

Visualisation avec R

Laurent Rouvière

2021-02-02

Présentation

- **Prérequis** : niveau avancé en **R** - bases en statistique et programmation
- **Objectifs** :
 - Comprendre l'importance de la visualisation en **science des données**
 - Visualiser des **données**, des **modèles** et des **résultats**
 - Découvrir quelques packages de visualisation en **R**

Présentation

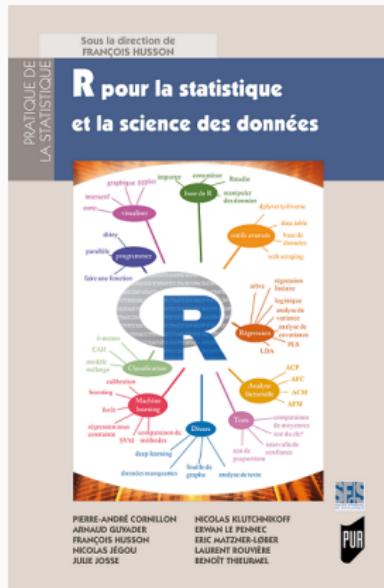
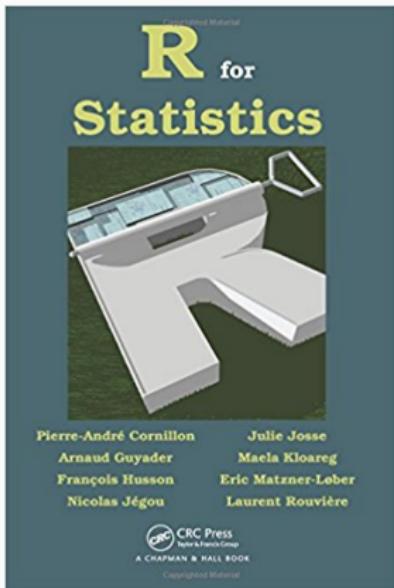
- **Prérequis** : niveau avancé en **R** - bases en statistique et programmation
- **Objectifs** :
 - Comprendre l'importance de la visualisation en **science des données**
 - Visualiser des **données**, des **modèles** et des **résultats**
 - Découvrir quelques packages de visualisation en **R**
- **Enseignant** : Laurent Rouvière, laurent.rouviere@univ-rennes2.fr
 - **Recherche** : statistique non paramétrique, apprentissage statistique
 - **Enseignement** : probabilités et statistique (Université, écodes, formation continue)
 - **Consulting**: énergie (ERDF), finance, marketing, sport

Ressources

- Slides et tutoriel (compléments de cours + exercices) disponibles à <https://lrouvriere.github.io/VISU/>
- Le web

Ressources

- Slides et tutoriel (compléments de cours + exercices) disponibles à <https://lrouviere.github.io/VISU/>
- Le web
- Livres



Pourquoi un cours de visualisation ?

- **Données** de plus en plus **complexes**
- **Modèles** de plus en plus **complexes**
- **Interprétations** des résultats de plus en plus **complexes.**

Pourquoi un cours de visualisation ?

- **Données** de plus en plus **complexes**
- **Modèles** de plus en plus **complexes**
- **Interprétations** des résultats de plus en plus **complexes**.
- Besoin de visualiser pour :
 - **décrire** les données
 - **calibrer** les modèles
 - **présenter** les résultats de l'étude.

Conséquence

- La visualisation se révèle **cruciale** tout au long d'une étude statistique.
- De plus en plus de packages R sont dédiés à la **visualisation**.

Plan

- (au moins) 2 façons d'appréhender la **visualisation** :
 1. **Méthodes/modèles statistiques** : PCA, LDA, arbres...
 2. **Outils informatique** : packages R.

Plan

- (au moins) 2 façons d'appréhender la **visualisation** :
 1. **Méthodes/modèles statistiques** : PCA, LDA, arbres...
 2. **Outils informatique** : packages R.
- Dans ce cours, on va présenter quelques **outils R** :
 1. **ggplot2** : un package R pour visualiser les données \Rightarrow 3-4h.
 2. **Cartes** avec **ggmap**, **sf** et **leaflet** \Rightarrow 3-4h.
 3. **Visualisation dynamique/intéractive**
 - données avec **rAmCharts** et **Plotly** \Rightarrow 1h.
 - tableaux de bord avec **flexdashboard** \Rightarrow 1h.
 - application web avec **shiny** \Rightarrow 3-4h.

Compméments

Workshop shiny en février.

Boxplot sur les iris

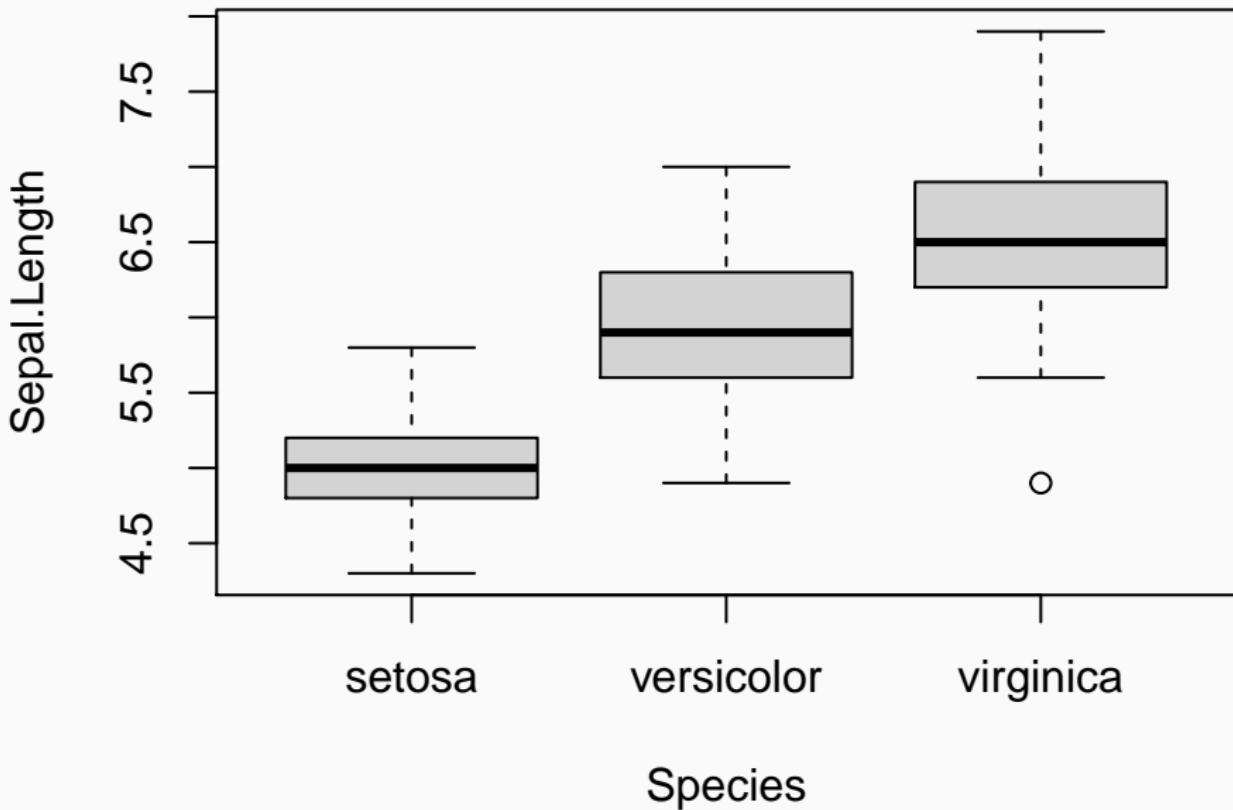
```
> data(iris)
> summary(iris)

  Sepal.Length      Sepal.Width       Petal.Length      Petal.Width      
Min.    :4.300      Min.    :2.000      Min.    :1.000      Min.    :0.100  
1st Qu.:5.100      1st Qu.:2.800      1st Qu.:1.600      1st Qu.:0.300  
Median :5.800      Median :3.000      Median :4.350      Median :1.300  
Mean   :5.843      Mean   :3.057      Mean   :3.758      Mean   :1.199  
3rd Qu.:6.400      3rd Qu.:3.300      3rd Qu.:5.100      3rd Qu.:1.800  
Max.   :7.900      Max.   :4.400      Max.   :6.900      Max.   :2.500  

  Species        
setosa    :50  
versicolor:50 
virginica:50
```

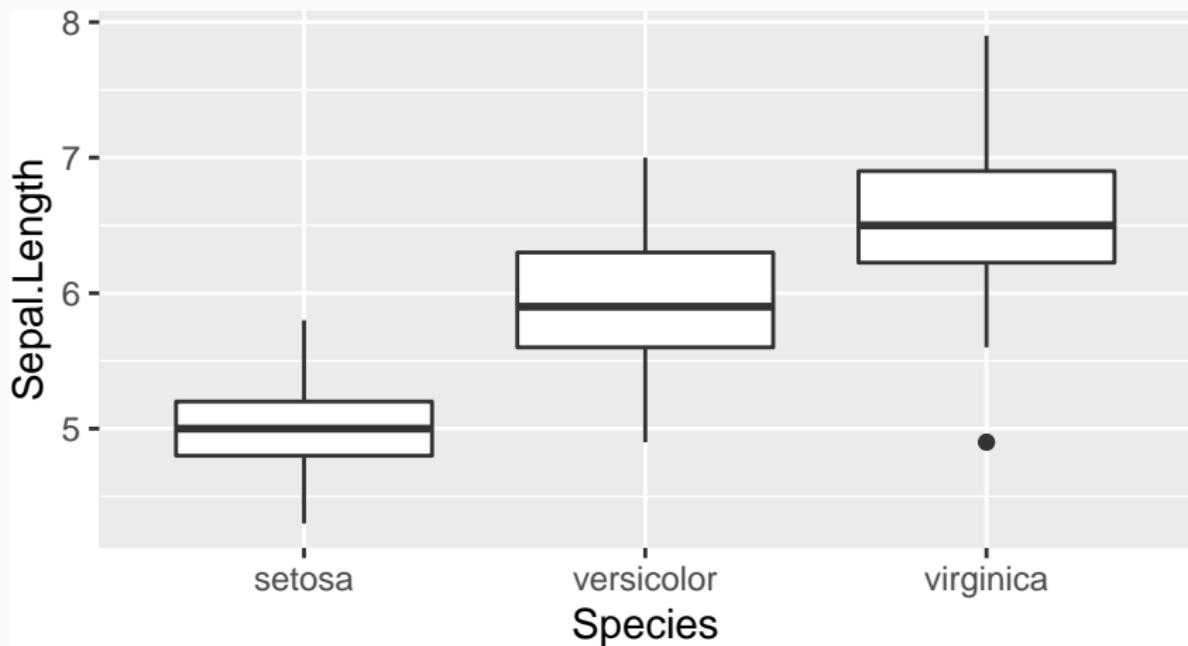
Fonctions conventionnelles

```
> boxplot(Sepal.Length~Species,data=iris)
```

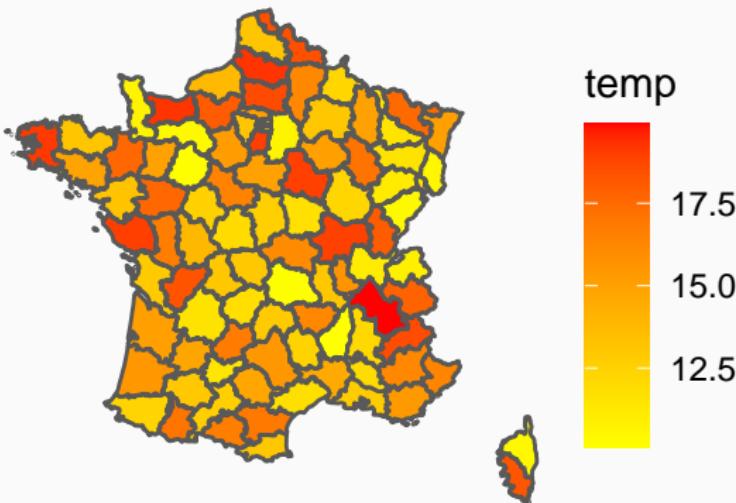


Grammaire ggplot

```
> library(tidyverse) #ggplot2 in tidyverse  
> ggplot(iris)+aes(x=Species,y=Sepal.Length)+geom_boxplot()
```



Une carte des températures



Diverses informations

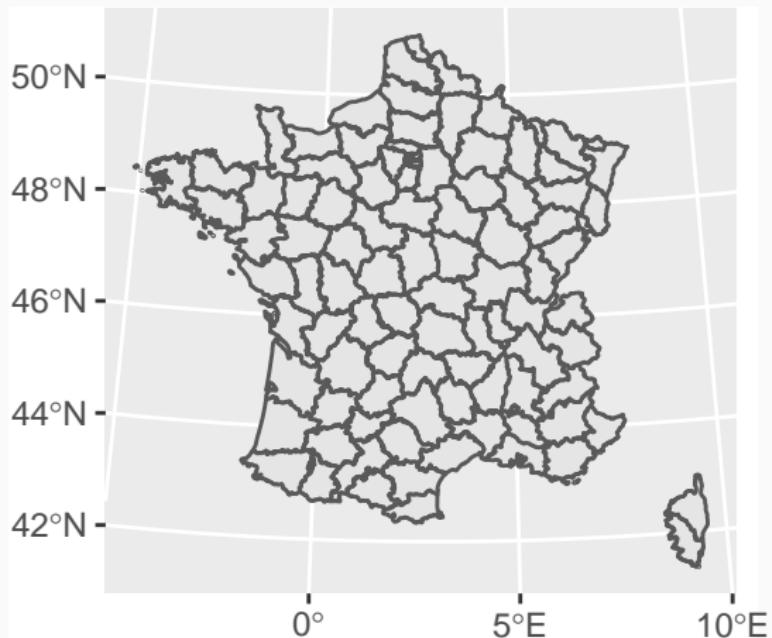
- Fond de cartes avec les frontières des départements ;
- Températures observées dans les départements (site web de météo france).

Carte shapefile

```
> library(sf)
> dpt <- read_sf("./DATA/dpt")
> dpt %>% select(NOM_DEPT,geometry) %>% head()
Simple feature collection with 6 features and 1 field
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 644570  ymin: 6272482  xmax: 1077507  ymax: 6997000
projected CRS: RGF93_Lambert_93
# A tibble: 6 x 2
  NOM_DEPT                                geometry
  <chr>                                     <MULTIPOLYGON [m]>
1 AIN           (((919195 6541470, 918932 6541203, 918628 6540523~
2 AISNE         (((735603 6861428, 735234 6861392, 734504 6861270~
3 ALLIER        (((753769 6537043, 753554 6537318, 752879 6538099~
4 ALPES-DE-HAUTE-P~ (((992638 6305621, 992263 6305688, 991610 6306540~
5 HAUTES-ALPES   (((1012913 6402904, 1012577 6402759, 1010853 6402~
6 ALPES-MARITIMES (((1018256 6272482, 1017888 6272559, 1016779 6272~
```

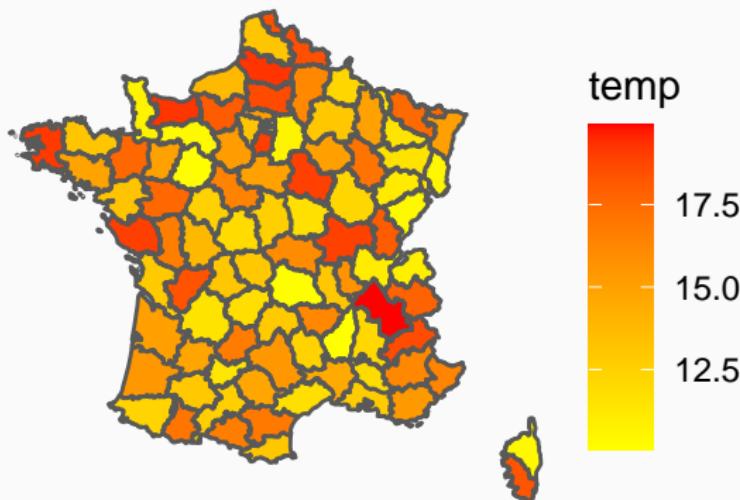
Fond de carte

```
> ggplot(dpt)+geom_sf()
```



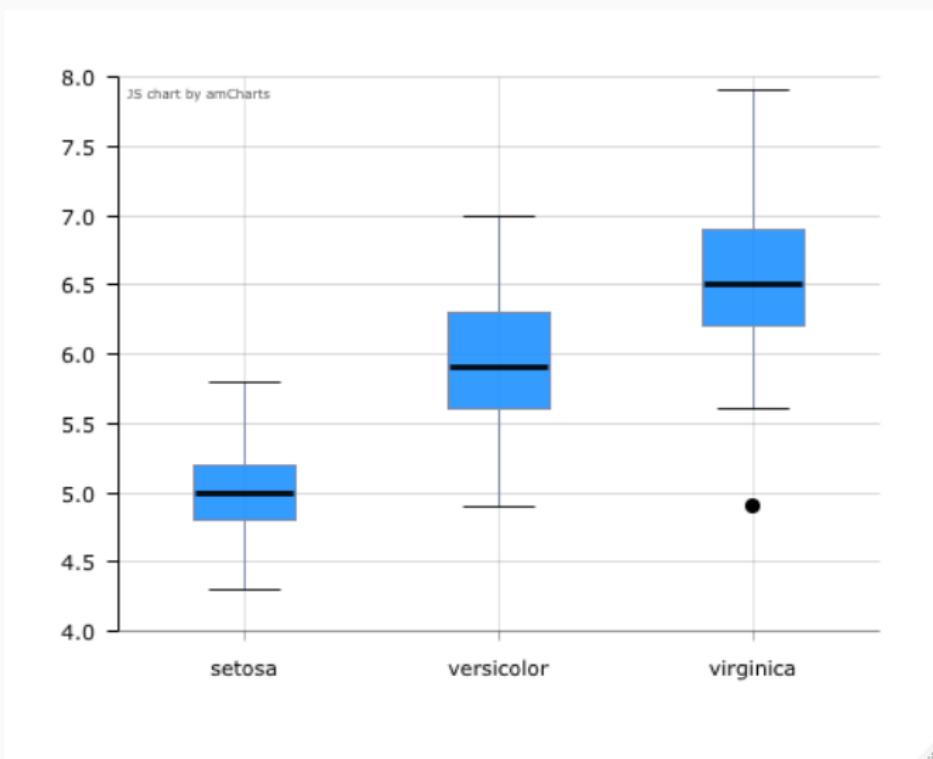
Ajout des températures

```
> ggplot(dpt) + geom_sf(aes(fill=temp)) +
+   scale_fill_continuous(low="yellow",high="red")+
+   theme_void()
```



Graphes interactifs avec rAmCharts

```
> library(rAmCharts)
> amBoxplot(Sepal.Length~Species,data=iris)
```



Tableaux de bord

- utile pour **publier** des synthèses d'**outils de visualisation** (données, graphes, modèles simples...)
- Package **flexdashboard**:
<https://rmarkdown.rstudio.com/flexdashboard/index.html>

Tableaux de bord

- utile pour publier des synthèses d'outils de visualisation (données, graphes, modèles simples...)
- Package flexdashboard:
<https://rmarkdown.rstudio.com/flexdashboard/index.html>
- Basé sur la syntaxe Rmarkdown
- Exemple: <https://lrouviere.shinyapps.io/dashboard/>

Applications web avec shiny

- Shiny est un package R qui permet de construire des applications web avec R (uniquement).
- Exemples:
 - overfitting en machine learning:
https://lrouviere.shinyapps.io/overfitting_app/
 - stations velib à Rennes: <https://lrouviere.shinyapps.io/velib/>

En résumé

- 12 (+5) heures pour 3 ou 4 thèmes.
- 1 thème = quelques slides + tutoriel (compléments à lire + exercices).
- Nécessite un **investissement personnel** ⇒ les heures en séance ne sont pas suffisantes pour tout faire !

Plan

1. Visualisation avec ggplot2

Graphiques R conventionnels (rappel)

La grammaire ggplot2

2. Cartes

ggmap

Contours shapefile contours avec sf

Cartes interactives avec leaflet

3. Quelques outils de visualisation dynamiques

rAmCharts et plotly

Graphes avec visNetwork

Tableau de bord avec flexdasboard

4. Évaluation

Visualisation avec ggplot2

- Visualiser des données à l'aide de **graphes** est souvent le **point de départ** d'un projet science des données.
- On peut bien entendu **créer différents types de graphes** avec **R**.
- On commence par un (court) rappel des **fonctions conventionnelles**.
- suivi d'une présentation de la **construction de graphes** avec le package **ggplot2**.

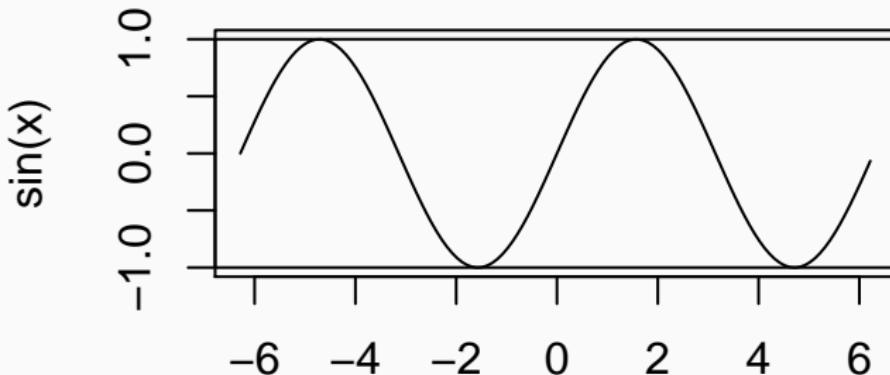
Visualisation avec ggplot2

Graphiques R conventionnels (rappel)

La fonction plot

- Fonction **générique** pour représenter (presque) **tous les types de données**.
- Pour un **nuage de points**, il suffit de renseigner un **vecteur** pour l'axe des **x**, et un autre vecteur pour celui des **y**.

```
> x <- seq(-2*pi,2*pi,by=0.1)
> plot(x,sin(x),type="l",xlab="x",ylab="sin(x)")
> abline(h=c(-1,1))
```



Graphes classiques pour visualiser des variables

- Histogramme pour une variable **continue**, diagramme en barres pour une variable **qualitative**.
- Nuage de points pour 2 variables continues.
- Boxplot pour une distribution continue.

Graphes classiques pour visualiser des variables

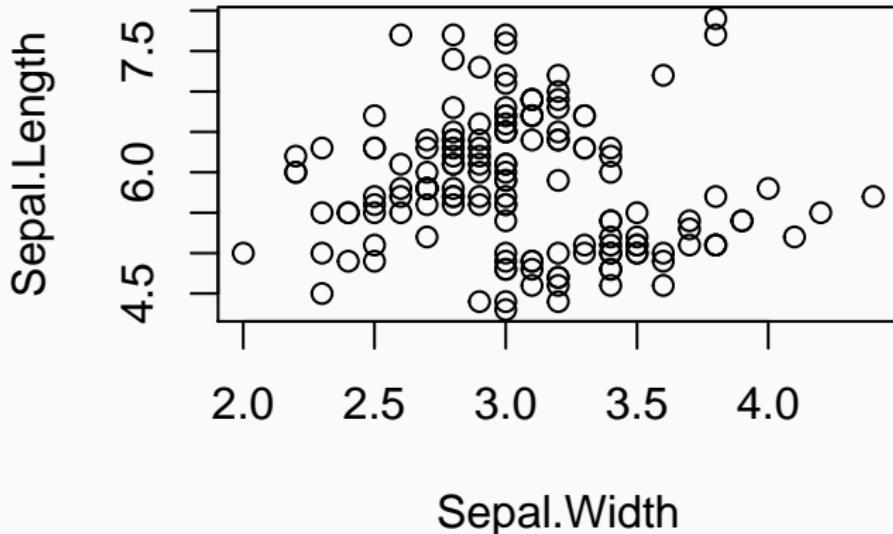
- Histogramme pour une variable **continue**, **diagramme en barres** pour une variable **qualitative**.
- **NUAGE DE POINTS** pour 2 variables continues.
- **Boxplot** pour une distribution continue.

Constat (positif)

Il existe une **fonction R** pour toutes les représentations.

Nuage de points sur un jeu de données

```
> plot(Sepal.Length~Sepal.Width,data=iris)
```



```
> #same as  
> plot(iris$Sepal.Width,iris$Sepal.Length)
```

Histogramme (variable continue)

```
> hist(iris$Sepal.Length, probability=TRUE,  
+       col="red", xlab="Sepal.Length", main="Histogram")
```

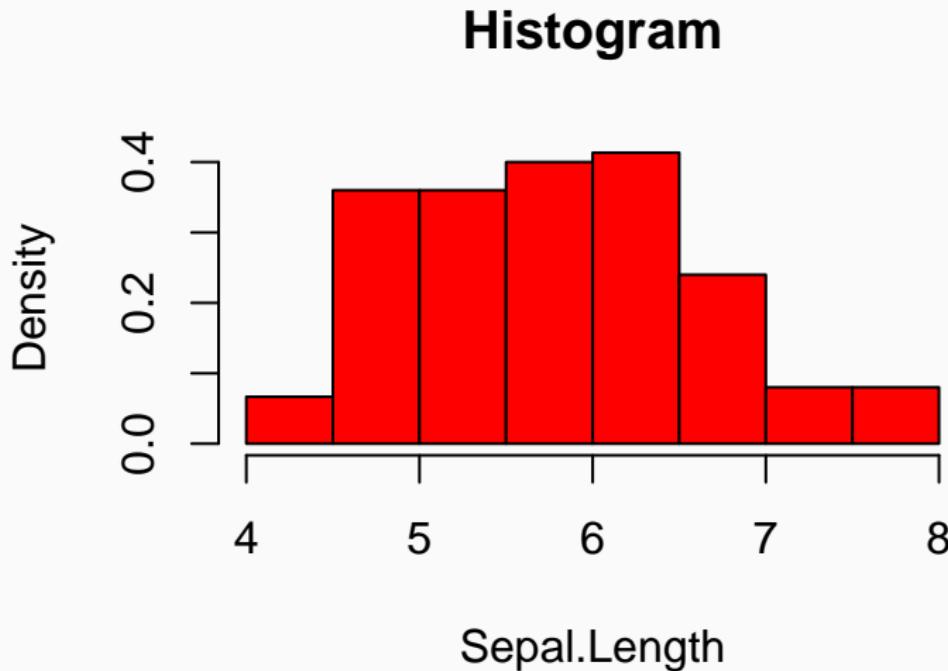
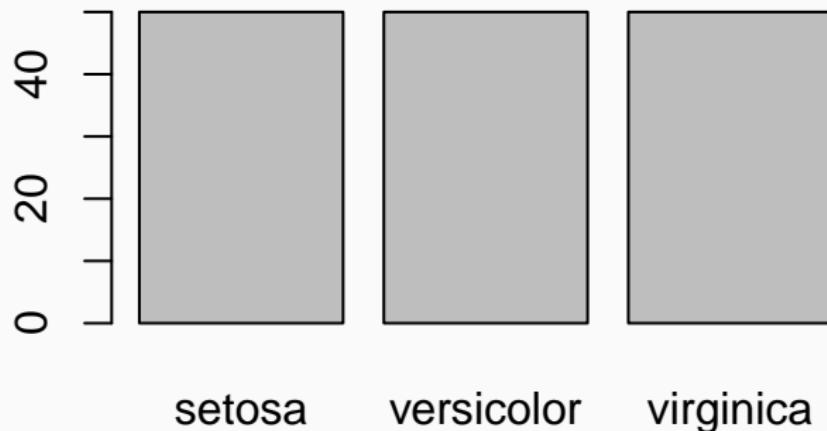


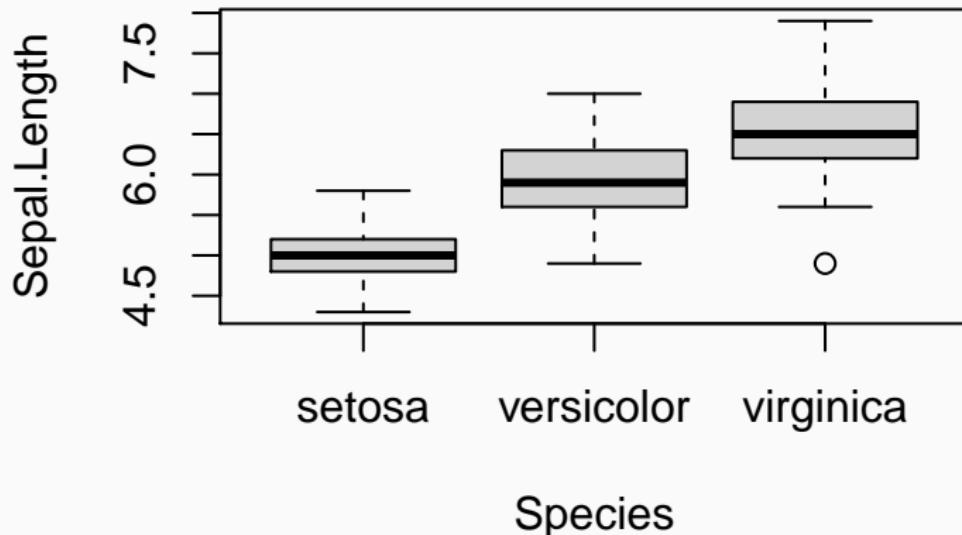
Diagramme en barres (variable qualitative)

```
> barplot(table(iris$Species))
```



Boxplot (distribution)

```
> boxplot(Sepal.Length~Species,data=iris)
```



Visualisation avec ggplot2

La grammaire ggplot2

- `ggplot2` permet de faire des graphes R en s'appuyant sur une **grammaire des graphiques** (équivalent de `dplyr` pour manipuler les données).
- L'objectif est d'utiliser une **syntaxe claire** pour construire des graphes : **graphes "complexes"** avec une **syntaxe courte et lisible**.
- Les graphes produits sont **agréables à regarder** (pas toujours le cas avec les graphes conventionnels).
- Documents : [tutoriel](#), [livre](#)

Pour un tableau de données fixé, un graphe est défini comme une succession de **couches**. Il faut toujours spécifier :

- les **données**
- les **variables** à représenter
- le **type de représentation** (nuage de points, boxplot...).

Pour un tableau de données fixé, un graphe est défini comme une succession de **couches**. Il faut toujours spécifier :

- les **données**
- les **variables** à représenter
- le **type de représentation** (nuage de points, boxplot...).

Les graphes **ggplot** sont construits à partir de ces couches. On indique

- les **données** avec **ggplot**
- les **variables** avec **aes** (aesthetics)
- le **type de représentation** avec **geom_**

La grammaire

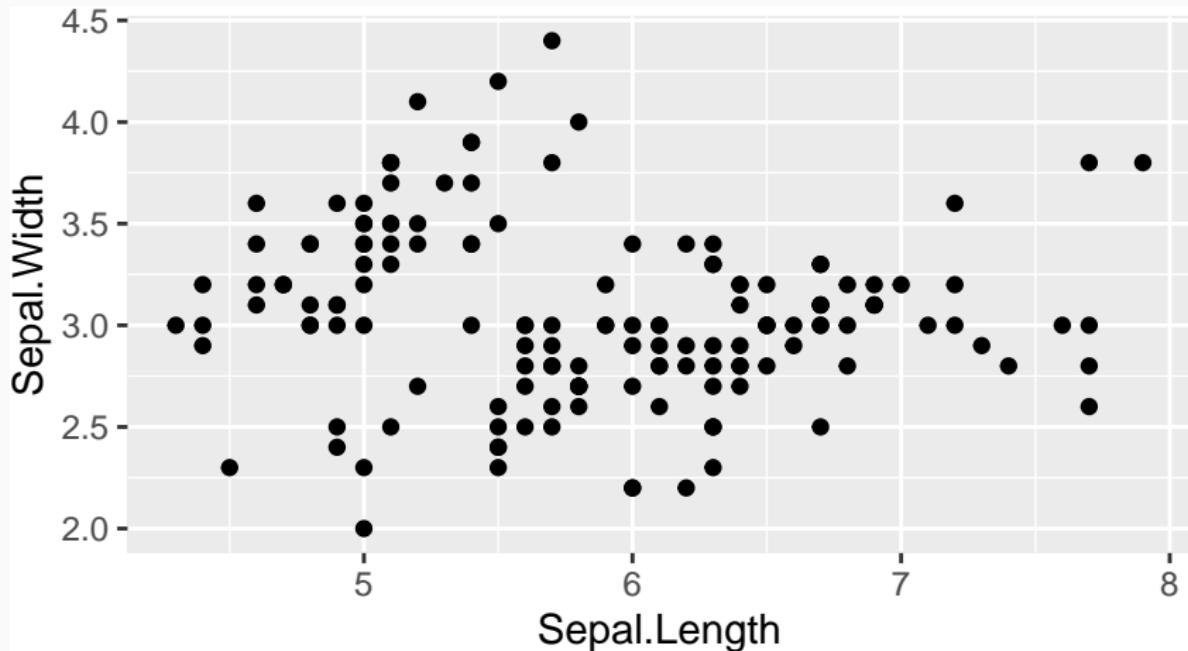
Les principaux **verbes** sont

- **Data (ggplot)** : les **données**, un dataframe ou un tibble.
- **Aesthetics (aes)** : façon dont les **variables** doivent être représentées.
- **Geometrics (geom_...)** : **type** de représentation.
- **Statistics (stat_...)** : spécifier les **transformations** des données.
- **Scales (scale_...)** : modifier certains **paramètres du graphe** (changer de couleurs, de taille...).

Tous ces éléments sont **séparés par un +**.

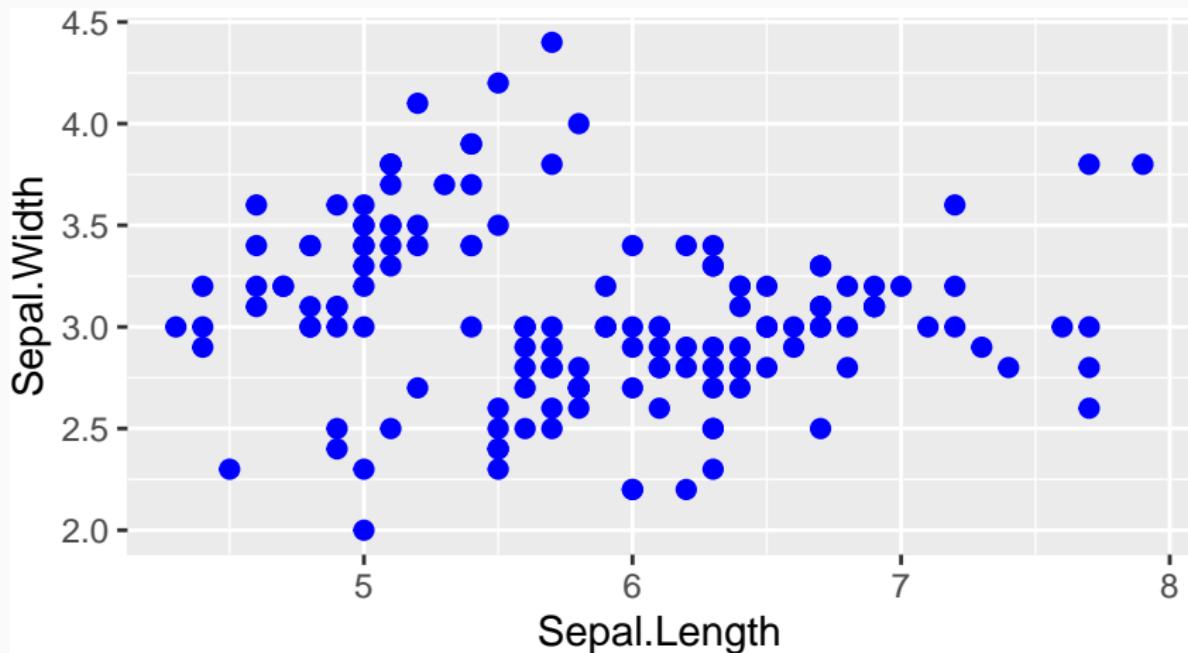
Un exemple

```
> ggplot(iris)+aes(x=Sepal.Length,y=Sepal.Width)+geom_point()
```



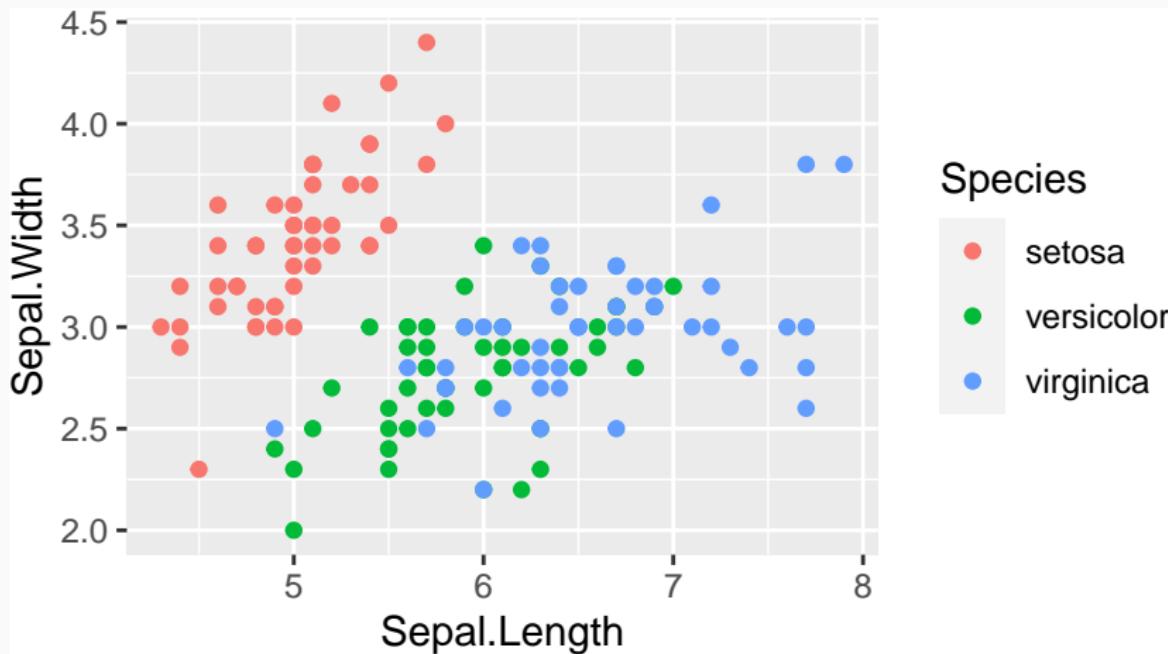
Couleur et taille

```
> ggplot(iris)+aes(x=Sepal.Length,y=Sepal.Width)+  
+   geom_point(color="blue",size=2)
```



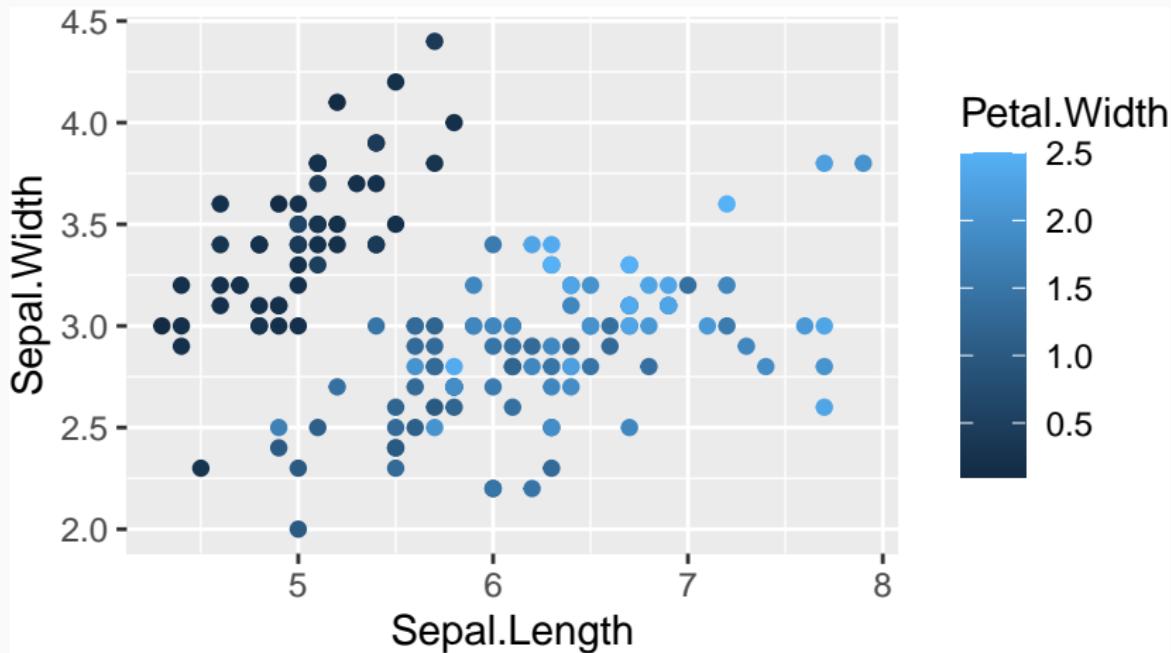
Couleur avec une variable qualitative

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



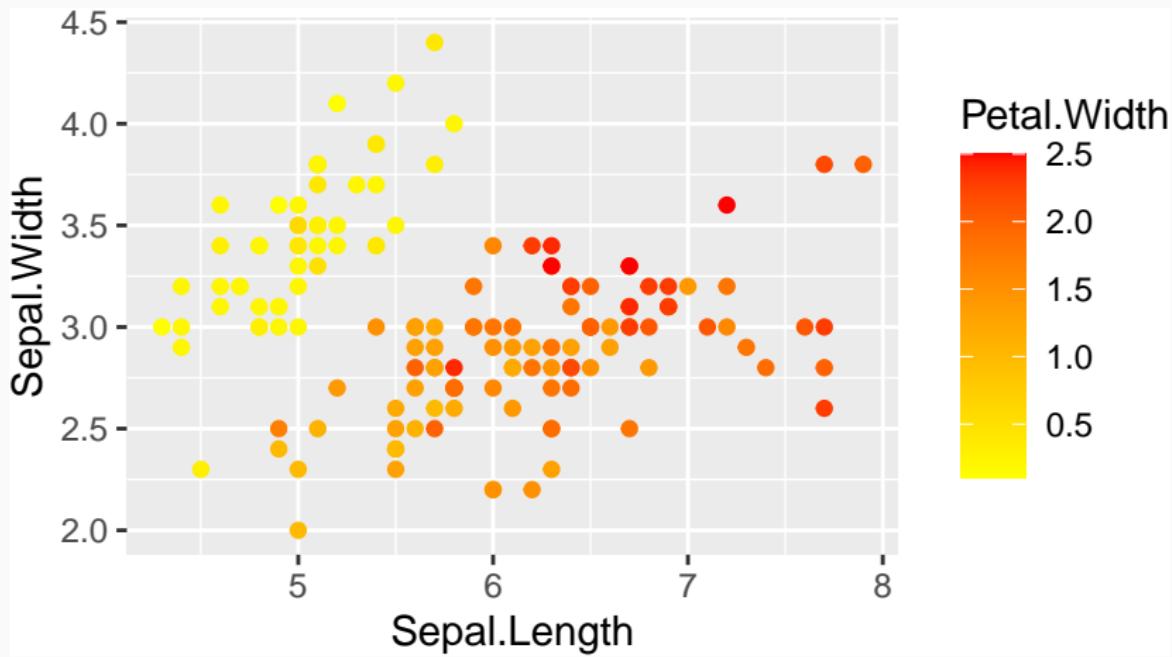
Couleur avec une variable continue

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



Changer la couleur

```
> ggplot(iris)+aes(x=Sepal.Length,y=Sepal.Width,  
+                     color=Petal.Width)+geom_point()+  
+                     scale_color_continuous(low="yellow",high="red")
```



Histogramme

```
> ggplot(iris)+aes(x=Sepal.Length)+geom_histogram(fill="red")
```

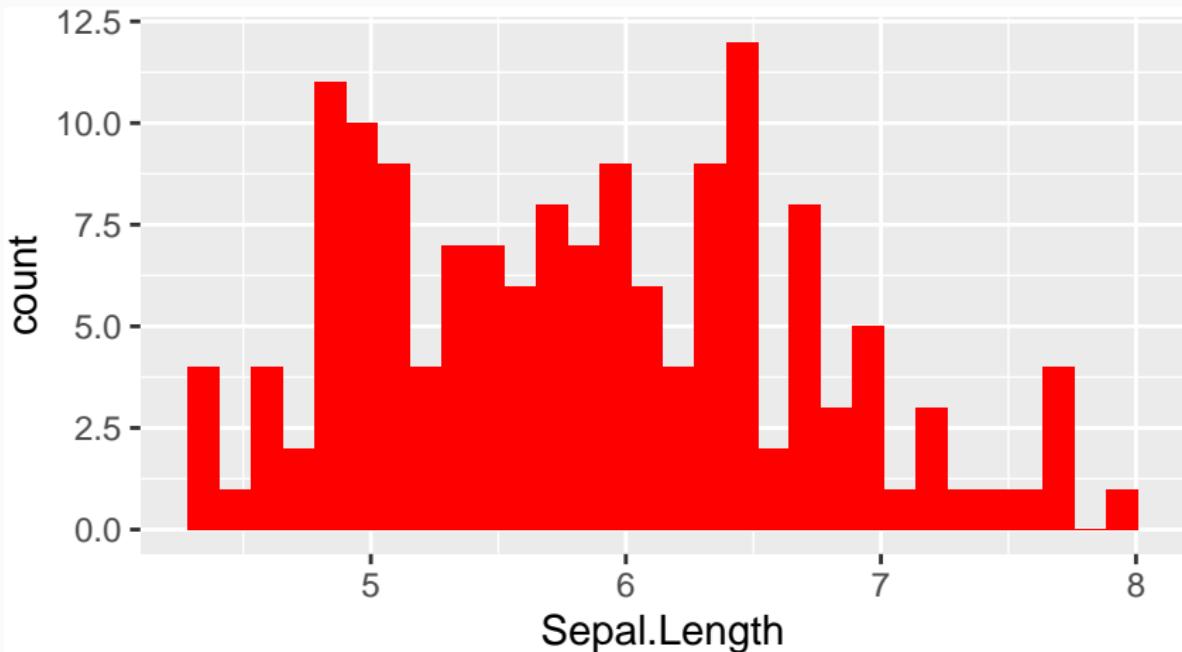
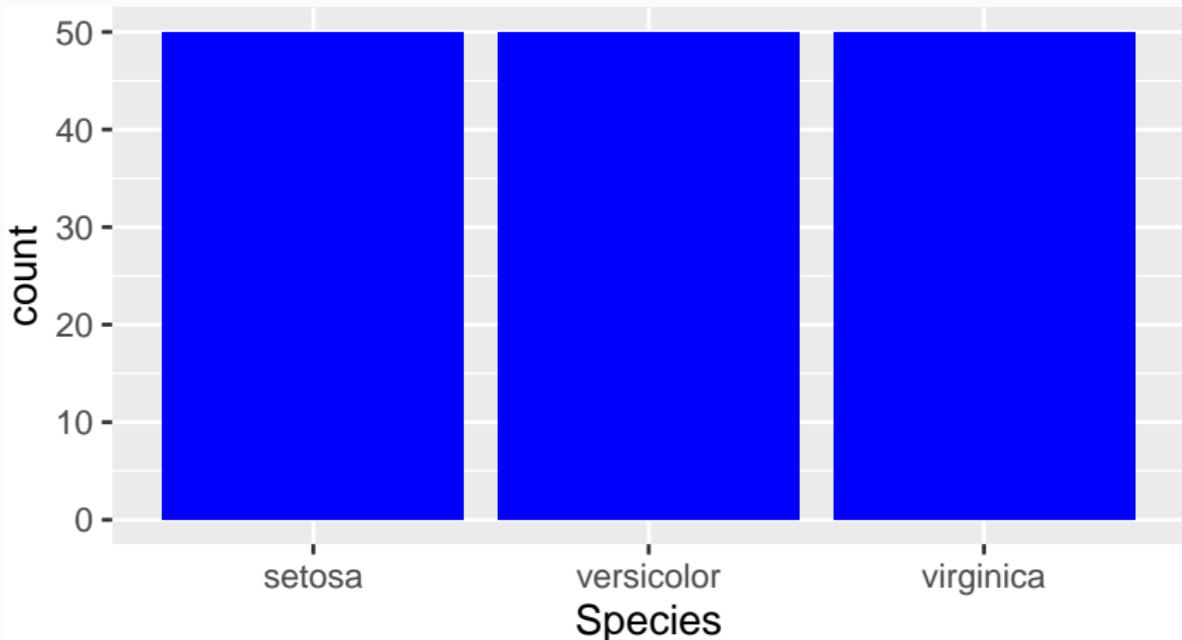


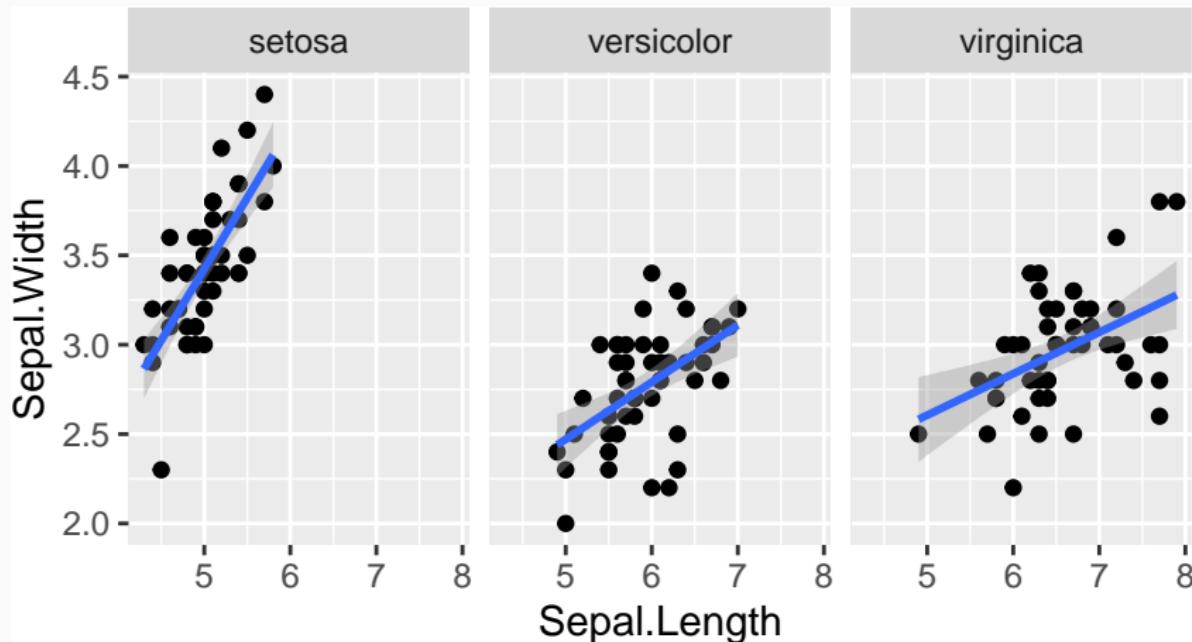
Diagramme en barres

```
> ggplot(iris)+aes(x=Species)+geom_bar(fill="blue")
```



Facetting (plus “compliqué”)

```
> ggplot(iris)+aes(x=Sepal.Length,y=Sepal.Width)+geom_point()+
+   geom_smooth(method="lm")+facet_wrap(~Species)
```

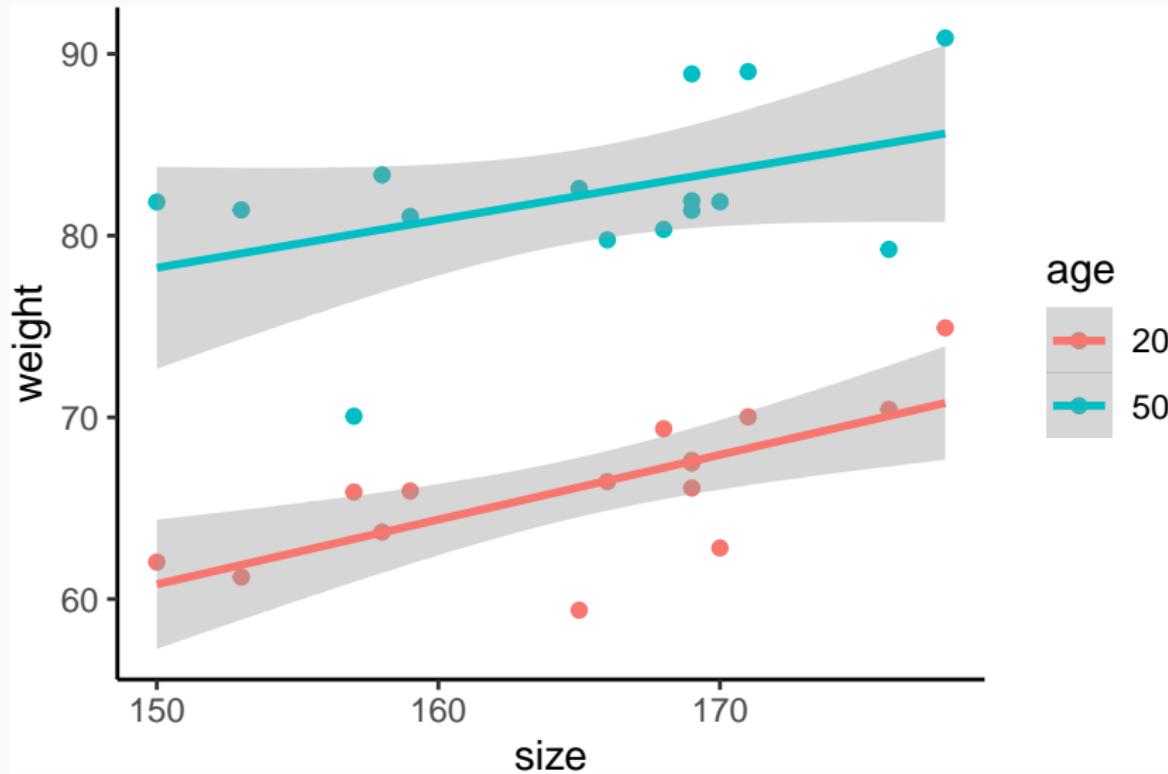


Combiner ggplot et dplyr

- Souvent important de construire un bon jeu de données pour obtenir un bon graphe.
- Par exemple

```
> head(df)
# A tibble: 6 x 3
  size weight.20 weight.50
  <dbl>     <dbl>     <dbl>
1   153      61.2     81.4
2   169      67.5     81.4
3   168      69.4     80.3
4   169      66.1     81.9
5   176      70.4     79.2
6   169      67.6     88.9
```

Objectif



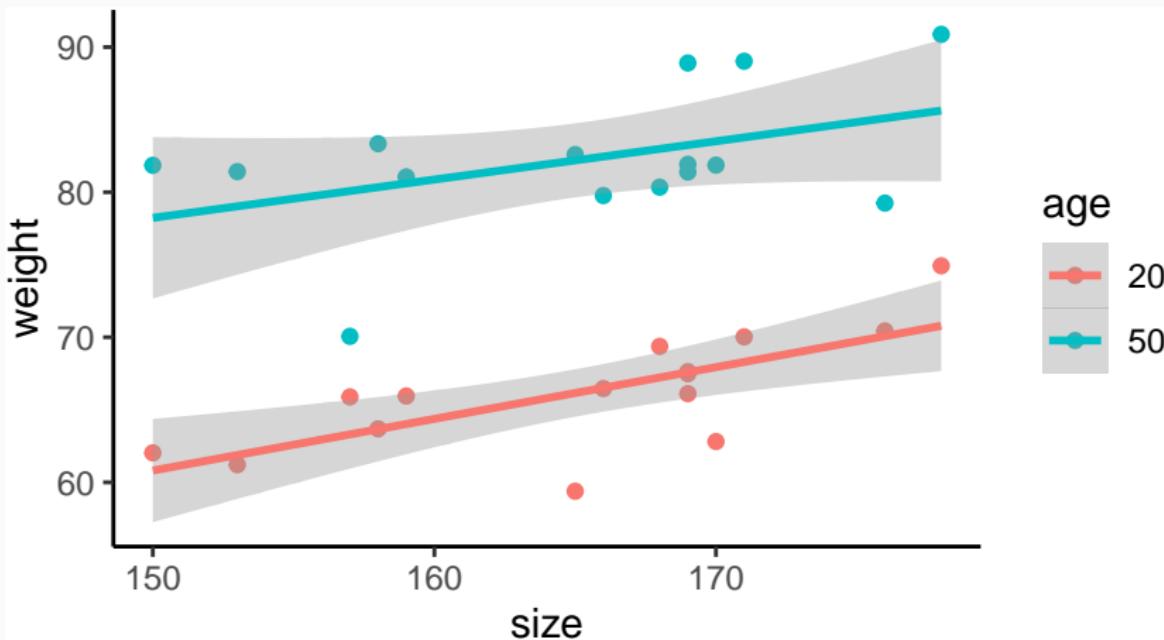
Etape dplyr

Assembler les colonnes `weight.M` et `weight.W` en une colonne `weight` avec `pivot_longer`:

```
> df1 <- df %>% pivot_longer(-size, names_to="age", values_to="weight")
> df1 %>% head()
# A tibble: 6 x 3
  size age      weight
  <dbl> <chr>    <dbl>
1 153  weight.20  61.2 
2 153  weight.50  81.4 
3 169  weight.20  67.5 
4 169  weight.50  81.4 
5 168  weight.20  69.4 
6 168  weight.50  80.3 
> df1 <- df1 %>%
+   mutate(age=recode(age,"weight.20"="20","weight.50"="50"))
```

Etape ggplot

```
> ggplot(df1)+aes(x=size,y=weight,color=age)+  
+   geom_point() +geom_smooth(method="lm") +theme_classic()
```



Compléments : quelques démos

```
> demo(image)
> example(contour)
> demo(persp)
> library("lattice");demo(lattice)
> example(wireframe)
> library("rgl");demo(rgl)
> example(persp3d)
> demo(plotmath);demo(Hershey)
```

Cartes

Introduction

- De nombreuses applications nécessitent des **cartes** pour **visualiser** des **données** ou les résultats d'un **modèle**.
- De **nombreux packages R** : ggmap, RgoogleMaps, maps...
- Dans cette partie : **ggmap**, **sf** (cartes **statiques**) et **leaflet** (cartes **dynamiques**).

Cartes

ggmap

Syntaxe

- Proche de `ggplot`...

Syntaxe

- Proche de `ggplot`...
- Au lieu de

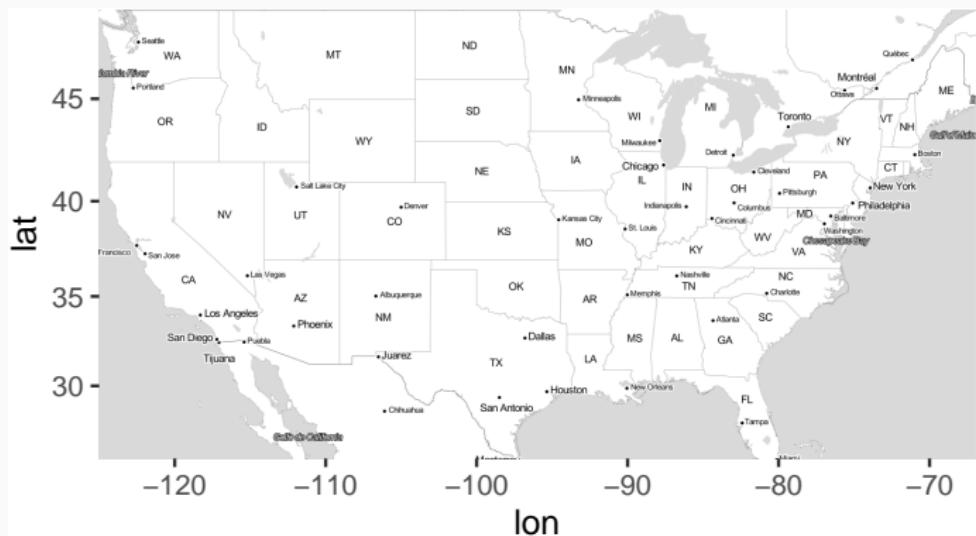
```
> ggplot(data)+...
```

- on utilise

```
> ggmap(backgroundmap)+...
```

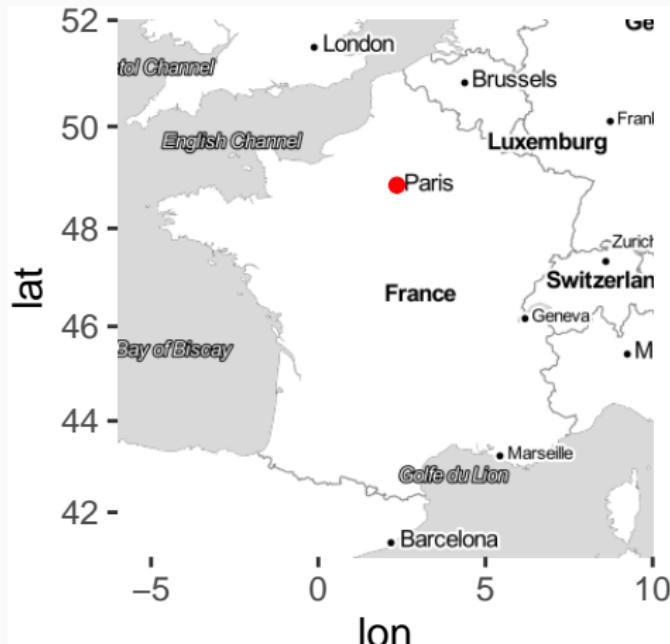
Fonds de carte ggmap

```
> library(ggmap)  
> us <- c(left = -125, bottom = 25.75, right = -67, top = 49)  
> map <- get_stamenmap(us, zoom = 5, maptype = "toner-lite")  
> ggmap(map)
```



Ajouts avec ggplot

```
> fr <- c(left = -6, bottom = 41, right = 10, top = 52)
> fond <- get_stamenmap(fr, zoom = 5,"toner-lite")
> Paris <- data.frame(lon=2.351499,lat=48.85661)
> ggmap(fond)+geom_point(data=Paris,aes(x=lon,y=lat),color="red")
```



Cartes

Contours shapefile contours avec sf

Le package sf

- **Ggmap** : bien pour des cartes “simples” (fond et quelques points).
- **Pas suffisant** pour des **représentations plus complexes** (colorier des pays à partir de variables).

Le package sf

- **Ggmap** : bien pour des cartes “simples” (fond et quelques points).
- **Pas suffisant** pour des **représentations plus complexes** (colorier des pays à partir de variables).
- **sf** permet de gérer des **objets spécifiques à la cartographie** : notamment les différents **systèmes de coordonnées** et **leurs projections en 2d** (latitudes-longitudes, World Geodesic System 84...)
- Fonds de carte au format **shapefile** (**contours = polygones**)
- Compatible avec **ggplot** (verbe **geom_sf**).

Références

- <https://statnmap.com/fr/2018-07-14-initiation-a-la-cartographie-avec-sf-et-compagnie/>
- **Vignettes** sur le cran :
<https://cran.r-project.org/web/packages/sf/index.html>.

Exemple

```
> library(sf)
> dpt <- read_sf("dpt")
> dpt[1:5,3]
Simple feature collection with 5 features and 1 field
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 644570 ymin: 6290136 xmax: 1022851 ymax: 6997000
projected CRS: RGF93_Lambert_93
# A tibble: 5 x 2
  NOM_DEPT                           geometry
  <chr>                               <MULTIPOLYGON [m]>
1 AIN        (((919195 6541470, 918932 6541203, 918628 6540523~
2 AISNE      (((735603 6861428, 735234 6861392, 734504 6861270~
3 ALLIER     (((753769 6537043, 753554 6537318, 752879 6538099~
4 ALPES-DE-HAUTE-P~ (((992638 6305621, 992263 6305688, 991610 6306540~
5 HAUTES-ALPES (((1012913 6402904, 1012577 6402759, 1010853 6402~
```

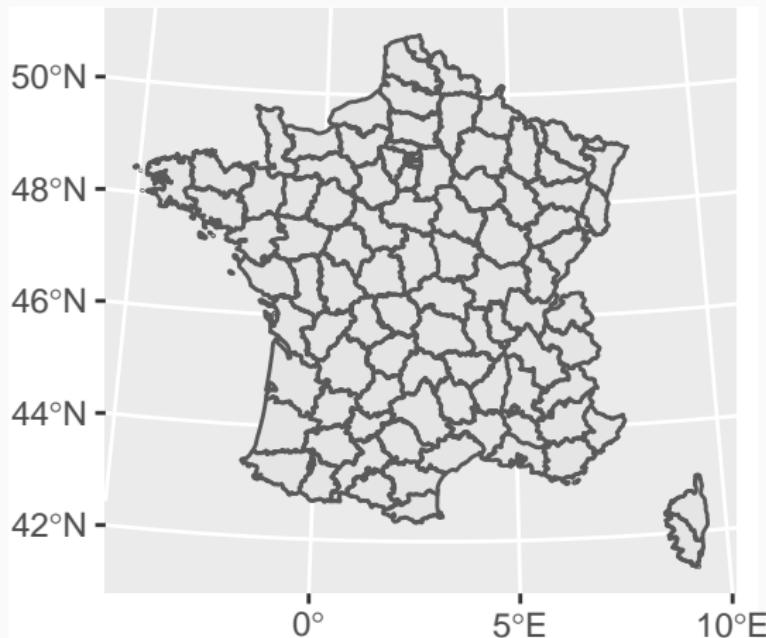
Visualisation avec plot

```
> plot(st_geometry(dpt))
```



Visualisation ggplot

```
> ggplot(dpt)+geom_sf()
```



Ajouter des points sur le graphe

- Définir des coordonnées avec `st_point`

```
> point <- st_sfc(st_point(c(2.351462,48.85670)),  
+                   st_point(c(4.832011,45.75781)),  
+                   st_point(c(5.369953,43.29617)))
```

Ajouter des points sur le graphe

- Définir des coordonnées avec `st_point`

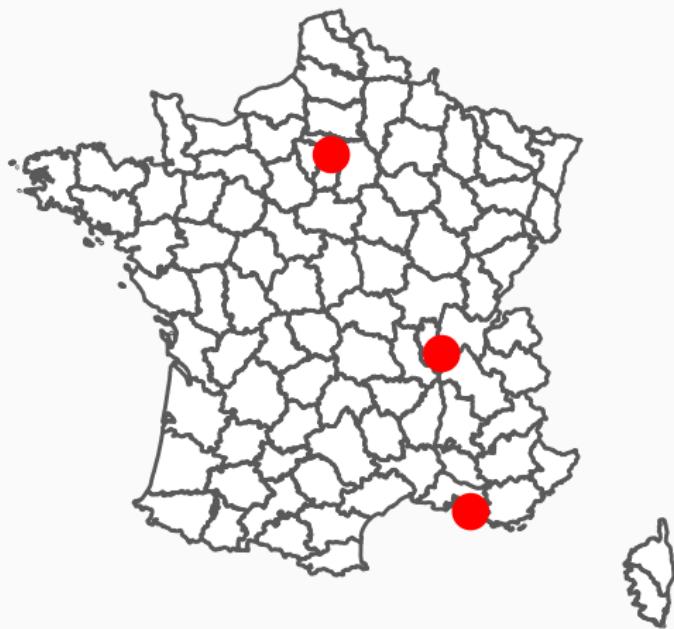
```
> point <- st_sfc(st_point(c(2.351462,48.85670)),  
+                   st_point(c(4.832011,45.75781)),  
+                   st_point(c(5.369953,43.29617)))
```

- Spécifier le `système de coordonnées` (4326 pour lat-lon)

```
> st_crs(point) <- 4326 #coord sont des long/lat  
> point  
Geometry set for 3 features  
geometry type:  POINT  
dimension:      XY  
bbox:           xmin: 2.351462 ymin: 43.29617 xmax: 5.369953 ymax: 48.85670  
geographic CRS: WGS 84
```

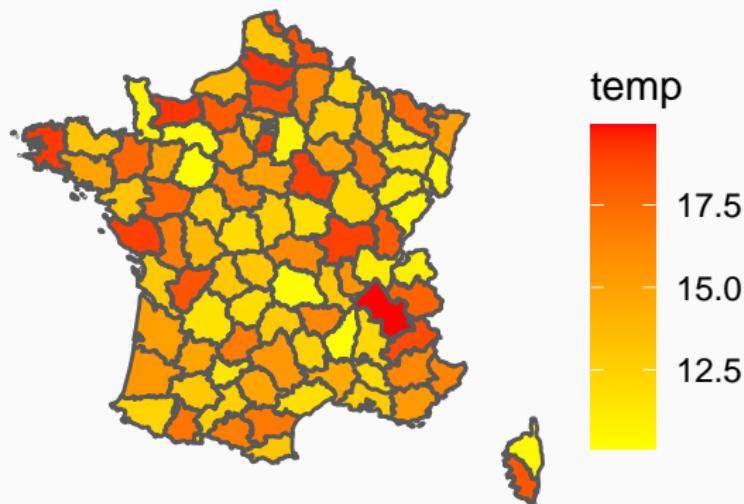
Étape ggplot

```
> ggplot(dpt) + geom_sf(fill="white")+
+   geom_sf(data=point,color="red",size=4)+theme_void()
```



Colorier des polygones

```
> set.seed(1234)
> dpt1 <- dpt %>% mutate(temp=runif(96,10,20))
> ggplot(dpt1) + geom_sf(aes(fill=temp)) +
+   scale_fill_continuous(low="yellow",high="red")+
+   theme_void()
```



Compléments : la classe geometry

- Une des forces de `sf` est la classe `geometry` qu'il propose.

Compléments : la classe geometry

- Une des forces de `sf` est la classe `geometry` qu'il propose.
- C'est cette classe qui conduit la représentation avec `plot` ou `geom_sf` :
 - `point` ou `multipoint` \Rightarrow points pour localiser un lieu
 - `polygon` ou `multipolygon` \Rightarrow contours pour représenter des frontières.

Compléments : la classe geometry

- Une des forces de **sf** est la classe **geometry** qu'il propose.
- C'est cette classe qui conduit la représentation avec **plot** ou **geom_sf** :
 - **point** ou **multipoint** \Rightarrow points pour localiser un lieu
 - **polygon** ou **multipolygon** \Rightarrow contours pour représenter des frontières.
- Quelques fonctions utiles :
 - **st_point** et **st_multipoint** : créer des points ou suite de points
 - **st_sfc** : donner une structure **geometry** à un objet (utile pour ensuite spécifier le système de coordonnées...)
 - **st_crs** : spécifier le système de coordonnées d'un geometry.
 - **st_cast** : transformer le type de geometry
 - ...

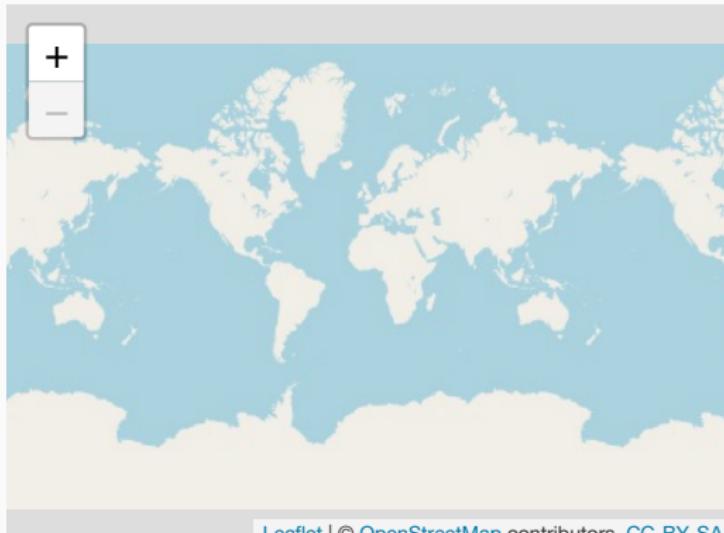
Cartes

Cartes interactives avec leaflet

Fonds de carte

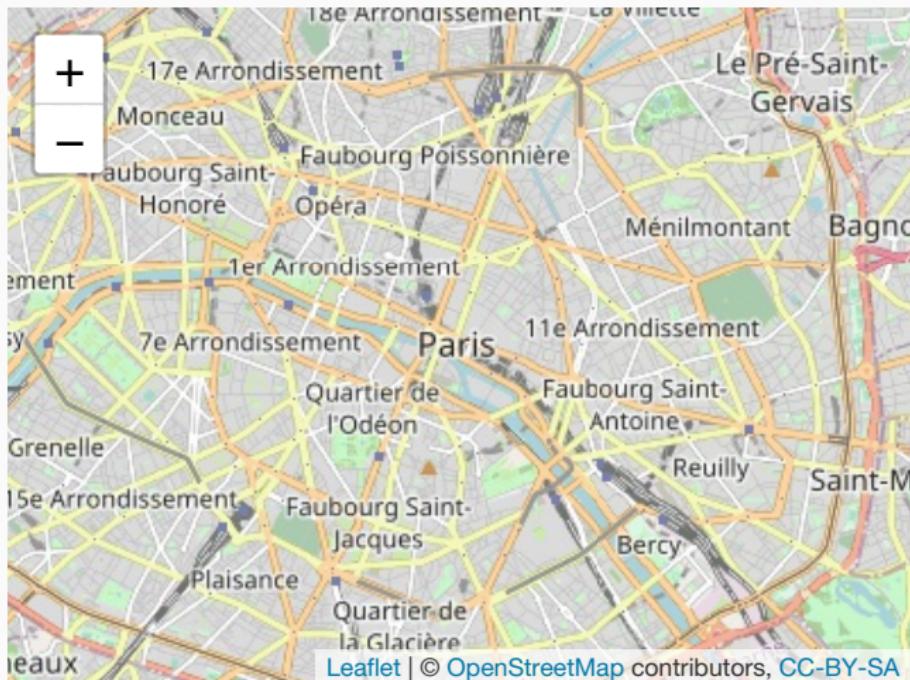
- Leaflet est une des librairies open-source JavaScript les plus populaires pour faire des cartes interactives.
- Documentation: [here](#)

```
> library(leaflet)  
> leaflet() %>% addTiles()
```



Différents styles de fonds de carte

```
> Paris <- c(2.35222, 48.856614)  
> leaflet() %>% addTiles() %>%  
+   setView(lng = Paris[1], lat = Paris[2], zoom=12)
```



```
> leaflet() %>% addProviderTiles("Stamen.Toner") %>%  
+   setView(lng = Paris[1], lat = Paris[2], zoom = 12)
```



Leaflet | Map tiles by Stamen Design, CC BY 3.0 — Map data © OpenStreetMap contributors

Avec des données

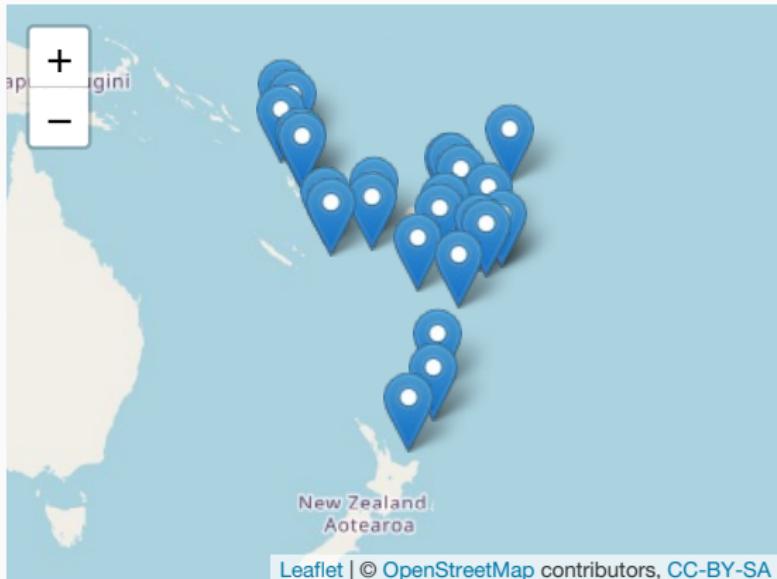
- Localiser 1000 séismes près des Fiji

```
> data(quakes)
> head(quakes)

  lat   long depth mag stations
1 -20.42 181.62    562 4.8      41
2 -20.62 181.03    650 4.2      15
3 -26.00 184.10     42 5.4      43
4 -17.97 181.66    626 4.1      19
5 -20.42 181.96    649 4.0      11
6 -19.68 184.31    195 4.0      12
```

Séismes avec une magnitude plus grande que 5.5

```
> quakes1 <- quakes %>% filter(mag>5.5)
> leaflet(data = quakes1) %>% addTiles() %>%
+   addMarkers(~long, ~lat, popup = ~as.character(mag))
```



Remarque

La magnitude apparaît lorsqu'on clique sur un marker.

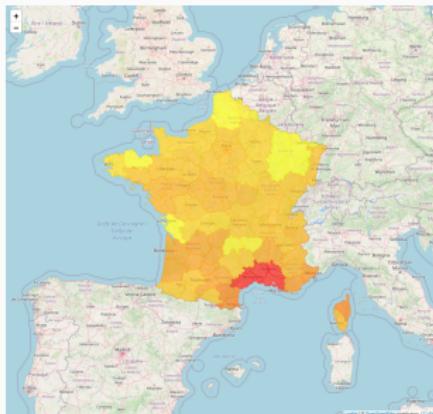
addCircleMarkers

```
> leaflet(data = quakes1) %>% addTiles() %>%  
+   addCircleMarkers(~long, ~lat, popup=~as.character(mag),  
+                     radius=3,fillOpacity = 0.8,color="red")
```



Colorier polygones en combinant leaflet et sf

```
> leaflet() %>% addTiles() %>%  
+   addPolygons(data = dpt2,color=~pal1(t_prev),fillOpacity = 0.6,  
+                 stroke = TRUE,weight=1,  
+                 popup=~paste(as.character(NOM_DEPT),  
+                               as.character(t_prev),sep=" : "))
```



Quelques outils de visualisation dynamiques

Des packages R

- Graphiques classiques avec `rAmCharts` et `plotly`.
- Graphes avec `visNetwork`.
- Tableaux de bord avec `flexdashboard`.

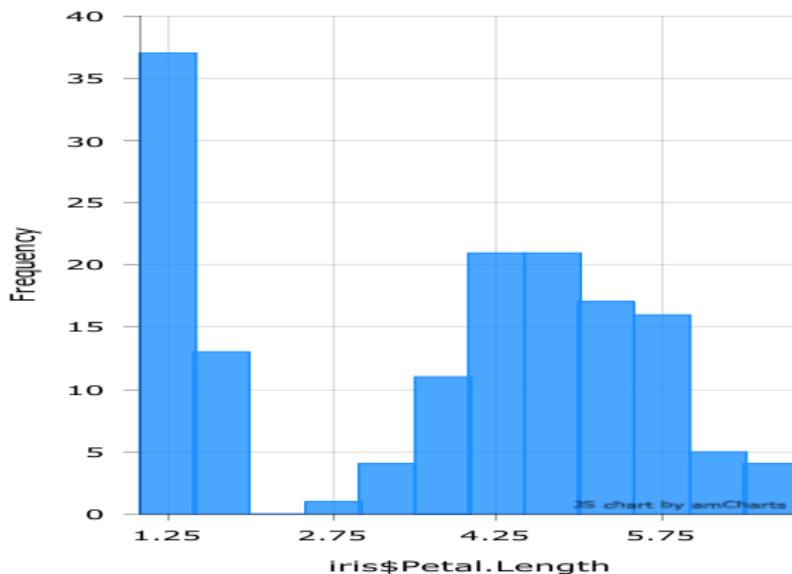
Quelques outils de visualisation dynamiques

rAmCharts et plotly

- **User-friendly** pour des graphes standards (nuages de points, séries chronologiques, histogrammes. . .).
- Il suffit d'utiliser la fonction **R** classique avec le préfixe **prefix am**.
- **Exemples** : **amPlot**, **amHist**, **amBoxplot**.
- **Références**: https://datastorm-open.github.io/introduction_ramcharts/

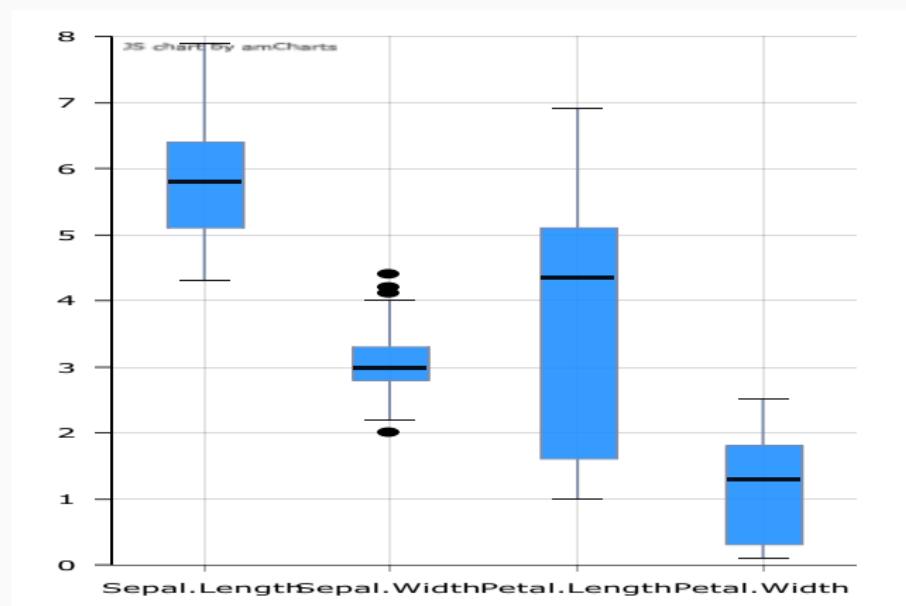
rAmCharts Histogramme

```
> library(rAmCharts)
> amHist(iris$Petal.Length)
```



rAmcharts Boxplot

```
> amBoxplot(iris)
```

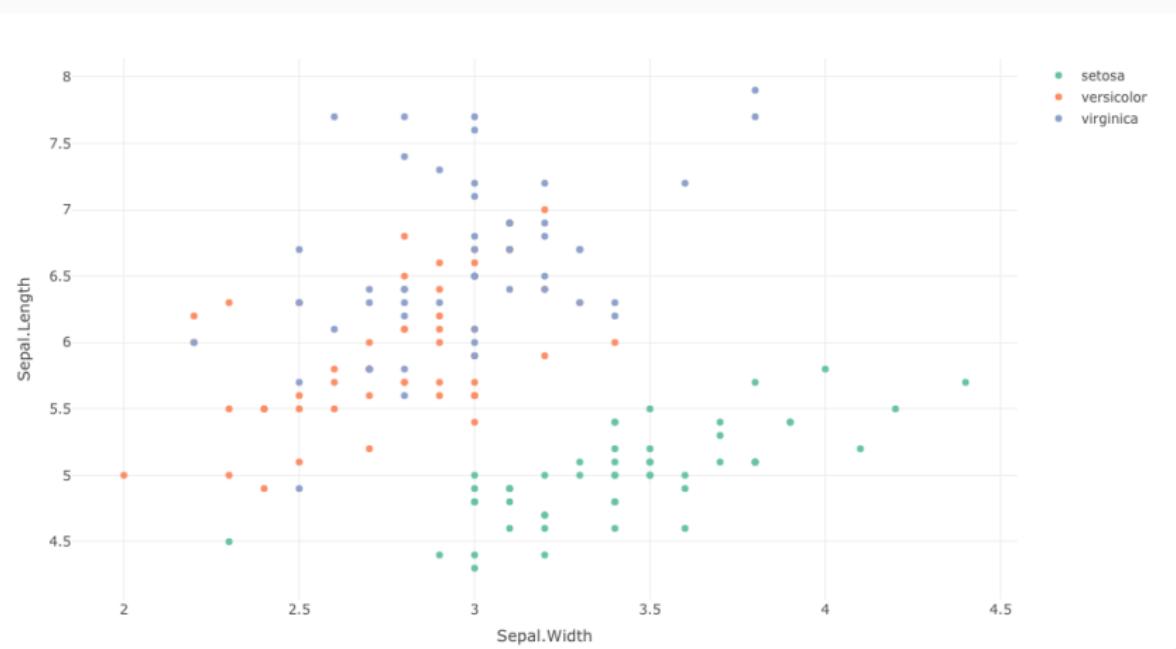


Plotly

- Package **R** pour créer des **graphes interactifs** à partir de la librairie open source **Javascript plotly.js**.
- La syntaxe se décompose en **3 parties** :
 - données et variables (**plot_ly**) ;
 - type de représentation (**add_trace**, **add_markers**...) ;
 - options (axes, titres...) (**layout**).
- Références: <https://plot.ly/r/reference/>

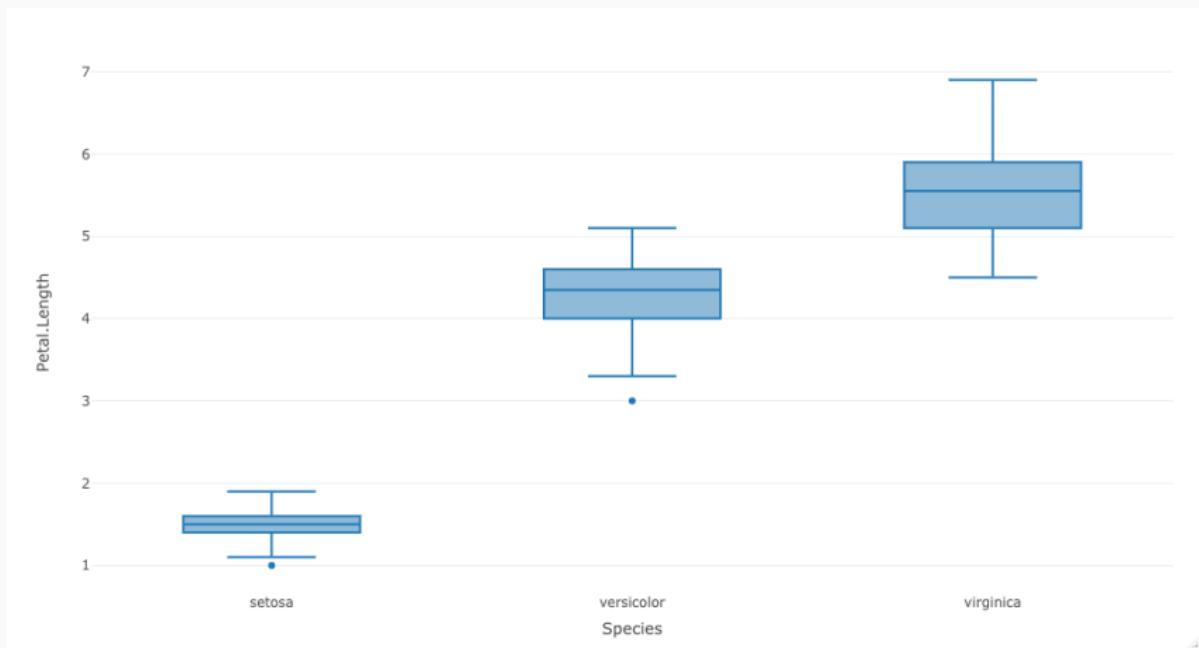
Nuage de points

```
> library(plotly)  
> iris %>% plot_ly(x=~Sepal.Width,y=~Sepal.Length,color=~Species) %>%  
+   add_markers(type="scatter")
```



Plotly boxplot

```
> iris %>% plot_ly(x=~Species,y=~Petal.Length) %>% add_boxplot()
```



Quelques outils de visualisation dynamiques

Graphes avec visNetwork

Connexions entre individus

- De nombreux jeux de données peuvent être visualisés avec des **graphes**, notamment lorsque l'on souhaite étudier des **connexions** entre individus (génétique, réseaux sociaux, système de recommandation...)

Connexions entre individus

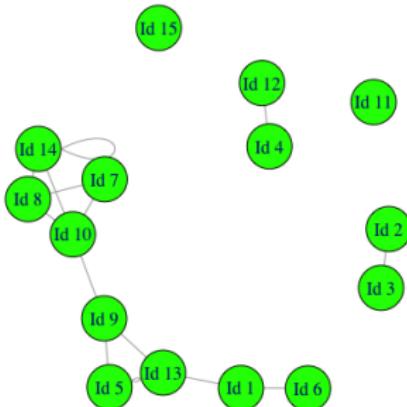
- De nombreux jeux de données peuvent être visualisés avec des **graphes**, notamment lorsque l'on souhaite étudier des **connexions** entre individus (génétique, réseaux sociaux, système de recommandation...)
- Un individu = **un nœud** et une connexion = **une arête**.

```
> set.seed(123)
> nodes <- data.frame(id = 1:15, label = paste("Id", 1:15))
> edges <- data.frame(from = trunc(runif(15)*(15-1))+1,
+                       to = trunc(runif(15)*(15-1))+1)
> head(edges)
   from to
1     5 13
2    12  4
3     6  1
4    13  5
5    14 14
6     1 13
```

Graphe statique : le package igraph

- Références: <http://igraph.org/r/>, <http://kateto.net/networks-r-igraph>

```
> library(igraph)
> net <- graph_from_data_frame(d=edges, vertices=nodes, directed=F)
> plot(net,vertex.color="green",vertex.size=25)
```

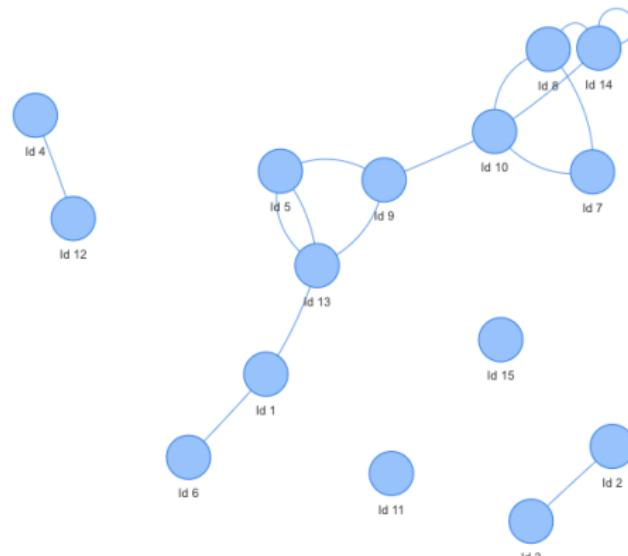


Graph dynamique : le package visNetwork

- Référence:

<https://datastorm-open.github.io/visNetwork/interaction.html>

```
> library(visNetwork)
> visNetwork(nodes, edges)
```



Quelques outils de visualisation dynamiques

Tableau de bord avec flexdasboard

- Juste un outil... mais **un outil important** en science des données
- Permet **d'assembler des messages importants** sur des données et/ou modèles

- Juste un outil... mais **un outil important** en science des données
- Permet **d'assembler des messages importants** sur des données et/ou modèles
- **Package:** flexdashboard
- **Syntaxe:** simple... juste du**Rmarkdown**
- **Référence:** <https://rmarkdown.rstudio.com/flexdashboard/>

Header

```
---
```

```
title: "My title"
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill
    theme: default
---
```

- Le thème par défaut peut être remplacé par d'**autres thèmes** (cosmo, bootstrap, cerulean...) (see [here](#)). Il suffit d'ajouter

```
theme: yeti
```

Flexdashboard | code

```
Descriptive statistics
```

```
=====
```

```
Column {data-width=650}
```

```
### Dataset
```

```
```{r}
```

```
DT::datatable(df, options = list(pageLength = 25))
```

```
```
```

```
Column {data-width=350}
```

```
### Correlation matrix
```

```
```{r}
```

```
cc <- cor(df[,1:11])
```

```
mat.cor <- corrplot::corrplot(cc)
```

```
```
```

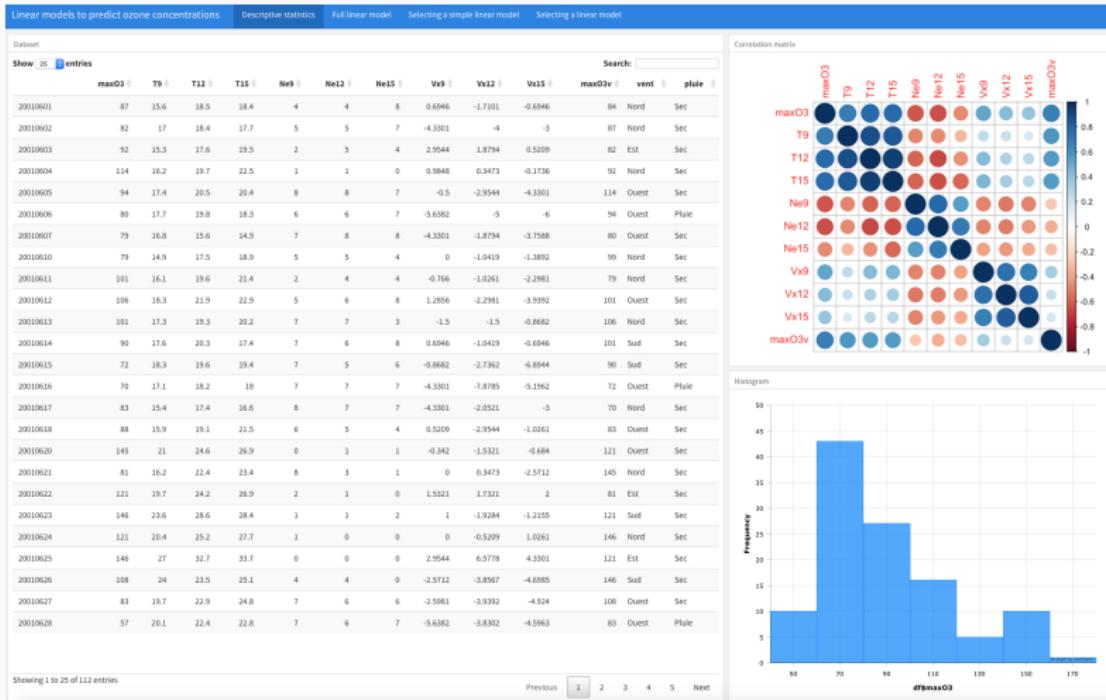
```
### Histogram
```

```
```{r}
```

```
amHist(df$max03)
```

```
```
```

Flexdashboard | dashboard



Évaluation

Projet visualisation

- Groupe (2 membres)
- Trouver un **problème de visualisation** (par exemple un jeu de données avec un problème statistique classification, regression...)
- Construire une application shiny pour répondre au problème
- Ne pas hésiter à utiliser les outils présentés dans le cours (possibilité de proposer de nouveaux outils)
- Déployer l'application sur le net avec **shinyapps**, voir
<https://docs.rstudio.com/shinyapps.io/index.html>
- Produire un document markdown (4 ou 5 pages) qui présente votre travail, utiliser une sortie pdf ou html.

Protocole

Vous devrez déposer dans cursus

- le document markdown (html ou pdf) qui présente votre travail ainsi que l'url de l'application
- les fichiers shiny (app.R ou ui.R et server.R, global, css, img...)
- le jeu de données (ou l'url où on peut le récupérer facilement)
- tous les fichiers nécessaires au lancement de l'application

avant le vendredi 12 mars, 18h.

Quelques exemples (promo précédente)

- <https://kabdallah.shinyapps.io/shinyapp/>
- https://quentincarric.shinyapps.io/FooTooL_by_CARRIC_DIALLO_MAHE/
- <https://pierrelepagnol.shinyapps.io/Reporting/>
- <https://hersantmarc.shinyapps.io/projet/>
- https://euvrardq.shinyapps.io/etude_tendances_youtube/