

Introduction à ggplot

Denis Puthier

21 August 2017

Présentation de ggplot2

Lorsqu'on fait des statistiques descriptives on souhaite souvent partitionner la fenêtre graphique en fonction des différents niveaux pris par une variable catégorielle (*i.e* qualitative) ou ordinale. Réaliser de tels graphiques se révèle vite assez compliqué avec les librairies de base (*graphics*, *lattice*...). Dans le but de faciliter la réalisation de tels graphiques *Hadley Wickham* a développé la librairie *ggplot2* qui est rapidement devenue populaire dans le monde de la bioinformatique (ici les variables catégorielles peuvent être des gènes, groupe de gènes, chromosomes, voies de signalisation, marque de chromatine...) et les variables ordinales des classes d'expression par exemple. L'une des particularités de la librairie *ggplot2* est que son développement est basé sur un modèle proposé par *Leland Wilkinson* dans son ouvrage "The Grammar of Graphics". Dans ce modèle le graphique est vu comme une entité composée de couches successives (*layers*), d'échelles (*scales*), d'un système de coordonnées et de facettes. Il faut donc créer un graphique et venir ajouter les différents éléments à l'aide de l'opérateur '+'. Le principe est assez déconcertant pour les utilisateurs des librairies basiques de R. Cependant, avec le temps on mesure l'intérêt de cette solution car elle nécessite moins de manipulations pour réaliser des graphiques complexes.

Réaliser un graphique basique

Nous utiliserons par la suite le jeu de données "chickwts". L'aide sur ce jeu de données indique:

Chicken Weights By Feed Type: An experiment was conducted to measure and compare the effectiveness of various feed supplements on the growth rate of chickens.

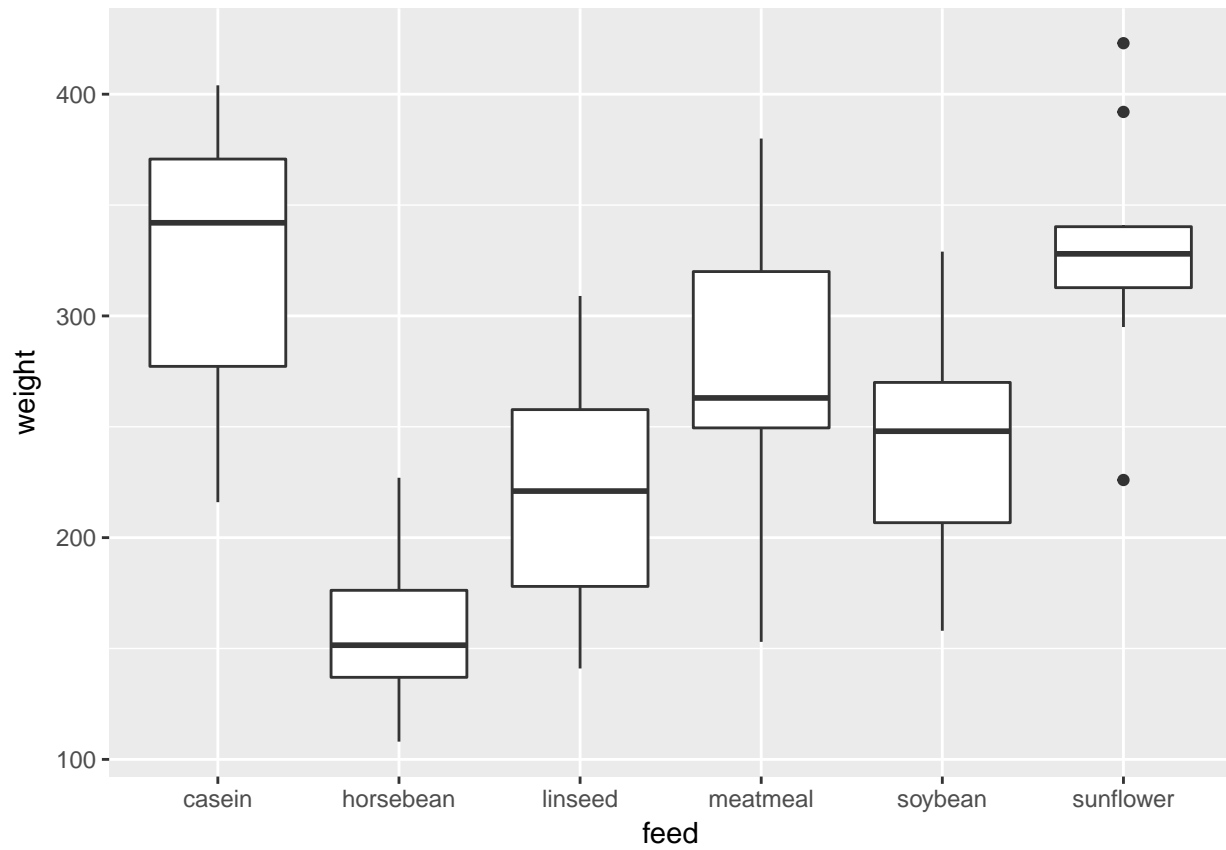
boxplot et violin plot

Les boîtes à moustaches (*boxplots*) et diagramme en violon (*violin plots*) peuvent être utilisés pour représenter les distributions associées à un jeu de données. On donne ci-dessous quelques exemples.

```
## loading ggplot2 package
library(ggplot2)
## Then we can load a demonstration dataset
data(chickwts)
View(chickwts)

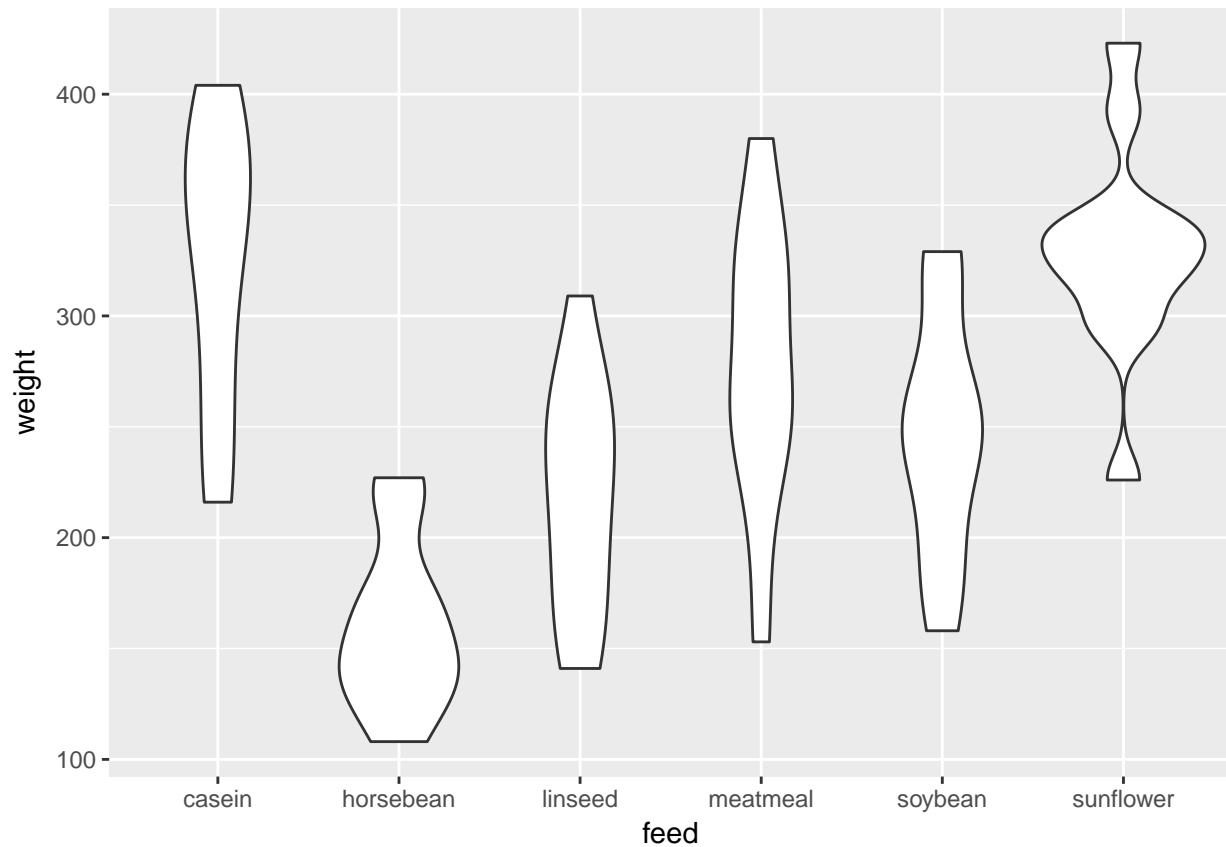
## Then we declare a new graphics and associate
## a dataset. Here the aes (aesthetic) argument is set
## to feed and weight.
p <- ggplot(data=chickwts, aes( x=feed, y=weight))

## We have to indicate the type of requested graphics
p.bp <- p + geom_boxplot()
print(p.bp)
```



```
## We can also easily produce a violin plot using the following instructions
```

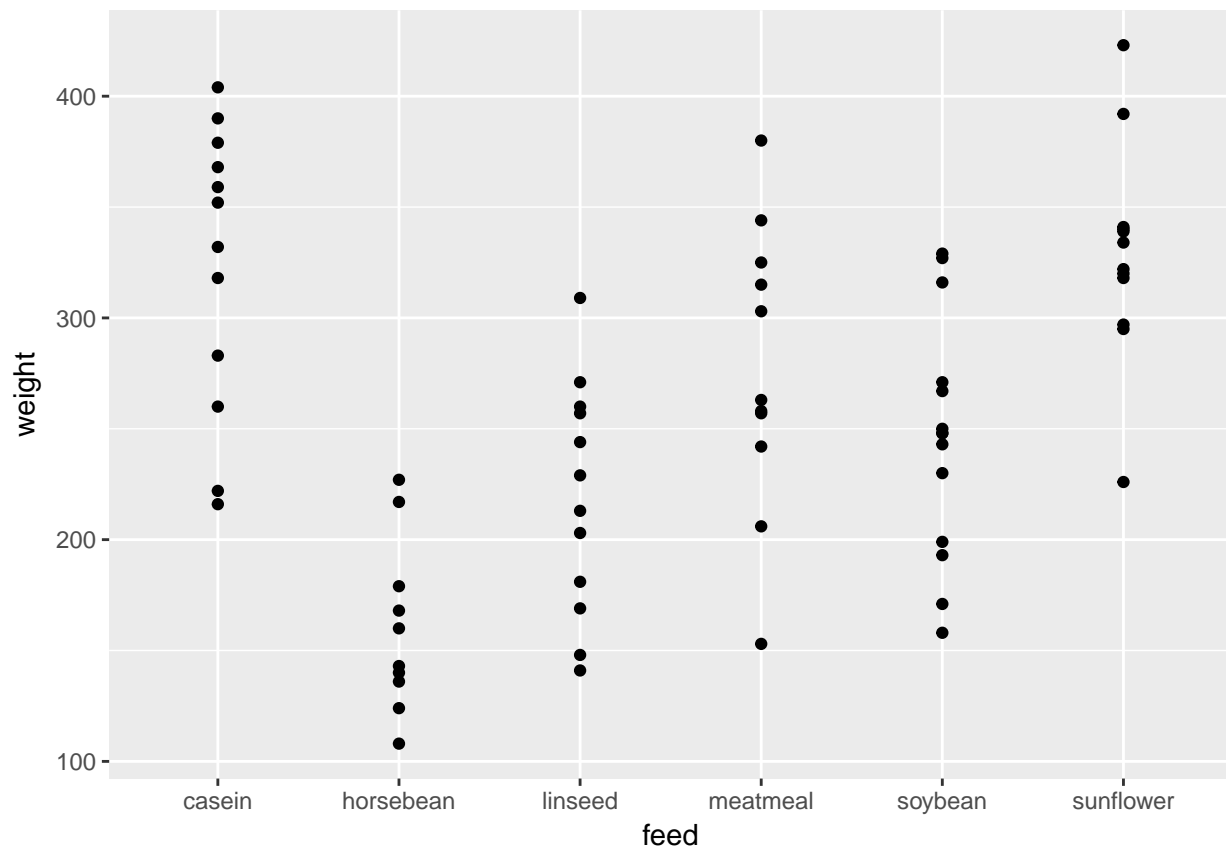
```
p.vp <- p + geom_violin()  
print(p.vp)
```



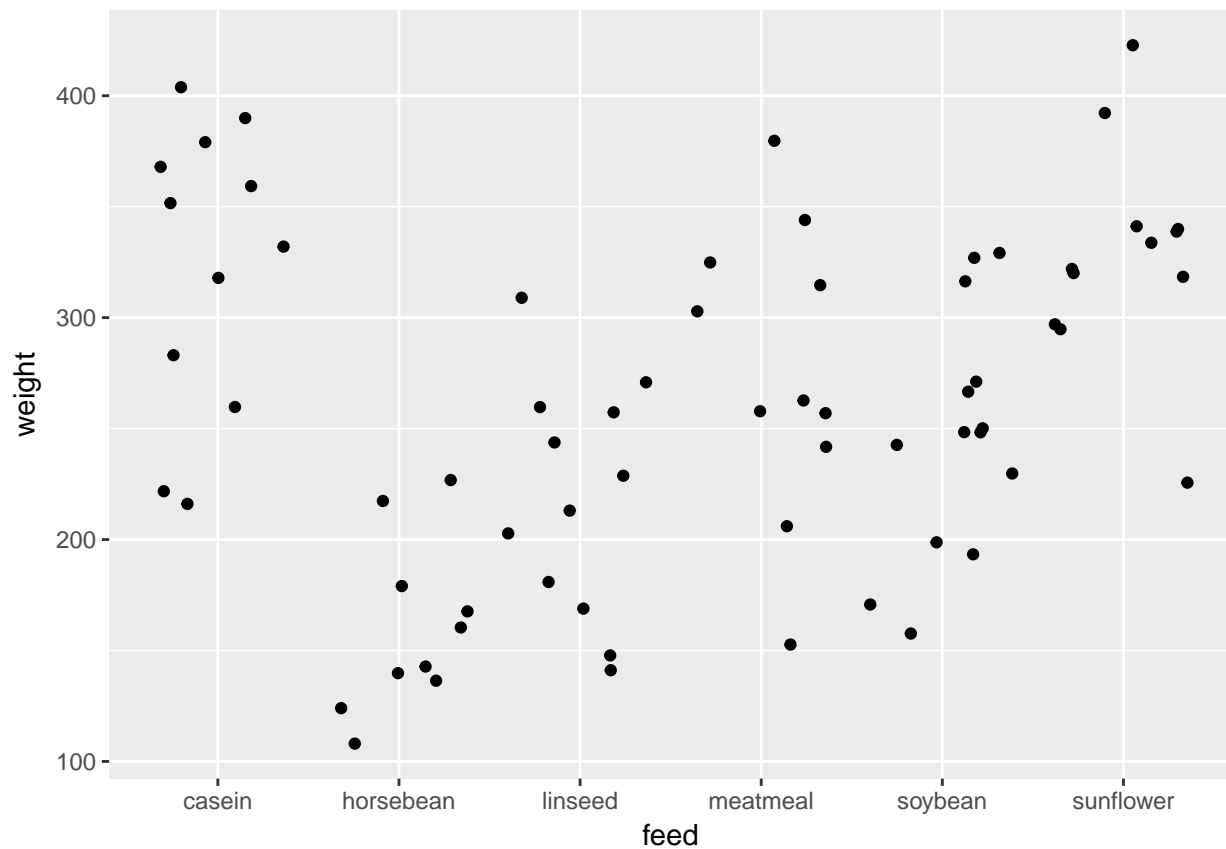
Nuages de points

Il y a environ 40 types de graphiques disponibles. Quelques exemples présentent ci-dessous les instructions pour réaliser des nuages de points.

```
## We can for instance show the values associated to each feed
p.pt <- p + geom_point()
print(p.pt)
```



```
## However as there are some ties it may be advised to
## use the jitter option that will add some randomness to the value of the x axis (that here are categories)
p.jt <- p + geom_jitter()
print(p.jt)
```

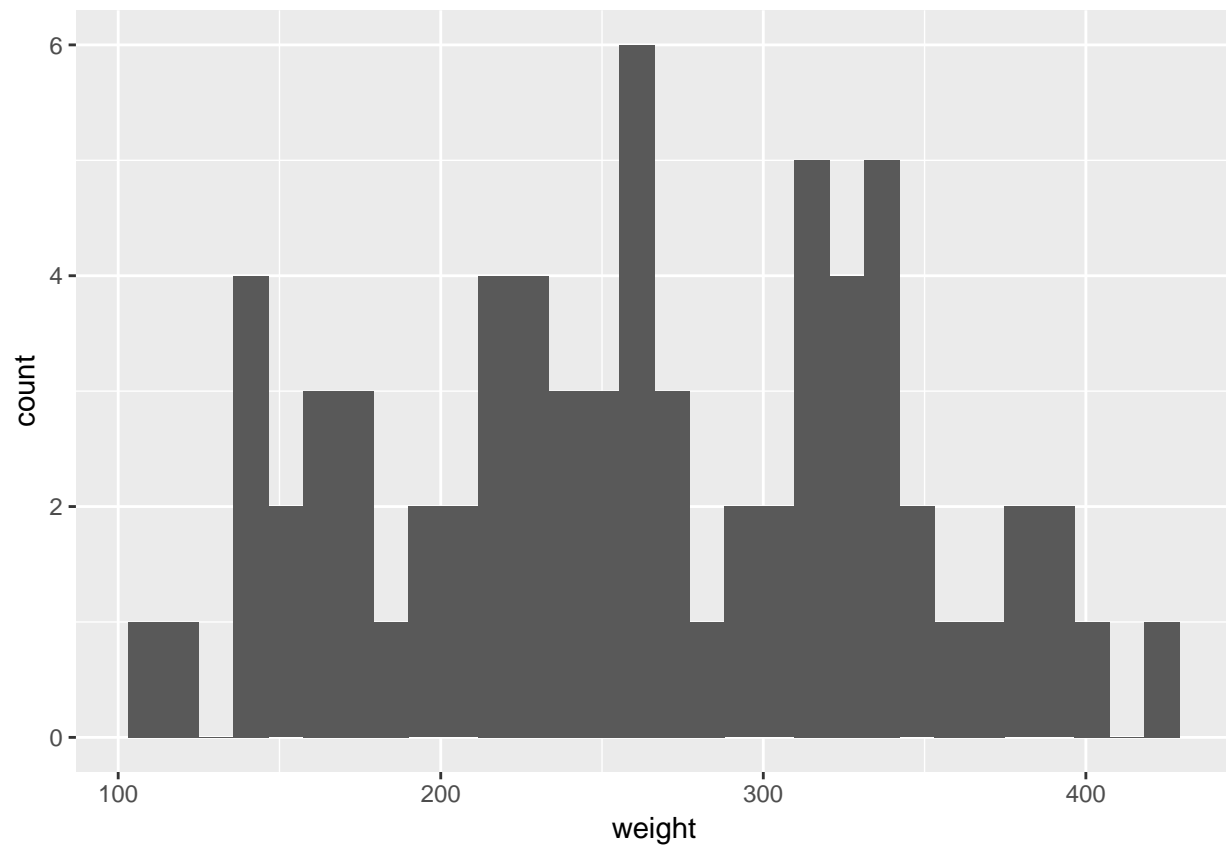


Histogrammes et densités

Dans le cas de l'histogramme, l'axe des x correspond à des interval (*bins*) et l'axe des y au nombre de fois on les valeurs de comptage sont observés dans ces intervals. Il n'y aura donc qu'une seule variable à fournir pour la fonction *aes*.

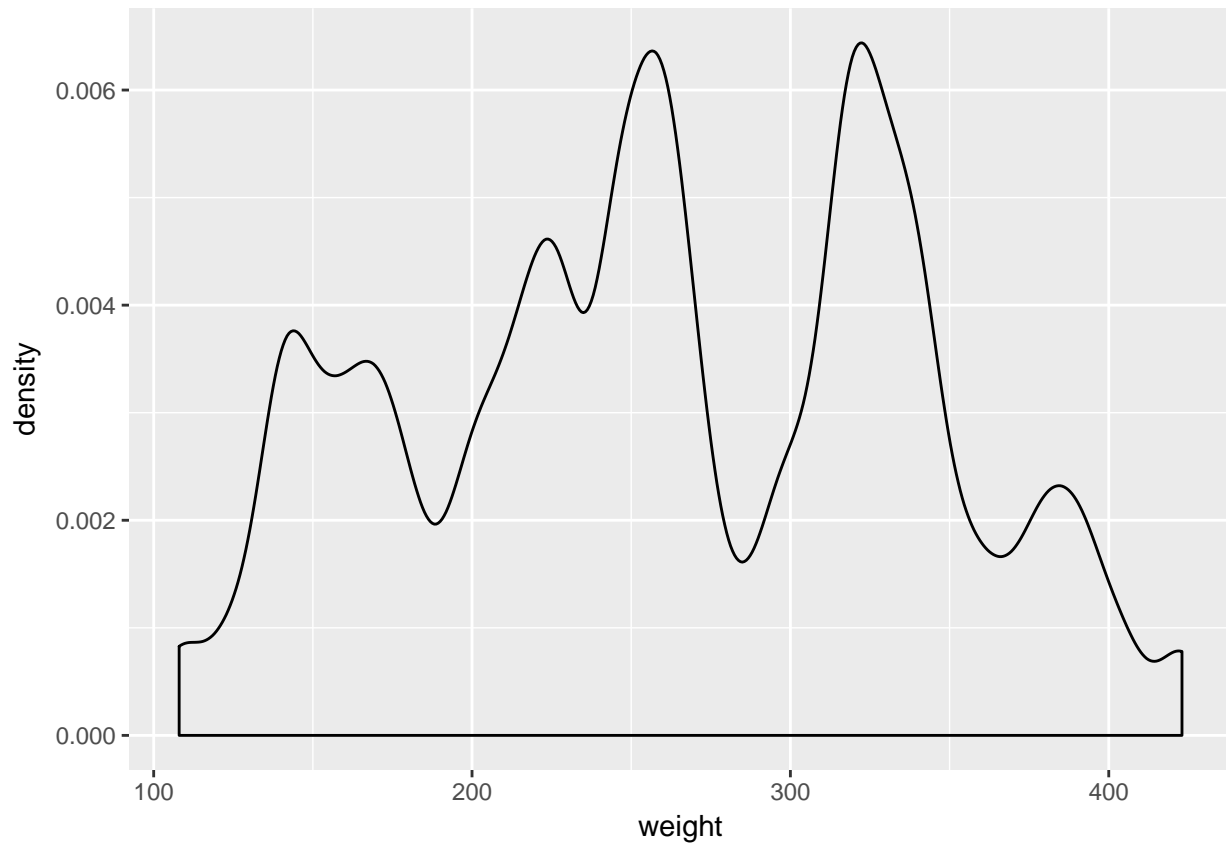
```
## Here the code for an histogram
ggplot(data=chickwts, aes(x=weight)) + geom_histogram()

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



On peut aussi réaliser un profil de densité de probabilité en utilisant la fonction `geom_density()`

```
ggplot(data=chickwts, aes(x=weight)) + geom_density(adjust = 1/4)
```

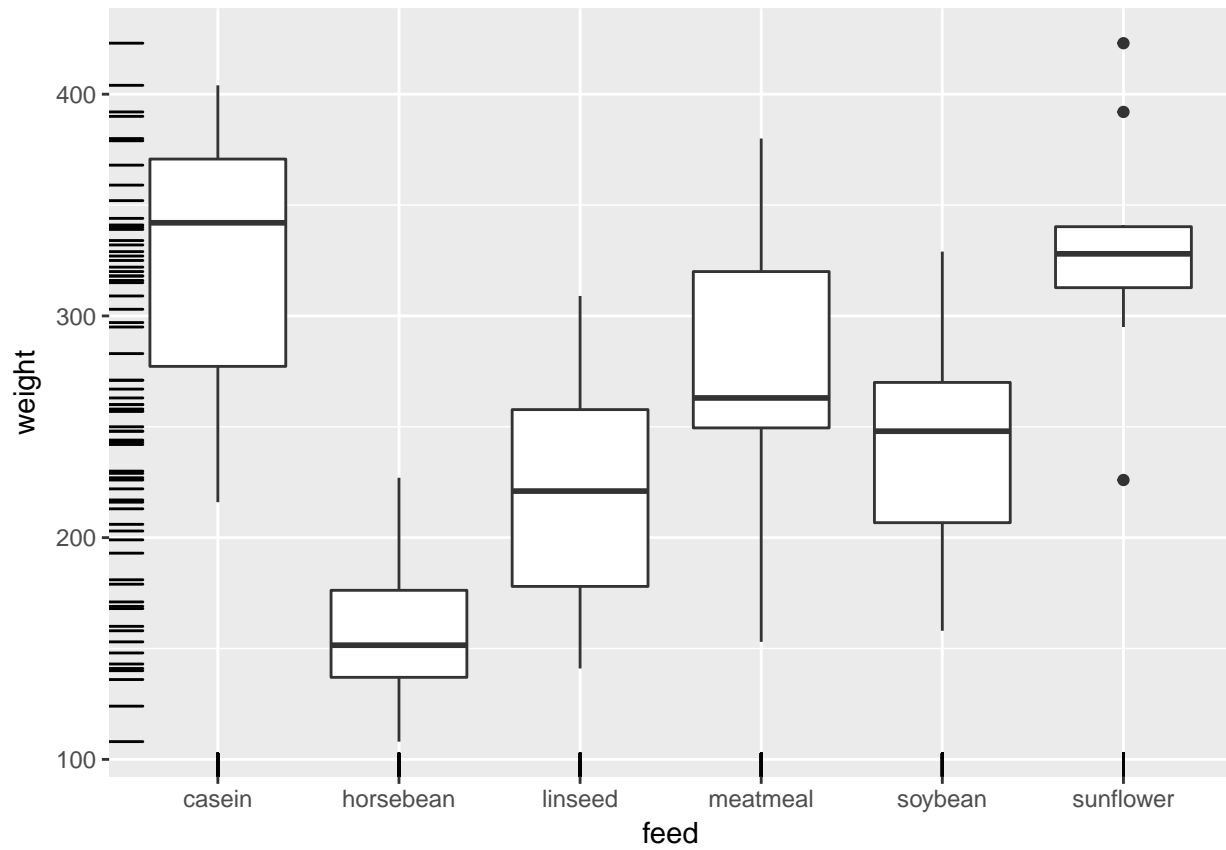


Superposer des éléments graphiques

Exemple autour du boxplot

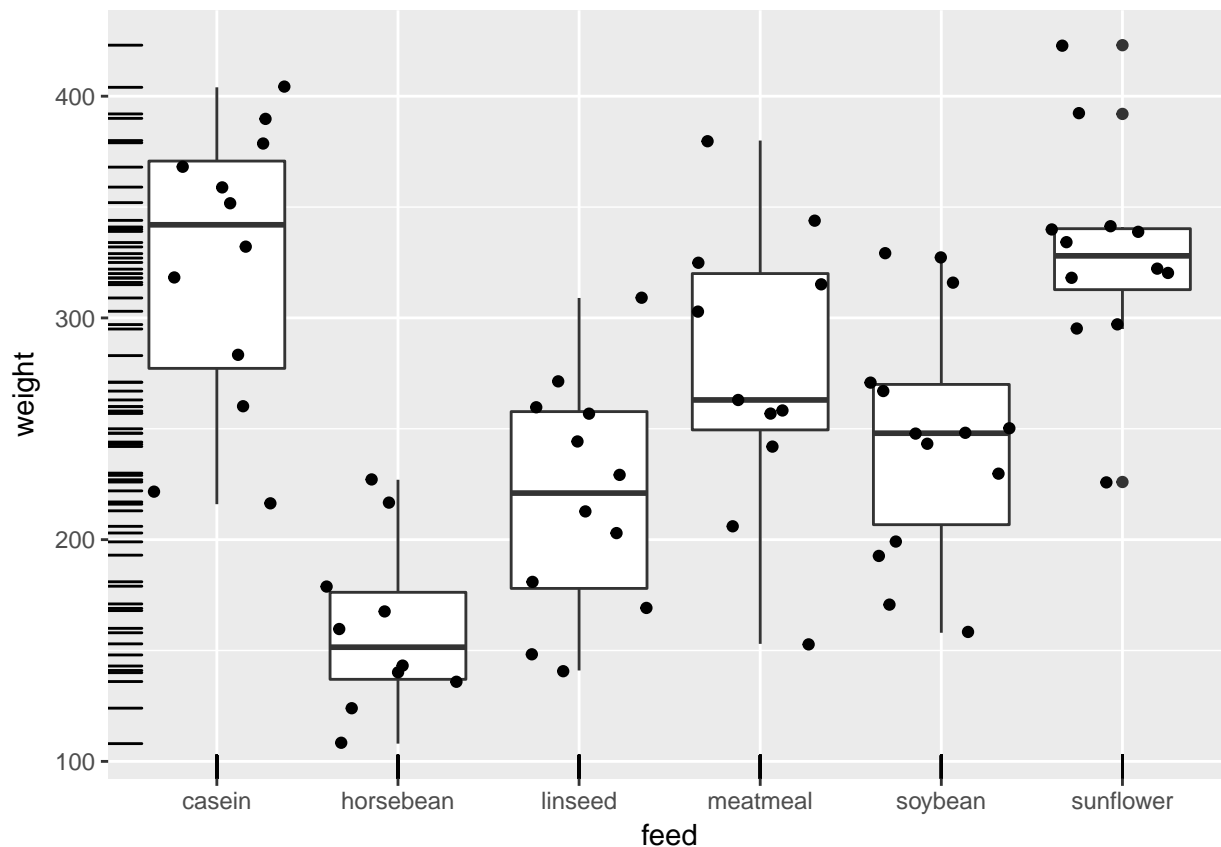
Le modèle sous-jacent à ggplot permet de superposer relativement facilement des couches graphiques.

```
## One can add a rug to the boxplot graphics  
p.bp + geom_rug()
```



```
## One can display the scattered values on the boxplot
```

```
p.bp + geom_jitter() + geom_rug()
```

Ajouter des barres d'erreurs

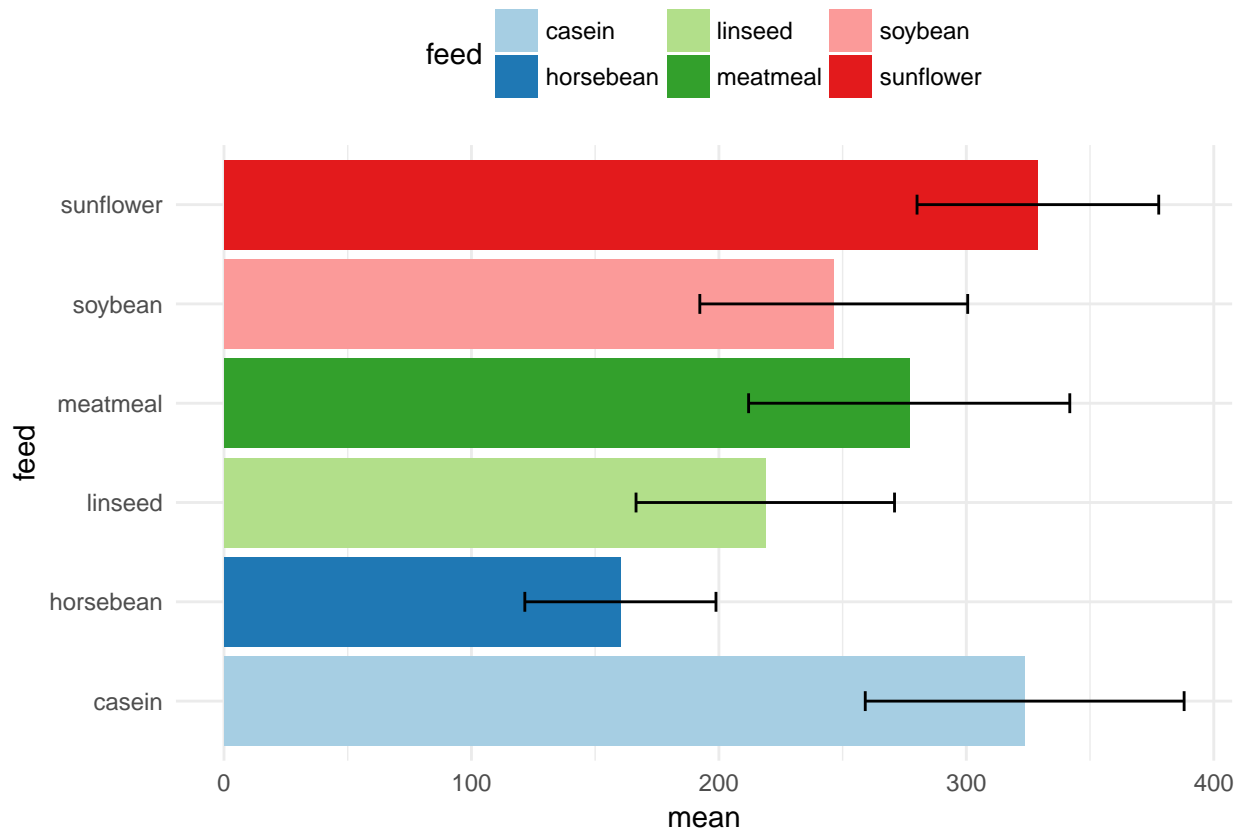
Pour présenter l'utilisation des barres d'erreurs nous allons tout d'abord calculer la moyenne et l'écart-type dans chacune des classes.

```
## First let's calculate the mean and standard deviation values
chickwts.mean <- tapply(chickwts$weight, chickwts$feed, mean)
chickwts.sd <- tapply(chickwts$weight, chickwts$feed, sd)
chickwts.summary <- data.frame(mean=chickwts.mean,
                                sd=chickwts.sd,
                                feed=names(chickwts.mean))
```

A partir de là, on peut produire le graphique.

```
ggplot(data=chickwts.summary,
       mapping=aes(x=feed, y=mean,
                   fill=feed)) +
  geom_bar(stat="identity") +
  geom_errorbar(mapping=aes(ymin=mean-sd, ymax=mean+sd),
               width=.2,
               position=position_dodge(.9)) +
  scale_fill_manual(values=c("#A6CEE3",
                              "#1F78B4",
                              "#B2DF8A",
                              "#33A02C",
                              "#FB9A99",
                              "#E31A1C")) +
```

```
theme_minimal() +
theme(legend.position="top") +
coord_flip()
```



Facettes

L'utilisation des facettes permet d'explorer les données en fonction d'un facteur ou d'un groupe de facteurs donnés. Pour l'exemple suivant nous allons créer une matrice contenant les résultats d'un test ELISA fictif dans lequel on mesure à 2 temps différents (jours) les expériences réalisés par quatre opérateurs différents.

```
url <- "https://tinyurl.com/ycdhmf8"
elisa <- read.table(url, sep="\t", header=TRUE, row.names=1)
head(elisa)
```

```
##   day rows columns value  user
## 1 Mon  cont      A  20.0 Alain
## 2 Mon   a      A  23.0 Alain
## 3 Mon   b      A  20.3 Alain
## 4 Mon   c      A  20.8 Alain
## 5 Mon   d      A  19.4 Alain
## 6 Mon   e      A  19.0 Alain
```

```
table(elisa$user, elisa$day)
```

```
##
##           Fri Mon
##  Alain      96  96
## Mathilde   96  96
```

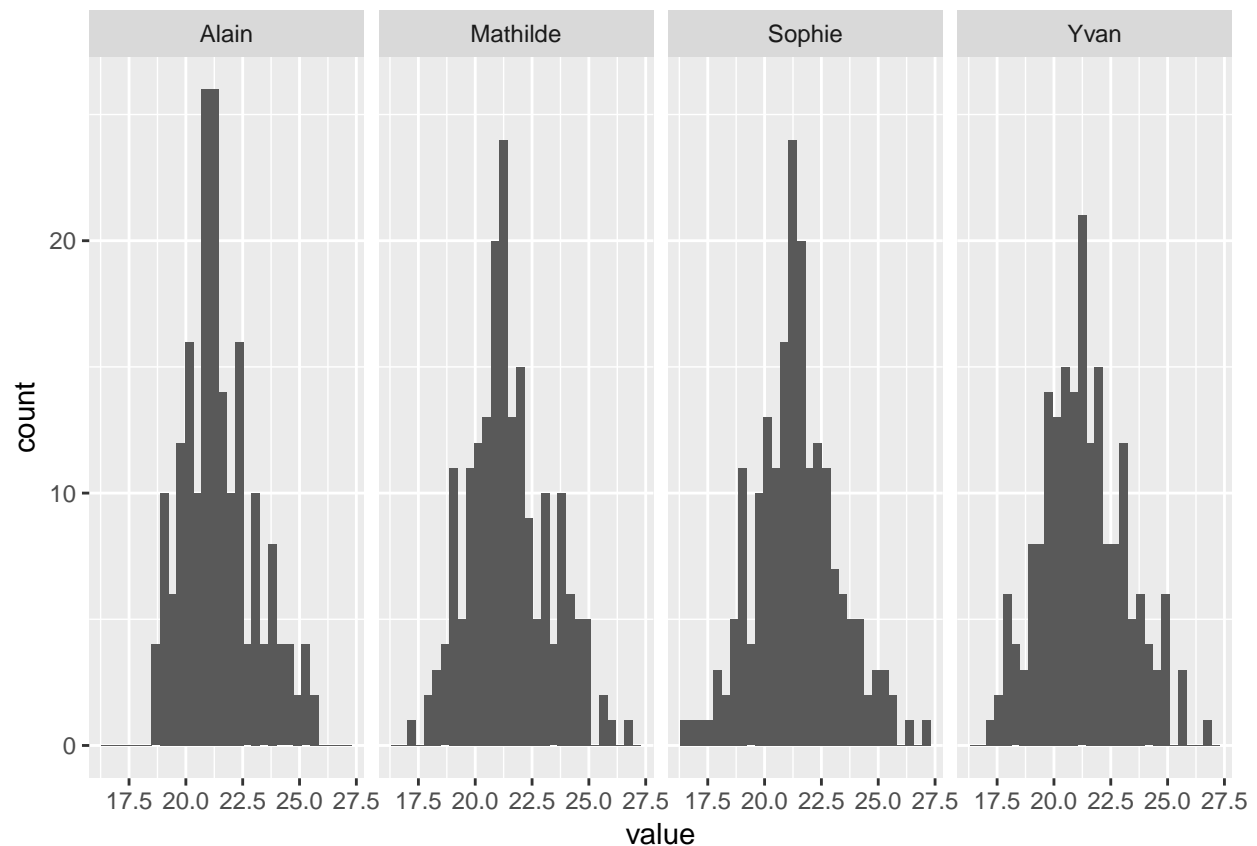
```
## Sophie 96 96
## Yvan 96 96
```

Histogrammes en facettes

Il devient très facile avec la syntaxe de ggplot2 de produire des histogrammes correspondant aux expériences réalisées à un jour donné par un expérimentateur donné.

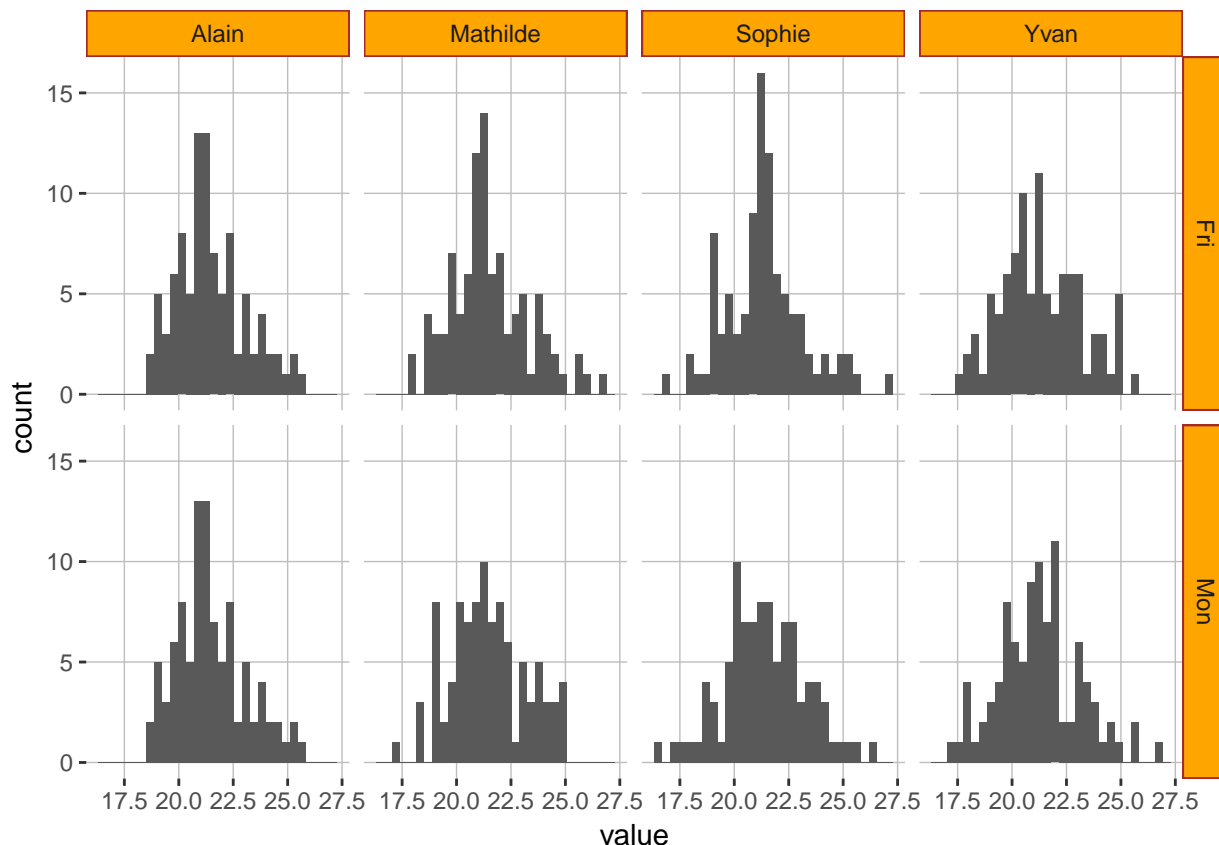
```
p <- ggplot(data = elisa, mapping = aes(x=value))
p + geom_histogram() + facet_grid(facets = ~ user )
```

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



```
p + geom_histogram() +
  facet_grid(facets = day ~ user) +
  theme(panel.background = element_rect(fill="white"),
        panel.grid.major = element_line(size = 0.25, linetype = 'solid', colour = "gray"),
        strip.background = element_rect(colour = "brown", fill="orange"),
        )
```

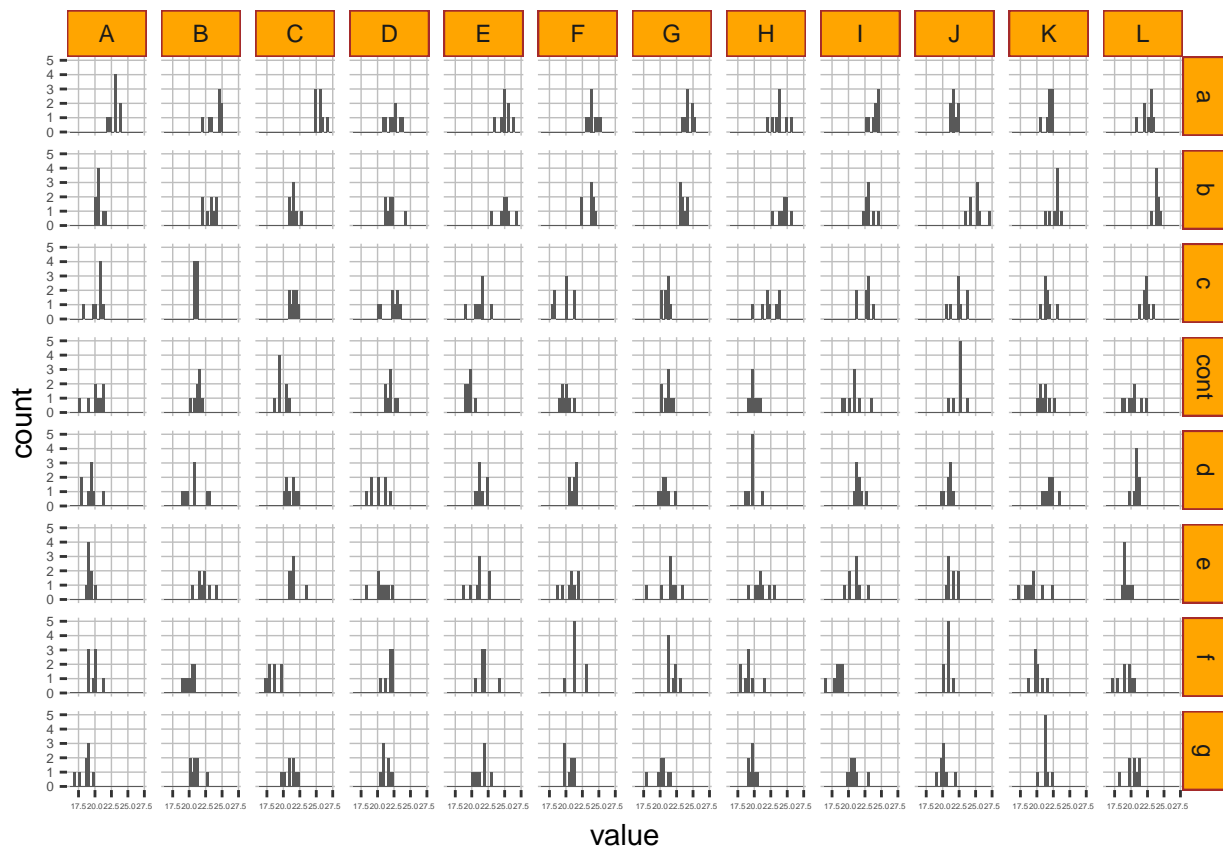
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Dans un but exploratoire, on peut, de même, analyser les résultats obtenues dans chaque ligne de l'ELISA en fonction des colonnes. Il suffit pour cela d'indiquer la formule suivante `rows ~ columns`. Dans l'exemple suivant on fait appel à la fonction `theme()` pour personnaliser un peu le graphique. Notez que chaque argument de la fonction `theme()` doit recevoir un objet renvoyé par les fonctions `element_rect()`, `element_text()`, ou `element_line()` qui correspondent respectivement à des éléments rectangulaires, du texte ou des lignes.

```
## Diagram faceted with rows and columns as variable.
p + geom_histogram() +
  facet_grid(facets = rows ~ columns) +
  theme(axis.text.x = element_text(size = 3),
        axis.text.y = element_text(size = 5),
        panel.background = element_rect(fill="white"),
        panel.grid.major = element_line(size = 0.25, linetype = 'solid', colour = "gray"),
        strip.background = element_rect(colour = "brown", fill="orange"))

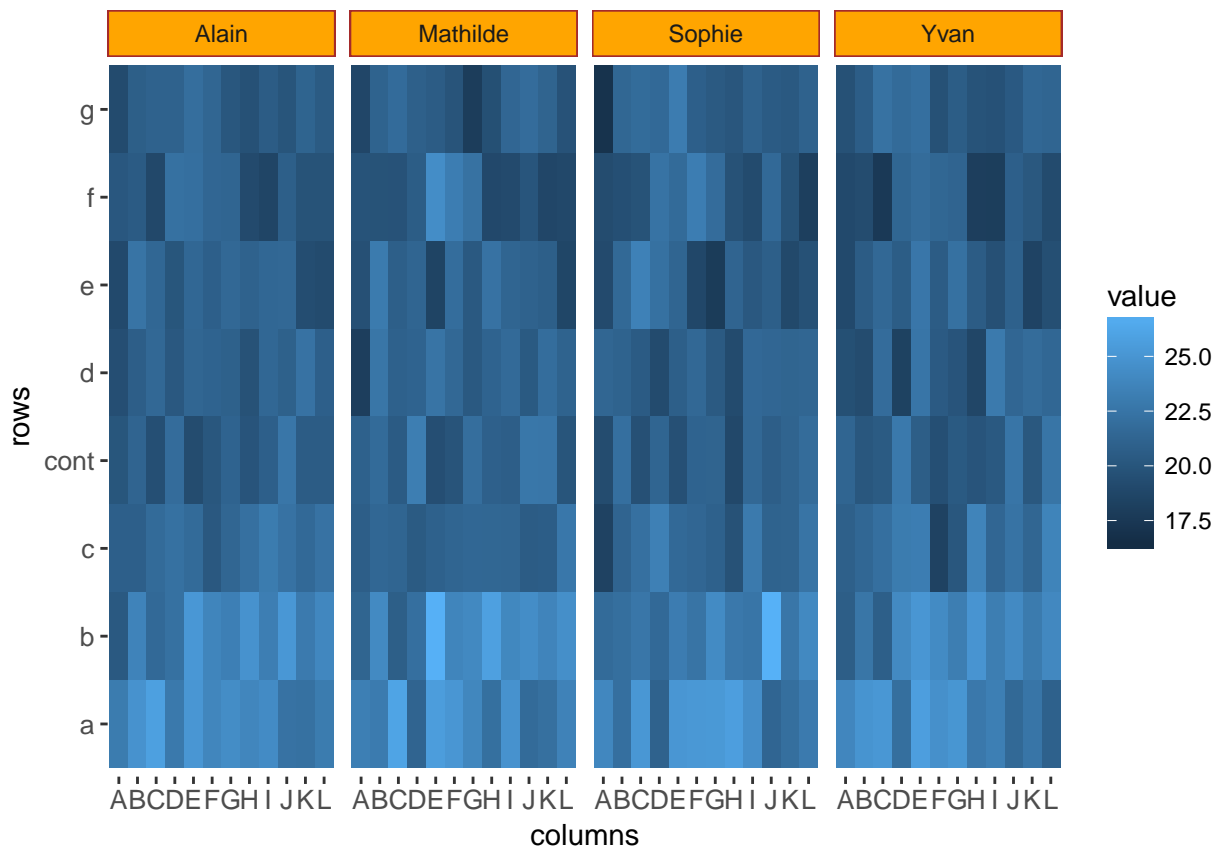
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Plans des plaques en facettes

A partir des données textuelles chargées dans R, on peut être intéressé à reproduire une image code-couleur (heatmap) des plaques ELISA produites par les différents utilisateurs. Pour ce faire il suffit d'utiliser la fonction `geom_raster()` et d'utiliser la formule adéquate.

```
p <- ggplot(data = elisa, mapping = aes(x=columns, y=rows, fill=value))
p + geom_raster() +
  facet_grid(facets = ~ user) +
  theme(axis.text.x = element_text(size = 10),
        axis.text.y = element_text(size = 10),
        panel.background = element_rect(fill="white"),
        panel.grid.major = element_line(colour = "white"),
        strip.background = element_rect(colour = "brown", fill="orange"))
```



Cependant, dans le diagramme précédent, les résultats des plaques ELISA sont mélangés... Il faut donc ajouter un facteur dans la formule, le jour.

```
p <- ggplot(data = elisa, mapping = aes(x=columns, y=rows, fill=value))
p + geom_raster() +
  facet_grid(facets = day ~ user) +
  theme(axis.text.x = element_text(size = 8, angle = 45, margin = margin(5)),
        axis.text.y = element_text(size = 10),
        panel.background = element_rect(fill="white"),
        panel.grid.major = element_line(colour = "white"),

        strip.background = element_rect(colour = "darkgray", fill="gray")) +
  scale_fill_gradientn(colours = c("#0000BF", "#0000FF", "#0080FF", "#00FFFF",
                                   "#40FFBF", "#80FF80", "#BFFF40", "#FFFF00",
                                   "#FF8000", "#FF0000", "#BF0000"))
```

