Apprendre à réaliser une régression linéaire

Marie Vaugoyeau

17 December 2024

Table of contents

1 import des packages **1**

2 Définition de la `régression linéaire` **2**

3 Etude des résidus **3**

4 Les points extrêmes **5**

5 Les données **7**

6 Réalisation d'une régression linéaire **8**

6.1 1ère étape : Réalisation d'un nuage de points 8
6.2 2ème étape : Vérifier les limites d'utilisation de la régression 10
6.3 3ème étape : Création du modèle linéaire 12
6.4 4ème étape : Validation du modèle 14
6.5 5ème étape : Réalisation d'un graphique résumé 15

7 En savoir un peu plus sur moi **16**

1 import des packages

```
library(tidyverse)
```

2 Définition de la `régression linéaire`

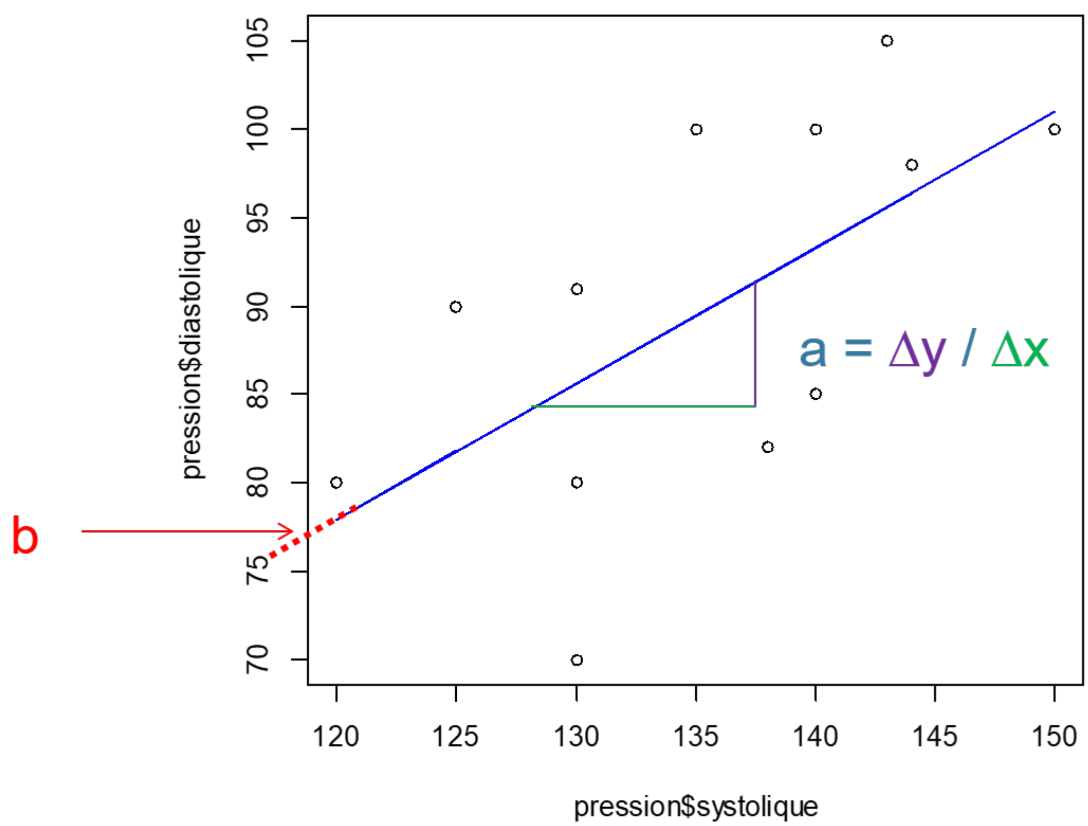
Objectif : Trouver une équation de type linéaire qui permet d'expliquer une **variable réponse quantitative** par **une ou plusieurs variable(s) explicative(s)**.

> **i** 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 :

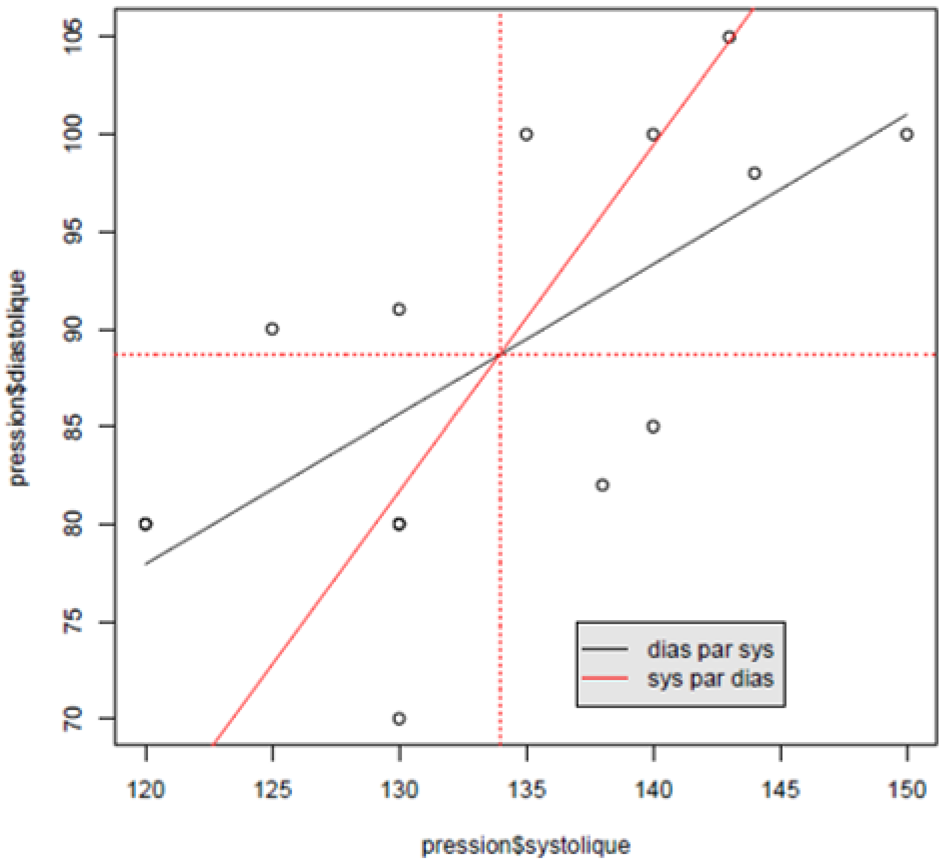
$$Y = aX + b$$

Avec `a : la pente (ou coefficient directeur)` 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`.

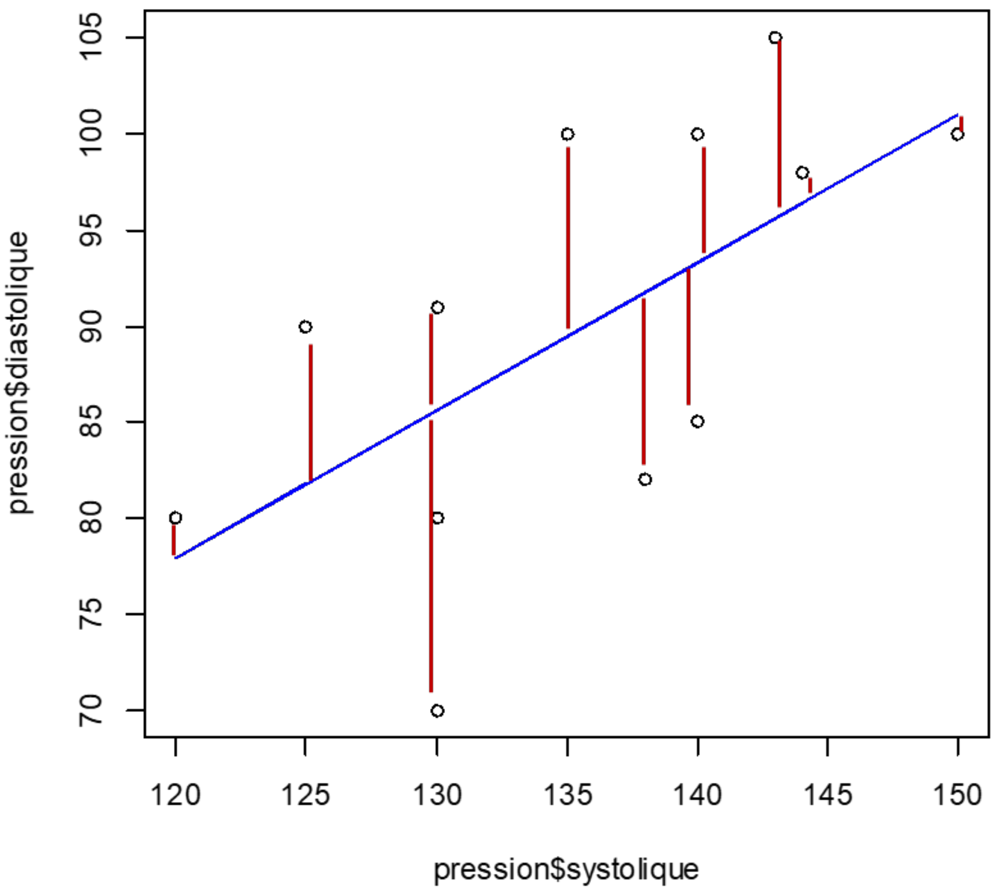


3 Etude des résidus

Pour ajuste la droite de régression, la méthode utilisée se base sur les **résidus** : **la méthode des moindres carrées**.
La somme du carré des résidus est calculée à chaque itération (création d'une nouvelle équation) et comparée aux autre. L'idée est d'avoir la plus petite somme des résidus possible.

Les résidus

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



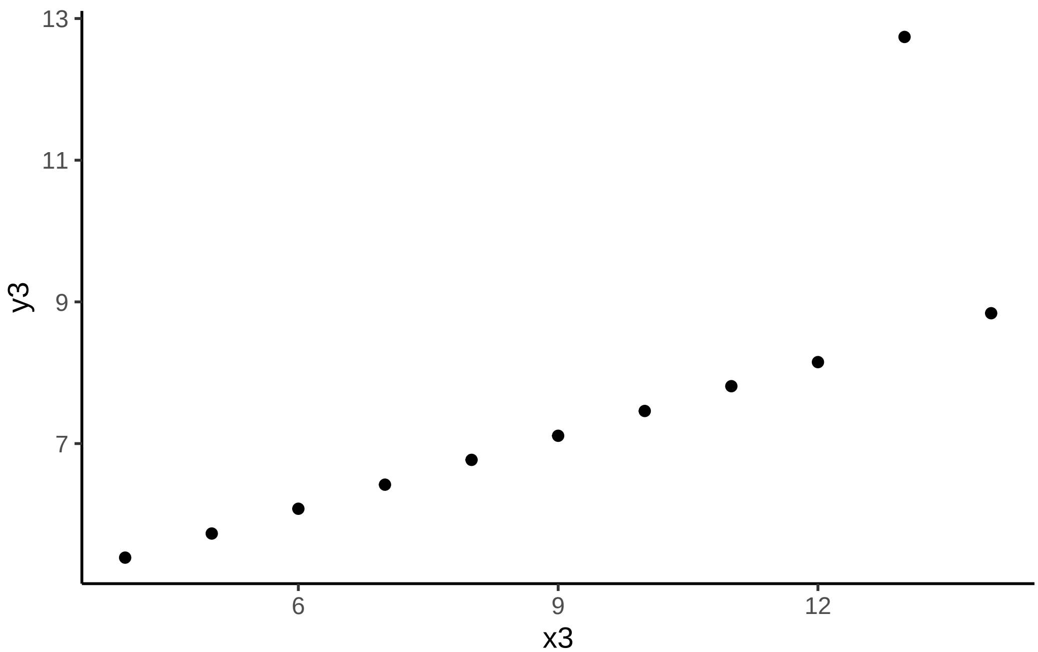
Les résidus doivent suivre une loi normale, vérifiable grâce à un `graphique quantile-quantile` (`QQplot`) ou le test de `Shapiro-Wilk`.
Plus d'information sur la loi normale dans cet article de blog.

4 Les points extrêmes

Il y a deux sortes d'extrêmes :

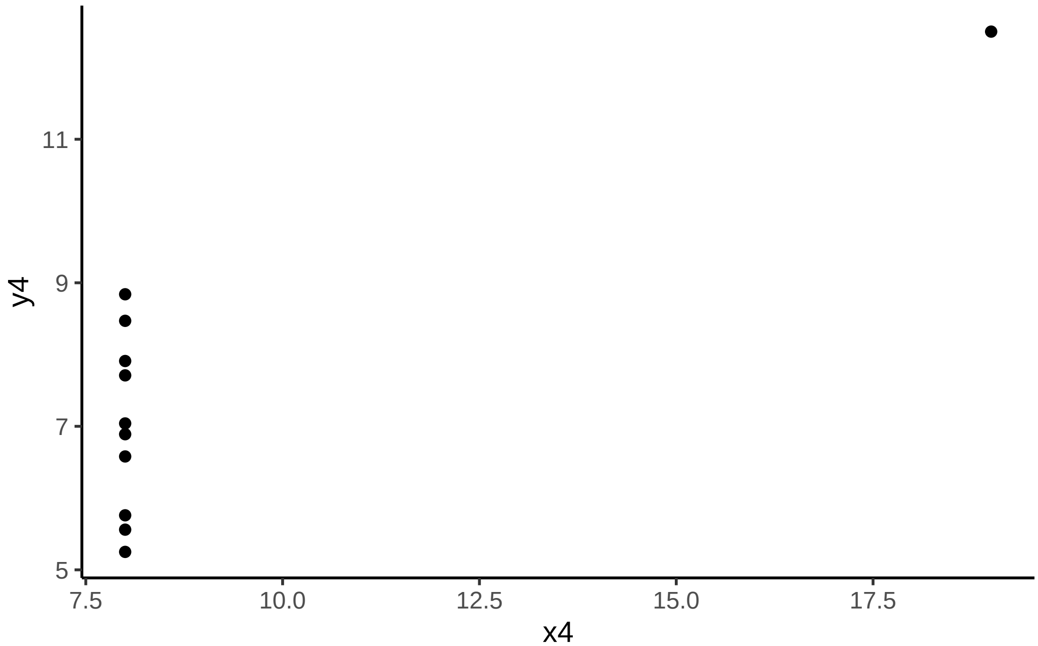
- **Extrême sur Y** : ordonnée très différente des autres points d'abscisse proche -> **Point non consistant**

```
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**

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

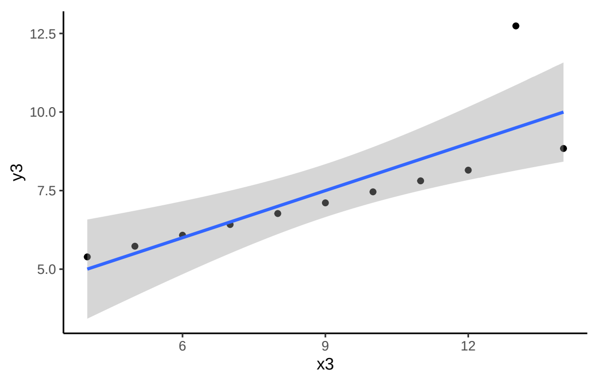


Point influent

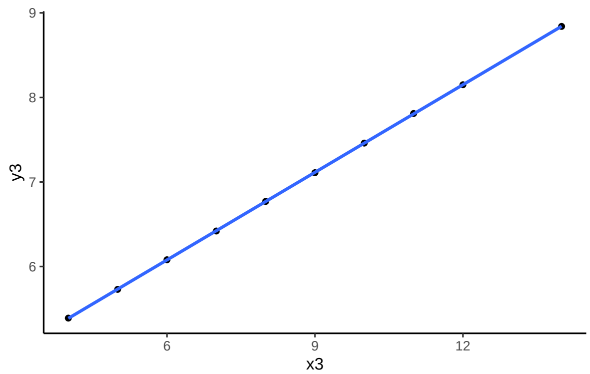
Dans les deux cas, un point est **influent** lorsque la régression pratiquée avec ou sans ce point conduit à des résultats très différents.

```
anscombe |>
  ggplot() +
  aes(x = x3, y = y3) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_classic()
anscombe |>
  filter(y3 < 10) |>
  ggplot() +
  aes(x = x3, y = y3) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_classic()
anscombe |>
  ggplot() +
  aes(x = x4, y = y4) +
  geom_point() +
  geom_smooth(method = "lm") +
```

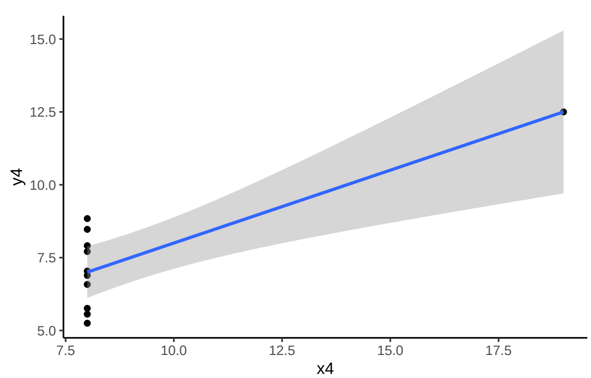
```
  theme_classic()
anscombe |>
  filter(x4 < 10) |>
  ggplot() +
  aes(x = x4, y = y4) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_classic()
```



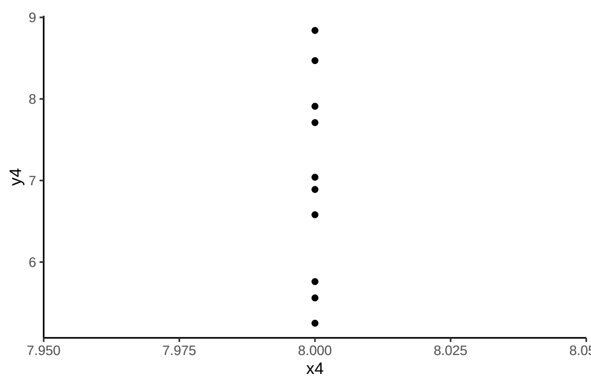
(a) Avec point consistant



(a) Sans point consitant



(a) Avec point levier



(a) Sans point levier

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)`.

```r
summary(iris)
```

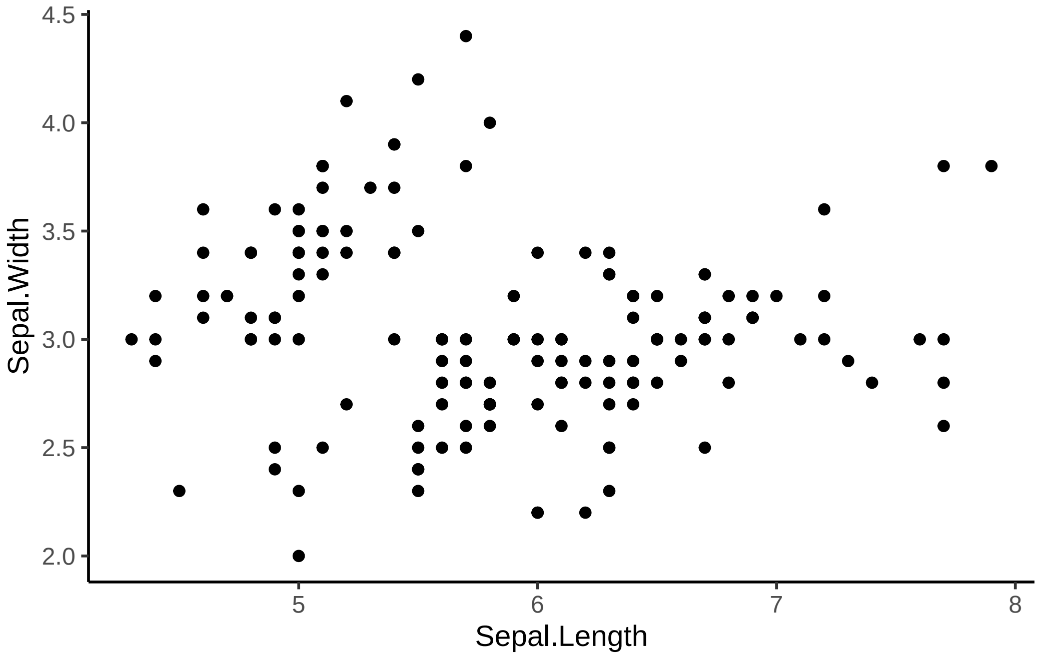
```
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean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

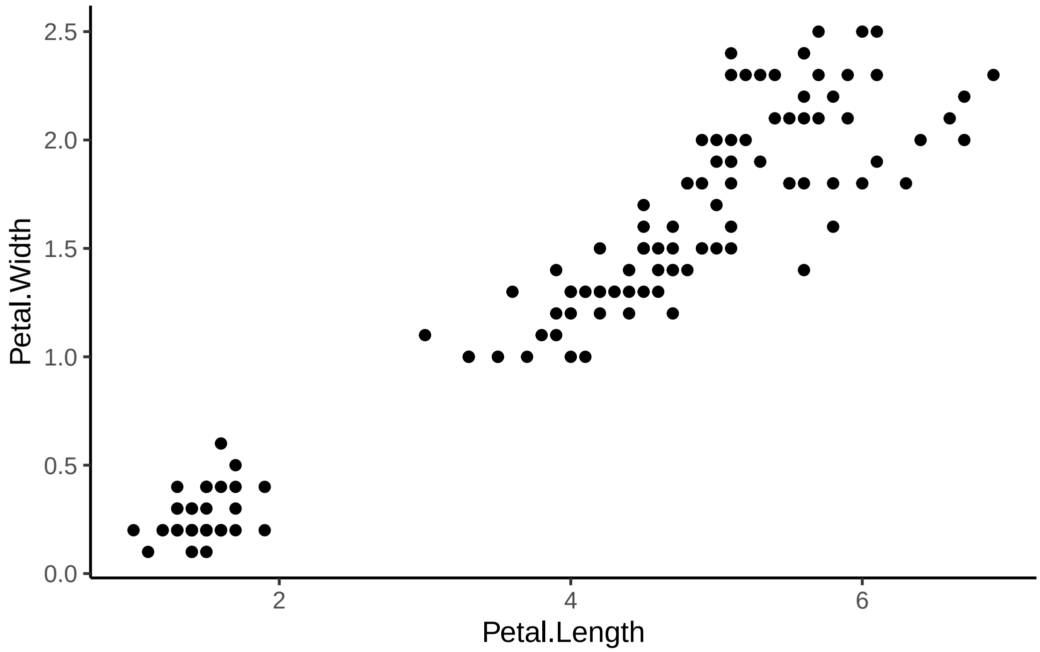
La visualisation des données est une étape indispensable afin de **vérifier les données** et de **contrôler la linéarité** des données.

```r
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.

```
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.

Test de Shapiro-Wilk

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

```

    Shapiro-Wilk normality test

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

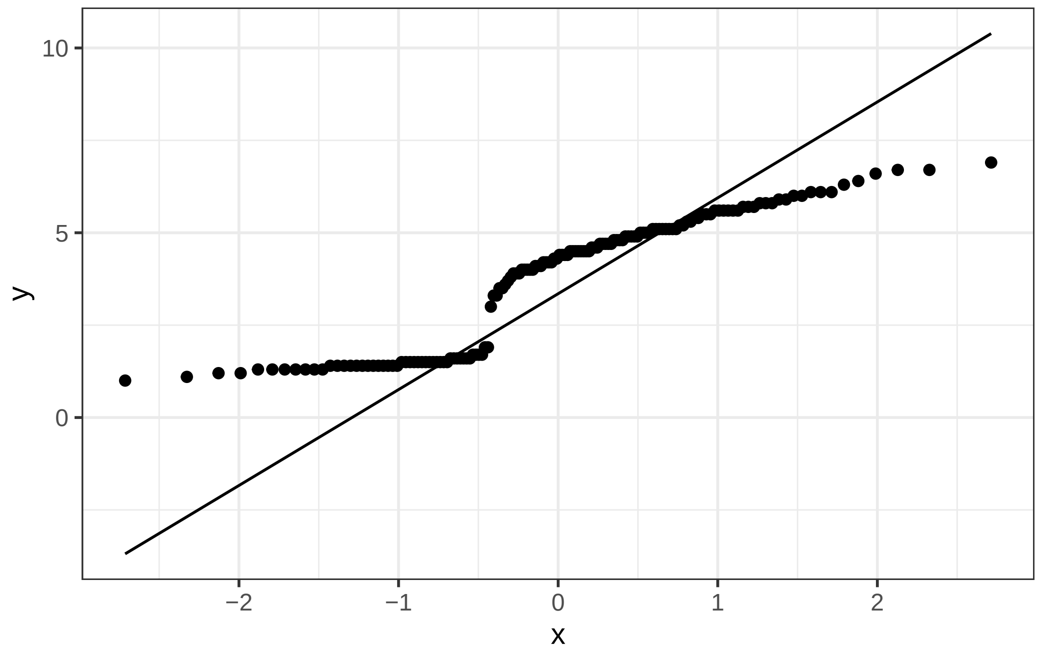
```
shapiro.test(iris$Petal.Width)
```

```

    Shapiro-Wilk normality test

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

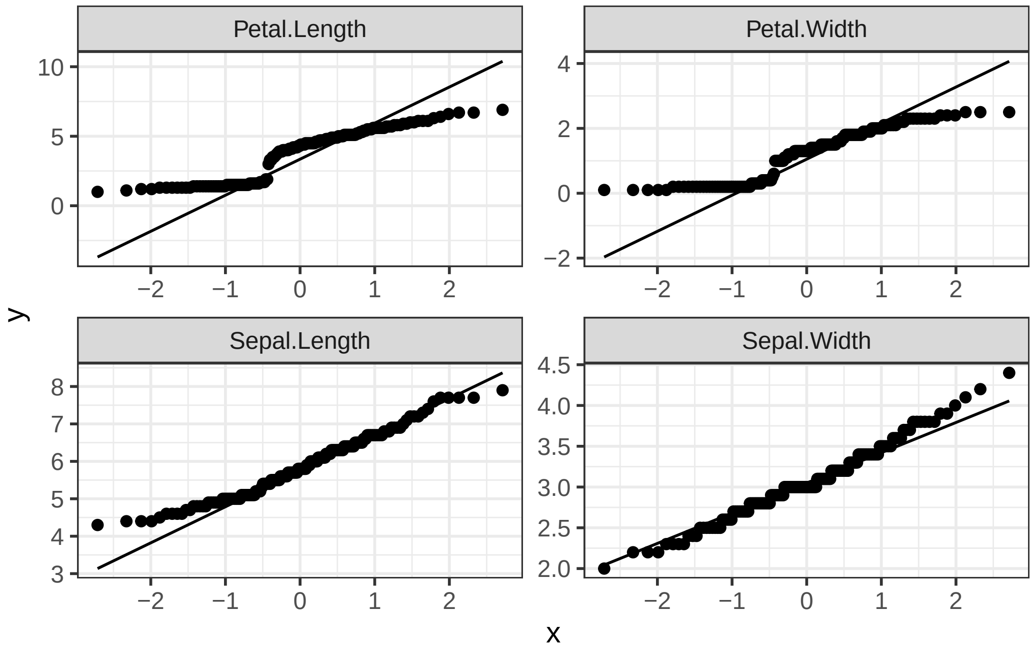
> **Note**
>
> Les longueurs et largeurs de pétales ne suivent pas des lois normales.

Représentation graphique

```r
iris |>
  ggplot() +
  aes(sample = Petal.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 : Ne donne pas les statistiques de test

- Utiliser la fonction `summary()` du package `{base}` : Le plus complet mais attention s'il y a plusieurs variables explicatives, les coeffcients et statistiques de test appliqués sont de type I.

- Applique la fonction `anova()` du package `{stats}` : Permet d'afficher facilement le tableau des coefficients mais type I aussi

- Prendre la fonction `Anova()` du package `{car}` : Même chose que précédent mais type II (et même III s'il y a une intéraction)

```
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**.

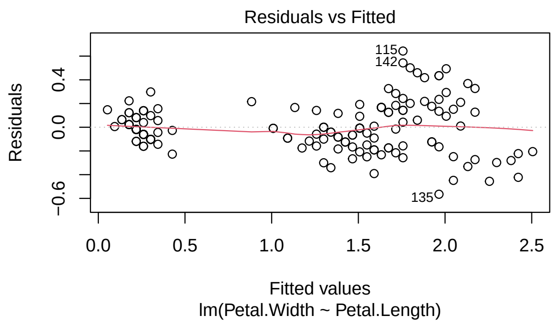
```r
modele_lineaire_petale$residuals |>
  shapiro.test()
```

```
	Shapiro-Wilk normality test

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

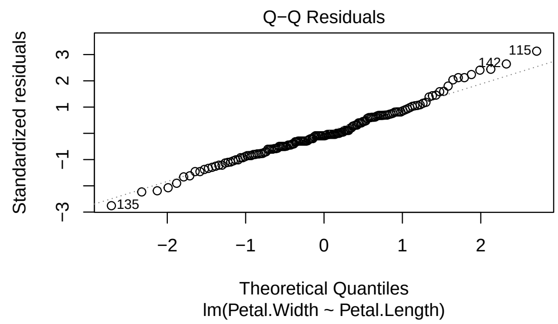
Les résidus suivent une loi normale (`p-valeur` > 0.05 -> impossible de rejeter l'hypothèse nulle selon laquelle les données suivent une loi normale).

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

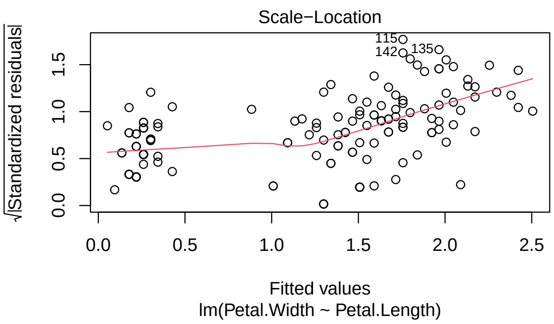
```
plot(modele_lineaire_petale)
```



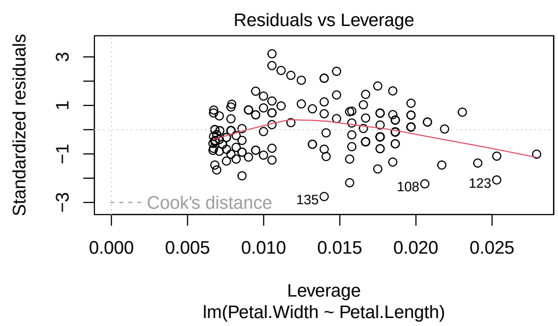
(a) La courbe rouge doit être la plus proche de la droite en pointillée



(a) Les points doivent suivre la première diagonale en pointillée



(a) La courbe rouge doit être la plus plate possible



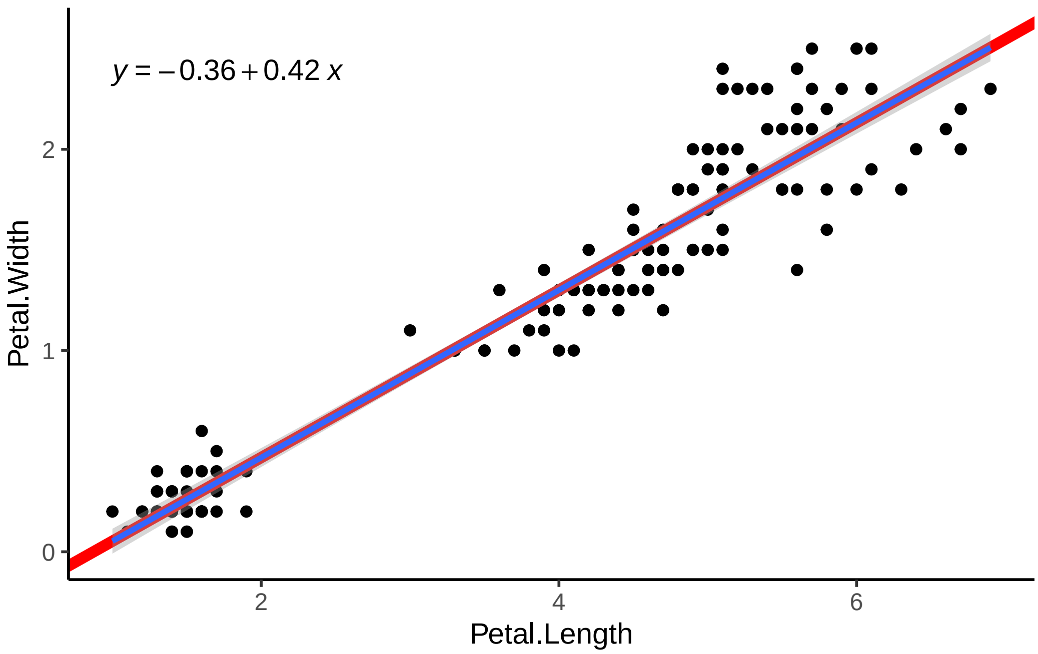
(a) La courbe rouge doit être proche de la droite horizontale en pointillée

6.5 5ème étape : Réalisation d'un graphique résumé

Le nuage de points avec une droite est la meilleur représentation.
La droite peut-être réalisé grâce à la fonction `geom_abline()` du package `{ggplot2}` et les paramètres du modèle linéaire ajusté (`modele_lineaire_petale`) ou automatiquement avec la fonction `geom_smooth()` du même package en précisant l'argument `method = "lm"`.

L'équation est affiché sur le graphique grâce à la fonction `stat_regline_equation()` du package `{ggpubr}`.

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



7 En savoir un peu plus sur moi

Bonjour,

Je suis Marie Vaugoyeau et je suis disponible pour des **missions en freelance** d'**accompagnement à la formation** à R et à l'analyse de données et/ou en **pro-**

grammation (reprise de scripts, bonnes pratiques de codage, développement de package). Ayant un **bagage recherche en écologie**, j'ai accompagné plusieurs chercheuses en biologie dans leurs analyses de données mais je suis ouverte à d'autres domaines.

Vous pouvez retrouver mes offres ici.

En plus de mes missions de consulting je diffuse mes savoirs en R et analyse de données sur plusieurs plateformes :

- J'ai écrit un **livre** aux éditions ENI

- Tous les mois je fais un **live sur Twitch** pour parler d'un package de R, d'une analyse

- Je rédige une **newsletter** de manière irrégulière pour parler de mes **inspirations** et transmettre **des trucs et astuces sur R**. Pour s'y inscrire, c'est par là. J'ai aussi un **blog** sur lequel vous pourrez retrouver une version de cet article.

Pour en savoir encore un peu plus sur moi, il y a LinkedIn et pour retrouver tous ces liens et plus encore, c'est ici

N'hésitez pas à me contacter sur marie.vaugoyeau@gmail.com !

Bonne journée

Marie