

Visualisation avec R

Laurent Rouvière

Janvier 2020

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**
- **Ressources**:
 - slides et notebook R disponibles à l'url <https://lrouviere.github.io/VISU/>

Complément

Workshop Shiny en février.

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.

Conséquence

La visualisation se révèle **cruciale** tout au long d'une étude statistique.

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.

Conséquence

La visualisation se révèle **cruciale** tout au long d'une étude statistique.

- Besoin de visualiser pour :
 - **décrire** les données
 - **calibrer** les modèles
 - **présenter** les résultats de l'étude.

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

Plan

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

Plan

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

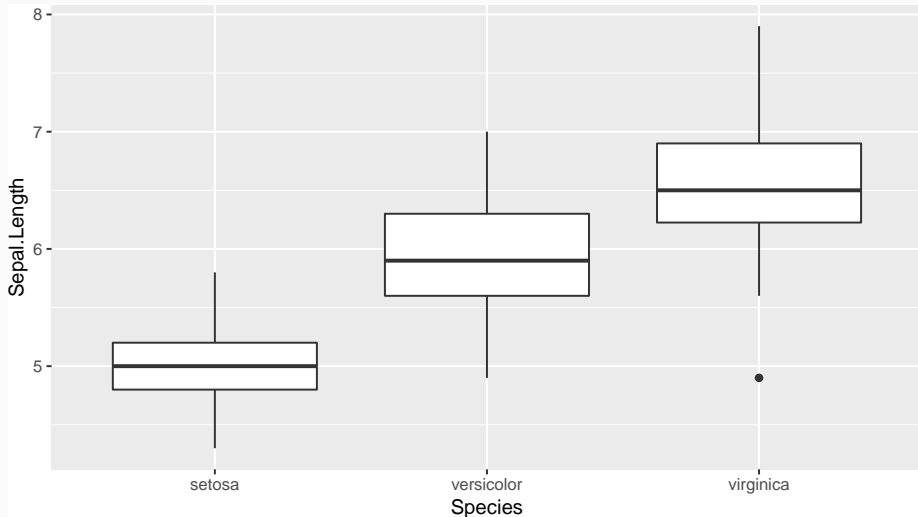
Remarque

De plus en plus de packages R sont dédiés à la **visualisation**.

Introduction

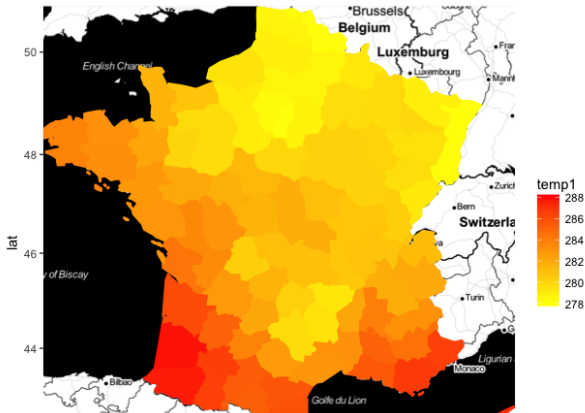
ggplot2

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



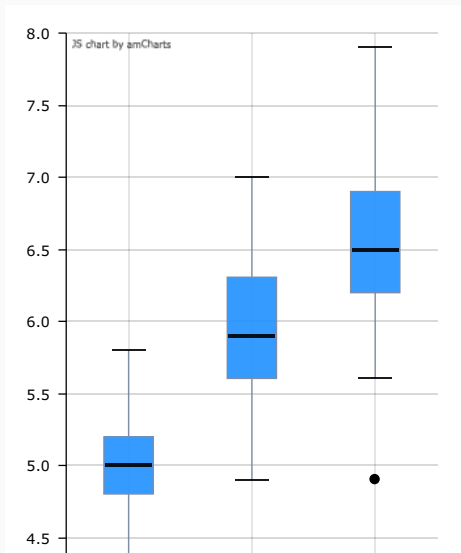
Une carte de températures avec ggmap

```
> library(ggmap)
> ggmap(fond)+geom_polygon(data=Test5,
+   aes(y=Latitude,x=Longitude,fill=temp1,color=temp1,
+     group=dept),size=1)+
+   scale_fill_continuous(low="yellow",high="red")+
+   scale_color_continuous(low="yellow",high="red")
```



Visu interactive avec rAmCharts

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



Applications web interactives avec shiny

- **Shiny** est un package R qui permet de faire “facilement” des applications web interactives.
- **Example:** stations velib à Rennes
<https://data.rennesmetropole.fr/page/home/>

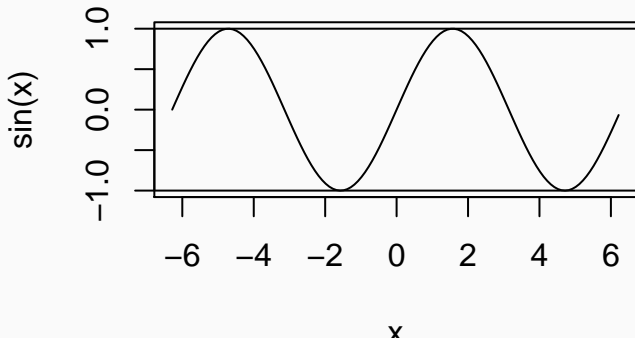
```
> library(shiny)
> runApp('desc_app.R')
> runApp('app_velib.R')
> runApp('trace_roc_app.R')
```

Visualisation avec ggplot2

La fonction plot (rappels)

- 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

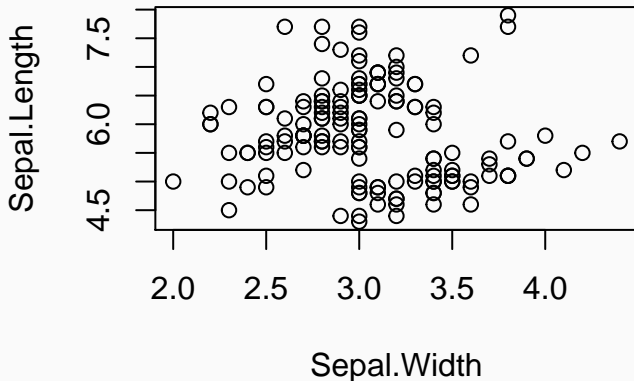
- **Histogramme** pour une variable **continue**, **diagramme en barres** pour une variable **qualitative**.
- **Nuage de points** pour 2 variables continues.
- **Boxplot** pour une distribution continue.

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,col="red")
```

Histogram of iris\$Sepal.Length

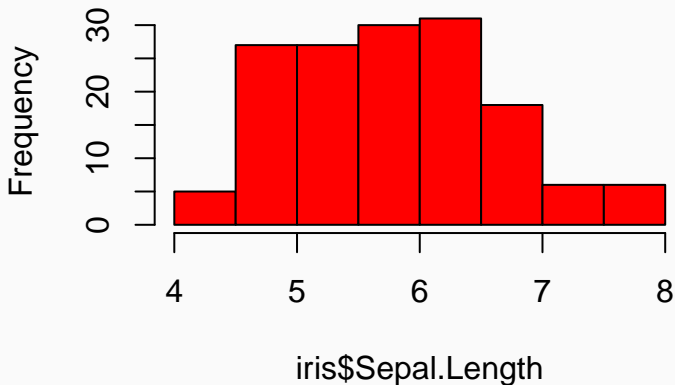
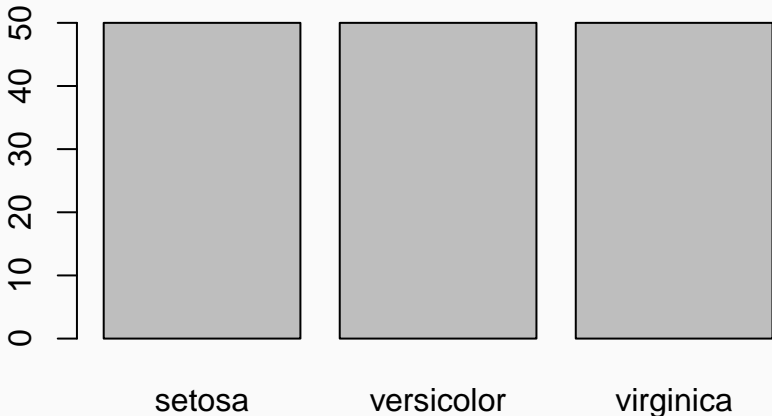


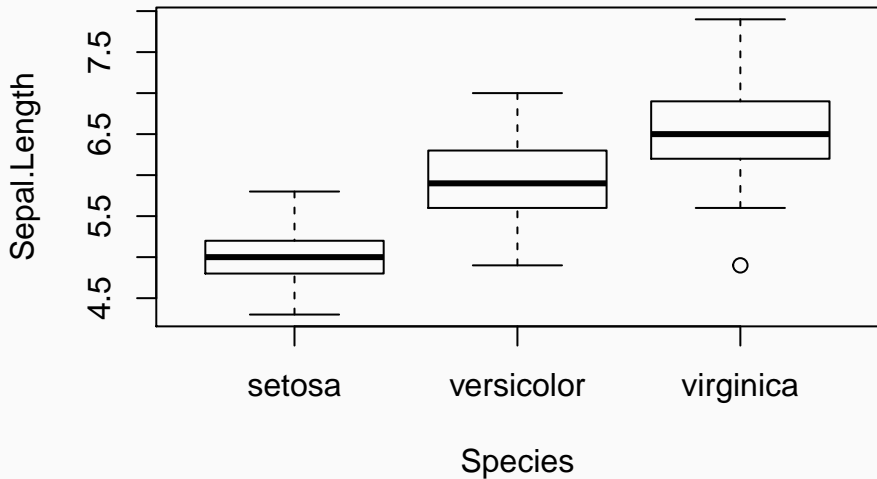
Diagramme en barres (variable qualitative)

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



Boxplot (distribution)

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



- `ggplot2` permet de faire des graphes R en s'appuyant sur une **grammaire des graphiques** (équivalent de **dplyr** pour manipuler les données).
- Les graphes produits sont **agréables à regarder** (pas toujours le cas avec les graphes conventionnels).
- La **grammaire ggplot** permet d'obtenir des **graphes "complexes"** avec une **syntaxe claire et lisible**.

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

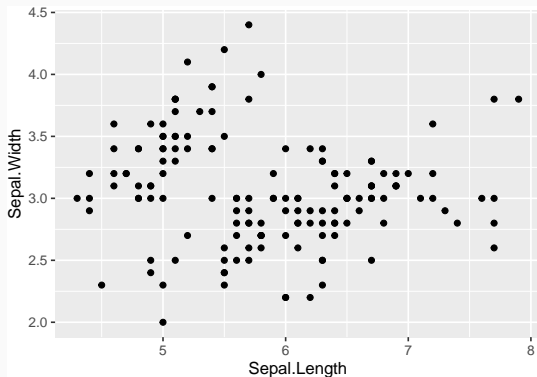
Les principaux **verbes** sont

- **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 +**.

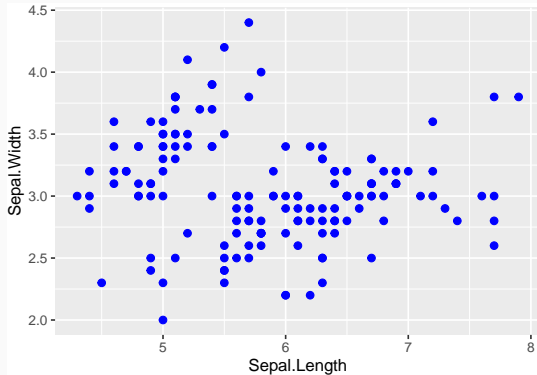
Un exemple

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



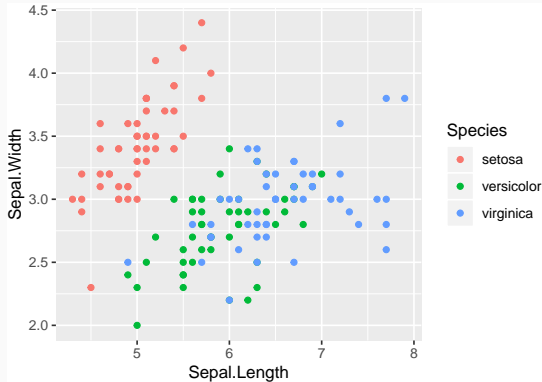
Couleur et taille

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



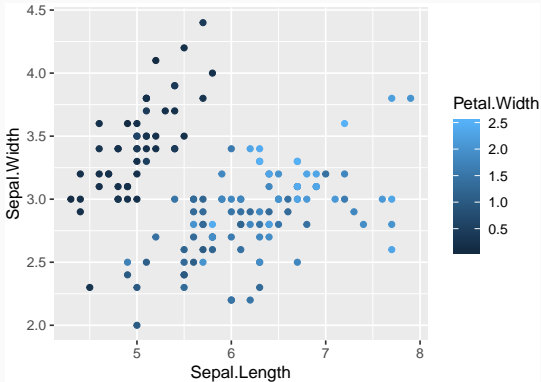
Couleur avec une variable qualitative

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



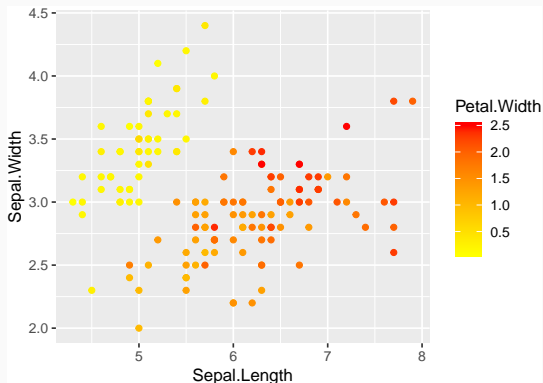
Couleur avec une variable continue

```
> 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")
```

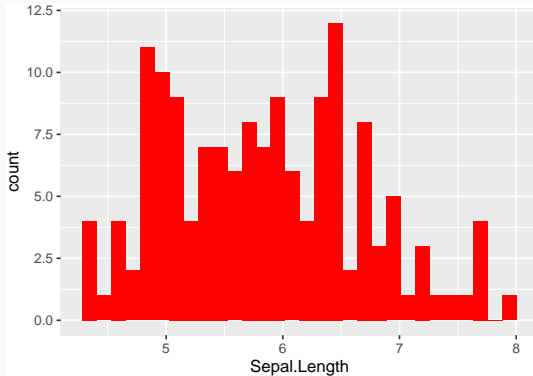
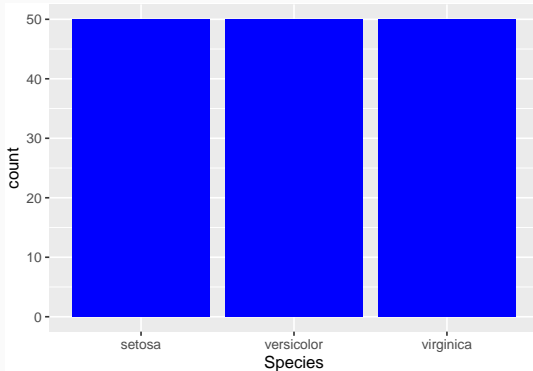


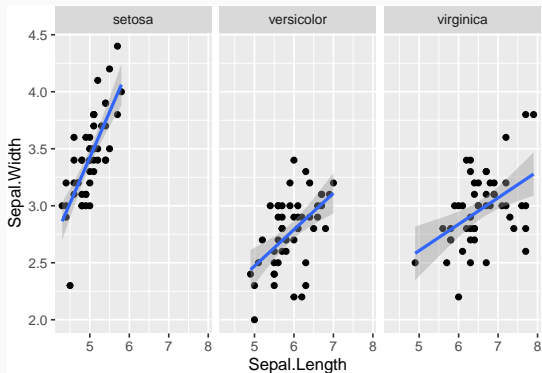
Diagramme en barres

```
> 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)
```

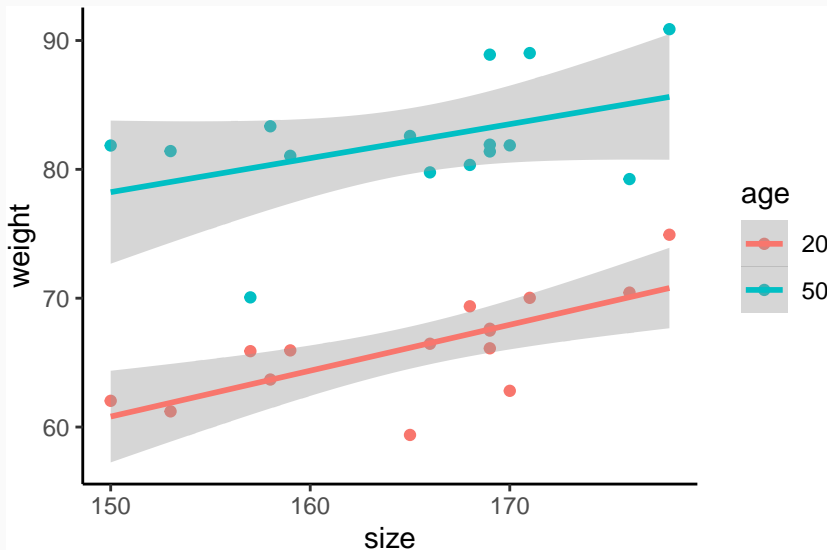


Combiner ggplot et dplyr

- 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



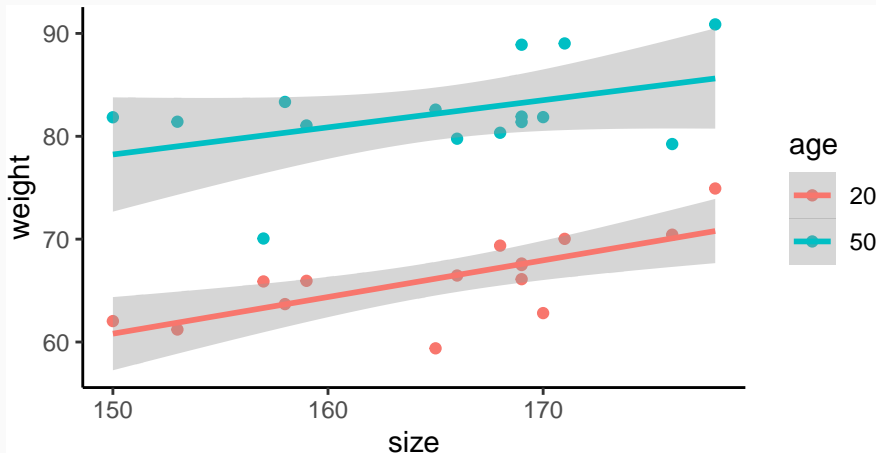
Etape dplyr

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

```
> df1 <- df %>% gather(key=age,value=weight,-size)
> df1 %>% head()
## # A tibble: 6 x 3
##   size age      weight
##   <dbl> <chr>    <dbl>
## 1   153 weight.20  61.2
## 2   169 weight.20  67.5
## 3   168 weight.20  69.4
## 4   169 weight.20  66.1
## 5   176 weight.20  70.4
## 6   169 weight.20  67.6
> 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

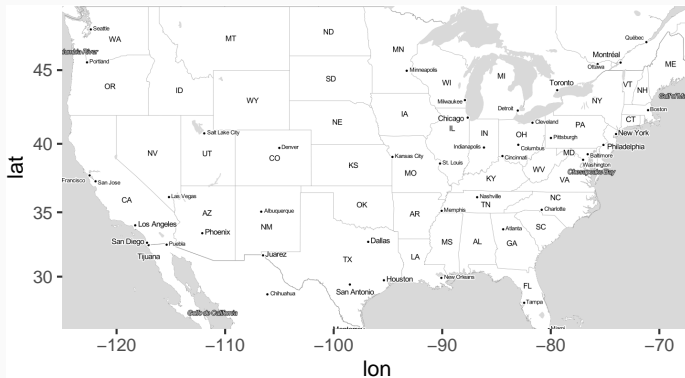
```
> demo(image)
> example(contour)
> demo(persp)
> library("lattice");demo(lattice)
> example(wireframe)
> library("rgl");demo(rgl)
> example(persp3d)
> demo(plotmath);demo(Hershey)
```

Cartes

- 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** et **leaflet**.

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")
```



Background map boundaries

- One can also draw background maps with **longitudes and latitudes of boundaries**.

```
> library(maps)
> france <- map(database="france",plot=FALSE)
> fr1 <- tibble(lon=france$x,lat=france$y)
> head(fr1)
## # A tibble: 6 x 2
##   lon   lat
##   <dbl> <dbl>
## 1  2.56  51.1
## 2  2.58  51.0
## 3  2.61  51.0
## 4  2.63  51.0
## 5  2.63  50.9
## 6  2.60  50.9
```

```
> ggplot(fr1)+geom_path(aes(y=lat,x=lon),size=1)+theme_void()
```



Le package sf

- Permet de gérer des **objets spécifiques à la cartographie** : notamment les différents **systèmes de coordonnées** et **leurs projections en 2d**.
- **Exemple** : colorier des régions (définies par des polygones).
- Compatible avec **ggplot**.

Références

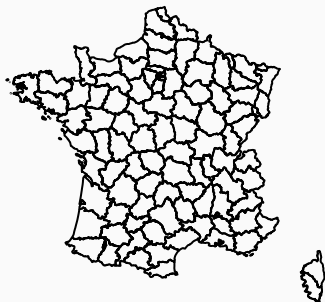
- <https://statnmap.com/fr/2018-07-14-initiation-a-la-cartographie-avec-sf-et-compagnie/>
- **Vignettes** sur le cran :
<https://cran.r-project.org/web/packages/sf/index.html>.

Exemple

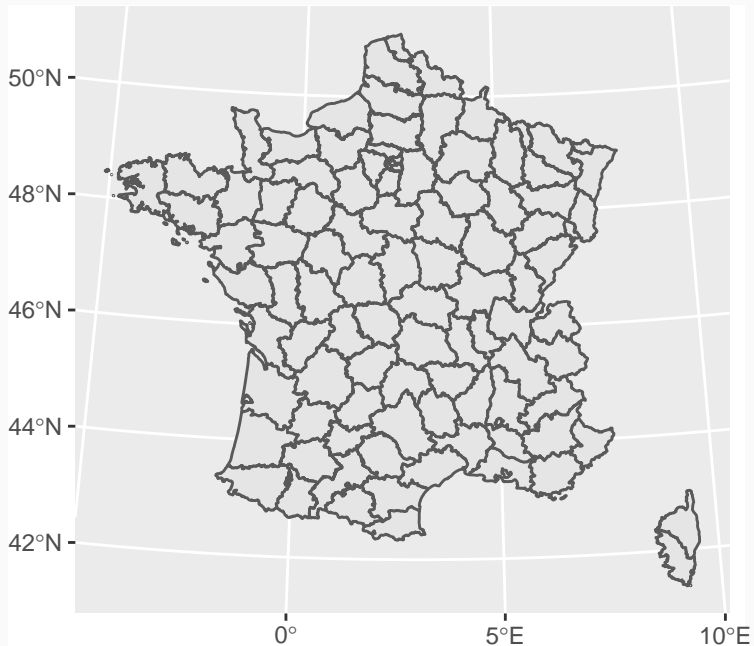
```
> library(sf)
> dpt <- read_sf("dpt")
> dpt[1:5,3]

## Simple feature collection with 5 features and 1 field
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 644570 ymin: 6290136 xmax: 1022851 ymax: 6997000
## epsg (SRID):    2154
## proj4string:     +proj=lcc +lat_1=49 +lat_2=44 +lat_0=46.5 +lon_0=3 +x_0=7
## # A tibble: 5 x 2
##   NOM_DEPT                                <MULTIPOLYGON>
##   <chr>
## 1 AIN          (((919195 6541470, 918932 6541203, 918628 6540523, 9
## 2 AISNE        (((735603 6861428, 735234 6861392, 734504 6861270, 7
## 3 ALLIER       (((753769 6537043, 753554 6537318, 752879 6538099, 7
## 4 ALPES-DE-HAUTE-PR~ (((992638 6305621, 992263 6305688, 991610 6306540, 9
## 5 HAUTES-ALPES (((1012913 6402904, 1012577 6402759, 1010853 6402931
```

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



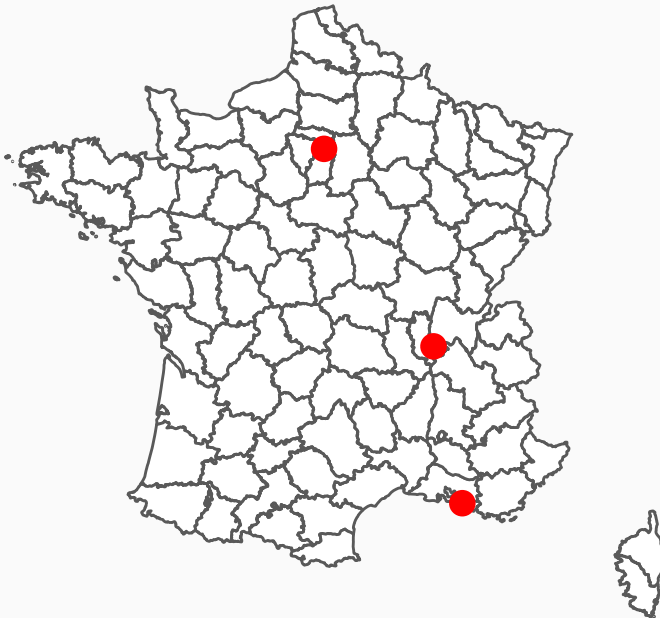
```
> ggplot(dpt)+geom_sf()
```



Ajouter des points sur le graphe

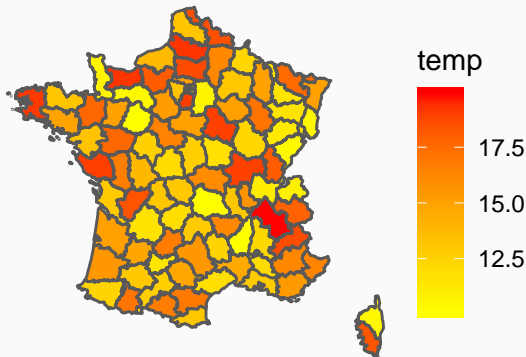
```
> point <- st_sfc(st_point(c(2.351462,48.85670)),  
+               st_point(c(4.832011,45.75781)),  
+               st_point(c(5.369953,43.29617)))  
> st_crs(point) <- 4326 #coord sont des long/lat  
> point  
## Geometry set for 3 features  
## geometry type: POINT  
## dimension: XY  
## bbox: xmin: 2.351462 ymin: 43.29617 xmax: 5.369953 ymax: 48.85670  
## epsg (SRID): 4326  
## proj4string: +proj=longlat +datum=WGS84 +no_defs
```

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



Colorier des polygones

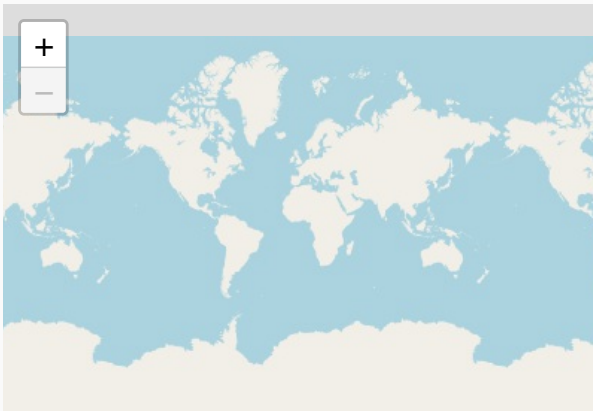
```
> set.seed(1234)
> dpt1 <- dpt %>% mutate(temp=runif(96,10,20))
> ggplot(dpt1) + geom_sf(aes(fill=temp)) +
+   scale_fill_continuous(low="yellow",high="red")+
+   theme_void()
```



Fonds de carte

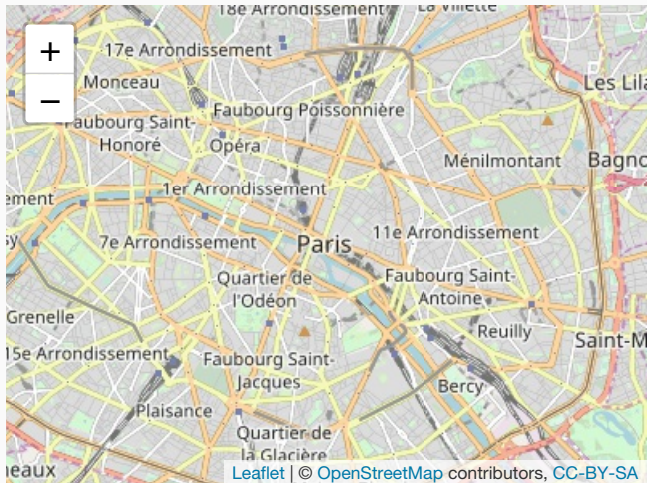
- **Leaflet** est une des librairies open-source JavaScript les plus populaires pour faire des **cartes interactives**.
- **Documentation**: [here](#)

```
> 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)
```



- Localiser 1000 séismes près des Fiji

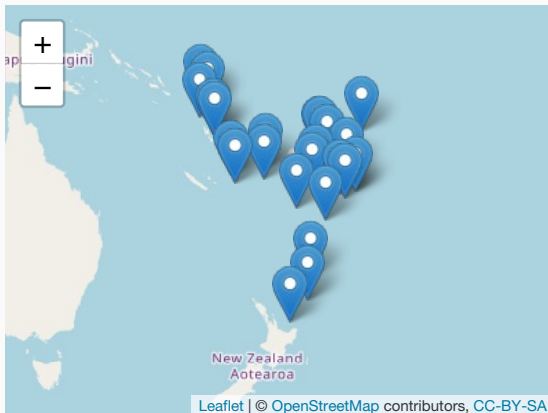
```
> data(quakes)
```

```
> 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")
```

