Apprendre à réaliser une régression linéaire

Marie Vaugoyeau

17 December 2024

Table of contents

1 import des packages

```
library(tidyverse)
```

2 Définition de la `régression linéaire`

Objectif : Trouver une équation de type linéaire qui permet d'expliquer une **variable réponse quantitative** par **une ou plusieurs variable(s) explicative(s)**.

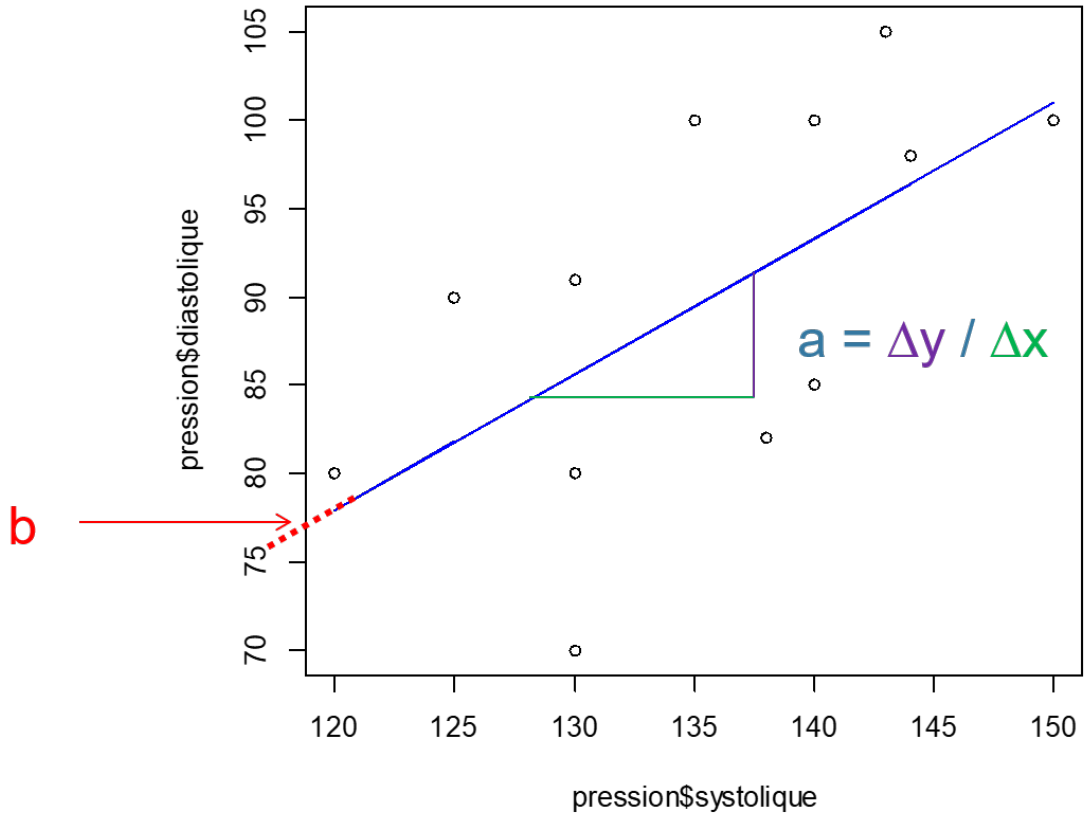
Différence entre régression linéaire et modèle linéaire

Il n'y en a pas !
Certaines personnes parlent de **modèle de régression linéaire**.

L'équation est de la forme :

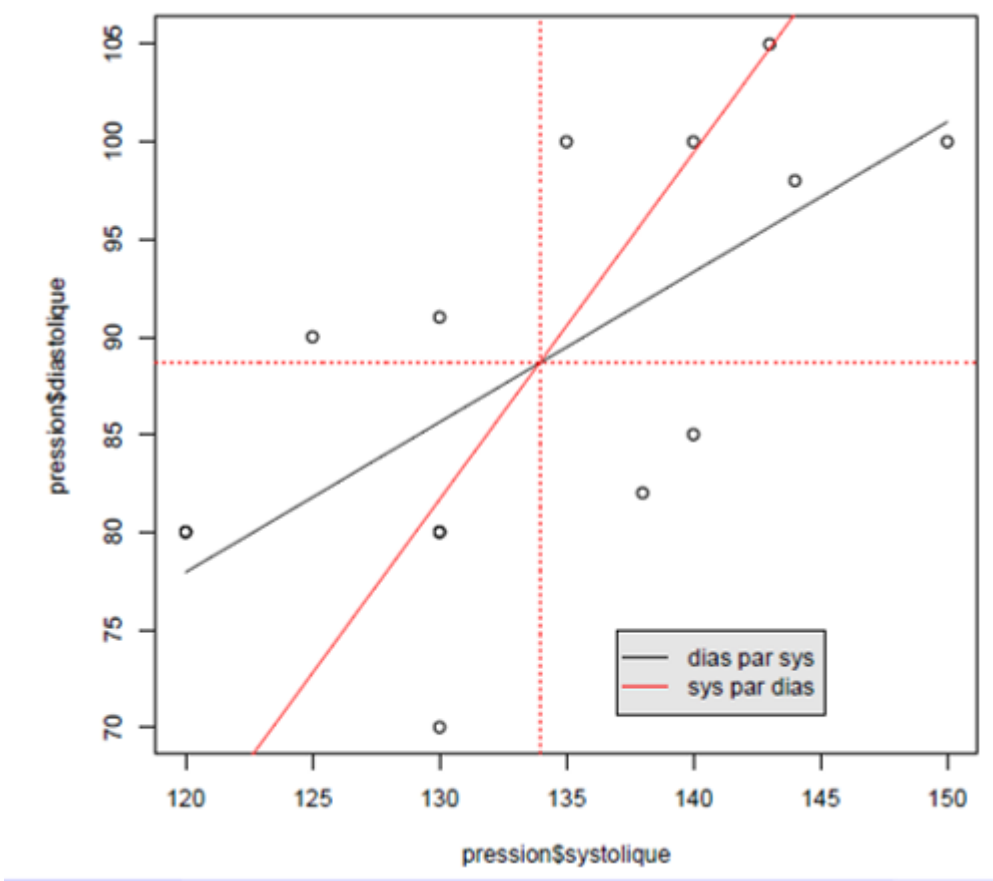
$$Y = aX + b$$

Avec `a : la pente` et `b : l'ordonnée à l'origine ou intecept`



Attention

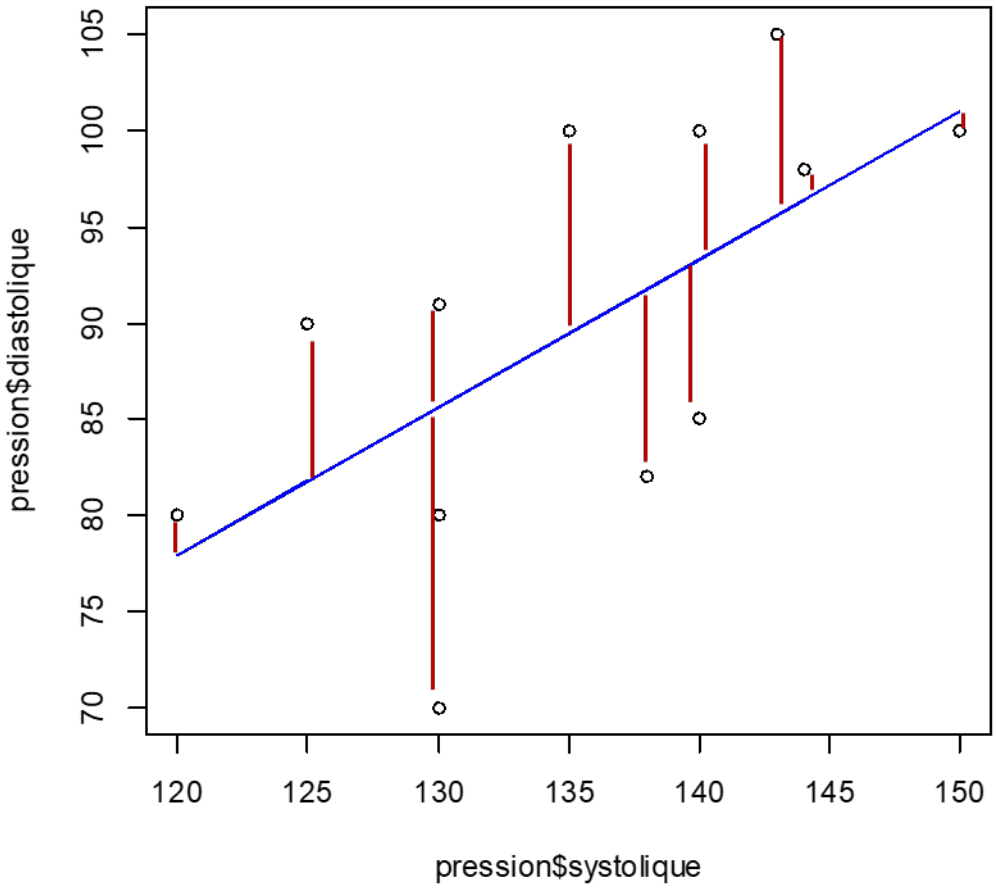
La régression de `Y` en fonction de `X` n'est pas la même que la régression de `X` en fonction de `Y`.



3 Etude des résidus

Pour ajuste la droite de régression, la méthode utilisée se base sur les **résidus** : **la méthode des moindres carrées**.
L'idée est d'avoir la somme la plus petite possible.

> **i** Les résidus
>
> Un résidu est la **différence** entre la **valeur observée** et la **valeur prédite par l'équation linéaire**.



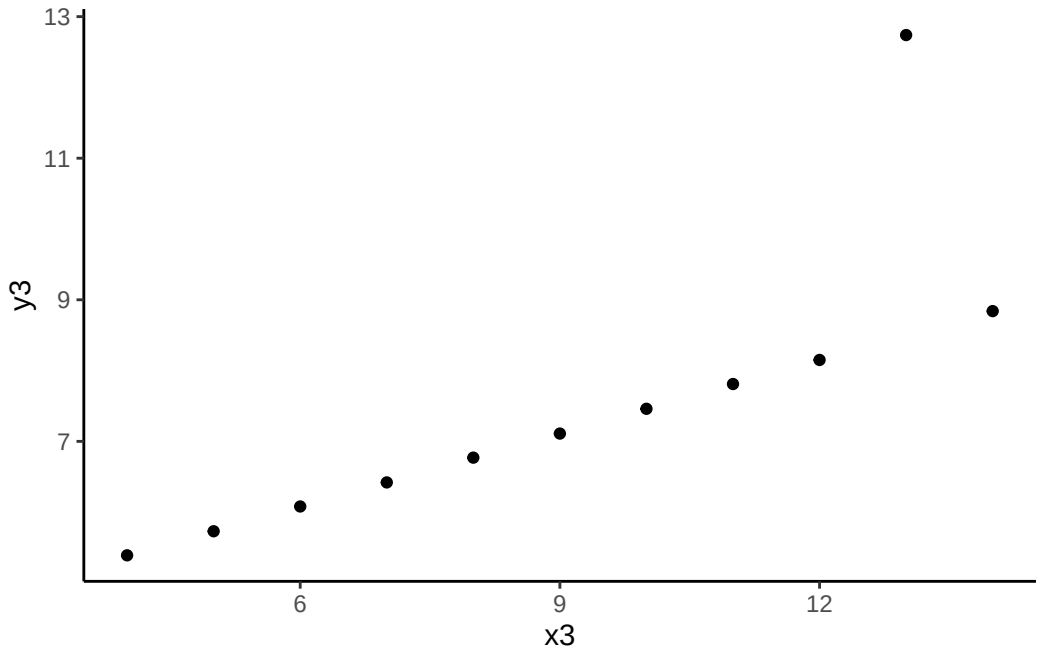
Les résidus doivent suivre une loi normale, vérifiable grâce à un `graphique quantile-quantile` (`QQplot`) ou le test de `Shapiro-Wilk`.

4 Les points extrêmes

Il y a deux sortes d'extrêmes :

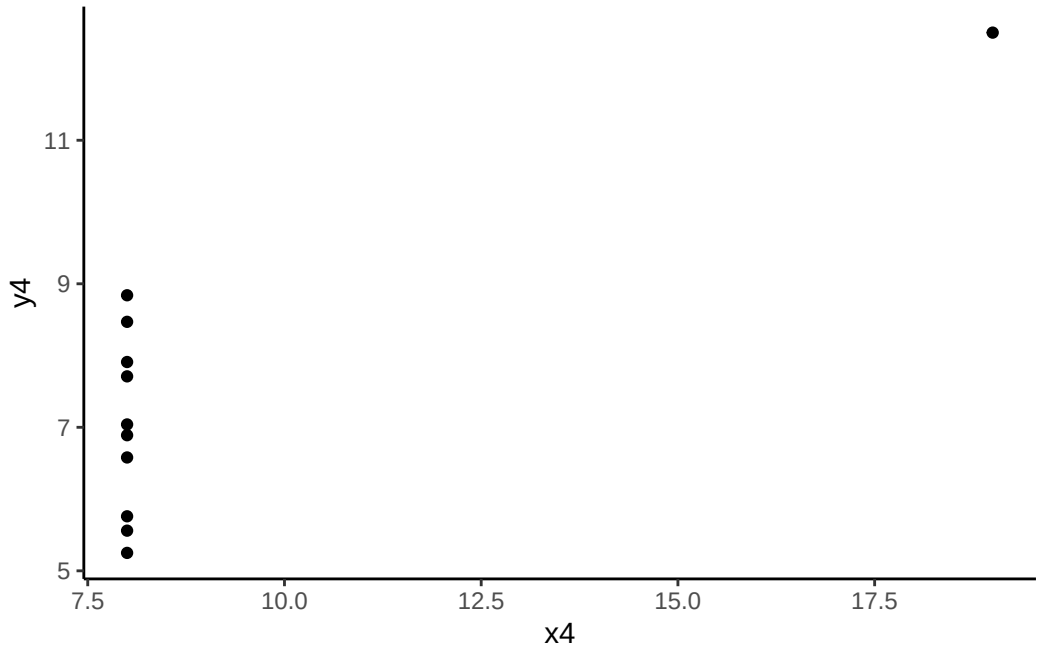
- **Extrême sur Y** : ordonnée très différente des autres points d'abscisse proche -> **Point non consistant**

```r
anscombe |>
  ggplot() +
  aes(x = x3, y = y3) +
  geom_point() +
  theme_classic()
```



- **Extrême sur X** : abscisse nettement plus petite ou plus grande que celle des autres points -> **Phénomène de levier**

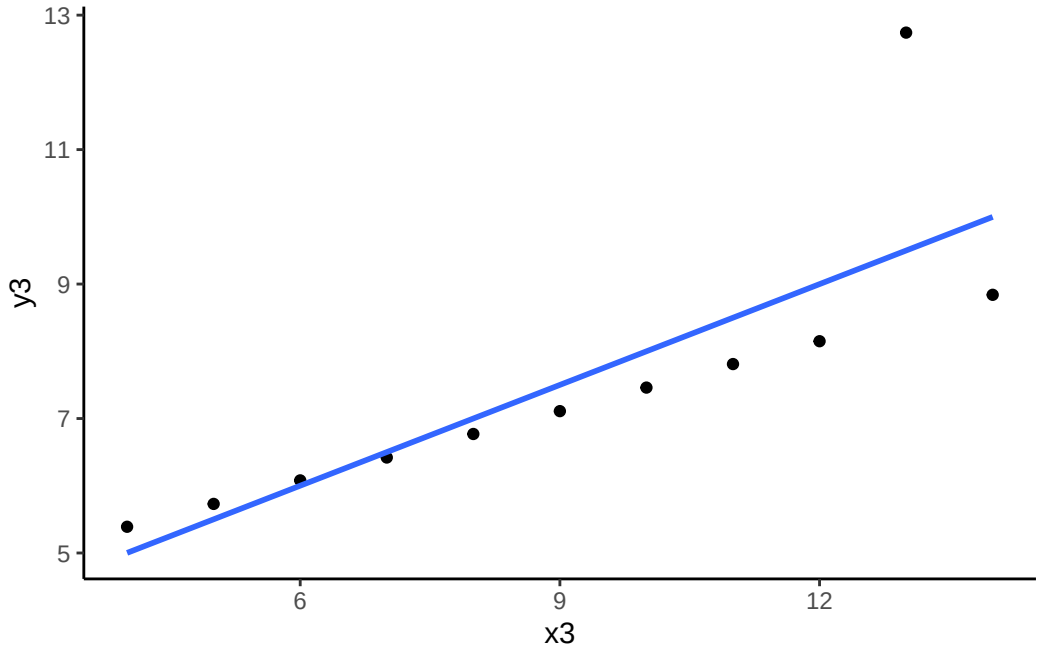
```r
anscombe |>
  ggplot() +
  aes(x = x4, y = y4) +
  geom_point() +
  theme_classic()
```



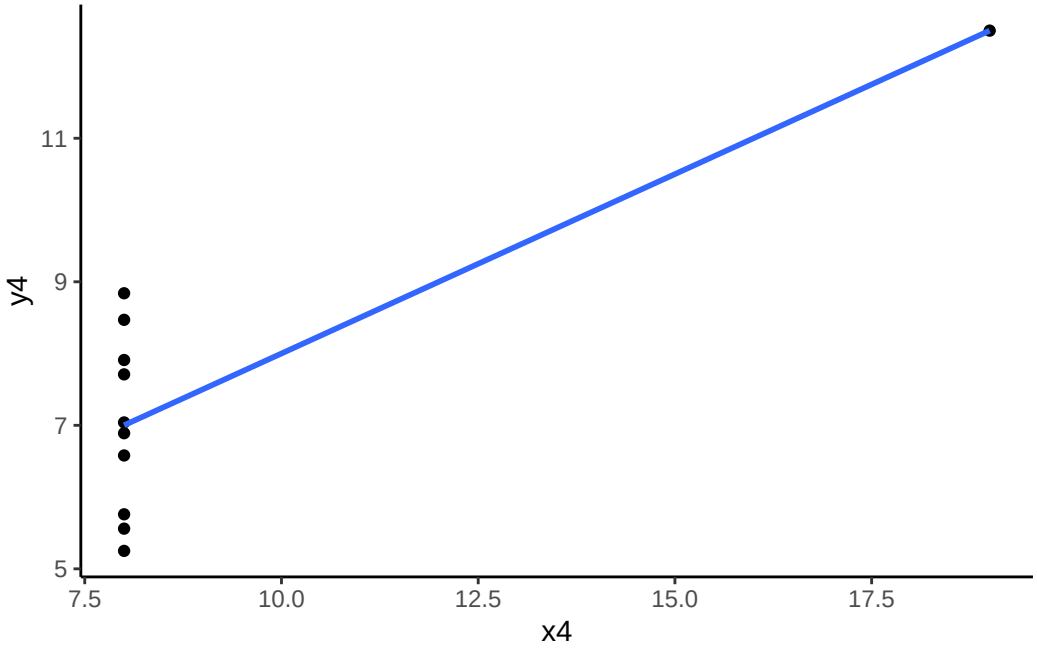
Point influent

Dans les deux cas, un point est **influant** lorsque la régression pratiquée avec ou sans ce point conduit à des résultats très différents.

```
anscombe |>
  ggplot() +
  aes(x = x3, y = y3) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  theme_classic()
```




```
anscombe |>
  ggplot() +
  aes(x = x4, y = y4) +
  geom_point() +
  geom_smooth(
    method = "lm",
    se = FALSE) +
  theme_classic()
```



5 Les données

Les données utilisées sont celles du jeu de données `iris`. Les longueurs et largeurs de sépales et pétales ont été mesurées sur 50 iris de 3 espèces, plus d'information sur la page d'aide `help(iris)`.

```
summary(iris)
```

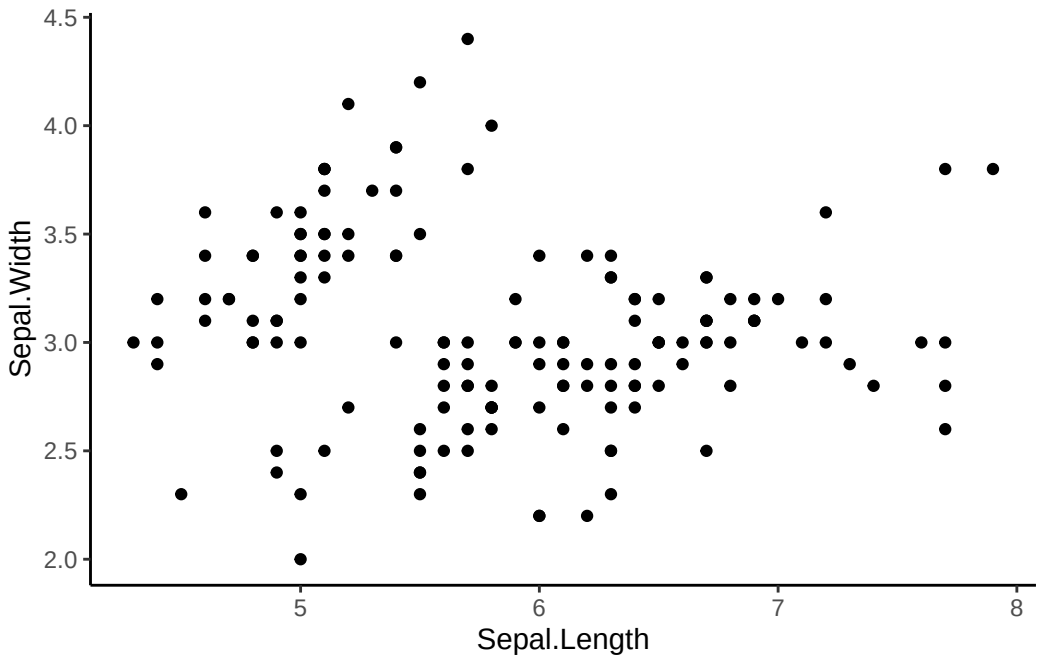
```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
 Median :5.800   Median :3.000   Median :4.350   Median :1.300
 Mean   :5.843   Mean   :3.057   Mean   :3.758   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
```

6 Réalisation d'une régression linéaire

6.1 1ère étape : Réalisation d'un nuage de points

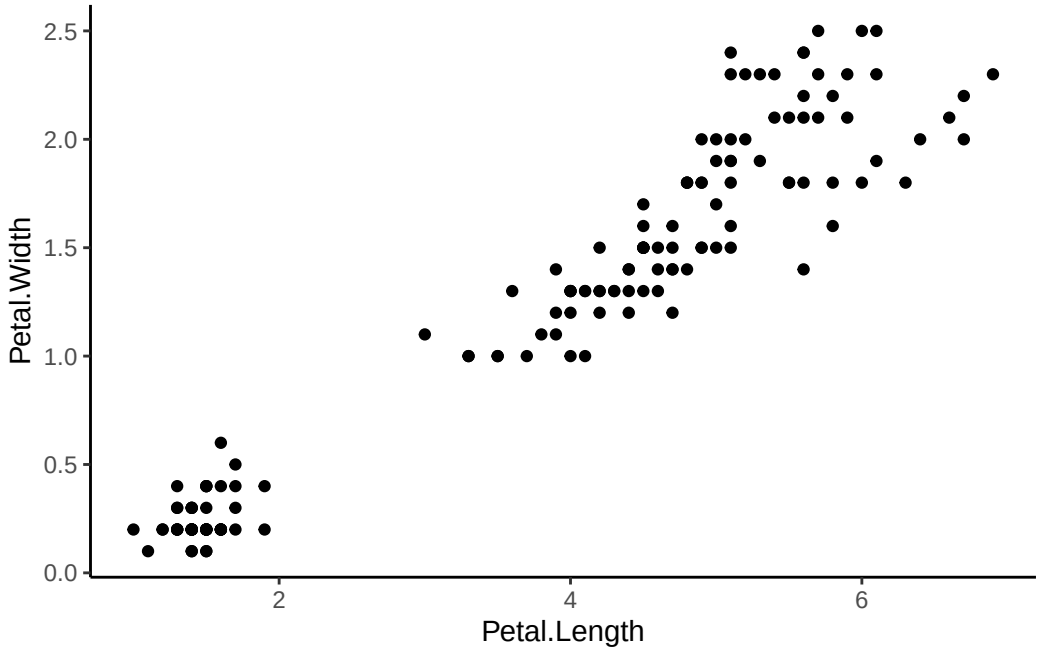
La visualisation des données est une étape indispensable afin de **vérifier les données** et de **contrôler la linéarité** des données.

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



> **Attention**
>
> Il ne faut pas réaliser de régression linéaire si graphiquement on ne distingue pas de relation linéaire entre les données.

```r
ggplot(iris) +
  aes(x = Petal.Length, y = Petal.Width) +
  geom_point() +
  theme_classic()
```



6.2 2ème étape : Vérifier les limites d'utilisation de la régression

Les données doivent-être indépendantes et suivre (ou être approximées par) des lois normales.

```r
shapiro.test(iris$Sepal.Length)
```

```
    Shapiro-Wilk normality test

data:  iris$Sepal.Length
W = 0.97609, p-value = 0.01018
```

```r
shapiro.test(iris$Sepal.Width)
```

```
    Shapiro-Wilk normality test
```

```
data:  iris$Sepal.Width
W = 0.98492, p-value = 0.1012
```

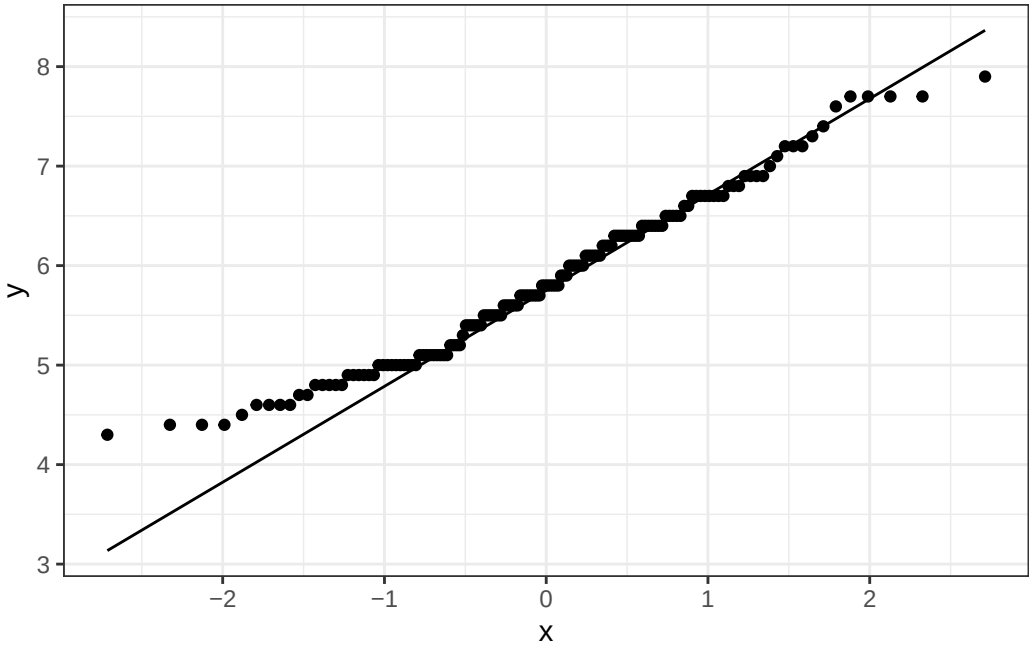
```r
shapiro.test(iris$Petal.Length)
```

```
    Shapiro-Wilk normality test

data:  iris$Petal.Length
W = 0.87627, p-value = 7.412e-10
```

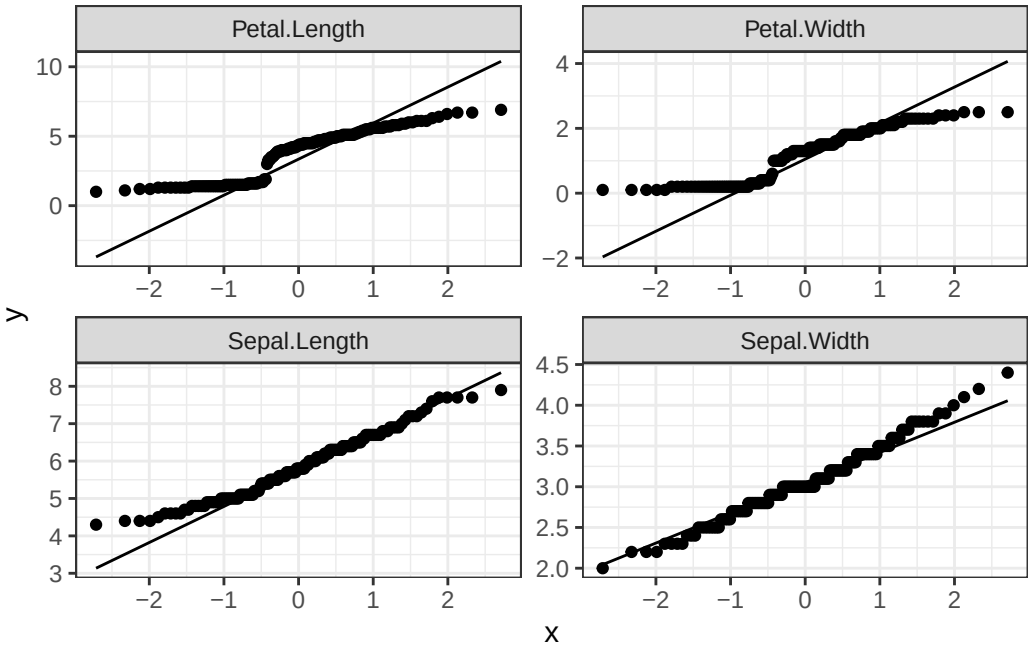
```r
shapiro.test(iris$Petal.Width)
```

```
    Shapiro-Wilk normality test

data:  iris$Petal.Width
W = 0.90183, p-value = 1.68e-08
```

```r
iris |>
  ggplot() +
  aes(sample = Sepal.Length) +
  geom_qq() +
  geom_qq_line() +
  theme_bw()
```



```r
iris |>
  pivot_longer(
    cols = - Species
  ) |>
  ggplot() +
  aes(sample = value) +
  geom_qq() +
  geom_qq_line() +
  facet_wrap(~ name, scales = "free") +
  theme_bw()
```



> **Note**
>
> La régression linéaire est assez résistante à l'absence de normalité et il est possible de la faire ici en prenant en compte **la loi des grands nombres**.

6.3 3ème étape : Création du modèle linéaire

Plusieurs packages ont des fonctions qui permettent de réaliser un modèle linéaire.
Ici je vais rester sur la fonction `lm()` du package `{stats}` automatiquement chargé dans l'environnement.

Cette fonction prend comme premier argument la `formula`, c'est-à-dire la formule de type `y ~ x` et en deuxième argument `data`, le jeu de données utilisé.

```
modele_lineaire_petale <- lm(
  Petal.Width ~ Petal.Length,
  data = iris
)
```

Pour accéder aux coefficients, il y a plusieurs solutions :

- Rappeler le nom du modèle
- Utiliser la fonction `summary()` du package `{base}`
- Applique la fonction `anova()` du package `{stats}`
- Prendre la fonction `Anova()` du package `{car}`

```
modele_lineaire_petale
```

```

Call:
lm(formula = Petal.Width ~ Petal.Length, data = iris)

Coefficients:
 (Intercept)  Petal.Length
     -0.3631        0.4158
```

```
summary(modele_lineaire_petale)
```

```

Call:
lm(formula = Petal.Width ~ Petal.Length, data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-0.56515 -0.12358 -0.01898  0.13288  0.64272

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.363076   0.039762  -9.131  4.7e-16 ***
Petal.Length  0.415755   0.009582  43.387  < 2e-16 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2065 on 148 degrees of freedom
Multiple R-squared:  0.9271,    Adjusted R-squared:  0.9266
F-statistic:  1882 on 1 and 148 DF,  p-value: < 2.2e-16
```

```r
anova(modele_lineaire_petale)
```

```
Analysis of Variance Table

Response: Petal.Width
              Df Sum Sq Mean Sq F value    Pr(>F)
Petal.Length   1  80.26  80.260  1882.5 < 2.2e-16 ***
Residuals    148   6.31   0.043
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
car::Anova(modele_lineaire_petale)
```

```
Anova Table (Type II tests)

Response: Petal.Width
             Sum Sq  Df F value    Pr(>F)
Petal.Length  80.26   1  1882.5 < 2.2e-16 ***
Residuals      6.31 148
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pour voir la différence entre les deux `anova` il faut ajouter des variables.

La sortie `summary()` nous dit que le modèle est significatif (`p-value: < 2.2e-16`) mais il faut vérifier qu'il est valide.

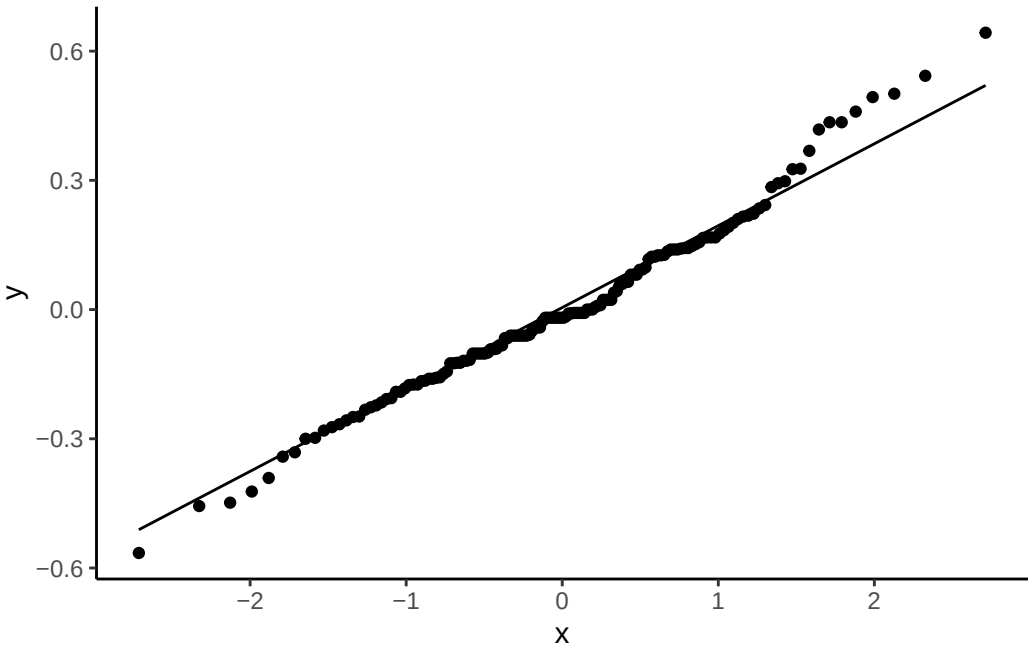
6.4 4ème étape : Validation du modèle

Le modèle est accepté si les **résidus** suivent une **loi normale**.

```r
modele_lineaire_petale$residuals |>
  shapiro.test()
```

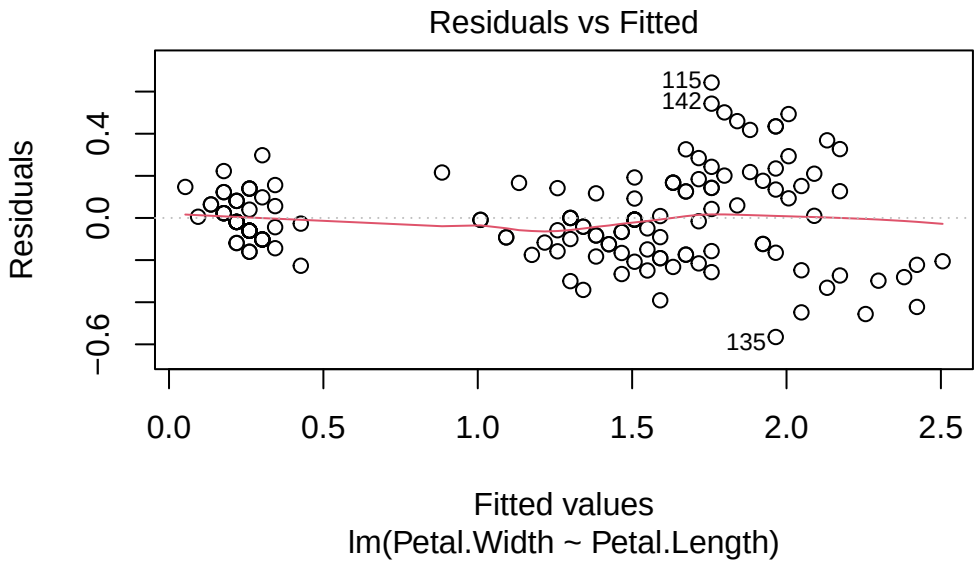
```
    Shapiro-Wilk normality test

data:  modele_lineaire_petale$residuals
W = 0.98378, p-value = 0.07504
```

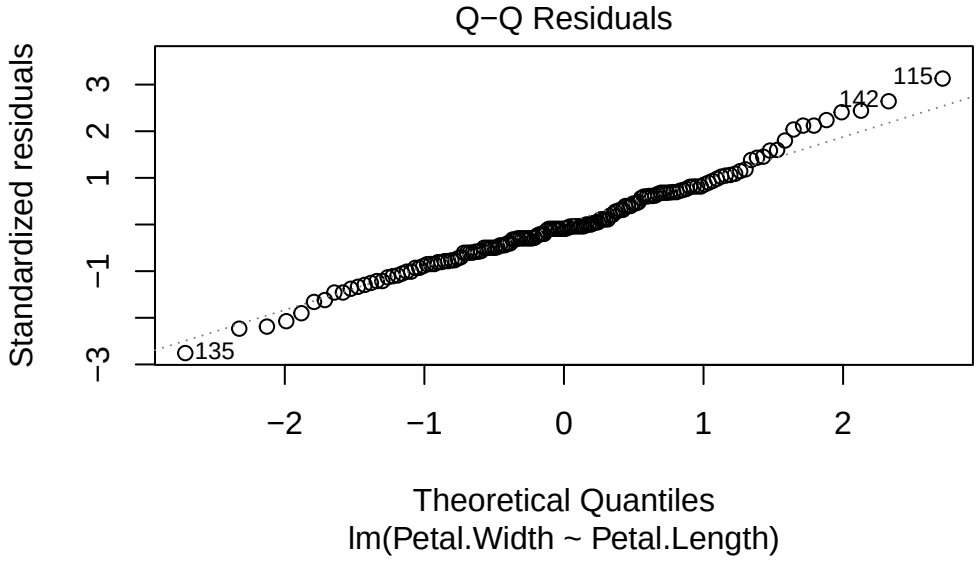
```
modele_lineaire_petale$residuals |>
  as_tibble() |>
  ggplot() +
  aes(sample = value) +
  geom_qq() +
  geom_qq_line() +
  theme_classic()
```

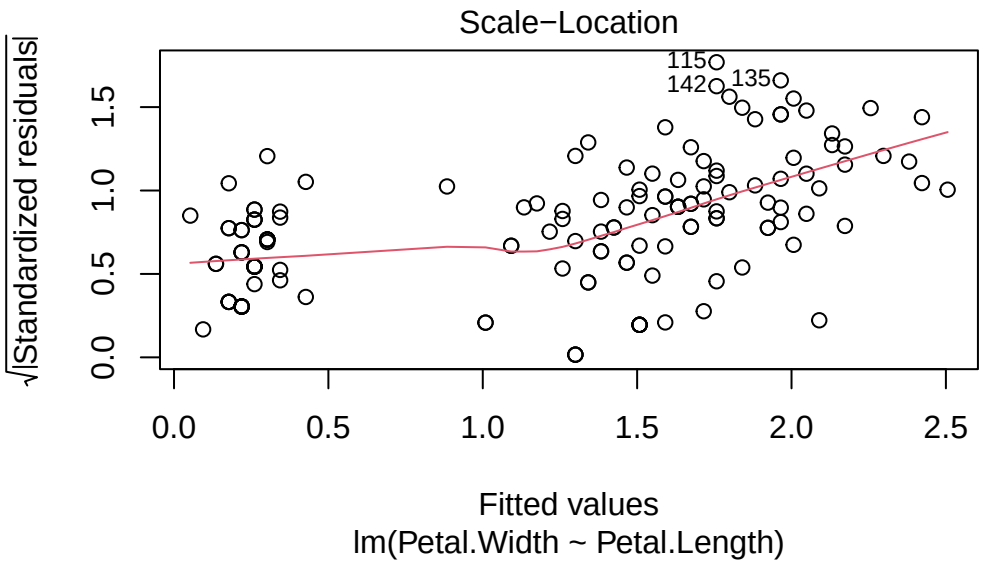


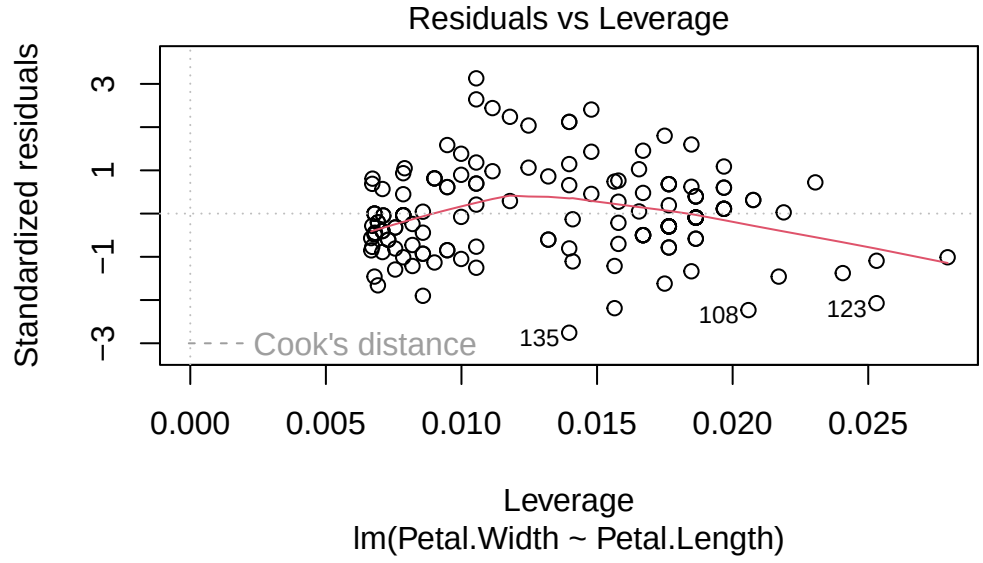
Il est aussi bien de visualiser le modèle grâce à la fonction `plot()`.

```
plot(modele_lineaire_petale)
```


Residuals vs Fitted
Residuals
Fitted values
lm(Petal.Width ~ Petal.Length)
115
142
135
−0.6
0.0
0.4
0.0
0.5
1.0
1.5
2.0
2.5


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(Petal.Width ~ Petal.Length)
115
142
135
−3
−1
1
2
3
−2
−1
0
1
2





6.5 5ème étape : Réalistion d'un graphique résumé

Le nuage de points avec une droite est la meilleur représentation.

```r
ggplot(iris) +
  aes(x = Petal.Length, y = Petal.Width) +
  geom_point() +
  geom_abline(
    intercept = modele_lineaire_petale$coefficients[[1]],
    slope = modele_lineaire_petale$coefficients[[2]],
    color = "red",
    linewidth = 2
  ) +
  geom_smooth(method = "lm") +
  ggpubr::stat_regline_equation() +
  theme_classic()
```