Séance 2 : Représentations graphiques sous R avec ggplot2

Laurent Tournier

20 septembre 2022

L'objectif de cette séance est de présenter et illustrer les vastes possibilités de visualisation de données (ce qui suppose parfois implicitement des calculs) dans R, à l'aide de l'extension `ggplot2`, qui offre de plus nombreuses options que les fonctions de base, mais surtout les présente au sein d'une "grammaire" cohérente qui la rend d'usage pratique, et force à réfléchir au sens des représentations graphiques choisies.

Cela sera aussi l'occasion de mettre en pratique les manipulations de structures de données en R vues la semaine dernière.

```
install.packages(ggplot2)
```

```
library(ggplot2)
```

1 Principe général

Le principe de cette grammaire est qu'un graphique est composé de couches superposées :

- les *données* à représenter, à partir desquelles nous définissons des *attributs esthétiques* (soit identiques pour toutes les observations, soit fonctions d'une des variables) :
  - les coordonnées x et y,
  - la couleur
  - la taille
  - le symbole
- la *géométrie* de la représentation (points, lignes, barres,...)
- les *transformations statistiques* opérées en amont de la représentation (dénombrement, ajustement,...)
- les *échelles* de chaque attribut
- le *système de coordonnées* (linéaire, logarithmique, polaire,...)
- le découpage (ou non) en *facettes* (sous-graphes)
- l'*allure générale* du graphe (encadrement, titre, etc.)

La fonction `ggplot()` crée un graphique, et le renvoie : on peut stocker un graphique dans une variable pour l'afficher plus tard ou le retravailler.

On lui *ajoute* littéralement des couches : `ggplot()+geom_point()+title()` par exemple (sans paramètre, le graphe reste vide). Les paramètres peuvent s'appliquer aux couches ou à `ggplot`, auquel cas les valeurs sont transmises dans toutes les couches. Les principaux sont

- `data = ...` : le dataframe utilisé
- `mapping = aes(x=..., y=..., ...)` : le lien entre les aspects esthétiques (x et y principalement mais aussi color, fill, size,...) et les variables du jeu de données

Les couches peuvent prendre les formes suivantes :

- `geom_xxx()` indique la représentation à choisir (xxx étant remplacé par histogram, boxplot, ...)
- `stat_xxx` indique les transformations statistiques à utiliser, si besoin
- `scale_xxx` s'emploie pour des changements d'échelle
- `coord_xxx` s'utilise pour des modifications de systèmes de coordonnées

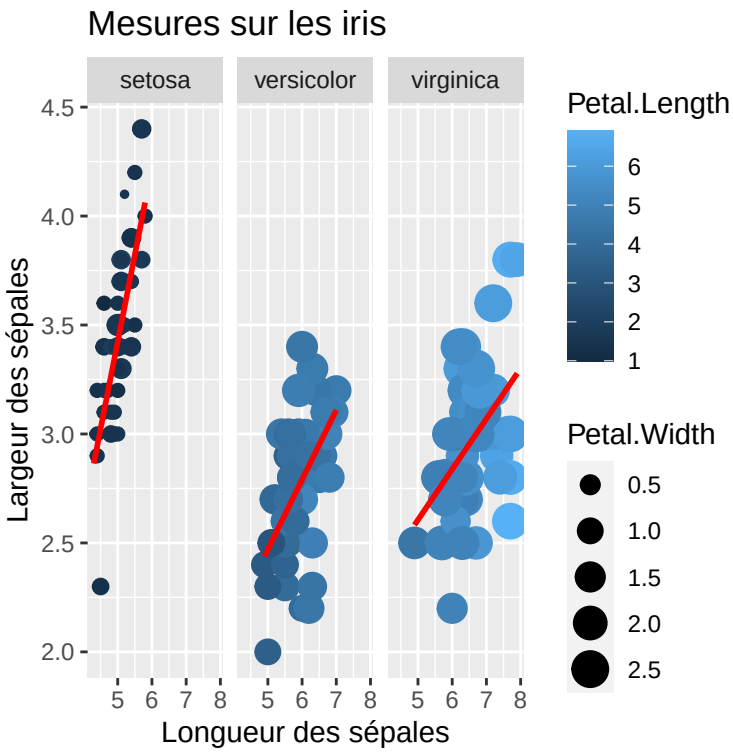
— `facet_grid()` découpe les données (et donc le graphique) en plusieurs facettes selon les variables fournie dans la formule
— `theme_xxx`, `labs()`, `xlab()`, `ylab()`, `ggtitle()`, ... pour des améliorations du graphique (annotation, couleurs, ...)

https://ggplot2-book.org/

Pour sauvegarder le graphe dans un fichier : `ggsave("plot.png",width=5, height=3)`

```
ggplot(data = iris,aes(x=Sepal.Length, y = Sepal.Width,
                       colour = Petal.Length,
                       size = Petal.Width)) +
  geom_point()+
  facet_wrap(~Species)+
  geom_smooth(method="lm",color="red",se=F,show.legend = F)+
  ggtitle("Mesures sur les iris")+
  xlab("Longueur des sépales")+
  ylab("Largeur des sépales")
```

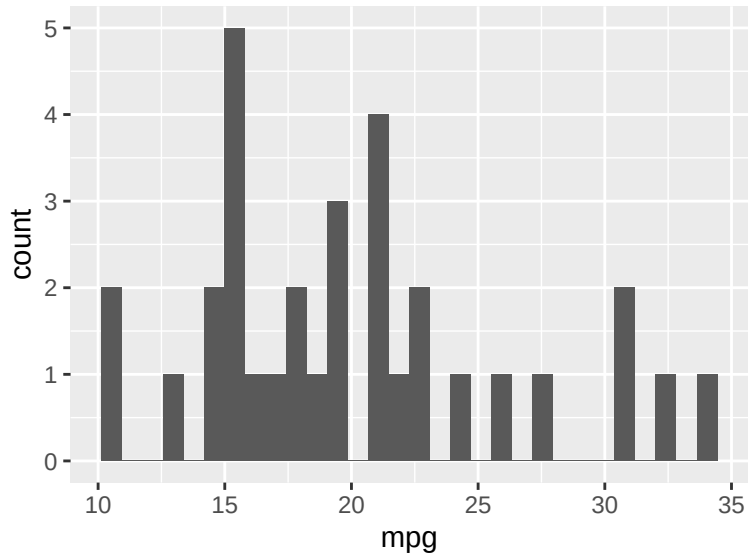
```
## `geom_smooth()` using formula 'y ~ x'
```



1.1 Héritage des paramètres

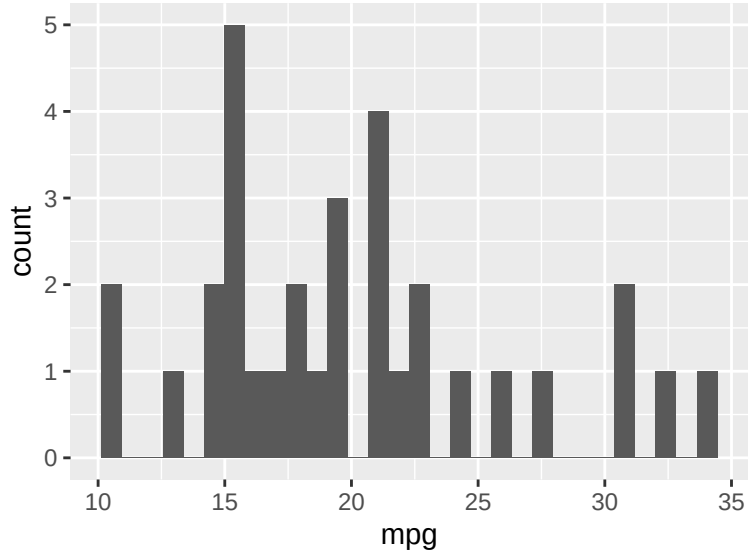
```
ggplot(mtcars, aes(x = mpg)) + geom_histogram()
```

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



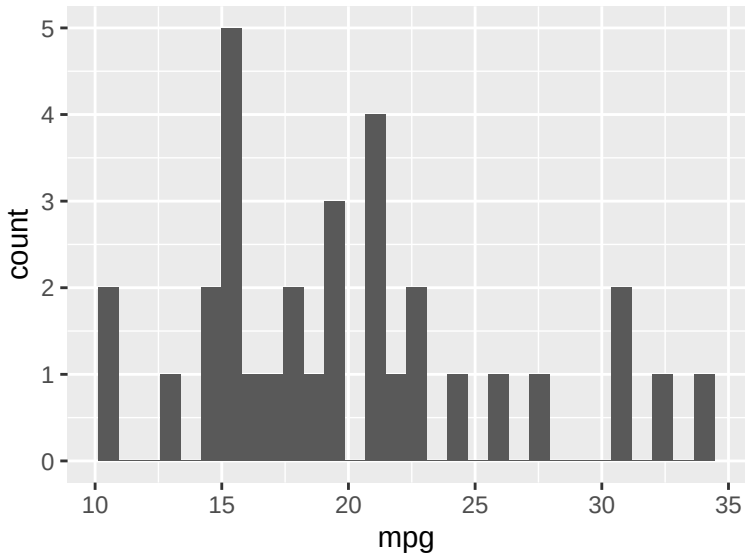

```
ggplot(mtcars) + geom_histogram(aes(x = mpg))
```

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




```
ggplot() + geom_histogram(data = mtcars, aes(x = mpg))
```

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



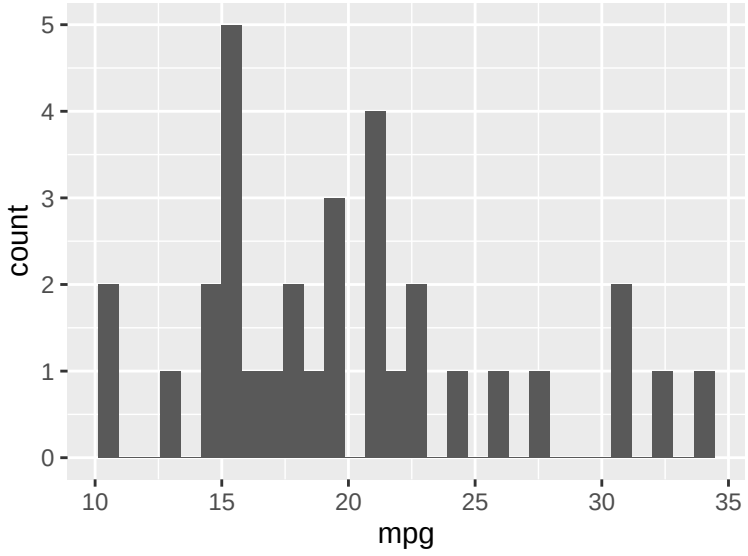
2 Une variable quantitative

Histogrammes

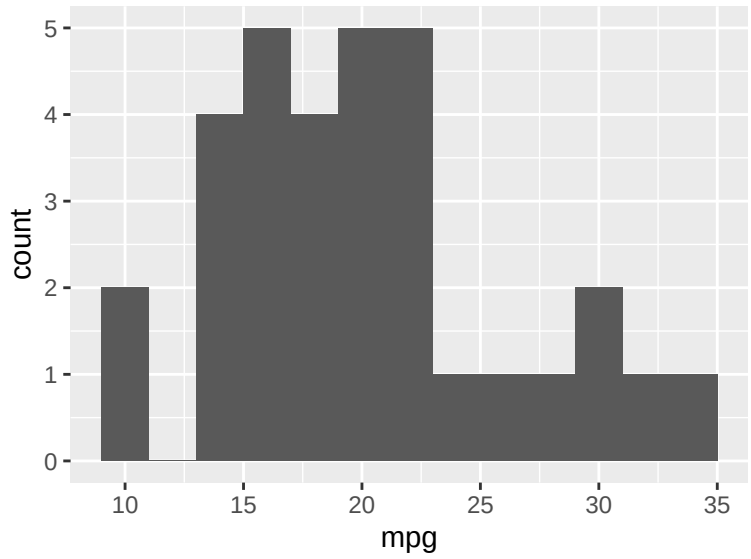
NB. Donne par défaut le *nombre* d'observations dans chaque corbeille.

```
ggplot(mtcars, aes(x = mpg)) + geom_histogram()
```

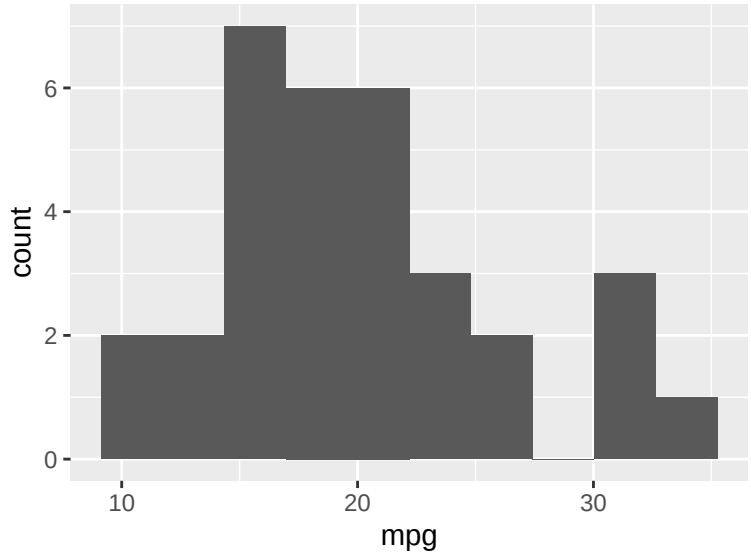
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



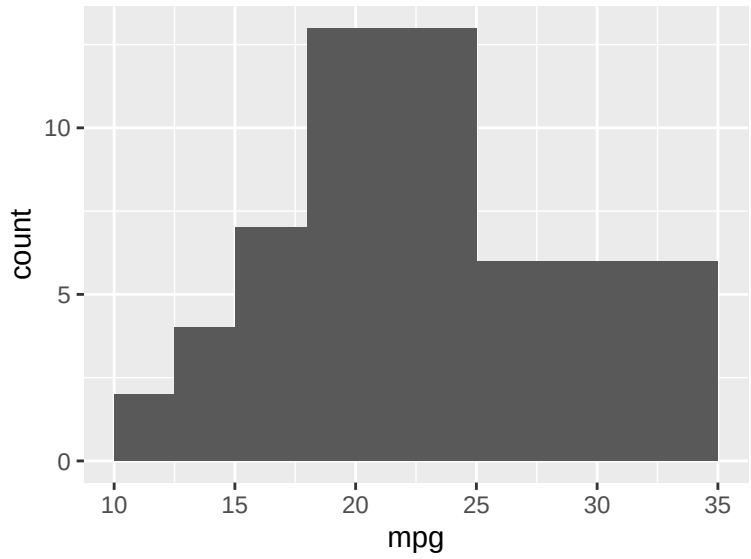

```
ggplot(mtcars, aes(x = mpg)) + geom_histogram(binwidth = 2)
```



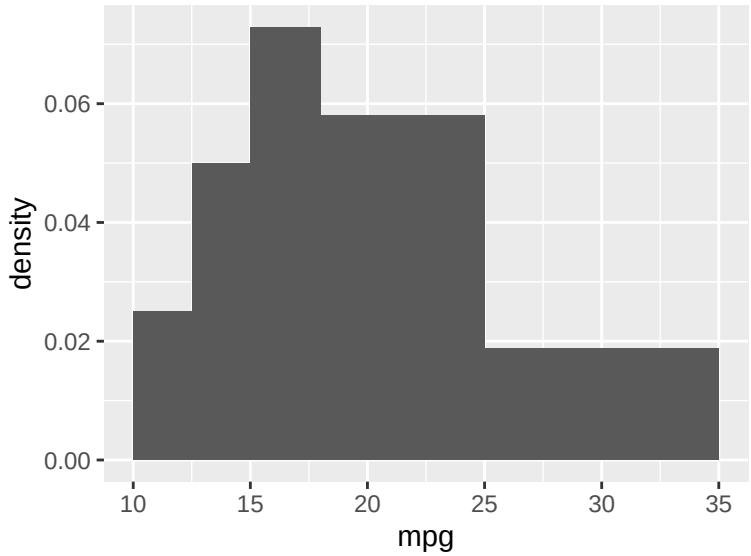

```
ggplot(mtcars, aes(x = mpg)) + geom_histogram(bins = 10)
```




```
ggplot(mtcars, aes(x = mpg)) + geom_histogram(breaks = c(10,12.5,15,18,25,35))
```




```
ggplot(mtcars, aes(x = mpg)) +
    geom_histogram(aes(y = ..density..), breaks = c(10,12.5,15,18,25,35))
```




```
ggplot(mtcars, aes(x = mpg)) +
    geom_histogram(aes(y = ..density..), binwidth = 2) +
    geom_density()
```

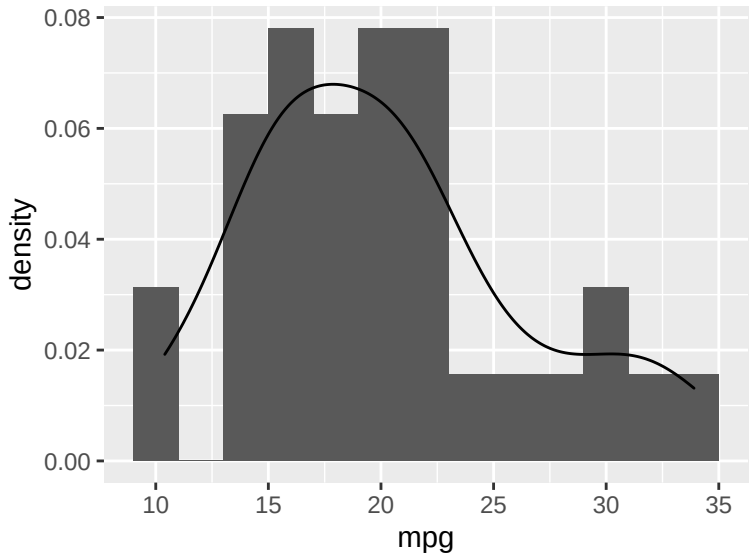
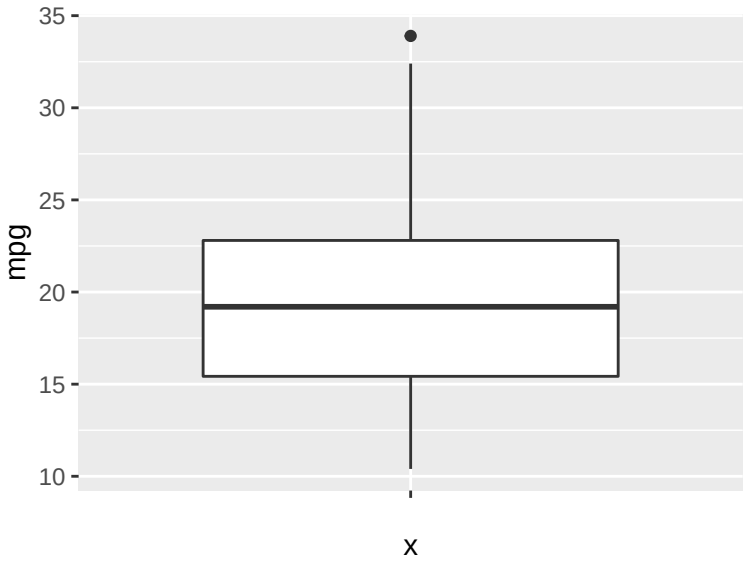
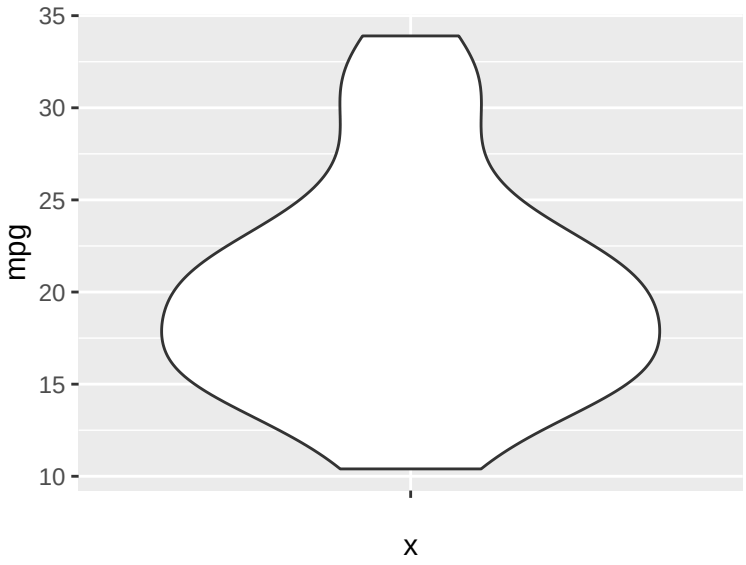


Diagramme en boîte

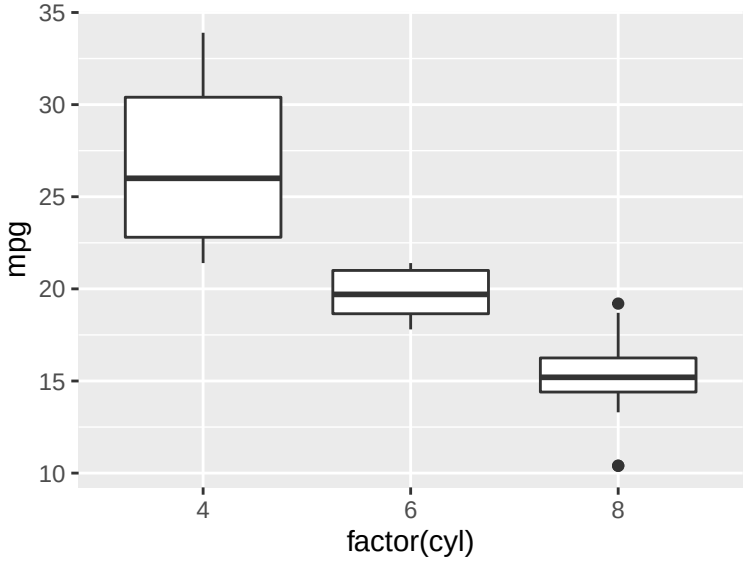
```
ggplot(mtcars, aes(y = mpg, x = "")) + geom_boxplot()
```



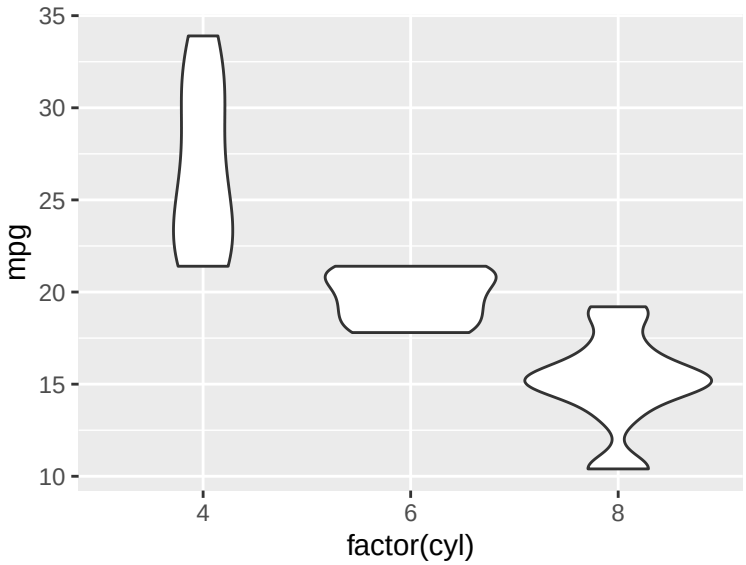

```
ggplot(mtcars,aes(y=mpg,x=""))+geom_violin()
```




```
ggplot(mtcars, aes(y = mpg, x = factor(cyl))) + geom_boxplot()
```



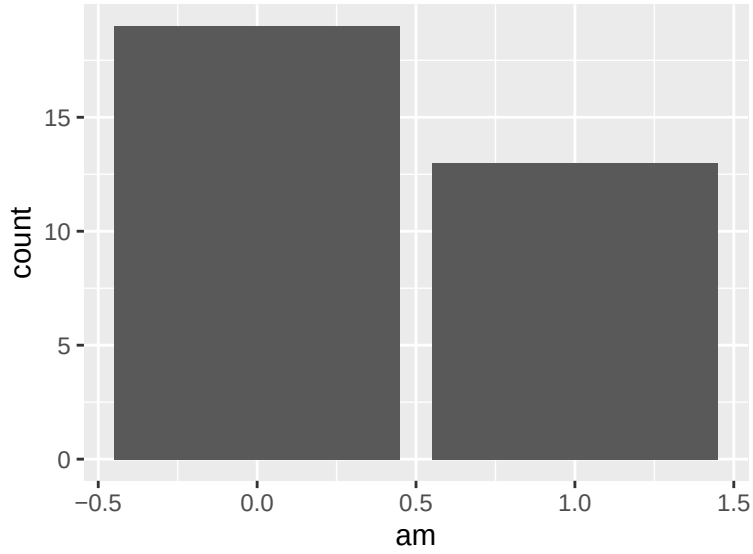

```
ggplot(mtcars, aes(y = mpg, x = factor(cyl))) + geom_violin()
```



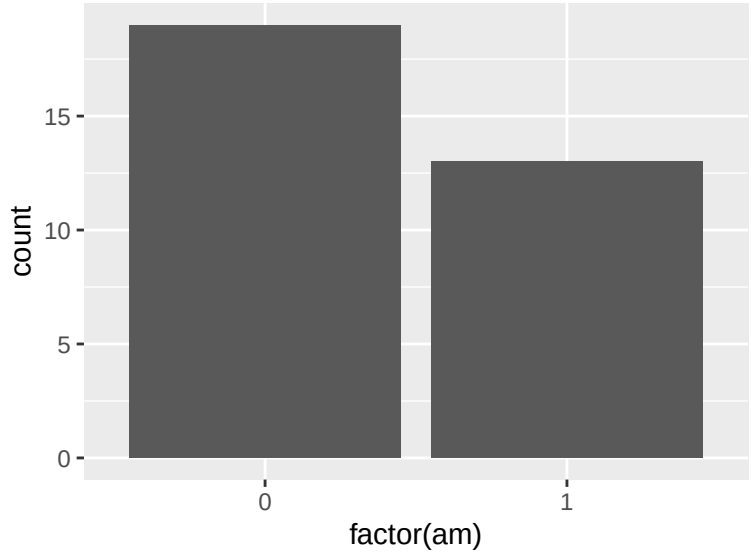
3 Une variable qualitative

Histogramme, diagramme en barres

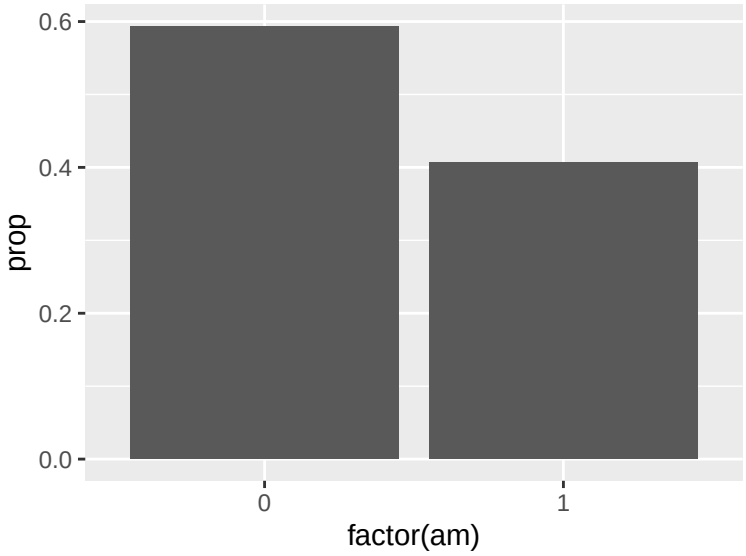
```
ggplot(mtcars, aes(x = am)) + geom_bar()
```



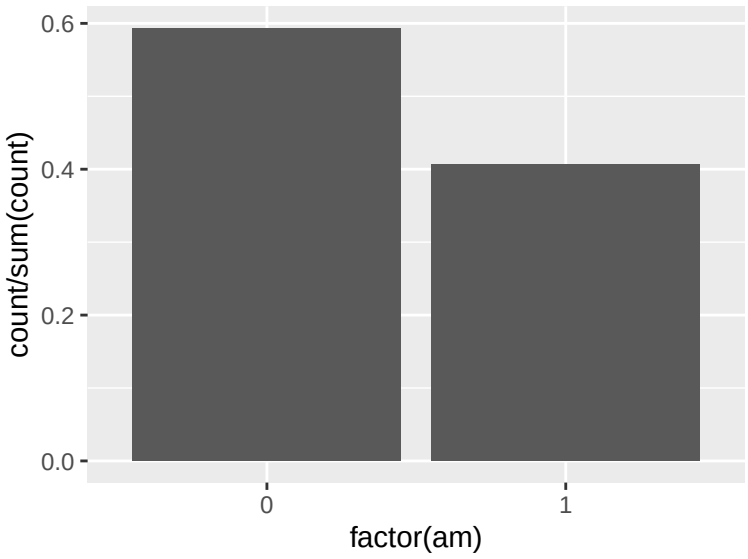

```
ggplot(mtcars, aes(x = factor(am))) + geom_bar()
```



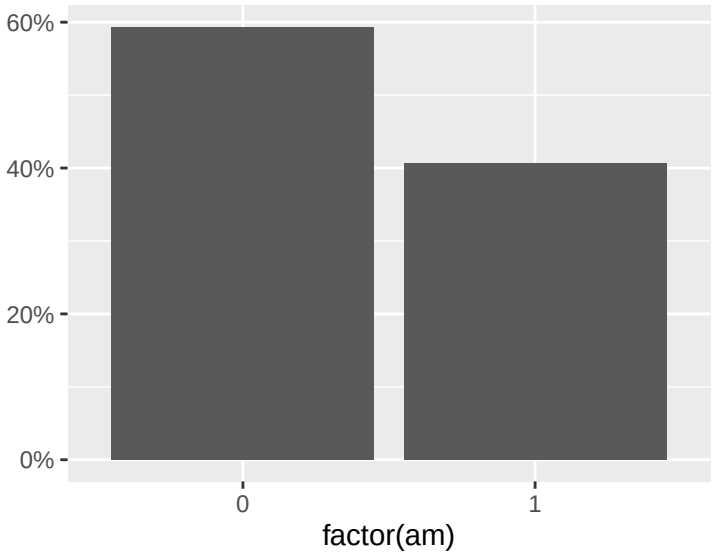

```
ggplot(mtcars, aes(x = factor(am))) + geom_bar(aes(y=stat(prop),group=1))
```




```
ggplot(mtcars, aes(x = factor(am))) + geom_bar(aes(y=stat(count)/sum(stat(count))))
```



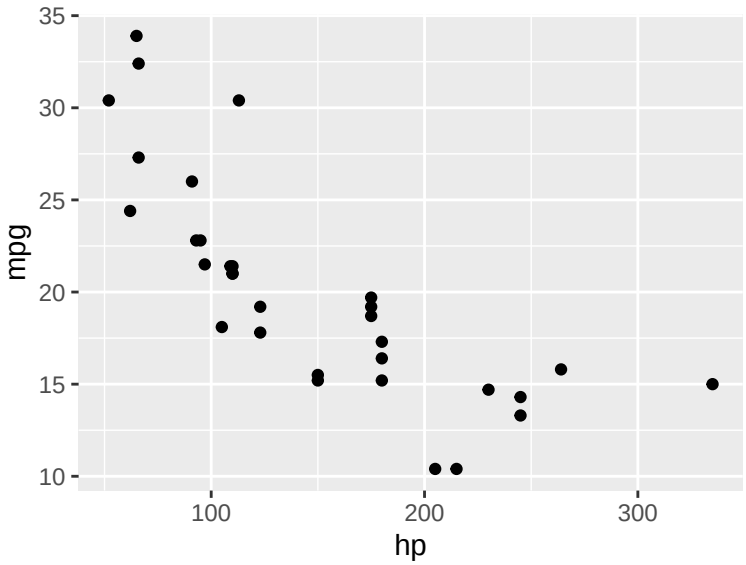

```
ggplot(mtcars, aes(x = factor(am))) +
    geom_bar(aes(y = (..count..)/sum(..count..))) +
    scale_y_continuous(labels = scales::percent) +
    ylab("")
```



4 Deux variables quantitatives

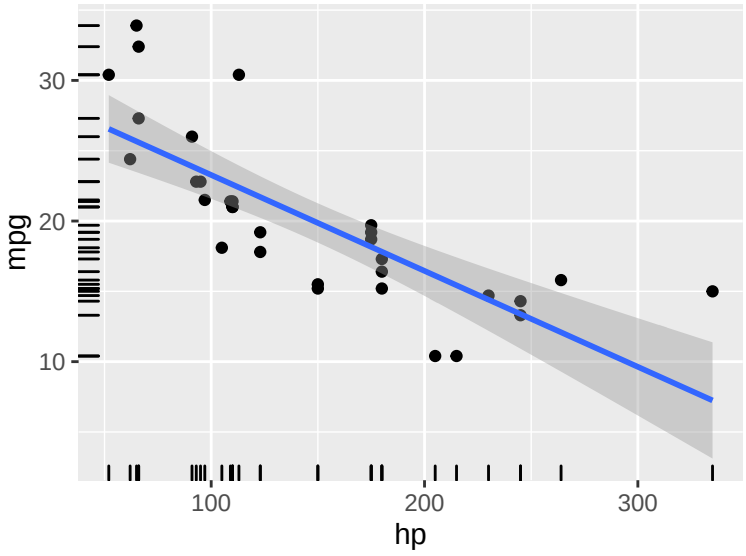
Nuage de points

```
ggplot(mtcars, aes(hp, mpg)) + geom_point()
```



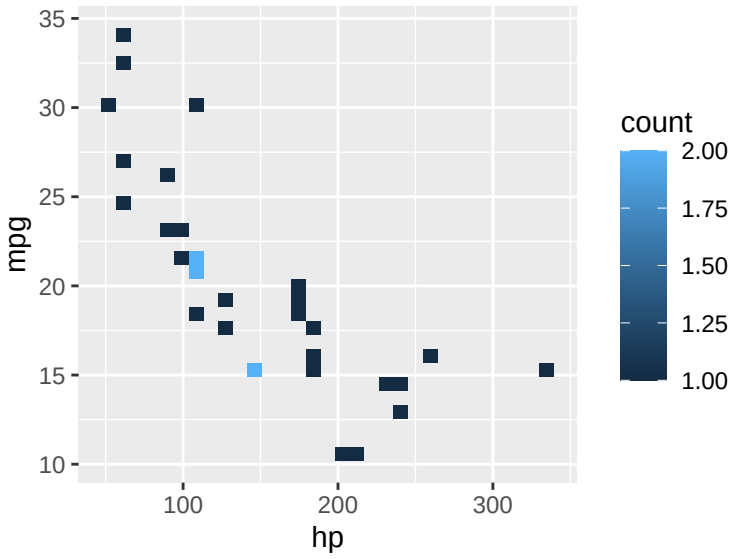

```
ggplot(mtcars, aes(hp, mpg)) + geom_point() +
    geom_rug() +
    geom_smooth(method = "lm")
```

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



Histogramme 2d : carte de chaleur

```
ggplot(mtcars, aes(hp, mpg)) + geom_bin2d()
```

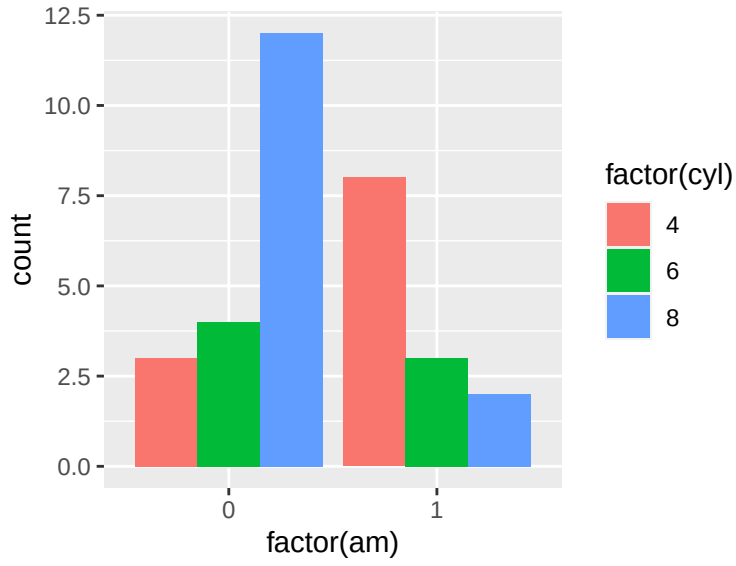


5 Deux variables qualitatives

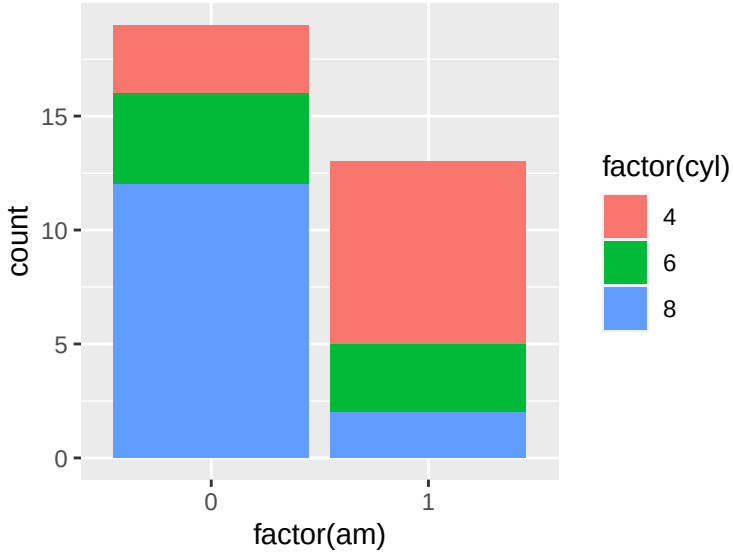
Diagramme en barres empilées/juxtaposées

Principe : assigner la couleur de remplissage (`fill`) selon une variable qualitative.

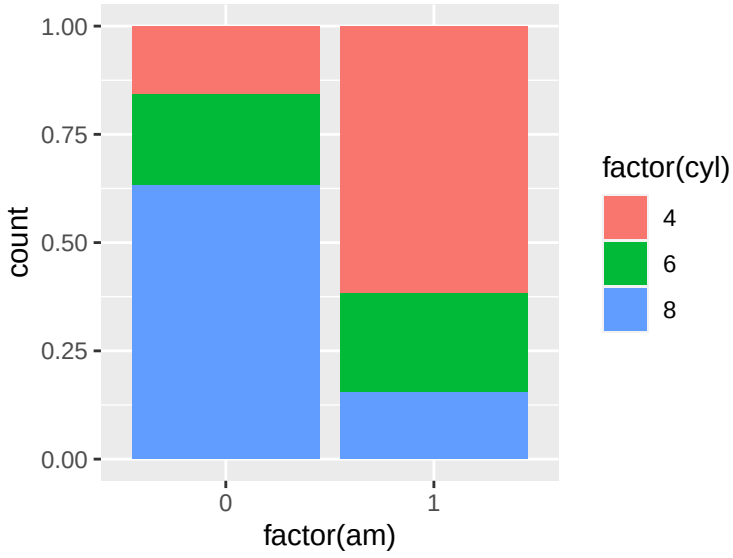
```
ggplot(mtcars, aes(x=factor(am), fill = factor(cyl))) + geom_bar(position="dodge")
```



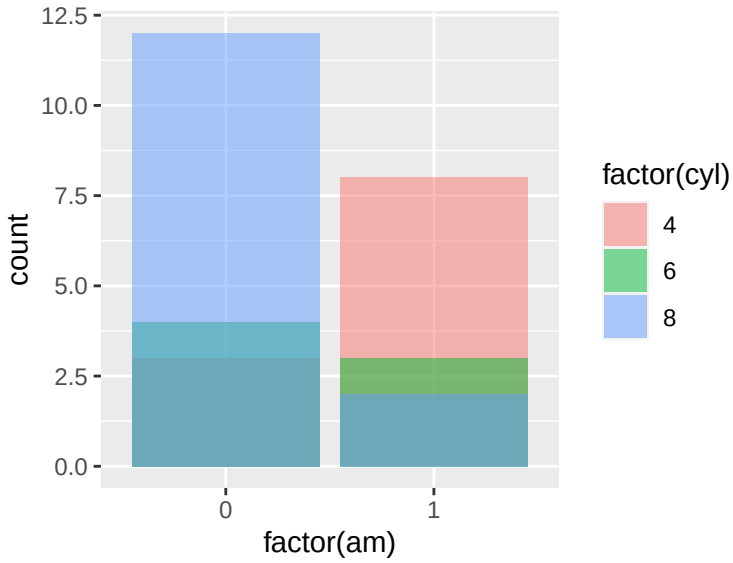

```
ggplot(mtcars, aes(x=factor(am), fill = factor(cyl))) + geom_bar(position="stack")
```




```
ggplot(mtcars, aes(x=factor(am), fill = factor(cyl))) + geom_bar(position="fill")
```




```
ggplot(mtcars, aes(x=factor(am), fill = factor(cyl))) + geom_bar(position="identity",alpha=.5)
```

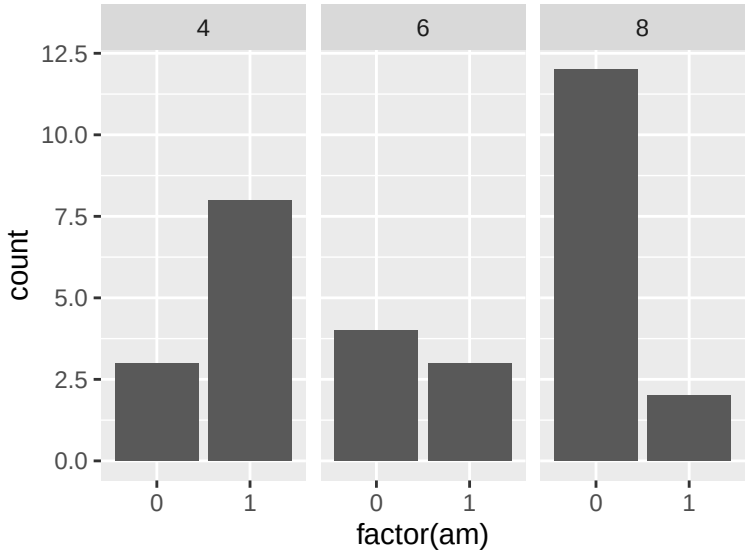


Découpage en plusieurs sous-graphes

Principe : ajouter la couche `facet_wrap(~variable)` (ou `~variable1+variable2` pour décomposer selon plusieurs variables qualitatives)

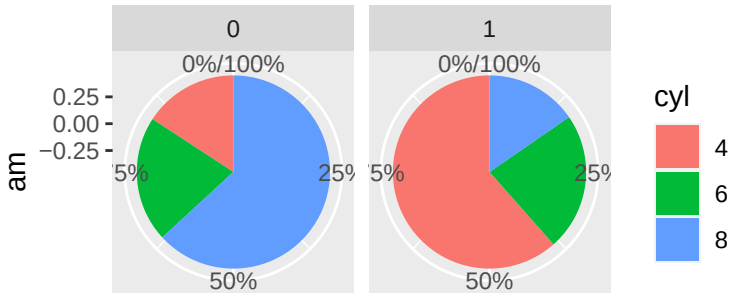
Cela vaut aussi pour une variable qualitative et une variable quantitative.

```
ggplot(mtcars, aes(factor(am))) + geom_bar() + facet_wrap( ~ cyl)
```



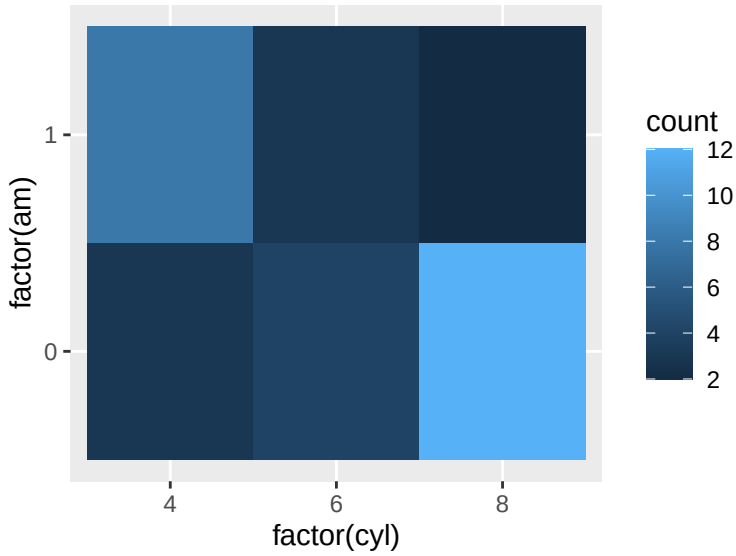
(on peut préciser `nrow=2` pour mettre sur deux lignes, etc.)

```
ggplot(mtcars, aes(0, fill = factor(cyl))) +
    geom_bar(position = "fill") +
    scale_y_continuous(labels = scales::percent) +
    xlab("am") + ylab("") + labs(fill = "cyl") +
    coord_polar(theta = "y") +
    facet_wrap(~ factor(am))
```



Histogramme 2D

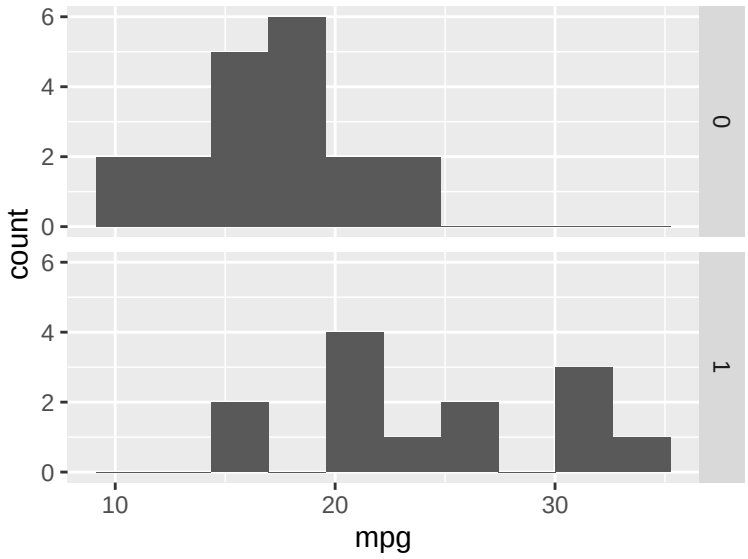
```
ggplot(mtcars, aes(factor(cyl), factor(am))) + geom_bin2d()
```



6 Variable quantitative et variable qualitative

Découpage en sous-graphes

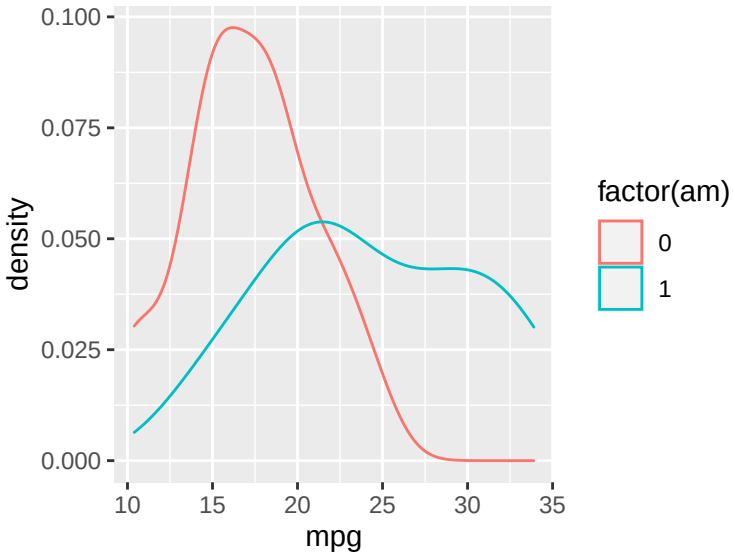
```
ggplot(mtcars, aes(mpg)) + geom_histogram(bins = 10) +
    facet_grid(am ~ .)
```



Superposition avec couleur différente

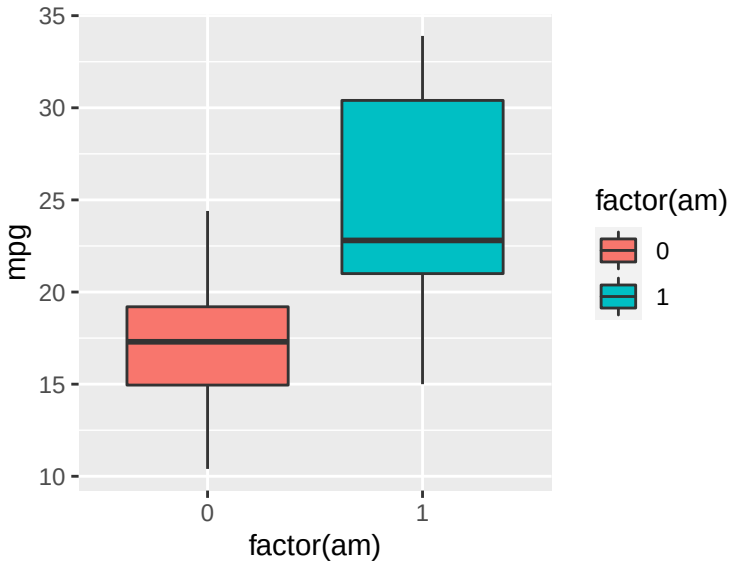
Principe : assigner à la couleur une variable qualitative.

```
ggplot(mtcars, aes(mpg, colour = factor(am))) +
    geom_density()
```



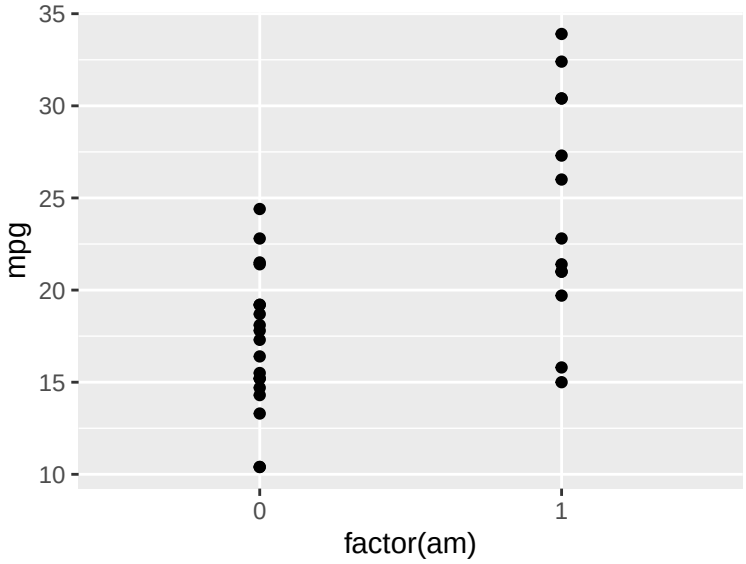
Boîtes de dispersion juxtaposées

```
ggplot(mtcars, aes(factor(am), mpg, fill = factor(am))) +
    geom_boxplot()
```

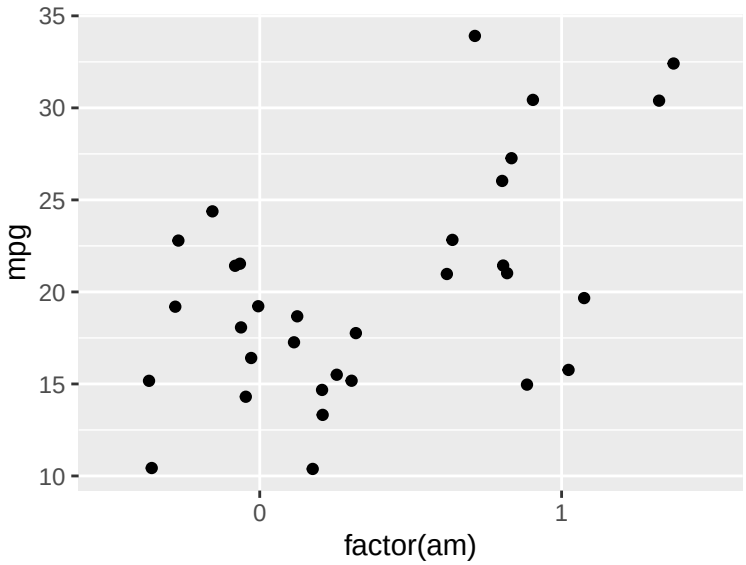


Nuage de points

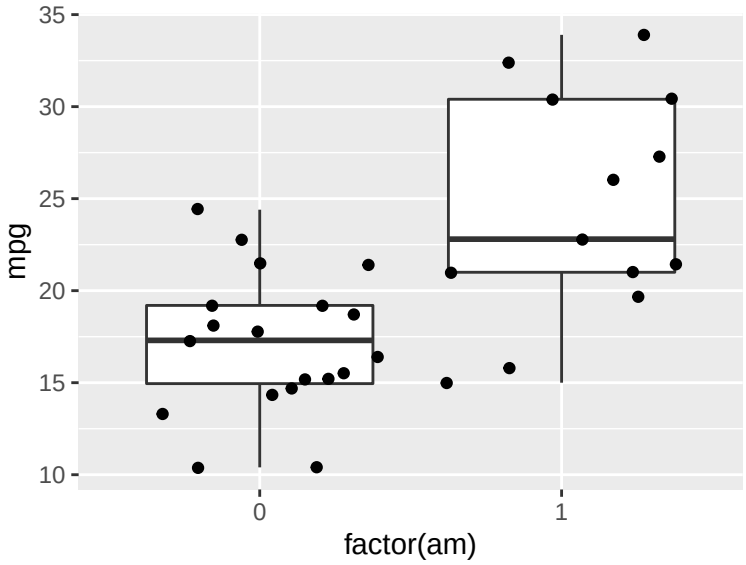
```
ggplot(mtcars, aes(factor(am), mpg)) +
    geom_point()
```




```
ggplot(mtcars, aes(factor(am), mpg)) +
    geom_jitter()
```



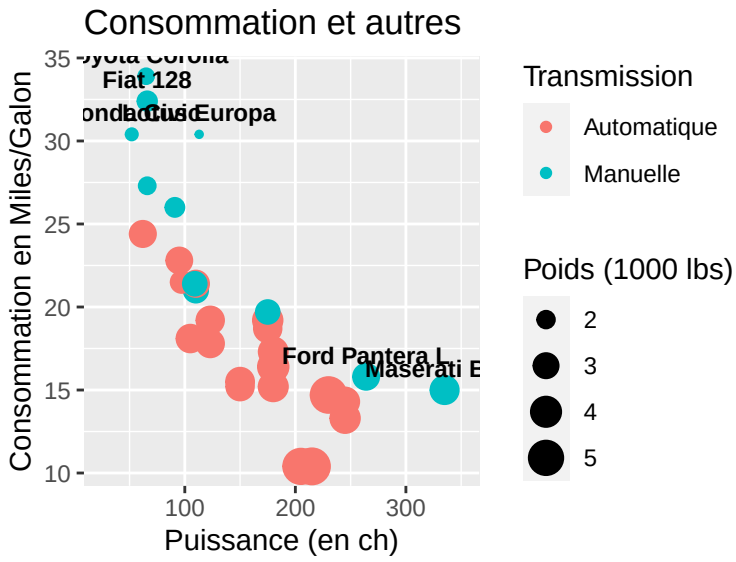

```
ggplot(mtcars, aes(factor(am), mpg)) +
    geom_boxplot() +
    geom_jitter()
```



6.0.1 Exemple plus sophistiqué

```
nom = rownames(mtcars)
nom[mtcars$hp <= 250 & mtcars$mpg <= 30] = ""
ggplot(mtcars, aes(x = hp, y = mpg,
                   color = factor(am, labels = c("Automatique", "Manuelle")),
                   size = wt,
                   label = nom)) +
    geom_point() +
    geom_text(size = 3, color = "black", vjust = -.75, fontface = "bold") +
    ggtitle("Consommation et autres") +
    xlab("Puissance (en ch)") + xlim(25, 350) +
    ylab("Consommation en Miles/Galon") +
```

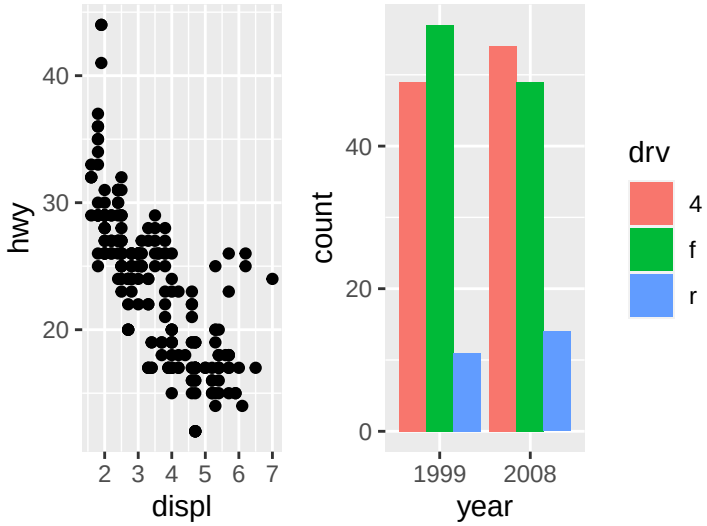
```
    labs(color = "Transmission", size = "Poids (1000 lbs)")
```



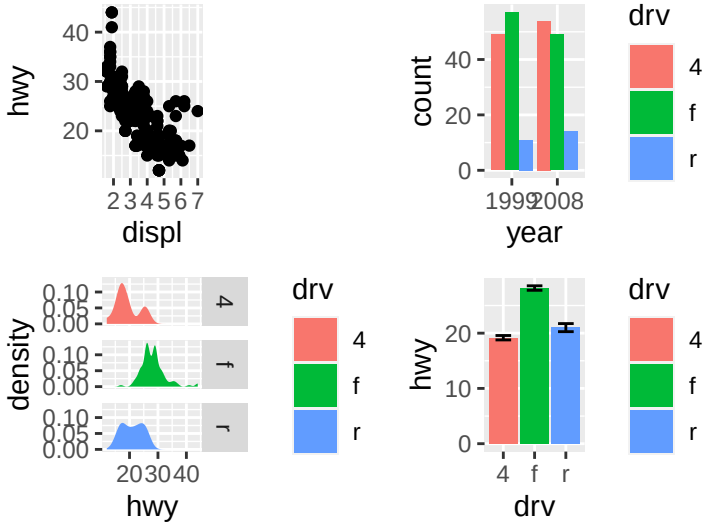
Combiner plusieurs graphes

Le package `patchwork` permet d'associer des graphes obtenus par `ggplot`, de façon cohérente avec la grammaire précédente. Par exemple,

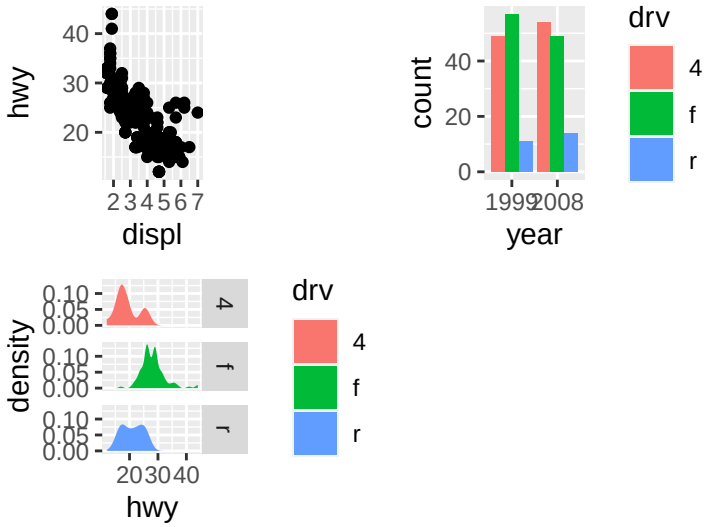
```
library(patchwork)

p1 <- ggplot(mpg) +
  geom_point(aes(x = displ, y = hwy))

p2 <- ggplot(mpg) +
  geom_bar(aes(x = as.character(year), fill = drv), position = "dodge") +
  labs(x = "year")

p3 <- ggplot(mpg) +
  geom_density(aes(x = hwy, fill = drv), colour = NA) +
  facet_grid(rows = vars(drv))

p4 <- ggplot(mpg) +
  stat_summary(aes(x = drv, y = hwy, fill = drv), geom = "col", fun.data = mean_se) +
  stat_summary(aes(x = drv, y = hwy), geom = "errorbar", fun.data = mean_se, width = 0.5)

p1+p2
```



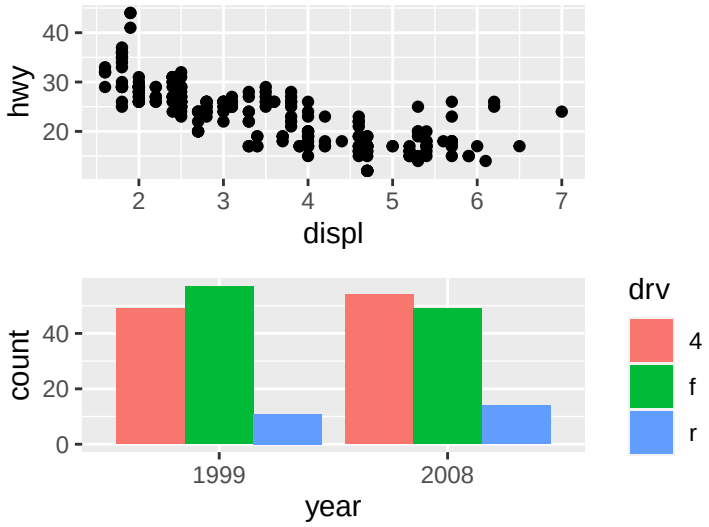

```
p1+p2+p3+p4
```



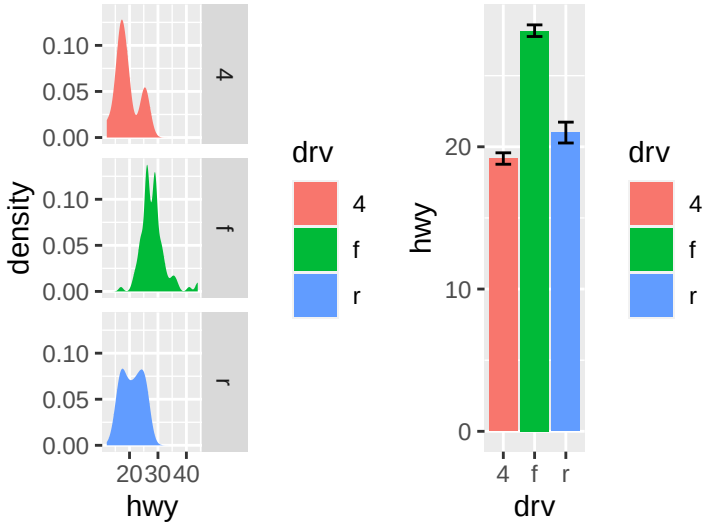

```
p1 + p2 + p3 + plot_layout(ncol = 2)
```



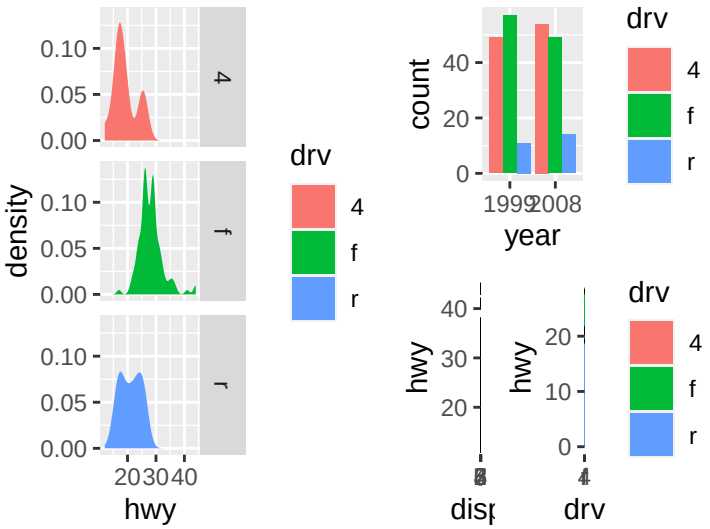
```
p1/p2
```




```
p3|p4
```




```
p3 | (p2 / (p1 | p4))
```



cf. https://ggplot2-book.org/arranging-plots.html

7 Exercices

Attention à ne pas oublier la commande `library(ggplot2)` (sans guillemets) au début du fichier.

7.1 Exercice

Le fichier de données `tips.csv` (page moodle) fait la liste des pourboires (*tips*) reçus par un serveur sur une période de quelques mois dans un restaurant. Il a noté

- `tip` le montant du pourboire
- `total_bill` le montant de l'addition
- `sex` le sexe du client payant l'addition
- `smoker` s'il y a des fumeurs dans le groupe
- `day` le jour de la semaine
- `time` le repas concerné
- `size` le nombre de convives.

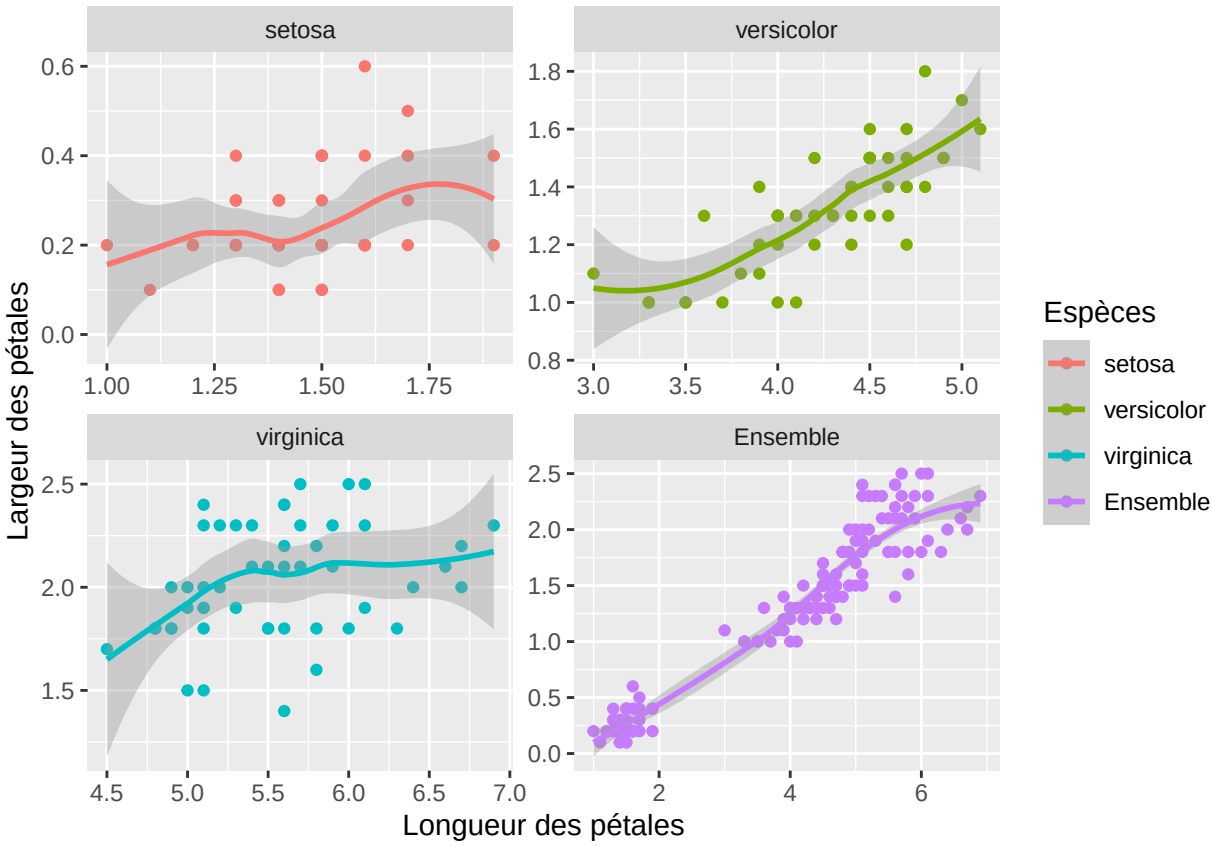
1. Charger le fichier dans un dataframe `tips`.
2. Représenter le montant de l'addition en fonction du sexe du payeur. *Proposer deux représentations.*
3. Représenter `total_bill` en fonction de `sex` et `smoker`. *Adapter chacune des deux représentations précédentes.*
4. Comment répondre aux questions suivantes : (on ne fera pas de test, on cherche juste une première intuition graphique)

- Les pourboires (tip) dépendent-ils du montant (total_bill)
- Et des jours de la semaine (day) ?
- Et du nombre de convives (size) ?

7.2 Exercice

Produire le graphe suivant, à partir des données `iris`.

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Indications :

— La courbe s'obtient par `geom_smooth` sans paramètre (ou `method=loess`)
— Pour le titre de la légende, utiliser `+scale_color_discrete("Espèces")`.
— Pour obtenir la case "Ensemble", modifier le dataframe...
— Pour avoir une échelle différente pour chaque graphe, utiliser le paramètre `scales` de `facet_wrap`.

7.3 Exercice. Histogramme des tailles

On va simuler des variables puis les représenter graphiquement.

1. Créer un dataframe `toise` possédant les variables "Taille" et "Sexe", et comportant les tailles de 200 hommes, supposées de loi normale de moyenne 175cm et d'écart-type 6cm et les tailles de 200 femmes, supposées de loi normale de moyenne 163cm et d'écart-type 5,5cm.
2. Représenter un histogramme de la taille dans la population totale. (Prendre 1cm comme largeur de corbeille)
3. Représenter sur un même graphique un histogramme de la taille des hommes et des femmes, en s'assurant de la lisibilité.
4. Représenter sur un même graphe des estimations de la densité de probabilité de la taille des hommes, des femmes, et de la population totale.
5. Ajouter des barres verticales au niveau des moyennes par sexe, et globale. *On utilisera `geom_vline(xintercept=)`.*

7.4 Exercice. Comparaison de densité théorique et simulée

On rappelle la méthode suivante pour simuler une variable aléatoire de loi exponentielle de paramètre λ : étant donnée une v.a. U de loi uniforme sur $[0,1]$, on pose $X = -\log(U)/\lambda$. Alors X suit la loi $\mathcal{E}(\lambda)$.

1. Rappeler pourquoi.

2. Poser $N = 10000$ et définir un vecteur `tirages` comportant N tirages $X_1, X_2, \ldots, X_N$ indépendants et de loi exponentielle de paramètre 1. On pourra le convertir en dataframe avec `df.tirages<-as.data.frame(tirages)`.
3. Représenter un histogramme des résultats et une estimation de la densité par méthode à noyau (`geom_density`).
4. Sur le même graphe, représenter la densité de la loi $\mathcal{E}(1)$ pour comparaison. *On pourra utiliser `stat_function(fun=f)` où `f` est le nom d'une fonction ou une définition de fonction "en ligne" : `function(x) x^2` par exemple.*
5. Comparer également la fonction de répartition empirique (`stat_ecdf()`) et la fonction de répartition théorique.

7.5 Exercice. Données 2D

1. Tester la géométrie `geom_density2d`sur 10000 données simulées de même loi que

$$\xi \cdot (X, X + 2Y) + (1 - \xi) \cdot (3 + X, X + Y)$$

où X et Y sont indépendantes, de loi gaussienne standard, et ξ suit une loi de Bernoulli de paramètre $0,4$.