

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.

- (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).

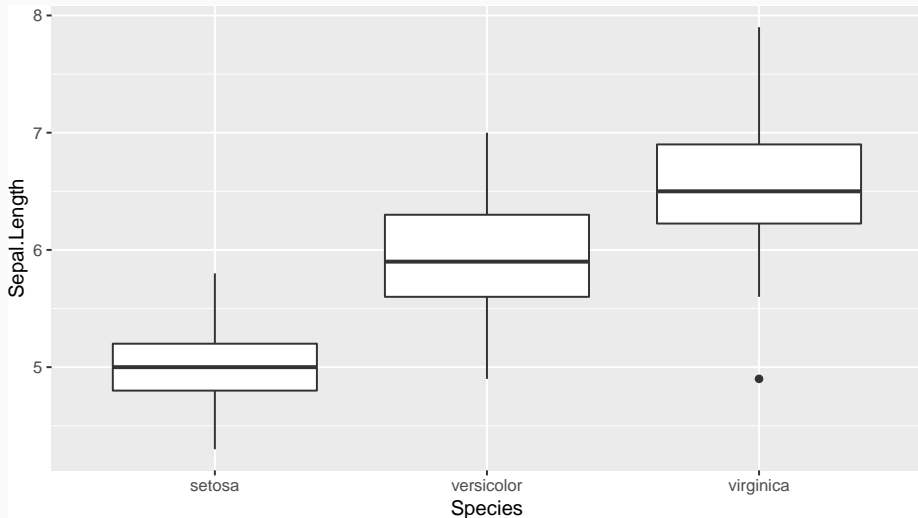
Remarque

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

Introduction

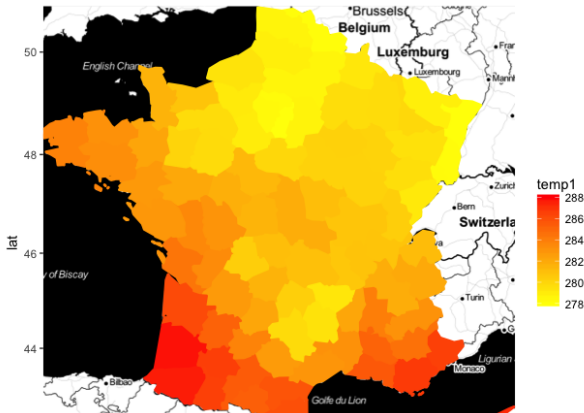
ggplot2

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



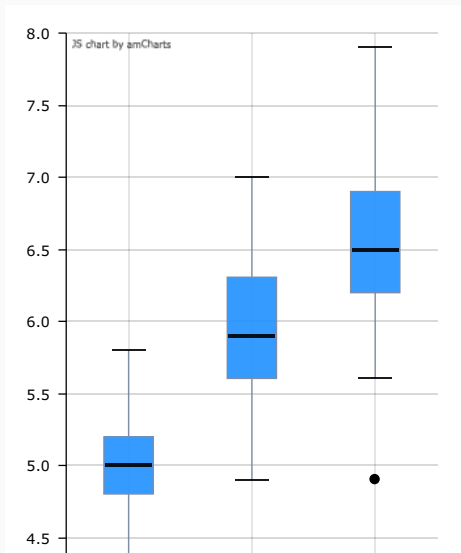
Une carte de températures avec ggmap

```
> 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/>

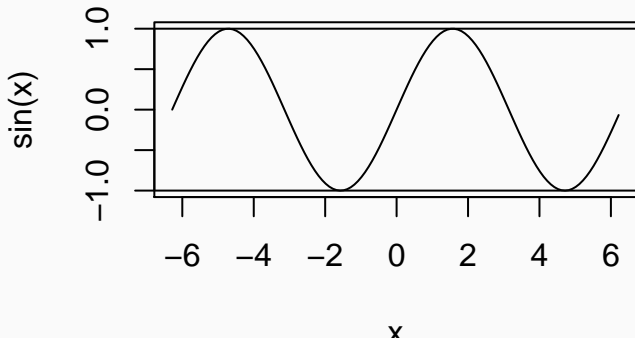
```
> library(shiny)
> runApp('desc_app.R')
> runApp('app_velib.R')
> runApp('trace_roc_app.R')
```

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

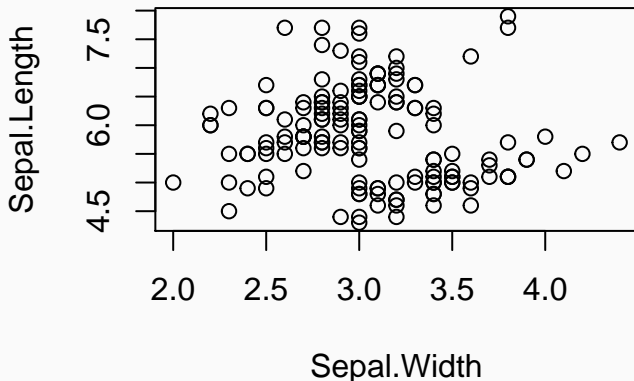
- **Histogramme** pour une variable **continue**, **diagramme en barres** pour une variable **qualitative**.
- **Nuage de points** pour 2 variables continues.
- **Boxplot** pour une distribution continue.

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

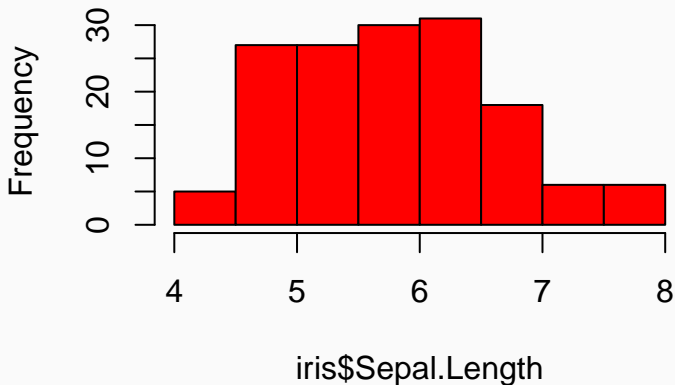
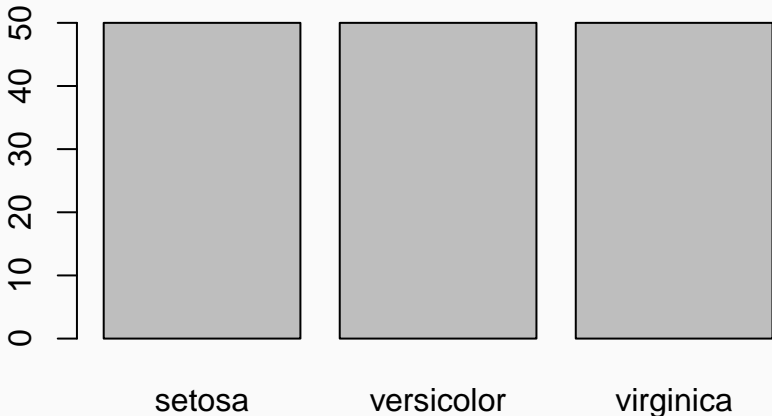


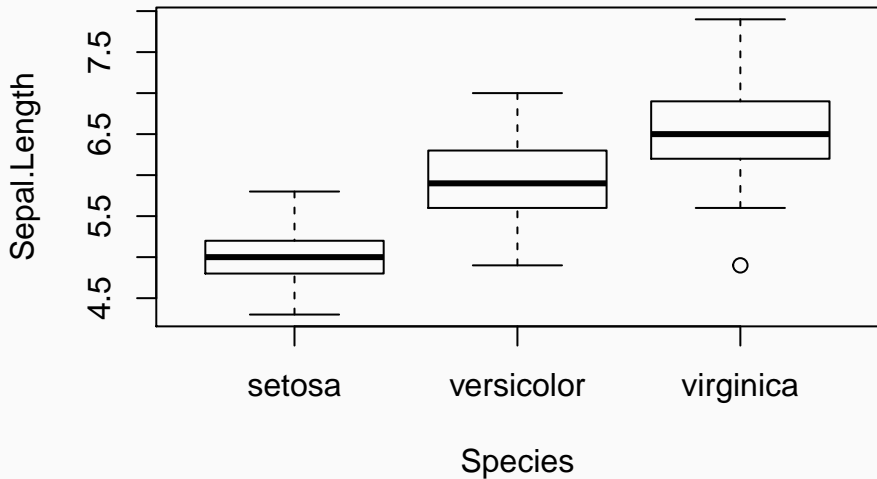
Diagramme en barres (variable qualitative)

```
> 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

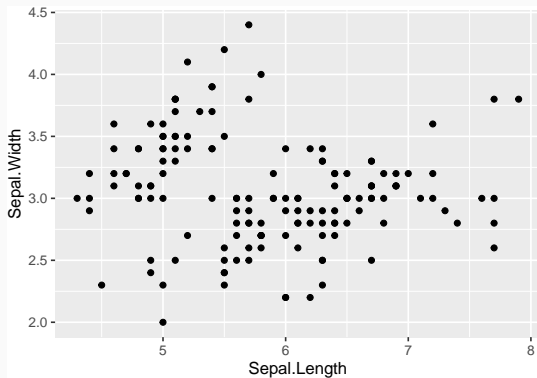
Les principaux **verbes** sont

- **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 +**.

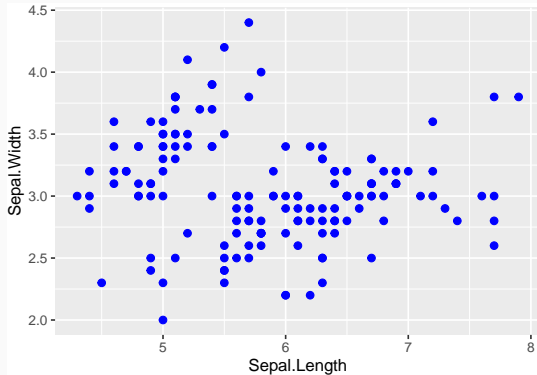
Un exemple

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



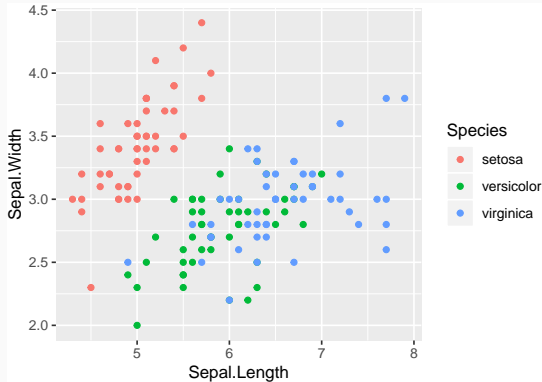
Couleur et taille

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



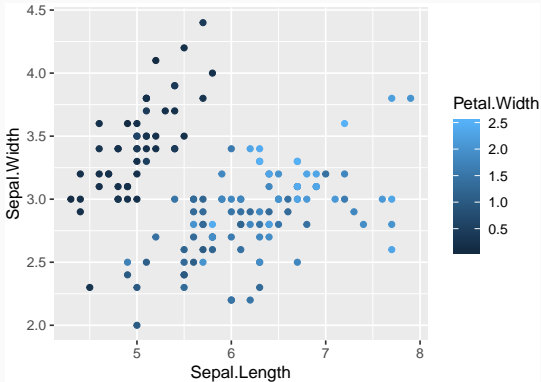
Couleur avec une variable qualitative

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



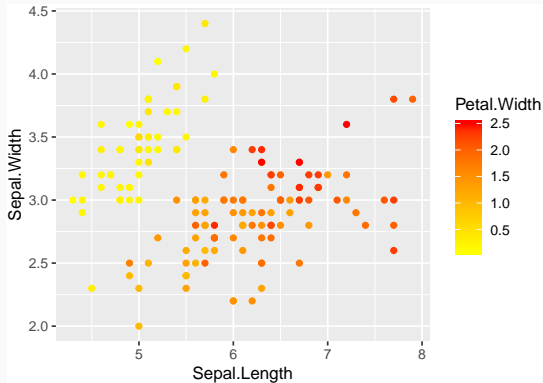
Couleur avec une variable continue

```
> 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")
```

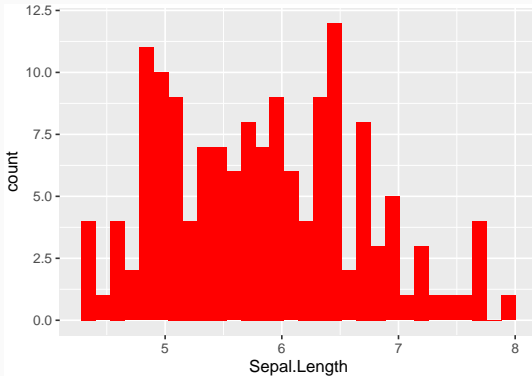
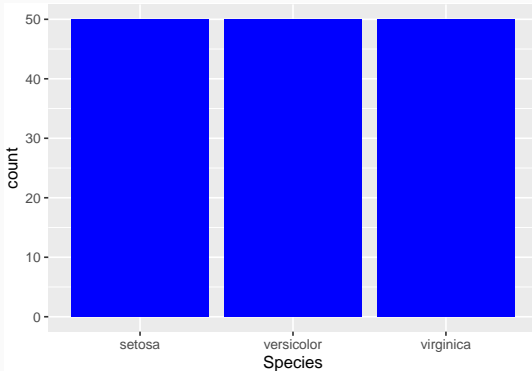


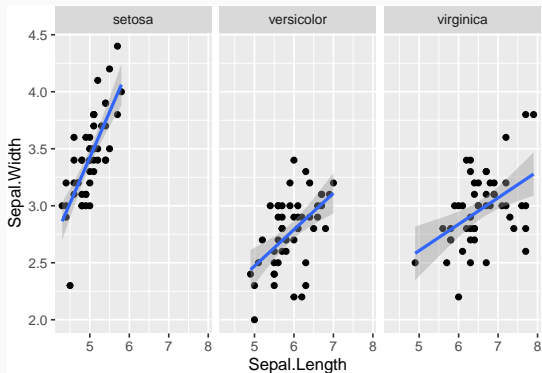
Diagramme en barres

```
> 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)
```

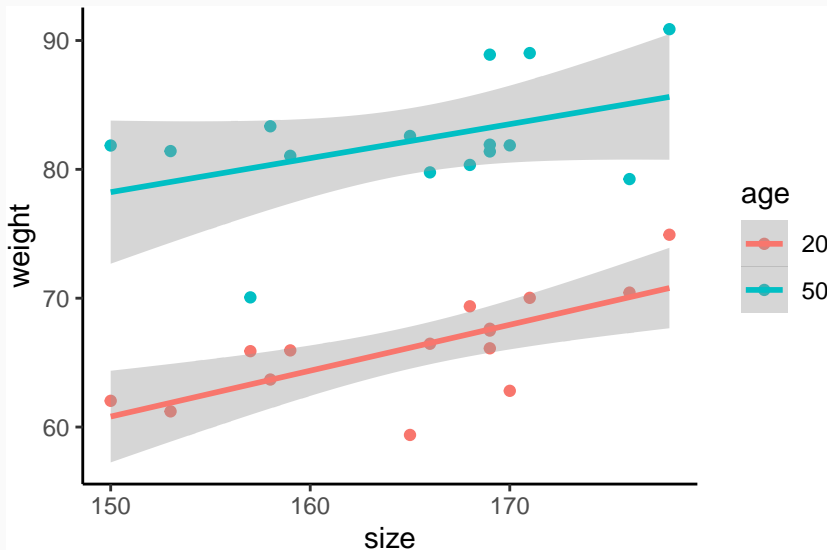


Combiner ggplot et dplyr

- 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



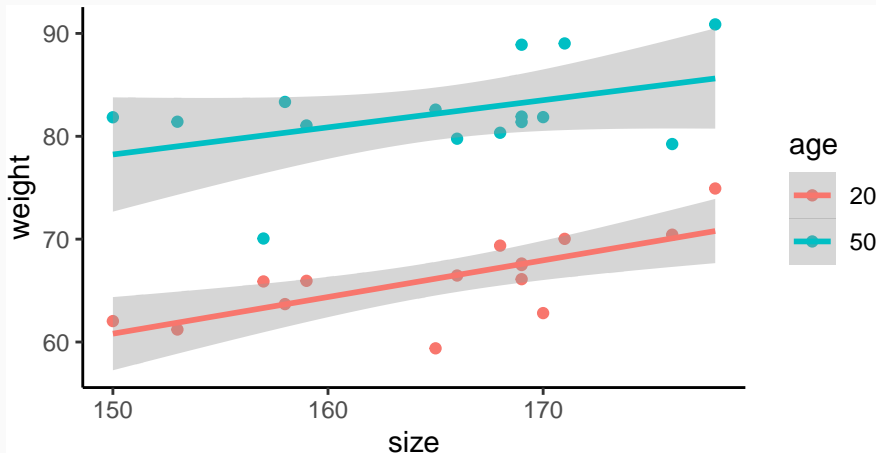
Etape dplyr

- 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)
```