UE Visualisation

2019-2020

Dr. Maxime Wack

AHU Informatique médicale

Hôpital Européen Georges Pompidou,
Université de Paris

Objectif

Ajouter de l'**interactivité** à une visualisation

Interactivité

Réactivité

L'objet **réagit** à l'utilisateur, mais n'est pas modifié

Interactivité

L'utilisateur peut **influer** sur l'objet
Un objet peut en **modifier** un autre

Outils

ggplot2 + ggplotly

plotly

D3

VegaLite

Shiny

ggplot2 + ggplotly

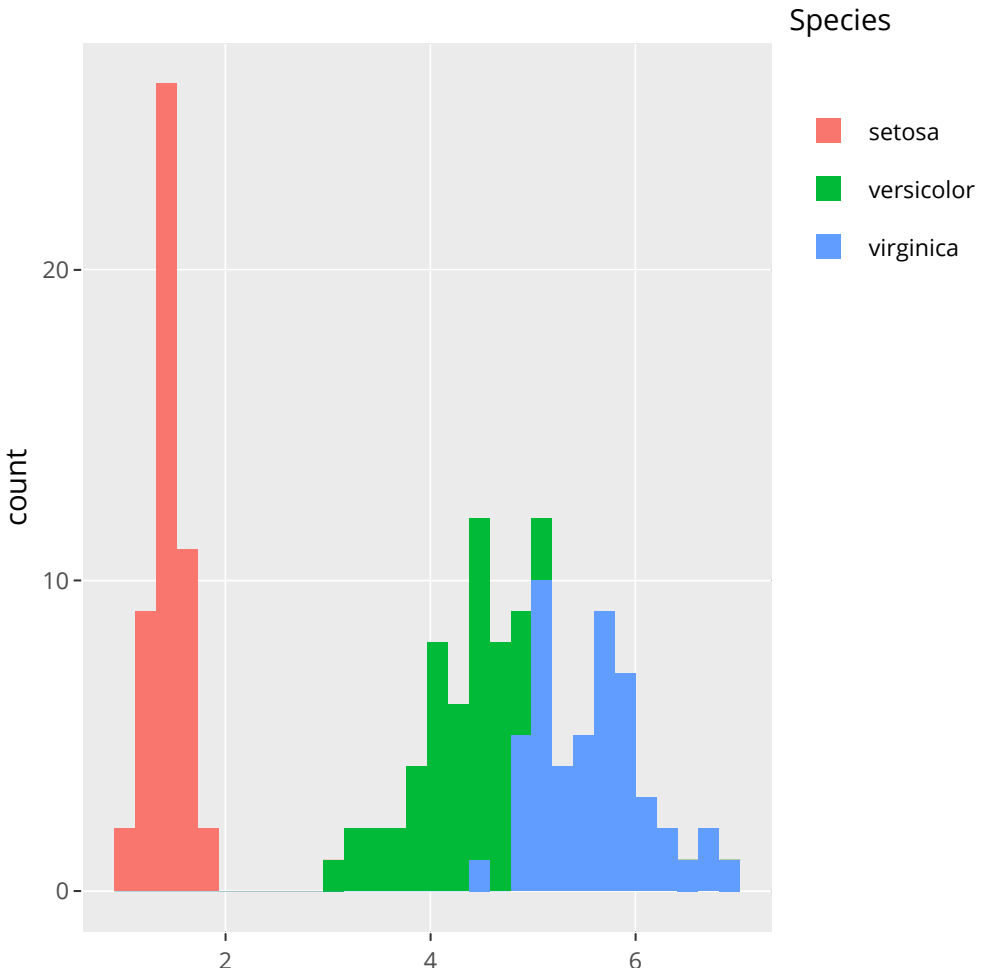
Installer `plotly`

```
install.packages("plotly")
```

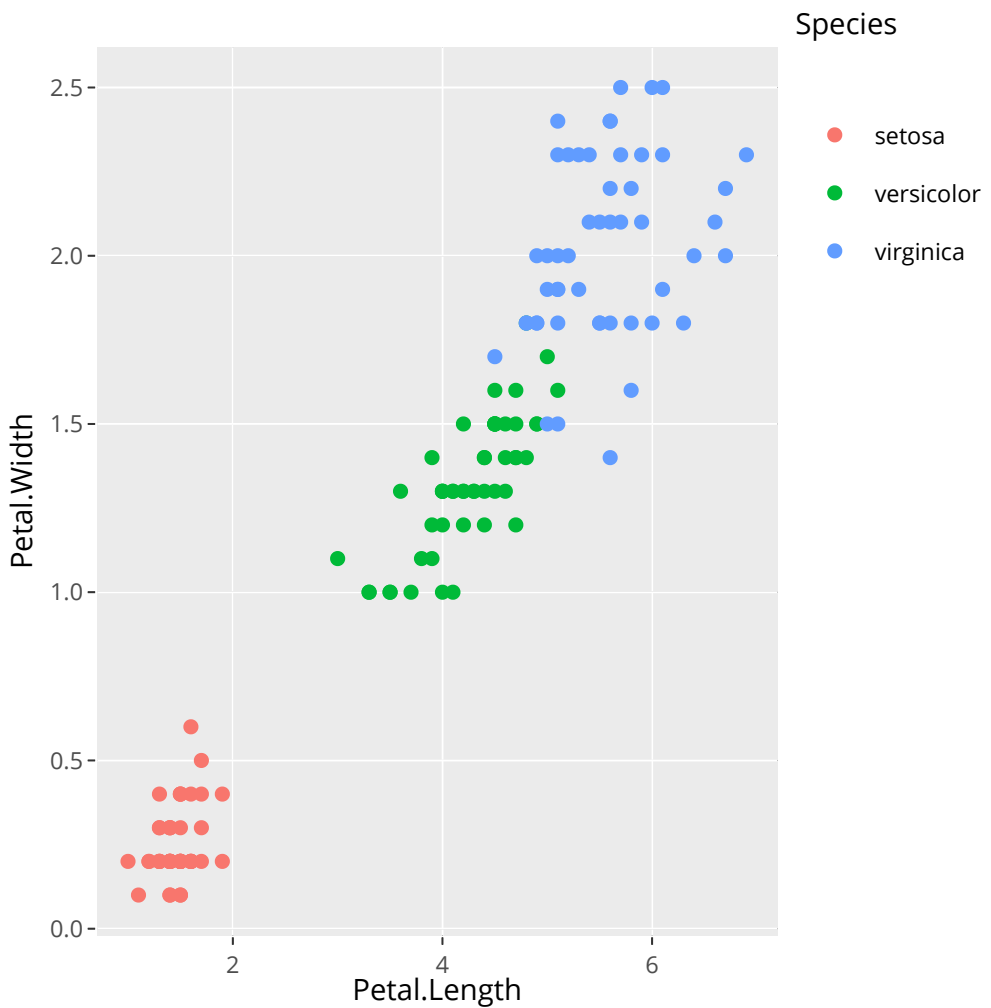
```
library(tidyverse)
library(plotly)
```

```
iris %>%
  ggplot() +
  aes(x = Petal.Length, fill = Species) +
  geom_histogram() -> plot

ggplotly(plot)
```

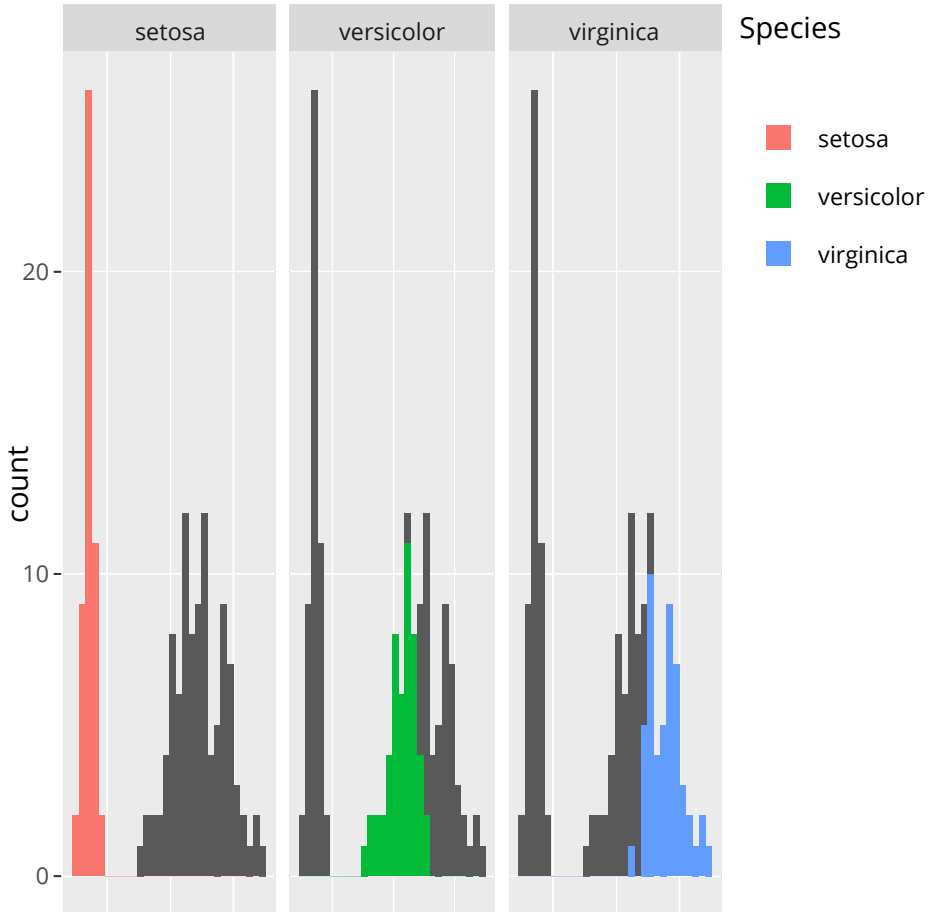


```
(iris %>%
  ggplot() +
   aes(x = Petal.Length, y = Petal.Width, color = Species) +
   geom_point()) %>%
  ggplotly
```



```
iris %>%
  ggplot() +
  geom_histogram(data = iris %>% select(-Species), aes(x = Petal.Leng
  geom_histogram(aes(x = Petal.Length, fill = Species)) +
  facet_grid(~Species) -> plot

ggplotly(plot)
```



Plotly

Plotly

Utilise aussi une **grammaire de graphiques**

Bibliothèque **JS**

Interface par un **package R**

Couche d'**abstraction** pour ggplot2

https://plot.ly/r

Plotly

`ggplot` → `plot_ly()`

`geom_*` → `add_*` ou `add_trace`

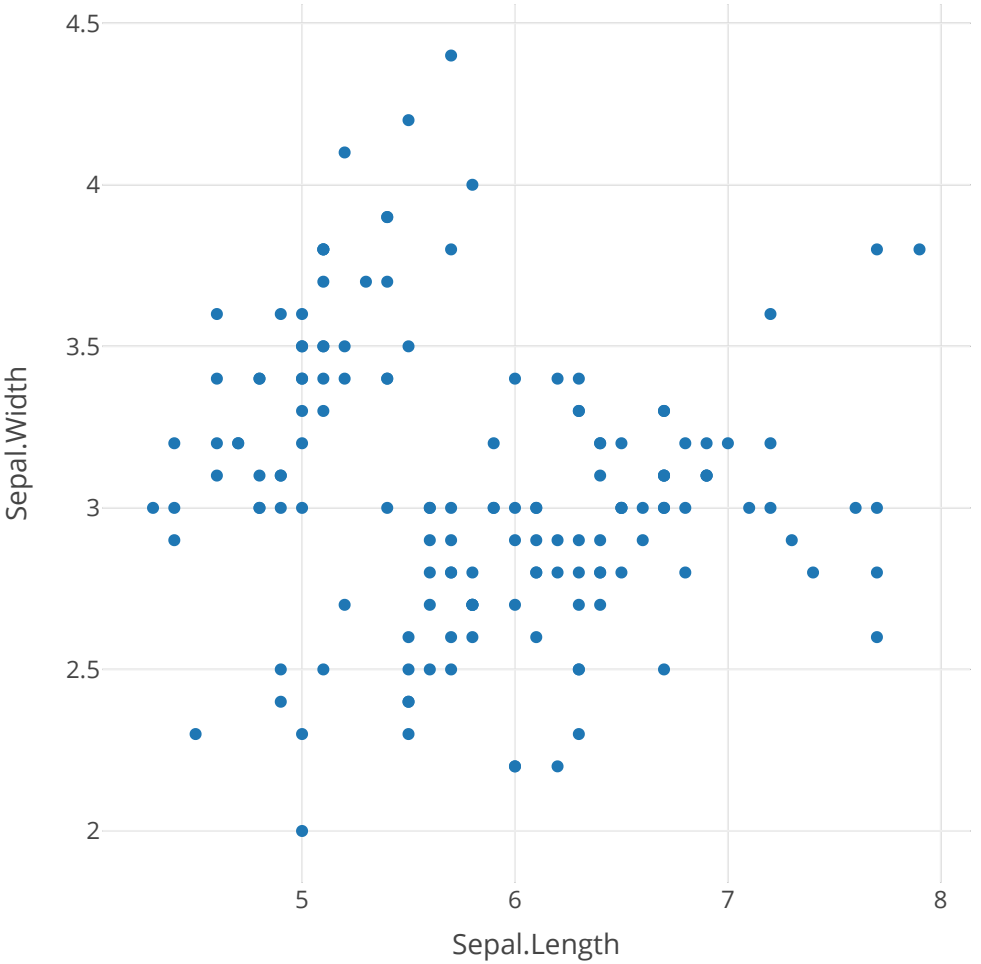
Les *aes* sont données avec des formules dans les traces

`facet_*` → **subplots**

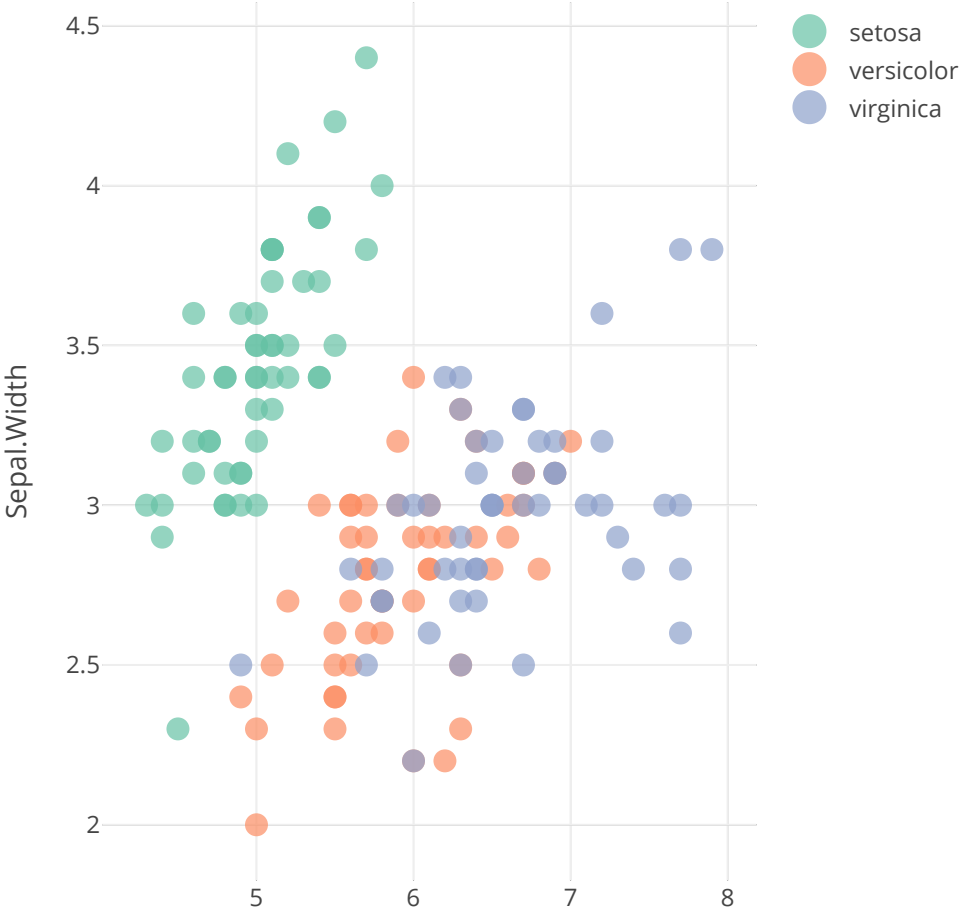
`scale_*_*` + `theme` → `layout`

`+` → `%>%`

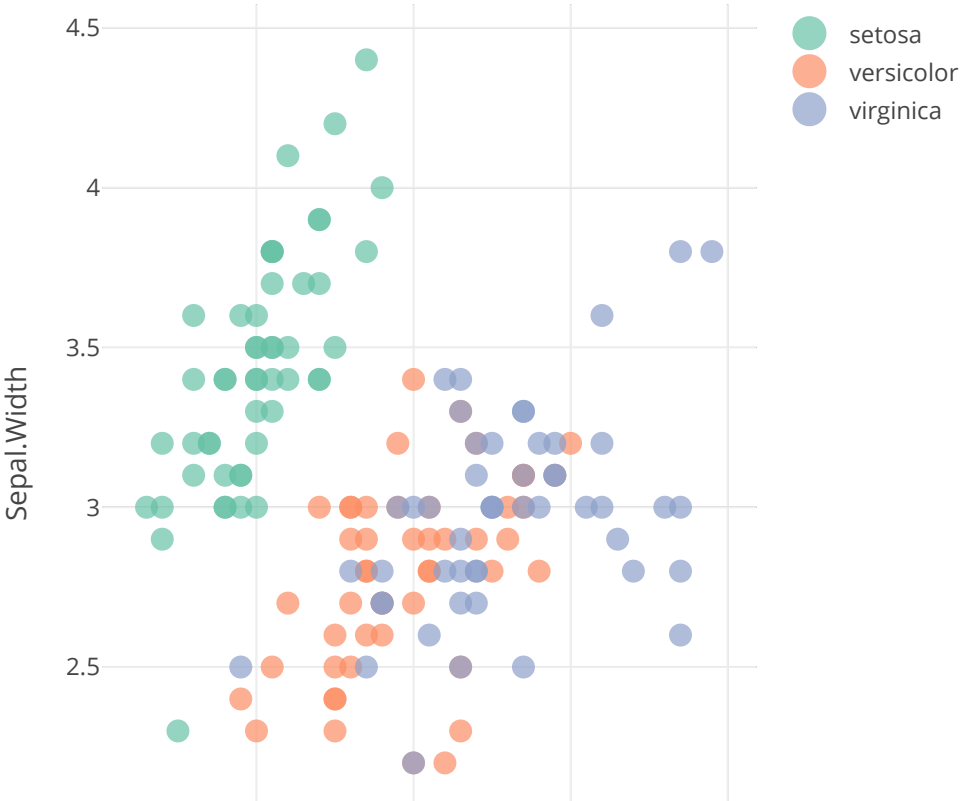
```
iris %>%
  plot_ly() %>%
  add_markers(x = ~Sepal.Length,
              y = ~Sepal.Width)
```



```
iris %>%
  plot_ly(x = ~Sepal.Length,
          y = ~Sepal.Width) %>%
  add_trace(color = ~Species,
            size = 4,
            mode = "markers")
```



```
iris %>%
  plot_ly(x = ~Sepal.Length,
          y = ~Sepal.Width,
          color = ~Species) %>%
  add_markers(size = 2,
              hoverinfo = "text",
              text = ~str_c("Espèce : ", Species, "\n",
                            "Longueur de sépale :", Sepal.Length, "\r
                            "Largeur de sépale :", Sepal.Width))
```

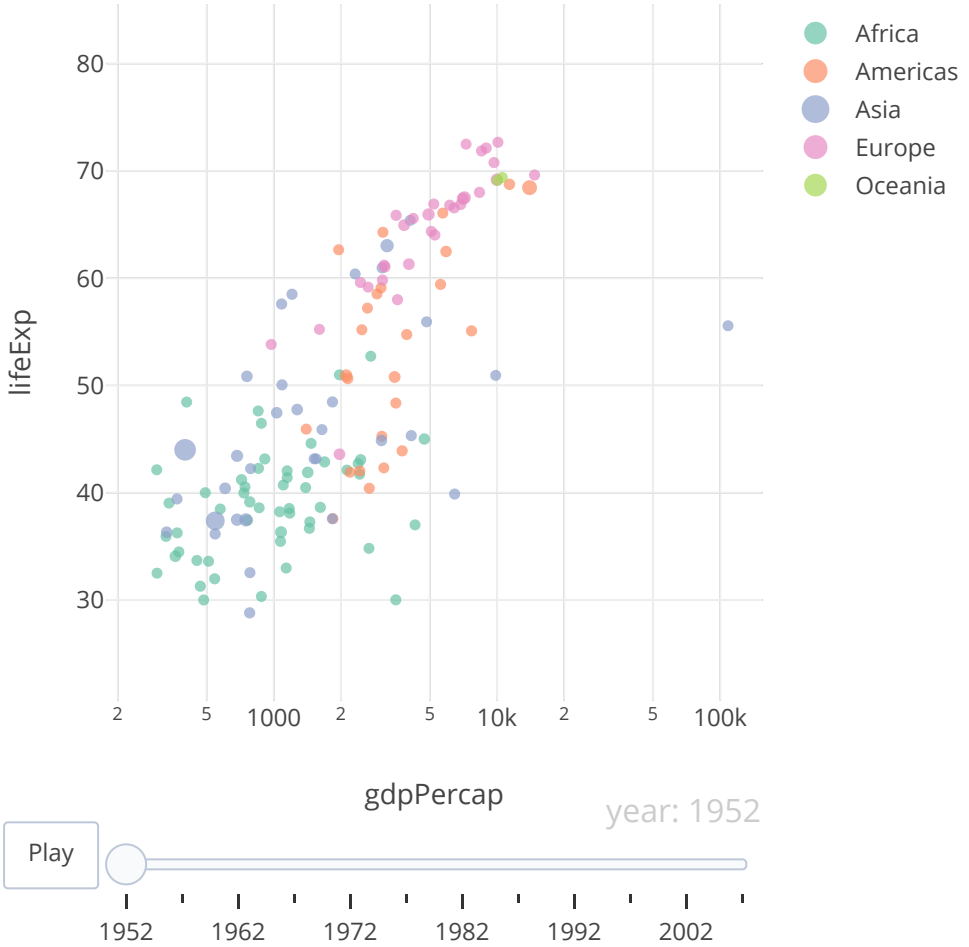


Plotly

Interaction poussée (JS)

Animations (frame)

Aesthetic d'animation, variable définissant l'état à chaque image clé


Africa
Americas
Asia
Europe
Oceania
80
70
60
50
40
30
lifeExp
2
5
1000
2
5
10k
2
5
100k
gdpPercap
year: 1952
Play
1952
1962
1972
1982
1992
2002

D3

Data **D**riven **D**ocuments

Créé par **Mike Bostock**

Data journalist au NYT

Moteur *bas niveau* pour manipuler des éléments du DOM et du SVG à partir de données.

Notion de grammaire liant données et propriétés CSS, dont **animations** et **transitions**

https://d3js.org

Outils dérivés de D3.js

Plot.ly

Vega(lite)

C3.js

R2D3 https://rstudio.github.io/r2d3/

Réactivité

Tous les outils précédents sont **réactifs**

→ affichage d'un overlay

→ changement de position/échelle

→ afficher / masquer / réordonner

(modulo interactions en *JS* à bricoler soi-même)

Interactivité

Manipulation des données

Interactions avec la visualisation

Interactions entre éléments de visualisation

Shiny

Architecture client/serveur

Client = interface web

Serveur = runtime R

https://shiny.rstudio.com/

Shiny

Programmation **événementielle** → programmation **réactive**

L'interface est mise à jour *quand nécessaire*

Interface web

Créée dans R

Génère HTML + CSS + JS

Définit des **input** et **output**

UI

```
ui <- fluidPage(titlePanel = "Titre",
                sidebarLayout(
                    sidebarPanel(
                      sliderInput(inputId = "bins",
                                  label = "Bins",
                                  min = 1,
                                  max = 50,
                                  value = 30,
                                  animate = animationOptions(interval
                    mainPanel(plotOutput(outputId = "figure"))))
```

Serveur

Créé et exécuté dans R

Génère les éléments d'**output** en fonction des **input**

Les **outputs** peuvent *générer* de l'input (htmlwidgets : DT, plotly, etc.)

Le serveur peut créer des **input** dynamiques

Server

```
server <- function(input, output, session)
{
  set.seed(1234)
  normale <- tibble(val = rnorm(1000))

  output$figure <- renderPlot(
  {
    normale %>%
      ggplot() +
      aes(x = val) +
      geom_histogram(bins = input$bins)
  })
}
```