Graphs with ggplot2

Vincent Guillemot

Nov. 25

First things first

We will need the package `ggplot2` :

- Check that `ggplot2` is installed
- If not, install it, then load it

```
library(ggplot2)
```

We also need the “fruits” data:

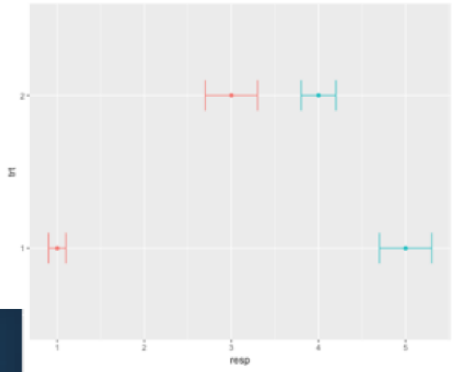
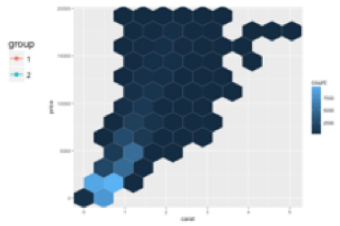
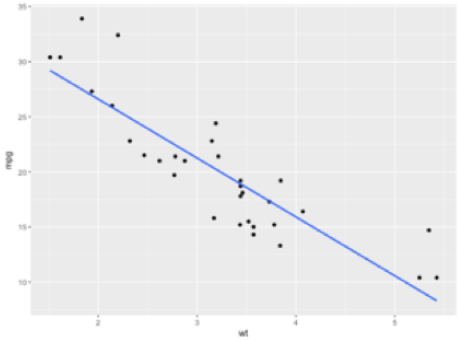
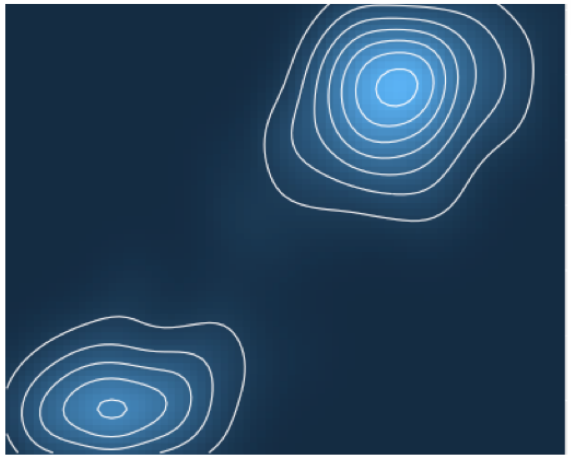
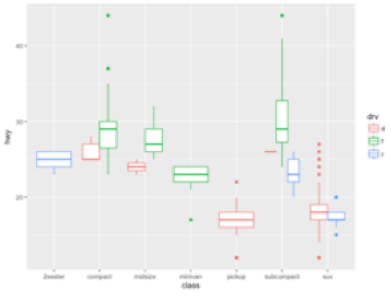
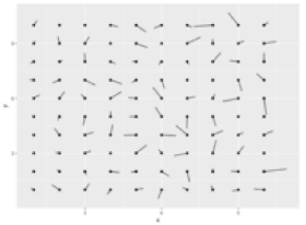
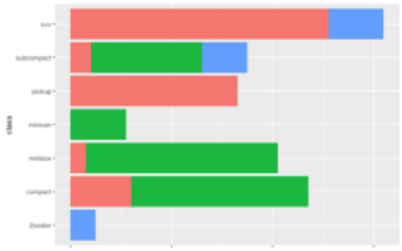
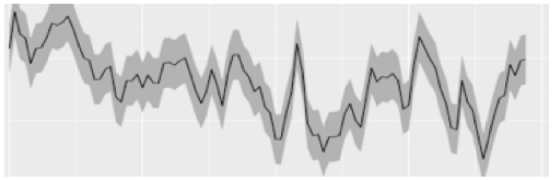
```
data("fruits", package = "ReMUSE")
```

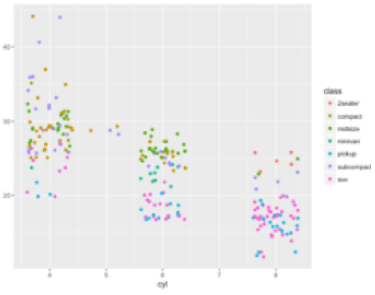
Choose your graph!

From Data to Viz : https://www.data-to-viz.com/

A sample of ggplots



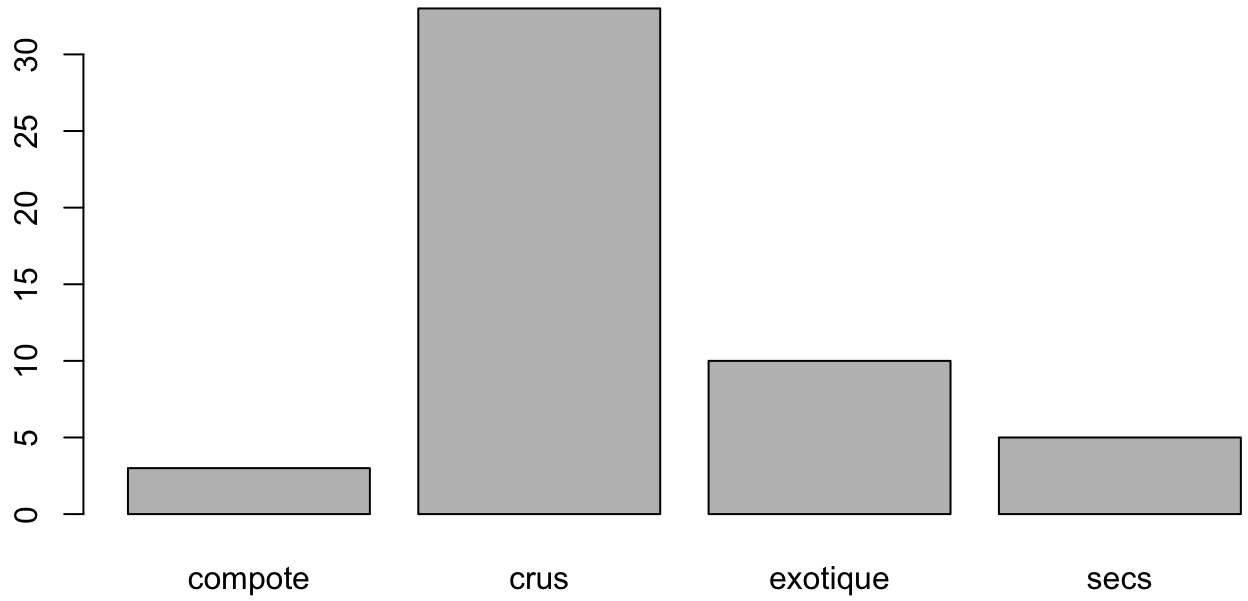


Bar Plot

Reminder : the barplot function

LThe base function for bar plots is `barplot` :

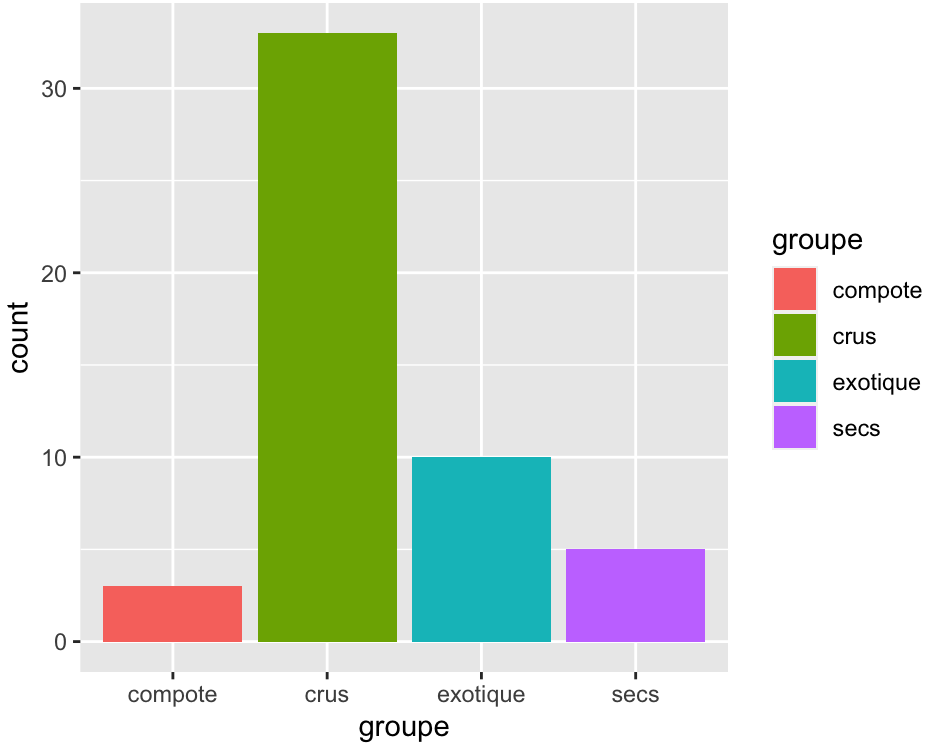
```
barplot(table(fruits$groupe))
```



With colors:

The geom_bar "function"

```
ggplot(data = fruits, aes(x = groupe, fill = groupe)) +
  geom_bar()
```



STOP !

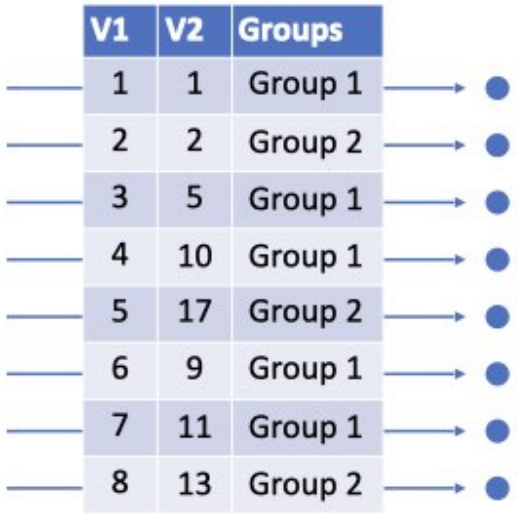
Décomposition de la commande

- `ggplot` : create an exmpry canvas
- `aes` : declare aesthetic parameter (position, color, width, shape, opacity, etc…)
- `geom_bar` : use a *geometry*

Data

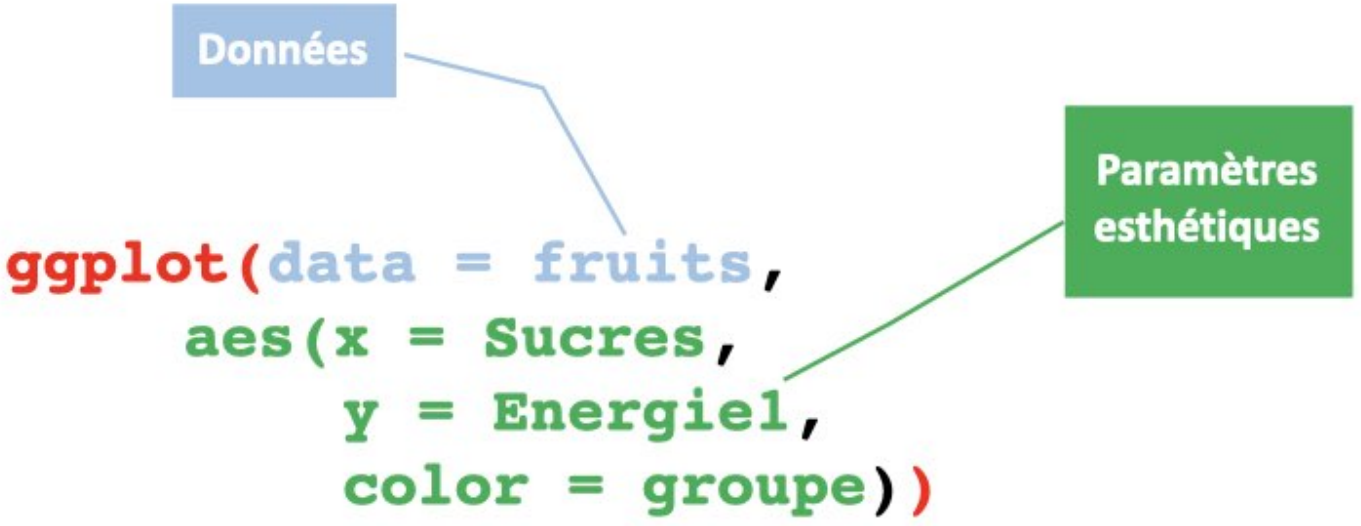
Données

```
ggplot(data = fruits,
```

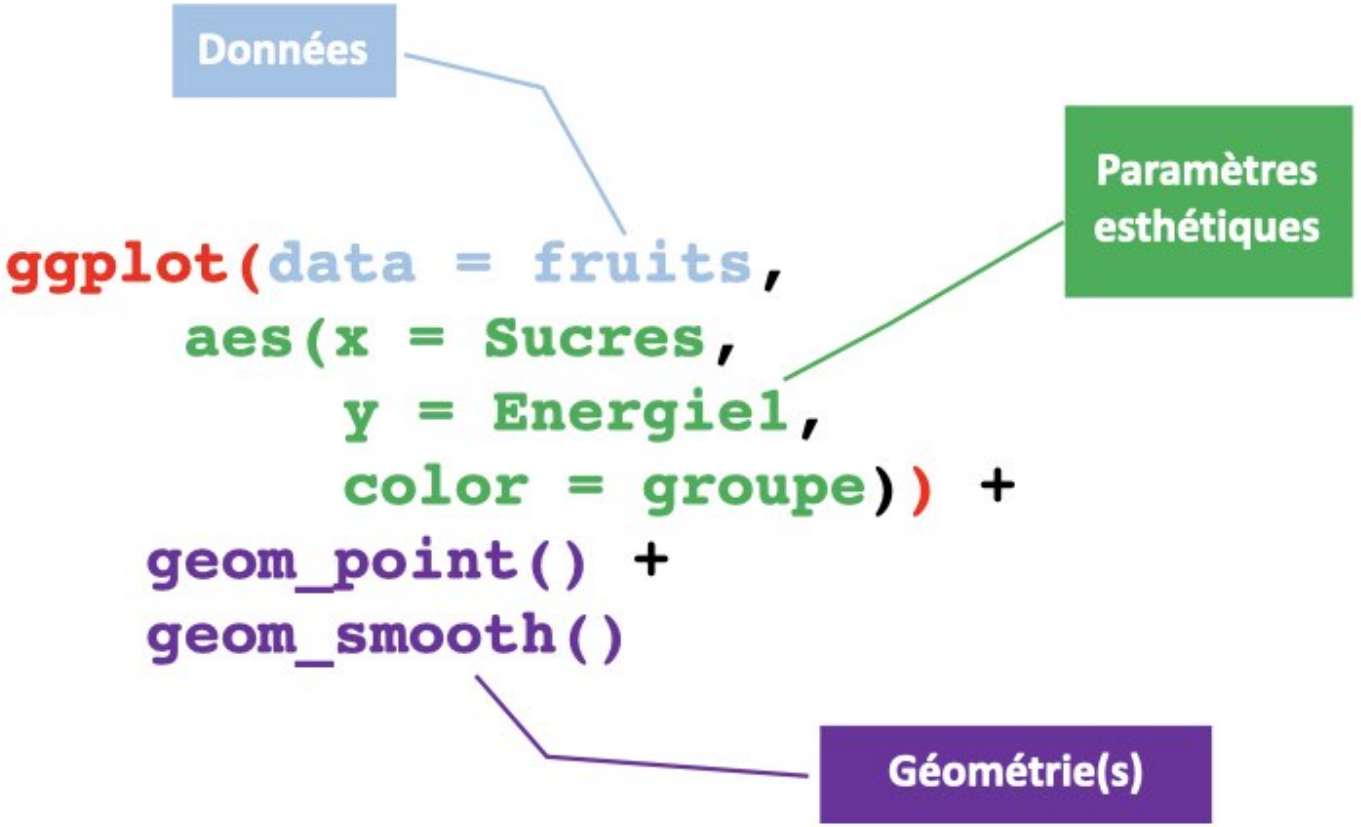


V1	V2	Groups
1	1	Group 1
2	2	Group 2
3	5	Group 1
4	10	Group 1
5	17	Group 2
6	9	Group 1
7	11	Group 1
8	13	Group 2

Aesthetic parameters



Geometries


Données
Paramètres esthétiques
ggplot(data = fruits,
aes(x = Sucres,
y = Energiel,
color = groupe)) +
geom_point() +
geom_smooth()
Géométrie(s)

What you need to remember

[G]rammar of [G]raphics

1) DATA: a set of data operations that create variables from datasets,
2) TRANS: variable transformations (*e.g.*, *rank*),
3) SCALE: scale transformations (*e.g.*, *log*),
4) COORD: a coordinate system (*e.g.*, *polar*),
5) ELEMENT: graphs (*e.g.*, *points*) and their aesthetic attributes (*e.g.*, *color*),
6) GUIDE: one or more guides (*axes*, *legends*, etc.).

The grammar of Graphics, Leland Wilkinson

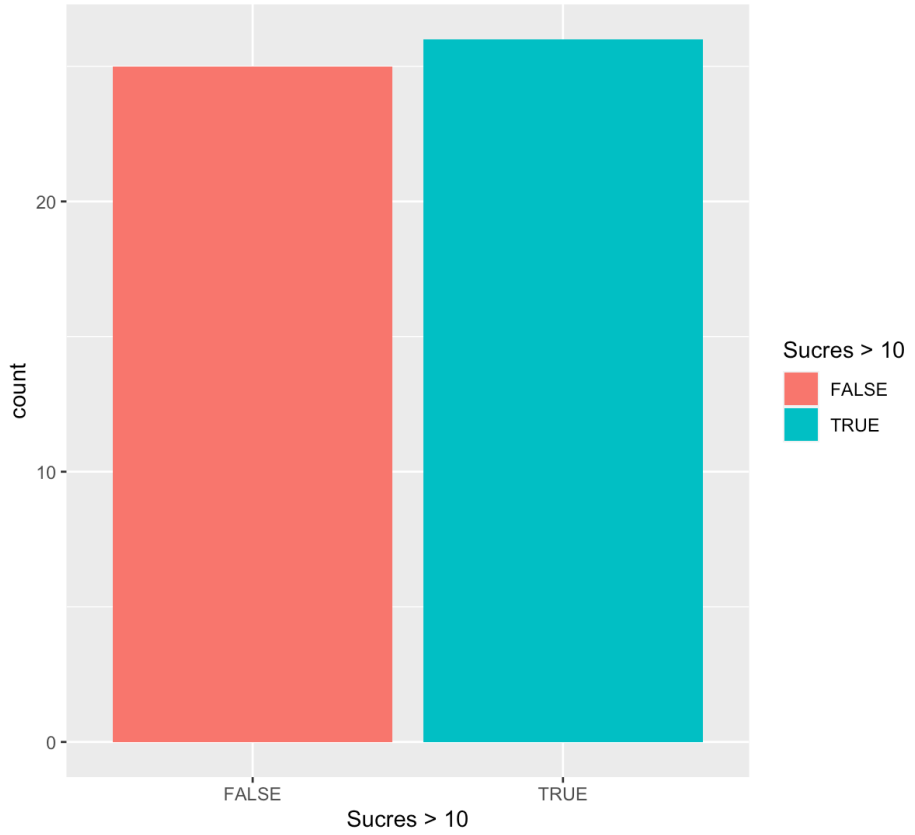
Implementation in ggplot2

Data `da	ta` Th	e data used to create the graph. Each line represents an object to add to the graph.
Geometry `	geom_`	How to represent the objects: point, lines, surfaces etc.
Aesthetics	`aes()`	Aesthetic parameters of the shapes: position, color, shape, size etc.
Scale `sc	ale_` Fu	nctions used to parameter how the shapes are created from the objects and the aesthetic parameters. For example the function `scale_color_manual` allows the users to pick their own colors.

Your turn!

Reproduce the graph on the right:

```
ggplot(***,
    aes(***,
    fill = Sucres > 10)) +
  geom_***()
```



A little bit of history

- There was a `ggplot`"1" (see ici)
- Development began in 2005
- Hadley Wickham (*Chief Scientist at RStudio + Adjunct Professor of Statistics*)
- Excellent courses, sometimes with his sister Charlotte

Hadley Wickham

Some geometries

Nous allons voir ensemble quelques géométries particulières qui permettent de créer des graphes classiques.

`geom_bar`	Bar plot on non-aggregated data
`geom_col`	Bar plot on existing counts
`geom_histogram`	Histogram of a quantitative variable
`geom_boxplot`	Tukey diagram aka *boxplot*
`geom_violin`	“Violin” plot
`geom_point`	Scatter plot
`geom_line`	Line plot

Bar plots

With geom_bar

We already know how to do it:

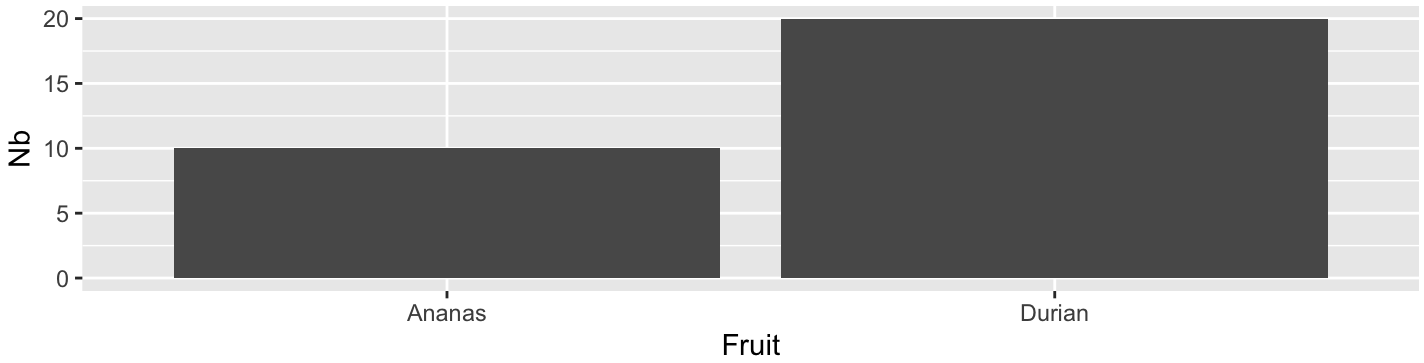
```
ggplot(fruits, aes(cut(Eau, c(0, 84.2, 100)))) +
  geom_bar(fill = "steelblue")
```

With geom_col

When you **already have** counts.

```
dat.count <- data.frame(
  Fruit = c("Ananas", "Durian"),
  Nb = c(10, 20)
)

ggplot(data = dat.count, aes(x = Fruit, y = Nb)) +
  geom_col()
```



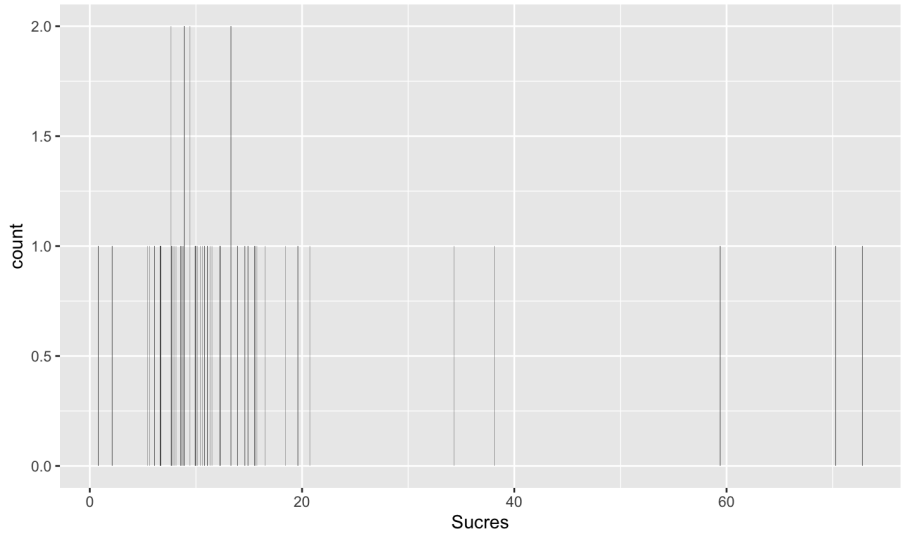
Your turn

Add colors to the previous bar plot!

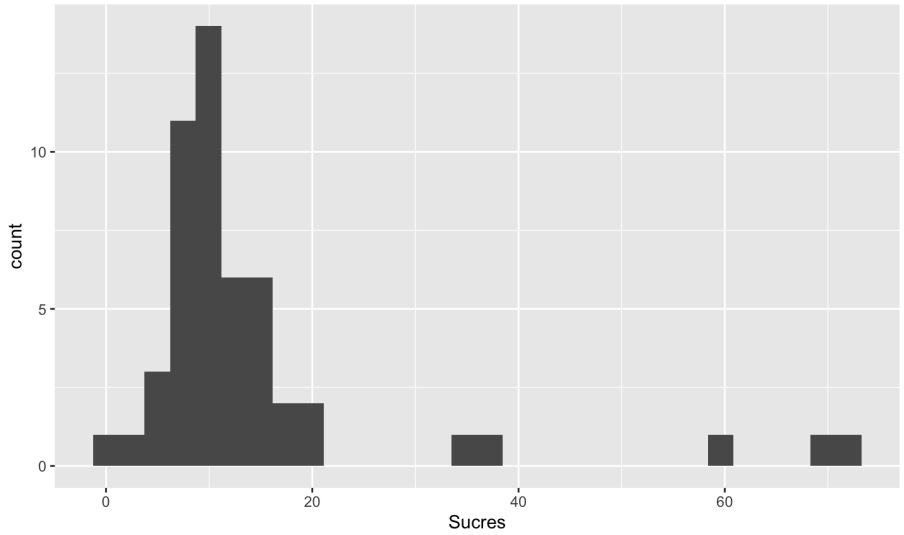
Histograms

Histogram or bar plot?

```
ggplot(fruits, aes(Sucres)) +
  geom_bar()
```




```
ggplot(fruits, aes(Sucres)) +
  geom_histogram()
```



Histogram or bar plot?

Bar plot

To plot counts for :

- Nominal variables
- Ordinal variables
- Discrete variables

Histogram

To plot counts or densities for:

- Continuous variables
- Discrete variables

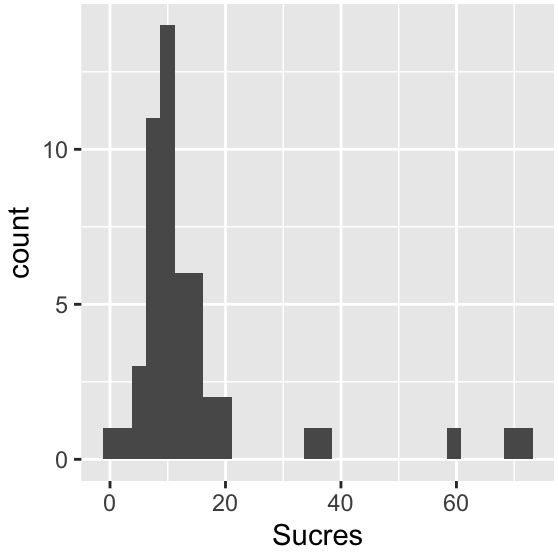
In this case, it is very important to choose the intervals!

Default histogram

- \(y\)-axis: counts for the given interval (also called “class”)
- \(x\)-axis:
  - same width intervals,
  - 30 intervals,
  - no visual separation between them,
  - dark grey *rectangles*,
  - a *message*,

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

```
ggplot(fruits, aes(Sucres)) +
  geom_histogram()
```

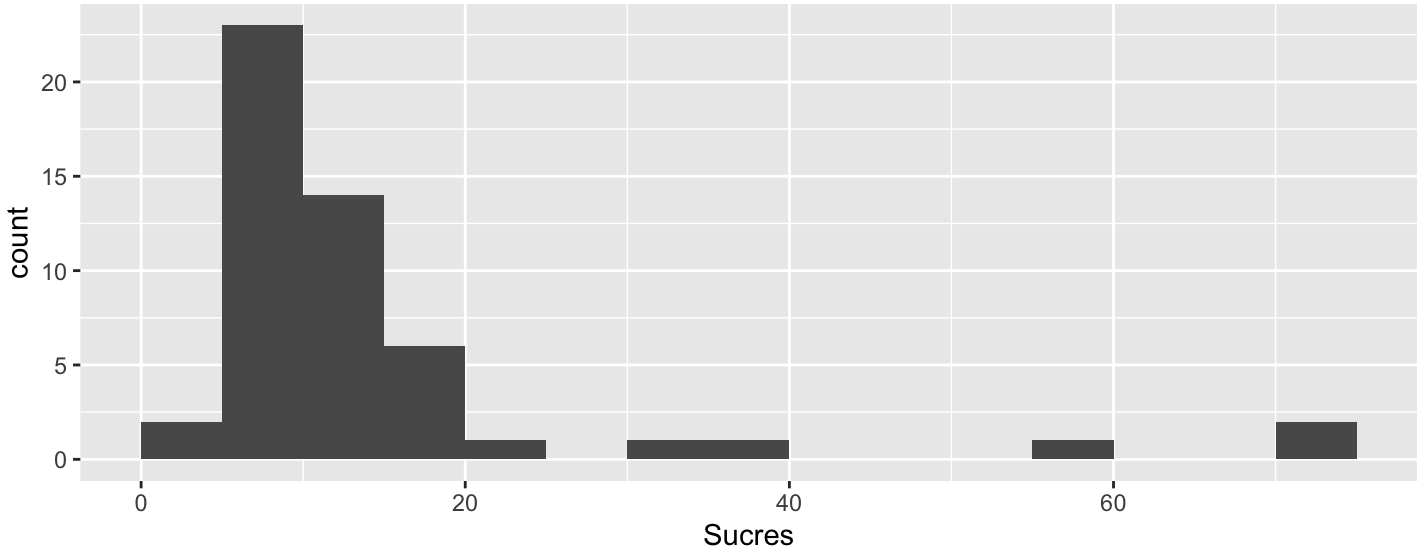


What does the message mean?

To create a histogram, one needs to distributes values into classes.

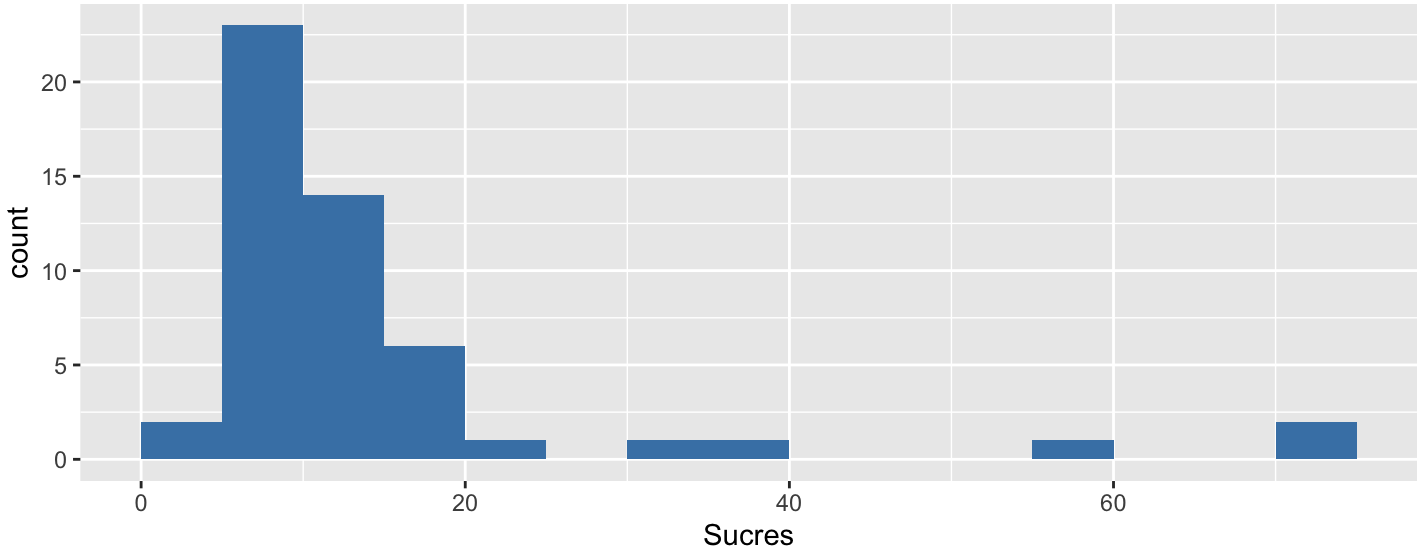
- `hist` does it automatically with an algorithm (Sturges by default, but the user can use Scott, or Friedman-Diaconis algorithms). If `n` is specified, the function will choose a close value for `n` that gives pretty intervals. To force the classes, use `breaks`. *`geom_histogram` create 30 classes by default, it is the user's job to specify their classes or the number of classes they want.

Modify the intervals



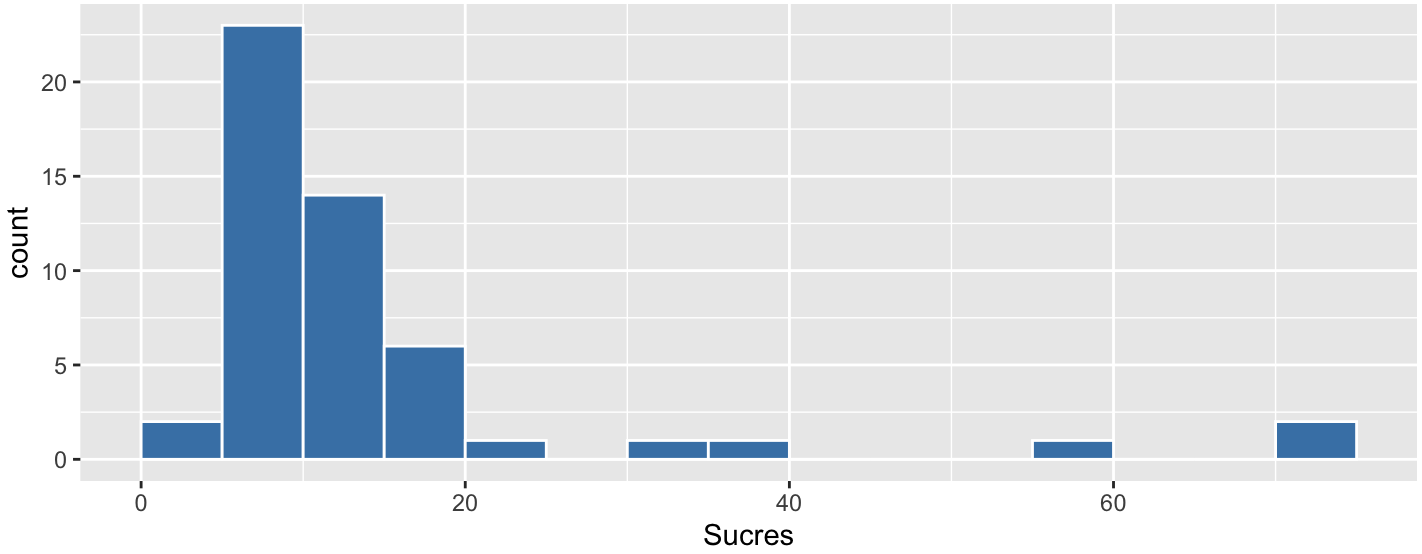

```
ggplot(fruits, aes(Sucres)) +
  geom_histogram(breaks = seq(0, 75, 5))
```

Change the color



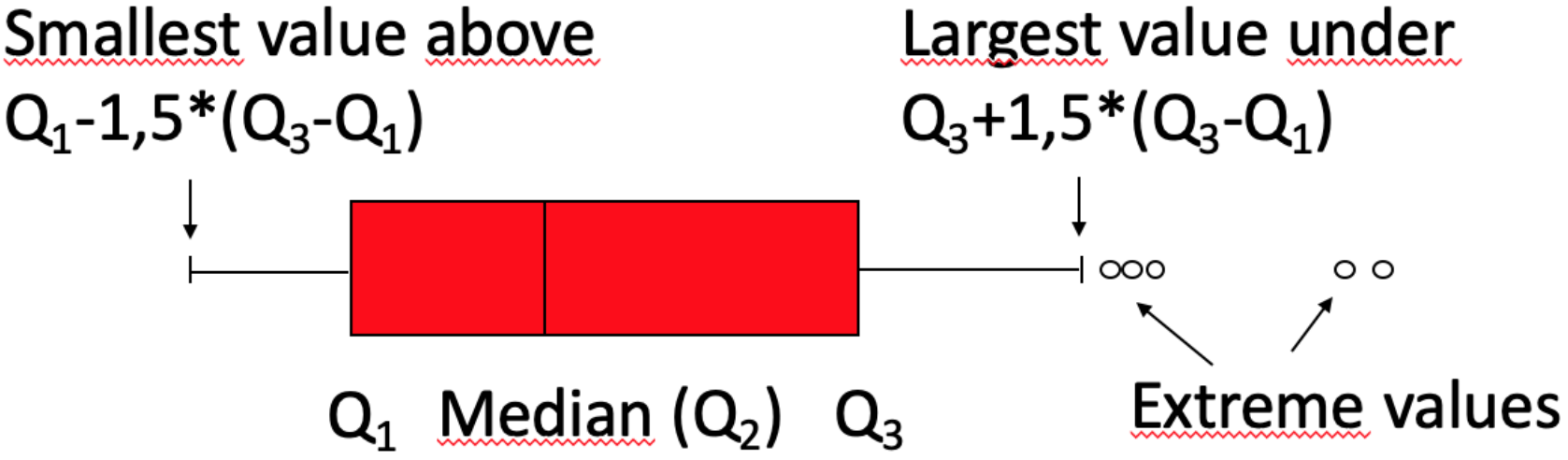

```
ggplot(fruits, aes(Sucres)) +
  geom_histogram(breaks = seq(0, 75, 5),
                 fill = "steelblue")
```

Change the color



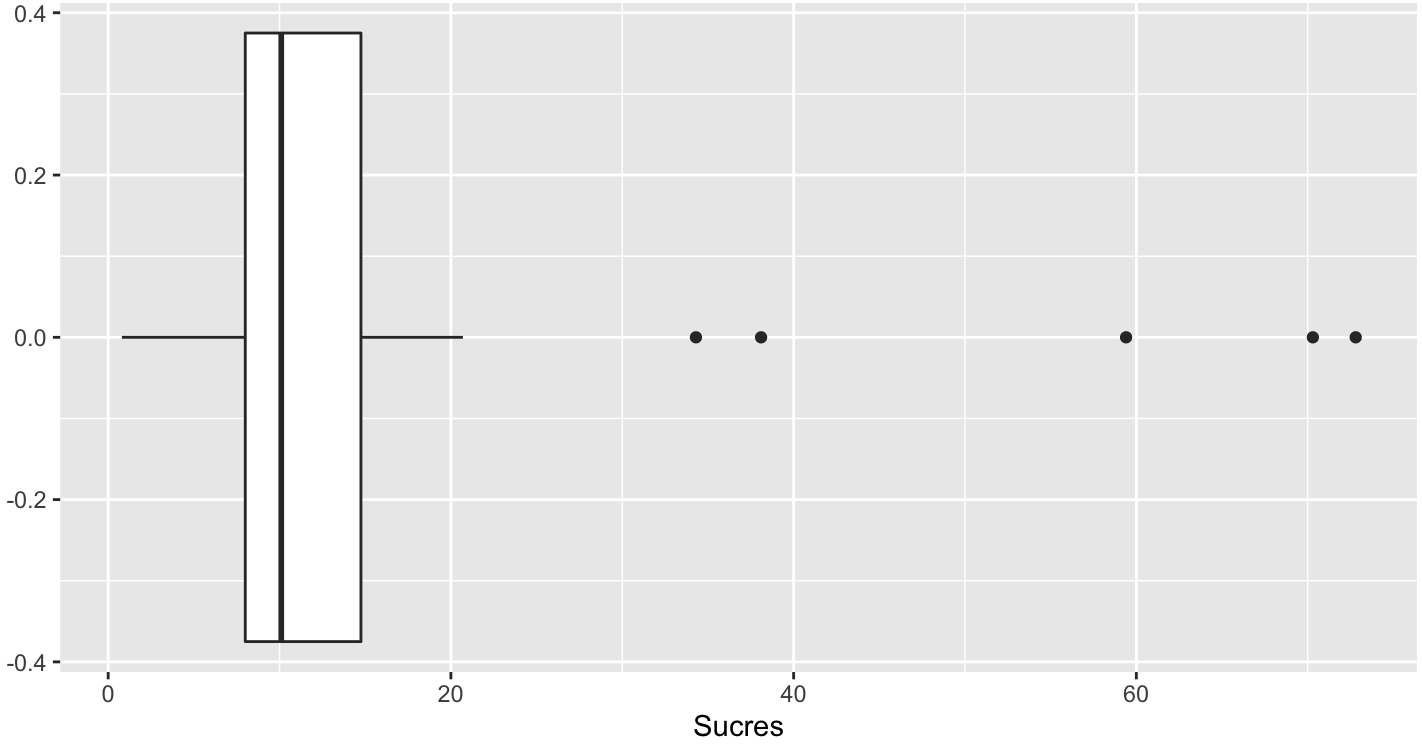

```
ggplot(fruits, aes(Sucres)) +
  geom_histogram(breaks = seq(0, 75, 5),
                 fill = "steelblue",
                 color = "white")
```

Boxplot


Smallest value above
Q_1-1,5*(Q_3-Q_1)
Largest value under
Q_3+1,5*(Q_3-Q_1)
Q_1 Median (Q_2) Q_3
Extreme values

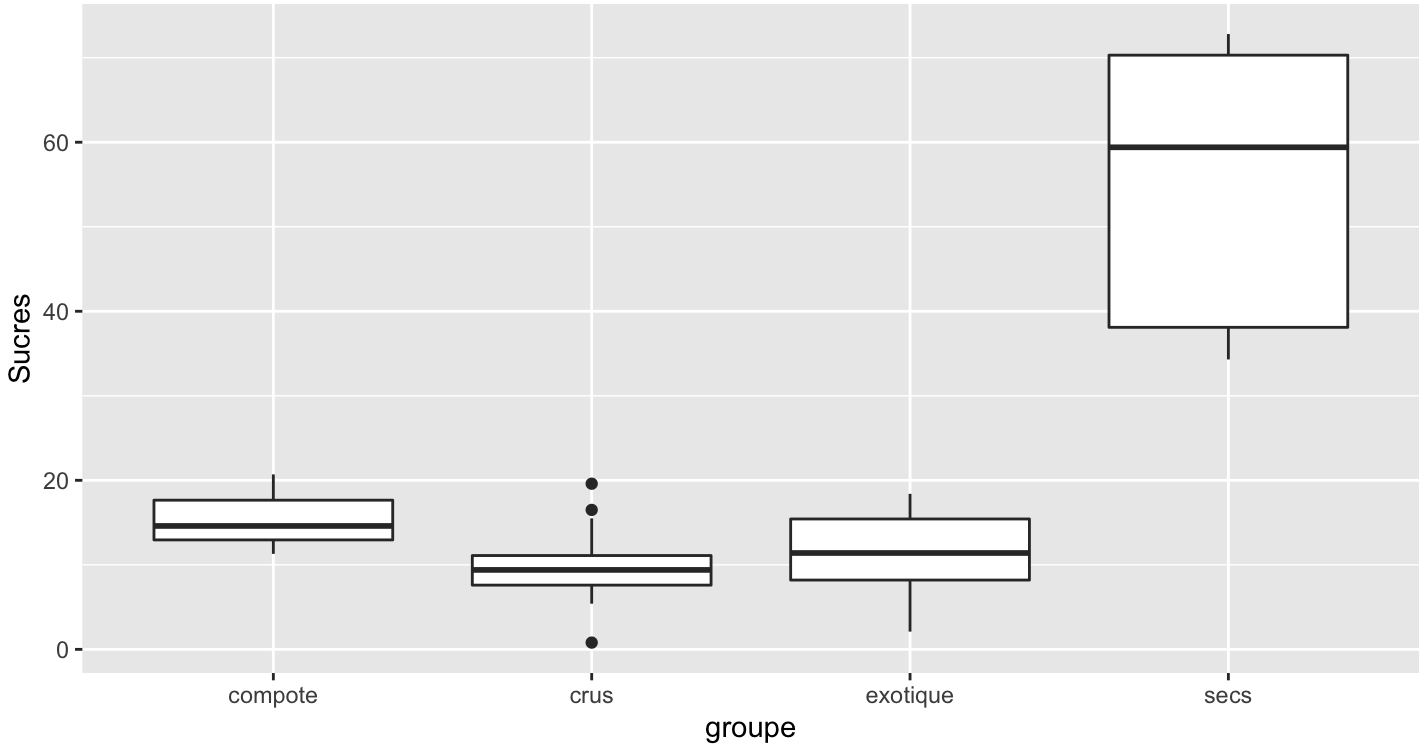
Boxplot

```
ggplot(data=fruits, aes(x = Sucres)) +
  geom_boxplot()
```



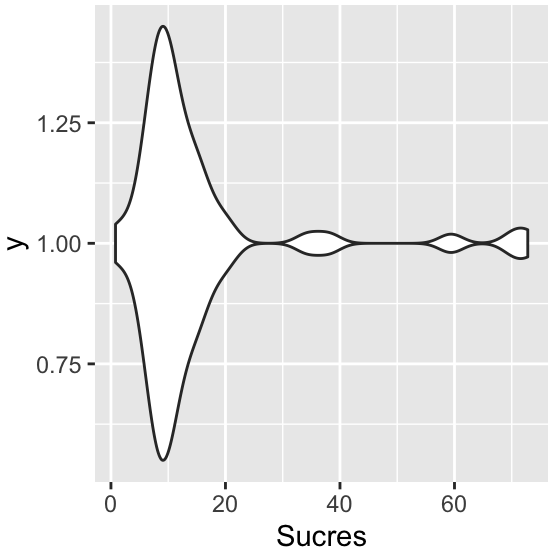
Boxplot : link between a categorical variable and a quantitative variable

```
ggplot(data=fruits, aes(x=groupe, y=Sucres)) +
  geom_boxplot()
```

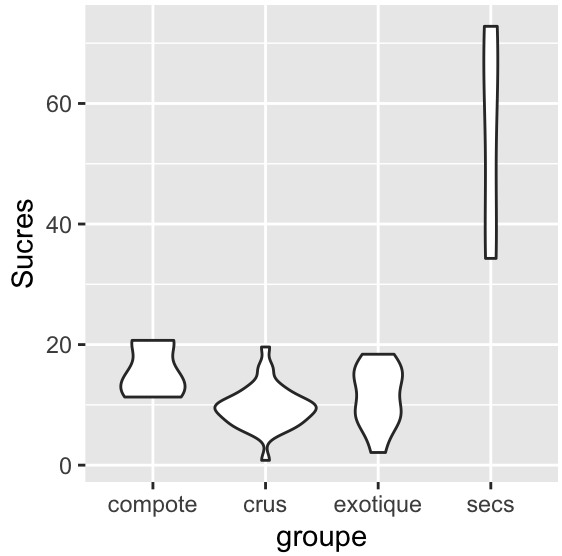


Violins

```
ggplot(data=fruits,
       aes(x = Sucres, y = 1)) +
  geom_violin()
```



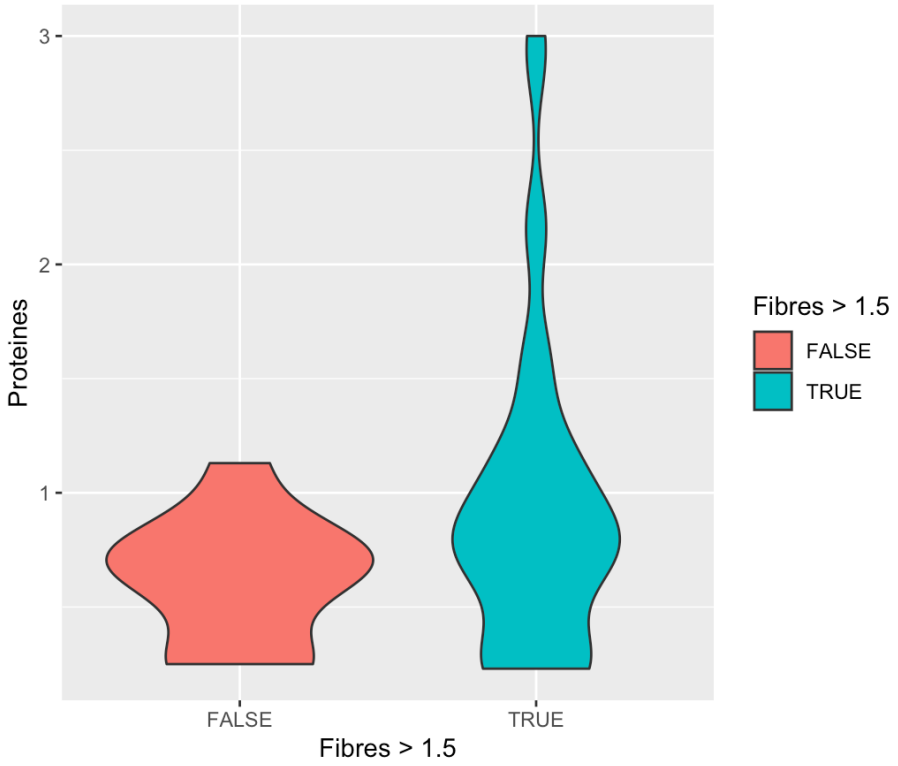

```
ggplot(data=fruits,
       aes(x = groupe, y = Sucres)) +
  geom_violin()
```



Your turn!

Complete the code to obtain the graph on the right:

```
ggplot(fruits,
       aes(x = Fibres > 1.5,
           y = Proteines,
           fill = ***)) +
  geom_***()
```



Customization

Themes

Themes are pre-defined functions that change the appearance of ggplots:

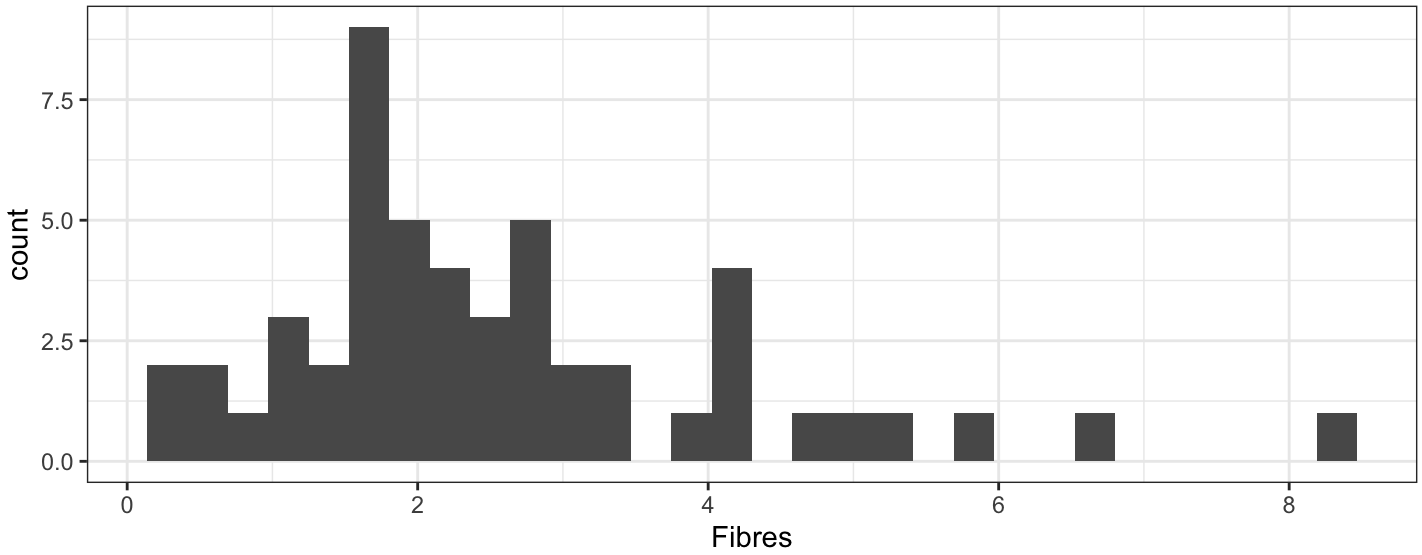
- background color,
- axes color,
- major and minor grids,
- etc.

Examples (`theme_***()`) :

- `theme_bw()` for a black and white theme,
- `theme_minimal()` for a minimalist theme,
- `theme_void()` for an empty theme

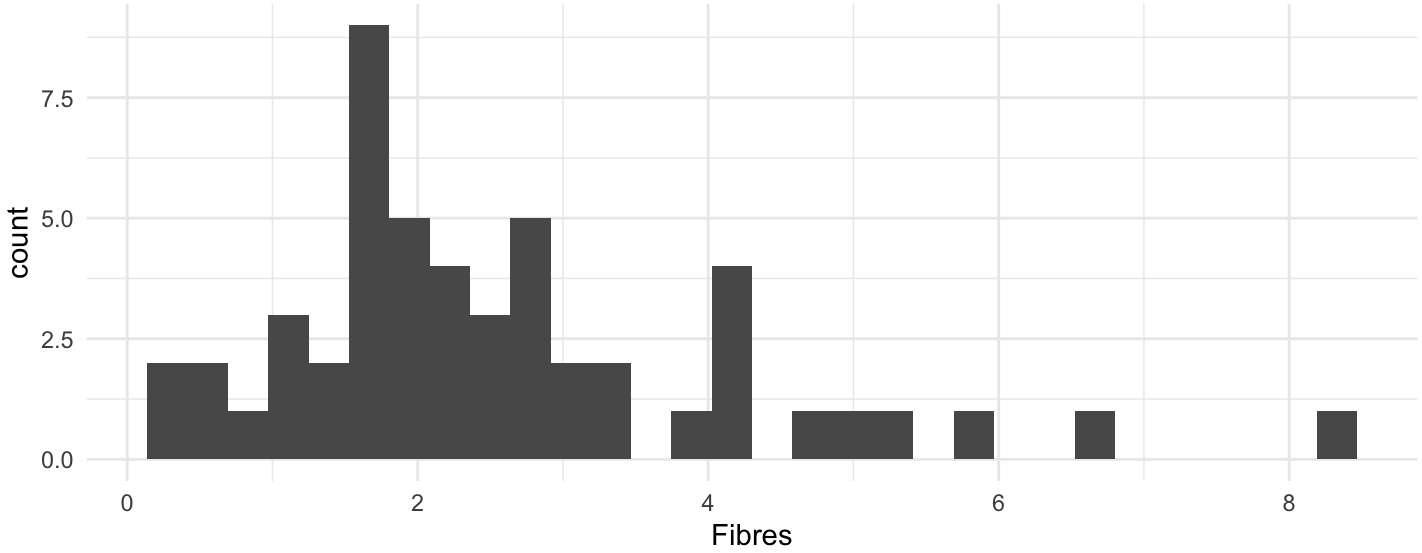
Example on a histogram : `theme_bw()`

```
ggplot(fruits, aes(Fibres)) +
  geom_histogram() +
  theme_bw()
```



Example on a histogram : `theme_minimal()`

```
ggplot(fruits, aes(Fibres)) +
  geom_histogram() +
  theme_minimal()
```



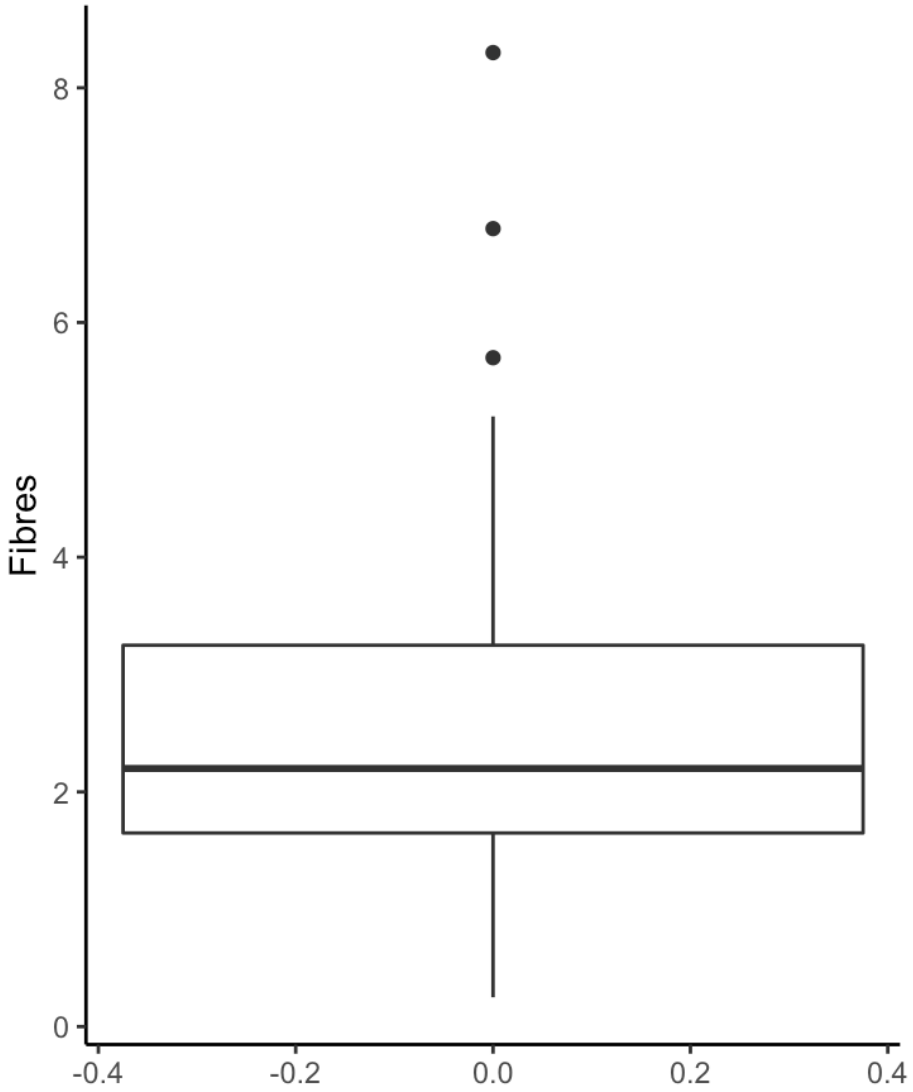
Example on a histogram : `theme_void()`

```
ggplot(fruits, aes(Fibres)) +
  geom_histogram() +
  theme_void()
```

Your turn!

1. Consult the help page for `theme_bw` with the command `?theme_bw`
2. Choose the appropriate theme to obtain the result on the right.

```
ggplot(fruits, aes(y = Fibres)) +
  geom_boxplot() +
  theme_***()
```



Other “simple” customization

- **Titles** : with `ggtitle`
- **Title for the \(x\) axis** : with `xlab`
- **Title for the \(y\) axis** : with `ylab`

Use the wrapper function `labs` to go even faster:

```
labs(
  title = "Titre du graphe",
  subtitle = "Sous-titre du graphe",
  x = "Titre de l'axe des x",
  y = "Titre de l'axe des y",
  color = "Titre de la légende des couleurs",
  shape = "Titre de la légende des formes"
)
```

Advanced customization

With the function `theme()`: each element has to be defined according to its nature.

- To change text, use `element_text(size=, colour = "", family = "")` (e.g. titles)
- To change lines, use `element_line(colour="", size=)` (e.g. major and minor grids)
- To change someting shaped like a rectangle, use `element_rect(fill = "")` (e.g.: background)

Some of the things one can change with `theme()`

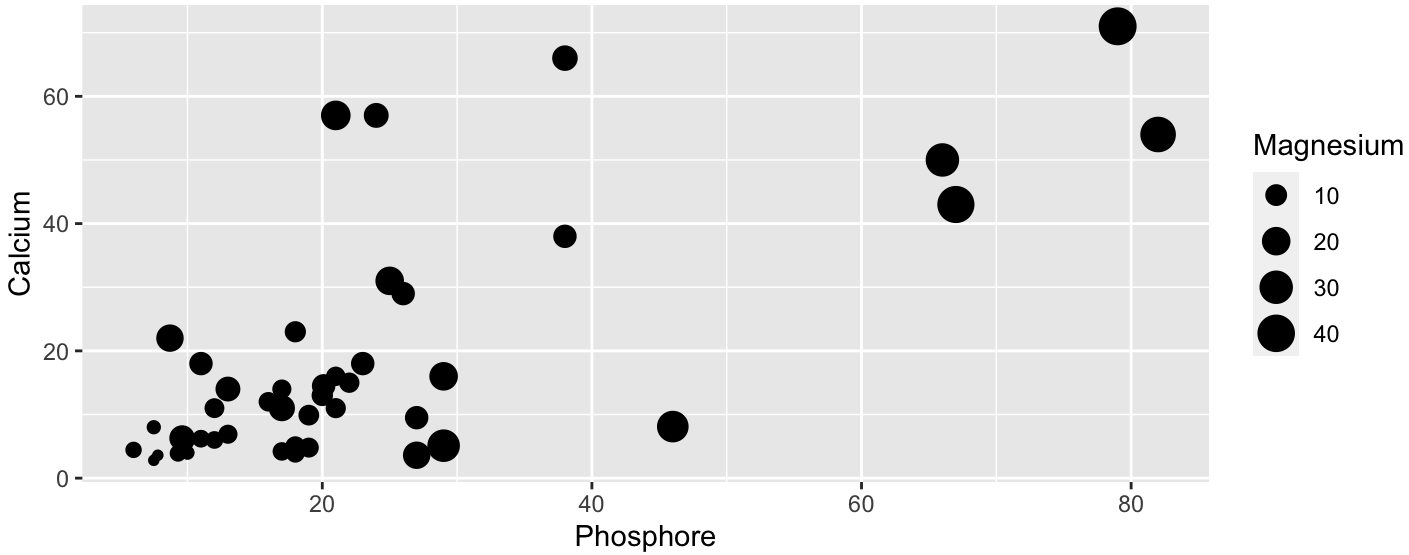
- `axis.title`, `axis.title.x`, `axis.title.y` : size, font, color, …
- `axis.text`, `axis.text.x`, `axis.text.y` : size, font, color, …
- `axis.ticks`, `axis.ticks.x`, `axis.ticks.y`
- `axis.line`, `axis.line.x`, `axis.line.y`
- `panel.background` : color
- `panel.grid.major`, `panel.grid.minor` : color, size
- `legend.text`: size, font, color
- `legend.position`
- `plot.title` : size, font, color

Scatterplots

With `geom_point`

This geometry **needs** \(x\) et \(y\) aesthetic parameters, and will accept optionnally size, color and shape.

```
ggplot(fruits, aes(x = Phosphore, y = Calcium, size = Magnesium)) +
  geom_point()
```



Aesthetic parmaeters

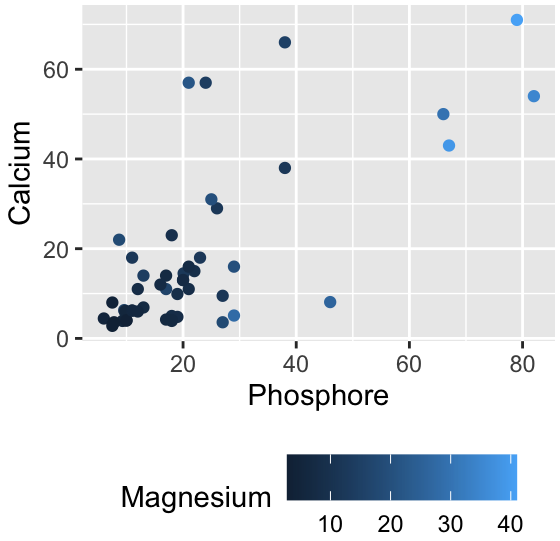
When they are specified in `aes`, they apply values (from the dataset) to a characteristic of the objects that are drawn on the graph.

- `color` or `colour` : color (of the point)
- `fill` : color (inside a shape)
- `size` : size
- `shape` : shape
- `alpha` : opacity
- `linetype` : type of line
- `label` : labels

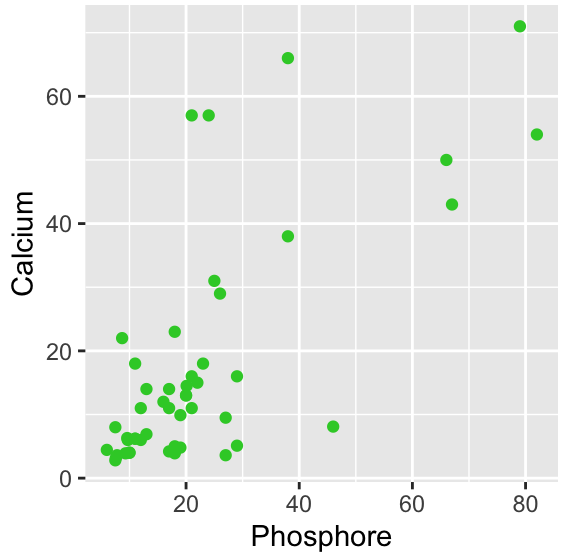
Specified outside of `aes()`, they behave in a more general way!

Example

```
ggplot(fruits,
       aes(x = Phosphore, y = Calcium,
           color = Magnesium)) +
  geom_point() +
  theme(legend.position = "bottom")
```



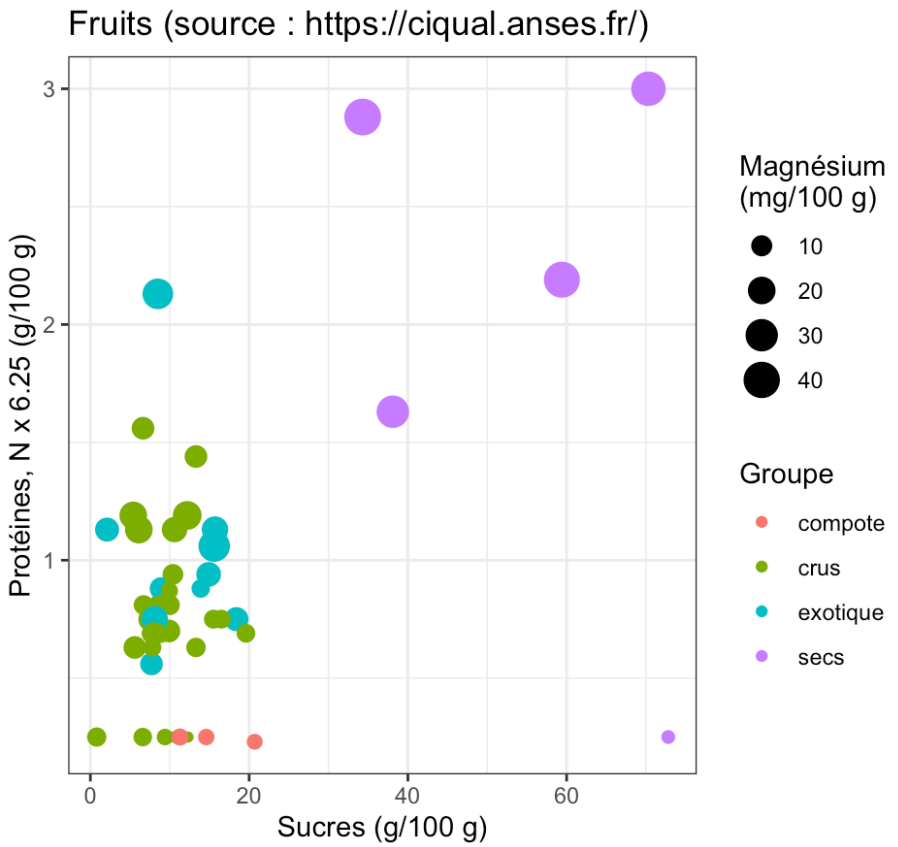

```
ggplot(fruits,
    aes(x = Phosphore, y = Calcium)) +
  geom_point(color = "limegreen")
```



Your turn!

Complete the code to obtain the graph on the right:

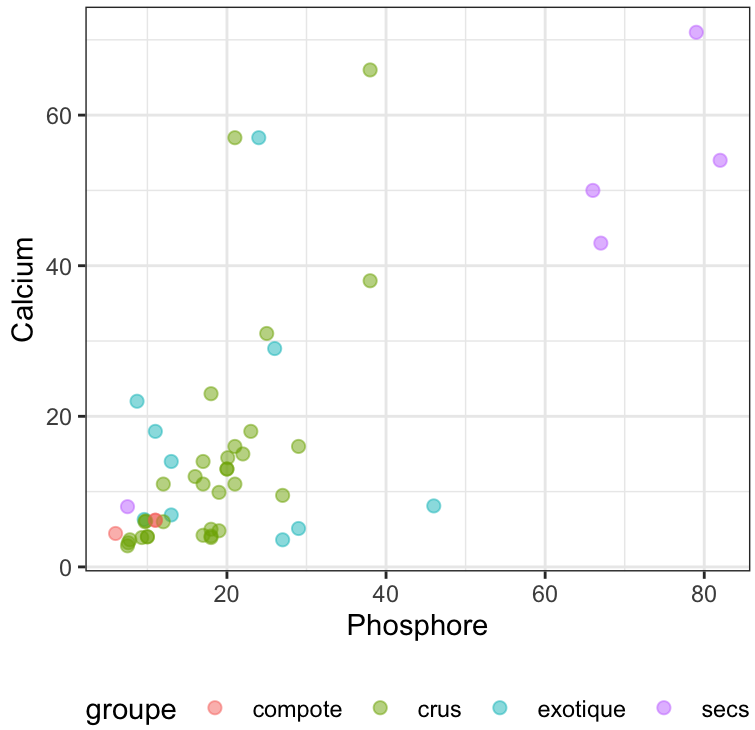
```
ggplot(fruits,
       aes(x = Sucres,
           y = Proteines,
           *** = Magnesium,
           *** = ***)) +
  geom_***() +
  ***(title = "Fruits",
     x = "Sucres (g/100 g)",
     y = "Protéines, N x 6.25 (g/100 g)",
     size = "Magnésium\n(mg/100 g)",
     ***= "Groupe") +
  theme_***()
```



Help, my dots are on top of one another!

Don’t panick, use opacity (aka `alpha`) :

```
ggplot(fruits,
       aes(x = Phosphore,
           y = Calcium,
           color = groupe)) +
  geom_point(alpha = 0.5,
             size = 2) +
  theme_bw() +
  theme(legend.position =
          "bottom")
```



Changing the scales

With the `scale_***` functions

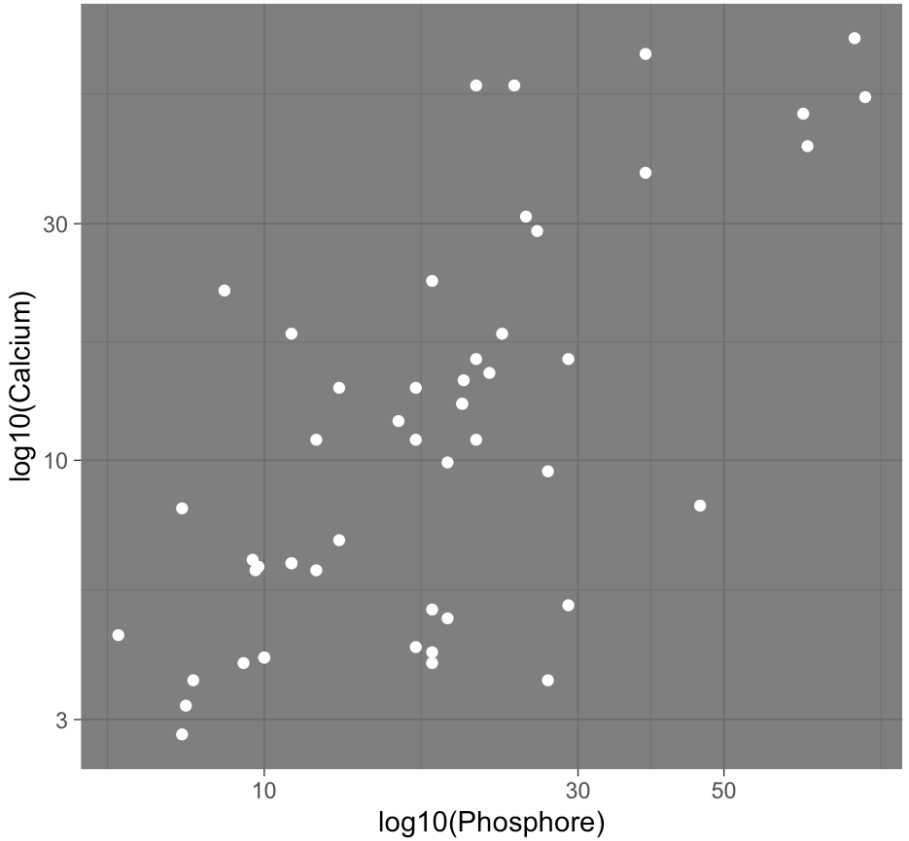
They allow the use to customize a scale (in \(x\) or \(y\) but not only)!

- `scale_x_log10()` changes the \(x\) scale to a logarithmic scale,
- `scale_y_log10()` changes the \(y\) scale to a logarithmic scale,
- `scale_color_manual()` customizes the colors,
- `scale_fill_manual()` cutomizes the colors inside shapes,
- `scale_x_continuous()` customizes the \(x\) scale for a continuous variable,
- `scale_y_continuous()` customizes the \(y\) scale for a continuous variable,
- `scale_x_discrete()` customizes the \(x\) scale for a discrete variable,,
- `scale_y_discrete()` customizes the \(y\) scale for a discrete variable,,

Your turn!

Complete the code to obtain the graph on the right:

```
ggplot(fruits,
       aes(Phosphore,
           Calcium)) +
  geom_point(*** = "white") +
  scale_***() +
  scale_***() +
  labs(x = "log10(Phosphore)",
       y = "log10(Calcium)") +
  theme_dark()
```



With the `coord_***` functions

They allow the user to change the coordinate system **after** applying all the scaling transformations (with `scale_***` functions). For example:

- `coord_fixed` to fix the ratio between the units on the \(y\) axis and the units on the \(x\) axis,
- `coord_equal` when the ratio is set to 1,
- `coord_flip` to flip the axes,
- `coord_polar` to get a plot in the polar coordinate system.

With the `*lim*` functions

That allow the users to specify the limits (minimum and maximum) on a specified axis. Caution: the values outside are **eliminated** from the graph!

- `xlim`, `ylim` or `lims` to change ghe range,
- `expand_limits` to *extend* the range.

To “zoom in” without loosing data, use `coord_cartesian` or `scale_***`

“Facetting”

With `facet_wrap`

Used to divide the graph into panels.

Careful about the syntax: it is based on `vars`.

To divide a graphe `g` into several panels according to the value of a factor `fac`:

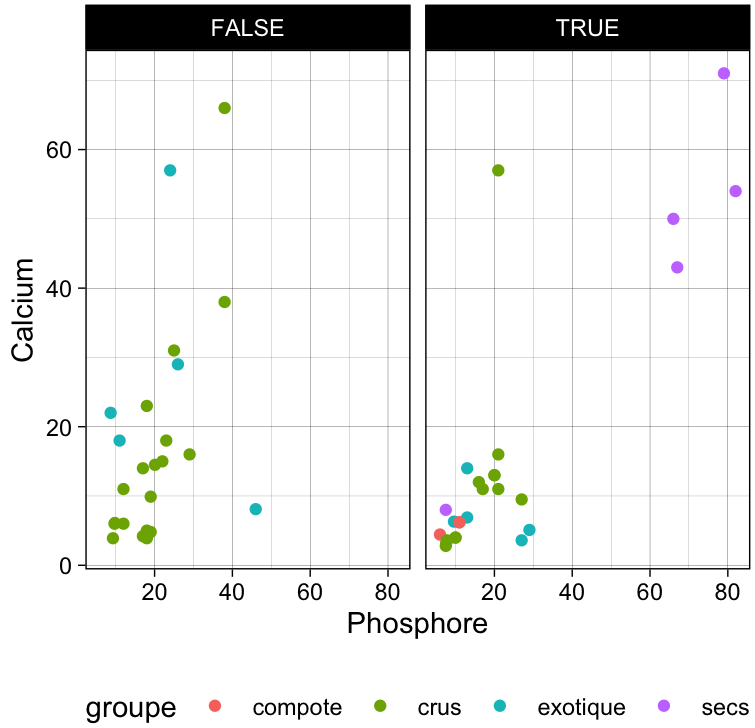
```
g + facet_wrap(facets = vars(fac))
```

One can also use a “formula” :

```
g + facet_wrap(~ fac)
```

Example

```
ggplot(fruits,
       aes(x = Phosphore,
           y = Calcium,
           color = groupe)) +
  geom_point() +
  facet_wrap(vars(Sucres > 10)) +
  theme_bw() +
  theme(legend.position =
          "bottom")
```



Or with `facet_grid`

That is used the same way as `facet_wrap`.

To divide a graphe `g` into several panels according to the value of a factor `factorow` for the lines and `factocol` for the columns:

```
g + facet_grid(rows = vars(factorow), cols = vars(factocol))
```

One can also use a “formula” :

```
g + facet_grid(factorow ~ factocol)
```

A PIECE OF ADVICE: when using facetting, be careful about the levels of the categorical variables that your are going to use.

Save a graph

The easiest method: ggsave

Use and example:

```
g <- ggplot(fruits, aes(groupe)) + geom_bar()
ggsave(filename = "mongraphe.png", plot = g)
```

The extension given in `filename` will be magically used to save the graph in the correct format!

Conclusion

`gplot2` is very complete :

- Use the cheatsheet !
- Go to the online documentation