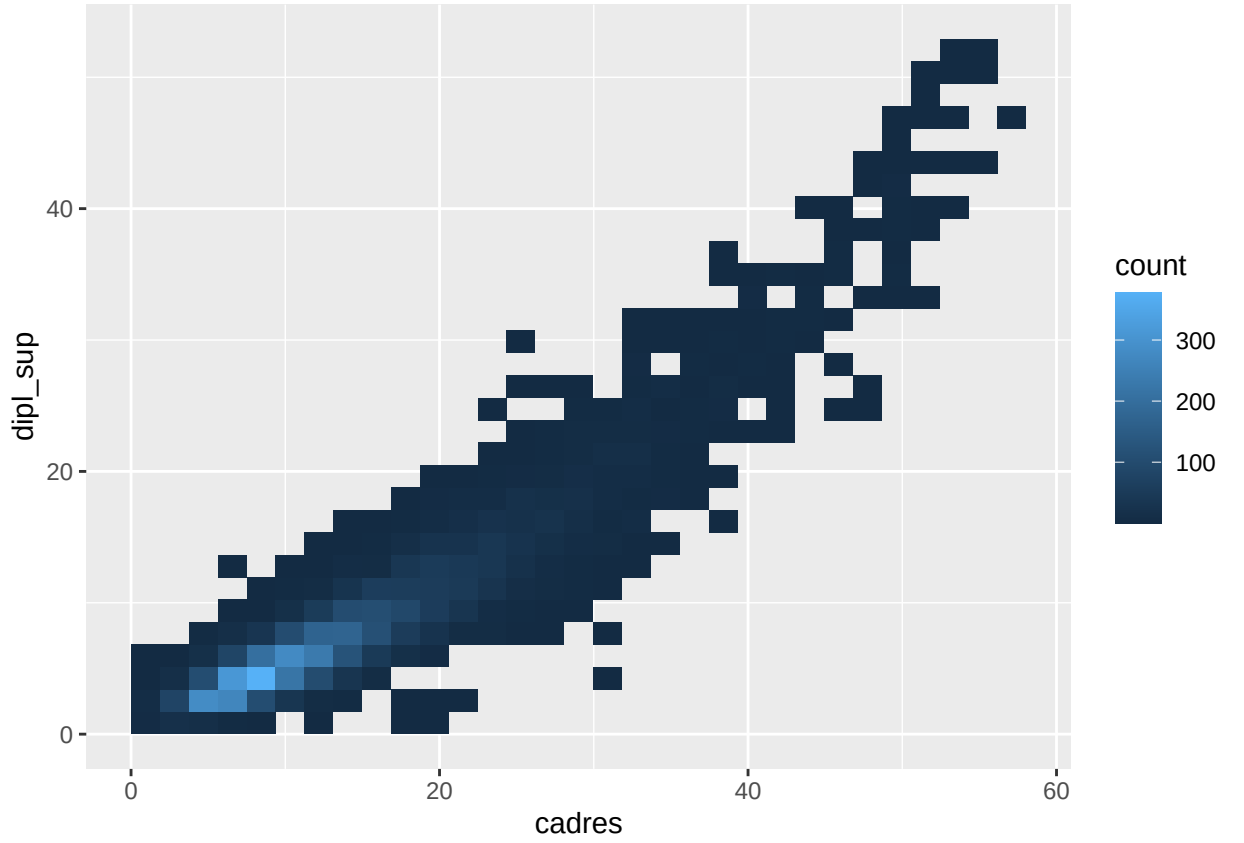
Lorsque le nombre de points est important, la représentation sous forme de nuage peut vite devenir illisible : la superposition des données empêche de voir précisément leur répartition.

```
ggplot(rp2018) + geom_point(aes(x = cadres, y = dipl_sup))
```



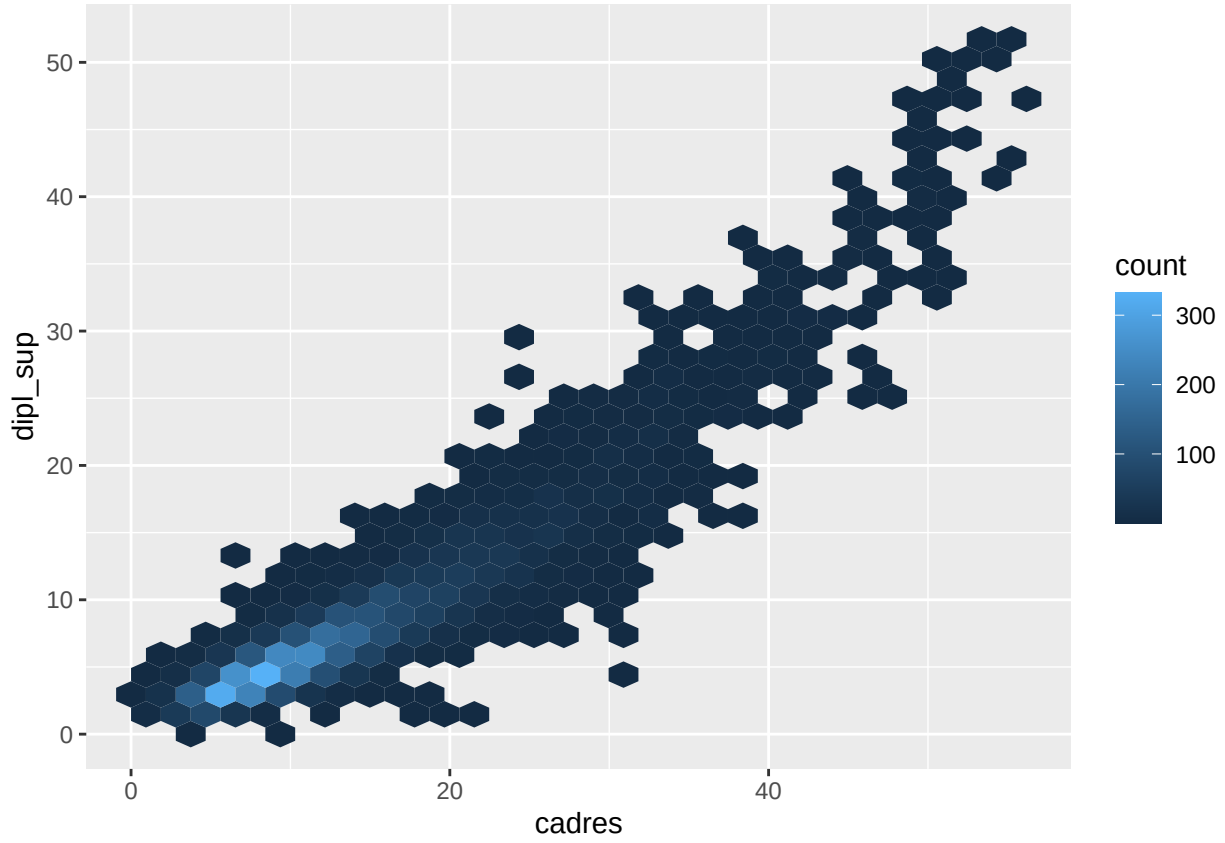
Dans ces cas-là, on peut utiliser geom_bin2d, qui va créer une grille sur toute la zone du graphique et colorier chaque carré selon le nombre de points qu'il contient (les carrés n'en contenant aucun restant transparents).

```
ggplot(rp2018) +
  geom_bin2d(aes(x = cadres, y = dipl_sup))
```



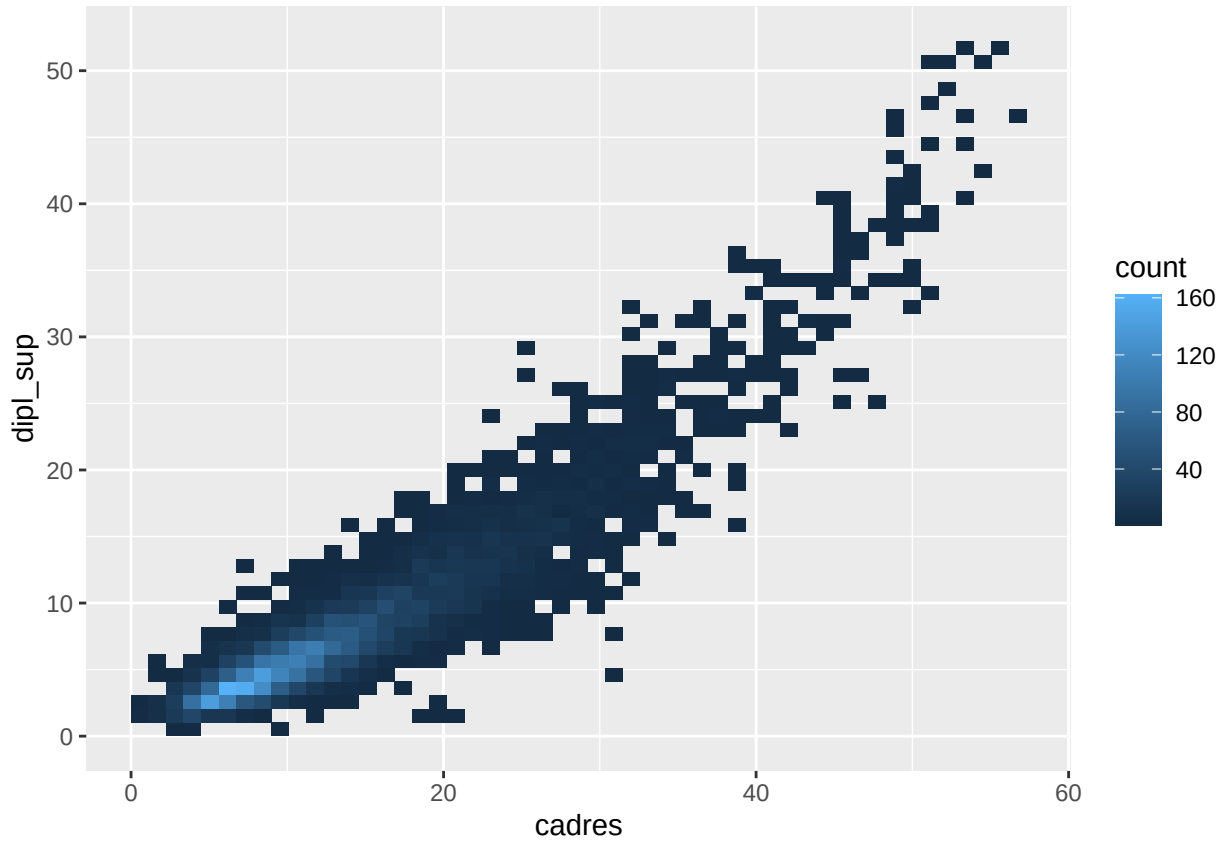
Une variante fonctionnant de manière très semblable est geom_hex, qui elle crée une grille constituée d'hexagones.

```
ggplot(rp2018) +
  geom_hex(aes(x = cadres, y = dipl_sup))
```

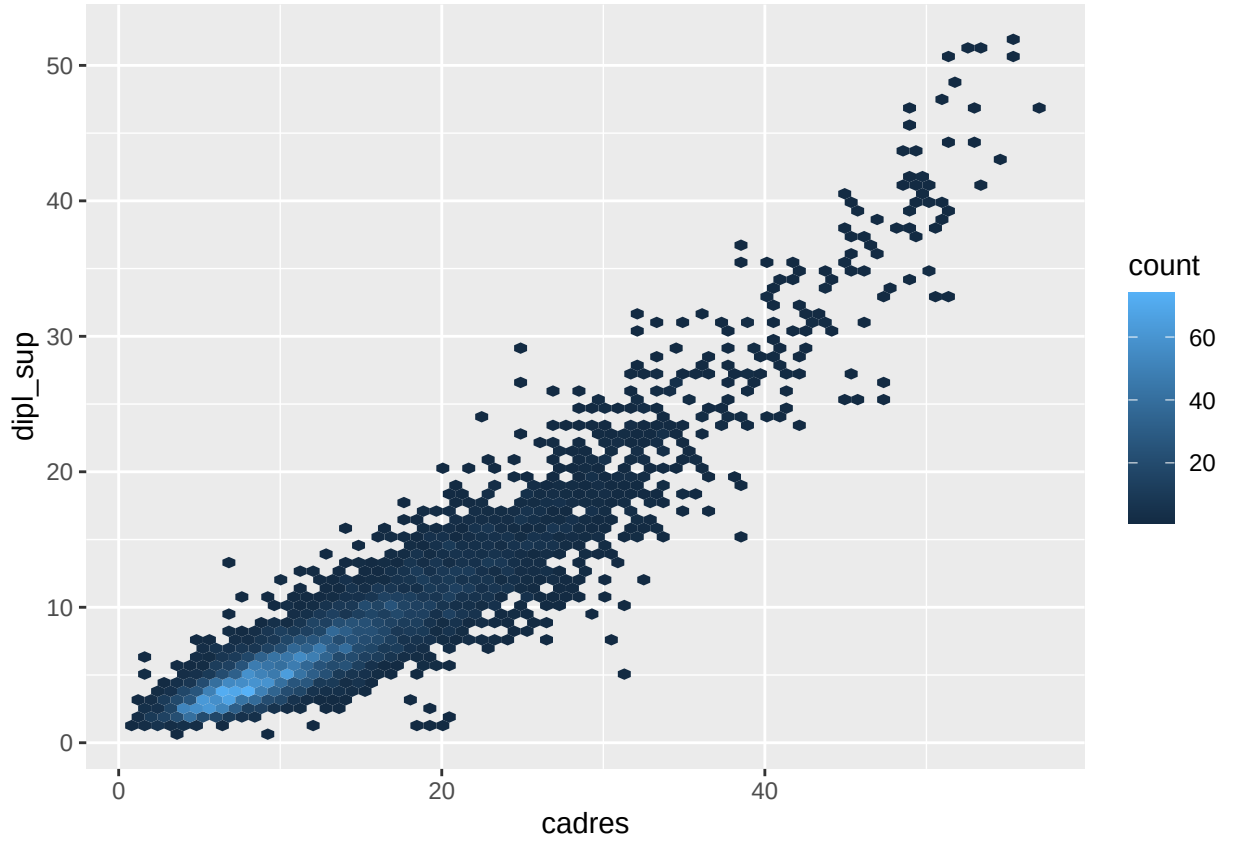


Dans les deux cas, on peut faire varier le nombre de zones, et donc la finesse du “quadrillage”, en utilisant l’argument bins (dont la valeur par défaut est 30).

```
ggplot(rp2018) +
  geom_bin2d(
    aes(x = cadres, y = dipl_sup),
    bins = 50
  )
```




```
ggplot(rp2018) +
  geom_hex(
    aes(x = cadres, y = dipl_sup),
    bins = 70
  )
```

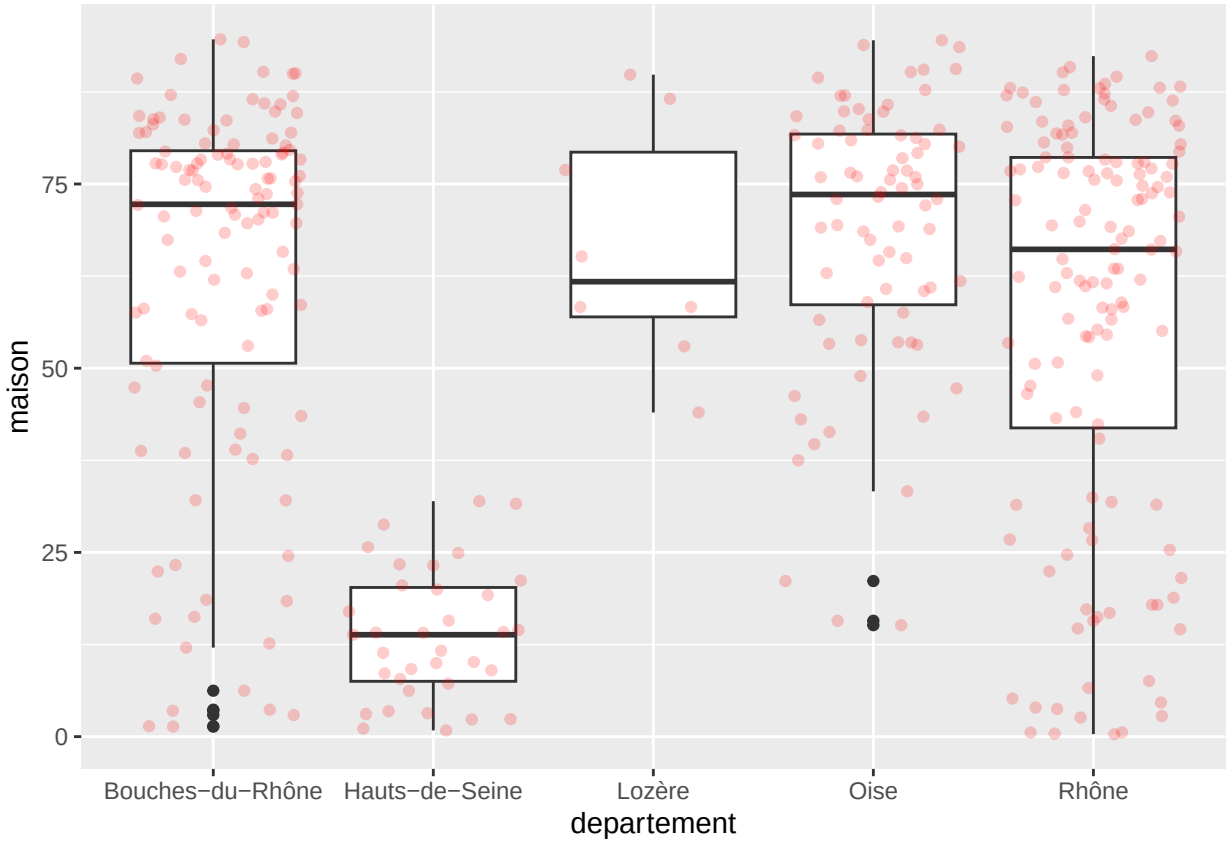


3.3 Représentation de plusieurs geom

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

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

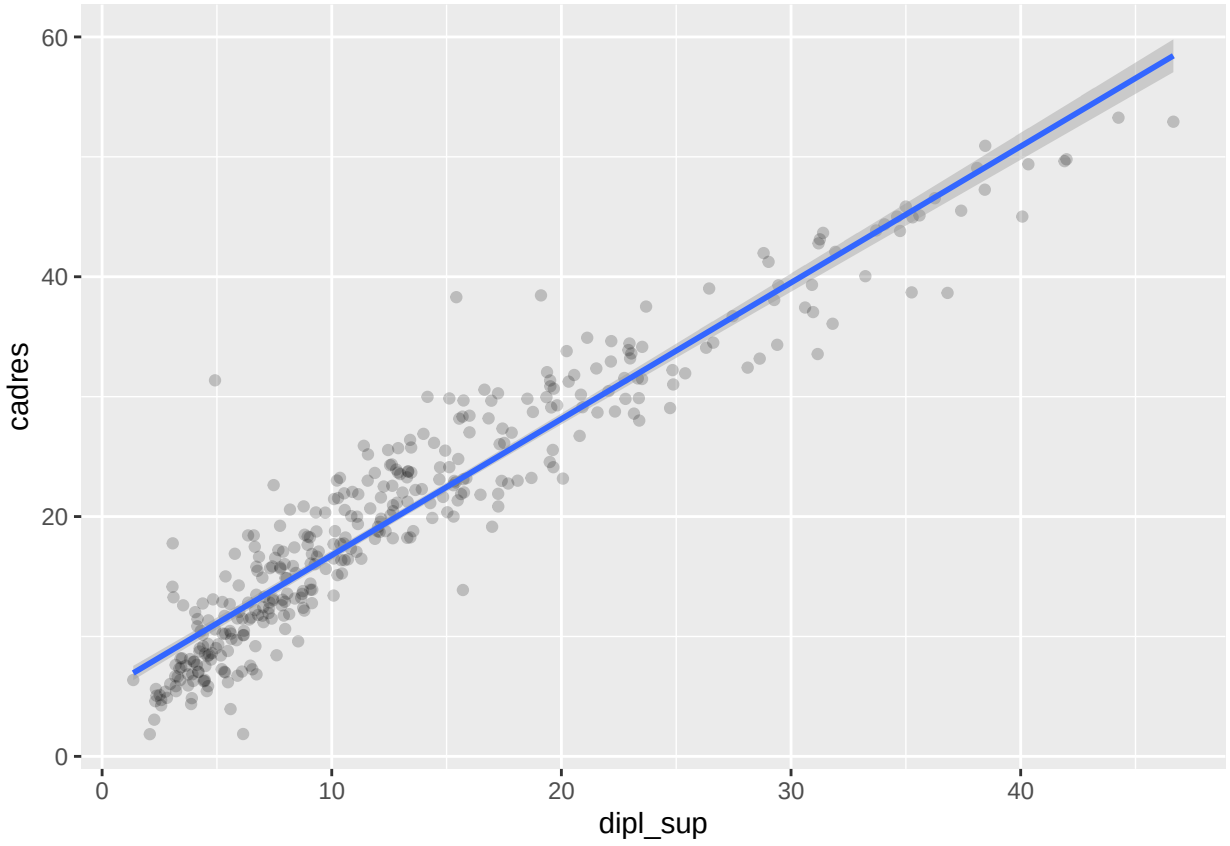
```
ggplot(rp, aes(x = departement, y = maison)) +
  geom_boxplot() +
  geom_jitter(color = "red", alpha = 0.2)
```



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'
```



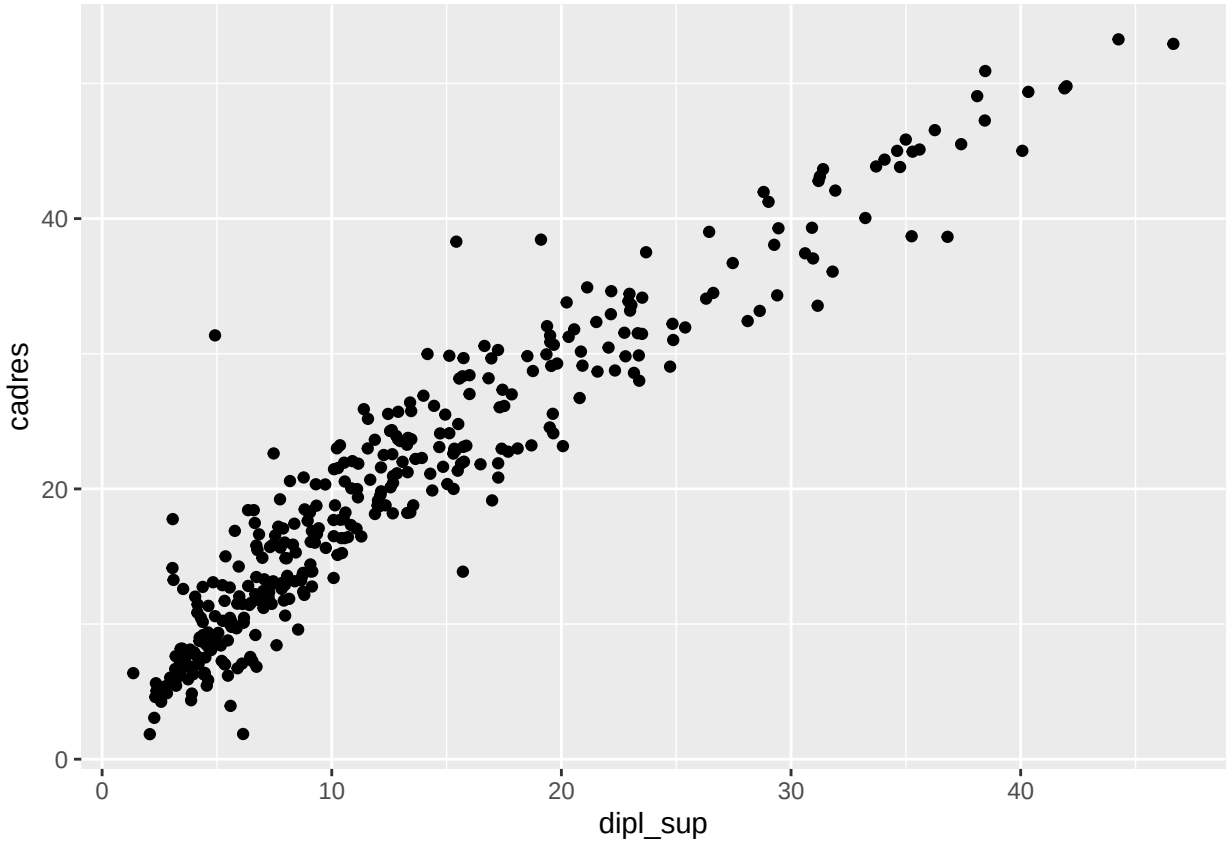

```
#> `geom_smooth()` using formula = 'y ~ x'
```

3.4 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.

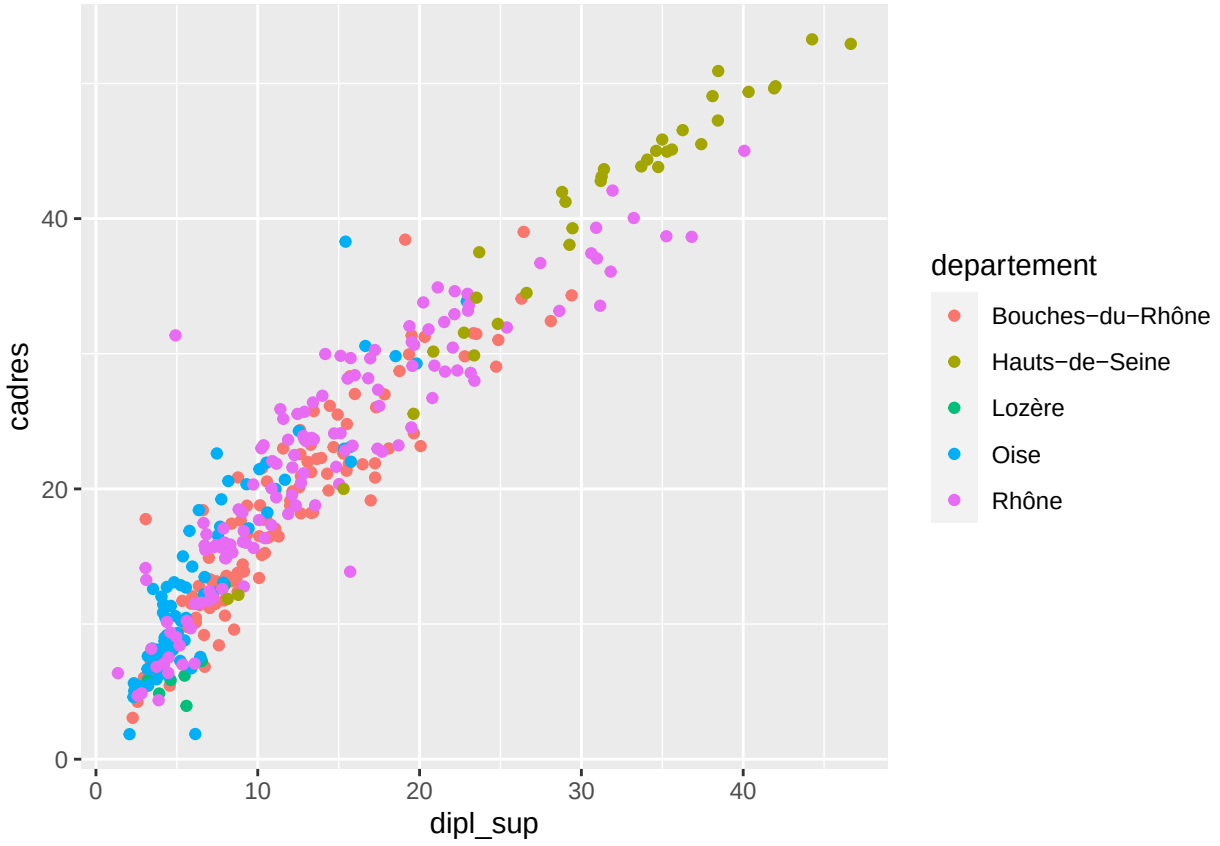
Ces mappages sont passés aux différents geom via la fonction aes() (abbréviation d'aesthetic).

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



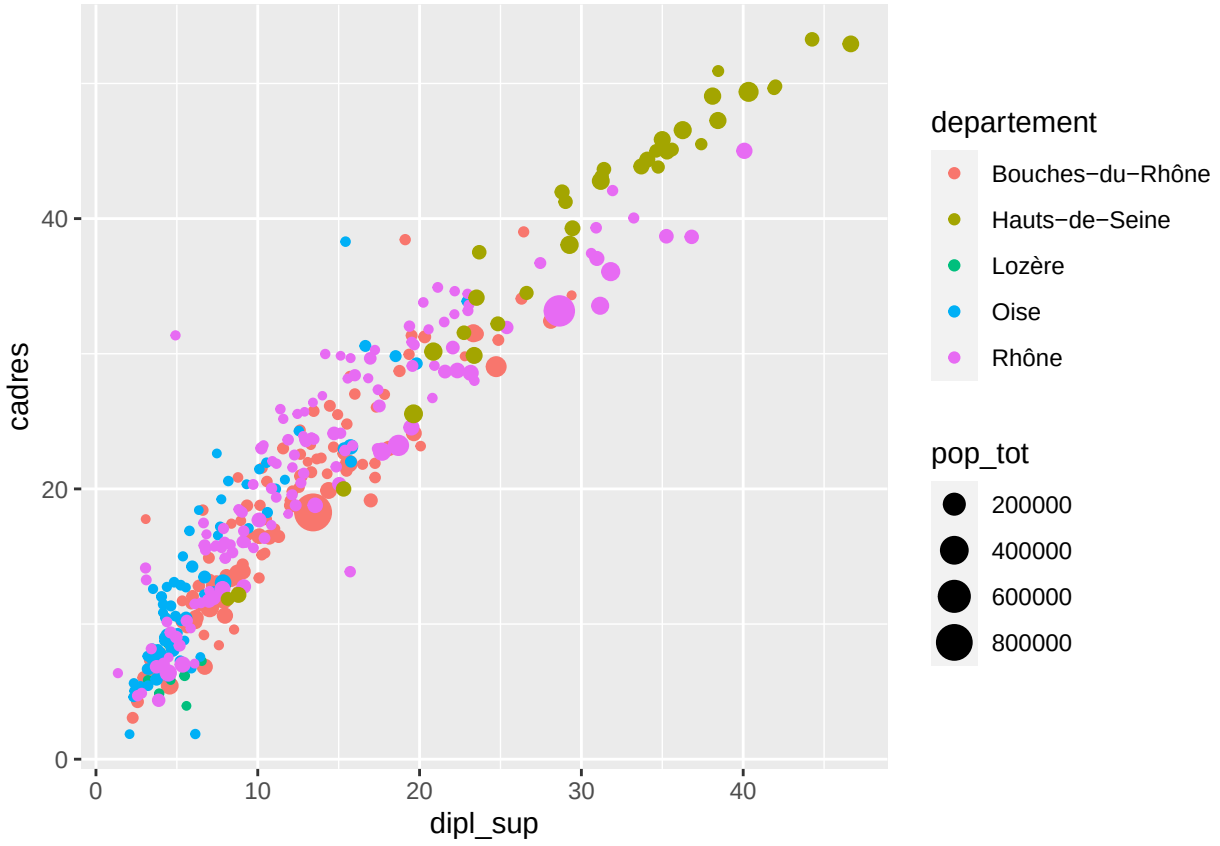
Mais on peut ajouter d'autres mappages. Par exemple, color permet de faire varier la couleur des points automatiquement en fonction des valeurs d'une troisième variable. Ainsi, on peut vouloir colorer les points selon le département de la commune correspondante.

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



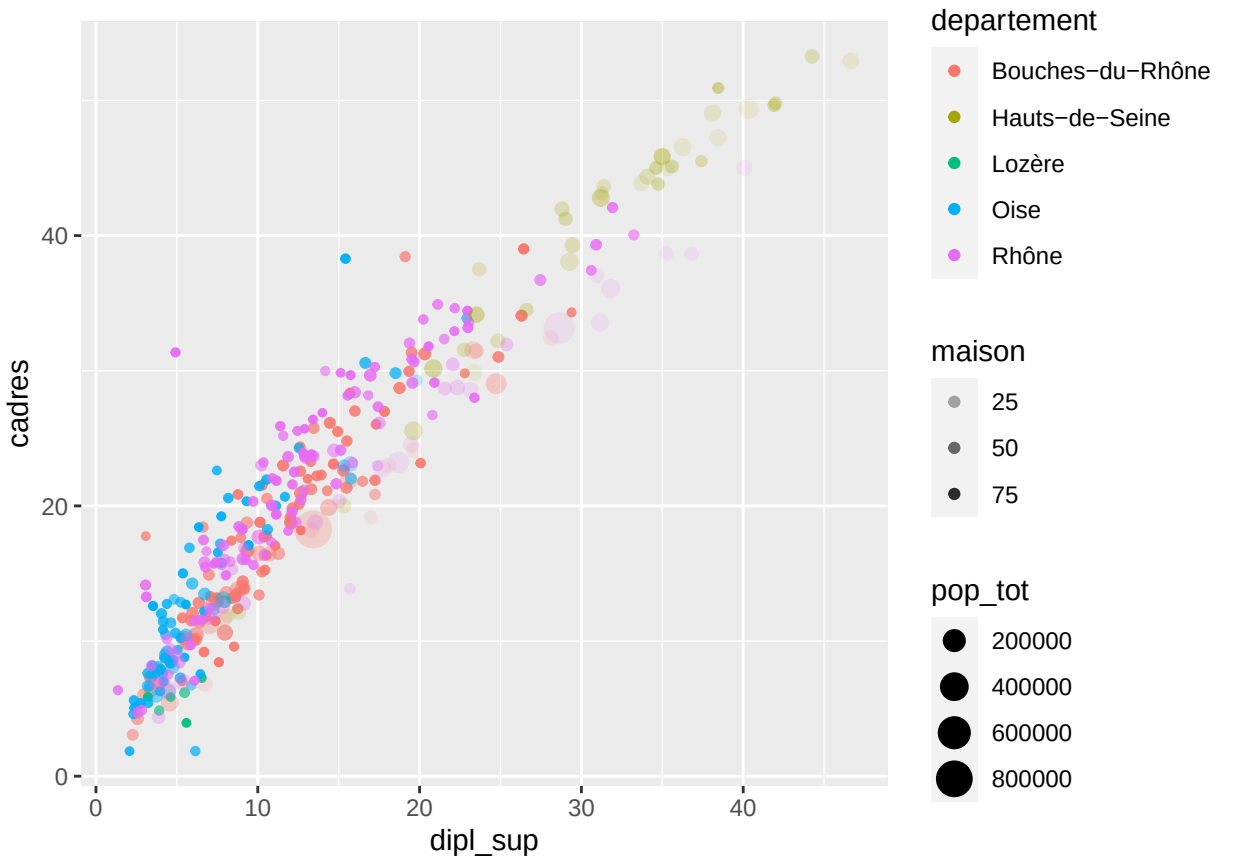
On peut aussi faire varier la taille des points avec size. Ici, la taille dépend de la population totale de la commune :

```
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.

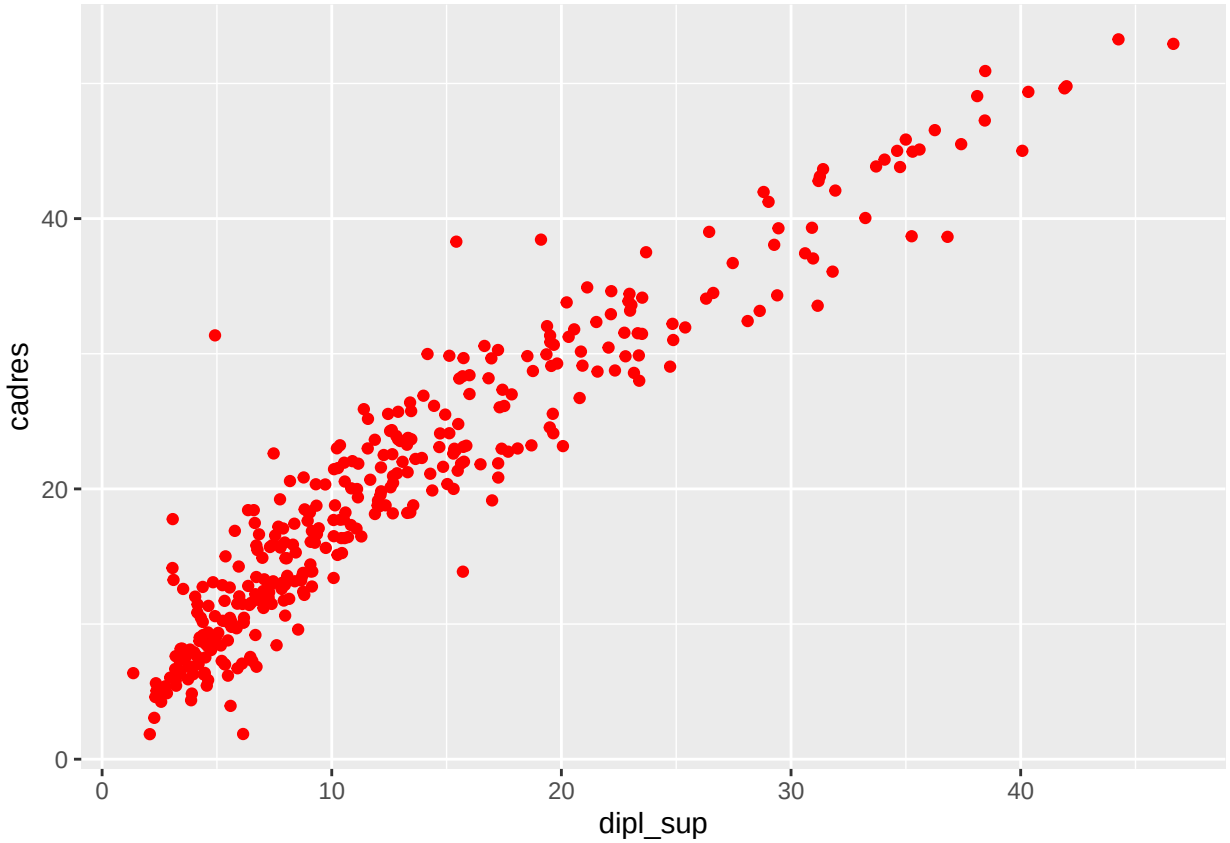
```
ggplot(rp) +
  geom_point(
    aes(x = dipl_sup, y = cadres, color = departement, size = pop_tot, alpha = maison)
  )
```



- aes() or not aes()

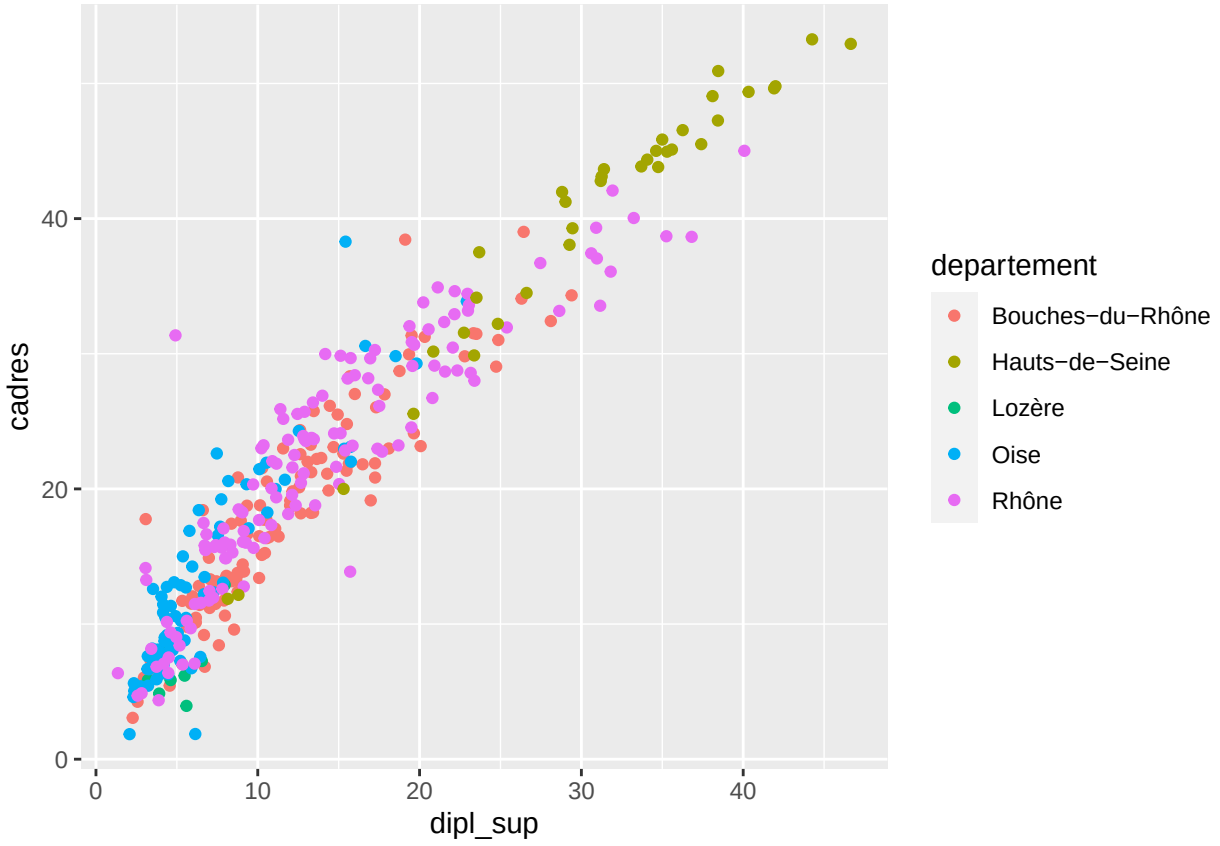
Comme on l'a déjà vu, parfois on souhaite changer un attribut sans le relier à une variable : c'est le cas par exemple si on veut représenter tous les points en rouge. Dans ce cas on utilise toujours l'attribut color, mais comme il ne s'agit pas d'un mappage, on le définit à l'extérieur de la fonction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

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



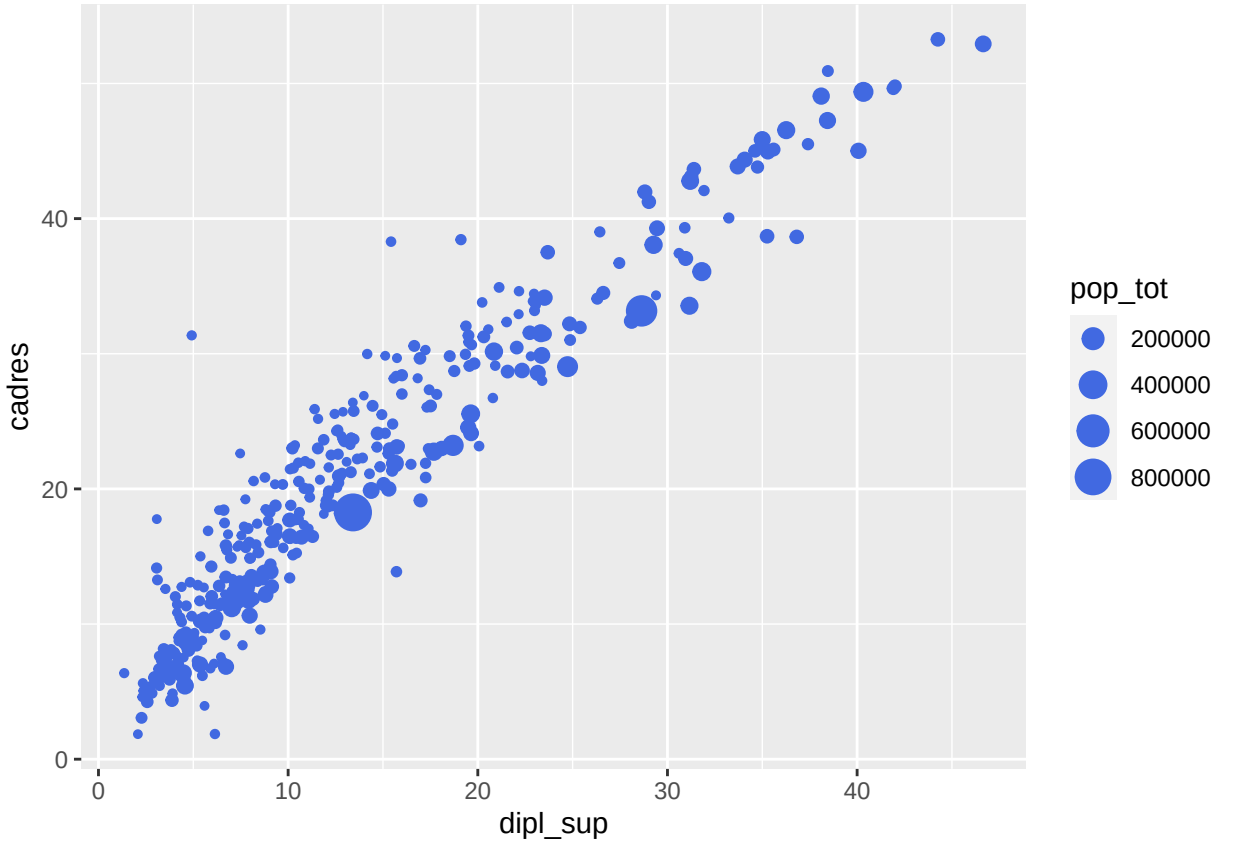
Par contre, si on veut faire varier la couleur en fonction des valeurs prises par une variable, on réalise un mappage, et on doit donc placer l'attribut color à l'intérieur de aes().

```
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"
  )
```



3.5 Scales

On a vu qu'avec ggplot2 on définit des mappages entre des attributs graphiques (position, taille, couleur, etc.) et des variables d'un tableau de données. Ces mappages sont définis, pour chaque geom, via la fonction aes().

Les scales dans ggplot2 permettent de modifier la manière dont un attribut graphique va être relié aux valeurs d'une variable, et dont la légende correspondante va être affichée. Par exemple, pour l'attribut color, on pourra définir la palette de couleur utilisée. Pour size, les tailles minimales et maximales, etc.

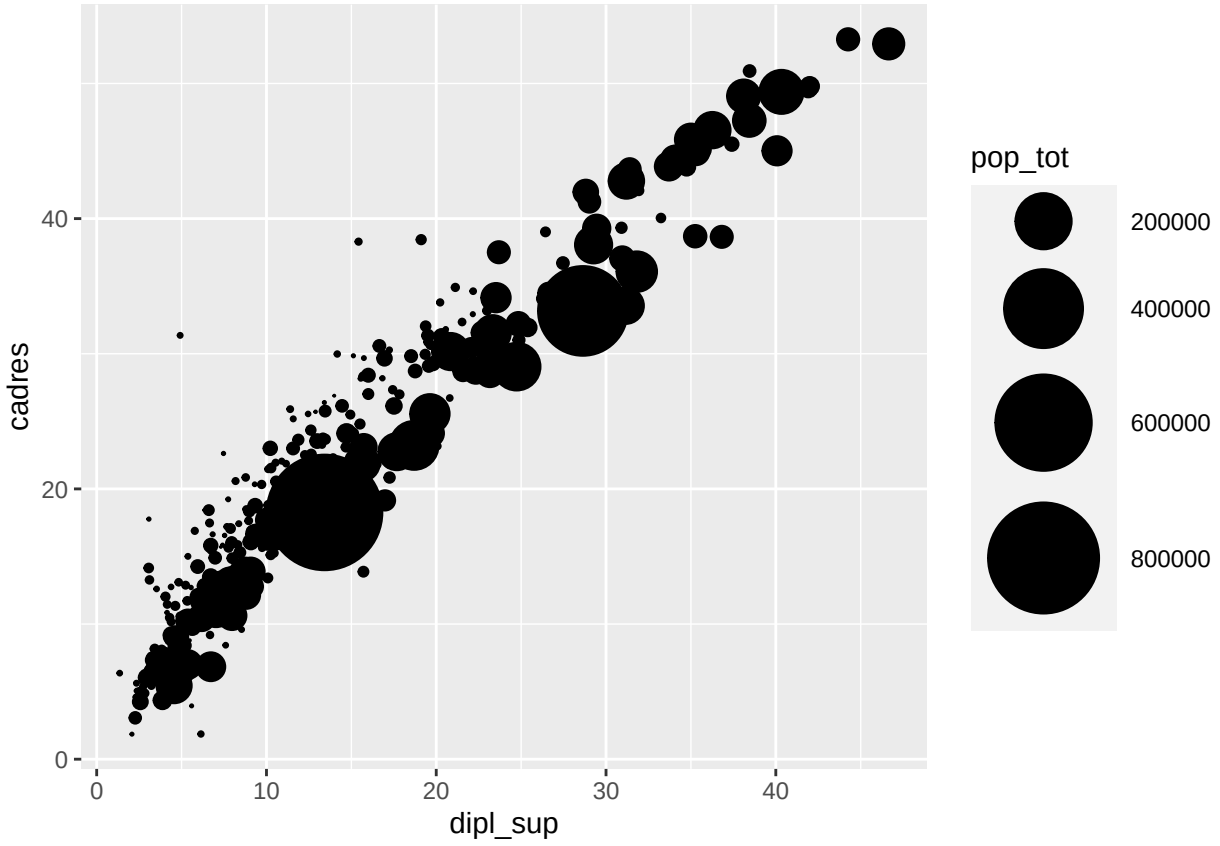
Pour modifier une scale existante, on ajoute un nouvel élément à notre objet ggplot2 avec l'opérateur +. Cet élément prend la forme scale___.

Voyons tout de suite quelques exemples.

3.5.1 scale-size

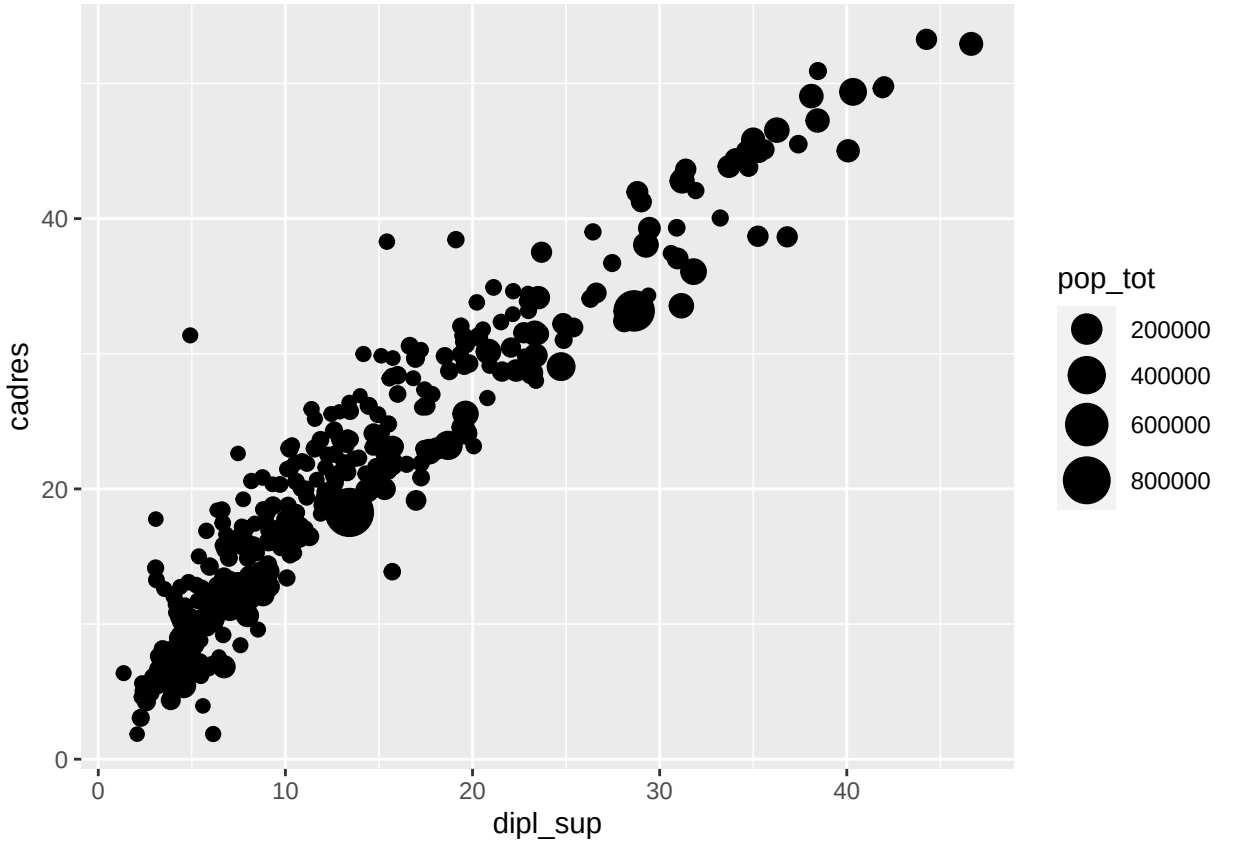
Si on souhaite modifier les tailles minimales et maximales des objets quand on a effectué un mappage de type size, on peut utiliser la fonction scale_size et son argument range.

```
ggplot(rp) +
  geom_point(aes(x = dipl_sup, y = cadres, size = pop_tot)) +
  scale_size(range = c(0, 20))
```



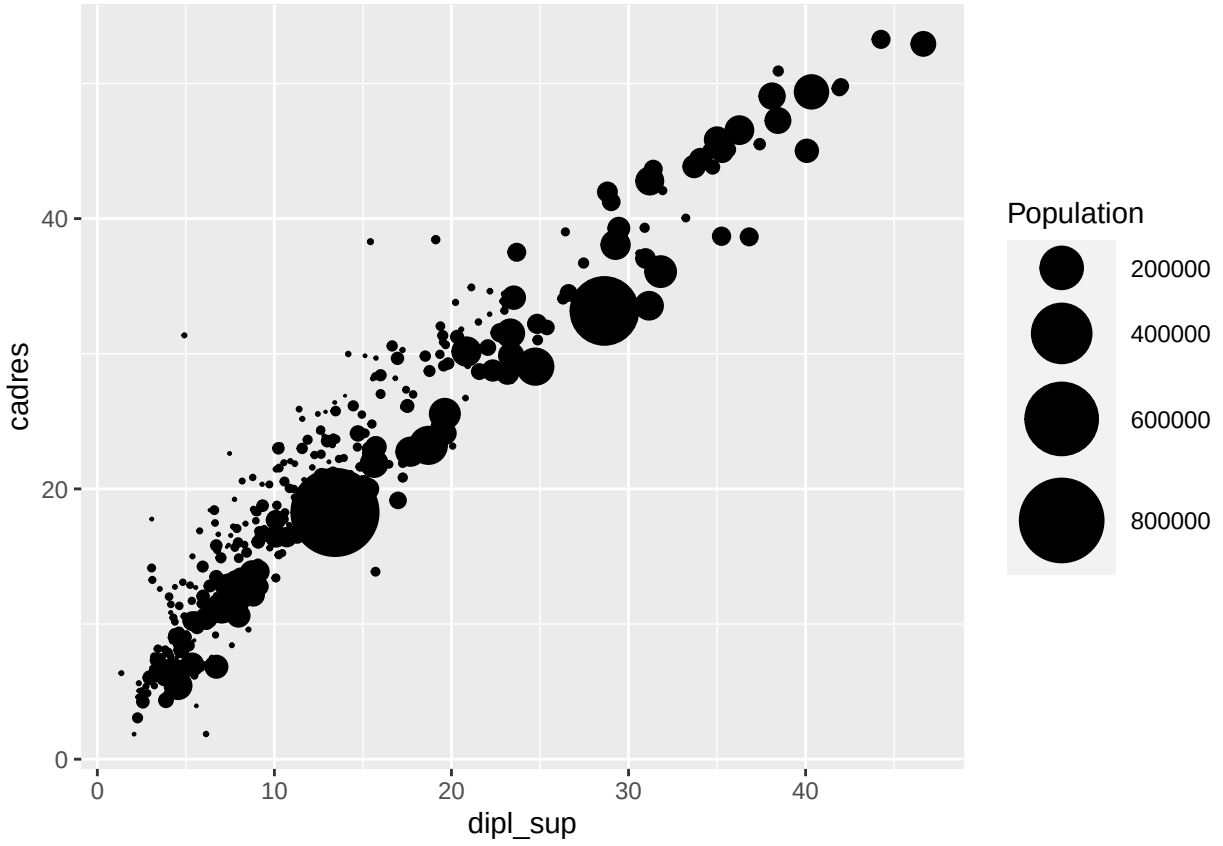
À comparer par exemple à :

```
ggplot(rp) +
  geom_point(aes(x = dipl_sup, y = cadres, size = pop_tot)) +
  scale_size(range = c(2, 8))
```



On peut ajouter d'autres paramètres à scale_size. Le premier argument est toujours le titre donné à la légende.

```
ggplot(rp) +
  geom_point(aes(x = dipl_sup, y = cadres, size = pop_tot)) +
  scale_size(
    "Population",
    range = c(0, 15)
  )
```



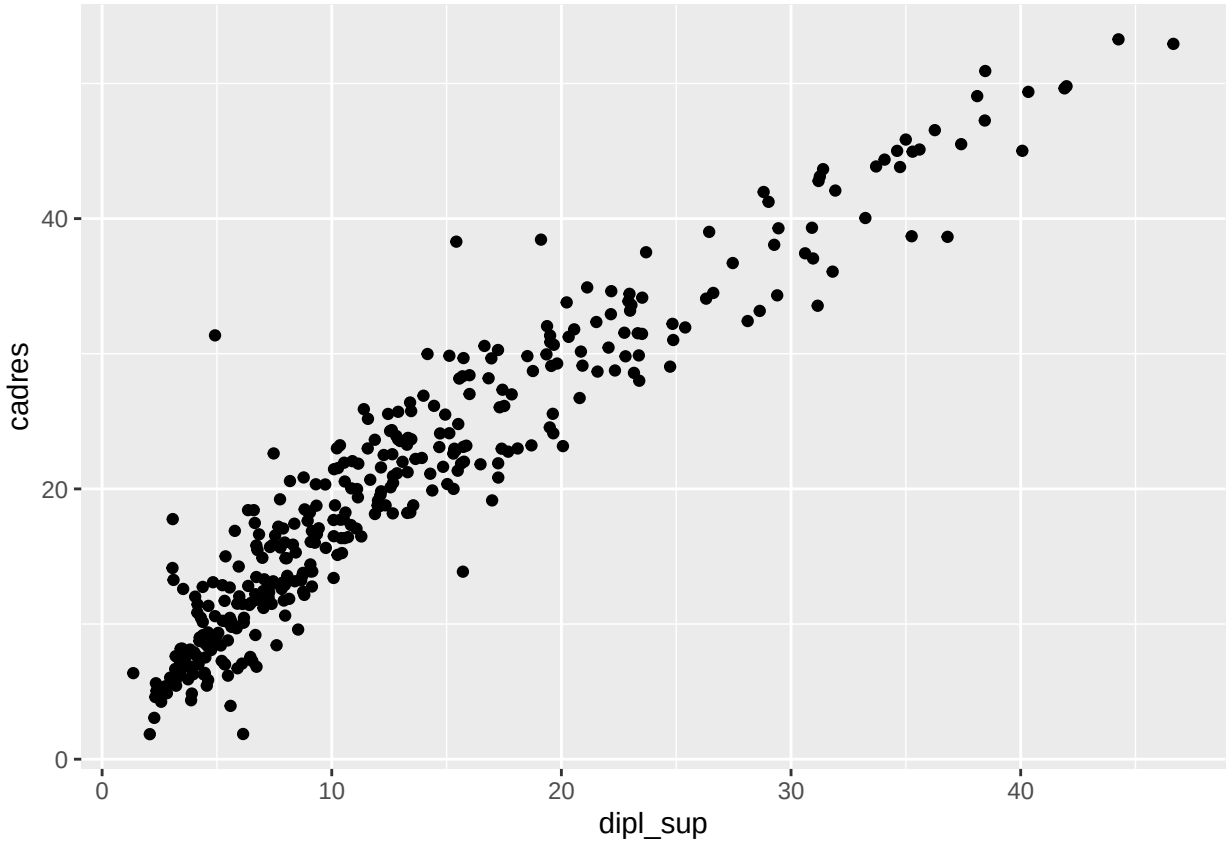
3.5.2 scale-x, scale-y

Les scales scale_x_ et scale_y_ modifient les axes x et y du graphique.

scale_x_continuous et scale_y_continuous s'appliquent lorsque la variable x ou y est numérique (quantitative).

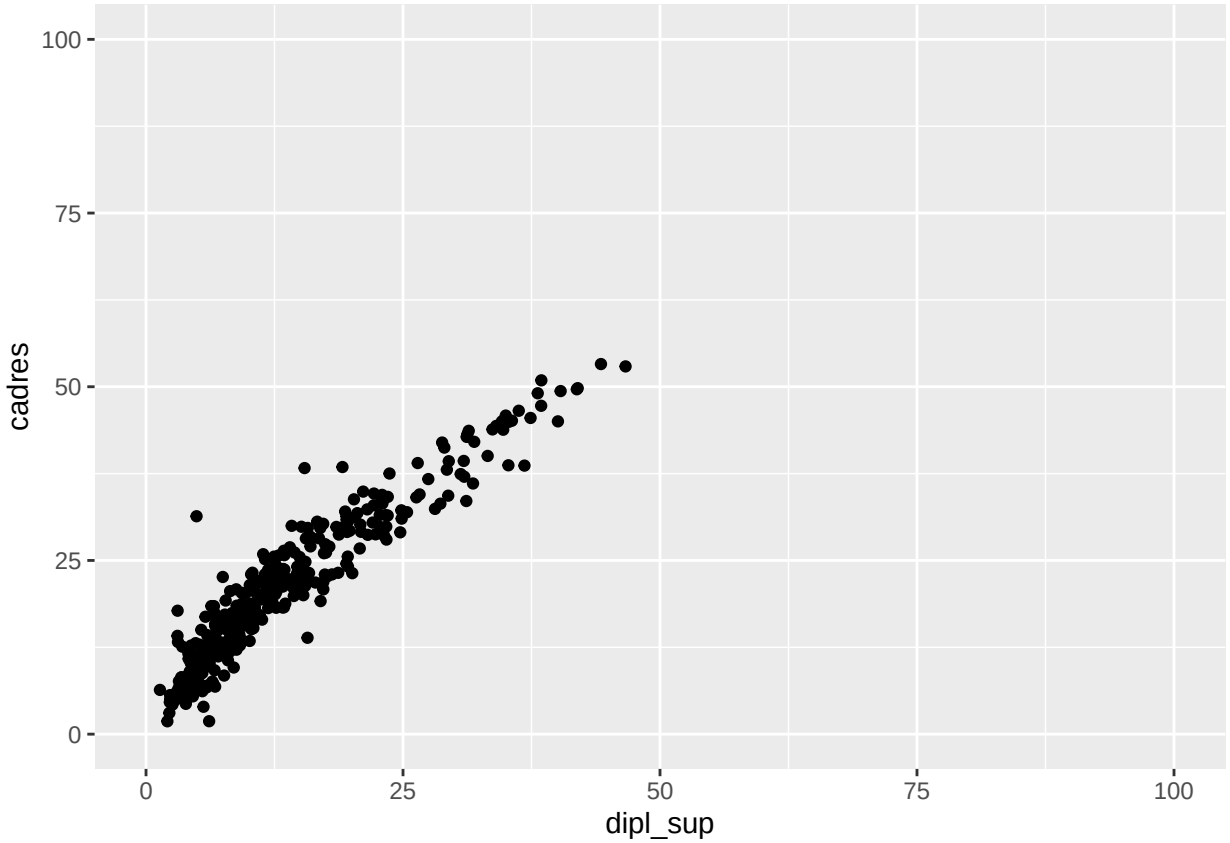
C'est le cas de notre nuage de points croisant part de cadres et part de diplômés du supérieur.

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



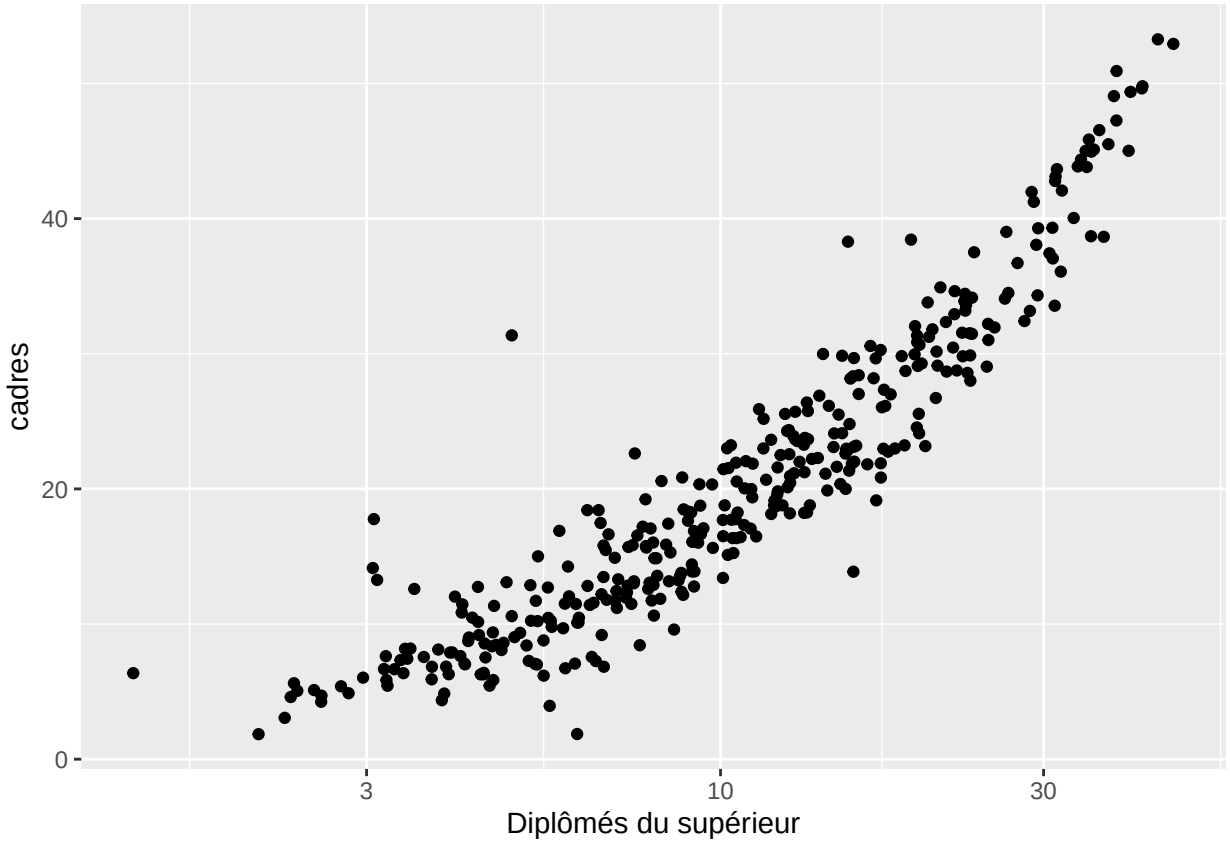
Comme on représente des pourcentages, on peut vouloir forcer les axes x et y à s'étendre des valeurs 0 à 100. On peut le faire en ajoutant un élément scale_x_continuous et un élément scale_y_continuous, et en utilisant leur argument limits.

```
ggplot(rp) +
  geom_point(aes(x = dipl_sup, y = cadres)) +
  scale_x_continuous(limits = c(0,100)) +
  scale_y_continuous(limits = c(0,100))
```



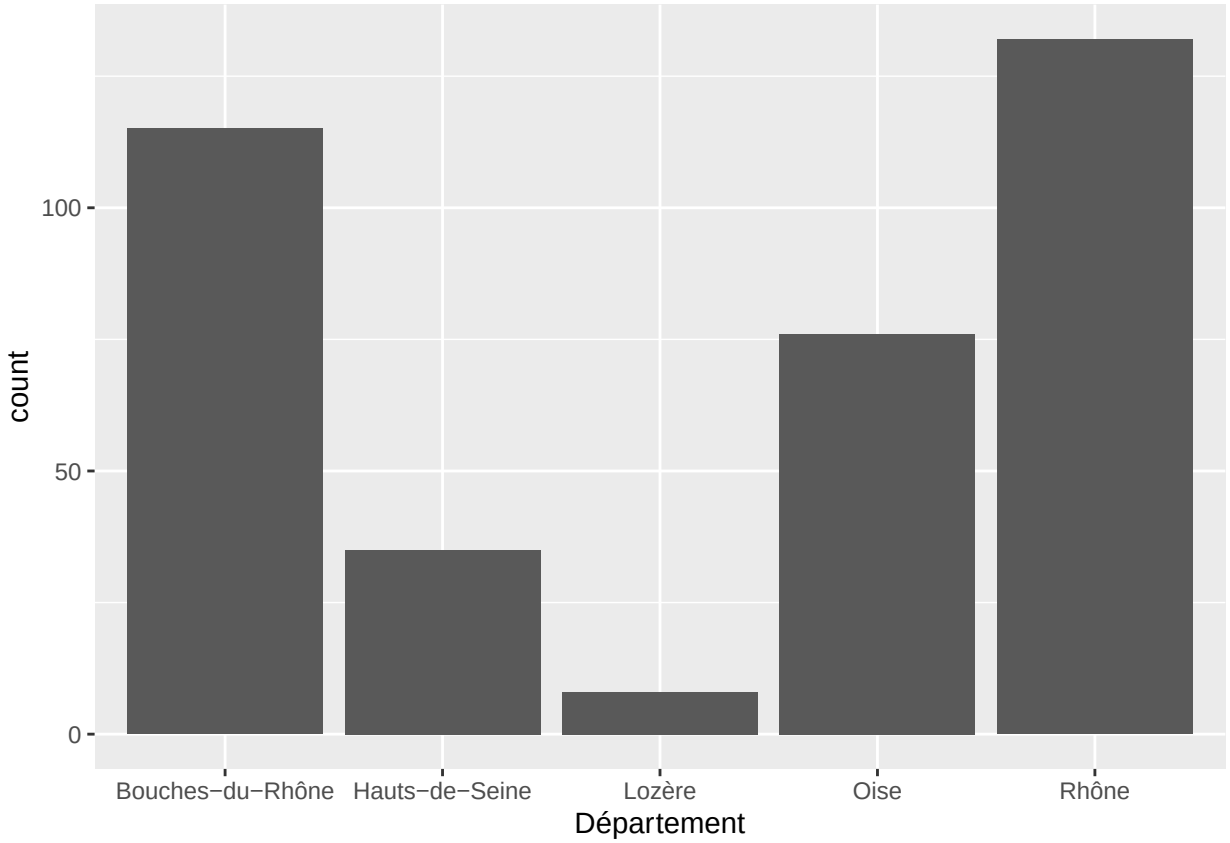
On peut utiliser scale_x_log10 et scale_y_log10 pour passer un axe à une échelle logarithmique.

```
ggplot(rp) +
  geom_point(aes(x = dipl_sup, y = cadres)) +
  scale_x_log10("Diplômés du supérieur")
```



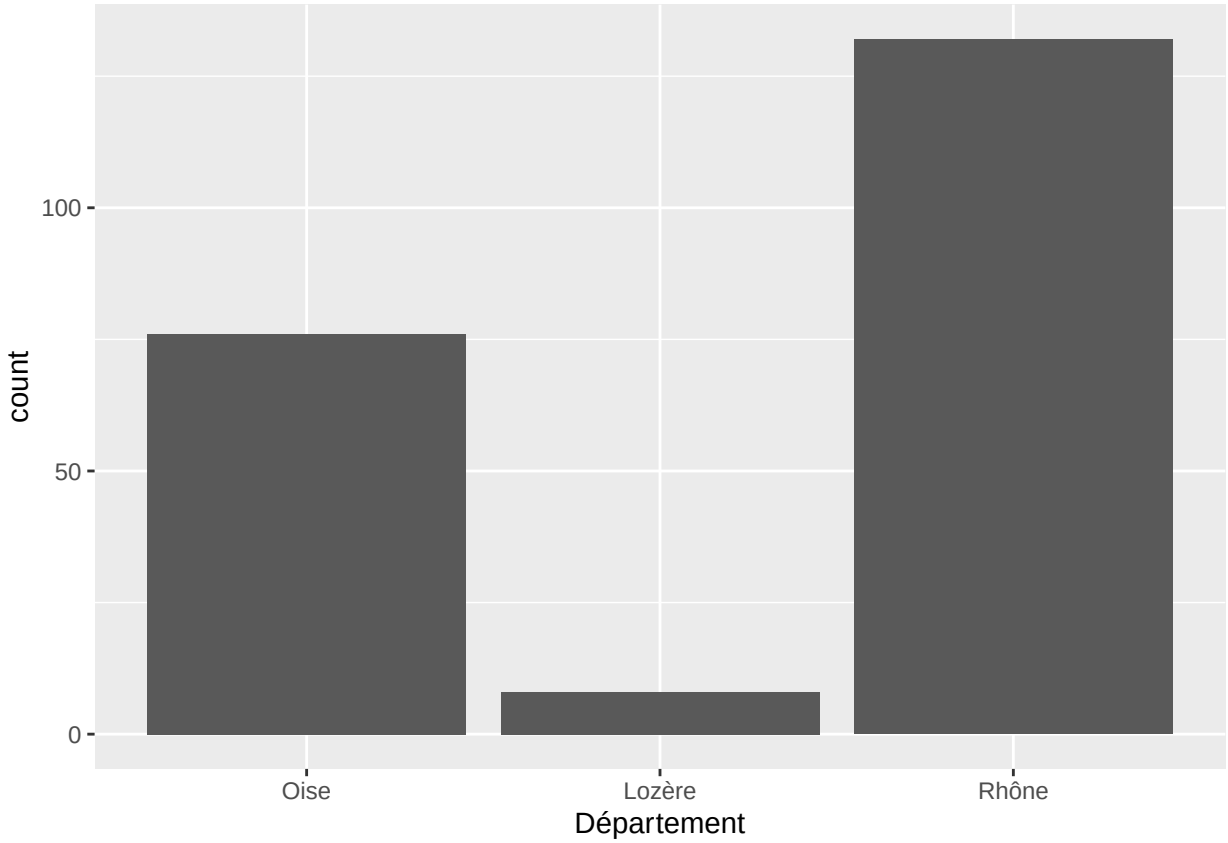
scale_x_discrete et scale_y_discrete s'appliquent lorsque l'axe correspond à une variable discrète (qualitative). C'est le cas par exemple de l'axe des x dans un diagramme en barres.

```
ggplot(rp) +
  geom_bar(aes(x = departement)) +
  scale_x_discrete("Département")
```



L'argument limits de scale_x_discrete permet d'indiquer quelles valeurs sont affichées et dans quel ordre.

```
ggplot(rp) +
  geom_bar(aes(x = departement)) +
  scale_x_discrete("Département", limits = c("Oise", "Lozère", "Rhône"))
```




```
#> Warning: Removed 150 rows containing non-finite values (`stat_count()`).
```

3.5.3 scale-color, scale-fill

Ces scales permettent, entre autre, de modifier les palettes de couleur utilisées pour le dessin (color) ou le remplissage (fill) des éléments graphiques. Dans ce qui suit, pour chaque fonction scale_color présentée il existe une fonction scale_fill équivalente et avec en général les mêmes arguments.

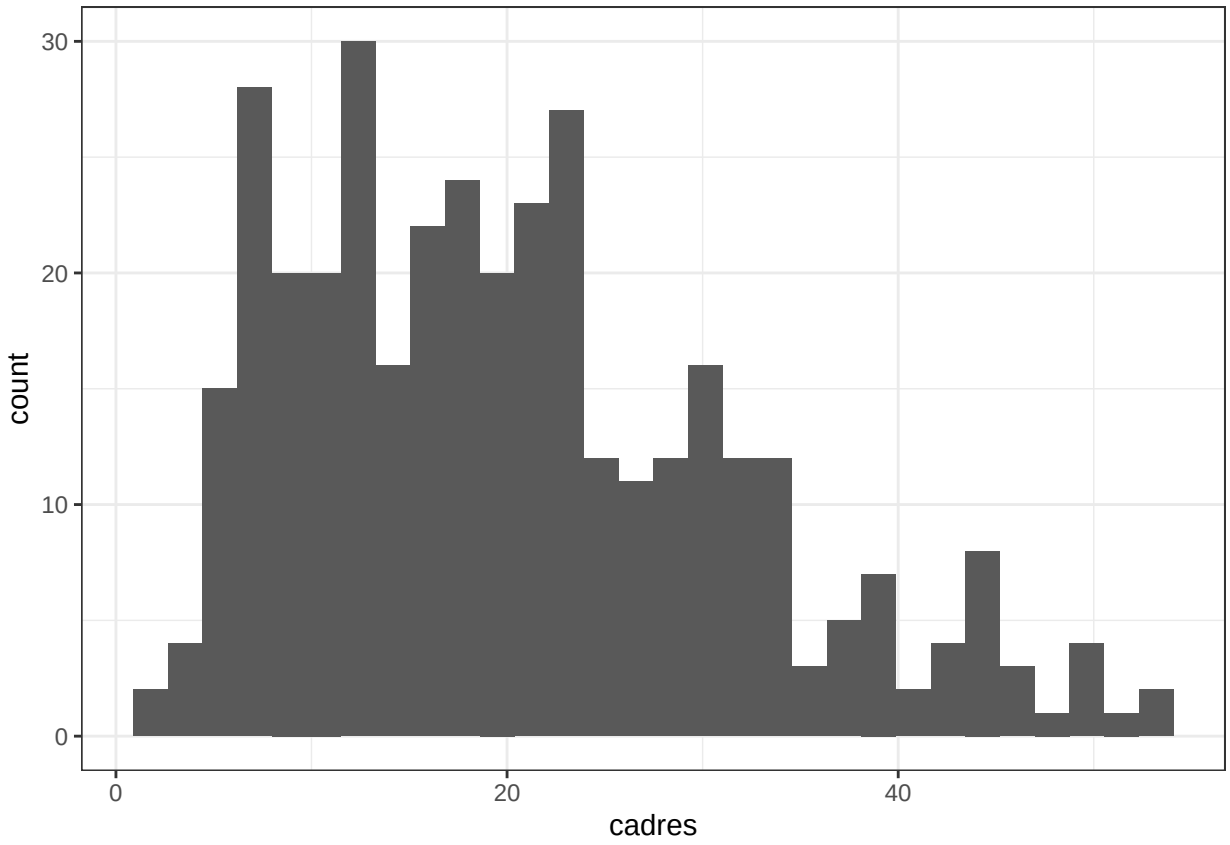
3.6 Thèmes

Les thèmes permettent de contrôler l'affichage de tous les éléments du graphique qui ne sont pas reliés aux données : titres, grilles, fonds, etc.

Il existe un certain nombre de thèmes préexistants à savoir Le thème "Excel","theme_minimal" ,le"theme_classic",le "theme_bw".... Par exemple le thème theme_bw qui propose un style noir et blanc qui convient bien pour des graphiques avec beaucoup de détails.

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

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

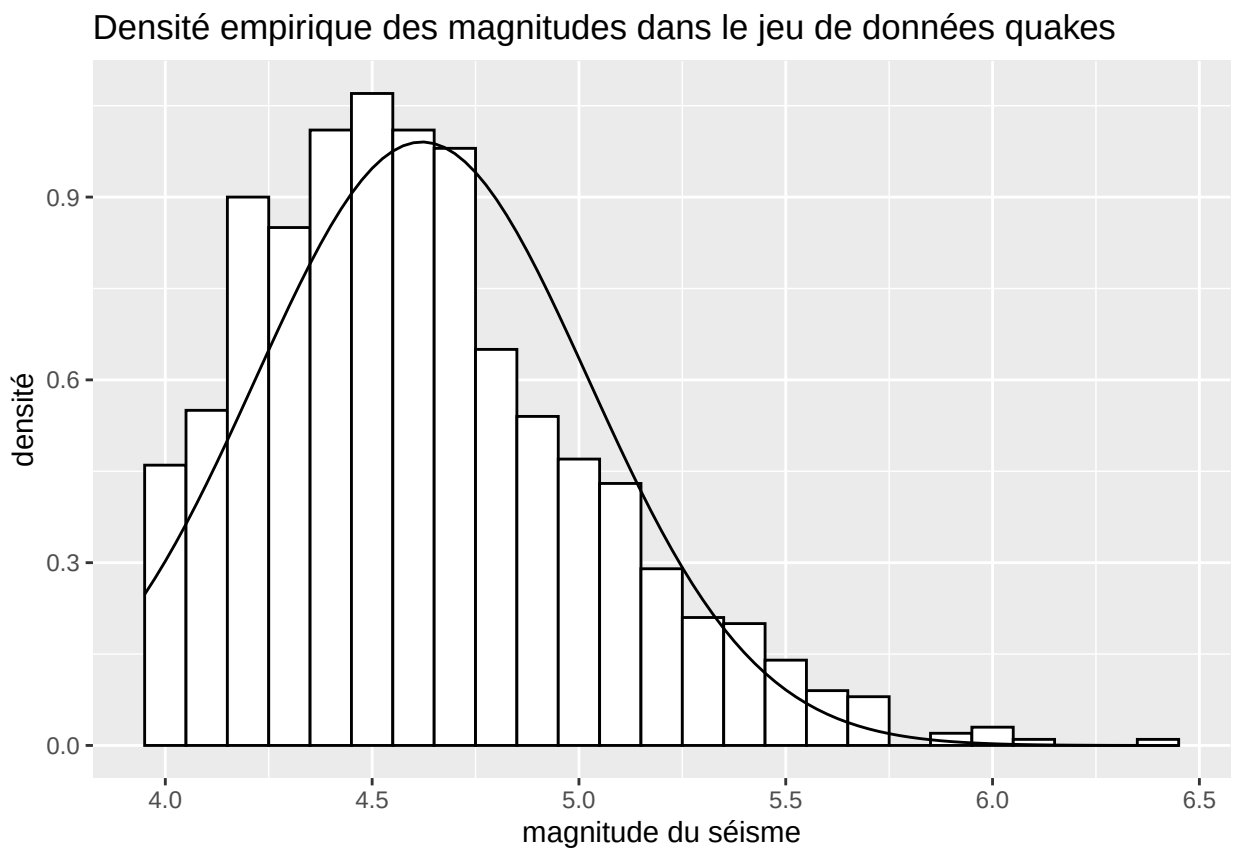


3.7 Représentation graphique d'une fonction

La fonction stat_function permet de représenter une fonction mathématique, un peu comme le fait la fonction curve du système graphique de base.

```
# Statistique dont nous aurons besoin
moy <- mean(quakes$mag)
et <- sd(quakes$mag)
```

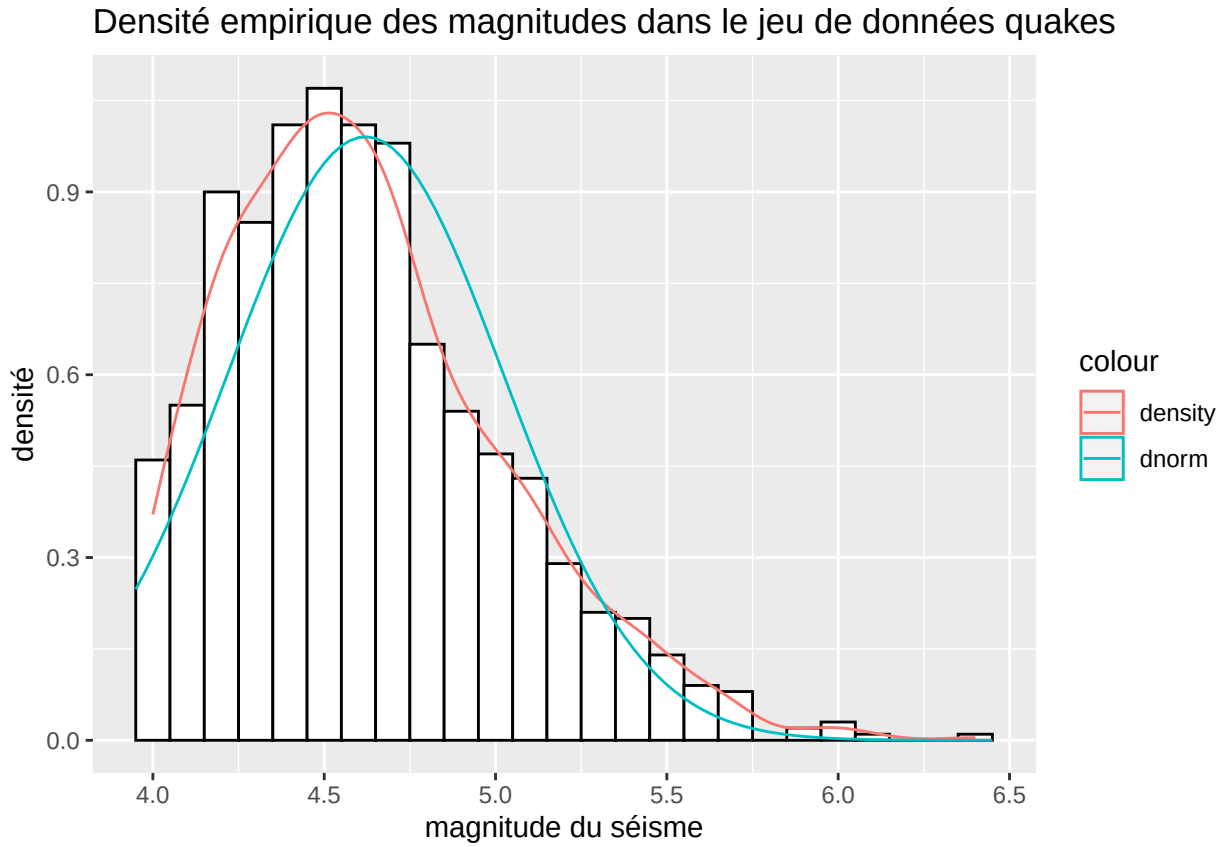
```
histogramme <- ggplot(data = quakes, mapping = aes(x = mag)) +
  geom_histogram(
    mapping = aes(y = after_stat(density)),
    binwidth = 0.1,
    colour = "black",
    fill = "white"
  ) +
  labs(
    title = "Densité empirique des magnitudes dans le jeu de données quakes",
    x = "magnitude du séisme",
    y = "densité"
  )

histogramme + stat_function(           # ajout d'une courbe de densité normale
    fun = dnorm,
    args = list(mean = moy, sd = et),  # avec paramètres estimés à partir des données
    xlim = c(3.95, 6.45)
  )
```



Ajoutons une courbe supplémentaire au graphique précédent.

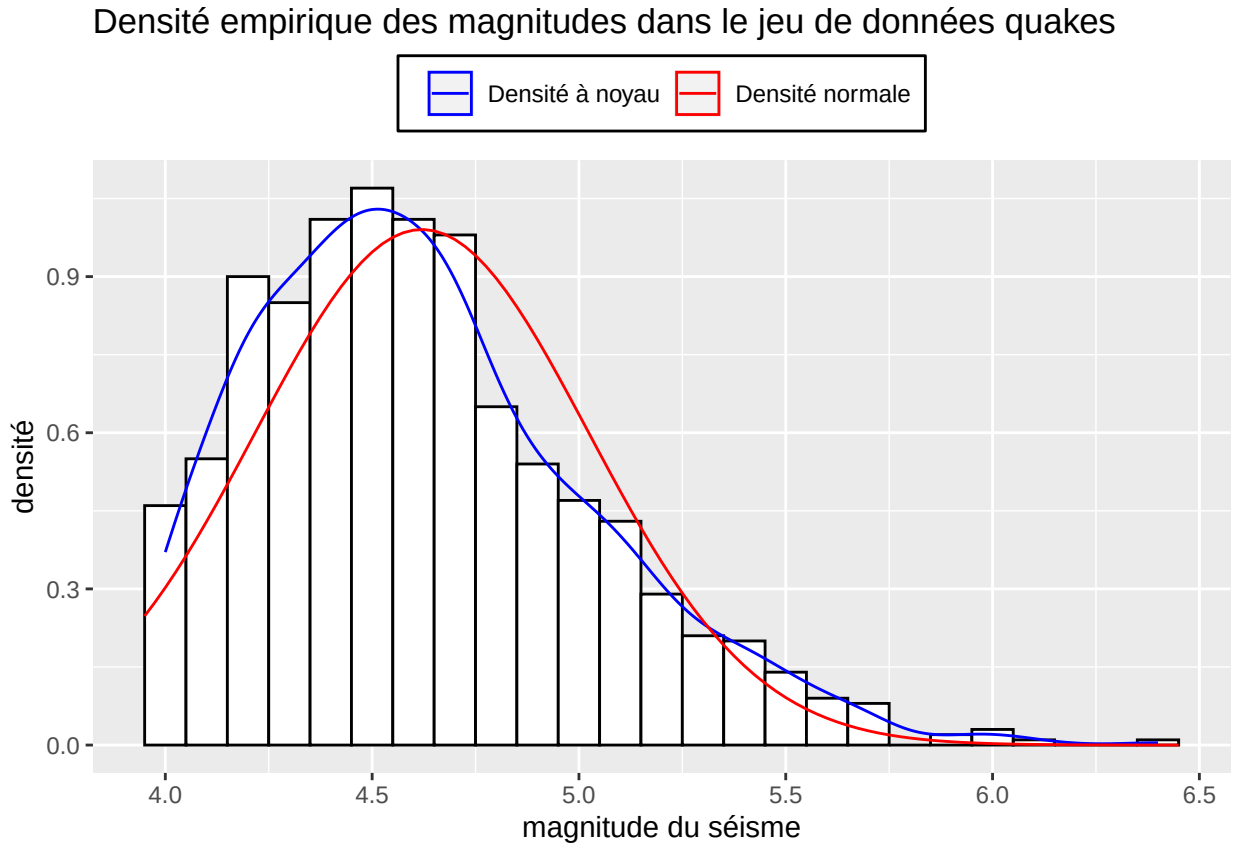
Afin de pouvoir identifier chacune des courbes, il faudrait choisir une propriété visuelle pour les distinguer. Choisissions la couleur.

```
histogramme <- histogramme +
  geom_density(
    aes(colour = "density"),  # association de la propriété visuelle colour à une valeur
  ) +
  stat_function(
    aes(colour = "dnorm"),    # association de la propriété visuelle colour à une valeur
    fun = dnorm,
    args = list(mean = moy, sd = et),
    xlim = c(3.95, 6.45)
  )
histogramme
```



Modifions les couleurs ainsi que les étiquettes des courbes et Grâce à la fonction theme, nous pouvons aussi contrôler l'allure de la légende.

```
histogramme + scale_colour_manual(
    name = "",
    values = c("density" = "blue", "dnorm" = "red"),
    labels = c("density" = "Densité à noyau", "dnorm" = "Densité normale")
  ) +
  theme(
    legend.position = "top",            # modification de la position de la légende
    legend.background = element_rect(  # ajout d'une bordure rectangulaire autour de la légende
      size = 0.5, linetype = "solid", colour = "black"
    )
  )
```



3.8 Packages souvent utilisés avec ggplot2

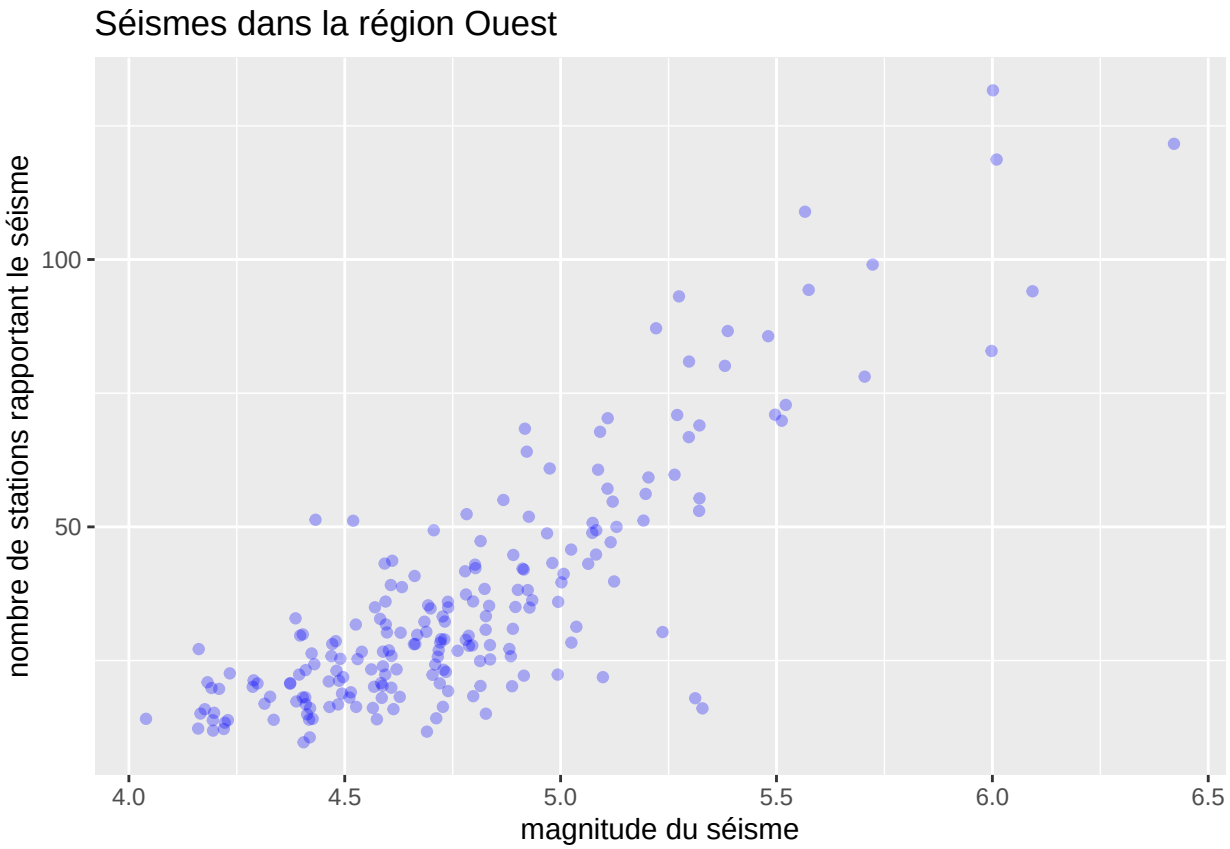
Le package ggplot2 est souvent utilisé conjointement à d'autres packages du tidyverse, notamment :

- dplyr pour la manipulation / le prétraitement de données;
- magrittr qui offre l'opérateur « pipe » %>%;
- forcats pour la manipulation de facteurs.

Exemple : production d'un graphique à partir d'un sous-ensemble de données

Supposons que nous voulions produire le diagramme de dispersion entre les variables mag et stations de quakes uniquement avec les observations dans la région Ouest. Nous pourrions nous servir de la fonction filter de dplyr et de l'opérateur %>% de magrittr de comme suit.

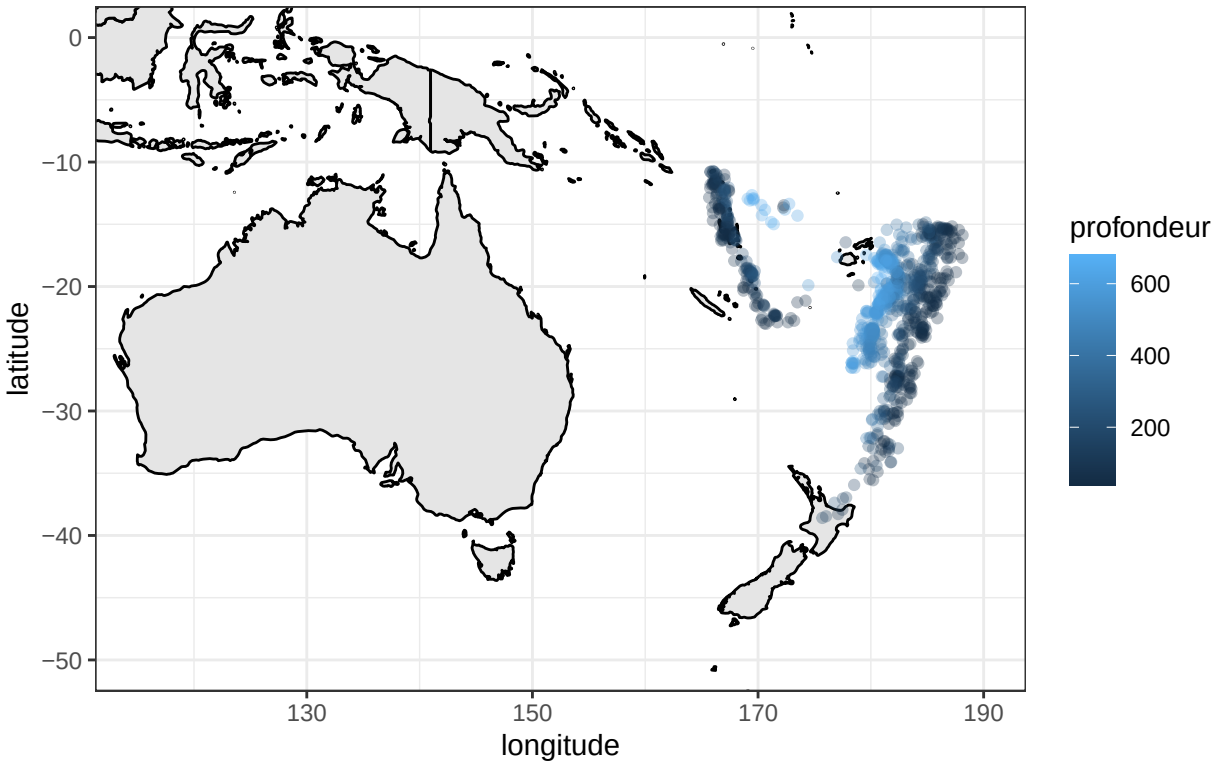
```
library(dplyr)

quakes %>%
  filter(region == "Ouest") %>%
  ggplot(mapping = aes(x = mag, y = stations)) +
  geom_jitter(colour = "blue", alpha = 0.3) +
  labs(
    title = "Séismes dans la région Ouest",
    x = "magnitude du séisme",
    y = "nombre de stations rapportant le séisme"
  )
```



3.9 Cartes géographiques

Comme avec le système graphique de base, le package maps permet d'ajouter une carte en arrière-plan d'un graphique ggplot. Voici un exemple d'utilisation d'une carte tirée du package maps avec ggplot2 :

```
library(maps)
monde <- map_data("world")  # va chercher des données provenant du package maps
ggplot() +
  geom_polygon(
    data = monde,
    aes(x = long, y = lat, group = group),
    fill = "gray90",
    col = "black"
  ) +
  geom_point(
    data = quakes,
    aes(x = long, y = lat, colour = depth),
    alpha = .3
  ) +
  coord_quickmap(           # ou coord_map() si le package mapproj est installé
    xlim = c(115, 190),
    ylim = c(-50, 0)
  ) +
  labs(x = "longitude", y = "latitude", colour = "profondeur") +
  theme_bw()
```

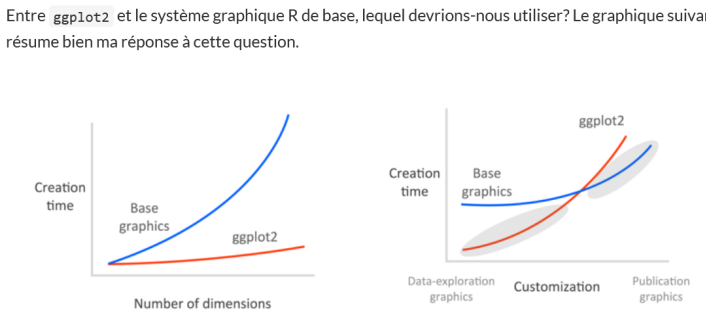


NB:L'extension ggmap permet également d'intégrer une carte Google Maps (requière un compte sur https://cloud.google.com/maps-platform/ pour avoir accès aux cartes) ou Stamen Maps à un graphique produit avec ggplot2.

4 Comparaison entre ggplot2 et le système graphique R de base

Entre ggplot2 et le système graphique R de base, lequel devrions-nous utiliser? Le graphique suivant résume bien ma réponse à cette question.

Figure 4: comparaison



Source : http://seananderson.ca/ggplot2-FISH554/

Lorsque le nombre de variables à représenter dans un graphique est grand (Number of dimensions élevé), utiliser ggplot2 peut potentiellement permettre de sauver beaucoup de temps. Les représentations multivariées sont plus faciles à produire avec ggplot2 qu'avec le système graphique R de base. En grande partie pour

cette raison, la production de graphiques pour de l'analyse exploratoire de données est souvent plus rapide à réaliser avec ggplot2 qu'avec le système graphique de base. Malgré tout, il existe encore une situation dans laquelle le système de base surpasse ggplot2 : la production de graphiques à mises en formes spécifiques pour des publications scientifiques. Les configurations graphiques sont souvent plus simples à réaliser avec le système de base qu'avec ggplot2.

Conclusion

Il ressort que, la bibliothèque ggplot2 est un outil puissant pour la visualisation de données en R. Elle offre une grande variété de graphiques et une flexibilité dans la personnalisation de ces graphiques. Avec ggplot2, vous pouvez facilement créer des graphiques attrayants et informatifs à partir de données brutes.

Référence

- Page web du package ggplot2 sur le CRAN : https://CRAN.R-project.org/package=ggplot2
- Documentation du package ggplot2 : http://ggplot2.tidyverse.org/
- Livres :

  1-Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. 2e édition. Springer.(URL pour la 3e édition en développement : https://ggplot2-book.org)

  2-Wickham, H. et Grolemund, G. (2016). R for Data Science. O'Reilly Media, Inc. Chapitre 3.(URL https://r4ds.had.co.nz/data-visualisation.html)

  3-Wilkinson, L. (2005). The grammar of graphics, 2e édition. Springer.

  4-Chang, W. (2012). R Graphics Cookbook: Practical Recipes for Visualizing Data. O'Reilly Media, Inc.(URL pour le code R : http://www.cookbook-r.com/Graphs/)