Représentation graphique de jeu de données en langage R

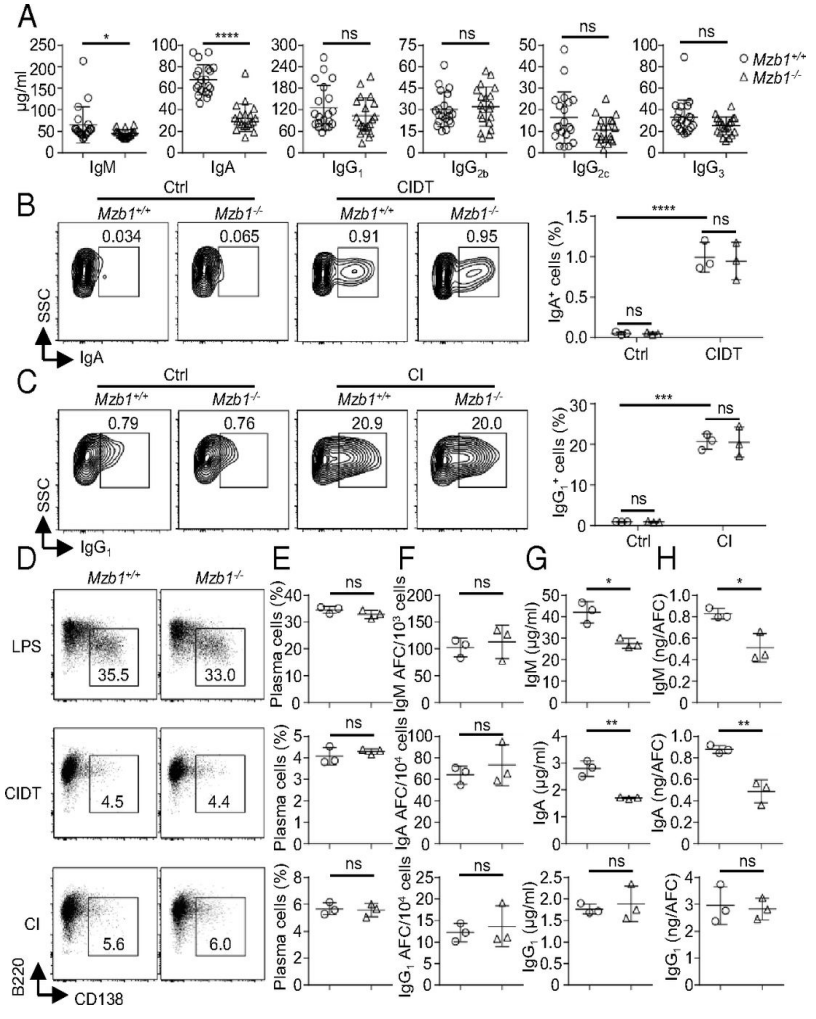
Rémy Villette

22 Octobre 2021

Mauvais exemple de représentation des données

- Enormément de graphiques différents
- Figure surchargée
- Pas de couleurs (les journaux peuvent facturé plus de 1000 euros supplémentaires pour l'impression en couleur)

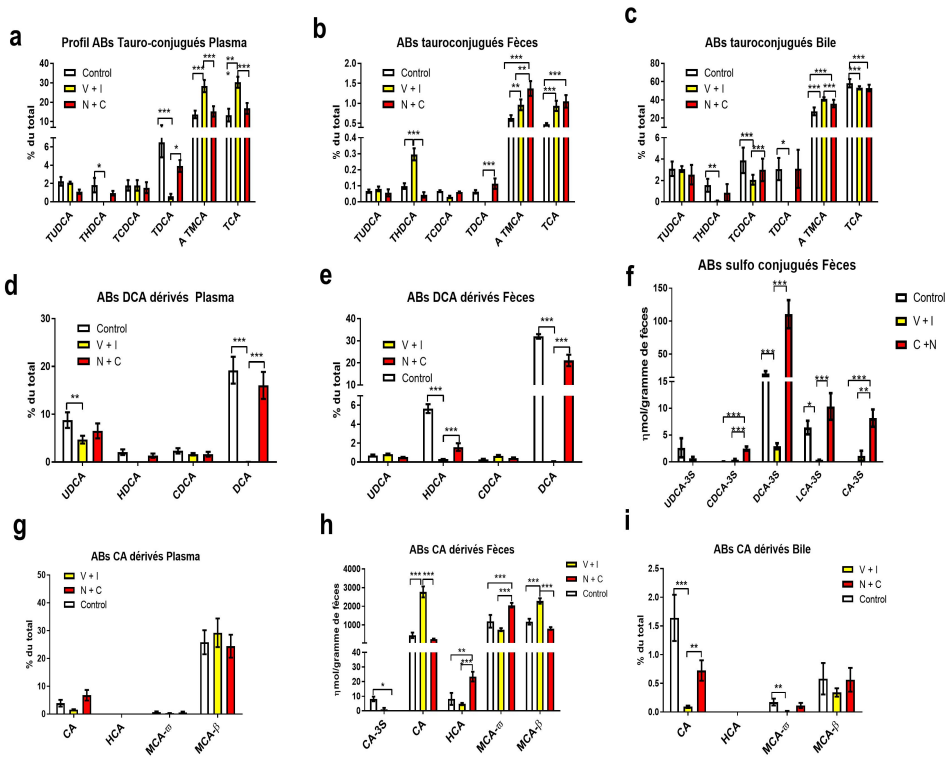
5/20 pour la note global (c'est pas fou)



MZB1 promotes the secretion of J-chain–containing dimeric IgA and is critical for the suppression of gut inflammation. Ermeng Xiong, et al., 2019

Mauvais exemple de représentation des données

Mes données de M2



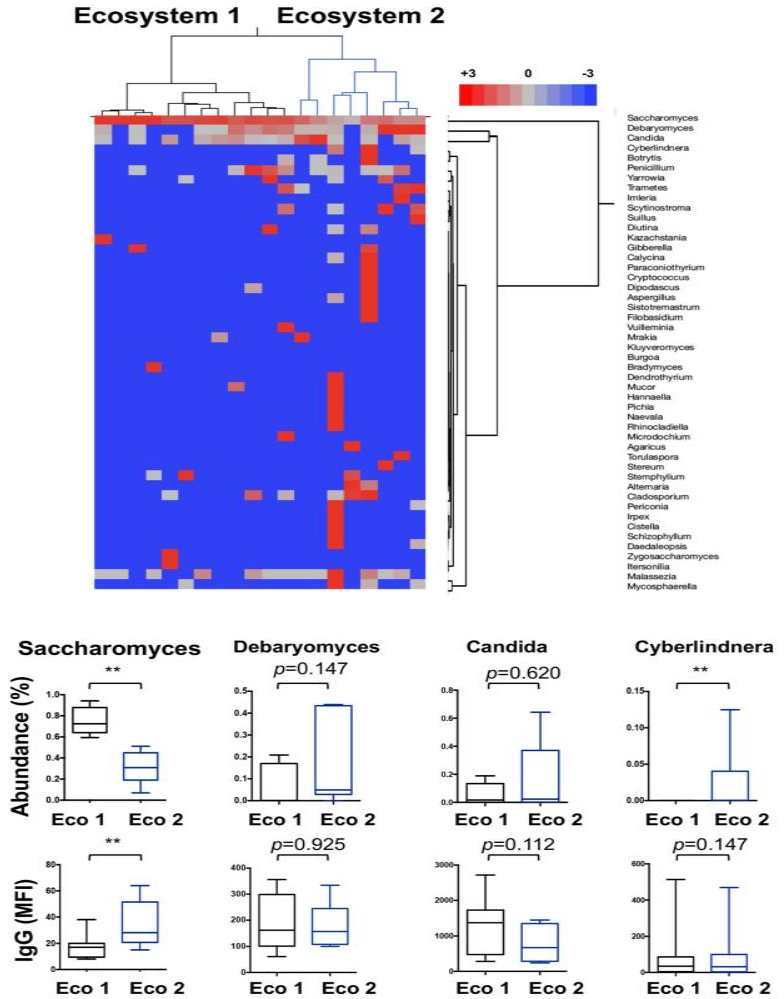
- Trop d'information !
- Pas de représentation des individus
  - Aucune idée de la répartition des individus dans chaque groupes
- Pas de couleurs (les journaux peuvent facturé plus de 1000 euros supplémentaires pour l'impression en couleur)

Graphiques en barre sont à proscrire (sauf pour les histogrammes de fréquences ou densité, mais ce ne sont pas des graphiques en barres mais des histogrammes) 7/20

Mauvais exemple de représentation des données

- Figure simple et intelligible
- Graphiques pas beaux
- Axes faux
- Pas de représentation des points

12/20 on se rapproche

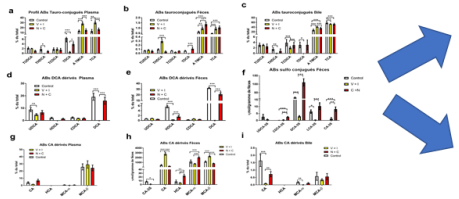


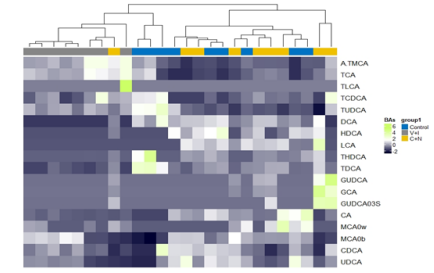
Moreno-Sabater, A. et al. Systemic anti-commensal response to fungi analyzed by flow cytometry is related to gut mycobiome ecology. Microbiome 8, 159 (2020).

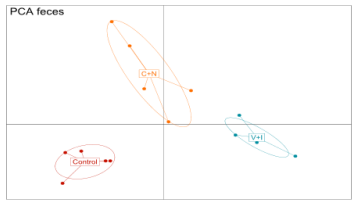
Comment bien faire ses graphiques scientifiques ?

- Travailler dans un logiciel de type R, Python ou MatPlotLib
  - Prism et Excel c'est éclaté au sol
  - R c'est Gucci
- Choisir les données les plus importantes
- Penser à la représentation multi-factorielle
- Heatmap : très simple !
- PCA : pour plus tard dans votre scolarité

Travailler l'esthétisme !



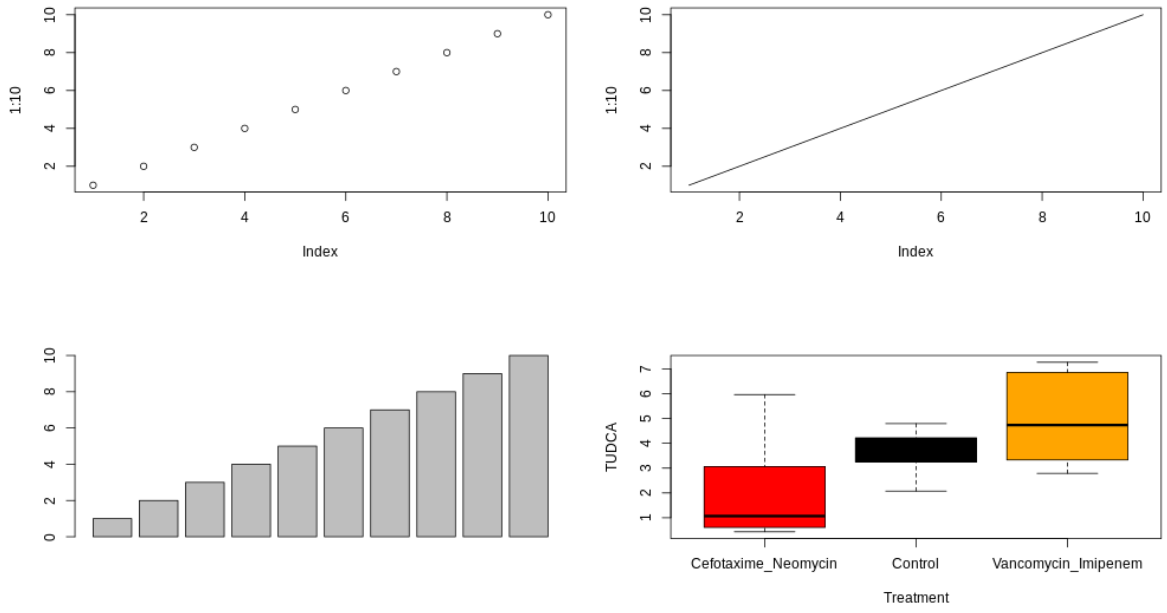




Les graphiques dans R

- Interface dans laquelle on donne les coordonnées x et y que l'on veut modéliser sur une grille
- Un vecteur x et un vecteur y
  - Premier exemple : 1 à 10 pour x et y
  - Deuxième exemple : Treatment (antibiotiques) pour x et TUDCA (un acide biliaire) pour y
- Ajout d'éléments graphiques
  - Type de graphique : points, ligne, barre, boîte à moustache, texte...
  - Paramètres graphiques : type de point, type de ligne, couleurs, type d'axes, ...

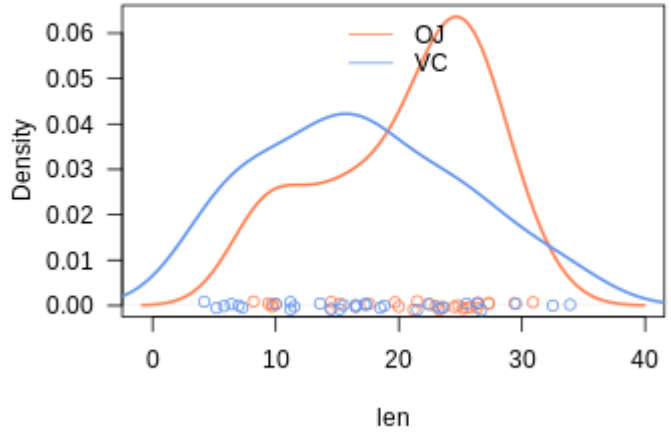
```
plot(1:10)
plot(1:10, type="l")
barplot(1:10)
boxplot(TUDCA~Treatment, data=csv, col=c("red","black","orange"))
```



Généralités (1), graphics

- graphics est le premier package de modélisation graphique apparu sur R.
- Il est développé par l'équipe Core de R.
- Utilisation s'avère assez compliquée
  - mais permet tout de même de faire des graphiques de très bonne qualité pour peu que vous sachiez manier sa grammaire.

```
plot(density(ToothGrowth$len[ToothGrowth$supp == "OJ"]), main = "", xlab = "len
lines(density(ToothGrowth$len[ToothGrowth$supp == "VC"]), lwd = 2, col = "cornf
points(
  x = ToothGrowth$len[ToothGrowth$supp == "OJ"],
  y = runif(length(ToothGrowth$len[ToothGrowth$supp == "OJ"]),
            min = -0.001, max = 0.001
  ), col = "coral"
)
points(
  x = ToothGrowth$len[ToothGrowth$supp == "VC"],
  y = runif(length(ToothGrowth$len[ToothGrowth$supp == "VC"]),
            min = -0.001, max = 0.001
  ), col = "cornflowerblue"
)
legend("top", levels(ToothGrowth$supp), col = c("coral", "cornflowerblue"), lty
```



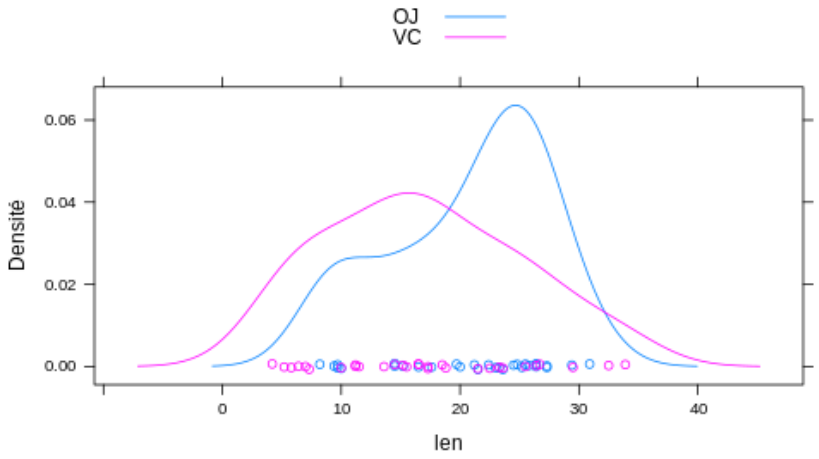
Généralités (2), lattice

`lattice` est un package crée en 2008 basé sur le package `grid`

- Il est plus simple à utiliser que le package `graphics`
- reste selon moi limité et inférieur au package `ggplot2`
    - sorti l'année précédente (2007).

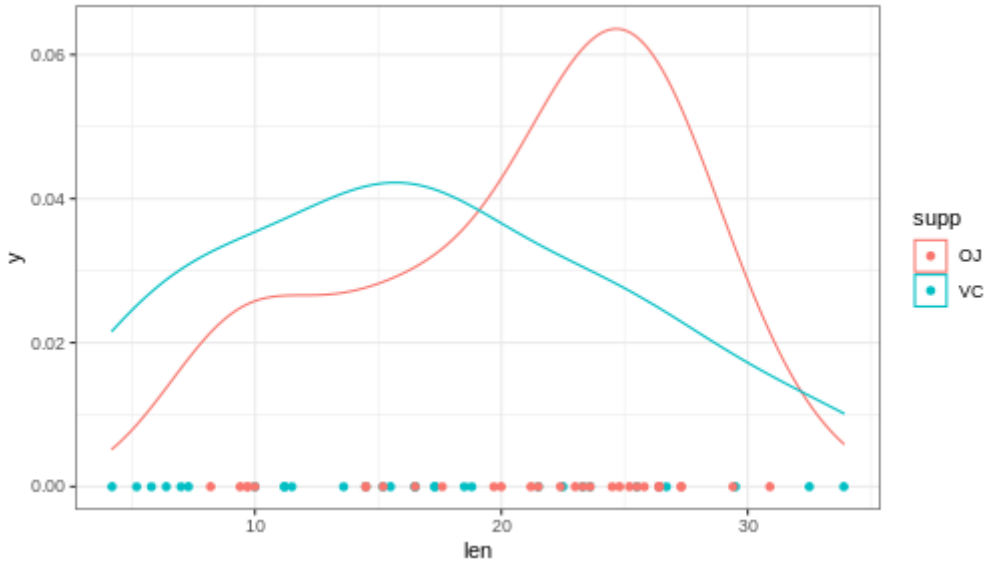
Bien que `lattice` rende le code plus simple que `graphics` il est plus compliqué à personnaliser que dans ggplot2 et donne de moins beaux graphiques.

```
densityplot(~len, data = ToothGrowth, group = supp, a
```



Globalement, la plupart des graphiques dans les publications sont soit issus de GraphPad Prism, de R avec `ggplot2` ou de python (pour les papiers très bioinfo).

```
ggplot(ToothGrowth, aes(x=len, color=supp))+geom_density(stat="density")+
  geom_point(aes(y=0))+
  theme_bw()
```



```
plot(density(ToothGrowth$len[ToothGrowth$supp == "OJ"]), main = "", xlab = "len", las = 1, lwd = 2, col = "coral")
lines(density(ToothGrowth$len[ToothGrowth$supp == "VC"]), lwd = 2, col = "cornflowerblue")
points(
  x = ToothGrowth$len[ToothGrowth$supp == "OJ"],
  y = runif(length(ToothGrowth$len[ToothGrowth$supp == "OJ"]),
            min = -0.001, max = 0.001
  ), col = "coral"
)
points(
  x = ToothGrowth$len[ToothGrowth$supp == "VC"],
  y = runif(length(ToothGrowth$len[ToothGrowth$supp == "VC"]),
            min = -0.001, max = 0.001
  ), col = "cornflowerblue"
)
legend("top", levels(ToothGrowth$supp), col = c("coral", "cornflowerblue"), lty = 1, bty = "n")
```

```
densityplot(~len, data = ToothGrowth, group = supp, auto.key = TRUE)

ggplot(ToothGrowth, aes(x=len, color=supp))+geom_density(stat="density")+geom_point(aes(y=0))+theme_bw()
```

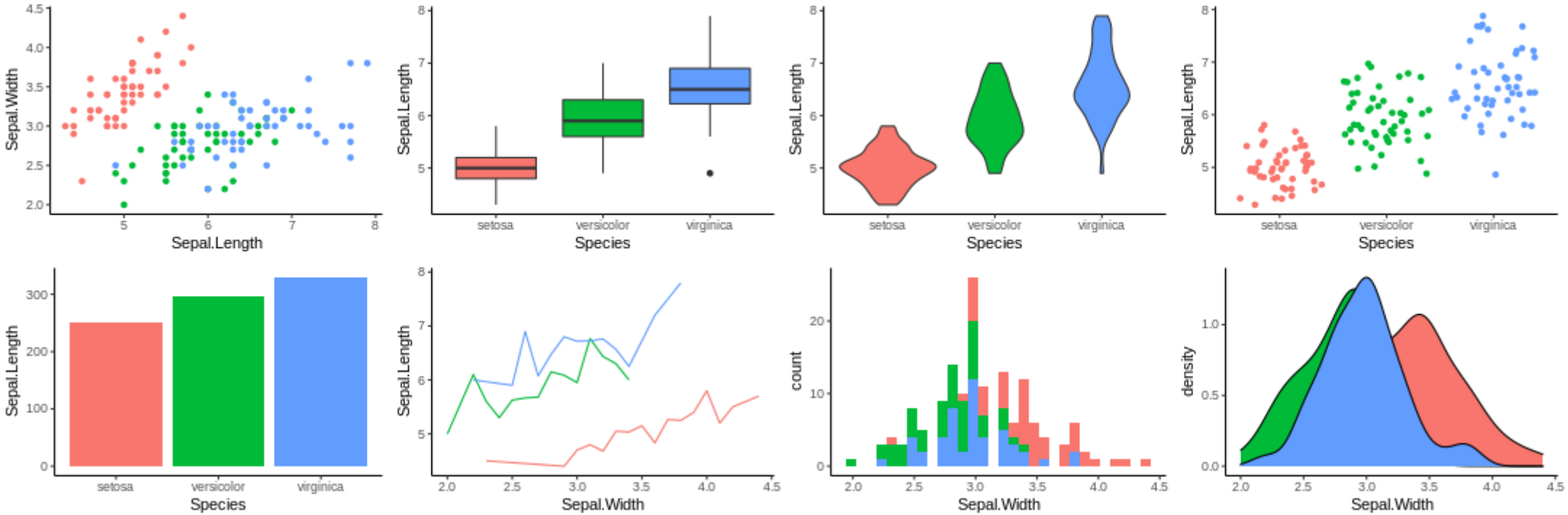
Introduction à ggplot2

- La base de `ggplot2` :
- Appeler la fonction `ggplot()`
- Donner le dataset et les éléments :
  - x
  - y
  - Le vecteur servant à colorier les contours et le remplissage
- Ajouter le `geom`
  - `geom` renvoie à la géométrie que vous voulez donner à votre graphique

Plot types	GGPlot2 functions
Initialize a ggplot	ggplot()
Scatter plot	geom_point()
Box plot	geom_boxplot()
Violin plot	geom_violin()
strip chart	geom_jitter()
Dot plot	geom_dotplot()
Bar chart	geom_bar()
Line plot	geom_line()
Histogram	geom_histogram()
Density plot	geom_density()
Error bars	geom_errorbar()

Il existe de nombreux types de graphiques possibles, les plus utilisés sont les suivants :

```
theme_set(theme_classic())
p = ggplot(data = iris, aes(x = Sepal.Length, y= Sepal.Width, color= Species))
p1= p + geom_point()
p2 = ggplot(data = iris, aes(x = Species, y= Sepal.Length, fill=Species))+ geom_boxplot()
p3 = ggplot(data = iris, aes(x = Species, y= Sepal.Length, fill=Species))+ geom_violin()
p4 = ggplot(data = iris, aes(x = Species, y= Sepal.Length, color=Species))+ geom_jitter()
p5 = ggplot(data = iris, aes(x = Species, y= Sepal.Length, fill=Species))+ geom_bar(stat = "identity")
p6 = ggplot(data = iris, aes(x = Sepal.Width, y= Sepal.Length, color=Species, group=Species))+ stat_summary(geom = 'line')
p7 = ggplot(data = iris, aes(x = Sepal.Width, fill=Species))+ geom_histogram()
p8 = ggplot(data = iris, aes(x = Sepal.Width, fill=Species))+ geom_density()
```



Accumuler les types de graphiques en un graphique

Il est possible d'accumuler les types de graphiques sur le même graphique. Il suffit pour cela d'ajouter avec un + les graphiques.

Attention l'ordre est important puisque le dernier `geom` ajouté sera celui au premier plan. Il est également possible de jouer avec la transparence des éléments `geom` en utilisant `alpha=`.

Le graphique le plus utiliser est le `geom_point` et le `geom_jitter`, personnellement j'aime beaucoup ajouter soit `geom_boxplot` ou `geom_violin`, voir les deux.

```
ggplot(data = iris, aes(x = Species, y= Sepal.Length, fill=Species))+
  geom_boxplot()+
  geom_violin()+
  geom_jitter()
```

Exercice pratique (1)

En utilisant le dataset cereal.csv :

- Faire un graphique montrant les calories pour chaque fabriquant (manufacturer), avec des jitters et un boxplot
- Ajouter de la couleur
    - De remplissage
    - De contour
    - Mettre `alpha=0.3`
- Faire un graphique montrant le ratio calories/fibre, avec les mêmes paramètres
- Faire un graphique montrant la relation entre les calories et la note d'appréciation des céréales (rating)
- Tester les *theme* :
    - `theme_classic()`
    - `theme_bw()`
    - `theme_void()`
    - `theme_minimal()`

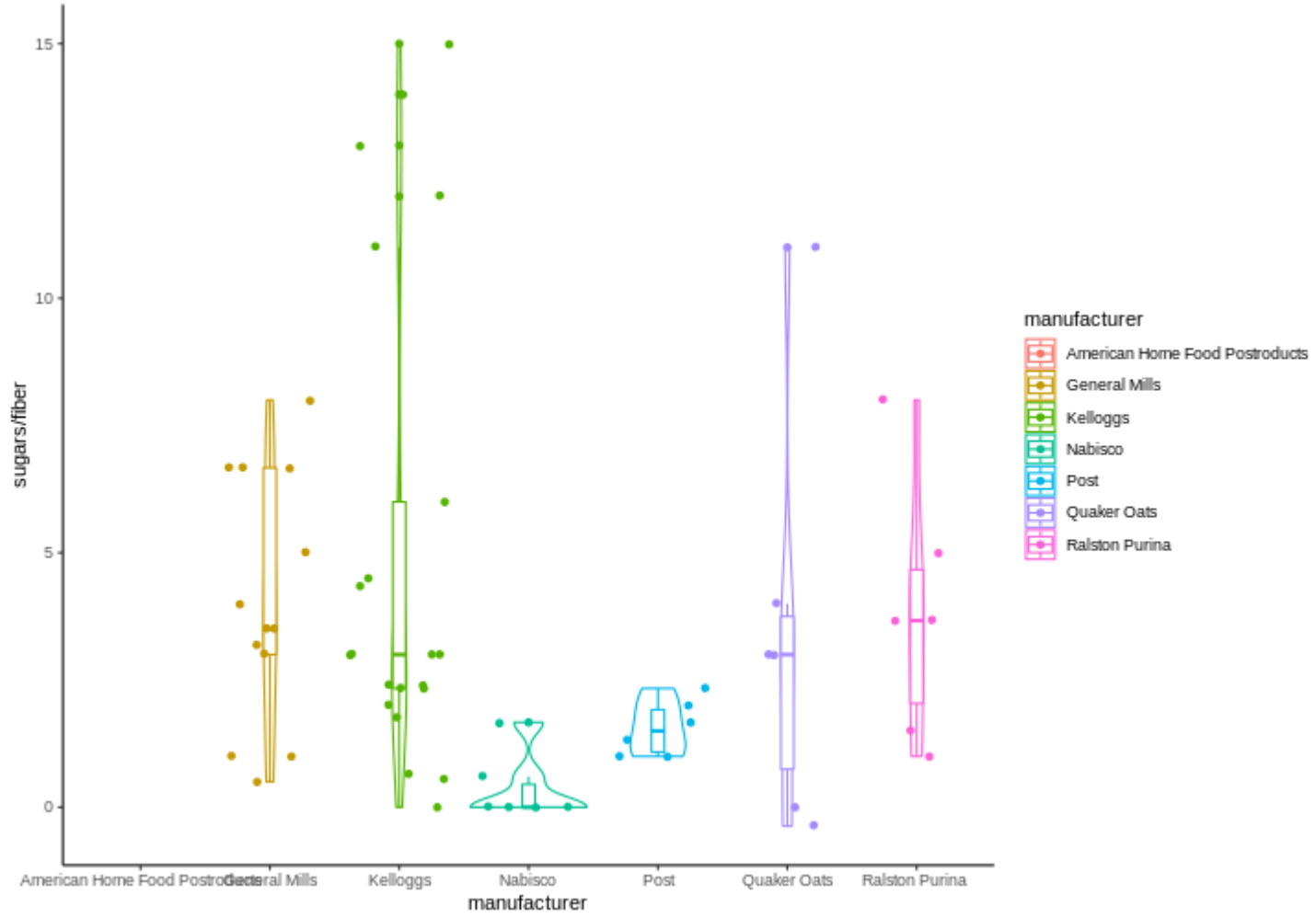
Exemple sur le dataset *cereal* (1)

Réponse 1

```
ggplot(data=cereal, aes(x= manufacturer, y= calories, color=manufacturer))+ geom_violin()+ geom_boxplot(width=0.1)+geom_jitter()
```

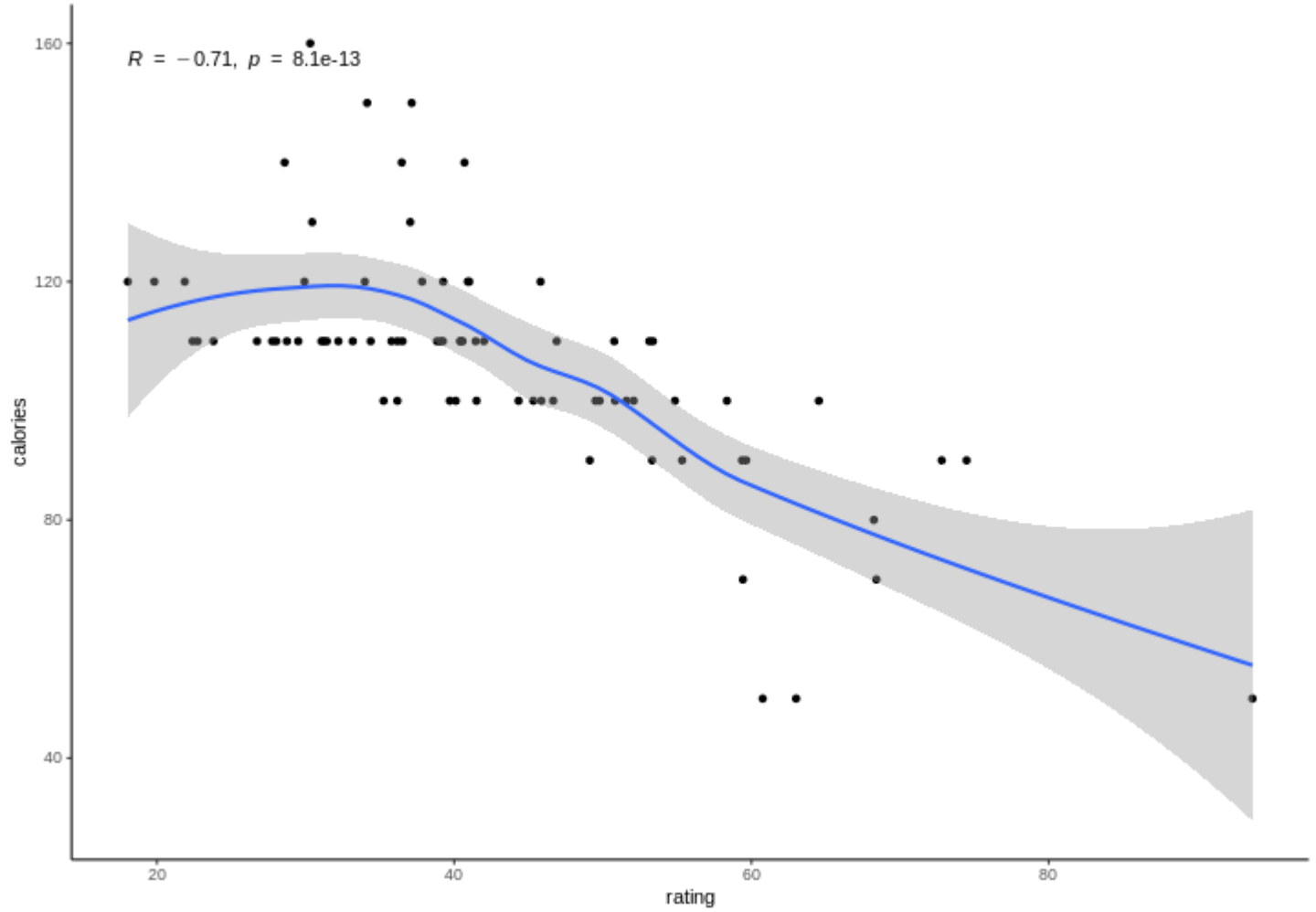
Réponse 2

```
ggplot(data=cereal, aes(x= manufacturer, y= sugars/fiber, color=manufacturer))+ geom_violin()+ geom_boxplot(width=0.1)+geom_jitter()
```



Réponse 3

```
ggplot(data=cereal, aes(x= rating, y= calories))+ geom_point()+geom_smooth()+stat_cor(method = "spearman")
```



Exemple sur le dataset *cereal* (2)

Il est possible d'effectuer directement des opérations dans `aes()`, ici pour faire le ratio sucres sur fibres il suffit de faire `sugars/fibers`

```
ggplot(data=cereal, aes(x= manufacturer, y= sugars/fiber, color=manufacturer))+ geom_violin()+
  geom_boxplot(width=0.1)+
  geom_jitter()+
  theme(legend.position = "bottom", axis.text.x = element_text(angle=30, face="bold", hjust=1))
```

Exemple sur le dataset *cereal* (3)

Il est possible de changer l'ordre des labels et les couleurs utilisées pour nos facteurs. Les fonctions commencent toutes par : `scale_`.

Pour les couleurs de remplissage utiliser : `scale_fill_` et pour les couleurs de contours utiliser : `scale_color`.

```
p = ggplot(data=cereal, aes(x= manufacturer, y= sugars/fiber, color=manufacturer))+ geom_violin()+
geom_boxplot(width=0.05)+
geom_jitter(size=3)
p+ scale_x_discrete(limits=c("Nabisco", "Quaker Oats","Kelloggs","Ralston Purina ", "General Mills","Post", "American Home Food Postr
  scale_color_manual(values= c("bisque", "thistle", "snow3", "burlywood4", "lightblue2", "salmon", "palegoldenrod"))+
theme(legend.position = "bottom", axis.text.x = element_text(angle=30, face="bold", hjust=1))
```

Exemple sur le dataset *cereal* (4)

Grâce à `ggplot` il est possible :

D'avoir des données en format large, comme c'est le cas pour notre jeu de donnée de base.

	name	manufacturer	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins
1	100% Bran	Nabisco	Cold	70	4	1	130	10	5	6	280	
2	100% Nabiscoatural Bran	Quaker Oats	Cold	120	3	5	15	2	8	8	135	
3	All-Bran	Kelloggs	Cold	70	4	1	260	9	7	5	320	
4	All-Bran with Extra Fiber	Kelloggs	Cold	50	4	0	140	14	8	0	330	
5	Almond Delight	Ralston Purina	Cold	110	2	2	200	1	14	8	-1	
6	Apple Cinnamon Cheerios	General Mills	Cold	110	2	2	180	1.5	10.5	10	70	
7	Apple Jacks	Kelloggs	Cold	110	2	0	125	1	11	14	30	
8	Basic 4	General Mills	Cold	130	3	2	210	2	18	8	100	
9	Bran Chex	Ralston Purina	Cold	90	2	1	200	4	15	6	125	
10	Bran Flakes	Post	Cold	90	3	0	210	5	13	5	190	

Mais également de passer en données format long, ici on obtient une colonne regroupant tous nos facteurs et une colonne regroupant toutes nos valeurs.

```
cereal_long = pivot_longer(cereal, cols = calories:rating, names_to = "factor", values_to = "value")

tmp = pivot_wider(cereal_long, names_from = factor, values_from = value)
```

Show 10 entries Search:

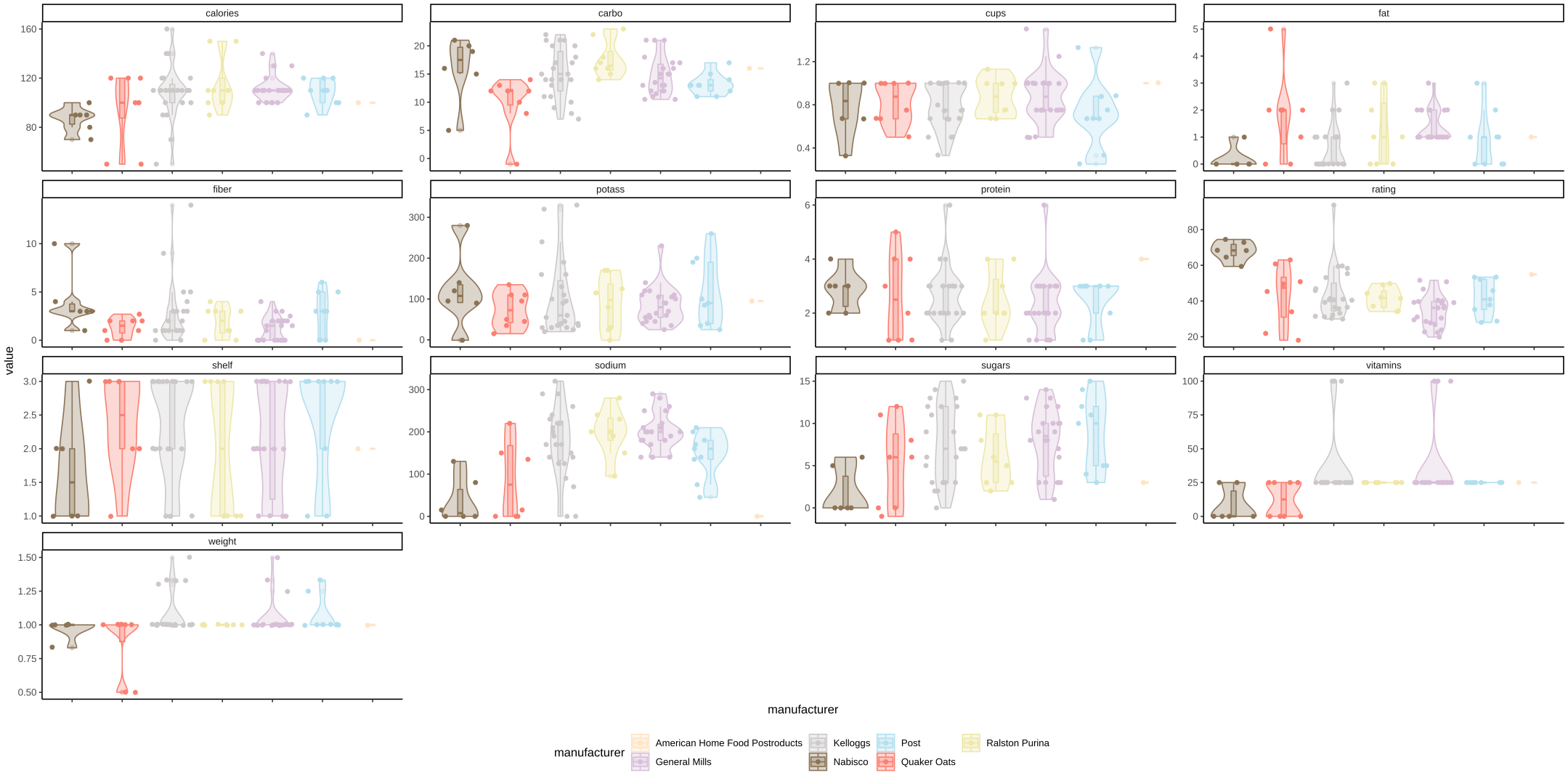
	name	manufacturer	type	factor	value
1	100% Bran	Nabisco	Cold	calories	70
2	100% Bran	Nabisco	Cold	protein	4
3	100% Bran	Nabisco	Cold	fat	1
4	100% Bran	Nabisco	Cold	sodium	130
5	100% Bran	Nabisco	Cold	fiber	10
6	100% Bran	Nabisco	Cold	carbo	5
7	100% Bran	Nabisco	Cold	sugars	6
8	100% Bran	Nabisco	Cold	potass	280
9	100% Bran	Nabisco	Cold	vitamins	25
10	100% Bran	Nabisco	Cold	shelf	3

Showing 1 to 10 of 1,001 entries Previous 1 2 3 4 5 … 101 Next

Quel sont les éléments que nous allons utiliser pour mettre nos nouvelles données en graphique ?

Exemple sur le dataset *cereal* (5)

```
ggplot(data=cereal_long, aes(x= manufacturer, y= value, color=manufacturer, fill=manufacturer))
```



Exemple sur le dataset *cereal* (6)

```
p= ggplot(data=cereal_long, aes(x= factor, y= value, color=factor, fill=factor))+   geom_violin(alpha=0.3)+
geom_boxplot(width=0.1, alpha=0.3)+
geom_jitter()+
facet_wrap(~manufacturer, scales = "free")+
theme(legend.position = "bottom", axis.text.x = element_text(angle=30, face="bold", hjust=1))
```

Les Heatmaps

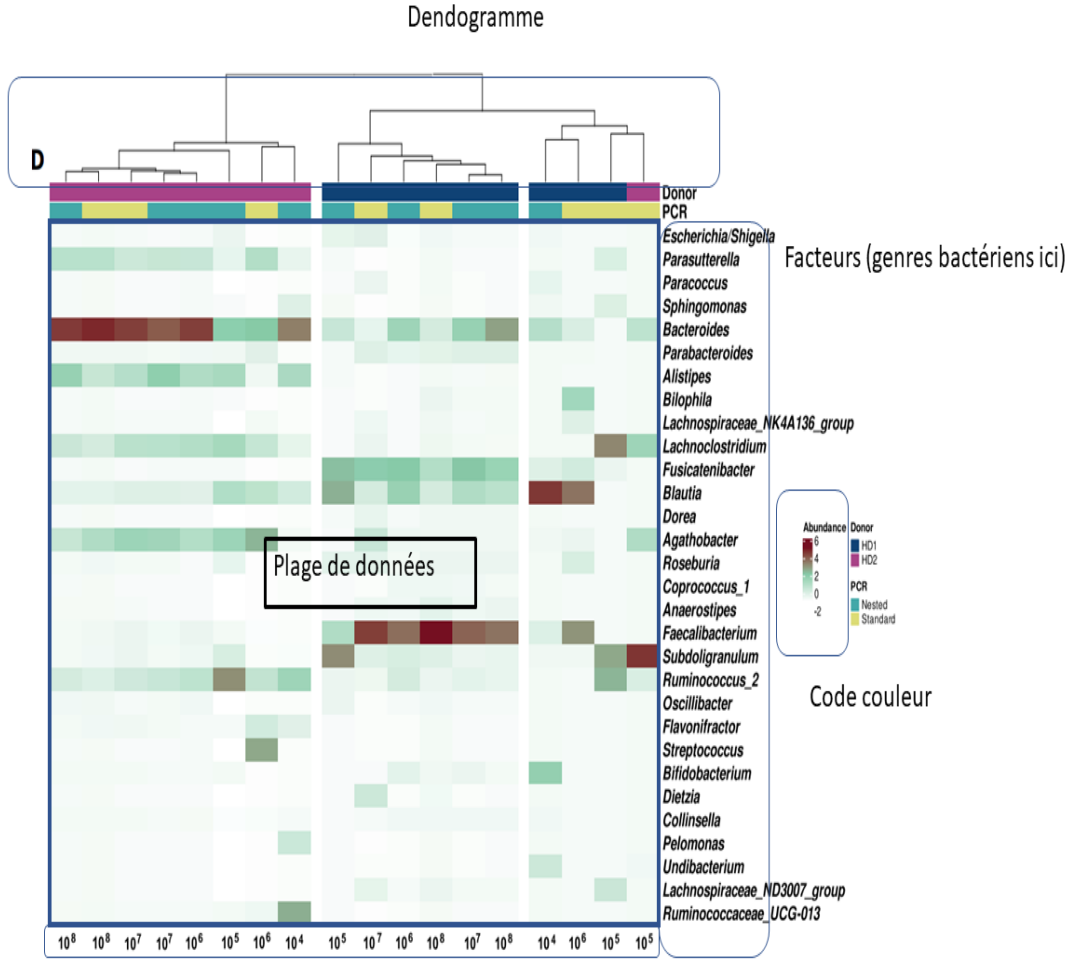
- Heatmap ou carte de chaleur en français
- Représentation d'une matrice montrant tous les facteurs d'un dataset pour chaque individu/ échantillon
- Système de couleur montrant la quantité de chaque facteur dans un échantillon par rapport au min et max du dataset
- Possibilité (recommandé) de classer les échantillons grâce à une clusterisation hiérarchique

Villette, R. et al. Refinement of 16S rRNA gene analysis for low biomass biospecimens. Sci Rep 11, 10741 (2021).

Eléments d'une heatmap

On va retrouver plusieurs éléments dans une heatmap:

- Les dendogrammes :
  - Un pour les colonnes et un optionnel pour les lignes
- Les labels de lignes et de colonnes
- La plage de données
- Le code couleur
- Des annotations (optionnel)



Construire sa heatmap

Plusieurs packages permettent de faire des heatmaps

- ggplot2 capable mais plus dur
  - ggheatmap, basé sur ggplot2, marche très bien
  - install.packages("heatmaply")
- Package base heatmap() : simple mais limité
- pheatmap : pas recommandé
- Heatmap2 : très beau mais assez limité
- ComplexHeatmap : mon package de choix
  - library(devtools)
  - install_github("jokergoo/ComplexHeatmap")

Que faut-il ?

- Une matrice ne contenant que des valeurs numériques
- Des vecteurs pour les colonnes et les lignes
- Et bien réfléchir

Dans notre dataset cereal de 77 par 16

- Quelles sont les colonnes numériques ?
- Quelles sont les colonnes factorielles/catégoriques ?

- Est-ce que nos données sont dans le même ordre de grandeur entre les différents facteurs ?

- Faites un max(cereal$sodium) et max(cereal$weight)

La classification hiérarchique

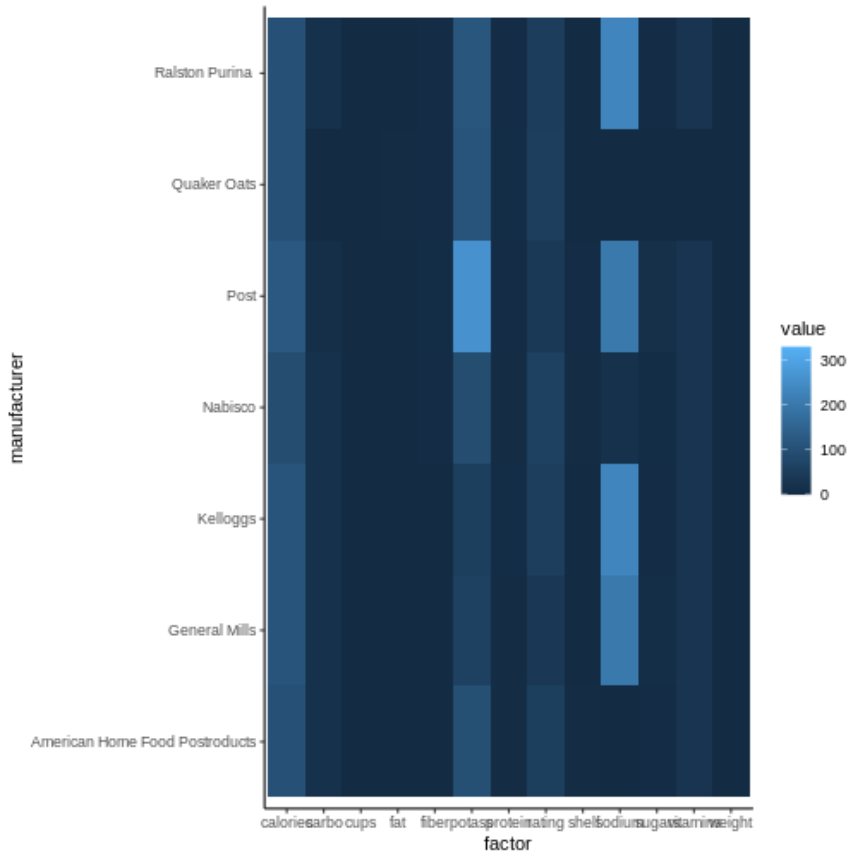
C'est un concept qui fait peur mais qui n'est pas si compliqué :

- Etablir la *différence* entre chaque échantillons/individus sur toutes les données/facteurs mesurés
    - Différence entre chaque facteurs entre l'individu 1 et 2
    - Différence entre chaque facteurs entre l'individu 1 et 3
    - ...
- Déterminer quels échantillons sont plus "proches" ou plutôt *homogènes* les uns par rapport aux autres

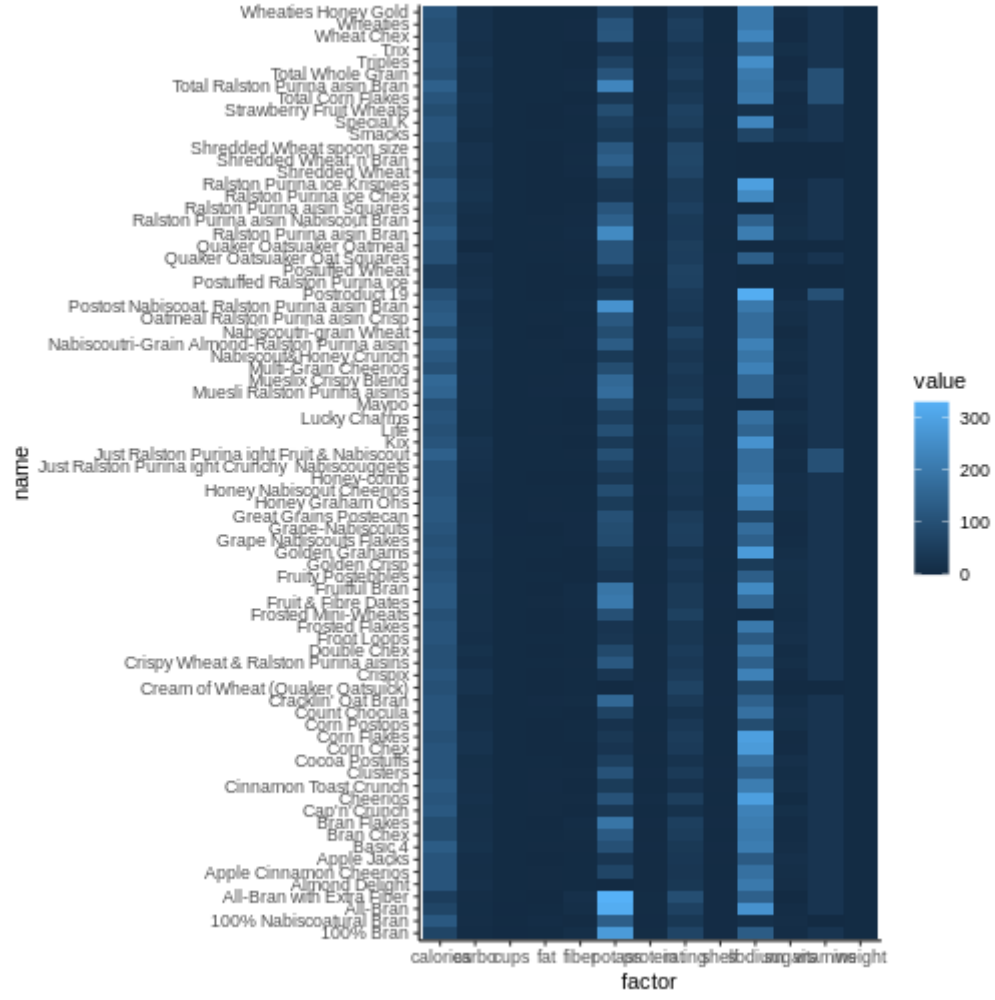
Quoiqu'il arrive vous verrez plus en détails ces algorythmes et méthode de représentation des données.

Heatmap avec ggplot2

```
ggplot(cereal_long, aes(factor, manufacturer, fill=va
  geom_tile()
```




```
ggplot(cereal_long, aes(factor, name, fill=value))+
  geom_tile()
```

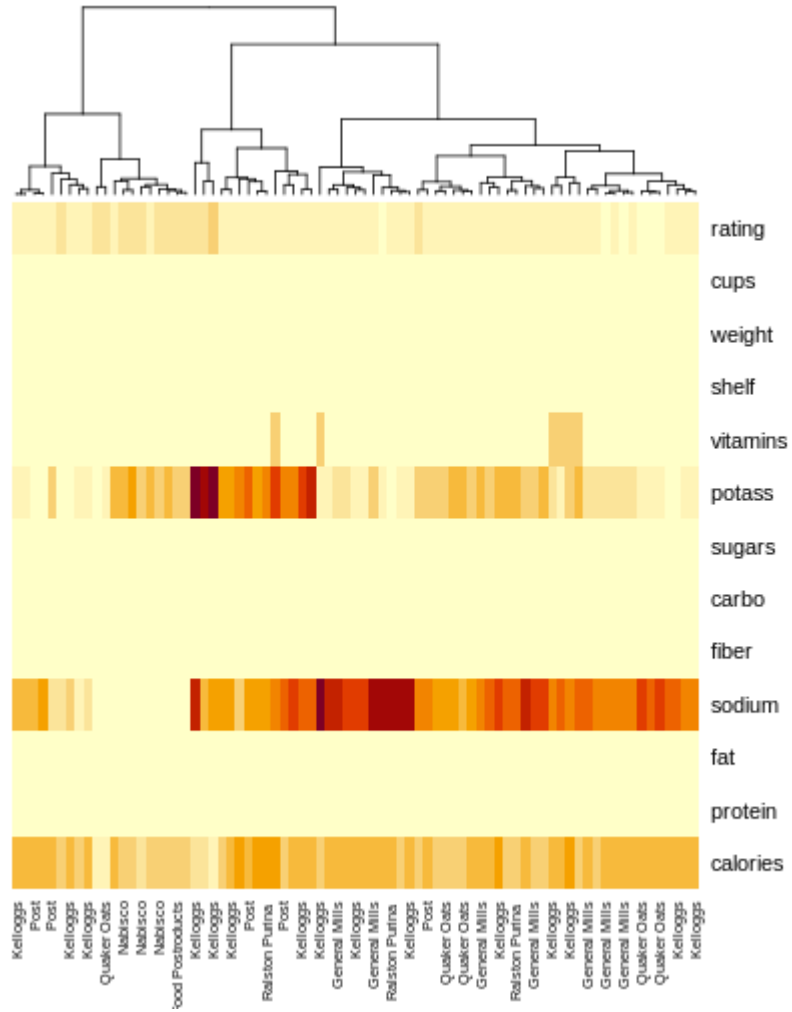


Heatmap avec une fonction "wrapper" : heatmaply

```
library(heatmaply)
p= ggheatmap(as.matrix(t(cereal[,4:16])), labCol = cereal$manufacturer,  hclustfun = function(x)hclust(x,"ward.D2"), Rowv=F)
```

Heatmap basé sur graphics

```
heatmap(as.matrix(t(cereal[,4:16])),  labCol = cereal$manufacturer, Rowv = NA, scale = "none", hclustfun = function(x)hclust(x,"ward.
```



Heatmap fait grâce à ComplexHeatmap

```
library(ComplexHeatmap)
tmp <- t(scale(as.matrix(csv[,3:32])))
Heatmap(tmp,  cluster_rows = F, row_names_gp = gpar(fontface="bold.italic"), name = "Acide biliaires", column_labels = csv$Treatment,
        column_names_gp = gpar(fontface="bold"), clustering_distance_columns = "euclidean", clustering_method_columns = "ward.D2")
```

Diapo bonus

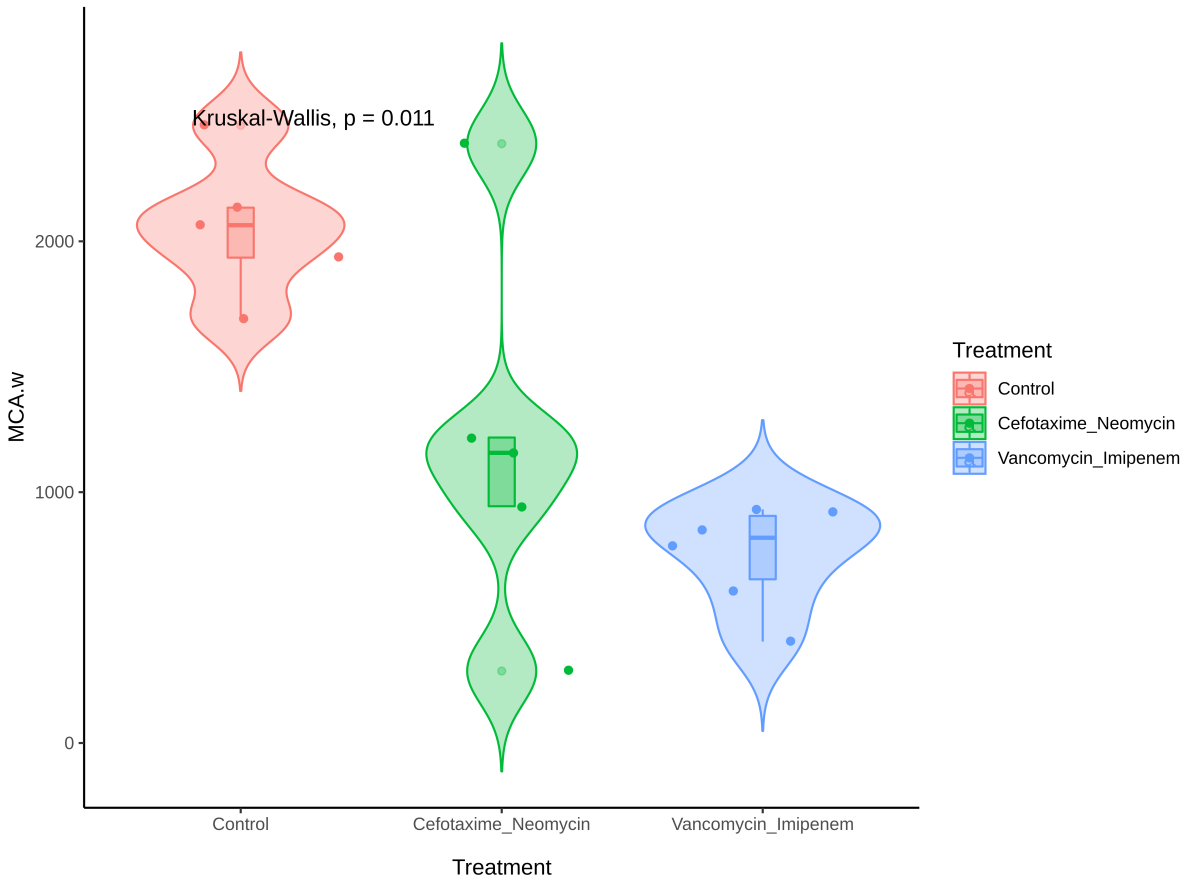
Maintenant qu'on a vu les graphiques on veut savoir si nous groupes sont statistiquement différents.

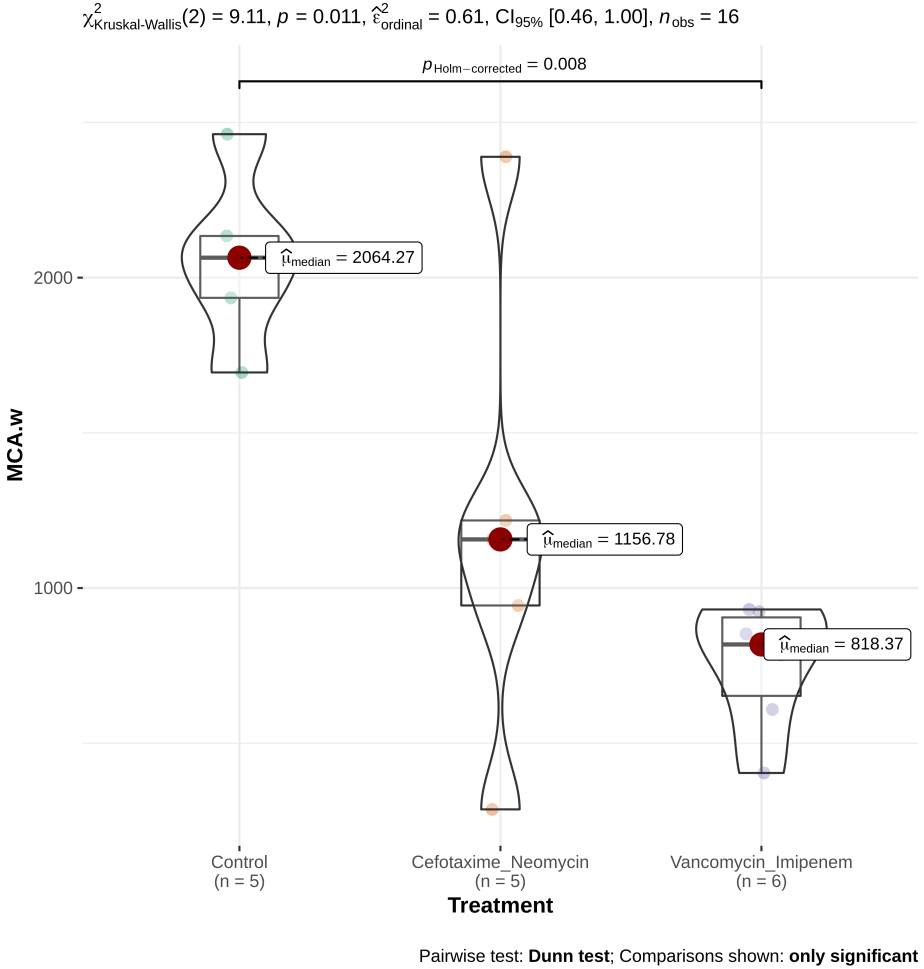
Il existe des packages basé sur ggplot qui permettent de faire les tests et mettre les statistiques directement sur le graphique.

- `ggpubr` qui va rajouter directement sur votre graphique
- `ggstatsplot` qui va créer ses propres graphiques basés sur `ggplot2`

```
# Donner le bon ordre à nos facteurs
csv$Treatment= factor(csv$Treatment, levels=c("Control","Cefotaxime_Neomycin","Vancomycin_Imipenem"))

p1 = ggplot(csv, aes(y= MCA.w, x=Treatment, fill=Treatment, color=Treatment))+geom_violin(alpha=0.3, trim = F)+
  geom_boxplot(alpha=0.3, width=0.1)+
  geom_jitter()

p2 = p1+ stat_compare_means()

p3 = ggstatsplot::ggbetweenstats(csv, x=Treatment, y=MCA.w, pairwise.comparisons = T, type = "nonparametric")
```

Résultat


Kruskal-Wallis, p = 0.011
MCA.w
2000
1000
0
Control
Cefotaxime_Neomycin
Vancomycin_Imipenem
Treatment
Treatment
Control
Cefotaxime_Neomycin
Vancomycin_Imipenem


$\chi^2_{\text{Kruskal-Wallis}}(2) = 9.11$, $p = 0.011$, $\hat{\varepsilon}^2_{\text{ordinal}} = 0.61$, $\text{CI}_{95\%}$ [0.46, 1.00], $n_{\text{obs}} = 16$
$p_{\text{Holm-corrected}} = 0.008$
$\hat{\mu}_{\text{median}} = 2064.27$
$\hat{\mu}_{\text{median}} = 1156.78$
$\hat{\mu}_{\text{median}} = 818.37$
MCA.w
2000
1000
Control
(n = 5)
Cefotaxime_Neomycin
(n = 5)
Vancomycin_Imipenem
(n = 6)
Treatment
Pairwise test: **Dunn test**; Comparisons shown: **only significant**