Chapitre 5

La statistique descriptive

5.1 Mesures

L'objet des statistiques est de collecter des observations liées à des objets qui présentent un certain attribut et de convertir ces observations en nombres qui permettent d'avoir des renseignements sur cette propriété. Le but de la statistique descriptive est de structurer et de représenter l'information contenue dans les données. La population est l'ensemble des sujets observés. Le caractère est la propriété étudiée sur ces sujets. Afin de bien présenter cette partie on va utiliser database iris (mesures multiples dans les problèmes taxonomiques, contient trois espèces de plantes (setosa, virginica, versicolor))

```
> library(datasets) #Charger le package datasets
> head(iris) # Fonction head() Renvoyer la première ou la dernière partie d'un objet
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

La fonction summary() permet d'avoir la description statistique d'une variable ou d'une table de donnée.

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 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
> summary(iris\$Sepal.Length) #description statistique pour une seul variable
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
  4.300   5.100   5.800   5.843   6.400   7.900
```

On peut utiliser les fonctions spéciales de R (voir 3.2)

```
> sapply(iris[1:4], mean) Calcule la moyenne pour les 4 premièrs colonnes
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
    5.843333     3.057333     3.758000     1.199333
> sapply(iris[1:4], sd) #Calcule l'ecarte type pour  premièrs colonnes 4 colonnes
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
   0.8280661    0.4358663    1.7652982    0.7622377
> sapply(iris[1:2],quantile) #Calcul du quantile
     Sepal.Length Sepal.Width
0%            4.3         2.0
25%           5.1         2.8
50%           5.8         3.0
75%           6.4         3.3
100%          7.9         4.4
> sapply(iris[1:2],range) #Calcule de l'etendue
     Sepal.Length Sepal.Width
[1,]          4.3         2.0
[2,]          7.9         4.4
> table(cut(iris\$Sepal.Length, breaks = 3,include.lowest = TRUE))
#Range la variable Sepan.Length en trois intervalles
[4.3,5.5] (5.5,6.7] (6.7,7.9]
       59        71        20
```

On est obliger toujours de vérifier si les variables suivent la noi normal, pour cela, il exist une fonction shapiro.test

```
> shapiro.test(iris\$Sepal.Length)

        Shapiro-Wilk normality test

data:  iris\$Sepal.Length
W = 0.97609, p-value = 0.01018
> library(moments)
> skewness(iris\$Sepal.Length)
[1] 0.3117531
> kurtosis(rnorm(100,0,1))
[1] 2.965781
```

le test renvoie une p-value ($0.01018 < 5\%$) significative. la variable ne suit pas une loi normale. La fonction skewness qui appartient au package moments, la valeurs de l'asymétrie est positive ce qui confirme le resultat du shapiro.test, donc cette valeurs indique que la distribution (Sepal.Length) est asymétrique à droite, donc on a mode $<$ mean $<$ mean. D'autre parte on a utilisé la fonction kurtosis pour avoir une information sur l'aplatissement de la distribution (100 valeurs $\sim \mathcal{N}(0,1)$), la valeur positive de la fonction kurtosis indique que nous avons une distribution à longue queue. Il existe de nombreuses statis-

tiques sommaires qui nous donne toute l'information d'une manière simple et puissante, prenons par exemple la fonction describe() dans le package psych (package qui fournit outils d'analyse des données)

```
> library(psych)
> describe(iris)
             vars   n mean   sd median trimmed  mad min max range  skew kurtosis   se
Sepal.Length    1 150 5.84 0.83   5.80    5.81 1.04 4.3 7.9   3.6  0.31    -0.61 0.07
Sepal.Width     2 150 3.06 0.44   3.00    3.04 0.44 2.0 4.4   2.4  0.31     0.14 0.04
Petal.Length    3 150 3.76 1.77   4.35    3.76 1.85 1.0 6.9   5.9 -0.27    -1.42 0.14
Petal.Width     4 150 1.20 0.76   1.30    1.18 1.04 0.1 2.5   2.4 -0.10    -1.36 0.06
Species*        5 150 1.00 0.00   1.00    1.00 0.00 1.0 1.0   0.0   NaN      NaN 0.00
```

5.2 Visualisation des données

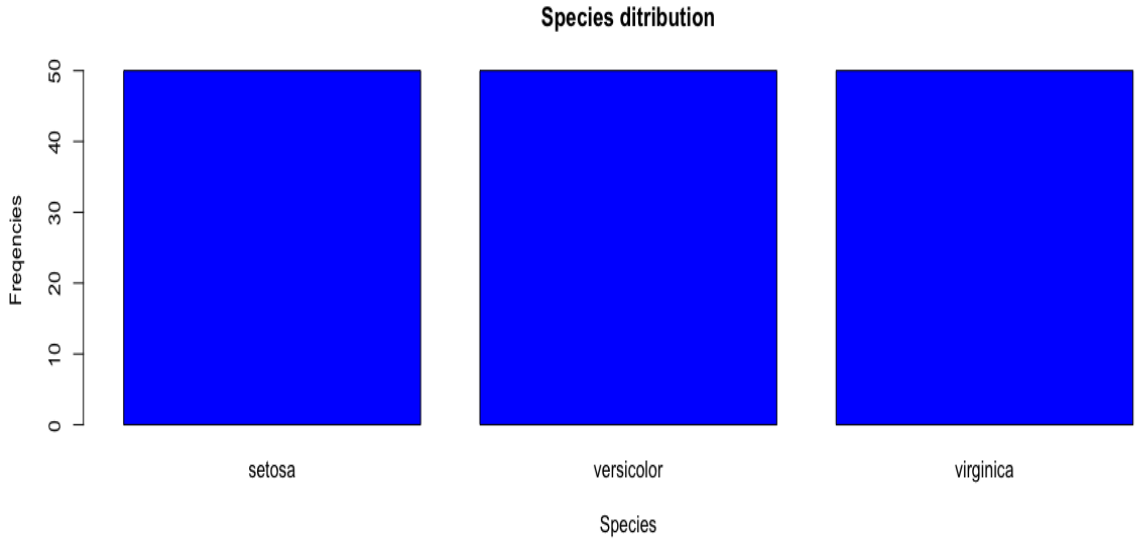
La fonction de visualisation la plus utilisée en programmation R est la fonction plot(). Il s'agit d'une fonction générique, ce qui signifie qu'elle possède de nombreuses méthodes qui sont appelées en fonction du type d'objet passé à plot(). Dans le cas le plus simple, nous pouvons passer un vecteur et nous obtiendrons un diagramme de dispersion de la magnitude par rapport à l'indice. Mais généralement, nous passons deux vecteurs et un nuage de points de ces points est tracé. Mais il existe aussi ggplot2 un package dédié à la visualisation de données. Il peut améliorer considérablement la qualité et l'esthétique de vos graphiques, et vous rendra beaucoup plus efficace dans leur création. il permet de construire presque n'importe quel type de graphique.

5.2.1 plot()

On import le cadre de données de l'iris en utilisant le package datasets, on essaye de faire la visualisation de quatre cas par l'appele du syntaxe plot(x,y) ou plot(y x).

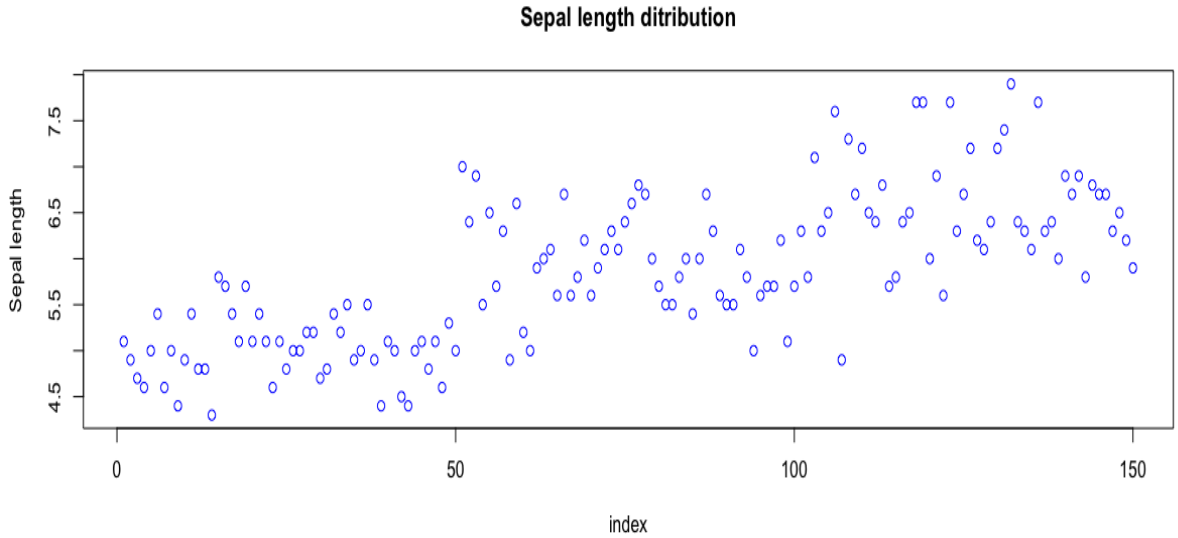
```
> library(datasets)
> plot(iris\$Species,main="plot",xlab="Species",ylab="Freqencies",col="#ff0000")
```

la repésentation d'une variable qualitative (Species) par plot() nous donne digramme en barre




```
plot(iris\$Sepal.Length,main="Sepal length ditribution",xlab="index",
+     ylab="Sepal length",col="#0000ff")
```

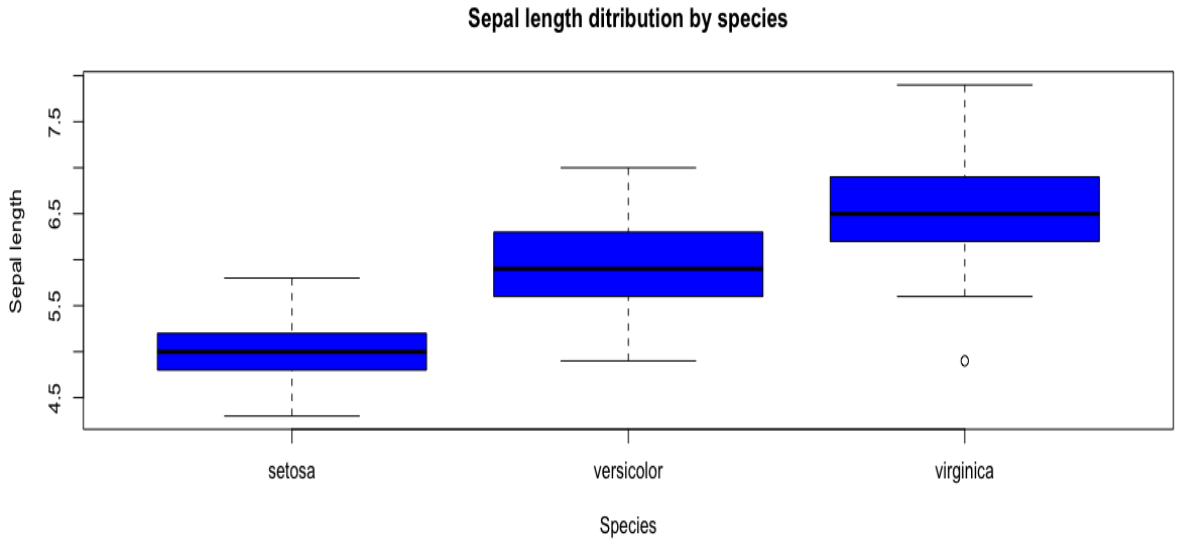
la représentation d'une variable quantitative (Sepal length) par la fonction plot() nous nuage de points




```
> plot(iris\$Species,iris\$Sepal.Length,main="Sepal length ditribution by species",
+      xlab="Species",ylab="Sepal length",col="#0000ff")
```

la représentation d'une variable quantitative (Sepal length) en fonction d'une variable qualitative (Species) par la fonction plot() nous donne des boîtes à moustaches pour

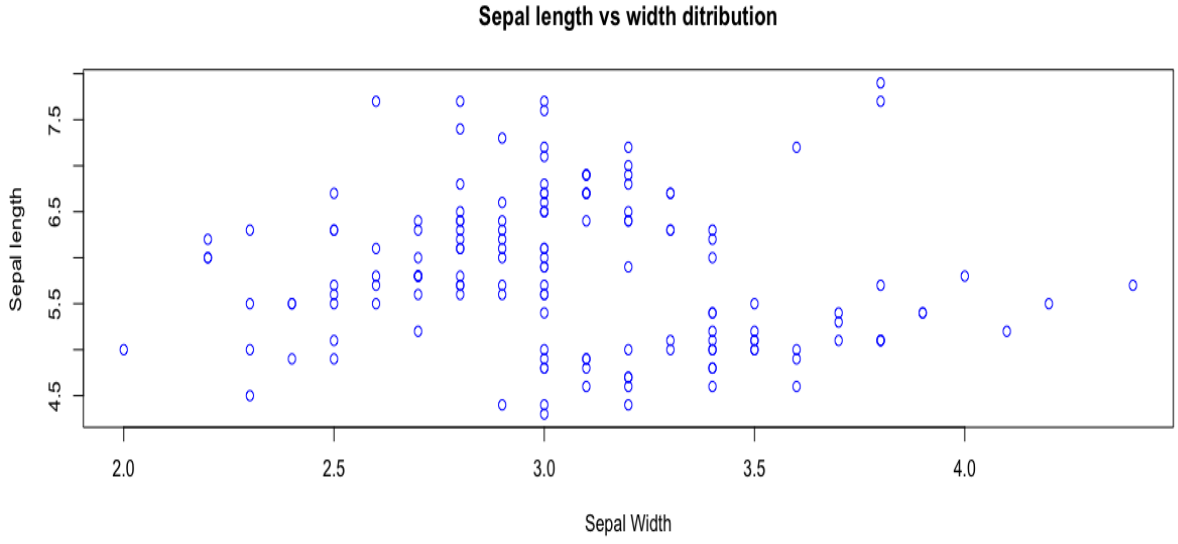
chaque modalité



la représentation d'une variable quantitative (Sepal length) en fonction d'une variable quatitative (Sepal length) par la fonction plot() nous donne nuage de points

```
> plot(iris\$Sepal.Width,iris\$Sepal.Length,main="Sepal length vs width ditribution",
+      xlab="Sepal Width",ylab="Sepal length",col="#0000ff")
```

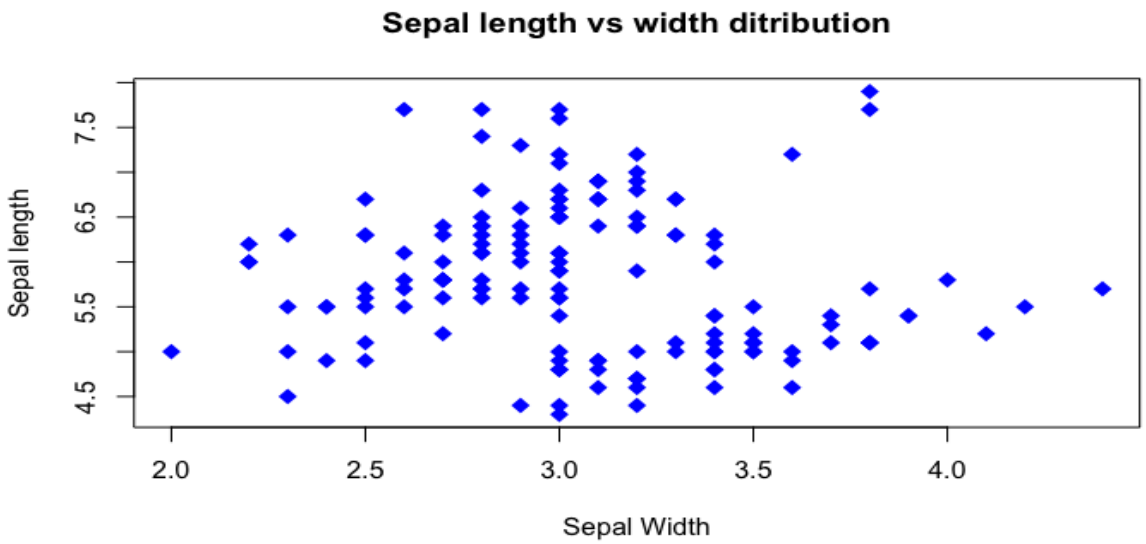
la représentation d'une variable quantitative (Sepal length) en fonction d'une variable par la fonction plot() nous donne des boîtes à moustaches pour chaque modalité



Il semble que la fonction plot() détecte automatiquement le type de variable, et les trace

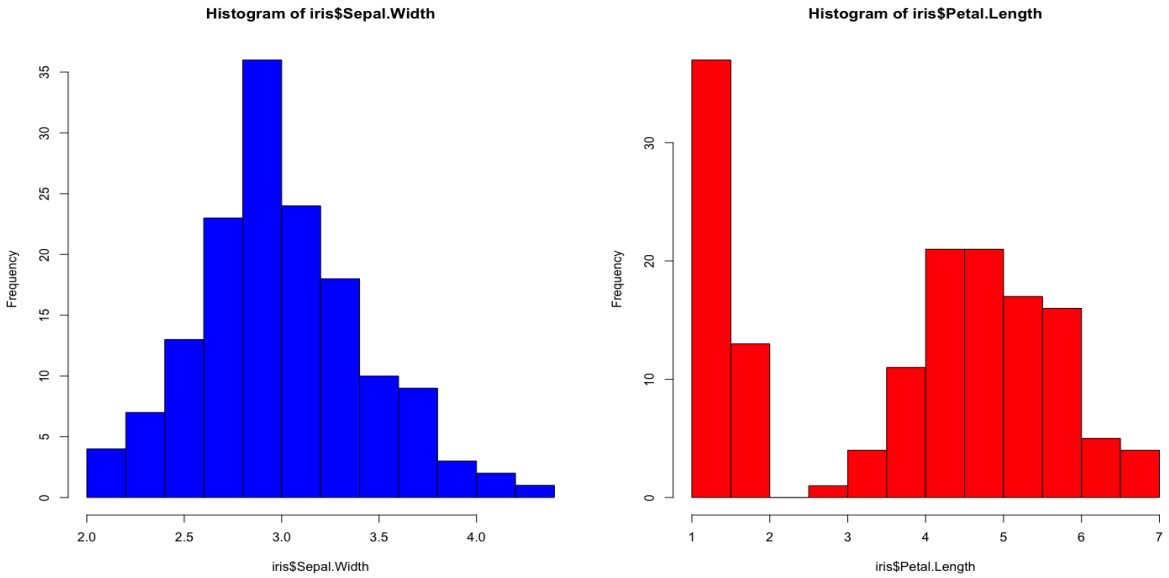
en fonction de leur type. On peut aussi personnaliser la visualisation en ajoutant quelques changements

```
> plot(iris\$Sepal.Width,iris\$Sepal.Length,
+      cex=1.5,
+      pch=18,
+      main="Sepal length vs width ditribution",
+      xlab="Sepal Width",
+      ylab="Sepal length",
+      col="#0000ff")
```



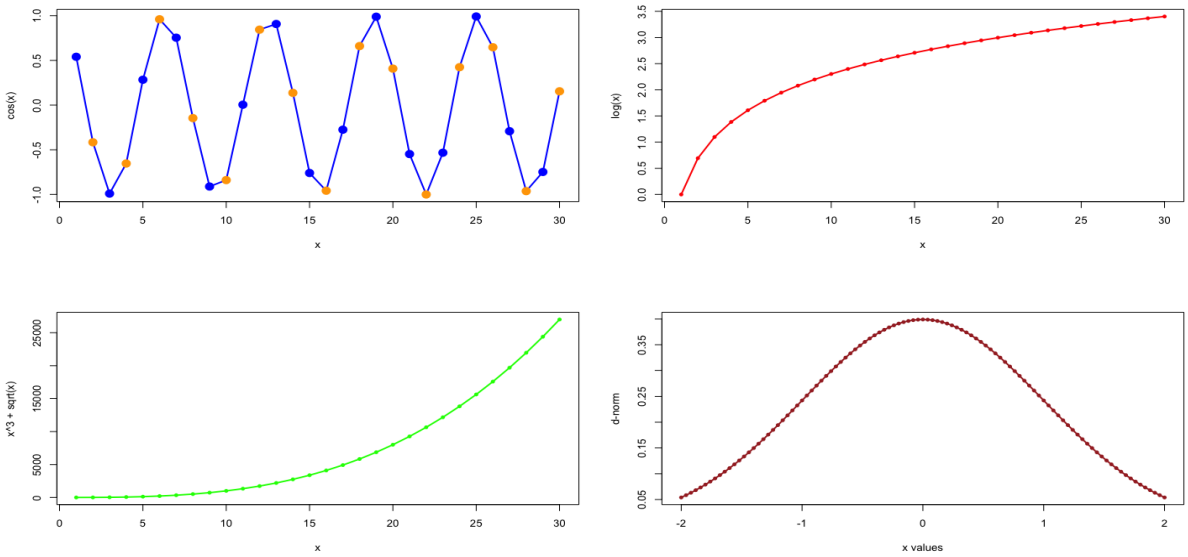
On peut présélectionner le type de graphique pour un certain type de données comme les données quantitative pour avoir un idée sur la distribution ou bien les aberrante...

```
 > par(mfrow=c(1,2))
> hist(iris\$Sepal.Width,col="blue")
> hist(iris\$Petal.Length,col="red")
```



On a la possibilité de mettre plusieurs graphics dans une seule par par la commande par(mfrow=c(l,c)), le vecteur c(l,c) (nombre de lignes et nombre de colonnes) divise la fenêtre graphique en l lignes et c colonnes.

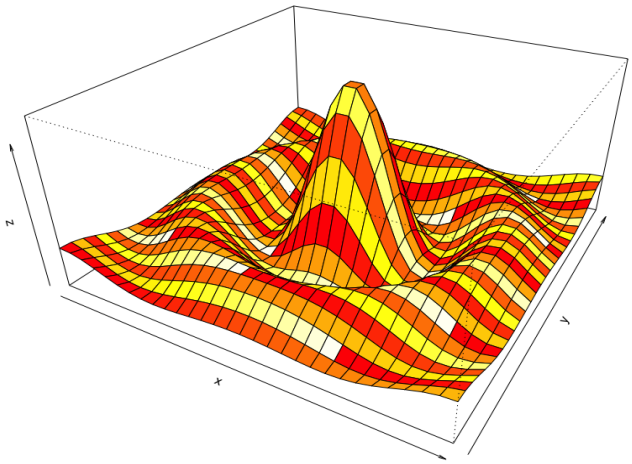
```
> par(mfrow=c(2,2))
> plot(x,cos(x),col=c("blue","orange"),type="o",pch=19,lwd=2,cex=1.5)
> plot(x,log(x),col="red",type="o",pch=19,lwd=2,cex=0.5)
> plot(x,x^3+sqrt(x),col="green",type="o",pch=19,lwd=2,cex=0.5)
> plot(dnorm,-2,2,col="brown",type="o",pch=19,lwd=2,cex=0.5,xlab="x values", ylab="d-norm")
```



Les fonctions classiques de représentation 3D sont les fonctions persp, prenons un cette

exemple suivante $f(x,y) = \dfrac{sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$

```
> f <- function(x,y){sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)}
> y <- x <- seq(-15,15,length=30)
> persp(x,y,z,theta=30,phi=30,expand=0.5,col=heat.colors(50)) #theta et phi pour la position
```



Un tableau Table 5.1 de certains fonctions avec ses paramètres qui pouvent être utiliser dans la visualisation des données.

Fonction	**Description**
barplot(vec)	trace un diagramme en barres des valeurs de vec
boxplot(vec)	trace le graphe en boîte à moustaches de vec
contour(vec_1,vec_2,vec_3)	trace des courbes de niveau, voir aussi la fonction filled.contour(vec_1,vec_2,vec_3)
coplot(vec_1,vec_2|val)	trace le graphe bivarié de vec_1 et vec_2 pour chaque valeur de val
filled.contour(vec_1,vec_2,vec_3)	trace des courbes de niveau mais les aires entre les contours sont colorées, voir aussi image(vec_1,vec_2,vec_3)
hist(vec,$prob = TRUE$)	trace un histogramme de vec
image(vec_1,vec_2,vec_3)	trace des rectangles aux coordonnées vec_1, vec_2 colorés selon vec_3, voir aussi contour(vec_1,vec_2,vec_3)
pairs(vec)	si vec est une matrice ou un data-frame, trace tous les graphes bivariés entre les colonnes de x
persp(vec_1,vec_2,vec_3)	trace une surface de réponse en 3D, voir demo(persp)
pie(vec)	trace un graphe en camembert
plot($objet$)	trace le graphique correspondant à la classe de objet
plot(vec_1,vec_2)	trace le nuage de points de coordonnées vec_1 et vec_2
qqnorm(vec)	trace les quantiles de vec en fonction de ceux attendusd'une loi normale
qqplot(vec_1,vec_2)	trace les quantiles de vec_2 en fonction de ceux de vec_1
spineplot(val_1,val_2)	trace le diagramme en bandes correspondant à val_1 et val_2
stripplot(vec)	trace le graphe des valeurs de vec sur une ligne
sunflowerplot(vec_1,vec_2)	idem mais les points superposés sont dessinés sous forme de fleurs dont le nombre de pétales correspond au nombre de points
symbols(vec_1,vec_2,...)	dessine aux coordonnées vec_1 et vec_2 des symboles (étoiles,cercles, boxplots, etc.)

TABLE 5.1 – Liste de fonctions graphiques

Un tableau Table 5.2 de certains paramètres qui pouvent étre utiliser par les fonctions graphiques.

Fonction	Description
adj	contrôle la justification du texte par rapport au bord gauche du texte : 0 à gauche, 0.5 centré, 1 à droite ; c($vec-1, vec_2$) justifie le texte horizontalement et verticalement
asp	précise le ratio entre $vec-2$ et vec_1 : asp=1 pour un graphe orthonormé axes par défaut TRUE, les axes et le cadre sont tracés
bg	spécifie la couleur de l'arrière-plan : 1, 2, ou une couleur
bty	contrôle le tracé du cadre, valeurs permises : "o","l", "7", "c","u" ou "]"
bty="n"	supprime le cadre
cex	contrôle la taille des caractères et des symboles par rapport à la valeur par défaut qui vaut 1
cex.axis	contrôle la taille des caractères pour l'échelle des axes
cex.lab	contrôle la taille des caractères pour les libellés des axes
cex.main	contrôle la taille des caractères du titre
cex.sub	contrôle la taille des caractères du sous-titre
col	précise la couleur du graphe, valeurs possibles 1, 2, ou une couleur.
col.axis	précise la couleur des axes
col.main	précise la couleur du titre
font	contrôle le style du texte
font.axis	contrôle le style pour l'échelle des axes
font.lab	contrôle le style pour les libellés des axes
font.main	contrôle le style du titre
font.sub	contrôle le style du sous-titre
las	contrôle la disposition des annotations sur les axes (0 : parallèles aux axes, 1 : horizontales, : perpendiculaires aux axes, 3 : verticales)
lty	contrôle le type de ligne tracée, (1 : continue, 2 : tirets, 3 : points, 4 : points et tirets alternés, 5 : tirets longs, 6 : tirets courts et longs alternés)
lwd	contrôle l'épaisseur des traits
main	précise le titre du graphe, par exemple main="titre"
mfrow	vecteur c(l,c) qui divise la fenêtre graphique en l lignes et c colonnes, les graphes sont ensuite dessinés en ligne
offset	précise le décalage du texte par rapport au point
pch	entier (entre 0 et 25) qui contrôle le type de symbole
pos	précise la position du texte, valeurs permises
	entier qui contrôle la taille en points du texte et des symboles
ps	précise le sous-titre du graphe, par exemple sub="sstitre"
sub	précise la longueur des graduations sur les axes
tck, tcl	précise le type de graphe dessiné, valeurs permises "n","p","l","b,"h","s","S"
xlim, ylim	précise les limites des axes, par exemple xlim=c(0,10)
xlab, ylab	précise les annotations des axes

TABLE 5.2 – Liste des paramètres

5.2.2 ggplot2

La première étape pour produire un graphique avec ggplot() est d'installer et de charger le package

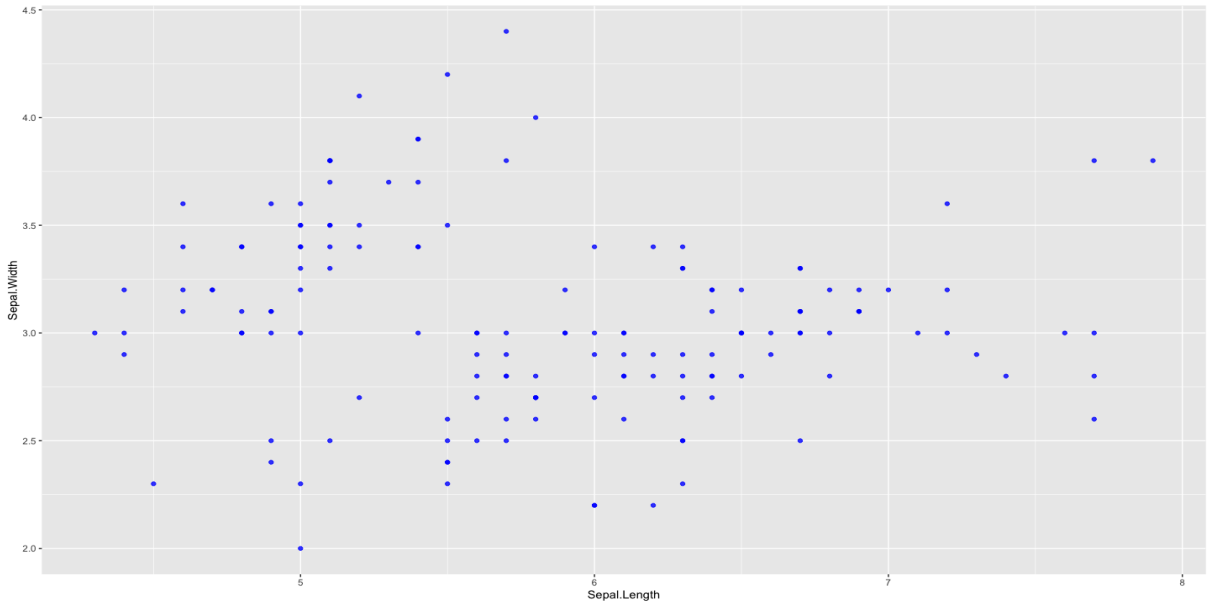
```
install.packages("ggplot2")
library(ggplot2)
```

Pour construire un ggplot(), nous utiliserons le modèle de base suivant qui peut être utilisé pour différents types de graphiques, par le syntaxe suivant

```
ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) +  <GEOM_FUNCTION>()
```

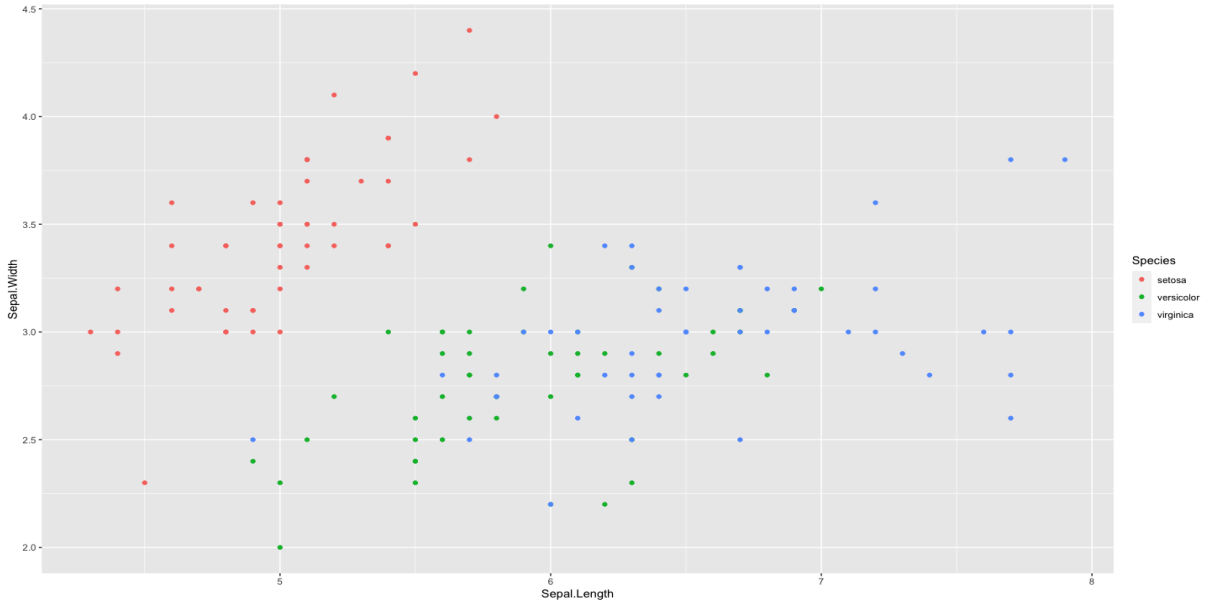
Bien entendu, nous utiliserons les mêmes données d' iris et aussi de mtcars qui se trouvent dans le paquet datasets, La création de graphiques avec ggplot2 est généralement un processus itératif. Nous commençons par définir le cadre de données que nous allons utiliser, nous disposons les axes et nous choisissons un géomètre, Sepal.Width en fonction de Sepal.Length

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



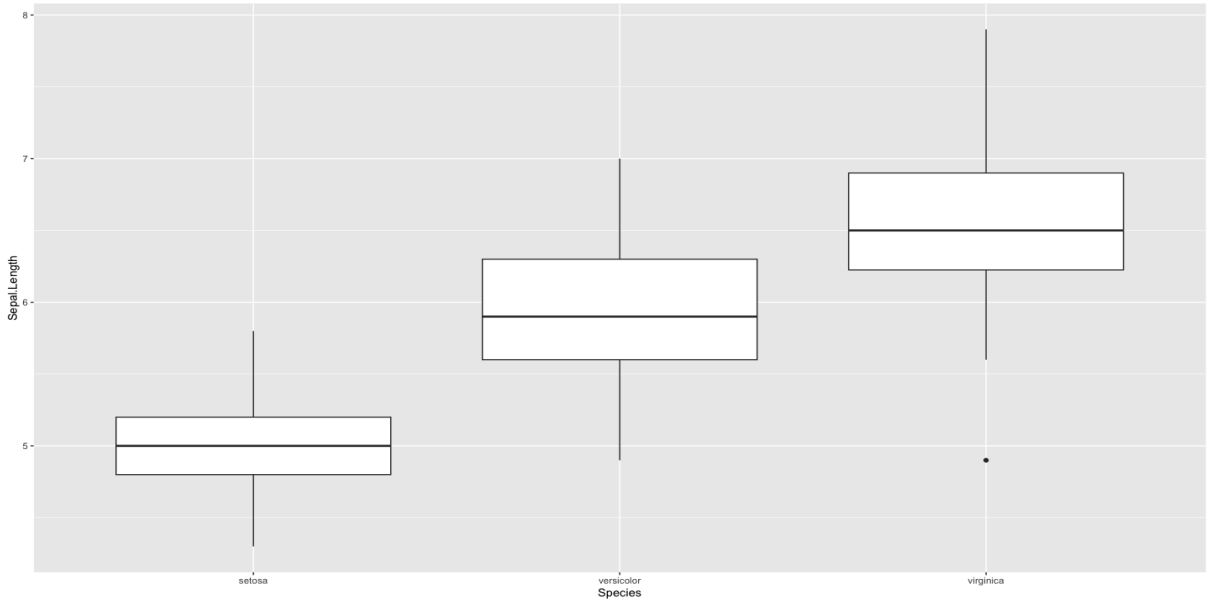
Même graphique mais en ajoutant les couleurs selon la variable Species

```
> ggplot(data = iris, mapping = aes(x = Sepal.Length, y = Sepal.Width)) +
+     geom_point(alpha = 2, aes(color = Species))
```

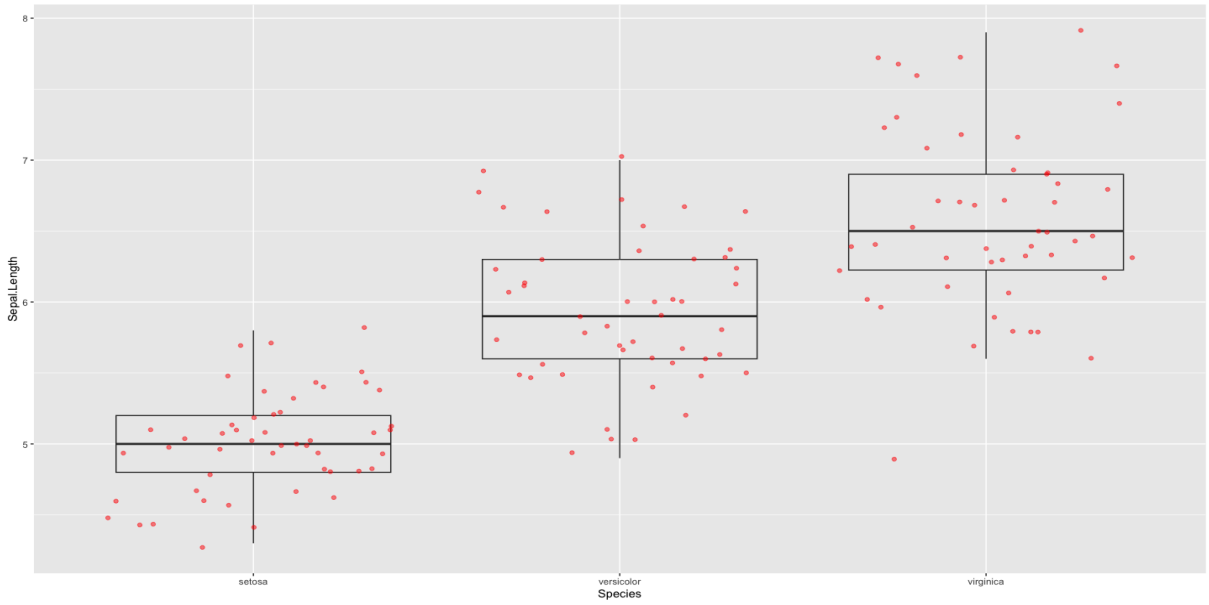


Nous pouvons utiliser des boxplots pour visualiser la distribution du Sepal.Length au sein de chaque type de Species,

```
> ggplot(data = iris, mapping = aes(x = Species, y = Sepal.Length)) +
+     geom_boxplot()
```

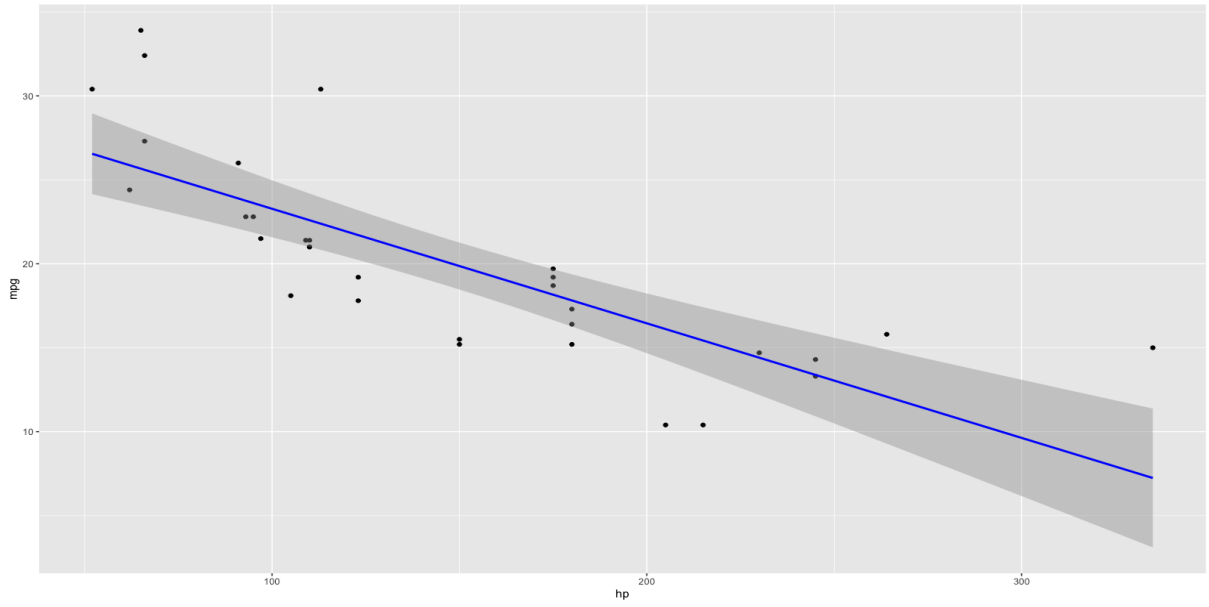



```
> ggplot(data = iris, mapping = aes(x = Species, y = Sepal.Length)) +
+     geom_boxplot(alpha = 0) +
+     geom_jitter(alpha = 0.5, color = "red")
```



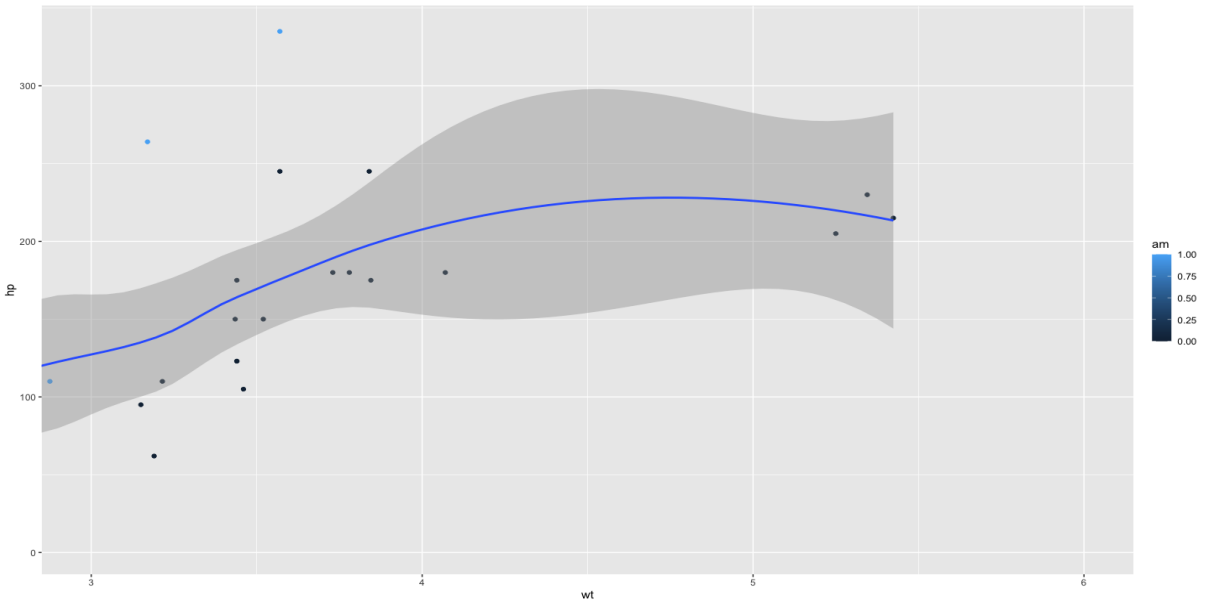
[H] Maintenantt avec le cadre de données mtcars (pour savoir ?mtcars), dans cette couche, nous transformons nos données (mpg :Miles/(US) gallon en fonction hp : Gross horsepower) en utilisant le binning, le lissage, la description, l'intermédiaire.

```
> ggplot(data = mtcars, aes(x = hp, y = mpg)) + geom_point() + stat_smooth(method = lm, col = "blue")
```

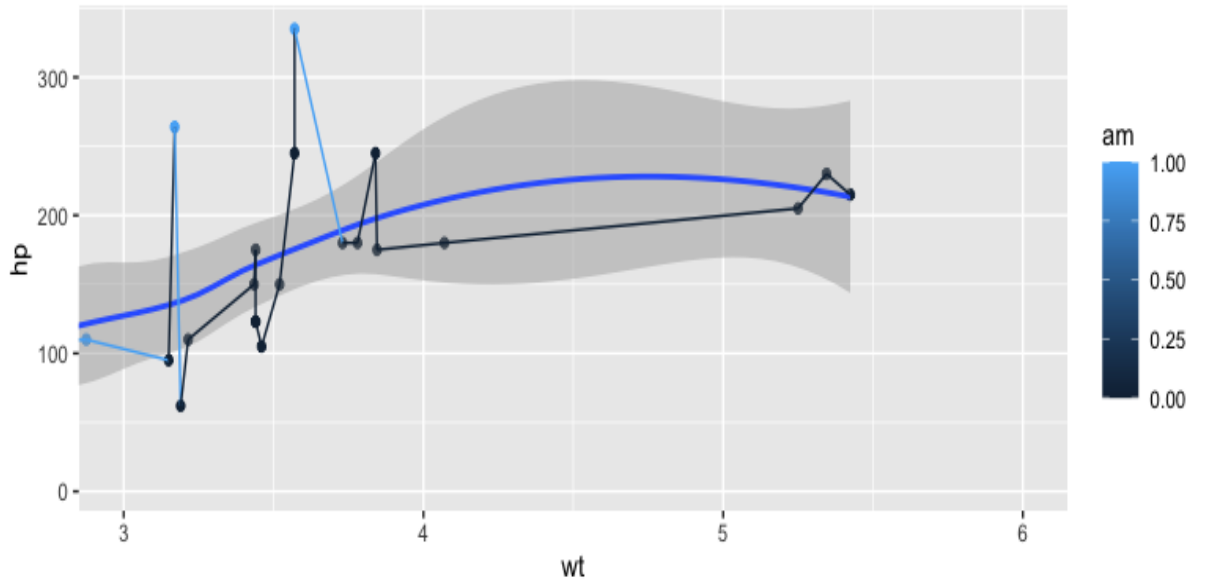


En ajoutant la fonction *coord_cartesian()* on contrôle les limites des axes x et y et vous permettent d'effectuer un zoom avant ou arrière sur votre tracé.

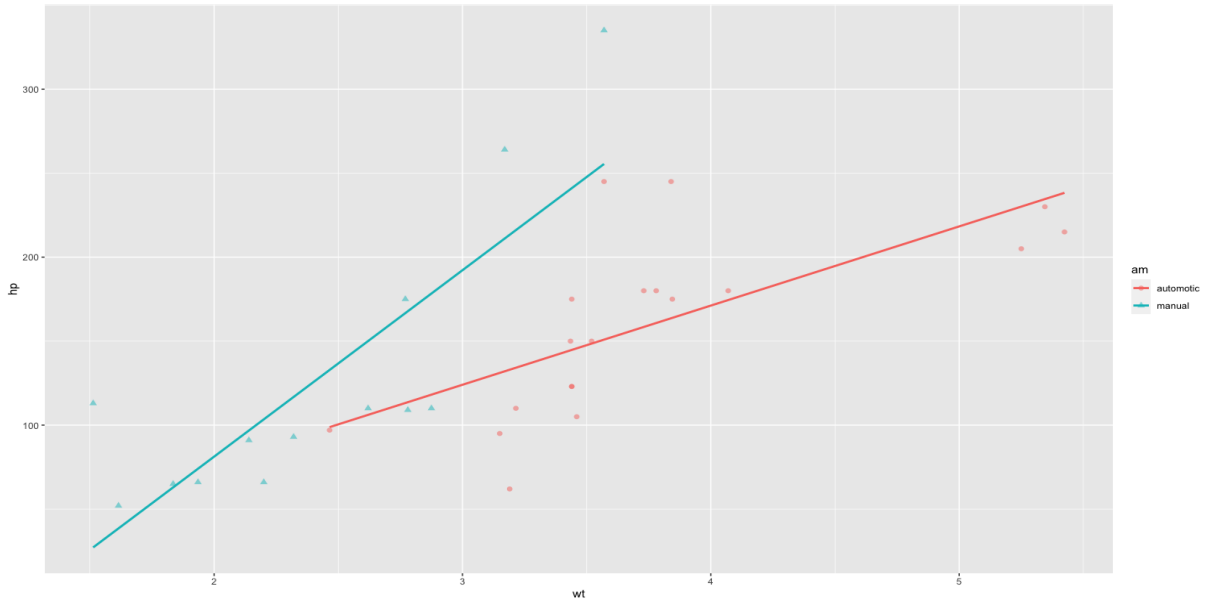
```
> ggplot(data = mtcars, aes(x = wt, y = hp, col = am)) + geom_point() + geom_smooth() +
+     coord_cartesian(xlim = c(3, 6))
```



relient les points par des ligne en utilisant la fonction $geom_point()$

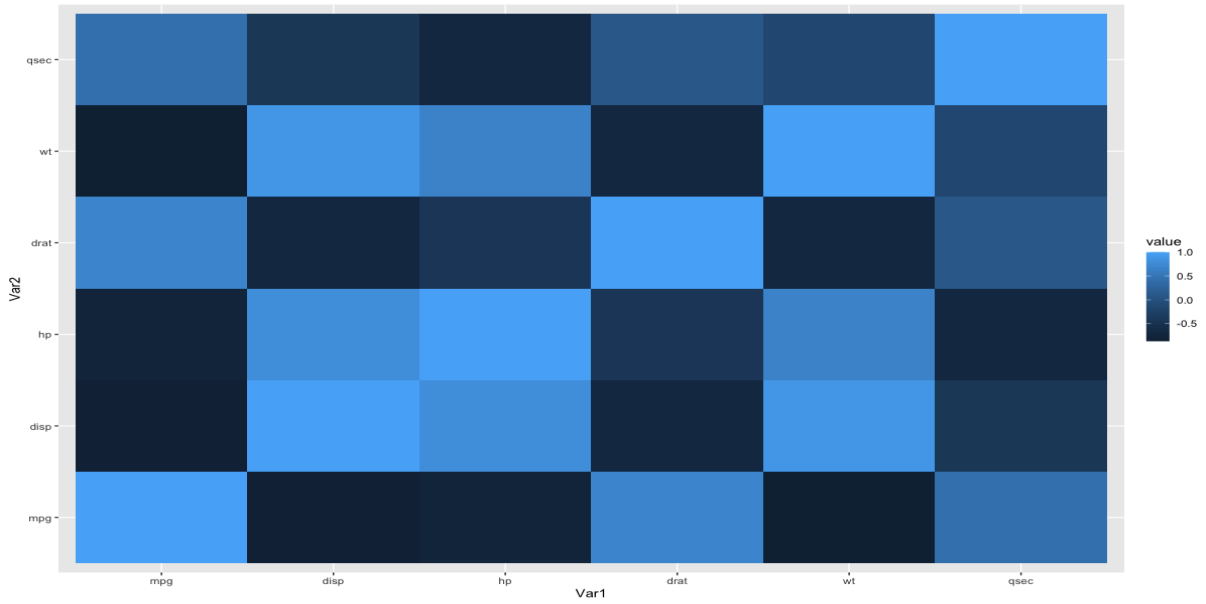



```
> ggplot(aes(x = wt, y = hp, colour = am), data = df) +
+     geom_point(aes(shape = am),size = 2, alpha=0.5) +
+     geom_smooth(method = "lm", se = FALSE)
```



En ajoutant la fonction avec $facet\_grid(vs \sim cyl)$, elle vas nous donnée une matrice de panneaux définie par des variables de facettes en ligne (Number of cylinders) et en colonne (Engine (0 = V-shaped, 1 = straight))
Nous visualisons une matrice de corrélation sur l'ensemble de données mtcars à l'aide du package ggplot et reshape2 en utilisant la fonction melt

```
> df1 <- mtcars[, c(1,3,4,5,6,7)]
> cormat <- round(cor(df1),2)
> library(reshape2)
> melted_cormat <- melt(cormat)
> ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +  geom_tile()
```



Les tables 5.3, 5.4 présentent les fonctions peuvent être utilisées pour créer une variété de représentations graphiques.

Fonction Geom	**Description**
geom_point()	Nuage de points
geom_line()	Ligne (ordonnées selon *vec*)
geom_abline()	Droite
geom_path()	Ligne (ordre original)
geom_text()	Texte
geom_rect()	Rectangle
geom_polygon()	Polygone
geom_segment()	Segment
geom_bar()	Diagramme en barres
geom_histogram()	Histogramme
geom_boxplot()	Boîtes à moustaches
geom_density()	Densité
geom_contour()	Lignes de contour
geom_smooth()	Lissage

TABLE 5.3 – Liste des fonctions geom

Fonction Stat	**Description**
stat_identity()	Aucune transformation
stat_bin()	Comptage
stat_density()	Densité
stat_smooth()	Lissage
stat_boxplot()	Boxplot

TABLE 5.4 – Liste des fonctions statistics

Chapitre 6

La statistique inférentielle

6.1 échantillonnage

nous allons explorer les statistiques inférentielles. Nous commencerons avec la distribution d'échantillonnage, et continuerons avec le théorème de la limite centrale, l'intervalle de confiance et le test d'hypothèse.

Nous allons utiliser un cadre de données appelé Ames. Il s'agit de données immobilières de la ville d'Ames, Iowa, USA. Les détails de chaque transaction immobilière à Ames sont enregistrés par le bureau de l'assesseur de la ville.

```
> library(dplyr) #Package pour la manipulation des données
> df <- read.csv("~/Desktop/data.csv")
> glimpse(df)
## Rows: 2,930
## Columns: 82
## \$ Order           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, ~
## \$ PID             <int> 526301100, 526350040, 526351010, 526353~
## ...
## ...
```

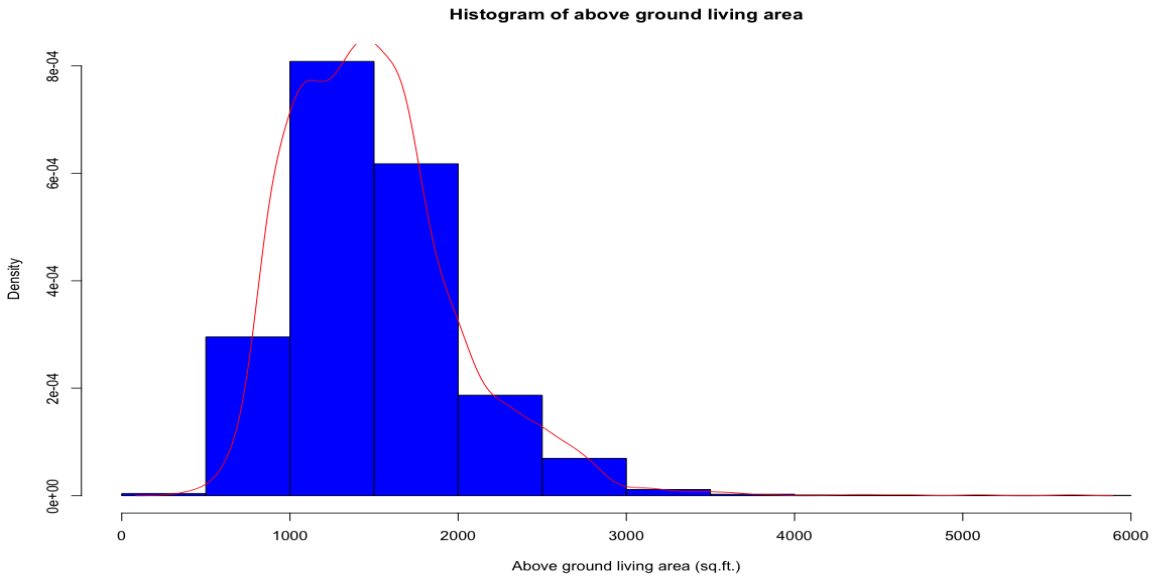
Nous limiterons notre attention à seulement deux des variables pour faire une analyse très approfondie : la superficie de la maison en pieds carrés (Gr.Liv.Area) et le prix de vente (SalePrice) en dollars.

```
> area <- df\$Gr.Liv.Area
> price <- df\$SalePrice
> area.sd <- sqrt(sum((area - mean(area))^2)/(2930)) #Écart type de la population
[1] 505.4226
> summary(area)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    334    1126    1442    1500    1743    5642
```

La distribution de la variable area a une moyenne de 1499,69 pieds carrés, une médiane

de 1442 pieds carrés et un écart type de population de 505,42 pieds carrés, comme on peut le voir. L'ensemble de données contient des observations dont la taille minimale est de 334 pieds carrés et la taille maximale de 5642 pieds carrés.

```
> hist(area,
+      main = "Histogram of above ground living area",
+      xlab = "Above ground living area (sq.ft.)",
+      col="blue",
+      prob=TRUE)
> lines(density(area),col="red")
```



L'histogramme de la variable (area) montre une asymétrie positive (droite) des données de la population.

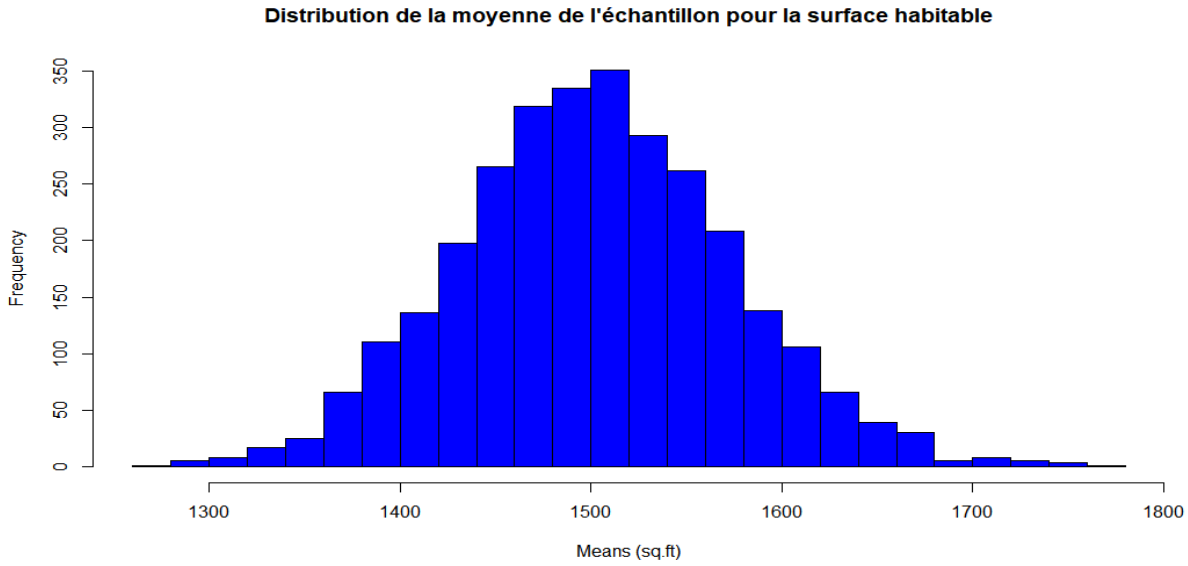
Il est souvent très coûteux ou difficile de collecter des données sur une population entière. Par conséquent, nous utilisons fréquemment un échantillon de la population pour comprendre les caractéristiques de cette dernière.

```
> sam <- sample(area, 50) #échantillon aléatoire de 50 observations d'area
> mean(sam)
[1] 1546.96
```

Nous obtenons une moyenne d'échantillon différente chaque fois que nous prenons un nouveau échantillon aléatoire. Il est utile de connaître le degré de variabilité à prévoir lorsque l'on mesure la moyenne de la population de cette manière. Nous pouvons comprendre cette variabilité en examinant la distribution d'échantillonnage, qui est la distribution des moyennes d'échantillon. Comme nous avons accès à la population dans ce laboratoire, nous pouvons effectuer les processus susmentionnés de manière répétée afin de construire la distribution d'échantillonnage de la moyenne de l'échantillon. La population sera divisée en 3000 échantillons, chacun de taille 50, et nous prendrons la moyenne de chaque

échantillon et stockerons les résultats dans un vecteur appelé sam50. L'histogramme de cette distribution d'échantillonnage sera ensuite tracé.

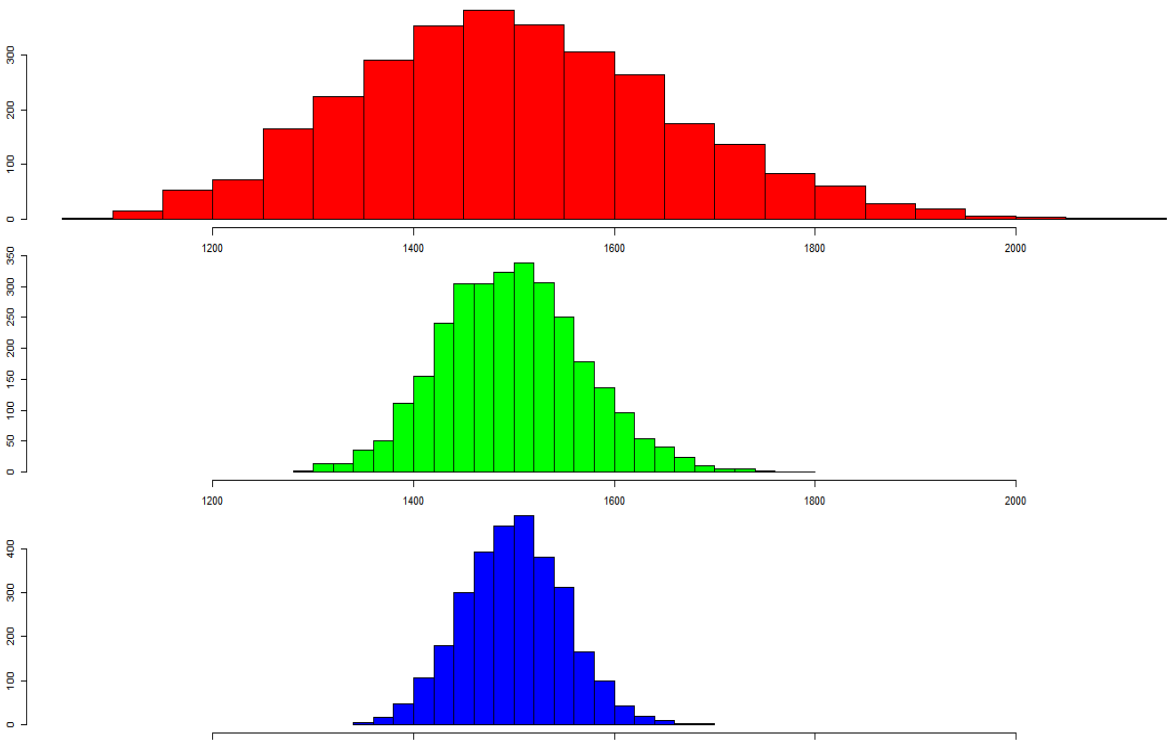
```
> sam50 <- rep(NA, 3000)
> for(i in 1:3000){samp <- sample(area, 50)
+ sam50[i] <- mean(samp)}
> hist(sam50, breaks = 25,
+      main = "Distribution de la moyenne de l'échantillon pour la surface habitable",
+      xlab = "Means (sq.ft)",
+      col="blue")
```



La distribution d'échantillonnage que nous avons calculée nous permet de déduire beaucoup de choses sur l'estimation de la surface habitable typique des logements d'Ames. La distribution d'échantillonnage est centrée sur la surface habitable moyenne réelle de la population, car la moyenne de l'échantillon est un estimateur sans biais, et l'étendue de la distribution montre la variabilité introduite par l'échantillonnage de seulement 50 ventes de logements. Construisons deux autres distributions d'échantillonnage, l'une basée sur un échantillon de 10 personnes et l'autre sur un échantillon de 100 personnes provenant d'une population de 3000 personnes, afin de mieux comprendre l'impact de la taille de l'échantillon sur notre distribution. Traçons les trois distributions l'une au-dessus (en utilisant la commande par(mfrow = c(3, 1)), en fixant le nombre de bars breaks en 21 bar, et aussi en utilisant meme l'étendue de l'axe des x par xlim) de l'autre pour illustrer comment le changement de taille d'échantillon affecte la distribution d'échantillonnage.

```
> area <- df\$Gr.Liv.Area
> samp10 <- rep(NA, 3000); samp50 <- rep(NA, 3000); samp100 <- rep(NA, 3000)
> for(i in 1:3000){
+ samp <- sample(area, 10)
+ samp10[i] <- mean(samp)
```

```
+ samp <- sample(area, 50)
+ samp50[i] <- mean(samp)
+ samp <- sample(area, 100)
+ samp100[i] <- mean(samp)}
> par(mfrow = c(3, 1))
> xlimits <- range(samp10)
> hist(samp10,  breaks = 21, xlim = xlimits, col="red")
> hist(samp50,  breaks = 21, xlim = xlimits, col="green")
> hist(samp100, breaks = 21, xlim = xlimits, col="blue")
```



Le centre n'est pas affecté par la taille de l'échantillon. La moyenne des moyennes de l'échantillon est toujours approximativement la même que la moyenne de la population . L'écart est plus faible pour les échantillons plus grands, donc l'écart type des moyennes de l'échantillon diminue lorsque la taille de l'échantillon augmente.

6.2 Test d'hypothèse

Nous utiliserons un vaste ensemble de données contenant des informations sur les naissances enregistrées dans l'État de Caroline du Nord (2004), aux États-Unis.

```
df <- read.csv("~/Desktop/data.csv")
> glimpse(df)
Rows: 1,000
Columns: 13
\$ fage          <int> NA, NA, 19, 21, NA, NA, 18, 17, NA, 20, 30, NA, NA,~
```

```
$ mage           <int> 13, 14, 15, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16,~
$ mature         <chr> "younger mom", "younger mom", "younger mom", "young~
$ weeks          <int> 39, 42, 37, 41, 39, 38, 37, 35, 38, 37, 45, 42, 40,~
$ premie         <chr> "full term", "full term", "full term", "full term",~
$ visits         <int> 10, 15, 11, 6, 9, 19, 12, 5, 9, 13, 9, 8, 4, 12, 15~
$ marital        <chr> "married", "married", "married", "married", "marrie~
$ gained         <int> 38, 20, 38, 34, 27, 22, 76, 15, NA, 52, 28, 34, 12,~
$ weight         <dbl> 7.63, 7.88, 6.63, 8.00, 6.38, 5.38, 8.44, 4.69, 8.8~
$ lowbirthweight <chr> "not low", "not low", "not low", "not low", "not lo~
$ gender         <chr> "male", "male", "female", "male", "female", "male",~
$ habit          <chr> "nonsmoker", "nonsmoker", "nonsmoker", "nonsmoker",~
$ whitemom       <chr> "not white", "not white", "white", "white", "not wh~
```

En utilisant la fonction symmary() Nous essayons de déterminer quelles variables sont catégoriques et quelles variables sont d'échelle, s'il y a des valeurs aberrantes pour les variables d'échelle, ainsi que les enregistrements NA pour chaque variable.

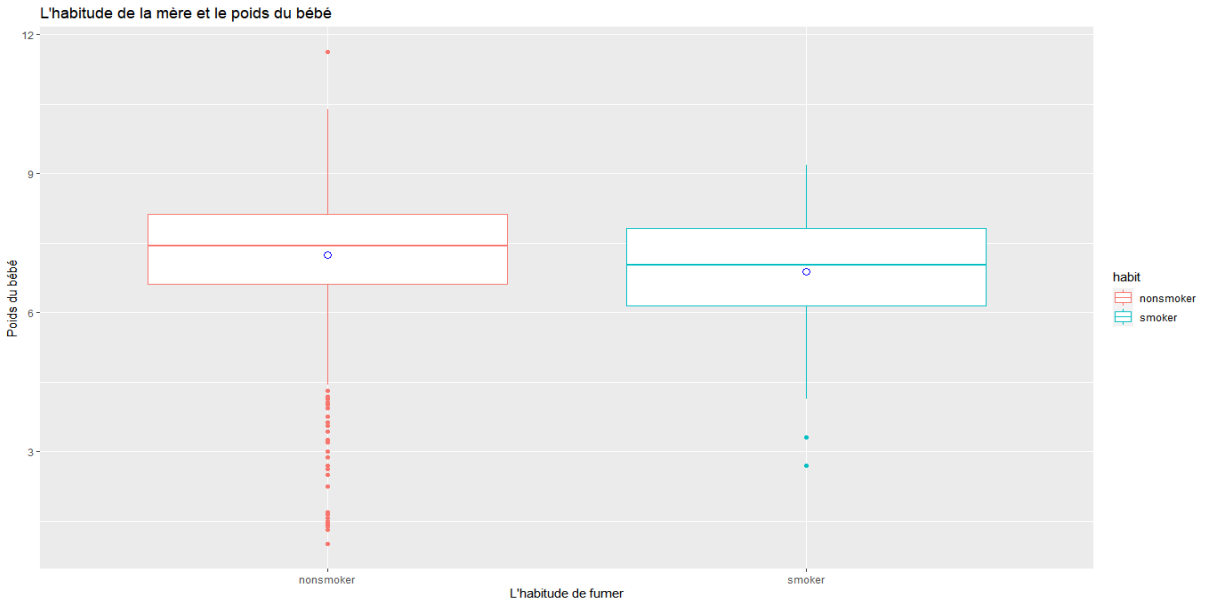
```
> summary(df)
      fage            mage       mature              weeks
 Min.   :14.00   Min.   :13   Length:1000        Min.   :20.00
 1st Qu.:25.00   1st Qu.:22   Class :character   1st Qu.:37.00
 Median :30.00   Median :27   Mode  :character   Median :39.00
 Mean   :30.26   Mean   :27                      Mean   :38.33
 3rd Qu.:35.00   3rd Qu.:32                      3rd Qu.:40.00
 Max.   :55.00   Max.   :50                      Max.   :45.00
 NA's   :171                                     NA's   :2
    premie              visits       marital              gained
 Length:1000        Min.   : 0.0   Length:1000        Min.   : 0.00
 Class :character   1st Qu.:10.0   Class :character   1st Qu.:20.00
 Mode  :character   Median :12.0   Mode  :character   Median :30.00
                    Mean   :12.1                      Mean   :30.33
                    3rd Qu.:15.0                      3rd Qu.:38.00
                    Max.   :30.0                      Max.   :85.00
                    NA's   :9                         NA's   :27
     weight       lowbirthweight        gender             habit
 Min.   : 1.000   Length:1000        Length:1000        Length:1000
 1st Qu.: 6.380   Class :character   Class :character   Class :character
 Median : 7.310   Mode  :character   Mode  :character   Mode  :character
 Mean   : 7.101
 3rd Qu.: 8.060
 Max.   :11.750

   whitemom
 Length:1000
 Class :character
 Mode  :character
```

Par exemple, nous considérons la relation possible entre l'habitude de fumer d'une mère

et le poids de son bébé pour les fumeuses et les non-fumeuses. En général, les observations avec des valeurs manquantes doivent être examinées, éventuellement éliminées, ou traitées (nous allons juste l'ignorer). Les données doivent être tracées dans un premier temps car cela nous permet de voir facilement les tendances, la distribution, et développer des questions de recherche.

Faisons donc un boxplot côte à côte de la variable habit et de la variable weight en utilisant la fonction ggplot.

```
> ggplot(data = na.omit(df), aes(x = habit,
+ y = weight, colour = habit)) + geom_boxplot() +
+ xlab("L'habitude de fumer") + ylab("Poids du bébé") +
+ ggtitle("L'habitude de la mère et le poids du bébé") +
+ stat_summary(fun = mean, colour = "blue", geom = "point", #for means comparing
+ shape = 1, size = 3)
```



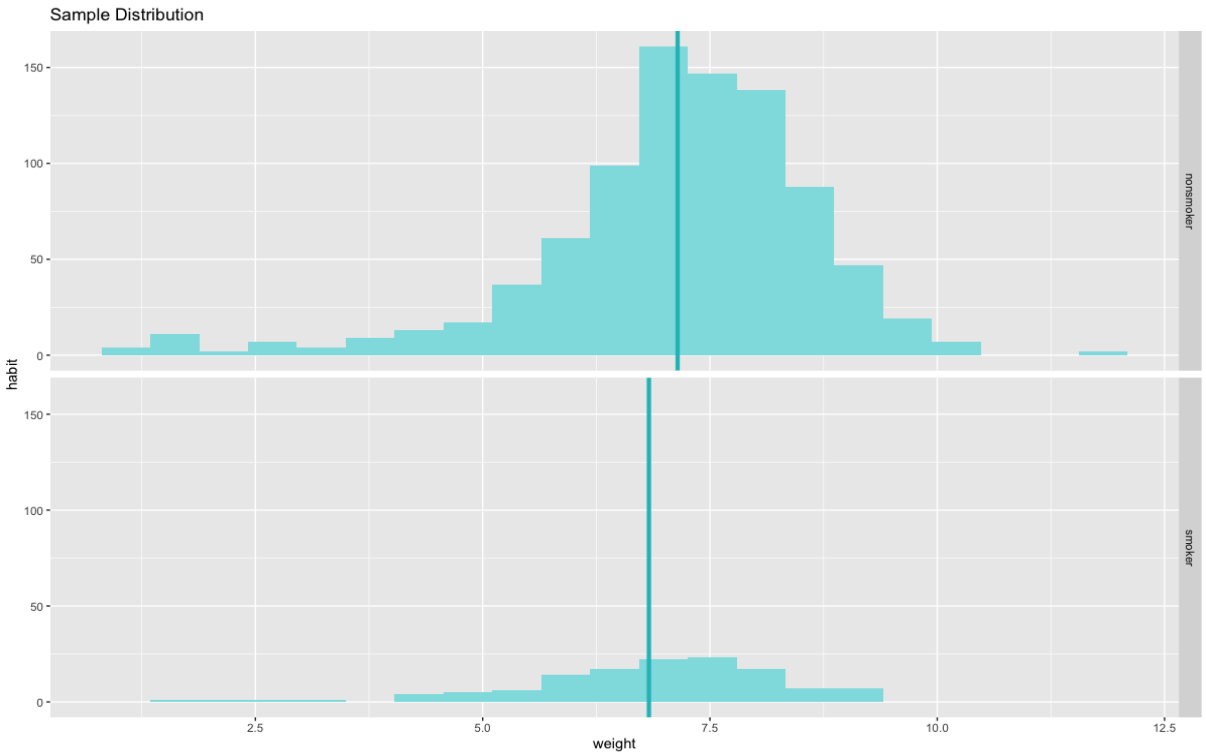
Les diagrammes en boîte montrent comment les médianes des deux distributions se comparent, mais en ajoutant les moyennes sur le diagramme en boîte à l'aide de la fonction stat_summary(), nous pouvons également comparer les moyennes des distributions. Bien qu'il y ait quelques différences de moyennes, Mais la question qui se pose ; cette variation est-elle statistiquement significative ? pour la répondre, Nous allons effectuer un test d'hypothèse.

Avant de procéder à un test d'hypothèse, il est important de vérifier si les conditions préalables à l'inférence sont remplies. Pour valider les conditions et vérifier les valeurs aberrantes, vous devez obtenir la taille des échantillons. La taille du groupe peut être calculée à l'aide de la formule by(nc$weight$, nchabit, length).

L'étape suivante consiste à développer les hypothèses pour tester si les poids moyens à la naissance des enfants nés de fumeurs et de non-fumeurs diffèrent. Nous utiliserons une fonction appelée inference() pour effectuer les tests. Nous pouvons effectuer des tests

d'hypothèses et créer des intervalles de confiance en utilisant cette méthode. Afin d'utiliser la fonction inference(), il faut installer le package statsr par la fameuse commande install.packages()

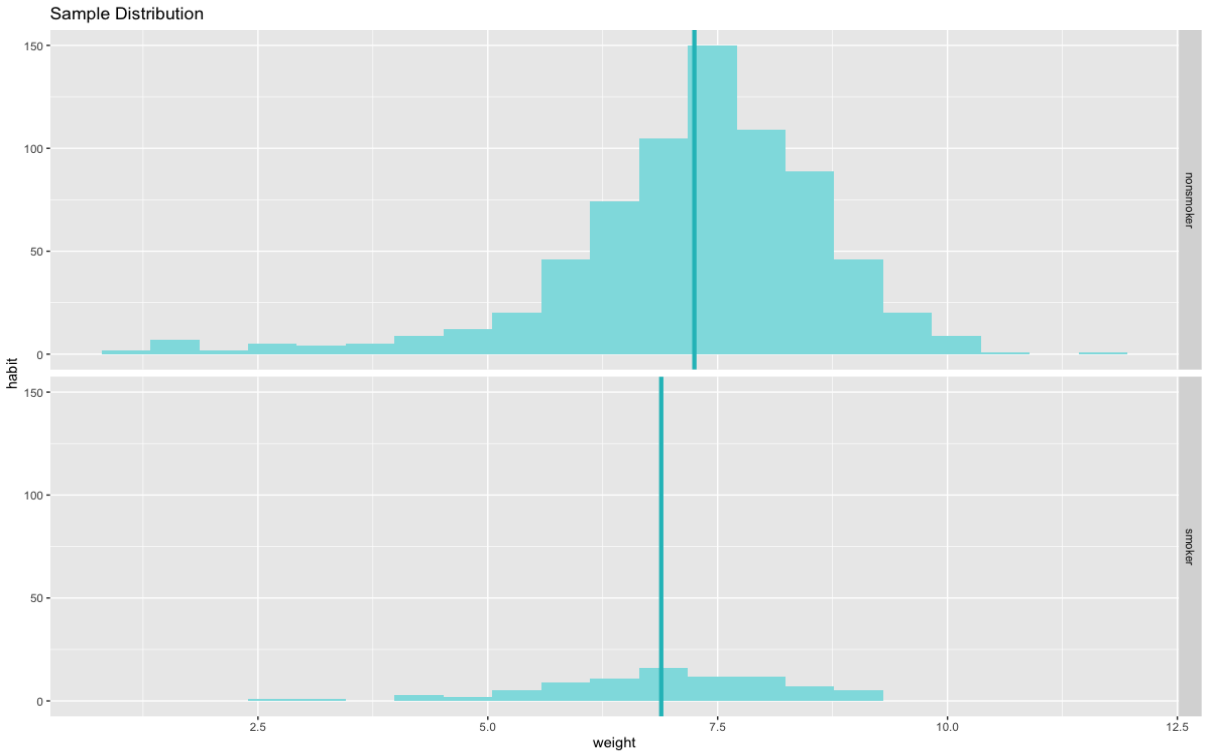
```
> install.packages("statsr")
> inference(y = weight, x = habit, data = df, # le poids est DV et l'habitude est IDV
+           statistic = c("mean"), #paramètre à estimer est la moyenne
+           type = c("ci"), # Intervalle de confiance
+           null = 0, #l'hypothèse nulle: les deux moyennes sont égales.
+           alternative = c("twosided"), #direction de l'hypothèse alternative
+           method = c("theoretical"), #Théorique parce que nous avons les données
+           conf_level = 0.95,
+           order = c("smoker","nonsmoker"))
Response variable: numerical, Explanatory variable: categorical (2 levels)
n_smoker = 126, y_bar_smoker = 6.8287, s_smoker = 1.3862
n_nonsmoker = 873, y_bar_nonsmoker = 7.1443, s_nonsmoker = 1.5187
95% CI (smoker - nonsmoker): (-0.5803 , -0.0508)
```



Maintenant, omettons les valeurs manquantes de nos calculs, et executons les même lignes précédentes

```
> dfomit = na.omit(df)
> inference(y = weight, x = habit, data = df, # le poids est DV et l'habitude est IDV
+           statistic = c("mean"), #paramètre à estimer est la moyenne
+           type = c("ci"), # Intervalle de confiance
```

```
+           null = 0, #l'hypothèse nulle: les deux moyennes sont egaux.
+           alternative = c("twosided"), #direction de l'hypothèse alternative
+           method = c("theoretical"), #Théorique parce que nous avons les données
+           conf_level = 0.95,
+           order = c("smoker","nonsmoker"))
Response variable: numerical, Explanatory variable: categorical (2 levels)
n_smoker = 84, y_bar_smoker = 6.8864, s_smoker = 1.3064
n_nonsmoker = 716, y_bar_nonsmoker = 7.2468, s_nonsmoker = 1.4521
95% CI (smoker - nonsmoker): (-0.6637 , -0.057)
```

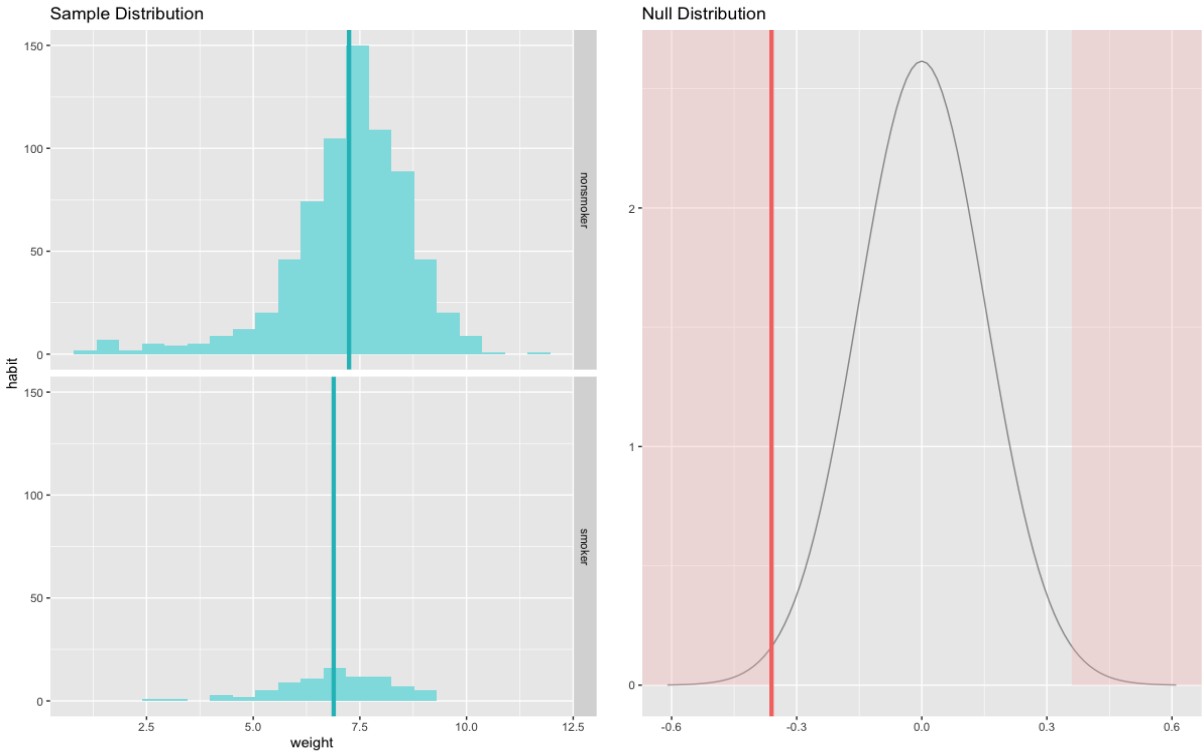


Dans les deux cas (avec ou sans valeurs manquantes), nous pouvons voir sur ces deux graphiques qu'il y a quelques variations dans la façon de mesurer la tendance centrale et l'écart. L'intervalles de confiance pour la différence entre les moyennes des fumeurs et des non-fumeurs sont également fournis dans le résultat. Nous pouvons affirmer que la différence est statistiquement significative et différente de 0 car l'intervalle de confiance exclut la valeur nulle (H0 : la difference entre moyens est 0). Pour le déterminer, nous pouvons exécuter un test d'hypothèse dans cet exemple, on utilise un t-test, pour cela, il suffit de spécifier le type "ht" dans la paramétre type comme suivante :

```
> dfomit = na.omit(df)
> inference(y = weight, x = habit, data = df, # le poids est DV et l'habitude est IDV
+           statistic = c("mean"), #paramètre à estimer est la moyenne
+           type = c("ci"), # Intervalle de confiance
+           null = 0, #l'hypothèse nulle: les deux moyennes sont égales.
+           alternative = c("twosided"), #direction de l'hypothèse alternative
+           method = c("theoretical"), #Théorique parce que nous avons les données
```

```
+           conf_level = 0.95,
+           order = c("smoker","nonsmoker"))
Response variable: numerical
Explanatory variable: categorical (2 levels)
n_smoker = 84, y_bar_smoker = 6.8864, s_smoker = 1.3064
n_nonsmoker = 716, y_bar_nonsmoker = 7.2468, s_nonsmoker = 1.4521
H0: mu_smoker =  mu_nonsmoker
HA: mu_smoker != mu_nonsmoker
t = -2.3625, df = 83
p_value = 0.0205
```

on passe directement vers La valeur t = -2,36 et la valeur p-value =0.0205 inférieure à alpha = 0,05 donc le test est significatif (nous rejetons l'hypthèse nulle), nous pouvons donc conclure au même résultat concernant la signification de la différence entre les moyennes.



6.3 tests statistiques

Nous rappelons encore une fois que le caractère qualitatif ou quantitatif des deux variables détermine le type d'analyse,du test, et de visualisation. Prenons le cadre de données (sans valeurs manquantes NA) utilié dans la partie 6.2 .

6.3.1 Deux variables qualitatives

Un tableau croisé est créé lorsque deux variables qualitatives (Situation et l'habitude de fumer) doivent être comparées. Un tri à plat est réalisé à l'aide de la fonction table.

```
> pairchi2 <- table(dfomit$marital,dfomit$habit)
> pairchi2

               nonsmoker smoker
  married            193     29
  not married        523     55
```

Afin de compléter le tableau croisé par un test d'indépendance $\mathcal{X}^2$. L'indépendance de la variable "Situation" et "l'habitude de fumer" du tableau peut être testée avec ce test et finalement rejetée. Pour effectuer un test de ce type, nous appliquons la fonction chisq.test à la table pairch2.

```
> chisq.test(pairchi2)

        Pearson's Chi-squared test with Yates' continuity correction

data:  pairchi2
X-squared = 1.787, df = 1, p-value = 0.1813
```

Dans ce cas, la valeur p-value égale à 0.1813 $>$ 0.05 (seuil de décision), nous on peut facilement retenir l'yhpothese nulle, c'est à dire les deux variavles Situation" et "l'habitude de fumer sont indépendantes.

6.3.2 Une variable quantitative et une variable qualitative

On a vue de la partie 6.2 comment on effectue un test de t-test en utilisant la fonction inference(). Prenons une autre fonction t.test,Ainsi, en utilisant la procédure suivante, nous pouvons tester l'idée que les âges moyens sont égaux pour les variables "fage" et "mature"

```
> t.test(dfomit$fage ~ dfomit$mature)

        Welch Two Sample t-test

data:  dfomit$fage by dfomit$mature
t = 18.681, df = 184.16, p-value < 2.2e-16
alternative hypothesis: true difference in means between group mature mom and group younger mom is not equ
95 percent confidence interval:
  8.440964 10.434428
sample estimates:
 mean in group mature mom mean in group younger mom
                 38.28814                   28.85044
```

Nous pouvons rejeter l'hypothèse nulle selon laquelle les moyennes des deux groupes sont égales, car le résultat du t-test est significatif et a une valeur p-value très faible (p-value $<$ 2.2e-16 « 0.05). Le test fournit même un intervalle de confiance de 95% pour la valeur de la différence entre les deux moyennes (CI=[8.440964,10.434428]). Remarque : nous avons

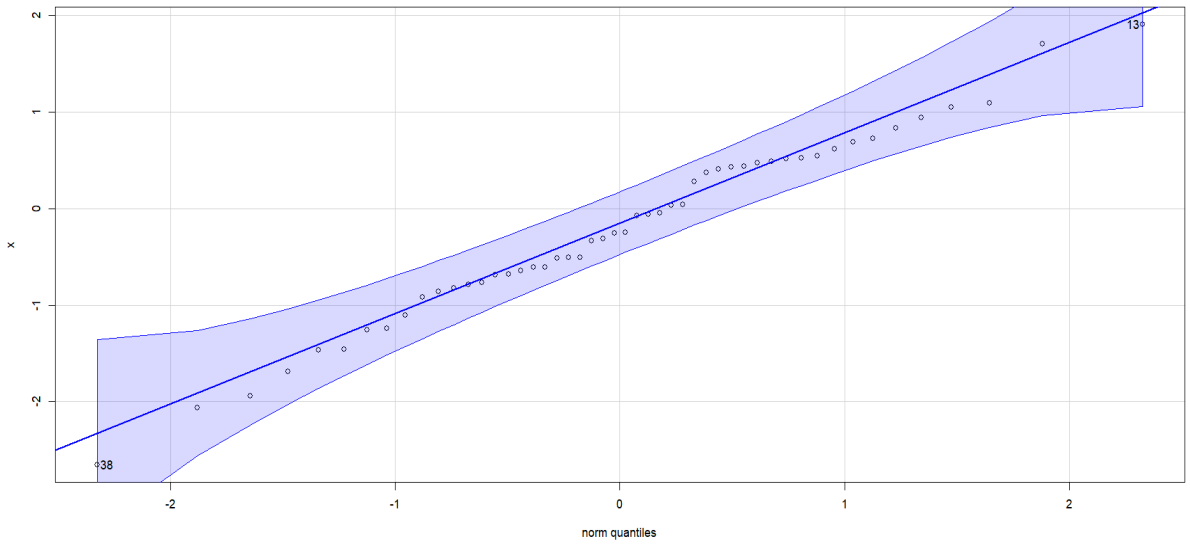
effectué le test en supposant que la normalité de la distribution est vérifiée pour les deux groupes (fage).

Supposons que nous ayons le cas, en testant l'hypothèse d'indépendance entre deux groupes (moyenne des visites selon l'habitude du fumeur). le test Shapiro-Wilk teste la normalié d'une distribution par la fonction shapiro.test()

```
> Smoker <-  dfomit[dfomit\$habit=="smoker",]
> Nonsmoker <-  dfomit[dfomit\$habit=="nonsmoker",]
> shapiro.test(Smoker\$visits)

        Shapiro-Wilk normality test

data:  Smoker\$visits
W = 0.9269, p-value = 0.0001406

> shapiro.test(Nonsmoker\$visits)

        Shapiro-Wilk normality test

data:  Nonsmoker\$visits
W = 0.95226, p-value = 1.828e-14
```

Le test est significatif dans les deux cas (p-value = 0.0001406, p-value = 1.828e-14) et réfute la normalité supposée des deux distributions.
On peut effectivement tester la normalité d'une distribution graphiquement par la fonction qqPlot() du package car,

```
> library("car")
> x <- rnorm(50,0,1)
> qqPlot(x)
```



La fonction Trace les résidus studentized d'un modèle linéaire, par rapport aux quantiles théoriques d'une distribution de comparaison. En cas des données sont normales, les points doivent former une ligne droite. Comme tous les points tombent approximativement le long de cette ligne de référence, nous pouvons supposer la normalité de cette série
On peut tester l'hypothèse d'égalité des visites moyennes selon l'habitude de fumer en utilisant un test non paramétrique appelé wilcox.test()

```
> wilcox.test(dfomit$visits~dfomit$habit)

        Wilcoxon rank sum test with continuity correction

data:  dfomit\$visits by dfomit\$habit
W = 31526, p-value = 0.4655
alternative hypothesis: true location shift is not equal to 0
```

Puique le test nous donne la valeur de p-value 0.4655 plus grand que 0.05, le test n'est pas significatif. On peut retenir l'hypothèse d'indépendance et considérer que les distributions des visites dans les deux groupes sont égales.

6.3.3 Deux variables quantitatives

Le lien linéaire entre deux variables quantitatives est mesuré par la corrélation. Sa valeur varie de -1 à 1. Il y a un lien linéaire négatif parfait si la corrélation est égale à 1. Il s'agit d'un lien linéaire positif parfait si la valeur est de 1. Il n'y a pas de relation linéaire entre les variables si elle est de 0. Afin de mesurer le degré d'association de deux variables quantitatives, on fait l'appele de la fonction cor()

```
> cor(dfomit\$fage,dfomit\$weight)
[1] 0.07333513
```

le coefficient de correlation de pearson égale à 0.07333513 très petit, Il y a pas donc un lien linéaire entre les deux variables "fage" et "weight".

Chapitre 7

Les modèles de régressions

La régression comprend plusieurs méthodes d'analyse statistique qui vous permettent d'approximer une variable en fonction d'autres variables qui lui sont corrélées. Les problèmes de régression se distinguent des problèmes de classification (En apprentissage automatique). Par conséquent, nous considérons le problème de la prédiction des variables quantitatives comme un problème de régression et le problème de la prédiction des variables qualitatives comme un problème de classification. Certaines méthodes comme La régression logistique est à la fois une méthode de régression au sens de prédire la probabilité d'appartenir à chaque classe et aussi une méthode de classification.

7.1 La régression linéaire simple

La régression linéaire simple est une méthode statistique pour trouver une relation linéaire entre la variable explicative X et la variable expliquere Y. Ce modèle consiste à traiter Y comme une fonction affine de X. Alors, le but de la régression linéaire est de trouver une ligne qui correspond au nuage de points de Y en fonction de X.

7.1.1 Modélisation mathématique

L'ajustement affine de Y par X nous informe que Y peut s'écrire comme équation d'une droite :

$$Y = \alpha X + \beta + \epsilon \tag{7.1}$$

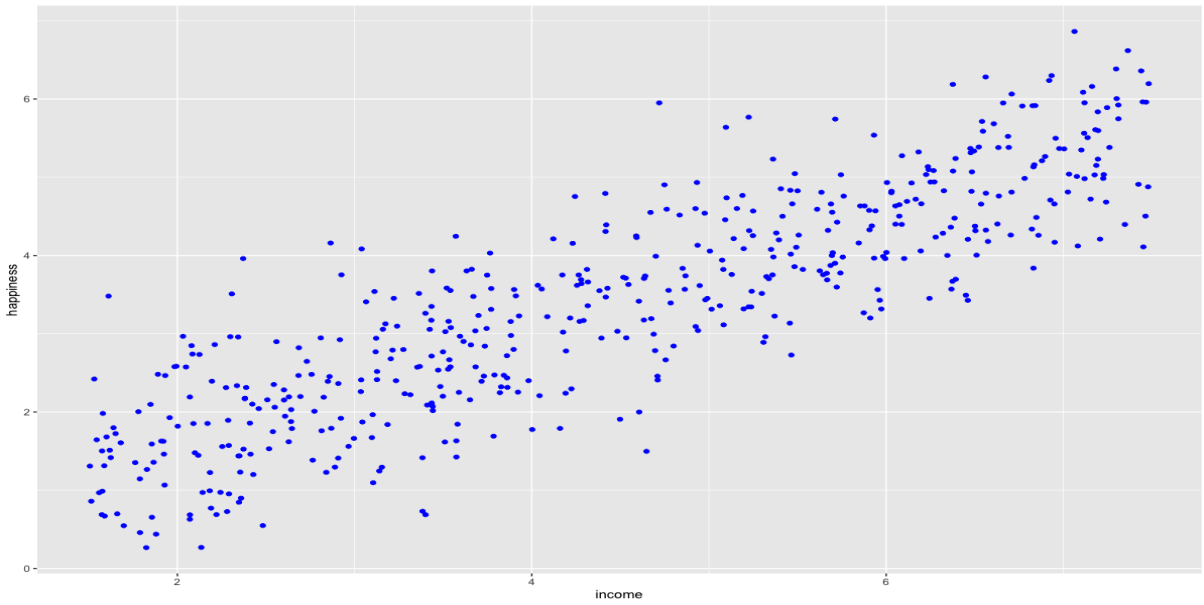
- Y : variable à expliquer (variable dépendante ou variable endogène)
- X : variable explicative (variable indépendante ou variable exogène)
- α : l'ordonnée à l'origine
- β : la pente de la droite
- ϵ : l'erreur $\sim \mathcal{N}(0, \sigma^2)$

7.1.2 Le modèle

Nous utiliserons les données de income, et nous allons prendre comme variable explicative X les revenus des individus "income" et le niveau de bonheur "happiness" comme variable à expliquer Y. La première étape c'est la représentation graphique du nuage de points.

En effet si le nuage ne s'apparente pas à une droite, la régression linéaire ne sera pas le meilleur modèle pour notre cas

```
> df <- read.csv("~/Desktop/income.csv")
> ggplot(data = df,aes(x =income, y = happiness)) + geom_point(col="blue")
```



Le graphique du nuage montre que la régression linéaire sera le meilleur modèle pour notre jeu de données. La régression linéaire simple tente de trouver la meilleure ligne pour prédire le niveau de bonheur sur la base du revenu des individus.

La fonction lm() peut être utilisée pour déterminer les coefficients du modèle linéaire (eq 7.1)

```
> SimRegModel <- lm(formula = happiness~income, data = df)
> summary(SimRegModel) #Le résumé statistique du modèle

Call:
lm(formula = happiness ~ income, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-2.02479 -0.48526  0.04078  0.45898  2.37805

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.20427    0.08884   2.299   0.0219 *
income       0.71383    0.01854  38.505   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```
Residual standard error: 0.7181 on 496 degrees of freedom
Multiple R-squared:  0.7493,        Adjusted R-squared:  0.7488
F-statistic:  1483 on 1 and 496 DF,  p-value: < 2.2e-16
```

Nous devons confirmer que ce modèle est statistiquement significatif avant de l'utiliser pour prédire le "revenu", c'est-à-dire :

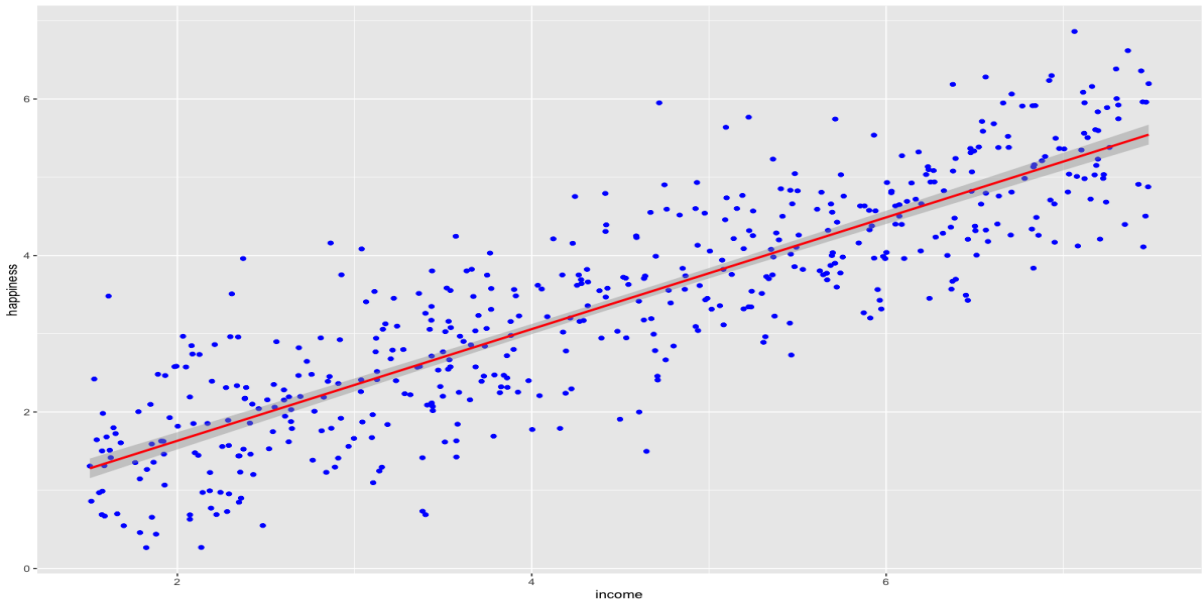
- Les variables dépendantes et indépendantes ont une relation statistiquement significative entre elles.
- Le modèle que nous avons créé correspond étroitement aux faits observés.
- Examiner l'erreur standard des résidus (RSE), la valeur R^2 et les F-statistiques pour déterminer dans quelle mesure le modèle s'adapte aux données.

dans le tableau des coefficients, les valeurs p-value pour l'ordonnée à l'origine et la variable prédicteur sont toutes deux hautement significatives ($p - value_\beta = 0.0219 < 0.05$, p_value_α=2e-16 $< 0.001 < 0.05$), nous pouvons donc rejeter l'hypothèse nulle et accepter l'hypothèse alternative, ce qui signifie qu'il existe une association significative entre la variable explicative et le variable à expliquer. Ainsi, notre modèle de régression simple peut être écrit comme une fonction du revenu

$$happiness = 0.71383 \times income + 0.20427 \tag{7.2}$$

Traçons la droite de régression avec la fonction geom_smooth()

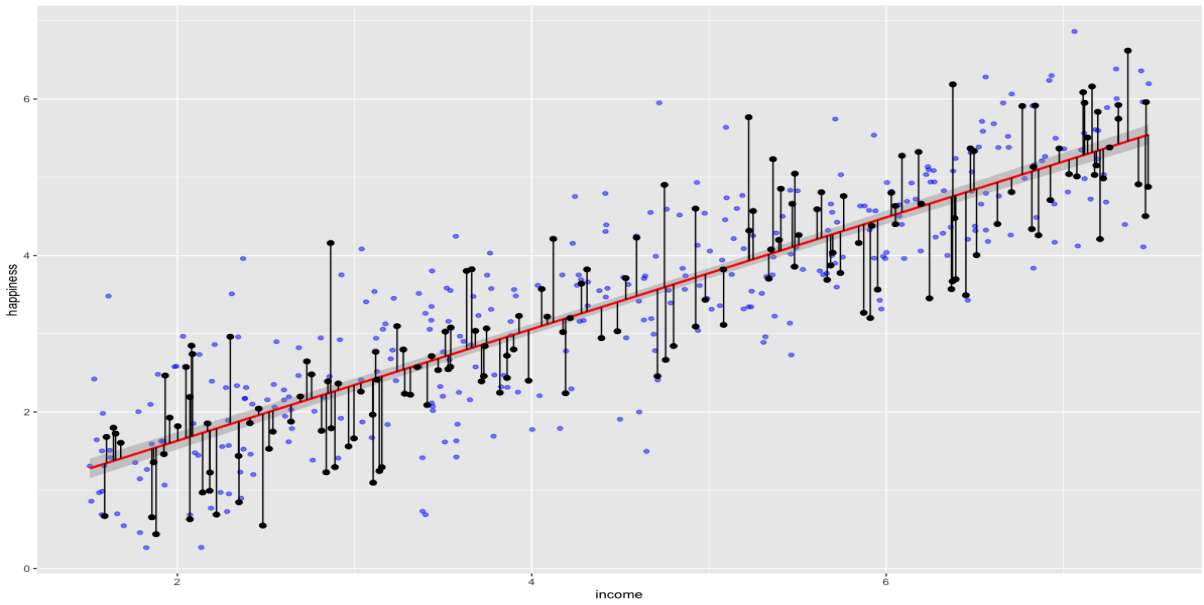
```
> ggplot(data = df,aes(x =income, y = happiness)) +
+     geom_point(col="blue") +
+     geom_smooth(method = "lm", col="red")
```



De plus on a Erreur standard résiduelle : l'estimation σ de ϵ qui est de 0.7181.

On visualise les résidus de 33% du cadre de donnée

```
> X <- df\$income
> samp = sample(1:498)[1:164] [1:164] #Indices de notre echantillon
> X_residuals = X[samp]
> y = df\$happiness[samp]
> y_hat = SimRegModel\$fitted.values[samp]
> ggplot() + geom_point(data = df,aes(x =income, y = happiness),alpha =0.5, col="blue") +
+     geom_smooth(data = df ,aes(x =income, y = happiness), method = 'lm',col="red") +
+     geom_point(aes( x= X_residuals, y = y), color ='black', size = 2,) +
+     geom_segment(aes(x = X_residuals, y = y_hat, xend = X_residuals, yend = y),color = 'black')
```



Dans le tableau de précision des modèles, 0,7493 est le coefficient de détermination R^2 (est le carré du coefficient de corrélation de Pearson dans le cas de la régression simple). Dans ce cas, notre coefficient est relativement faible étant donné qu'un bon coefficient se situe autour de 0,85. La F-statistique donne la signification globale du modèle. Elle évalue si au moins une variable prédictive a un coefficient non nul. Elle devient plus importante lorsque l'on commence à utiliser des prédicteurs multiples, comme dans la régression linéaire multiple.

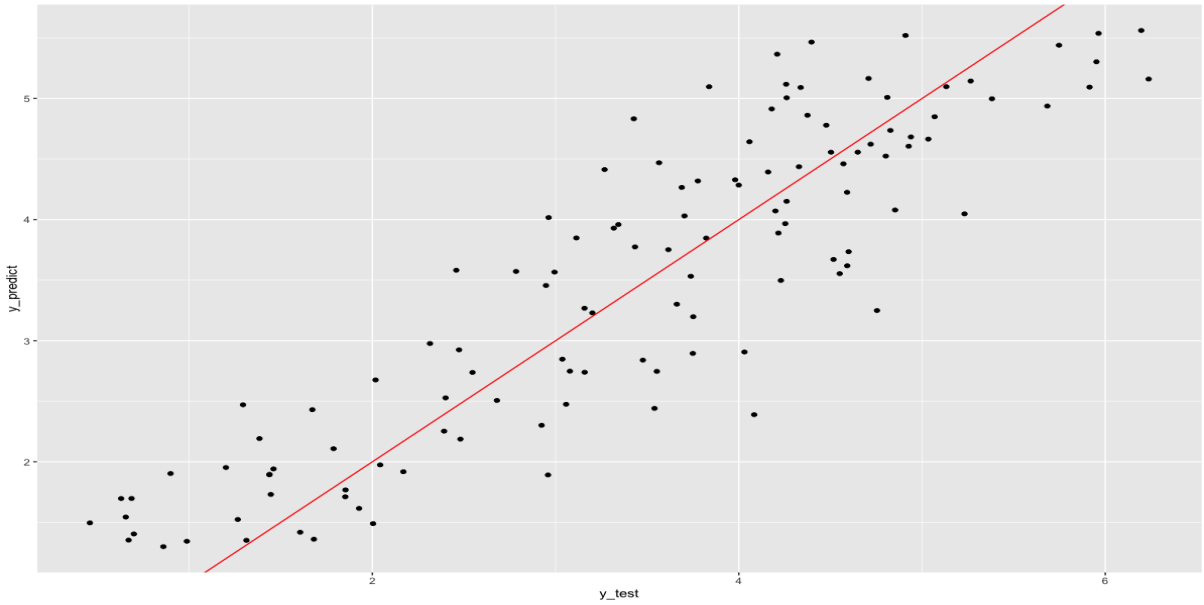
Nous utilisons la fonction predict() pour prévoir les variables que notre modèle devra expliquer.

```
> set.seed(207)
> sampT <- sample(1:498)[1:120]
> df_train <- df[-sampT,]
> df_test <- df[sampT,]
> SimRegModelT <- lm(data = df_train, formula = happiness~income)
> y_predict <- predict(SimRegModelT, data.frame(income=df_test[,2]))
>- y_predict[1:10] #Afficher les 10 valeurs de y_predict
        1         2         3         4         5         6         7         8         9        10
```

```
9  3.301276 1.697078 2.527711 3.229452 4.392723 5.165170 3.566386 1.941314 4.412933 2.839422
```

On peut visualiser la qualité de notre prédiction par le graphique du nuage

```
1  > y_test = df_test\$happiness
2  > ggplot() + geom_point(aes(x = y_test, y = y_predict)) +
3  +     geom_abline(slope = 1, color ='red')
```



7.2 la régression linéaire multiple

La régression linéaire multiple est une méthode statistique pour trouver une relation linéaire entre plusieurs (nombre fini) variables X_i et la variable à expliquer Y. Ce modèle consiste à traiter Y comme une fonction de X. Le modèle de régression linéaire multiple est une généralisation du cas simple.

7.2.1 Modélisation mathématique

L'ajustement affine de Y par X nous informe que Y peut s'écrire sour forme matricielle :

$$Y = \alpha X + \beta + \epsilon \tag{7.3}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nk} \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} \quad \epsilon = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

— Y : variable à expliquer (variable dépendante ou variable endogène)
— X : Matrice des variables explicatives (variable indépendante ou variable exogène)
— α : l'ordonnée à l'origine

— β_i : vecteur paramètres inconnus du modèle à estimer.

— ϵ_i : vecteur des erreurs ϵ_i associè à la ième obeservation $\sim \mathcal{N}(0, \sigma^2)$

7.2.2 Le modèle

L'ensemble de données cardiaques utilisé dans cette partie a été collecté à partir d'une enquête portant sur 498 villes, où des informations ont été recueillies sur le pourcentage de résidents souffrant de maladies cardiaques (heart.disease), de fumeurs (smoking) et de cyclistes (biking) dans chaque ville. Nous recherchons des éléments qui pourraient affecter les maladies cardiaques (heart.disease).

Ainsi, le modèle de régression linéaire que nous voulons construire est une fonction comme f(heart.disease)=(biking, smoking).

```
> df <- read.csv("~/Desktop/heart.csv")
> summary(df)
     biking          smoking        heart.disease
 Min.   : 1.119   Min.   : 0.5259   Min.   : 0.5519
 1st Qu.:20.205   1st Qu.: 8.2798   1st Qu.: 6.5137
 Median :35.824   Median :15.8146   Median :10.3853
 Mean   :37.788   Mean   :15.4350   Mean   :10.1745
 3rd Qu.:57.853   3rd Qu.:22.5689   3rd Qu.:13.7240
 Max.   :74.907   Max.   :29.9467   Max.   :20.4535
```

Nous utilisons le même processus que dans la section précédente et utilisons les mêmes fonctions.

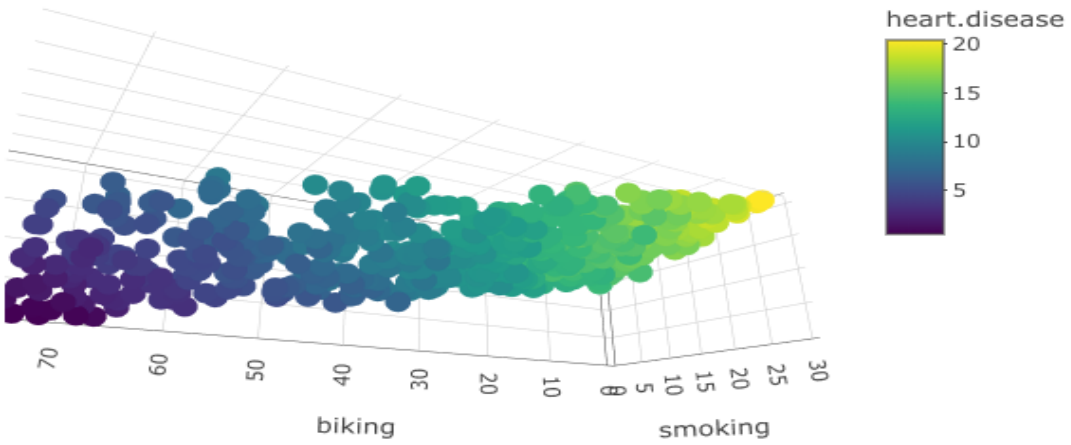
```
> MulRegModel <- lm(formula = heart.disease~biking+smoking, data = df)
> summary(MulRegModel)

Call:
lm(formula = heart.disease ~ biking + smoking, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-2.1789 -0.4463  0.0362  0.4422  1.9331

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 14.984658   0.080137  186.99   <2e-16 ***
biking      -0.200133   0.001366 -146.53   <2e-16 ***
smoking      0.178334   0.003539   50.39   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.654 on 495 degrees of freedom
Multiple R-squared:  0.9796,        Adjusted R-squared:  0.9795
F-statistic: 1.19e+04 on 2 and 495 DF,  p-value: < 2.2e-16
```

Nous rejetons l'hypothèse H_0 avec un niveau de confiance de 95% car nous pouvons voir que les valeurs, ($p-value_{intercept}$=$p-value_{biking}$=$p-value_{smoking}$=2e-16<0.05) sont significativement inférieures à 5% pour toutes les variables, y compris l'intercept. Ce qui signifie qu'il existe une association significative entre la variable explicatives (biking et smoking) et le variable à expliquer (heart.disease) De plus, le coefficient d'ajustement R^2 est de 0,9796, ce qui donne à notre modèle un score de 97,96%, ce qui est un excellent résultat. Ainsi, notre modèle de régression multiple

$$heart.disease = -0.200133 \times biking + 0.178334 \times smoking + 14.984658 \tag{7.4}$$

Utilisons la fonction plot_ly() du package plotly pour représenter interactivement les trois variables en trois dimensions.

```
library(plotly)
plot_ly(df, x = ~biking, y = ~smoking, z = ~heart.disease, color =~heart.disease)
\%>\% add_markers()
\%>\% layout(scene = list(xaxis = list(title = 'biking'),
                     yaxis = list(title = 'smoking'),
                     zaxis = list(title = 'heart.disease')))
```



7.3 La régression des moindres carrés partiels

La régression PLS (partial least square) est une méthode ancienne largement utilisée lors de l'analyse de données spectrales discrétisées et donc toujours de grande dimension. La régression PLS s'avère concrètement une méthode efficace qui justifie son emploi très répandu mais présente le défaut de ne pas se prêter à une analyse statistique traditionnelle qui exhiberait les lois de ses estimateurs. En pratique remplace souvent la régression

linéaire ordinaire. Cette méthode s'applique quand le nombre de variables est plus important que le nombre d'observations, ou encore lorsqu'il existe une forte multicolinéarité entre les variables explicatives. Elle s'applique également en présence de données manquantes Elle est ainsi restée un marge des approches traditionnelles de la Statistique.

La régression PLS recherche de manière itérative une séquence de composants (ou variables sous-jacentes) orthogonaux les uns aux autres. Ces composantes, appelées composantes PLS, sont choisies pour maximiser la covariance avec la variable explicative Y. Le choix du nombre de composantes est important car il influence fortement la qualité de la prévision.

Le modèle de régression PLS est construit à partir de l'échantillon d'apprentissage puis les prédictions sont faites sur l'échantillon de validation. Nous construisons d'abord un modèle PLS à un composant, puis nous répétons ce processus pour un PLS à 2, 3, ..., k composants. Le nombre de composants retenus correspond au modèle résultant en une erreur de prédiction minimale. Cette procédure se fait dans le package pls. Pour construire le modèle prédictif, nous disposons d'une base de données de 177 types de sols (individus) avec à la fois une teneur en carbone organique (CO, variable à expliquer) et un spectre obtenu dans le visible et le proche infrarouge (400 nm-2500 nm), fournissant 2101 variables explicatives. Nous avons ensuite voulu prédire la teneur en carbone organique de trois nouveaux sols.

les données sont importées du site $https://r-stat-sc-donnees.github.io/spe_bretagne.txt$, calculons la moyenne et l'écart-type de chaque spectre et impliquons un chagement qui consiste à centrer réduire les données sauf colonnes "CO" correspond à Y.

```
> df <- read.table("https://r-stat-sc-donnees.github.io/spe_bretagne.txt", sep=";",
+  header=TRUE, row.names=1)
> avr.raw <- apply(df[,-1], 1, mean)
> df[,-1] <- sweep(df[,-1], 1, avr.raw, FUN="-")
> sd.row <- apply(df[,-1], 1, sd)
> df[,-1] <- sweep(df[,-1], 1, sd.row, FUN="/")
```

La visualisation de la variable à expliquer Y par le graphique histogramme.

```
> hist(df[,"CO"], prob=TRUE, main="", xlab="Organic carbon content")
> lines(density(df[,"CO"]),col="red")
> rug(df[,"CO"], col="blue")
```

Dans la figure Figure 7.1 Toutes les observation sont moins de 5 sauf une qui est extrême (supérieure à 8)

```
> which(df[,1]>8)
[1] 79
> max(df[-79,1])
[1] 4.89
```

l'individu 79 prend une valeur très élevée (une valeur extême qui doit être enlevé pour la

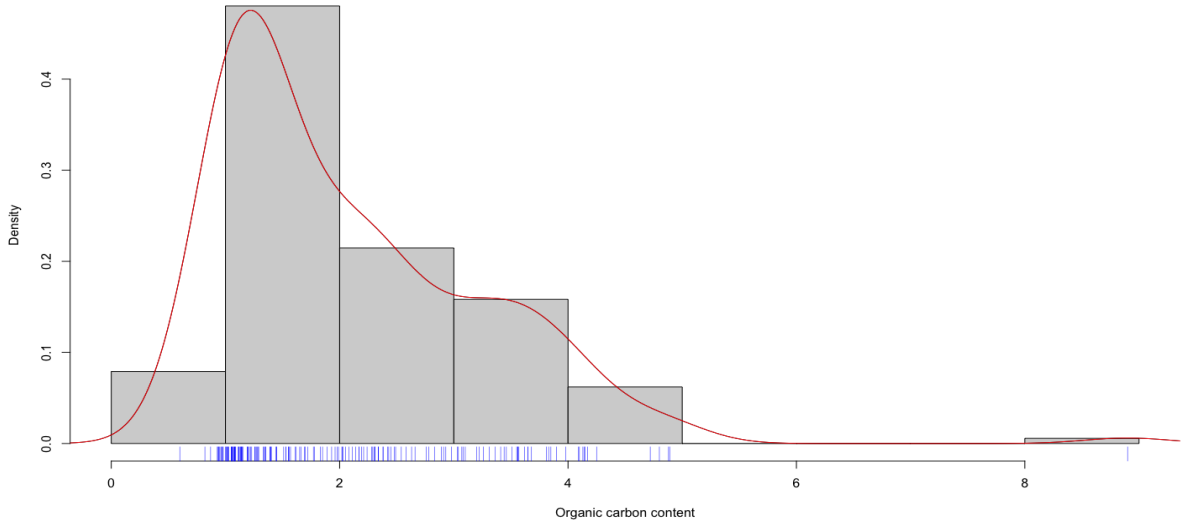


FIGURE 7.1 – Représentation de la teneur en carbone organique

construction du modèle), par contre les autres valeurs ne dépassent pas 4.89.

```
> df <- df[-79,]
> dim(df)
[1]  176 2102
```

En représentant chaque spectre par un trait différent (trait plein, tiret, etc) et/ou par une couleur différente selon la valeur de la variable CO, on divisera la variable CO en 7 classes environ égales par fonction cut

```
> coul <- as.numeric(cut(df[,1], quantile(df[,1], prob=seq(0,1,by=1/7)), include.lowest = TRUE))
> palette(terrain.colors(7,alpha=0.5))
> matplot(x=400:2500, y=t(as.matrix(df[,-1])), type="l", lty=1,
+         col=coul, xlab="wave length", ylab="Reflectance")
```

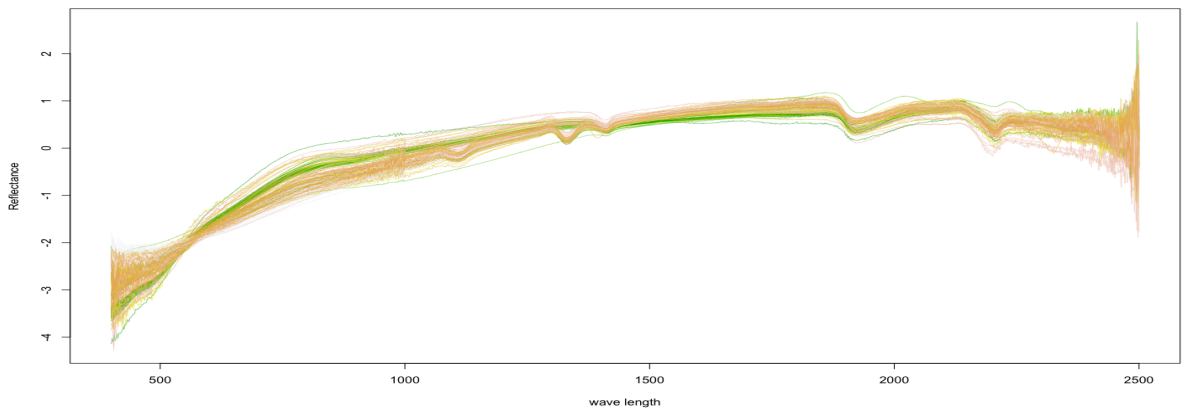


FIGURE 7.2 – Représentation des individus

des courbes assez proches supposent des valeurs proches pour la variable CO, en fait les courbes sont tracées avec le même couleur (voir Figure 7.2). Nous réduisons les variables explicatives grâce à l'argument scale, et nous fixer un nombre maximum de composantes PLS (notre cas 100).

```
library(pls)
> modele.pls <- plsr(CO~.,ncomp=100,data=df,scale=TRUE,validation="CV")
```

La validation croisée établit le nombre de composantes PLS par défaut. Tout d'abord, calculons l'erreur d'ajustement et l'erreur de prédiction acquises à partir des composantes PLS 1,2, ... ,100. (voir Figure 7.3)

```
> msepcv.pls <- MSEP(modele.pls, estimate=c("train","CV"))
> palette("default")
> plot(msepcv.pls, lty=1, type="l", legendpos="topright", main="")
```

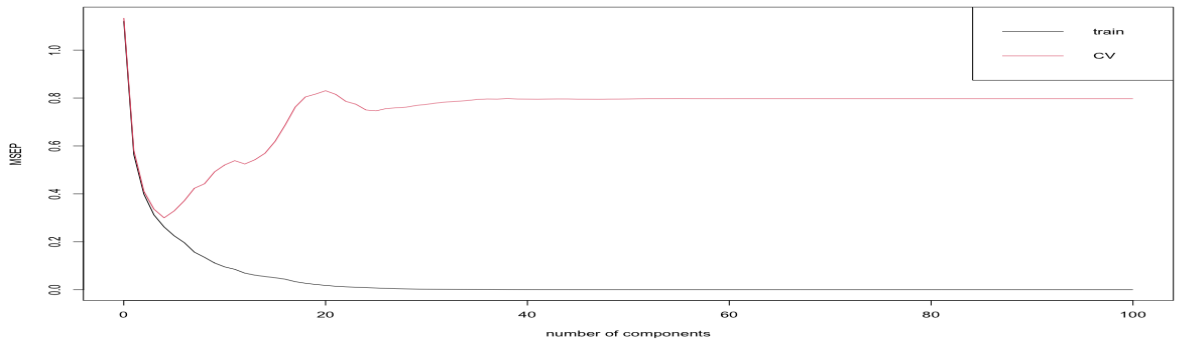


FIGURE 7.3 – Evolution des erreurs en fonction du nombre de composantes PLS.

L'erreur d'ajustement diminue avec le nombre de composants et une décroissance suivie d'une croissance pour l'erreur de prévision, le nombre optimal de composantes pour la prévision correspond à la valeur pour laquelle l'erreur de prévision est minimum.

```
> ncomp.pls <- which.min(msepcv.pls\$val["CV",,])-1
> ncomp.pls
4 comps
      4
```

On soustrait 1 car la première erreur est donnée pour 0 composante, et donc on a 4 composantes PLS. Et alors le modèle finale du PLS

```
> reg.pls <- plsr(CO~., ncomp=ncomp.pls, data=df, scale=TRUE)
```

La fonction residuals calcule les résidus du modèle à 4 composantes.

```
> res.pls <- residuals(reg.pls)
```

```
> plot(res.pls[,,ncomp.pls],pch=18,cex=.5,ylab="Résidus",main="",col="red")
> abline(h=c(-2,0,2), lty=c(2,1,2))
```

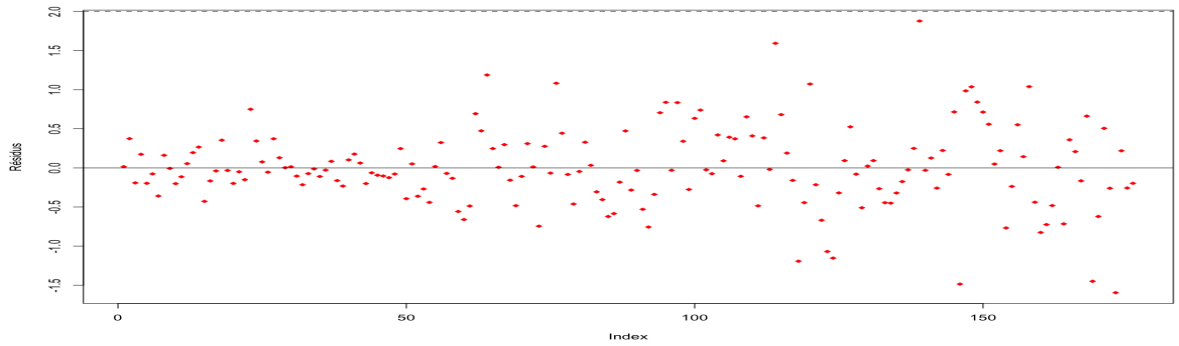


FIGURE 7.4 – Représentation des résidus du modèle à 4 composantes

Ça se voit clairement que la variance des résidus (plus grande pour les derniers individus, et plus faible pour les premiers) montre que l'échantillon n'est pas très homogène.

Pour prévoir une nouvelle valeur, et afin de comparer la teneur en CO prédite par la régression PLS avec les valeurs réelles observées, nous avons obtenu un ensemble de trois courbes complémentaires.

```
> dfNew <- read.table("https://r-stat-sc-donnees.github.io/spe_nouveau.txt", sep=";",+
+   header=TRUE, row.names=1)
> avr.raw <- apply(dfNew[,-1], 1, mean)
> dfNew[,-1] <- sweep(dfNew[,-1], 1, avr.raw, FUN="-")
> sd.row <- apply(dfNew[,-1], 1, sd)
> dfNew[,-1] <- sweep(dfNew[,-1], 1, sd.row, FUN="/")
```

La fonction predict est utilisée pour obtenir les valeurs prédites,

```
> pred <- predict(reg.pls, ncomp=ncomp.pls, newdata=dfNew[,-1])
> pred
, , 4 comps

                CO
3236     0.4065104
rmqs_726 2.6930854
rmqs_549 2.6035296
```

On compare les valeurs prédites aux valeurs mesurées :

```
> dfNew[,1]
[1] 1.08 1.60 1.85
```