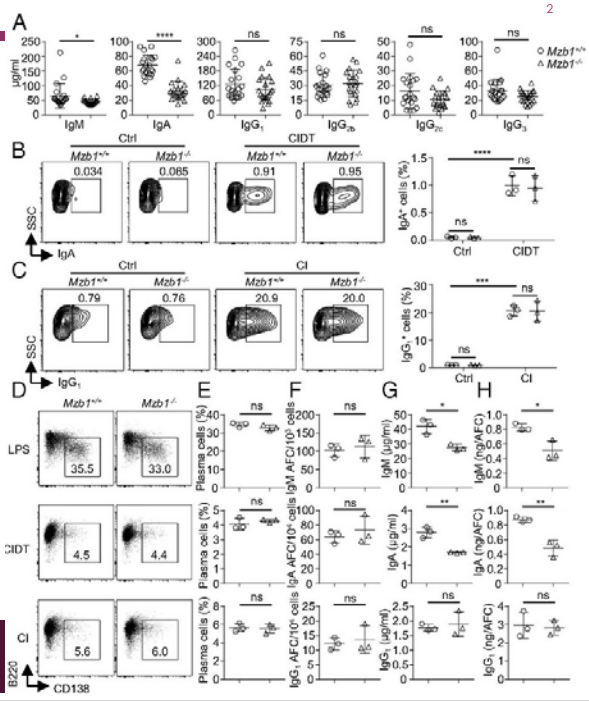
REPRÉSENTATION DE DONNÉES EN LANGAGE R

RÉMY VILLETTE

PHD STUDENT

MAUVAIS EXEMPLE DE REPRÉSENTATION DES DONNÉES

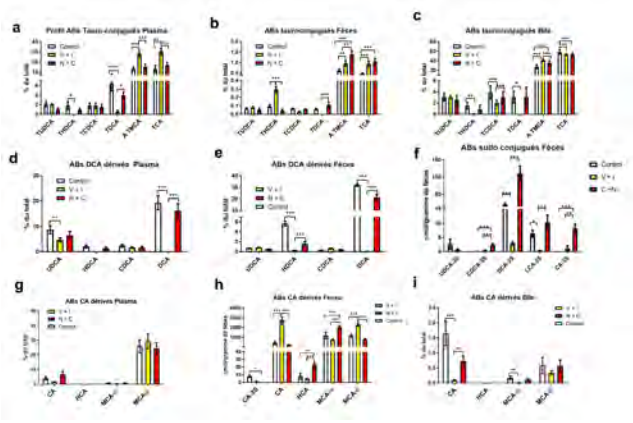
- Enormément de graphiques différents
- Figure surchargée
- Pas de couleurs (les journaux peuvent facturé plus de 1000 euros supplémentaires pour l'impression en couleur)
- 5/20



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

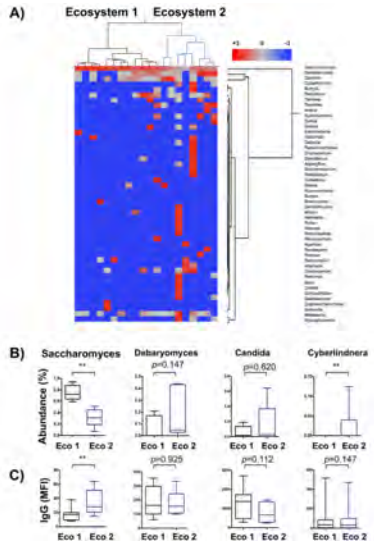
AUTRE EXEMPLE DE MAUVAISE REPRÉSENTATION GRAPHIQUE

- Trop d'information !
- Pas de représentation des individus
  - Aucune idée de la répartition des individus dans chaque groupes
    - 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



BONNE IDÉE MAUVAISE RÉALISATION

- Figure simple et intelligible
- Graphiques pas beaux
- Axes faux
- Pas de représentation des points
- 12/20

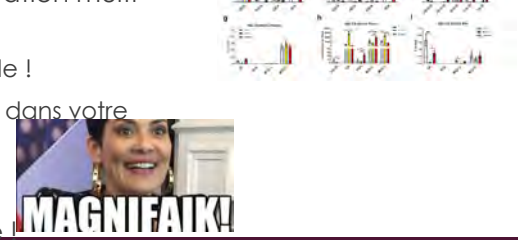


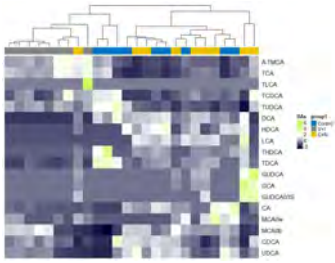
1. Moreno-Sabater, A. *et al.* Systemic anti-commensal response to fungi analyzed by flow cytometry is related to gut mycobiome ecology. *Microbiome* **8**, 159 (2020).

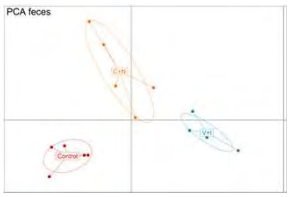
5

QUE FAUT-IL POUR BIEN MODÉLISER SES DONNÉES ?

- Travaillé 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 !





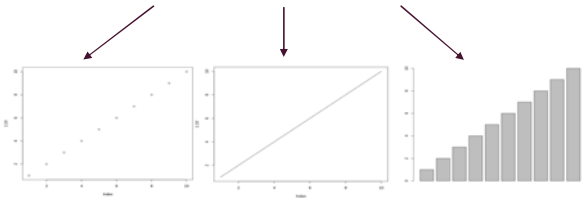


6

LES GRAPHIQUES DANS R, PRINCIPES DE BASE

- 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, …

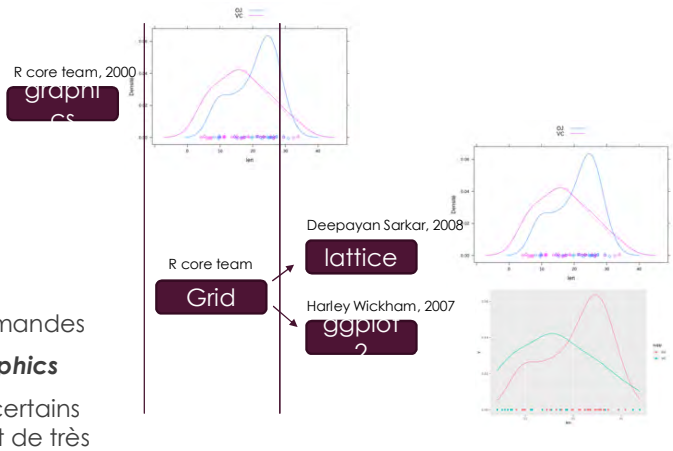
```
1 plot(1:10)
2 plot(1:10, type = "l")
3 barplot(1:10)
```



```
6 boxplot(TUDCA~Treatment, data= csv, col= c("red","black", "orange"))
```

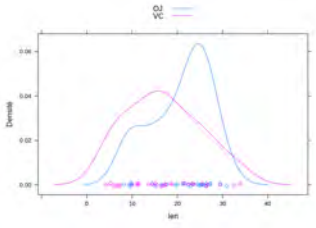
GRAPHIQUES DU PACKAGE GRAPHICS, GRID, LATTICE ET GGPLOT2

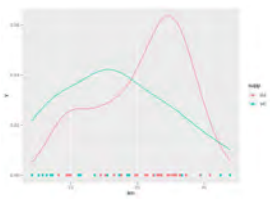
- Chronologiquement :
  - Graphics
    - Grid
      - lattice
      - ggplot2
- ***graphics*** (R Base) et ***lattice*** produisent des graphiques similaires en terme de style
- ***ggplot2*** donne des graphiques différents
- ***lattice*** et ***ggplot2*** demandent moins de commandes
- Simplicité d'utilisation : ***ggplot2*** > ***lattice*** > ***graphics***
- ***ggplot2*** est mieux pour les publications mais certains packages basés sur ***graphics*** ou ***grid*** donnent de très beaux graphiques !



DIFFÉRENCE DANS L'ÉCRITURE

- Pour produire le même graphique
  - ***graphics*** demande beaucoup de code
  - ***lattice*** très peu
  - ***ggplot2*** un peu plus





graphics

```
plot(density(ToothGrowth$len[ToothGrowth$supp == "OJ"]), main = "", xlab = "len", las
lines(density(ToothGrowth$len[ToothGrowth$supp == "VC"]), lwd = 2, col = "cornflowerbl
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, b
```

lattice

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

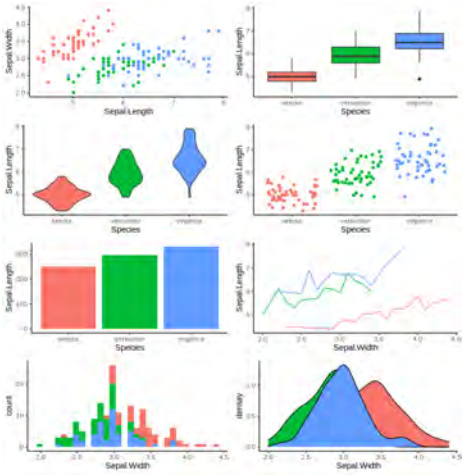
ggplot2

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

Exemple trouvé sur :
lattice : graphiques et formules. http://larmarange.github.io/analyse-R/lattice-graphiques-et-formules.html.

TYPES DE GRAPHIQUES POUR MODÉLISER SES DONNÉES

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()



PRINCIPE GÉNÉRAL DE GGPLOT2

- Toujours commencer par appeler ***ggplot2*** par : ggplot()
  - On va ensuite ajouter les éléments graphiques, c'est-à-dire le type de graphique, les couleurs, les tailles, etc.
- Fournir le dataset :
- En fonction du graphique voulu donner x et y, ou que l'un des deux :
  - Au sein de ***ggplot*** on va utiliser aes() qui signifie aesthetics, c'est dans cette partie que l'on va fournir les éléments esthétiques du graphiques
    - Donc x et y
    - Les couleurs de remplissage (fill=) et les couleurs de contours (color=)
- Donner le type de graphique désiré en utilisant geom_XXXX, ici une boîte à moustache ou boxplot :
- On peut ensuite ajouter d'autres couches graphiques, ici nous allons ajouter des jitters et un violin plot :
  - L'ordre des couches est important, le dernier appelé sera au premier plan
- Des thèmes prédéfinis :
  - theme_classic, theme_bw(), theme_void, ...

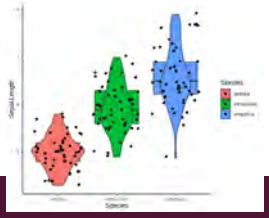
```
ggplot()
```

```
ggplot(data = iris)
```

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

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

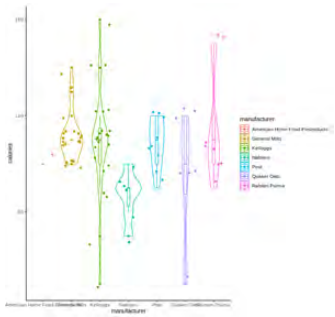
```
ggplot(data = iris, aes(x = Species, y= Sepal.Length, fill=Species))+ geom_boxplot()+ geom_violin()+ geom_jitter()
```



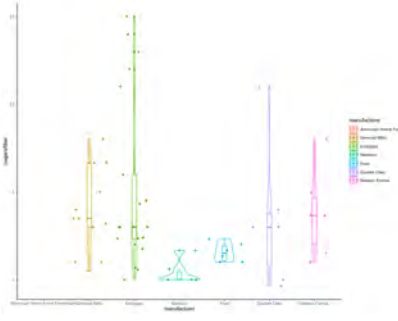
EXERCICE PRATIQUE 1

- 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
- 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)

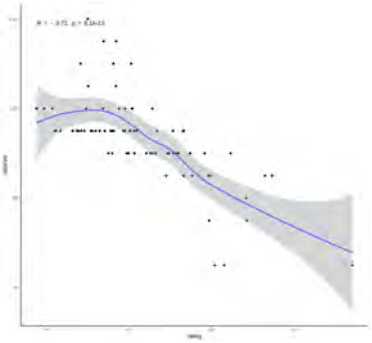
RÉSULTATS



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



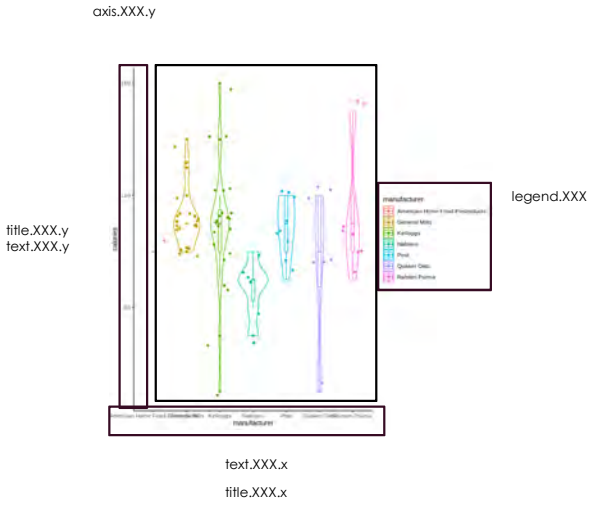
```
ggplot(data=cereal, aes(x= manufacturer, y= sugars/fiber,
color=manufacturer))+ geom_violin()+ geom_boxplot(width=0.1)+geom_jitter()
```



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

AMÉLIORER NOS GRAPHIQUES

- Chaque élément du graphique est personnalisable
- La fonction theme() permet de modifier chaque élément du graphique **mais pas le contenu**
  - Chaque élément graphique appartient aux classes suivantes :
    - Element_text : text des axes, titre, légende, …
    - Element_rect : grille, fond du graphique, encadrements du graphique ou de la légende
    - Element_line : les axes
- Les fonctions scale_XXX_XXX permettent de modifier le **contenu** données, les couleurs, l'ordre, le texte
  - Modifier les couleurs : scale_color_manual(), scale_color_brewer(), scale_fill_manual(), …
  - Modifier le texte : scale_x_discrete(), scale_y_discrete()
  - Modifier les valeurs numériques : scale_x_continuous(), scale_y_continuous, scale_x_log10(), scale_x_datetime(), …

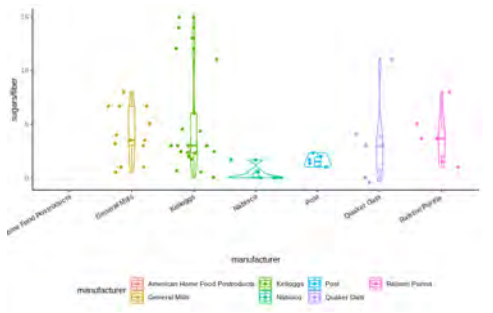


AMÉLIORER NOS GRAPHIQUES

- Ici je change la position de la légende
- L'angle du text en x
- La largeur du violin

- Je veux changer mes titres en « Ratio sucre/fibre et « Fabricants de diabète »
  - On va utiliser la fonction labs()
  - A vous …

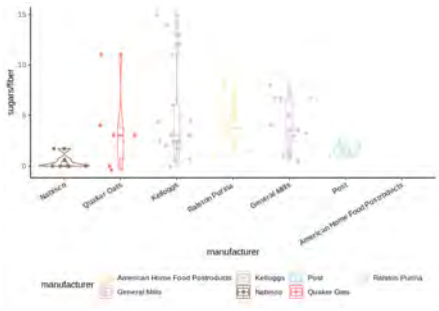
```
ggplot(data=cereal, aes(x= manufacturer, y= sugars/fiber, color=manufacturer))+
geom_violin()+
geom_boxplot(width=0.1)+
geom_jitter()+
theme(legend.position = "bottom", axis.text.x = element_text(angle=30, face="bold", hjust=1))
```



15

AMÉLIORER NOS GRAPHIQUES

- Plus dur …
- On veut changer les couleurs, les labels, l'ordre de nos groupes …
  - "Nabisco", "Quaker Oats","Kelloggs","Ralston Purina ", "General Mills","Post", "American Home Food Postroducts"
  - Couleurs : bisque, thistle, snow3, burlywood4, lightblue2, salmon, palegoldenrod
- Tout va se passer avec les fonctions : scale_XXX_XXX :
  - Scale_x_discrete()
  - Scale_color_manual()



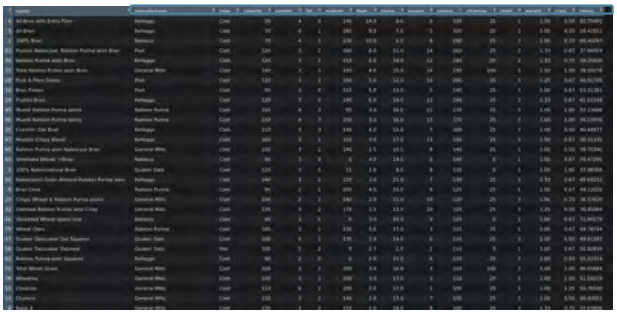

```
p + scale_x_discrete(limits=c("Nabisco", "Quaker Oats","Kelloggs","Ralston Purina ", "General Mills","Post", "American Home Food Postroducts"))+
  scale_color_manual(values= c("bisque", "thistle", "snow3", "burlywood4", "lightblue2", "salmon", "palegoldenrod"
))+
theme(legend.position = "bottom", axis.text.x = element_text(angle=30, face="bold", hjust=1))
```

16

MANIPULATION DES DONNÉES

- Grâce à R on va pouvoir montrer toutes nos données d'un coup
- Il faut d'abord arranger nos données
  - Format long
  - Format large

Une variable par colonne = format large



Toutes les variables dans une colonne = format long

Dimension : 77 lignes, 16 colonnes

MANIPULATION DES DONNÉES

Dimension : 1001 lignes, 5 colonnes

- Nous allons utiliser pivot_longer ou pivot_wider en fonction de ce que l'on veut faire, ici ***pivot_longer()***
- Nous avons maintenant une colonne qui contient tous nos facteurs
- Et une colonne qui contient 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)
```

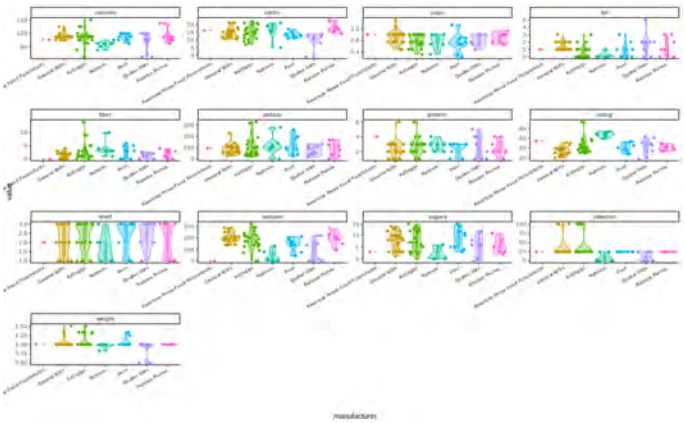

UTILISATION DES NOUVELLES DONNÉES

- Comment on se sert du nouveau jeu de données ?
  - Un graphique avec **manufacturer** en **x**
  - Quelle colonne va me servir pour donner le **y** et quelle colonne va me servir pour les couleurs ?

UTILISATION DES NOUVELLES DONNÉES

- La réponse :

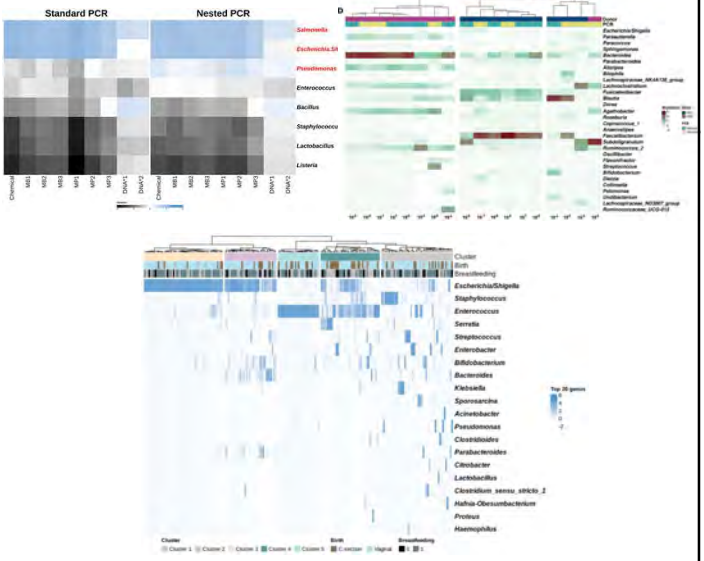
```
ggplot(data=cereal_long, aes(x= manufacturer, y= value,
color=manufacturer, fill=manufacturer))+ geom_violin(alpha=0.3)+
geom_boxplot(width=0.1, alpha=0.3)+
geom_jitter()+
facet_wrap(~factor, scales = "free")+
theme(legend.position = "none", axis.text.x = element_text(angle=30,
face="bold", hjust=1, size=7))
```



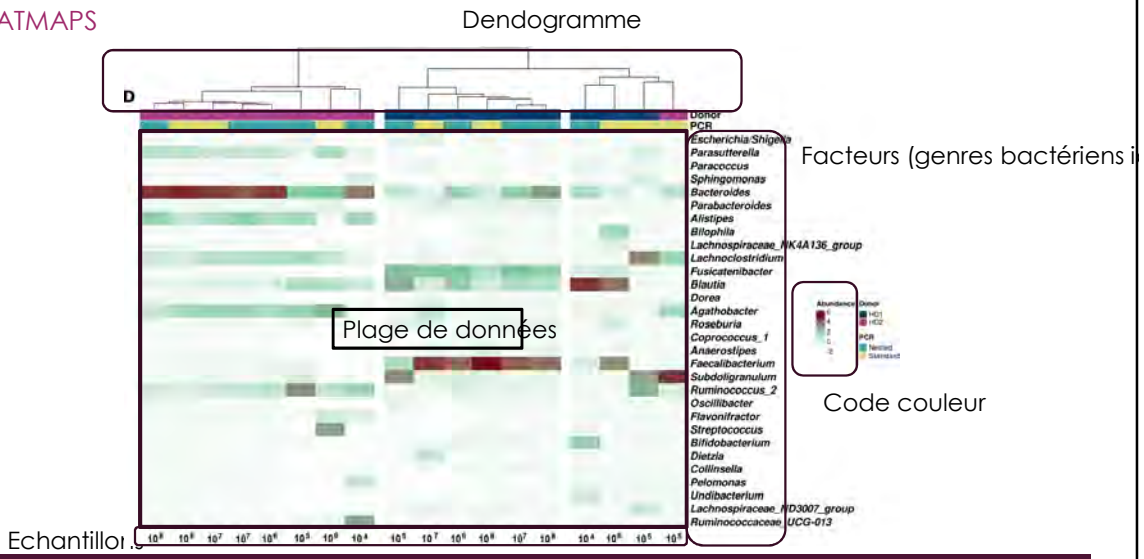
- Et si on mettait **manufacturer** en facet ?

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



LES HEATMAPS



LES HEATMAPS

- 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)

- Pour palier à la différence entres les valeurs :
  - Centrer-réduire les valeurs
    - Par colonnes
    - Par lignes