

Visualisation de données avec R

Bioinformatique I (BIF-7900)

Antoine Bodein

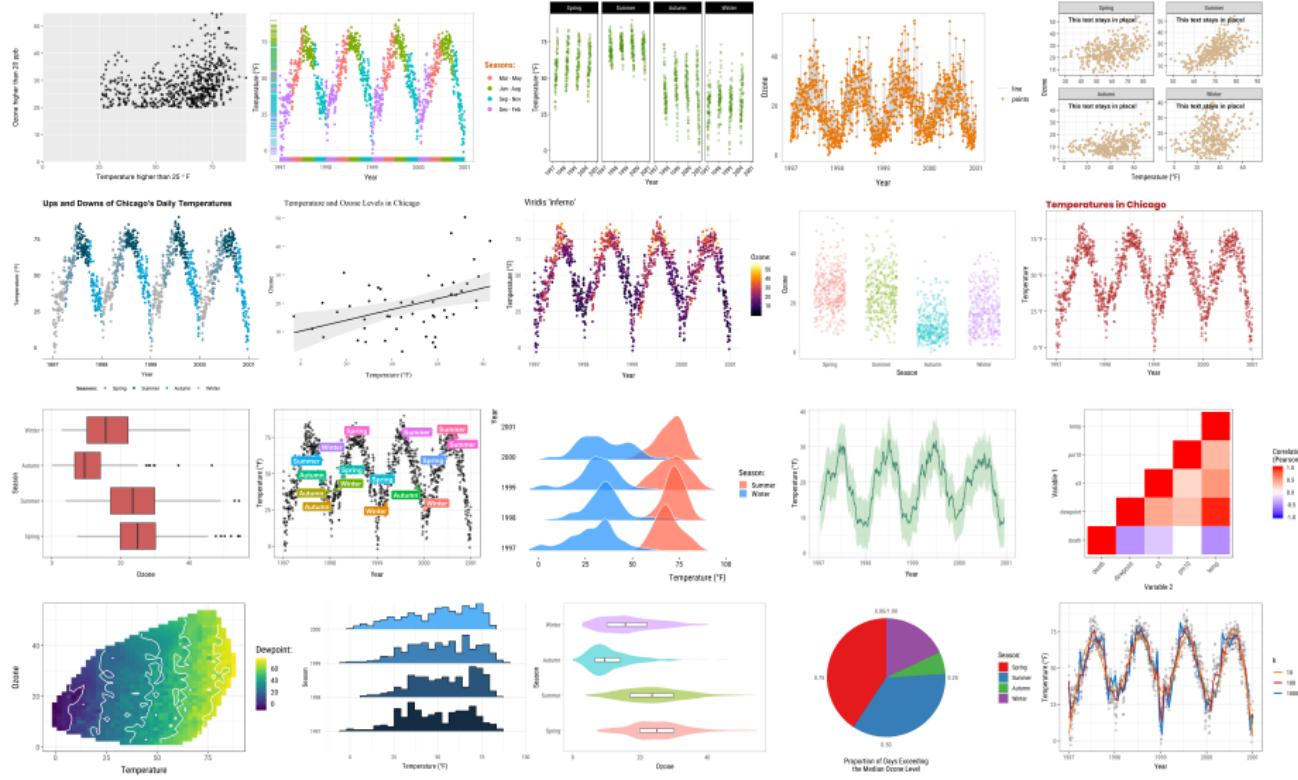
4 novembre 2021



Section 1

Introduction

Introduction



Plan

- 1 Introduction
- 2 Base
- 3 GGplot2
- 4 Quelques Graphiques utiles en Bioinformatique
- 5 Discussion sur les bonnes pratiques de visualisation des données

Objectifs

- graphiques usuels avec R (`base` et `graphics`)
- utilisation du package `ggplot2`
- `tidyverse` : manipulation de `data.frame` pour `ggplot2`

Les graphiques avec R

- Créer des graphiques (*plots*) avec des lignes de codes
- Répliquer et modifier
- Reproductibilité et partage
- R de base (*graphics*) pour des graphiques dans les publications scientifiques
- Des packages spécialisés: `ggplot2` et autres ...

Section 2

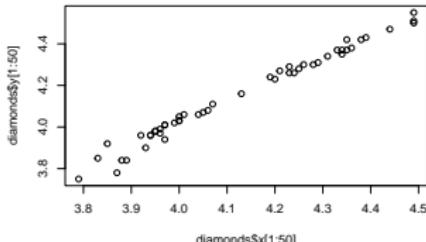
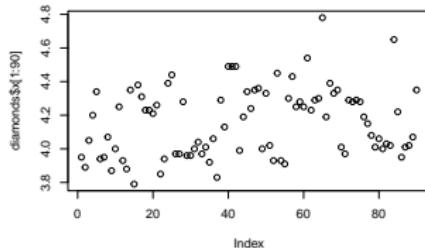
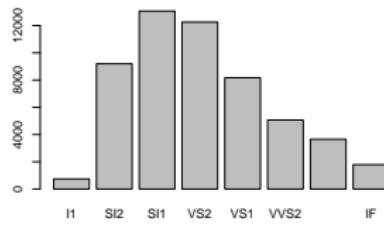
Base

La fonction plot()

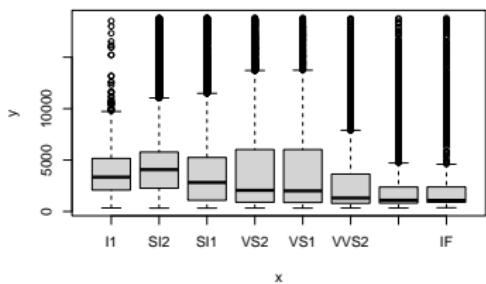
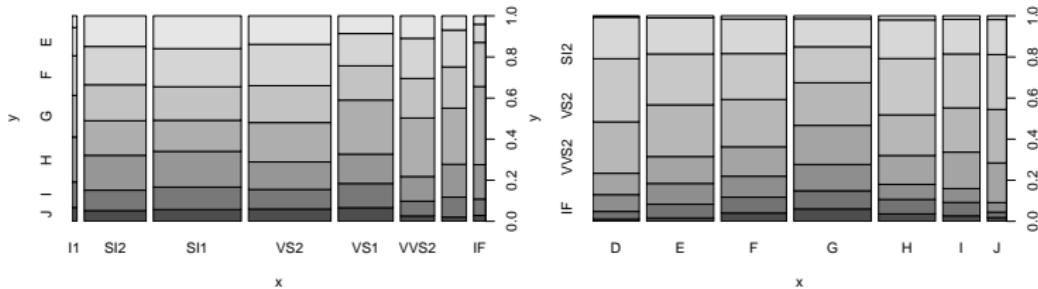
```

data("diamonds")
par(mfrow=c(2,2)) # affichage des graphiques en grille 2x2
plot(diamonds$clarity) # categorical / factor
plot(diamonds$x[1:90]) # continue / numeric
plot(diamonds$x[1:50], diamonds$y[1:50]) # numeric, numeric

```

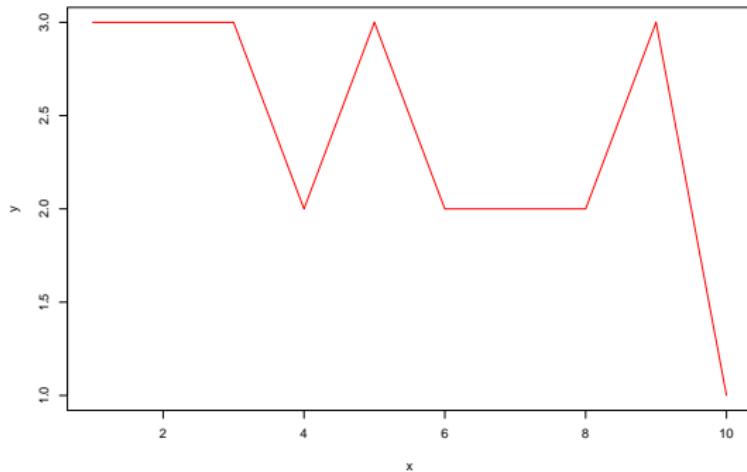


```
par(mfrow=c(2,2)) # affichage des graphiques en grille 2x2
plot(diamonds$clarity, diamonds$color) # factor / factor
plot(diamonds$color, diamonds$clarity) # factor / factor
plot(diamonds$clarity, diamonds$price) # factor / numeric
```



Courbe (line chart)

```
x <- 1:10  
y <- sample(1:4, size = 10, replace = TRUE)  
plot(x = x, y = y, type = "l", col = "red")
```



Graphique en 2 dimensions pour observer la relation entre 2 variables (ou une variable en fonction du temps).

Nécessite au moins 2 vecteurs:

- `x`: Valeurs sur l'axe des abscisses (vecteur)
- `y`: Valeurs sur l'axe des ordonnées (vecteur)
- Peut s'écrire: `plot(y ~ x)`

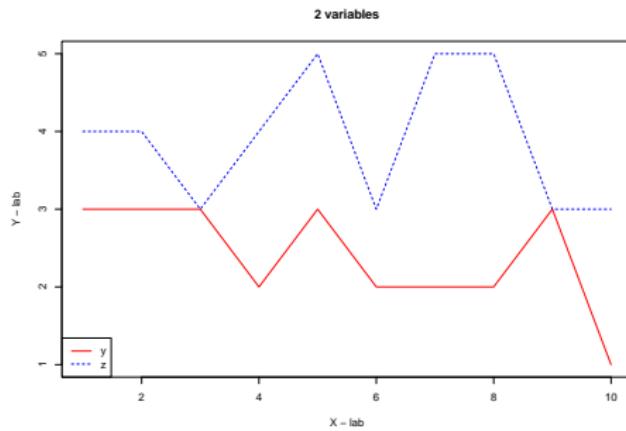
Autres arguments:

- `type`: 'p' pour des points, 'l' pour des lignes, 'b' pour les deux, ...
- `main`: ajouter un titre
- `xlim = c(1,3)`: zoom sur x (aussi `ylim`)
- `col`: couleurs
- `xlab, ylab`: légende en x, y
- `pch`: forme des points
- `lty`: type des lignes

Autres options: `help(plot)`

Ajouter des lignes avec line()

```
z <- sample(3:5, size = 10, replace = TRUE)
plot(x = c(), xlim = c(1,10), ylim = c(1,5), main = "2 variables",
      xlab = "X - lab", ylab = "Y - lab") # empty
lines(x, y, type = "l", col = "red")
lines(x, z, type = "l", col = "blue", lty = 2)
legend("bottomleft",
       legend = c("y", "z"),
       col = c("red", "blue"),
       lty = c(1,2))
```

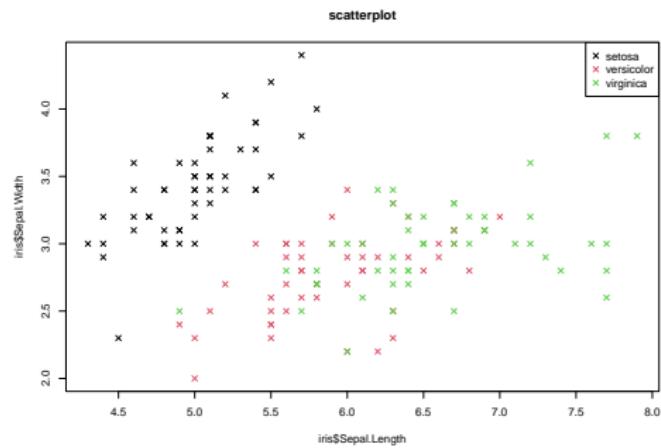


2 - Le nuage de points (scatterplot)

- Représente la relation entre 2 variables.
- Nécessite 2 vecteurs

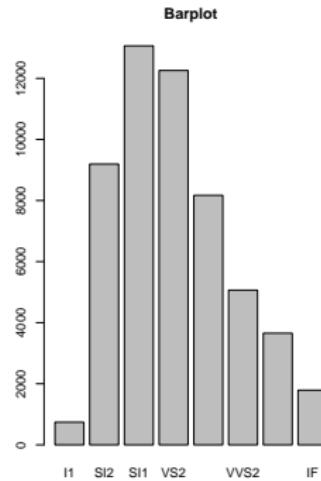
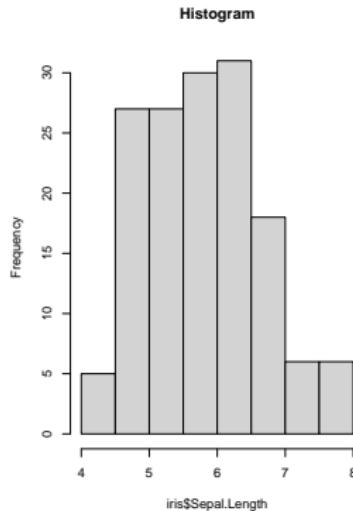
```
data(iris)
```

```
plot(x = iris$Sepal.Length, y = iris$Sepal.Width, col = as.factor(iris$Species), main = "scatterplot",  
      legend = c("setosa", "versicolor", "virginica"),  
      col = c(1, 2, 3),  
      pch = c(4))
```



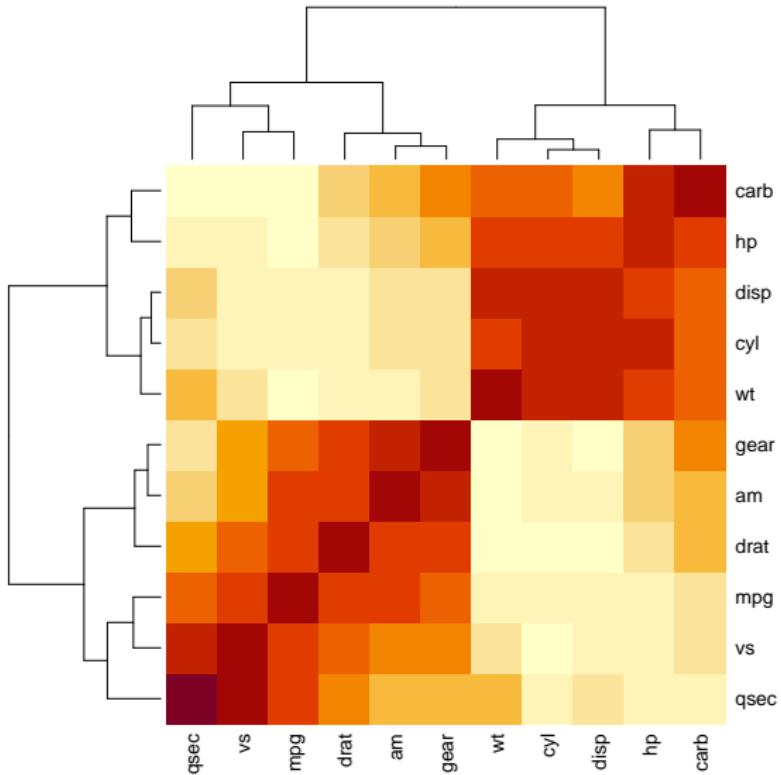
Autres plots

```
par(mfrow = c(1,2))
hist(iris$Sepal.Length, main = "Histogram")
barplot(table(diamonds$clarity), main = "Barplot")
```



```
boxplot(diamonds$clarity)
```

```
data <- as.matrix(cor(mtcars))
heatmap(data)
```



Sauvegarder un graphique

```
# 1- ouvrir une sortie graphique (png, jpeg, tiff, ...)  
png("~/Documents/my_plot.png")  
  
# 2- faire le graphique  
heatmap(data)  
  
# 3- fermer la sortie graphique  
dev.off()
```

Remarque: avec `pdf()` il est possible de sauvegarder plusieurs graphiques dans le même fichier.

Section 3

GGplot2

Introduction à ggplot2

Avantages:

- Grammaire graphique cohérente (Wilkinson, 2005)
- Permet une construction rapide de graphiques simples
- Très flexible
- Système de thème pour polir l'apparence des graphiques
- Système graphique mature et complet
- Beaucoup d'utilisateurs, liste de diffusion active

Limites:

- Graphiques en 3 dimensions,
- théorie des graphes (voir le paquet igraph)
- Graphiques interactifs

*"In brief, the grammar tells us that a graphic maps the **data** to the **aesthetic attributes*** (colour, shape, size) of **geometric objects** (points, lines, bars). The plot may also include statistical transformations of the data and information about the plot's **coordinate system**. **Facetting** can be used to plot for different subsets of the data. The combination of these independent components are what make up a graphic." H. Wickham*

appartée sur le tidyverse



Tidy data:

- ① Each variable forms a column.
- ② Each observation forms a row.
- ③ Each type of observational unit forms a table.

Syntaxe de ggplot2

```
library(ggplot2)
myplot <- ggplot(data = iris,
                  mapping = aes(x = iris$Sepal.Length,
                                 y = iris$Sepal.Width,
                                 color = iris$Species)) +
  geom_point()
```

- appel initial: `ggplot(...)`
- plus des *layers*

Un *layer* est défini par:

- data
- aesthetic mappings
- geometric objects (geom)
- statistical transformation (stats)
- scale
- facetting

1 - Data

Les données doivent être en format `data.frame` et contenir idéalement toutes les informations (colonnes) pour le mapping des aesthetics.

Les données peuvent être fourni lors de l'appel initial: `ggplot(data = ...)`

Ou à chaque layer (`geoms`): `ggplot() + geom_point(data = ...)`

Remarque: Le `data.frame` doit être souvent en format *long*.

```
tmp <- breast.TCGA$data.train$mrna[1:5,1:5]
rownames(tmp) <- paste0("Sample_", 1:5)

tmp2 <- tmp %>% as.data.frame() %>% rownames_to_column("SampleID") %>%
  pivot_longer(names_to = "Gene", values_to = "Value", -c("SampleID"))
```

X1	X2	X3	X4
A	1	0.1	10
B	2	0.2	20

→

X1	V1	V2
A	X2	1
B	X2	2
A	X3	0.1
B	X3	0.2
A	X4	10
B	X4	20

2 - Aesthetics

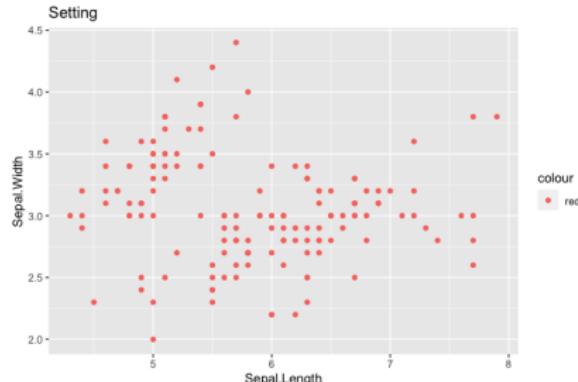
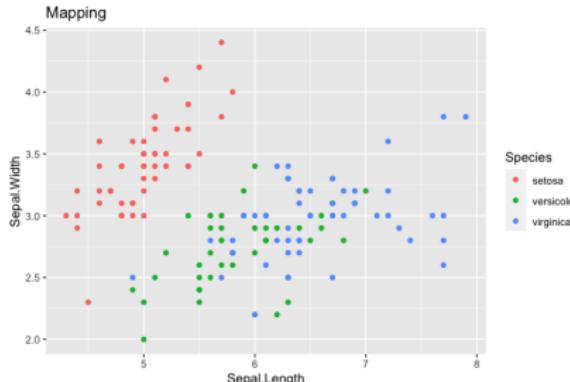
On utilise la fonction `aes()` pour mapper les valeurs du `data.frame` aux paramètres de `ggplot`.

Une colonne (variable) pour les valeurs en X, une colonne pour les valeurs en Y, une autre pour les couleurs, formes, etc ...

Remarque: Certains paramètres peuvent être défini globalement par une constante. On fait donc une différence entre le *setting* et le *mapping* (qui utilise `aes()`).

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point(aes(color = Species)) + ggtitle("Mapping")
```

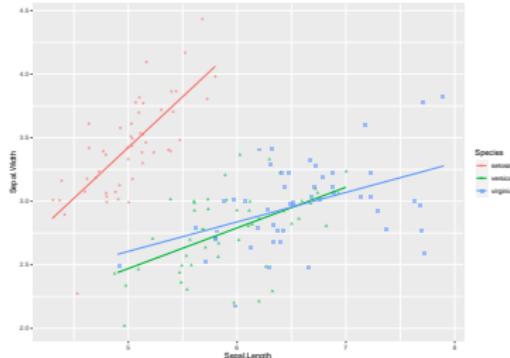
```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point(aes(color = "red")) + ggtitle("Setting")
```



3 - Les objets géométriques (*geoms*)

- Les *geoms* sont les objets géométriques pour représenter les données.
- Si les asthetics répondent à la question: *quelles sont les variables à représenter? (Quoi?)*; les *geoms* répondent à la question: *Comment?*
- En fonction du nombre de variable à représenter (1,2,3), nous avons accès à différentes formes géométriques (nuage de points, lignes, barres, surfaces, ...).
- Chaque geom appelle un *layer*. Les données, aes, et autres paramètres peuvent être défini globalement ou dans chaque geom.
- Par couches successive, on peut superposer plusieurs geom.

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
  geom_jitter(aes(shape = Species), alpha = 0.6) +
  geom_smooth(se = F, method = "lm")
```



GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank() and a + expand_limits()
Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1,
xend = long + 1), curvature = 1) -> x, yend, y, yend,
alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt",
linejoin = "round", linemitre = 1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)) -> x, y, alpha,
color, fill, group, polygon, linetype, size

b + geom_rect(aes(xmin = long - ymin = lat,
xmax = long + 1, ymax = lat + 1)) -> xmax, xmin,
ymin, ymax, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900,
ymax = unemploy + 900)) -> x, ymax, ymin,
alpha, color, fill, group, linetype, size
```

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

```
b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(intercept = lat))
b + geom_vline(aes(intercept = long))

b + geom_segment(aes(yend = lat + 1, xend = long + 1))
b + geom_spoke(aes(angle = 1:115, radius = 1))
```

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly()
x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy))
x, y, alpha, color, fill, linetype, size, weight
```

discrete

```
d <- ggplot(mpg, aes(f1))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight
```

TWO VARIABLES

both continuous
e <- ggplot(mpg, aes(cty, hwy))

```
e + geom_label(aes(label = cty), nudge_x = 1,
nudge_y = 1) -> x, y, label, alpha, angle, color,
family, fontface, hjust, lineheight, size, vjust

e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl")
x, y, alpha, color, linetype, size

e + geom_smooth(method = lm)
x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1,
nudge_y = 1) -> x, y, label, alpha, angle, color,
family, fontface, hjust, lineheight, size, vjust
```

one discrete, one continuous

```
f <- ggplot(mpg, aes(class, hwy))

f + geom_col()
x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot()
x, y, lower, middle, upper, ymax, ymin, alpha,
color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill, group

f + geom_violin(scale = "area")
x, y, alpha, color, fill, group, linetype, size, weight
```

both discrete

```
g <- ggplot(diamonds, aes(cut, color))

g + geom_count()
x, y, alpha, color, fill, shape, size, stroke

g + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size
```

THREE VARIABLES

sealsSz <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))

```
l + geom_contour(aes(z = z))
x, y, z, alpha, color, group, linetype, size, weight

l + geom_contour_filled(aes(fill = z))
x, y, alpha, color, fill, group, linetype, size, subgroup
```

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

```
h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d()
x, y, alpha, color, group, linetype, size

h + geom_hex()
x, y, alpha, color, fill, size
```

continuous function

i <- ggplot(economics, aes(date, unemploy))

```
i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size
```

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

```
j + geom_crossbar(fatten = 2) -> x, y, ymax,
ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() -> x, ymax, ymin,
alpha, color, group, linetype, size, width
Also geom_errorbarh().

j + geom_linerange()
x, y, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange() -> x, y, ymin, ymax,
alpha, color, fill, group, linetype, shape, size
```

maps

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map)
+ expand_limits(x = map$long, y = map$lat)
map_id, alpha, color, fill, linetype, size
```

```
l + geom_raster(aes(fill = z), hjust = 0.5,
vjust = 0.5, interpolate = FALSE)
x, y, alpha, fill

l + geom_tile(aes(fill = z))
x, y, alpha, color, fill, linetype, size, width
```

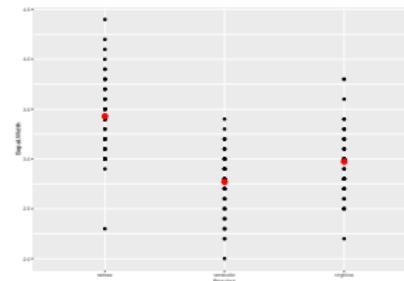
4 - stats

Les transformations statistiques (stats) transforment les données, typiquement en les résumant d'une certaine manière (ex: moyenne).

Ils s'utilisent soit :

- dans les geoms `geom_point()`

```
ggplot(iris, aes(x = Species, y = Sepal.Width)) +
  geom_point() + # toutes les données
  geom_point(stat = "summary", fun.y = "mean", col = "red", size = 4)
```



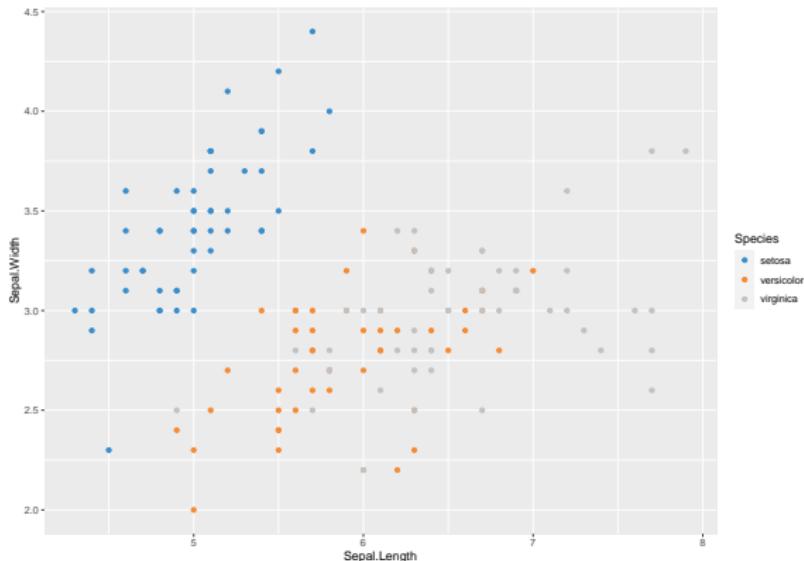
- Ou dans son propre layer (`stat_...()`)

```
ggplot(iris, aes(x = Species, y = Sepal.Width)) +
  geom_point() + # toutes les données
  stat_summary(fun = "mean", col = "red", size = 2)
```

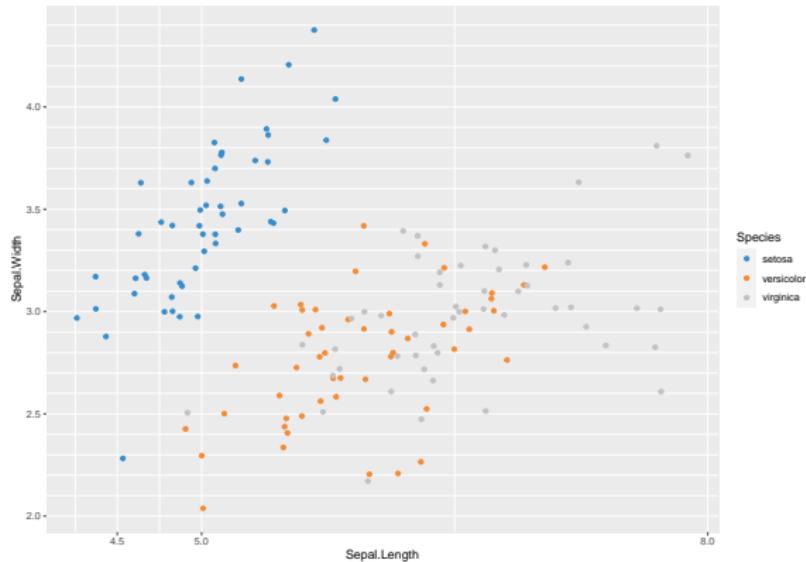
5 - scale

Les *scales* contrôlent le mapping des datas aux aesthetics. Ils contrôlent également les éléments graphiques comme les axes, légende, ...

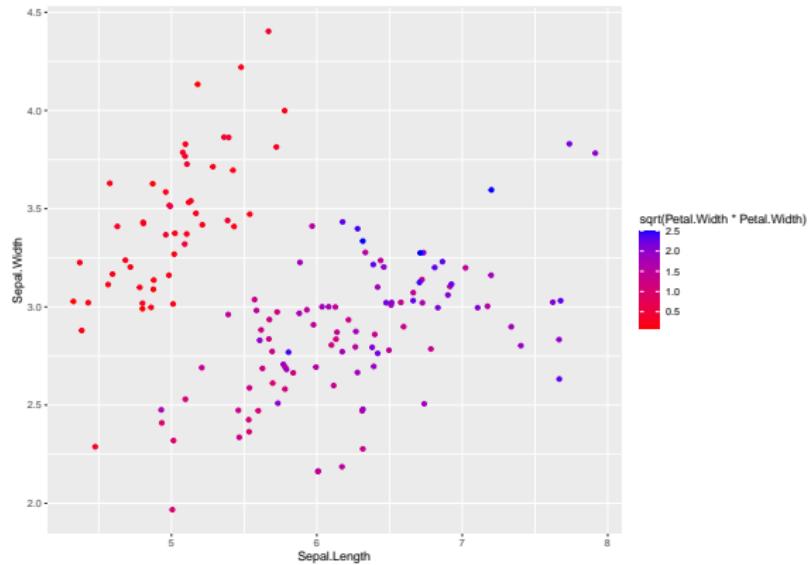
```
my_color <- c("#388ECC", "#F68B33", "#C2C2C2")
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(aes(col = Species)) +
  scale_color_manual(values = (my_color))
```



```
my_color <- c("#388ECC", "#F68B33", "#C2C2C2")
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_jitter(aes(col = Species)) +
  scale_color_manual(values = (my_color)) +
  scale_x_continuous(breaks = c(4.5,5,8)) +
  scale_y_continuous(minor_breaks = seq(2,5, by = 0.1))
```



```
gg <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_jitter(aes(col = sqrt(Petal.Width*Petal.Width))) +  
  scale_color_gradient(low = "red", high = "blue")  
gg
```



6 - Système de coordonnées

Pour:

- Définir les limites du graphe (et zoomer):

```
gg + coord_cartesian(xlim = c(6,7), ylim = c(2.5, 3))
```

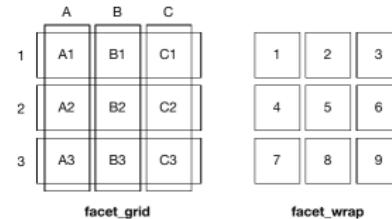
- Inverser X et Y

```
gg + coord_flip()
```

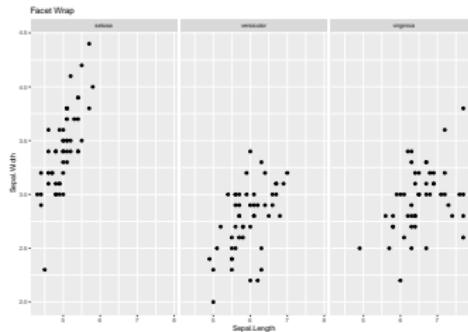
- ...

7 - Facetting

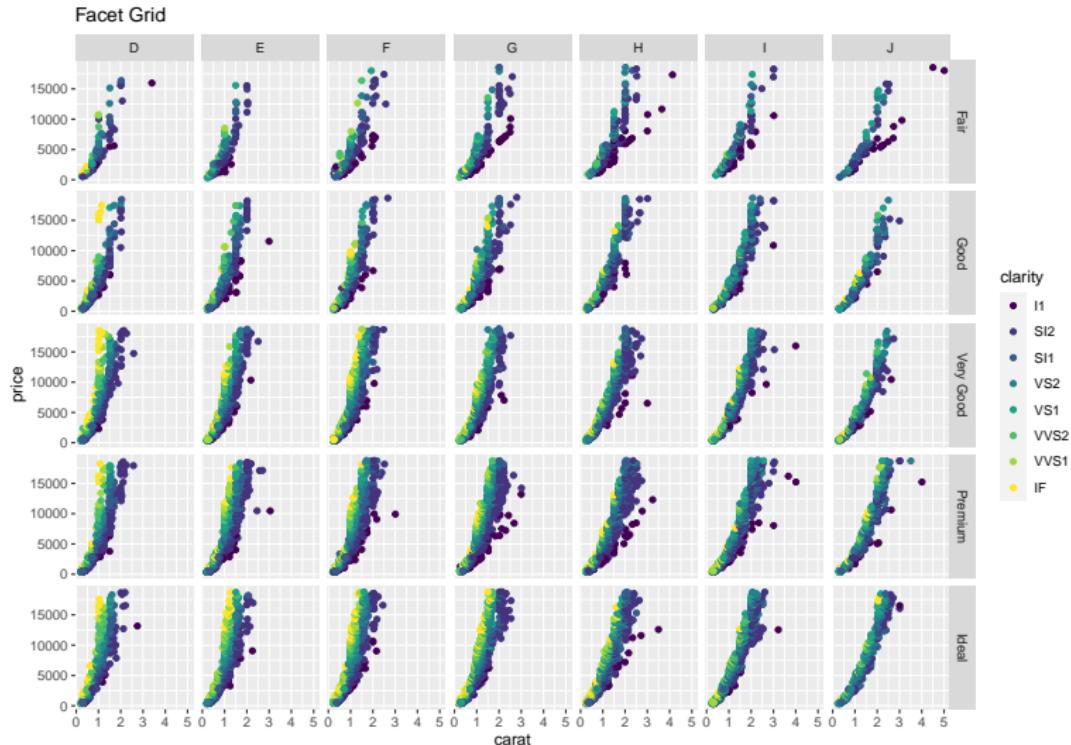
Pour générer plusieurs graphes à partir de sous ensembles de données.



```
ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point() +
  facet_wrap(~Species) +
  ggtitle("Facet Wrap")
```

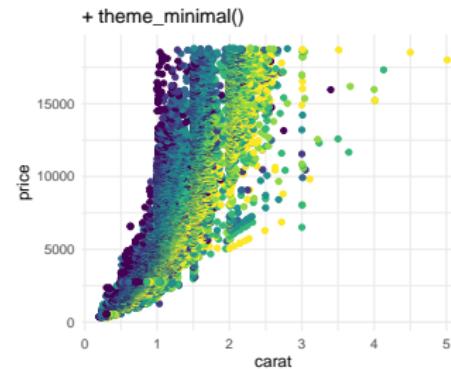
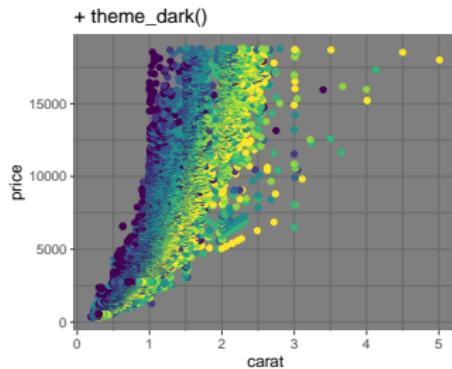
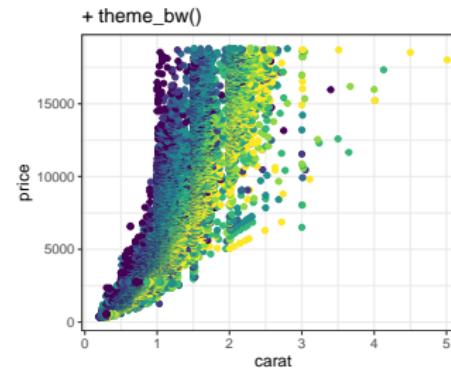
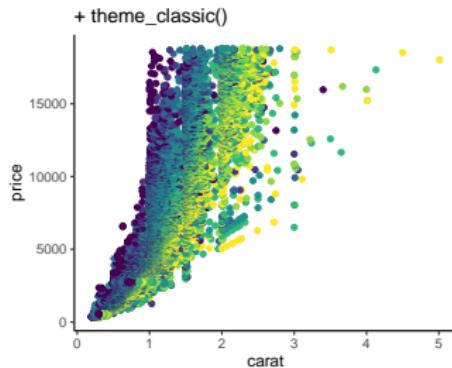


```
ggplot(diamonds, aes(carat, price, col = clarity)) +  
  geom_point() +  
  facet_grid(cut~color) +  
  ggtitle("Facet Grid")
```



8 - Thème

Pour customiser l'affichage. Quelques thèmes pré-défini:



Autres astuces

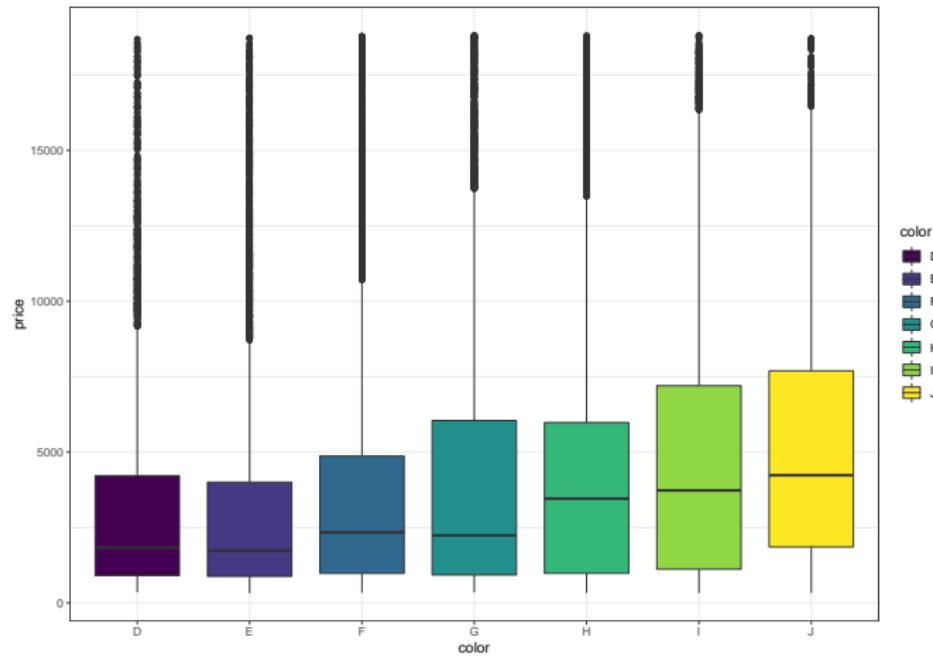
- sauvegarder: `ggsave()`
- position de la légende + `theme(legend.position = "bottom")`
- titre: + `ggtitle("Title")`
- titre des axes: + `labs(x = "Axe X")`
- titre légende: + `labs(color = "Legend title") ou fill, ... dépend du type d'aes`
- rotation et position des labels d'axe:
 - + `theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))`

Section 4

Quelques Graphiques utiles en Bioinformatique

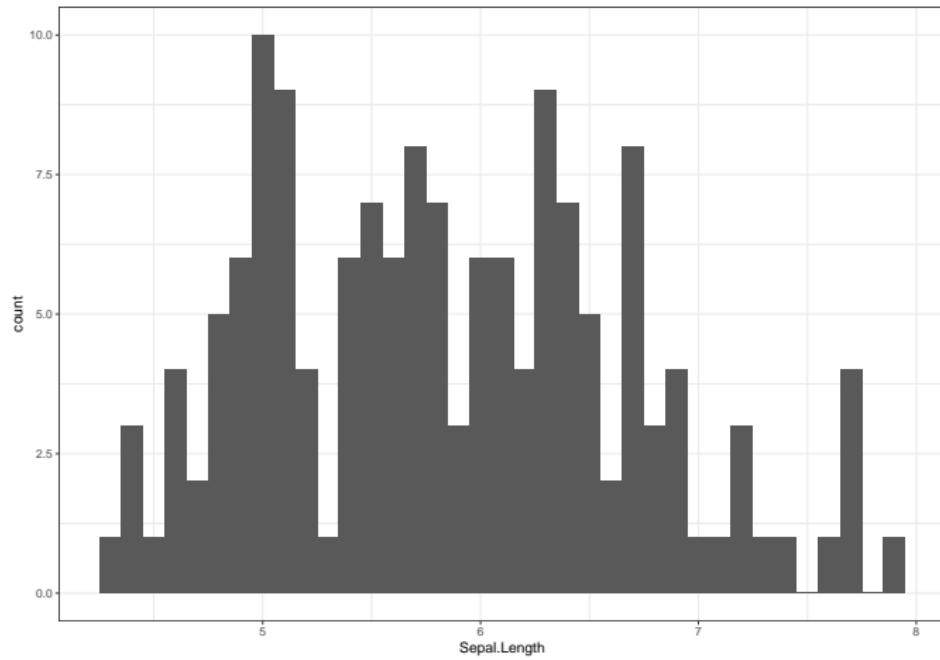
1- Boxplot

```
ggplot(diamonds, aes(x = color, y = price, fill = color)) +  
  geom_boxplot() + theme_bw()
```



2- Histogramme

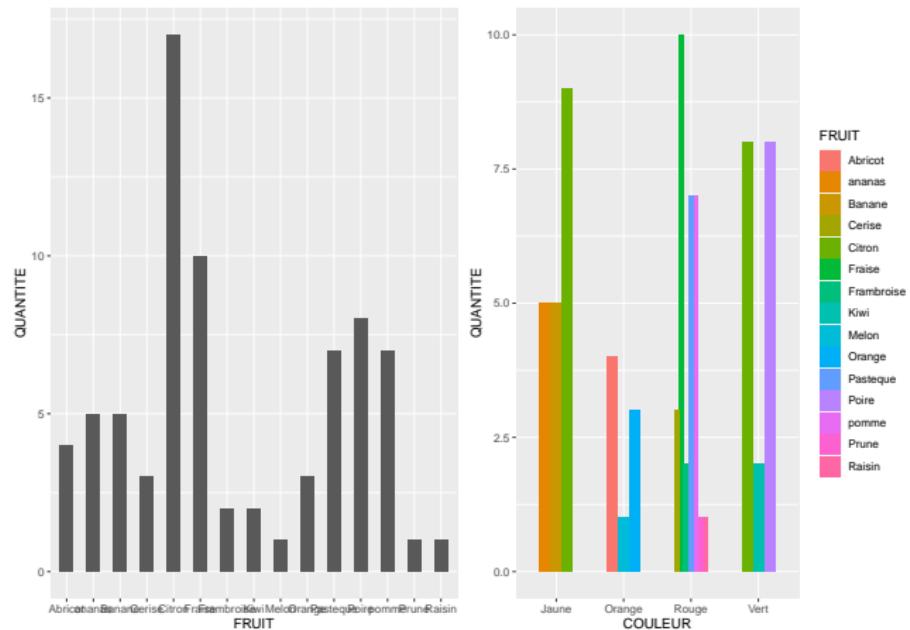
```
ggplot(iris, aes(Sepal.Length)) +  
  geom_histogram(binwidth = 0.1) + theme_bw()
```



3- Barplot

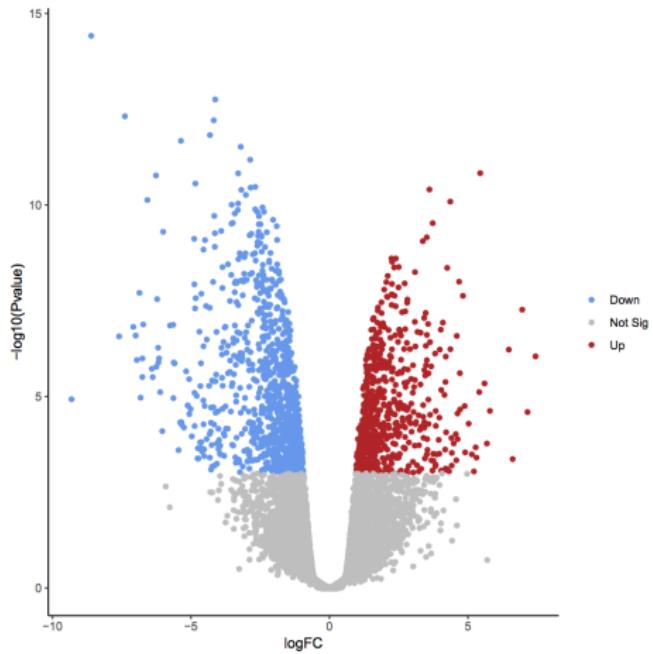
```
ggplot(fruit, aes(FRUIT, QUANTITE)) +
  geom_bar(stat = "identity", width = 0.5)

ggplot(fruit, aes(COULEUR, QUANTITE, fill=FRUIT)) +
  geom_bar(stat = "identity", position="dodge", width = 0.5)
```



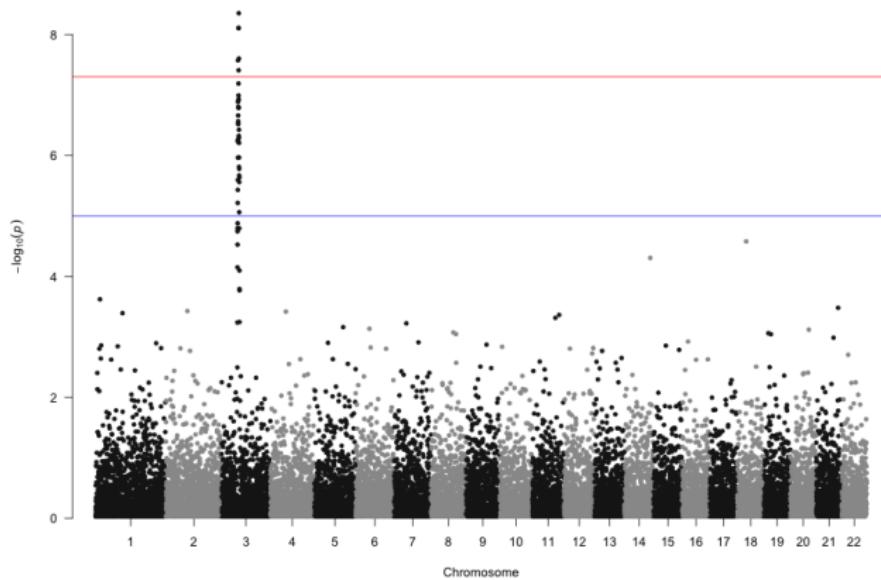
4- Volcano Plot

Un *Volcano plot* est un type de diagramme de nuage de points qui montre une signification statistique (valeur de p-value) par rapport à l'ampleur du changement ($\log\text{Foldchange}$)



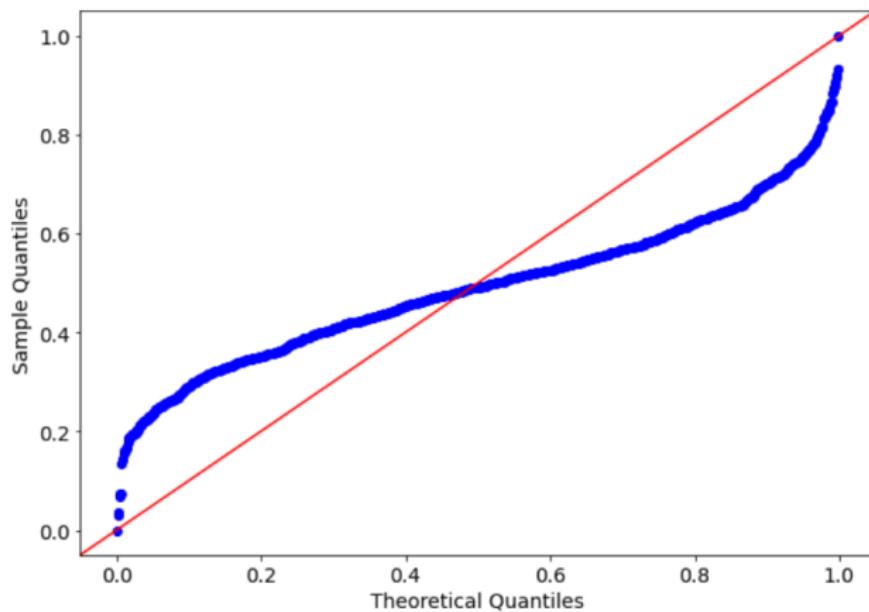
5- Manhantan plot

Le *manhantan plot* est un type de diagramme de nuage de points Le tracé est couramment utilisé dans les études d'association génomique (GWAS) pour afficher des SNP significatifs.



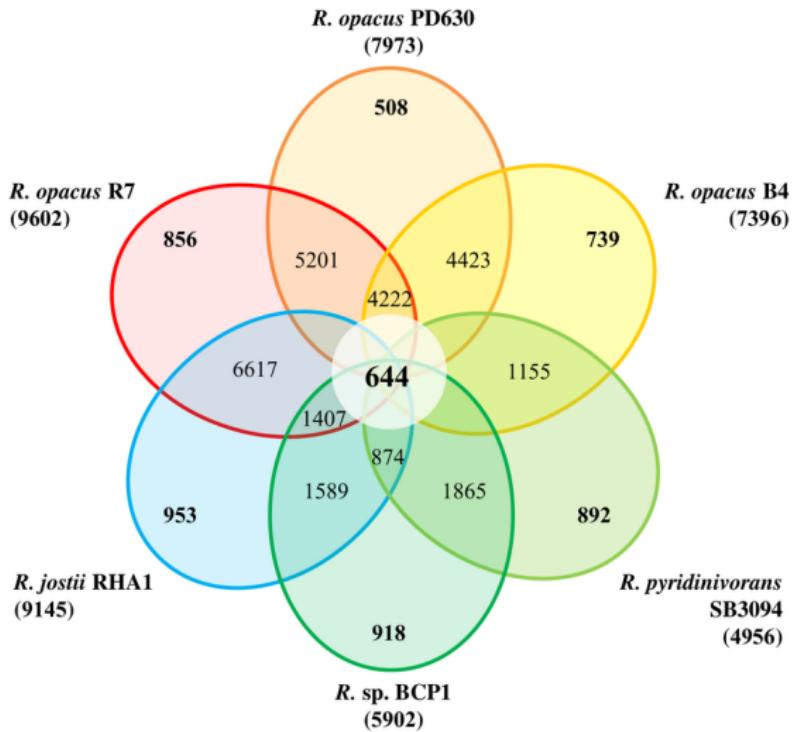
6- QQ plot

Un graphique quantile-quantile (QQ) peut être utilisé pour caractériser dans quelle mesure la distribution observée de la statistique suit les prévisions (distribution nulle).



7- Venn diagram

Graphique utilisé pour représenter des relations logiques entre ensembles



Section 5

Discussion sur les bonnes pratiques de visualisation des données

Bonne pratiques de visualisation des données

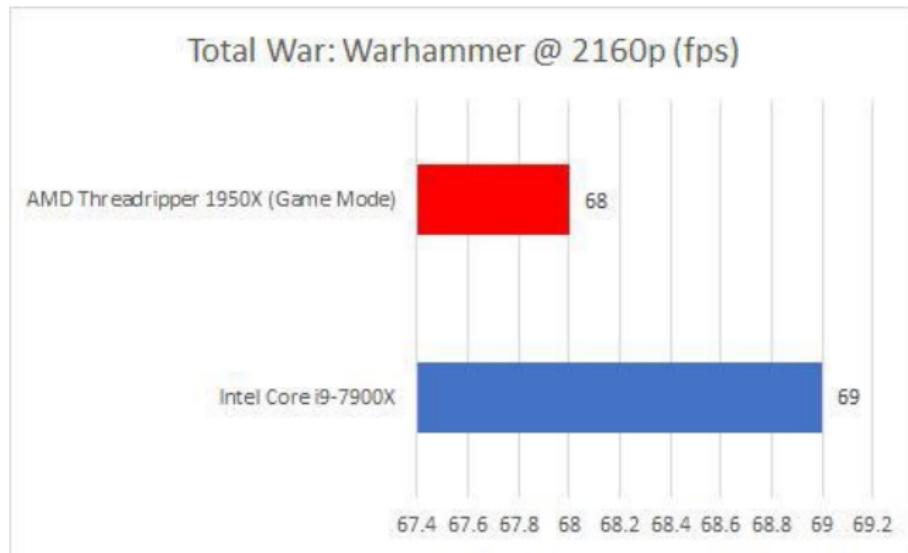
Quelques bonnes pratiques à garder à l'esprit:

- Choisissez le meilleur visuel pour vos données et son objectif.
- Assurez-vous que vos données sont facilement compréhensibles et visibles.
- Offrez le contexte nécessaire à votre public dans et autour de votre visuel.
- Gardez votre visuel aussi simple et direct que possible.

Mauvaises habitudes

- Manque d'étiquetage clair des axes
- Tronquer un axe (termine ou commence à un endroit trompeur)
- Les nombres ne s'additionnent pas correctement (Pie chart)
- Mauvaise utilisation d'un type de graphique
- *Cherrypicking* des données
- Mauvais choix graphiques (comme des choix de couleurs à faible contraste ou une perspective déformante)
- Pas de titre
- Pas de légende

Exemple



- L'erreur ici est simple: l'axe des x doit commencer à zéro.

https://getdolphins.com/wp-content/uploads/2018/02/DG5UK_WUAAAVQe5.jpg

Exemple

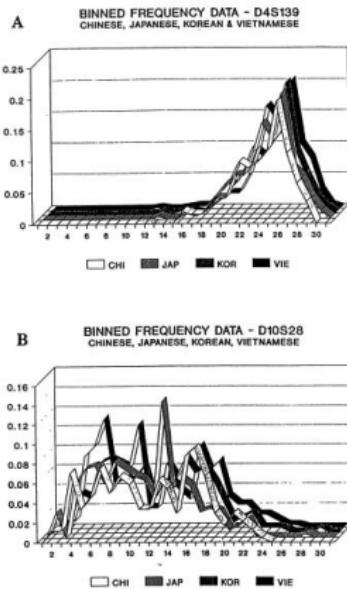


FIG. 4. Fixed bin distribution (histogram) for two loci and four Asian subpopulations (used with permission from John Hartmann): the boundaries of the 30 bins (vertical axis) are determined by the FBI; these bins are not of equal length. Sample sizes (numbers of individuals) for Chinese, Japanese, Korean and Vietnamese are 103, 125, 93 and 215 for D4S139 and 120, 137, 100 and 193 for D10S28. The horizontal axis is the bin number; bins are not of equal length.

- Le rendu tridimensionnel des courbes est inutile.

https://www.biostat.wisc.edu/%7Ekbroman/topten_worstgraphs/roeder_fig4.jpg

Exemple

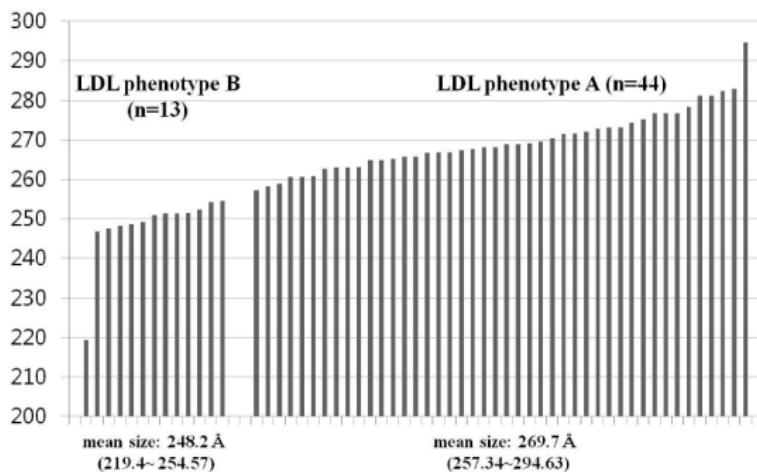
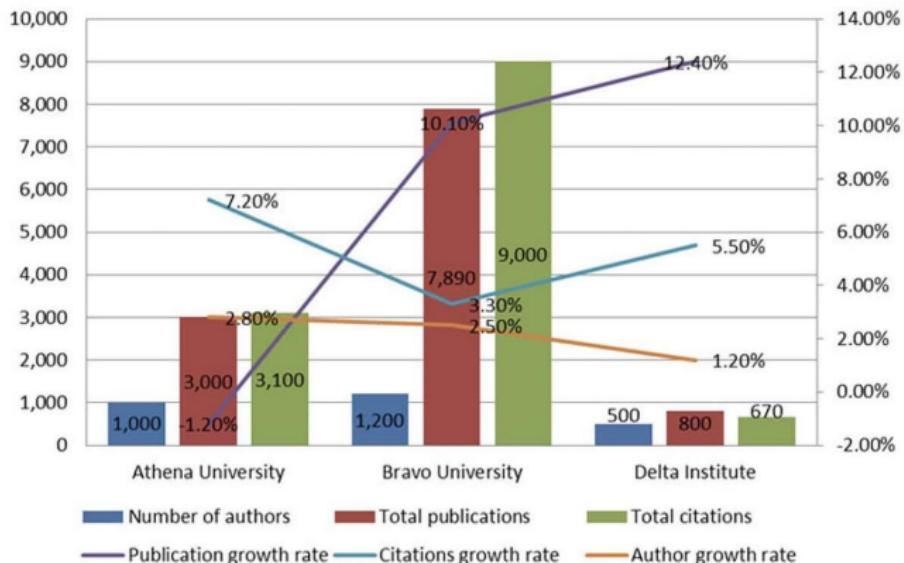


Fig. 1. Distribution of low-density lipoprotein (LDL) particle size in all study subjects (LDL phenotypes A and B). *LDL phenotype A group* (mean size: 269.7 Å, n = 44), subjects with buoyant-mode profiles [peak LDL particle diameter \geq 264 Å] including intermediate LDL subclass pattern [256 Å \leq peak LDL particle diameter \leq 263 Å]; *LDL phenotype B group* (mean size: 248.2 Å, n = 13), subjects with dense-mode profiles [peak LDL particle diameter \leq 255 Å]

Mauvais choix de graphe: l'utilisation d'une barre distincte pour chaque individu est inutile et difficile à lire.

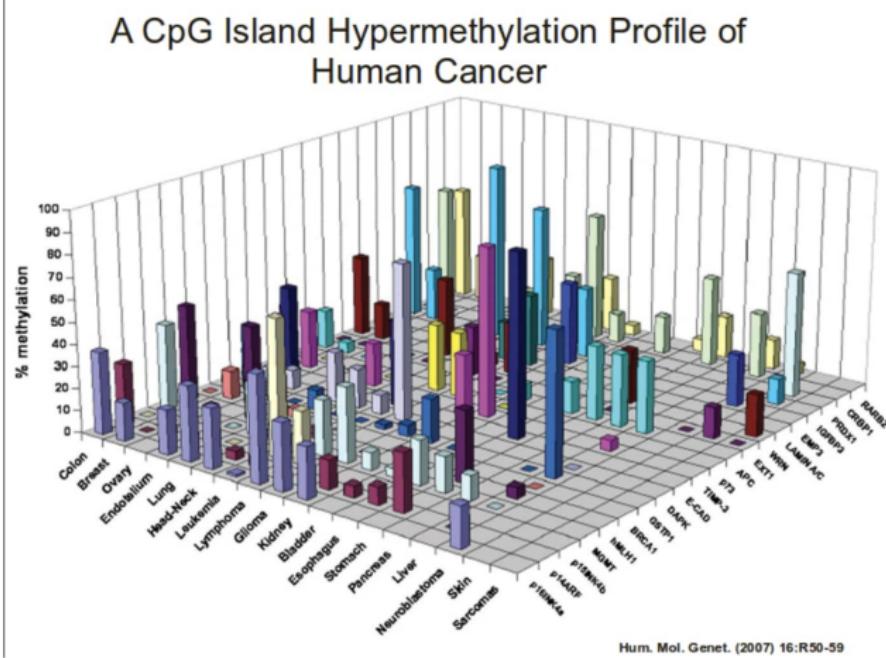
Exemple



- Plusieurs variables complètement différentes dans un même visuel.

<https://blog.hubspot.com/marketing/great-data-visualization-examples>

Exemple



- 3D inutile et rend le graphique plus compliqué à lire (heatmap).

<https://blog.hubspot.com/marketing/great-data-visualization-examples>