Une très courte introduction à GGPlot2

Vincent Jalby, Université de Limoges

Janvier 2025

Table des matières

1 Introduction 1

2 Histogramme 2

3 Boite à moustaches 4

4 Nuage de points 5

5 Diagramme en bâtons 7

6 Diagramme en secteurs 12

7 Variables de Likert 15

8 Suite de graphiques 16

9 Formes et couleurs 17

10 Themes 18

1 Introduction

Le package GGPlot 2 offre une alternative, très utilisée, aux fonctions graphiques de base de R (`plot()`, `hist()`, etc). Mais son utilisation demande un apprentissage supplémentaire.

On commence par installer et activer l'extension `ggplot2`, complétée par `scales` et `ggstats` :

```
install.packages( c('ggplot2','scales','ggstats') )
library(ggplot2)
library(scales)
library(ggstats)
```

Dans la suite, on réutilisera les données "courses" définies précédemment :

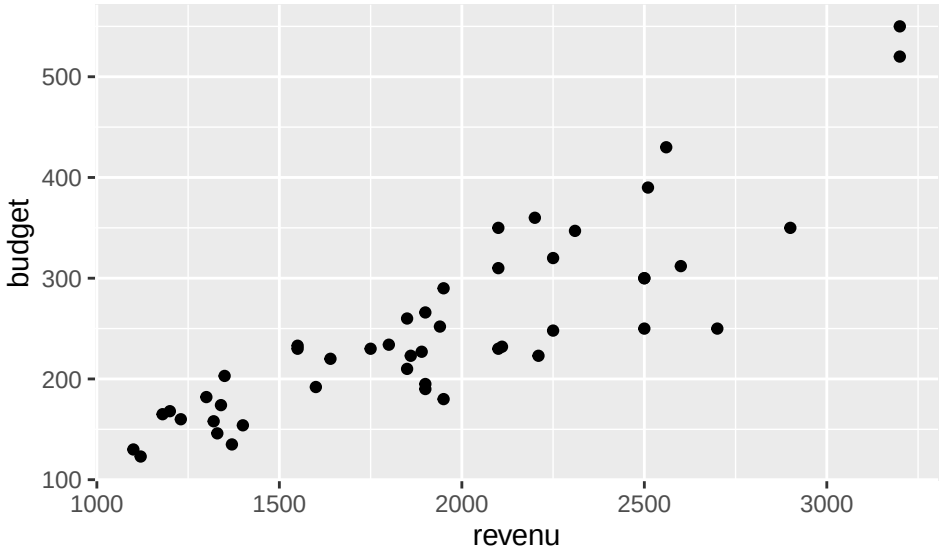
```
load("courses.RData")
```

La construction d'un graphique avec l'extension ggplot2 est basée sur une *grammaire des graphiques*. Chaque graphique est construit à partir des mêmes composants (fonctions), *additionnés* entre eux :

- la fonction **`ggplot`**`()` permet de préciser le `dataframe` utilisé et
- la fonction **`aes`**`()` précise les variables qui seront représentées
- un ensemble de fonctions **`geom`** permet de définir le type de graphique (nuage, barres, etc)
- de nombreuses autres fonctions permettent de changer les titres, axes, etc :
  - **`labs`**`()` pour modifier le titre et le nom des axes
  - **`coord_cartesian`**`()` pour modifier l'échelle des axes
  - **`scale_x_continuous`**`()` et **`scale_y_continuous`**`()` pour changer les graduations sur les axes
  - **`scale_x_discrete`**`()` pour changer l'échelle d'une variable qualitative
  - **`scale_colour_manual`**`()` et **`scale_fill_manual`**`()` pour indiquer les couleurs à utiliser
  - **`scale_colour_brewer`**`()` et **`scale_fill_brewer`**`()` pour utiliser des nuanciers de couleurs prédéfinies
  - **`theme_bw`**`()`, **`theme_gray`**`()`, **`theme_classic`**`()`, **`theme_minimal`**`()`, **`theme_dark`**`()`, **`theme_void`**`()`... pour changer l'apparence globale du graphique
  - **`facet_grid`**`()` et **`facet_wrap`**`()` pour reproduire le même graphique selon les valeurs d'une variable qualitative.

Voici un premier exemple

```
ggplot(courses, aes(x=revenu, y=budget) ) + geom_point( )
```



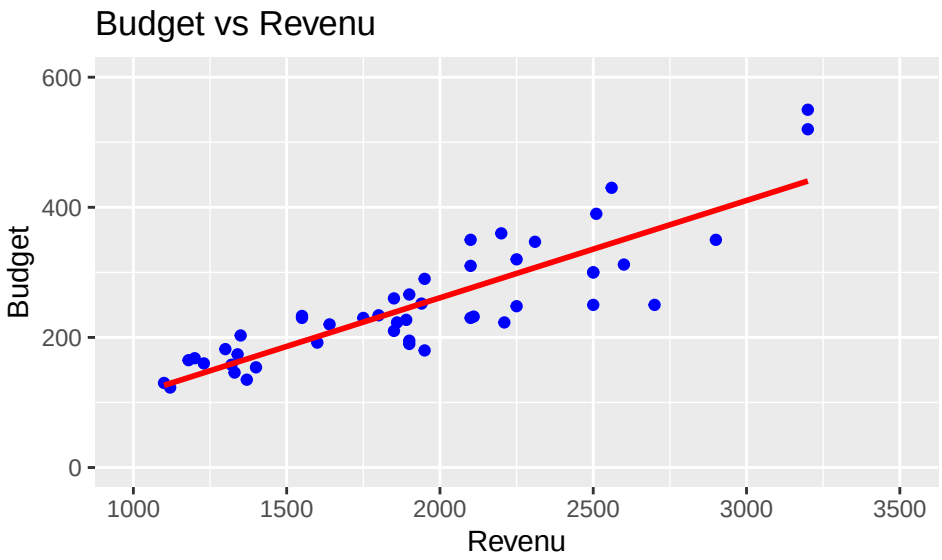
La fonction **ggplot**() indique le dataframe utilisé (course), et via l'instruction **aes**(), les variables représentées. La fonction **geom_point**() précise le type de graphique (nuage de points).

L'exemple suivant change les limites des axes (**coord_cartesian**()) ainsi que les étiquettes des axes et du graphique (**labs**()) et rajoute une droite d'ajustement (**geom_smooth**()).

Attention, de mettre le "+" à la fin de chaque ligne (et non au début!)

```
ggplot( courses, aes(x=revenu, y=budget) ) + geom_point(color="blue",na.rm=TRUE) +
        geom_smooth(method="lm", se=FALSE, na.rm=TRUE, color="red") +
        coord_cartesian( xlim=c(1000,3500), ylim=c(0,600) ) +
        labs( x= "Revenu", y = "Budget", title ="Budget vs Revenu" )

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

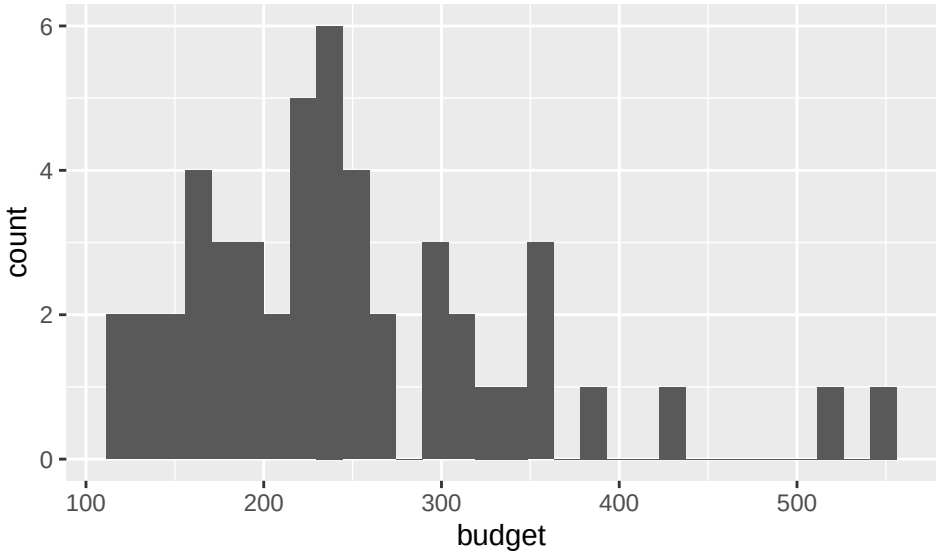


2 Histogramme

En utilisant le principe ci-dessus, **geom_histogram**() permet d'obtenir l'histogramme d'une variable quantitative :

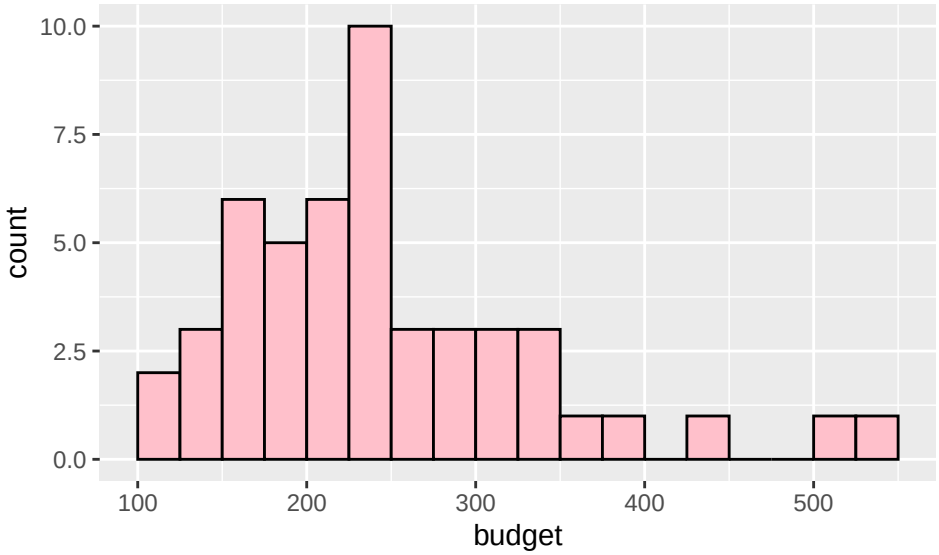
```
ggplot(courses, aes(x=budget)) + geom_histogram()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 1 row containing non-finite outside the scale range (`stat_bin()`).
```



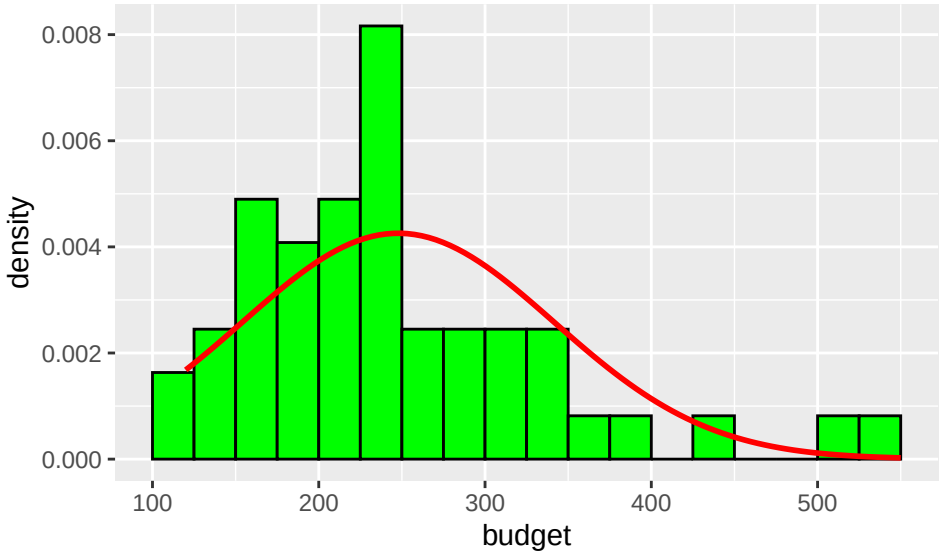
On peut préciser de nombreuses options, en particulier `na.rm=TRUE` pour ignorer les valeurs manquantes (et ne pas avoir de *warning*) et `binwidth` pour préciser la largeur des classes. `boundary` permet de préciser une valeur à partir de laquelle les classes seront calculées (ici 100, donc les classes seront [100,125[, [125, 150[, etc)

```
ggplot(courses, aes(x=budget)) +
  geom_histogram(binwidth=25, boundary=100, na.rm = TRUE, color="black", fill="pink")
```



La superposition de la densité d'une loi (normale) nécessite plus de travail. Tout d'abord, l'histogramme est tracé en fréquences (**`after_stat`**`(density)`) au lieu d'effectifs, puis la courbe de la densité est rajoutée via la fonction **`stat_function`**`()`.

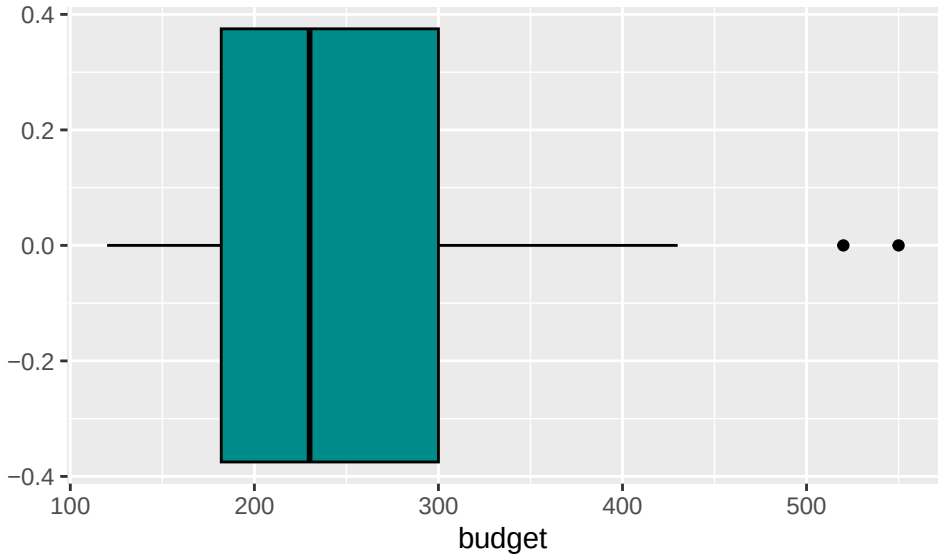
```
ggplot(courses, aes(x=budget)) +
  geom_histogram(aes(y = after_stat(density)), binwidth=25, boundary=0, color="black",
                                                fill="green", na.rm = TRUE) +
  stat_function(fun = dnorm,
                args = list(mean = mean(courses$budget, na.rm=TRUE),
                            sd = sd(courses$budget, na.rm=TRUE)),
                color = "red", linewidth = 1)
```



3 Boite à moustaches

Les boites à moustaches s'obtiennent avec la fonction **geom_boxplot**() de manière très similaire à l'histogramme.

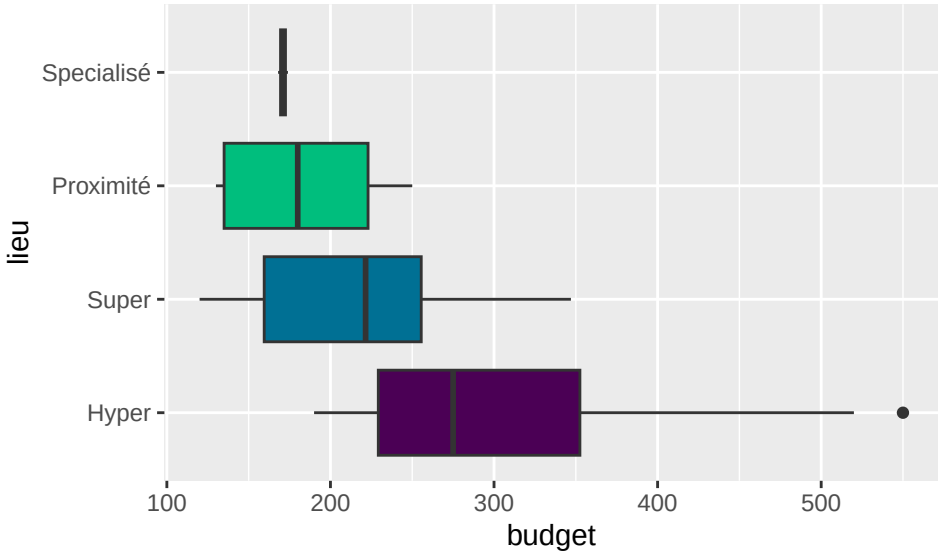
```
ggplot(courses, aes(x=budget)) + geom_boxplot(color="black", fill="cyan4", na.rm=TRUE)
```



Utiliser **aes**(y=budget) au lieu de **aes**(x=budget) pour une boite à moustaches verticale.

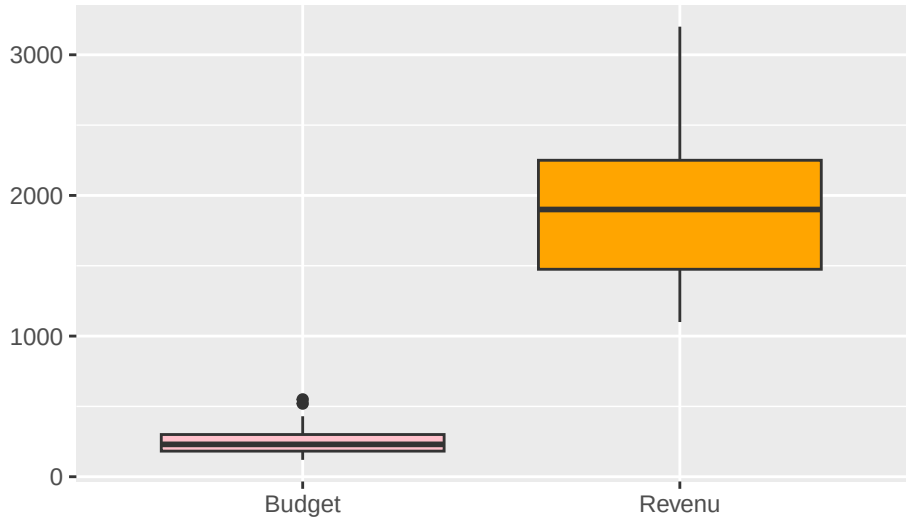
En rajoutant une variable qualitative (ici, lieu) dans la fonction **aes**(), on obtient des boites à moustaches par groupe. Par contre, il est nécessaire de supprimer les observations ayant une valeurs manquantes via la fonction **subset**().

```
ggplot(subset(courses,!is.na(courses$lieu)), aes(x=budget, y=lieu)) +
  geom_boxplot(na.rm=TRUE, fill=hcl.colors(4))
```



Finalement, il est facile de superposer les boites à moustaches de deux variables quantitatives (ayant la même unité !). Dans ce cas, on indique l'aes dans chaque geom (et non dans la fonction **`ggplot`**()).

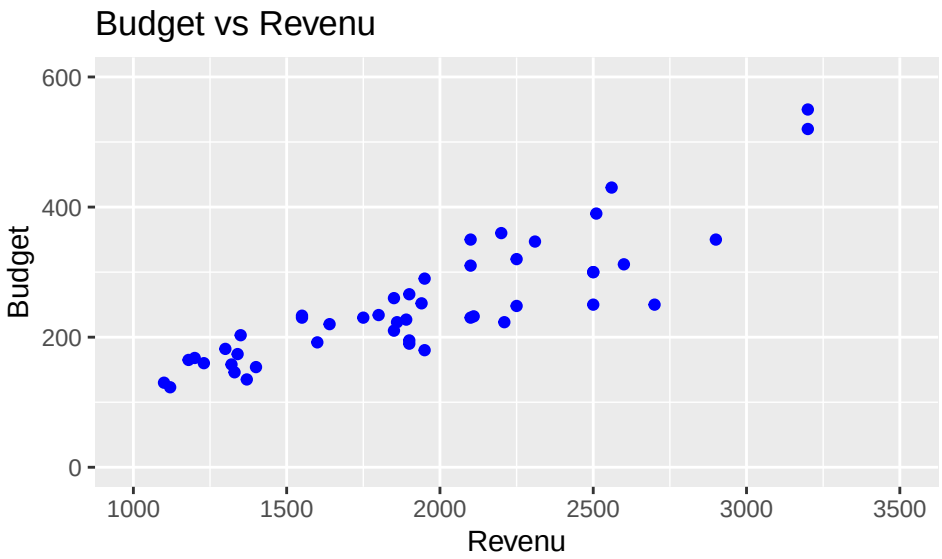
```
ggplot(courses) +
  geom_boxplot(aes(y=budget, x="Budget"), na.rm=TRUE, fill="pink") +
  geom_boxplot(aes(y=revenu, x="Revenu" ), na.rm=TRUE, fill="orange") +
  labs( y = "", x = "" )
```



4 Nuage de points

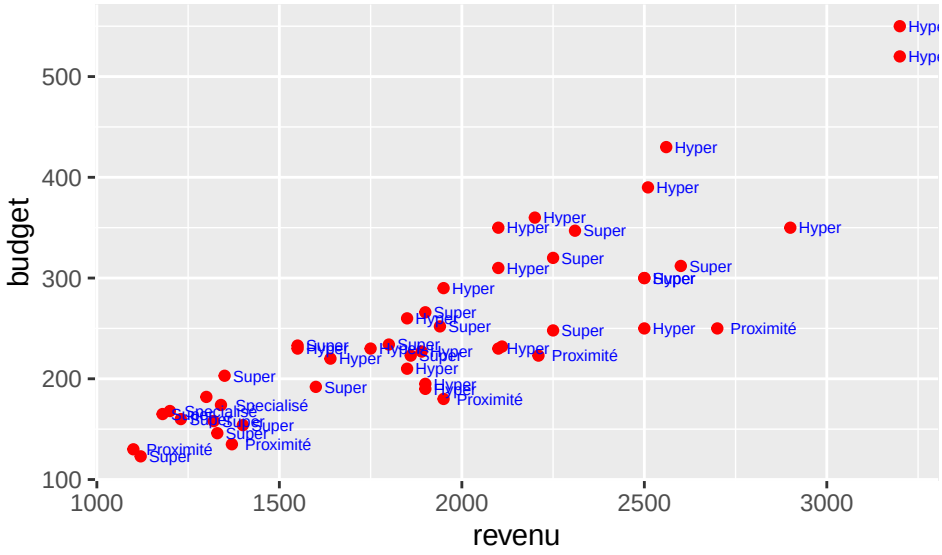
On a déjà vu en introduction l'utilisation de **`geom_point`**() pour obtenir un nuage de points :

```
ggplot( courses, aes(x=revenu, y=budget) ) + geom_point(color="blue",na.rm=TRUE) +
        coord_cartesian( xlim=c(1000,3500), ylim=c(0,600) ) +
        labs( x= "Revenu", y = "Budget", title ="Budget vs Revenu" )
```



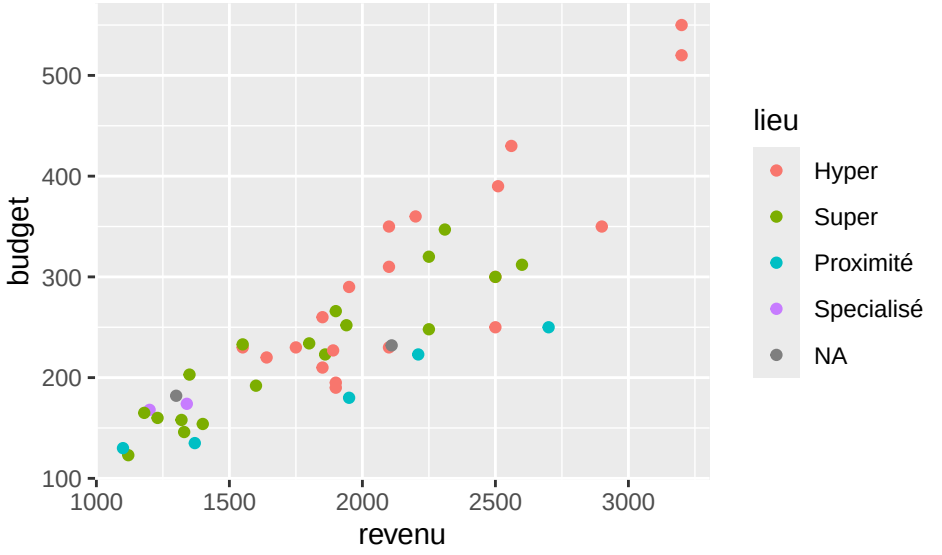
A l'aide de **`geom_text()`**, il est possible de rajouter des étiquettes à chaque point, par exemple le lieu :

```
ggplot(courses, aes(x=revenu, y=budget)) +
  geom_point(color="red", na.rm=TRUE ) +
  geom_text(label=courses$lieu, color="blue", na.rm=TRUE, size=2, hjust = -0.2)
```



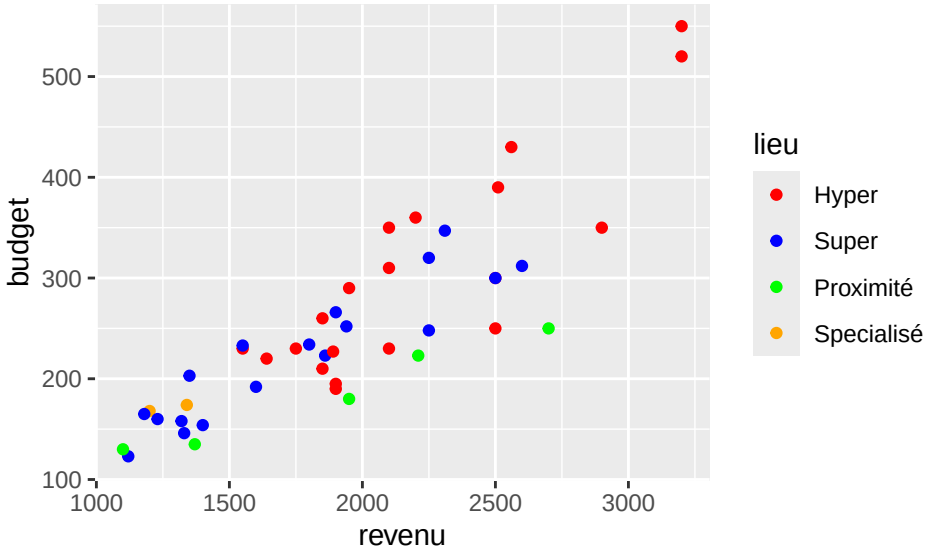
On peut aussi plus simplement utiliser un code couleur via l'indication `color=lieu` dans l'aes.

```
ggplot(courses, aes(x=revenu, y=budget, color=lieu)) + geom_point(na.rm=TRUE)
```



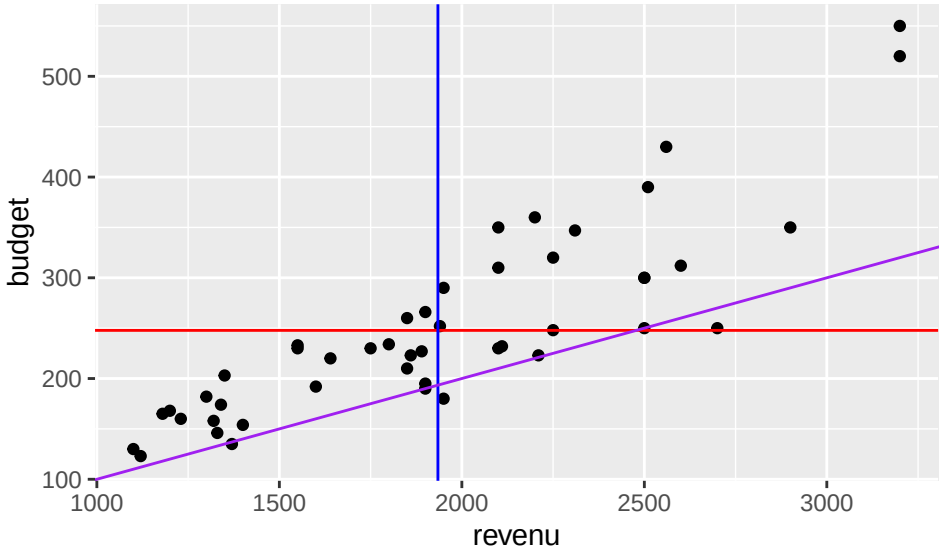
L'option `na.rm=TRUE` supprime les données manquantes des variables quantitatives (`revenu` et `budget`) mais pas de la variable qualitative (`lieu`). Pour cela, il est nécessaire, comme précédemment, de les supprimer directement dans le dataframe.
La fonction **`scale_colour_manual()`** permet de changer les couleurs par défaut. On aurait aussi pu utiliser un nuancier de couleurs avec l'instruction **`scale_colour_brewer`**`(palette = "Reds")`.

```
ggplot(subset(courses,!is.na(courses$lieu)), aes(x=revenu, y=budget, color=lieu)) +
   geom_point(na.rm=TRUE) + scale_colour_manual(values = c("red", "blue", "green","orange"))
```



D'autres geom permettent de rajouter des lignes de références :

```
ggplot(courses, aes(x=revenu, y=budget)) + geom_point(na.rm=TRUE) +
    geom_hline(yintercept=mean(courses$budget, na.rm=TRUE), color="red") +
    geom_vline(xintercept=mean(courses$revenu, na.rm=TRUE), color="blue") +
    geom_abline(intercept = 0, slope = 0.1, color="purple")
```

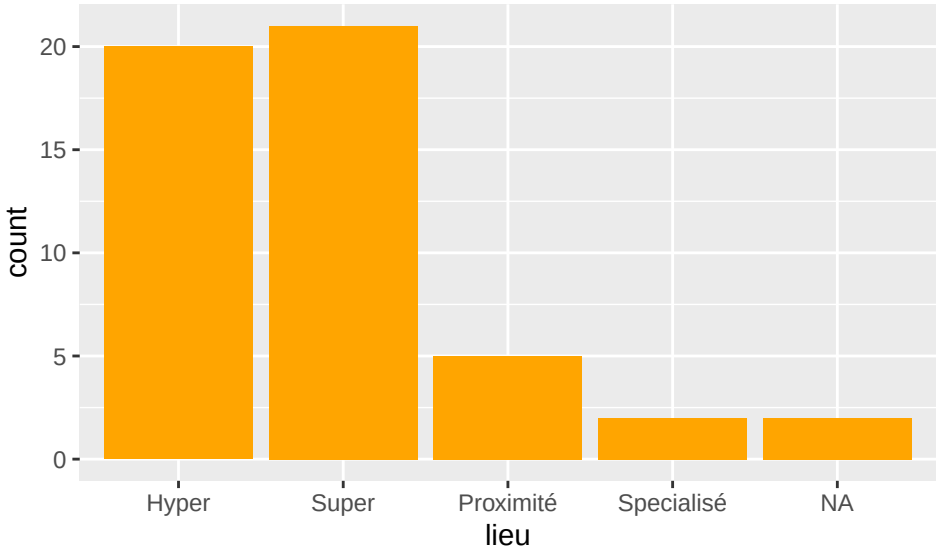


5 Diagramme en bâtons

5.1 Une seule variable qualitative

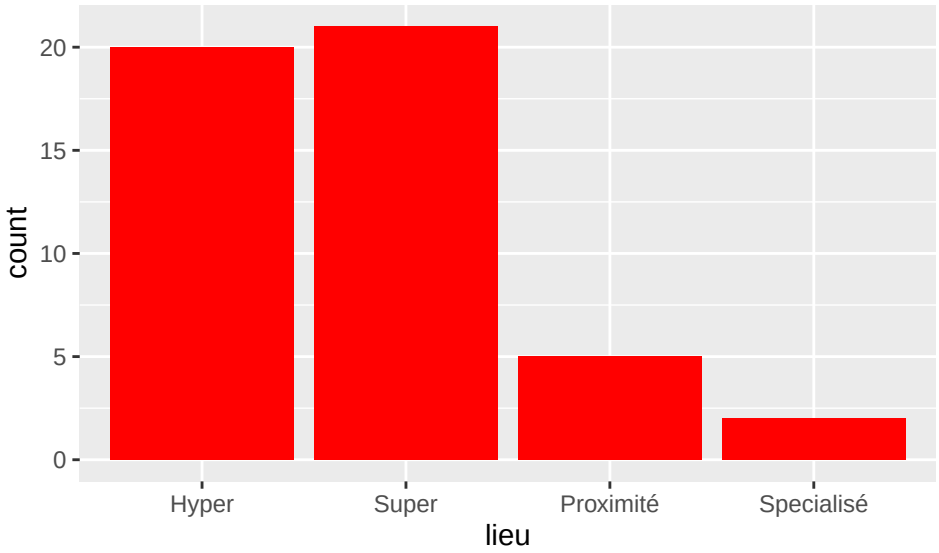
La fonction **`geom_bar()`** permet la représentation graphique des variables qualitatives sous forme de diagramme en bâtons :

```
ggplot(courses, aes(x = lieu)) + geom_bar(fill="orange")
```



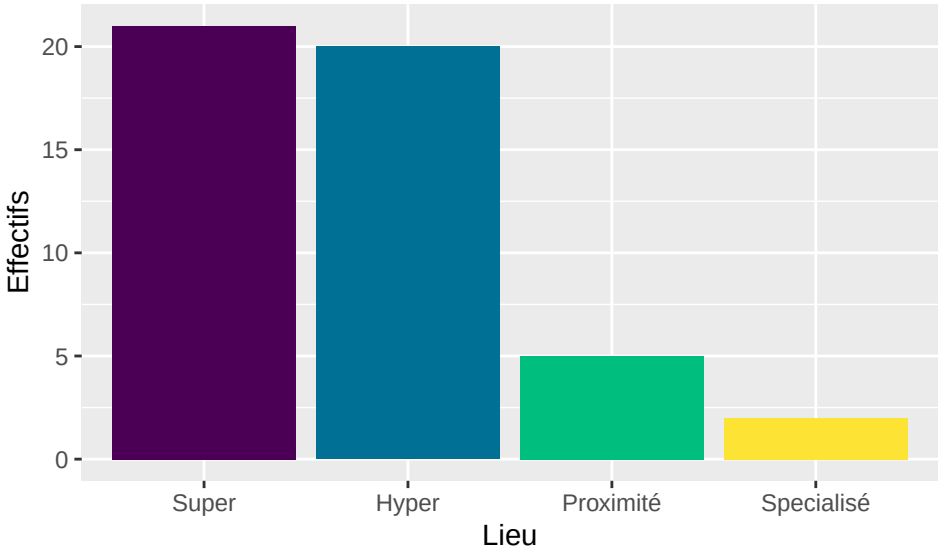
Pour supprimer les valeurs manquantes (NA), le plus simple est de les supprimer directement dans le dataframe, par exemple avec l'instruction **subset**() :

```
ggplot(subset(courses, !is.na(courses$lieu)), aes(x = lieu)) + geom_bar(fill="red")
```



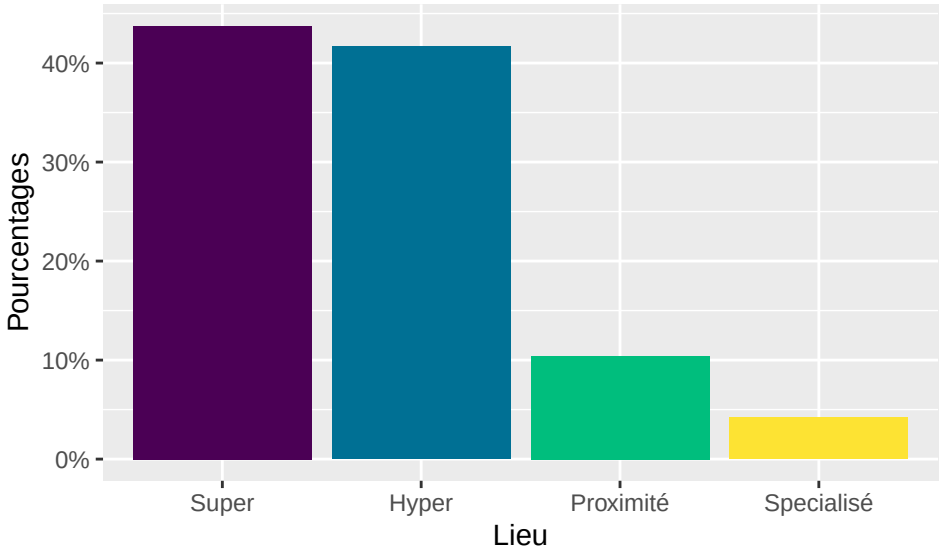
Le tri des modalités (pour les variables nominales) par effectifs décroissants se fait directement dans l'aes avec la fonction **reorder**() :

```
ggplot(subset(courses, !is.na(courses$lieu)),
   aes(x = reorder(lieu, lieu, length, decreasing = TRUE))) +
        geom_bar(fill=hcl.colors(4)) + labs(x="Lieu", y="Effectifs")
```



Finalement, on obtient les fréquences (pourcentages) au lieu des effectifs en modifiant l'aes de la sorte :

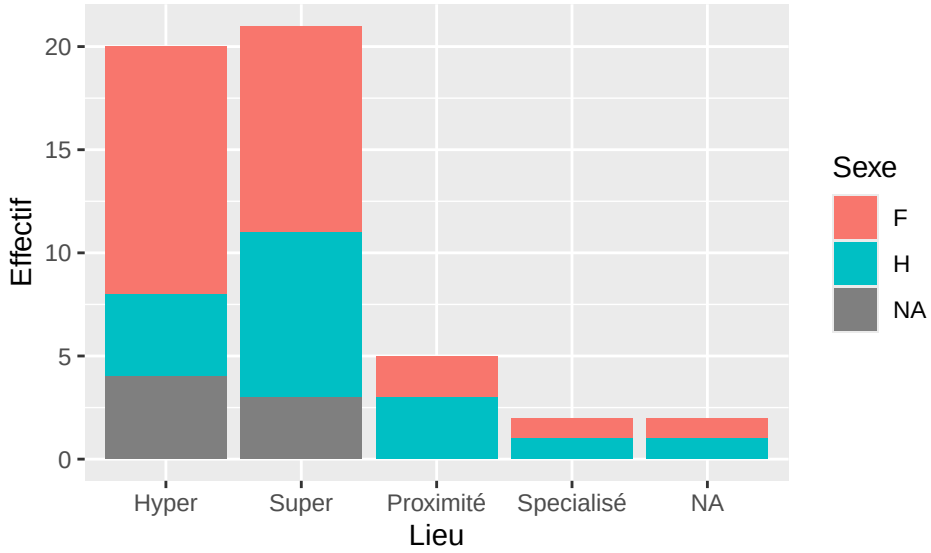
```
ggplot(subset(courses, !is.na(courses$lieu)),
  aes(x = reorder(lieu, lieu, length, decreasing = TRUE), y = after_stat(prop), group=1)) +
  geom_bar(fill=hcl.colors(4)) + labs(x="Lieu", y="Pourcentages") +
  scale_y_continuous(labels=percent_format())
```



5.2 Deux variables qualitatives

La fonction **geom_bar()** permet aussi la représentation graphique de couples de variables qualitatives

```
ggplot(courses, aes(x = lieu , fill=sexe)) + geom_bar() + labs(fill="Sexe", x="Lieu", y="Effectif")
```



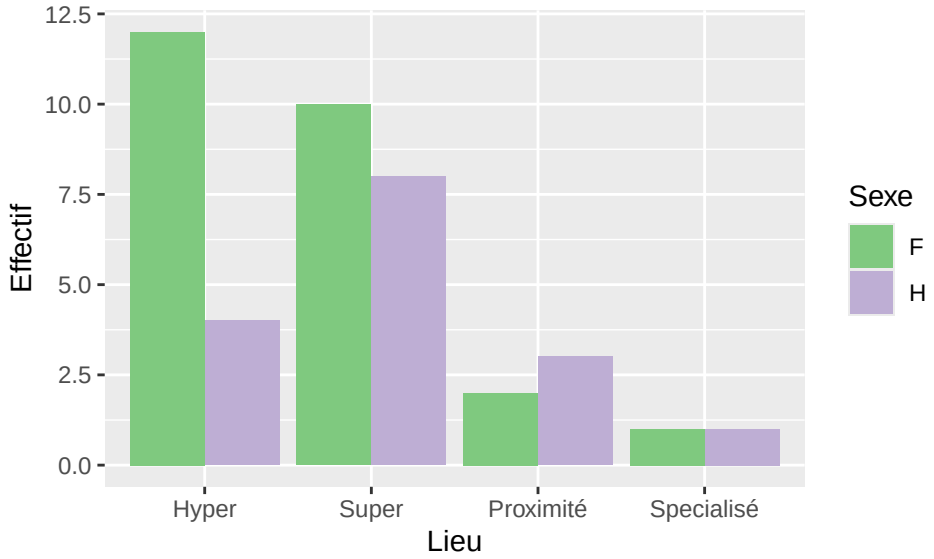
A nouveau, il est nécessaire de supprimer les données manquantes directement dans le dataframe. La fonction **`scale_fill_manual`**() permet de préciser les couleurs :

```
ggplot(subset(courses,!is.na(courses$lieu) & !is.na(courses$sexe)), aes(x = lieu , fill=sexe)) +
  geom_bar() + scale_fill_manual(values=c("pink", "lightblue")) +
  labs(fill="Sexe", x="Lieu", y="Effectif")
```

Pour utiliser un nuancier de couleurs prédéfinies, on utilise l'instruction **`scale_fill_brewer`**().

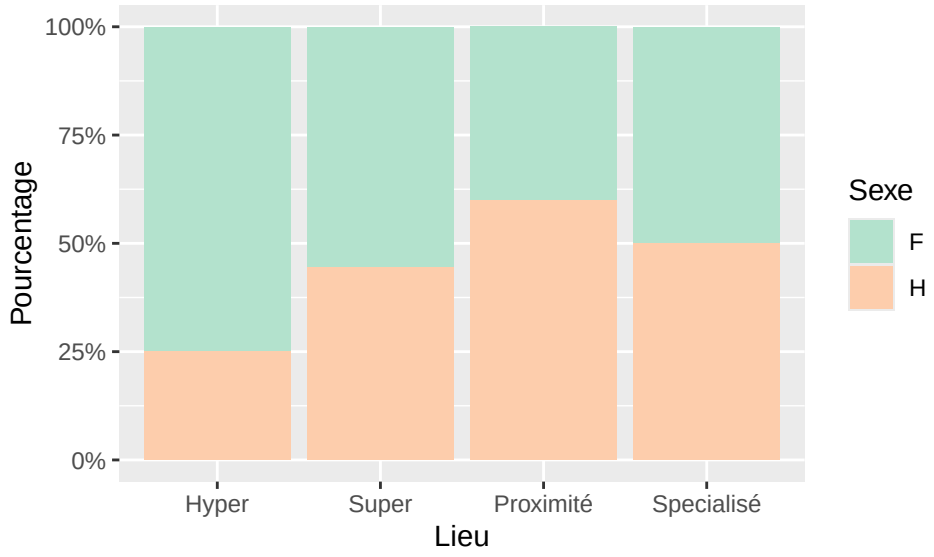
On obtient des bâtons juxtaposés simplement en rajoutant l'option `position = "dodge"` à **`geom_bar`**() :

```
ggplot(subset(courses,!is.na(courses$lieu) & !is.na(courses$sexe)), aes(x = lieu , fill=sexe)) +
  geom_bar(position = "dodge") + scale_fill_brewer(palette = "Accent") +
  labs(fill="Sexe", x="Lieu", y="Effectif")
```



De même pour des bâtons superposés à 100 % avec l'option `position = "fill"`, sans oublier de changer l'axe vertical avec la fonction **`scale_y_continuous`**`()` :

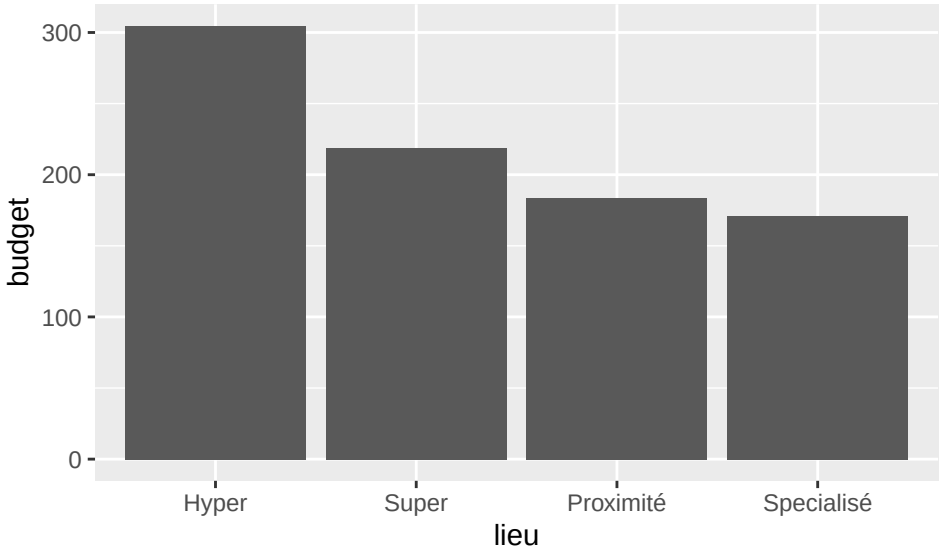
```
ggplot(subset(courses,!is.na(courses$lieu) & !is.na(courses$sexe)), aes(x = lieu , fill=sexe)) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels=percent_format()) +
  scale_fill_brewer(palette = "Pastel2") +
  labs(fill="Sexe", x="Lieu", y="Pourcentage")
```



5.3 Une variable quantitative

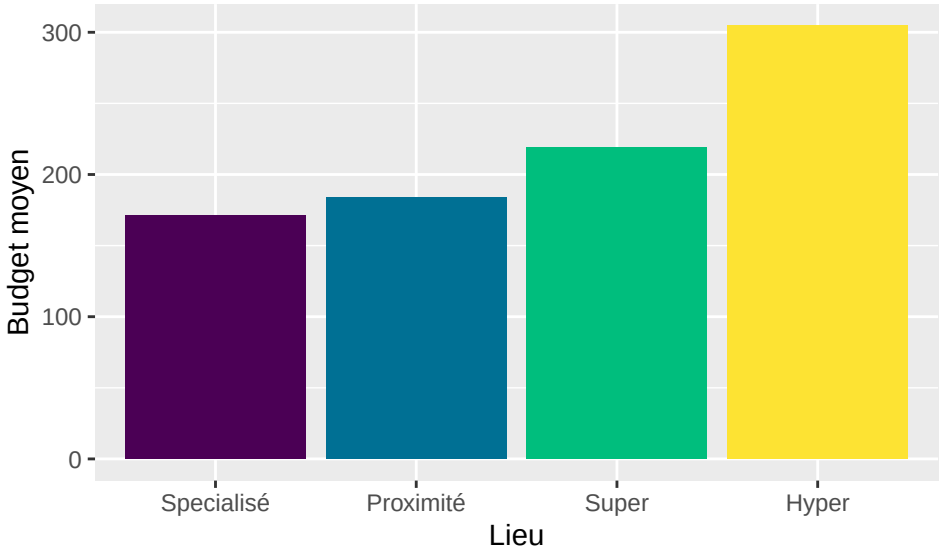
On utilise parfois des diagrammes en bâtons pour représenter la moyenne d'une variable quantitative selon une variable qualitative :

```
ggplot(subset(courses,!is.na(courses$budget) & !is.na(courses$lieu)), aes(x=lieu, y=budget)) +
  stat_summary(fun = "mean", geom = "bar")
```



Il est alors souvent préférable de classer les bâtons par ordre croissant ou décroissant :

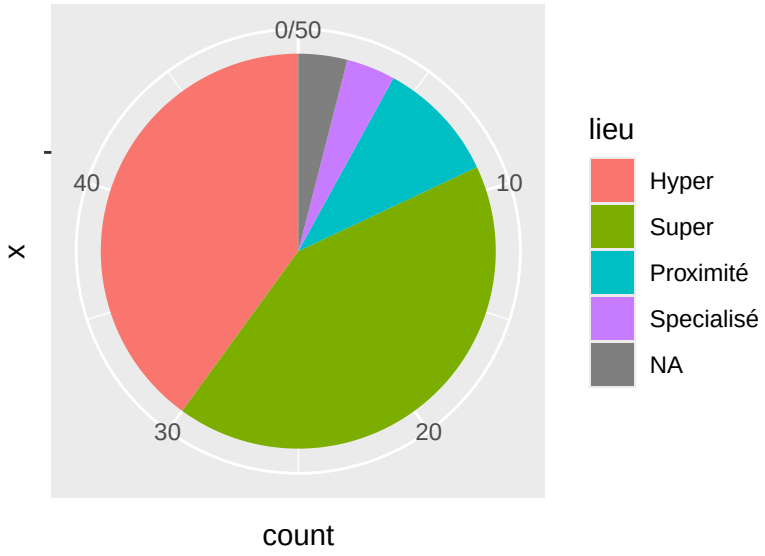
```
ggplot(subset(courses,!is.na(courses$budget) & !is.na(courses$lieu)),
                      aes(x=reorder(lieu,budget,mean, decreasing=FALSE), y=budget)) +
  stat_summary(fun="mean", geom="bar", fill=hcl.colors(4)) + labs(x = "Lieu", y = "Budget moyen")
```



6 Diagramme en secteurs

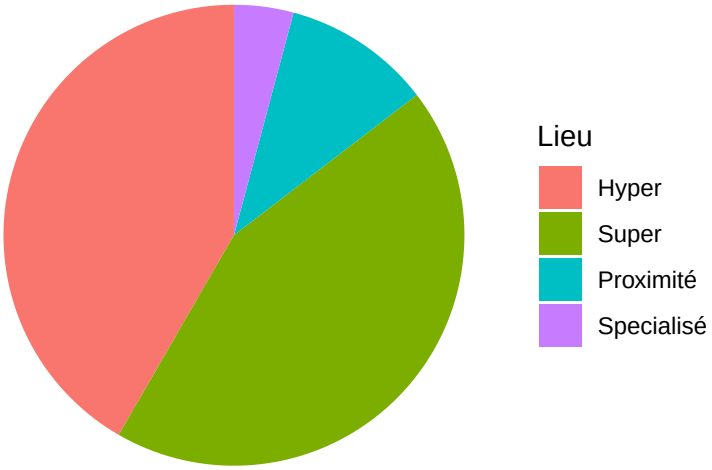
Bien que déconseillé par les concepteurs de R, il est possible d'obtenir un diagramme en secteurs. Il s'agit en fait d'un diagramme en bâtons (**geom_bar**()) transformé en *coordonnées polaires* !

```
ggplot(courses, aes(x="", fill=lieu)) + geom_bar() + coord_polar("y")
```



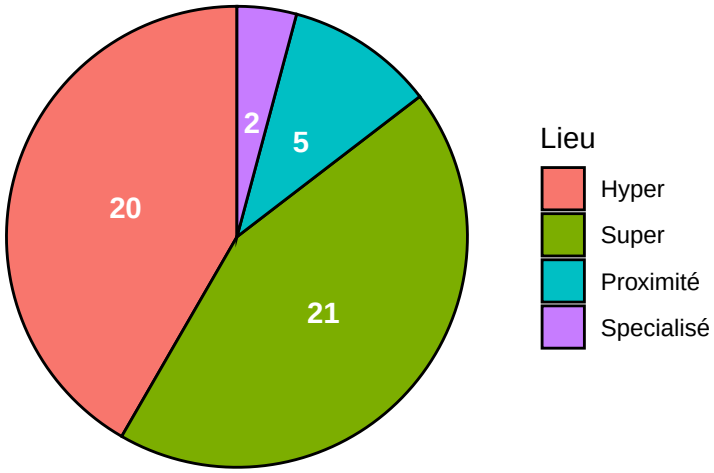
On supprime les valeurs manquantes de la même façon que précédemment (et **`theme_void`**`()` supprime les marques sur le graphique) :

```
ggplot(subset(courses,!is.na(courses$lieu)), aes(x="", fill=lieu)) + geom_bar() +
       coord_polar("y") + theme_void() + labs(fill="Lieu")
```



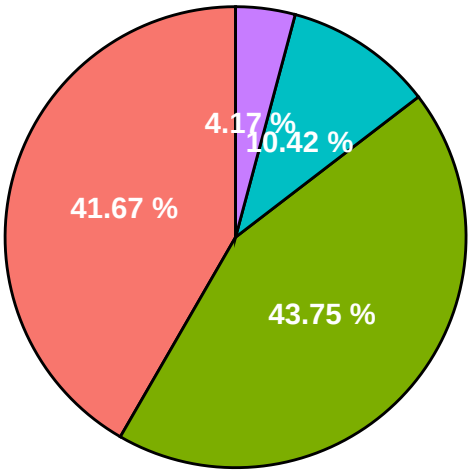
L'instruction **`geom_text`**`()` permet de superposer du texte à un graphique. Ici, on l'utilise pour afficher les effectifs dans chaque secteur :

```
ggplot(subset(courses,!is.na(courses$lieu)), aes(x="", fill=lieu)) + geom_bar(color="black") +
  geom_text(aes(label = after_stat(count) , y = after_stat(count)), stat = "count",
    color="white", fontface = "bold", position = position_stack(vjust = 0.5)) +
  coord_polar("y") + theme_void() + labs(fill="Lieu")
```



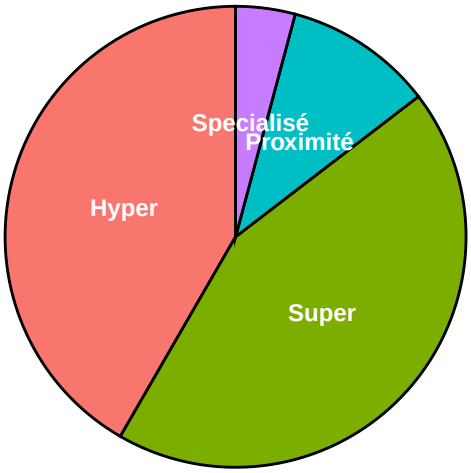
Un peu plus de travail pour afficher les pourcentages (l'instruction **paste**() permet de combiner la valeur du pourcentage et le symbole %) :

```
ggplot(subset(courses,!is.na(courses$lieu)), aes(x="", fill=lieu)) +
  geom_bar(color="black",show.legend = FALSE) +
  geom_text(aes(label = paste(round(100*after_stat(count/ sum(count)),2),"%"),
    y = after_stat(count)), stat = "count",
    color="white", fontface = "bold", position = position_stack(vjust = 0.5)) +
  coord_polar("y") + theme_void() + labs(fill="Lieu")
```



De la même façon, on peut aussi indiquer les étiquettes de modalité dans chaque secteur (en supprimant la légende avec l'instruction **theme**() :

```
ggplot(subset(courses,!is.na(courses$lieu)), aes(x="", fill=lieu)) + geom_bar(color="black") +
   geom_text(aes(label = lieu , y = after_stat(count)), stat = "count",
       color="white", fontface = "bold", size=3, position = position_stack(vjust = 0.5)) +
  coord_polar("y") + theme_void() + theme(legend.position = "none")
```



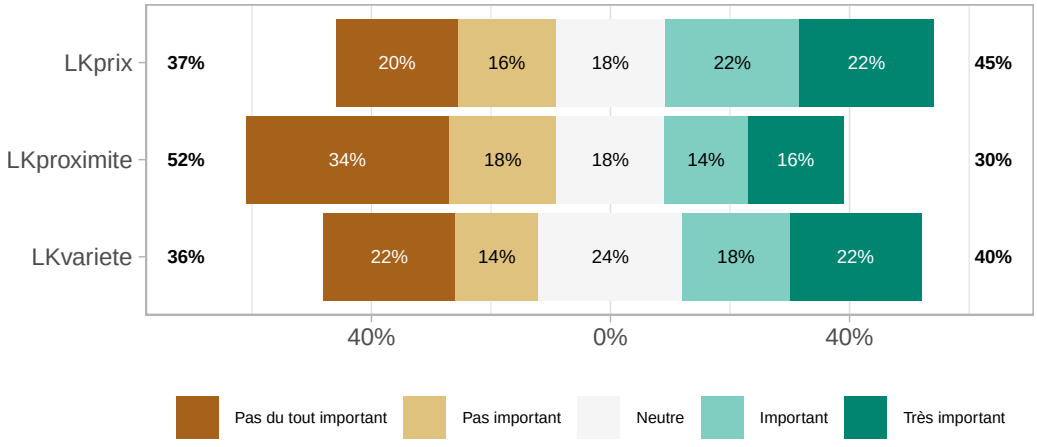
7 Variables de Likert

Le package ggstats permet de représenter très facilement des variables utilisant des échelles de Likert. Tout d'abord, il est préférable de transformer ces variables en facteurs.

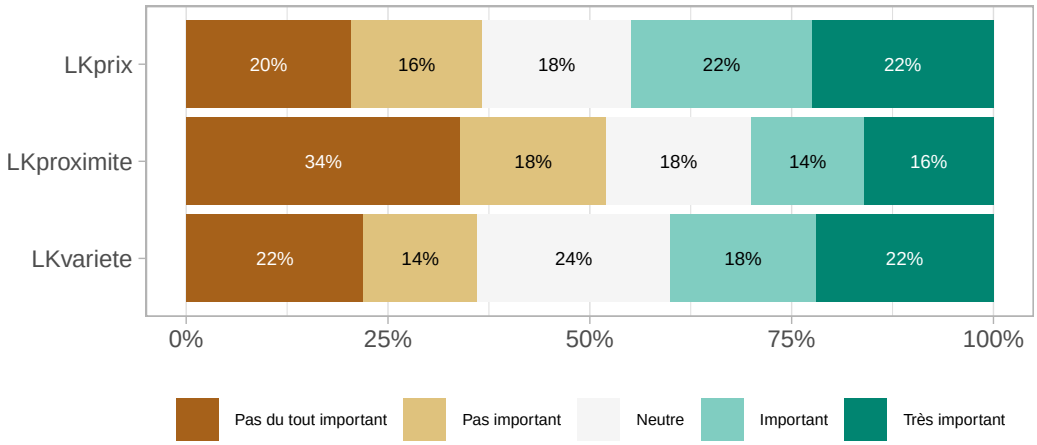
```
courses$LKprix <- ordered(courses$LKprix, level=c(1,2,3,4,5),
   labels=c('Pas du tout important','Pas important','Neutre','Important','Très important'))
courses$LKproximite <- ordered(courses$LKproximite, level=c(1,2,3,4,5),
   labels=c('Pas du tout important','Pas important','Neutre','Important','Très important'))
courses$LKvariete <- ordered(courses$LKvariete, level=c(1,2,3,4,5),
   labels=c('Pas du tout important','Pas important','Neutre','Important','Très important'))
```

On utilise alors une des deux fonctions `gglikert()` ou `gglikert_stacked()`.

```
gglikert(courses, include = c('LKprix','LKproximite','LKvariete'), labels_size=2.5) +
         theme(legend.text = element_text(size = 6))
```




```
gglikert_stacked(courses, include = c('LKprix','LKproximite','LKvariete'), labels_size=2.5) +
         theme(legend.text = element_text(size = 6))
```

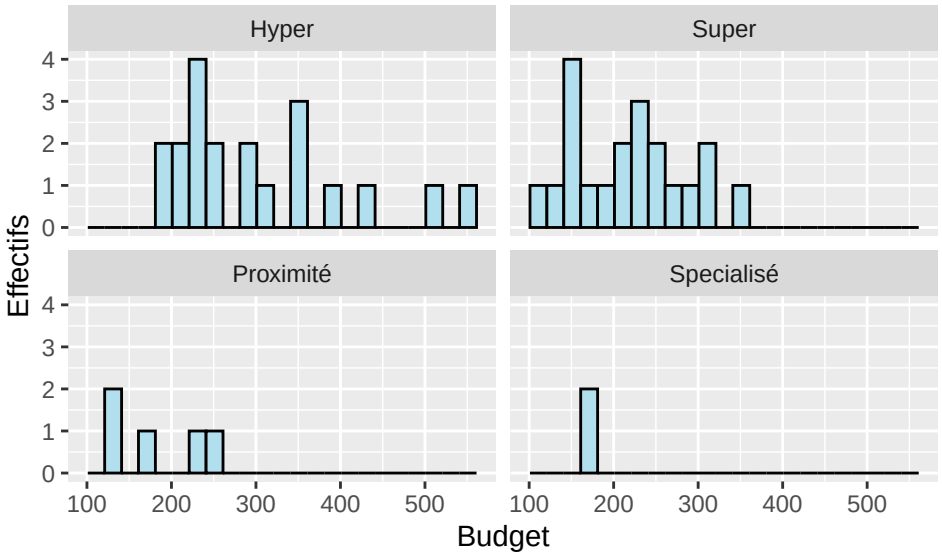


8 Suite de graphiques

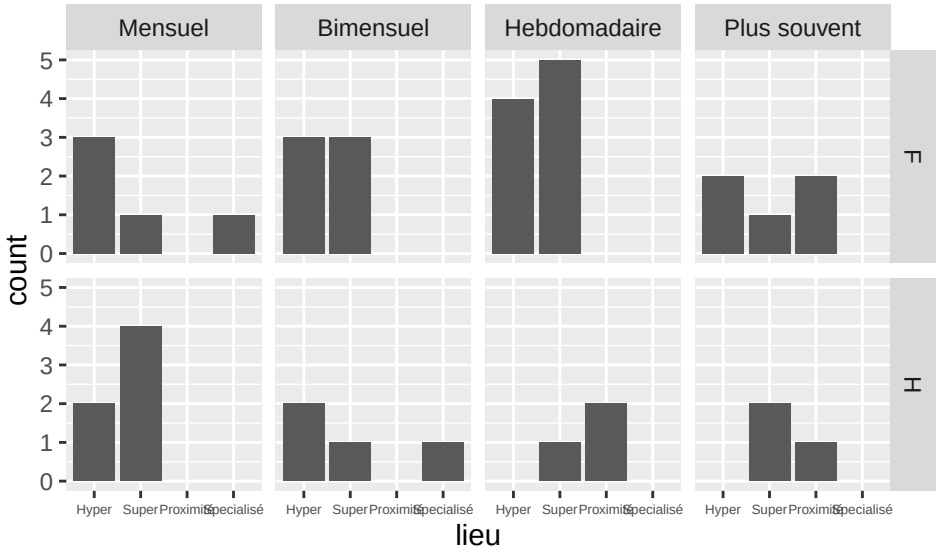
Pour générer le même graphique, par exemple un histogramme, sur des groupes d'observations définis par une ou plusieurs variables qualitatives, il suffit d'utiliser les fonctions **`facet_wrap`**`()` (pour une variable qualitative) et **`facet_grid`**`()` pour deux variables. A noter l'utilisation de la fonction **`vars`**`()` pour indiquer les variables qualitatives définissant les groupes.

```
ggplot(subset(courses, !is.na(courses$lieu)), aes(x=budget)) +
  geom_histogram(binwidth=20, color="black", fill="lightblue2", boundary=1) +
  labs(x= "Budget", y = "Effectifs") + facet_wrap(vars(lieu), nrow = 2)

## Warning: Removed 1 row containing non-finite outside the scale range (`stat_bin()`).
```



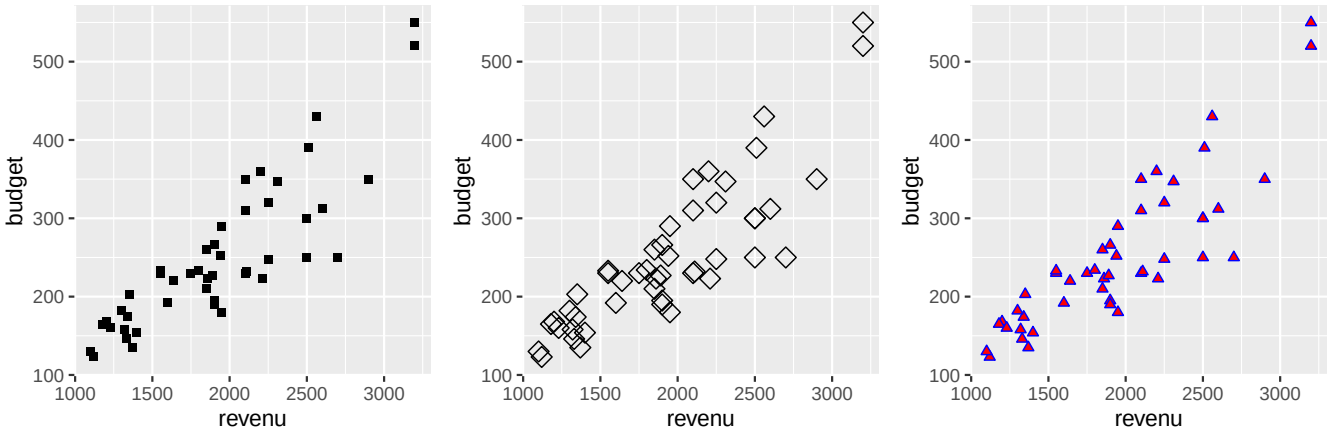

```
ggplot(subset(courses,!is.na(courses$lieu) & !is.na(courses$sexe)), aes(x=lieu)) +
        geom_bar() + theme(axis.text.x = element_text(size = 5)) +
        facet_grid(cols=vars(frequence), rows=vars(sexe))
```



9 Formes et couleurs

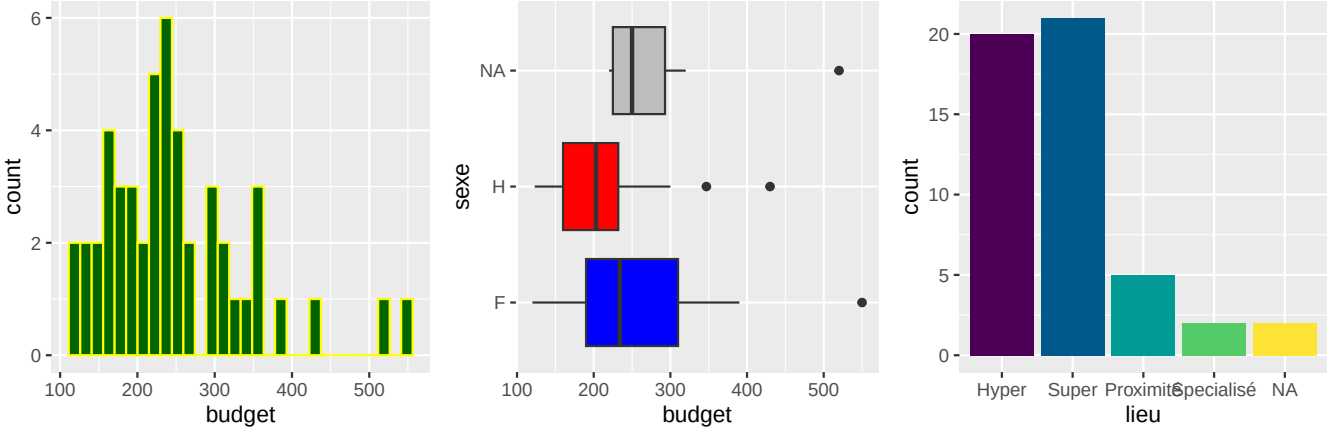
Le forme du symbole utilisé dans les nuages de points peut être modifiée avec l'option `shape`. Les valeurs possibles sont `circle`, `square`, `diamond`, `triangle` éventuellement complétées de `open` ou `filled` :

```
ggplot(courses, aes(x=revenu, y=budget)) + geom_point(na.rm=TRUE, shape="square")
ggplot(courses, aes(x=revenu, y=budget)) + geom_point(na.rm=TRUE, shape="diamond open", size=3)
ggplot(courses, aes(x=revenu, y=budget)) + geom_point(na.rm=TRUE, shape="triangle filled",
                                                      fill="red", color="blue")
```



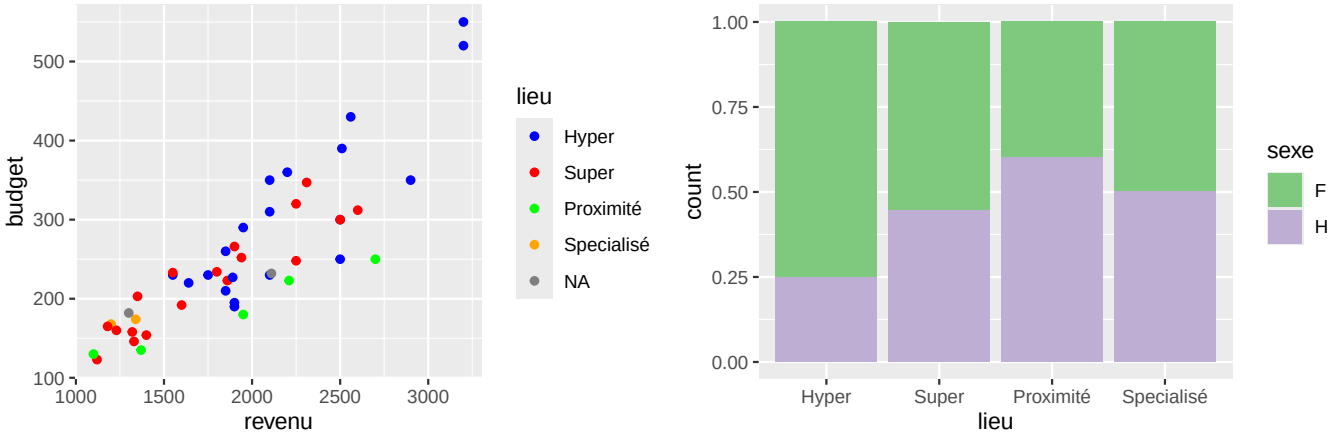
Il est possible de personaliser la couleur de la plupart des objets graphiques avec les options `color` (pour les contours) et `fill` pour l'intérieur. On utilise soit une ou plusieurs couleurs nommées (`"red"`, `"lightblue"`...) ou un nuancier (`rainbow(1)`, `hcl.colors(3)`).

```
ggplot(courses, aes(x=budget)) + geom_histogram(fill="darkgreen", color="yellow")
ggplot(courses, aes(x=budget, y=sexe)) + geom_boxplot(na.rm=TRUE, fill=c('blue','red','gray'))
ggplot(courses, aes(x = lieu)) + geom_bar(fill=hcl.colors(5))
```



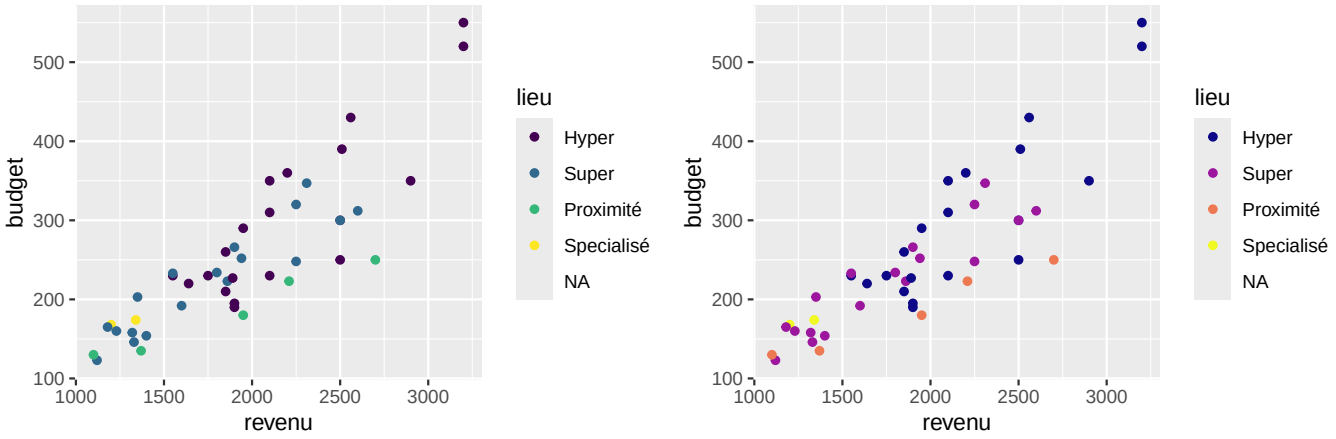
Lorsque les couleurs sont générées automatiquement (par une variable qualitative), il est nécessaire d'utiliser les fonctions **`scale_colour_manual`**() et **`scale_fill_manual`**() pour spécifier les couleurs du contour et de l'intérieur manuellement et **`scale_colour_brewer`**() et **`scale_fill_brewer`**() pour utiliser les nuanciers de couleurs Brewer.
Les principaux nuanciers Brewer sont les suivants : `Blues`, `Greens`, `Greys`, `Oranges`, `Purples`, `Reds`, `Accent`, `Dark2`, `Paired`, `Pastel1`, `Pastel2`, `Set1`, `Set2`, `Set3`, `Spectral`...

```
ggplot(courses, aes(x=revenu, y=budget, color=lieu)) + geom_point(na.rm=TRUE) +
        scale_colour_manual(values=c('blue','red','green','orange'))
ggplot(subset(courses,!is.na(courses$lieu) & !is.na(courses$sexe)), aes(x = lieu , fill=sexe)) +
        geom_bar(position = "fill") + scale_fill_brewer(palette = "Accent")
```



D'autres nuanciers sont disponibles, par exemple les nuanciers Viridis (compatibles avec la plupart des *dyschromatopsies*), via les fonctions **`scale_colour_viridis_d`**() et **`scale_fill_viridis_d`**(). Les principaux nuanciers Viridis sont `viridis` (par défaut), `magma`, `inferno`, `plasma`, `cividis`, `rocket`, `mako`, `turbo`.

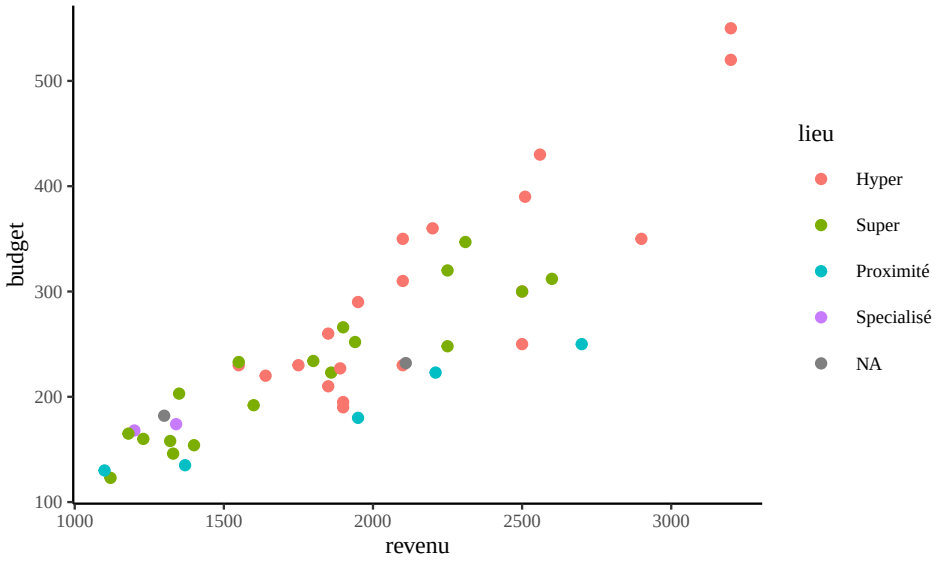
```
ggplot(courses, aes(x=revenu, y=budget, color=lieu)) + geom_point() + scale_colour_viridis_d()
ggplot(courses, aes(x=revenu, y=budget, color=lieu)) + geom_point() +
                                        scale_colour_viridis_d(option="plasma")
```



10 Themes

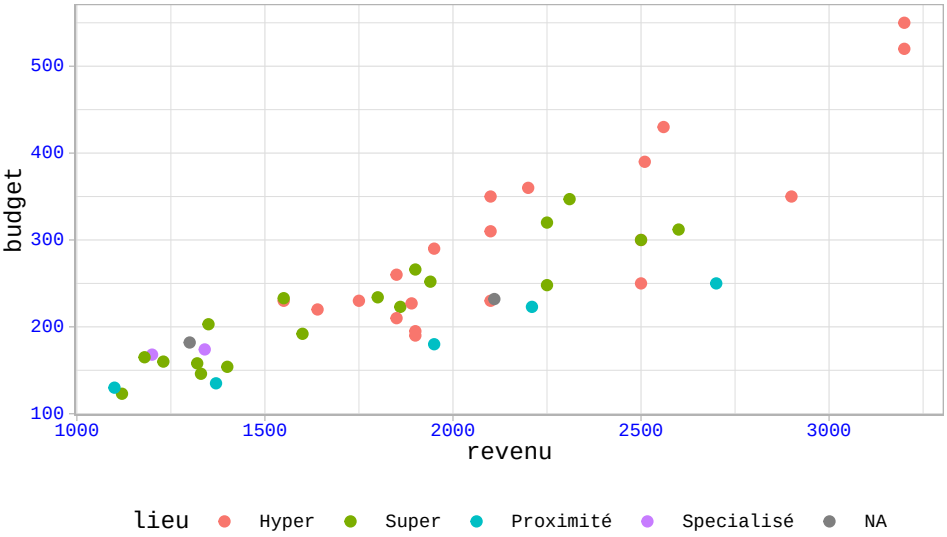
De nombreux thèmes prédéfinis permettent de changer l'apparence globale du graphique. Parmi ceux-ci, **`theme_bw`**(), **`theme_gray`**(), **`theme_classic`**(), **`theme_minimal`**(), **`theme_light`**(), **`theme_dark`**(), **`theme_void`**()...
Il est aussi possible de préciser la taille des caractères (`base_size`) et la police utilisée (`base_family`) :

```
ggplot(courses, aes(x=revenu, y=budget, color=lieu)) + geom_point(na.rm=TRUE) +
  theme_classic(base_size=9, base_family = "serif")
```



Plus généralement, la fonction **theme**() permet de changer certains aspects d'un graphique :

```
ggplot(courses, aes(x=revenu, y=budget, color=lieu)) + geom_point(na.rm=TRUE) +
  theme_light(base_size=9, base_family = "mono") +
  theme(legend.position = "bottom", axis.text = element_text(colour = "blue"))
```



Le thème par défaut est **theme_gray**(). Pour le changer en **theme_light**() ainsi que par exemple la taille par défaut de la police, on utilise **theme_set**() :

```
theme_set(theme_light(base_size = 13))
```