TP 5 – SY02
Régression linéaire
Corrigé

Les questions/sections marquées par un  sont des questions qui sont prévues pour être traitées en autonomie en dehors de la séance de TP.

En R, pour réaliser une régression linéaire, on appelle la fonction `lm` (*linear model*). Le premier argument de `lm` est un nouvel objet `R` qu'on appelle une formule et qui spécifie une « sortie » et des « entrées » séparées par le signe `~`. Les entrées et sortie sont des noms de colonnes d'un `data.frame` qu'il faut spécifier en deuxième argument. Par exemple, si on veut réaliser la régression des données `vary` en fonction des données `varx`, on écrira

```
donnees <- data.frame(varx = c(0, 0.2, 0.3, 0.6),
                      vary = c(1.01, 1.44, 1.55, 2.1))
lm(vary~varx, data = donnees)
```

On fera attention à l'ordre des éléments dans une formule. La variable à régresser se situe à gauche, le ou les régresseurs à droite.

(1) Quelles sont les estimations de l'ordonnée à l'origine (*intercept*) $\widehat{a}$ et de la pente $\widehat{b}$?

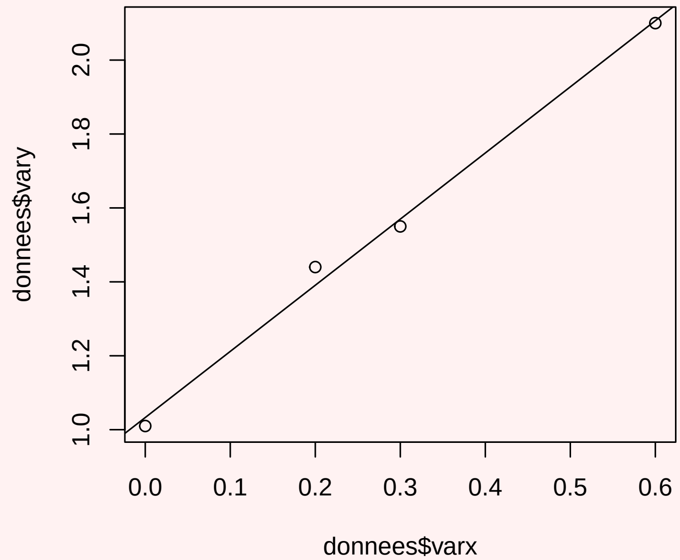
```
donnees <- data.frame(varx = c(0, 0.2, 0.3, 0.6), vary = c(1.01, 1.44, 1.55, 2.1))
lm(vary ~ varx, data = donnees)
Call:
lm(formula = vary ~ varx, data = donnees)

Coefficients:
(Intercept)         varx
      1.033        1.789
```

On trouve $\widehat{a} = 1.0329333$ et $\widehat{b} = 1.7893333$

(2) À l'aide des fonctions `plot` et `abline`, tracer les points de coordonnées `x` et `y` ainsi que la droite des moindres carrés.

```
plot(donnees$varx, donnees$vary)
m <- lm(vary ~ varx, data = donnees)
a <- m$coefficients[1]
b <- m$coefficients[2]
```

```
abline(a, b)
```



Pour avoir plus d'informations sur la régression effectuée, il faut stocker l'objet renvoyé par la fonction `lm` dans une variable et appeler la fonction `summary` avec cette variable en argument.

Toutes les données affichées par `summary` sont accessibles programmatiquement (voir la table 1 pour quelques exemples)

3 À l'aide des correspondances indiquées dans la table 1, vérifier que la somme des résidus vaut 0 et que l'image de $\overline{x}$ par la droite des moindres carrés est $\overline{y}$.

```
sum(m$residuals)
[1] -4.336809e-18
(a + b * mean(donnees$varx))
(Intercept)
      1.525
mean(donnees$vary)
[1] 1.525
```

Notations	Code R
x_i	`x`
y_i	`y`
$\overline{x}$	`mean(x)`
$\overline{y}$	`mean(y)`
$\widehat{y}_i$	`m$fitted.values`
$y_i - \widehat{y}_i$	`m$residuals`
$\widehat{a}$	`m$coefficients[1]`
$\widehat{b}$	`m$coefficients[2]`

Table 1 – Correspondances notations/code R, où `m` est l'objet renvoyé par la fonction `lm`

1 Qualité de l'ajustement

1.1 Équation d'analyse de la variance

(4) À l'aide des correspondances indiquées dans la table 1, calculer/vérifier successivement

1. la variance totale
$$S_Y^2 = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2;$$
2. la variance expliquée par le régression
$$S_{\text{reg}} = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{Y}_i - \overline{Y}\right)^2;$$
3. la variance résiduelle
$$S_{\text{res}} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2;$$
4. la variance totale est égale à la somme de la variance expliquée par la régression et de la variance résiduelle,
5. R^2, le **coefficient de détermination** qui est égal à la proportion de la variance expliquée dans la variance totale soit :
$$R^2 = \frac{S_{\text{reg}}}{S_Y^2};$$
vérifier que R^2 est égal au carré du coefficient de corrélation de Pearson entre les observations y_i et les prédictions $\widehat{y}_i$.

```
(SY2 <- mean((donnees$vary - mean(donnees$vary))^2))
[1] 0.150925
(Sreg <- mean((m$fitted.values - mean(donnees$vary))^2))
[1] 0.1500803
(Sres <- mean(m$residuals^2))
[1] 0.0008446667
Sres + Sreg
[1] 0.150925
```

On retrouve bien $S_Y^2 = S_{\text{reg}} + S_{\text{res}}$.

```
Sreg/SY2
[1] 0.9944034
summary(m)
```

```
Call:
lm(formula = vary ~ varx, data = donnees)

Residuals:
        1         2         3         4
-0.022933  0.049200 -0.019733 -0.006533

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.03293    0.03322   31.09  0.00103 **
varx         1.78933    0.09492   18.85  0.00280 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0411 on 2 degrees of freedom
Multiple R-squared:  0.9944,	Adjusted R-squared:  0.9916
F-statistic: 355.4 on 1 and 2 DF,  p-value: 0.002802
```

On retrouve le coefficient de détermination sous le vocable `Multiple R-squared`.

```
cor(donnees$varx, donnees$vary, method = "pearson")^2
[1] 0.9944034
cor(donnees$vary, m$fitted.values, method = "pearson")^2
[1] 0.9944034
```

Le coefficient de détermination est bien le carré du coefficient de corrélation de Pearson.

1.2 Homoscédasticité, indépendance et normalité des résidus

Le coefficient de détermination est insuffisant pour rendre compte de la qualité de l'ajustement. À titre d'exemple, on utilise le jeu de données d'Anscombe qui consiste en 4 ensembles de 11 points du plan décrits à la figure 1. Pour rendre directement disponibles les colonnes en tapant leur nom, on pourra « attacher » ce jeu de données avec l'instruction

```
attach(anscombe)
```

Dès lors, au lieu de spécifier le jeu de données puis le nom de colonne

```
anscombe$x1
```

on peut se contenter de spécifier `x1`.

De même, pour éviter de définir systématiquement un `data.frame` avant de l'utiliser dans `lm`, on peut utiliser directement des vecteurs dans des formules. On peut alors simplement écrire

```
lm(y1 ~ x1)
```

Attention, cette écriture rend impossible la prédiction en de nouveaux points. On préférera donc utiliser la syntaxe détaillée en début de TP (syntaxe qui précède la question 1) lorsqu'il sera nécessaire de faire de la prédiction.

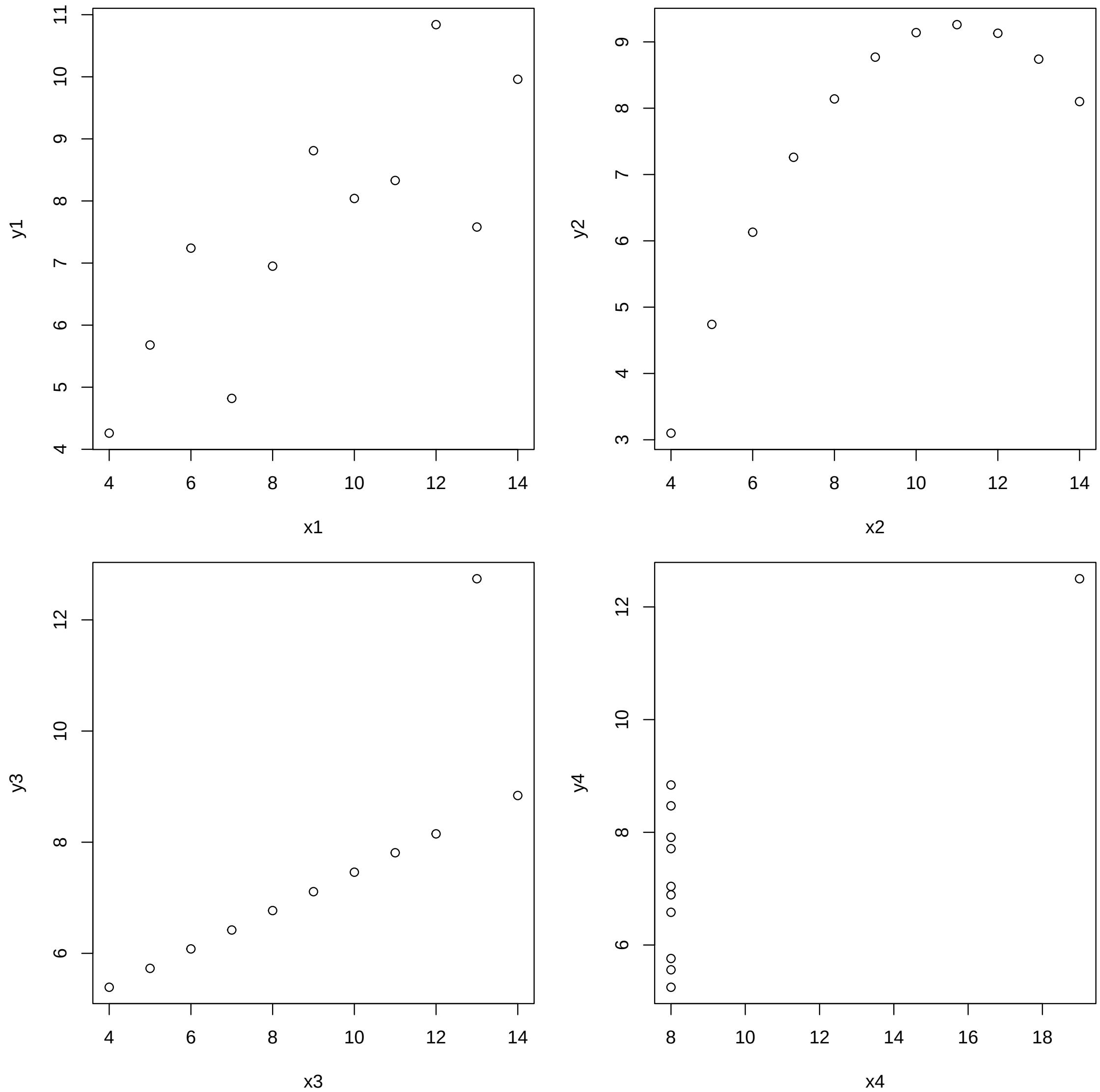


Figure 1 – Diagrammes de dispersion des 4 ensembles de 11 points du jeu de données d'Anscombe

(5) Effectuer les régressions linéaires sur les 4 ensembles de points. Que remarquez-vous ?

```
rl1 <- lm(y1 ~ x1)
summary(rl1)
```

```
Call:
lm(formula = y1 ~ x1)

Residuals:
     Min       1Q   Median       3Q      Max
-1.92127 -0.45577 -0.04136  0.70941  1.83882

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0001     1.1247   2.667  0.02573 *
x1            0.5001     0.1179   4.241  0.00217 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared:  0.6665,	Adjusted R-squared:  0.6295
F-statistic: 17.99 on 1 and 9 DF,  p-value: 0.00217
```

```
rl2 <- lm(y2 ~ x2)
summary(rl2)
Call:
lm(formula = y2 ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9009 -0.7609  0.1291  0.9491  1.2691

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.001      1.125   2.667  0.02576 *
x2             0.500      0.118   4.239  0.00218 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared:  0.6662,	Adjusted R-squared:  0.6292
F-statistic: 17.97 on 1 and 9 DF,  p-value: 0.002179
```

```
rl3 <- lm(y3 ~ x3)
summary(rl3)
Call:
lm(formula = y3 ~ x3)

Residuals:
    Min      1Q  Median      3Q     Max
-1.1586 -0.6146 -0.2303  0.1540  3.2411

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0025     1.1245   2.670  0.02562 *
x3            0.4997     0.1179   4.239  0.00218 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.236 on 9 degrees of freedom
Multiple R-squared:  0.6663,	Adjusted R-squared:  0.6292
F-statistic: 17.97 on 1 and 9 DF,  p-value: 0.002176
```

```
rl4 <- lm(y4 ~ x4)
summary(rl4)
```

```
Call:
lm(formula = y4 ~ x4)

Residuals:
   Min     1Q Median     3Q    Max
-1.751 -0.831  0.000  0.809  1.839

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0017     1.1239   2.671  0.02559 *
x4            0.4999     0.1178   4.243  0.00216 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.236 on 9 degrees of freedom
Multiple R-squared:  0.6667,	Adjusted R-squared:  0.6297
F-statistic:    18 on 1 and 9 DF,  p-value: 0.002165
```

Les valeurs de $\widehat{a}$, $\widehat{b}$ et R^2 coincident pour les 4 jeux de données alors qu'ils sont très différents : une régression linéaire est justifiée pour certains d'entre eux alors qu'elle est visiblement inadaptée pour les autres. Le coefficient R^2, même s'il est proche de 1 ne garantit absolument pas un bon ajustement.

Le modèle de régression linéaire fait les hypothèses suivantes :

1. Lin : la relation entre y_i et x_i est linéaire
2. Norm : normalité des résidus (induite par la normalité des termes d'erreurs)
3. Ind : indépendance des résidus (induite par l'indépendance des termes d'erreurs)
4. Hom : homoscédasticité : $\forall i,\ \mathrm{Var}(\epsilon_i) = \sigma^2$

Pour estimer la qualité de l'ajustement, on utilise quelques diagnostics graphiques qui permettent de vérifier empiriquement la validité de ces hypothèses (cf section 7.5 du poly de cours).

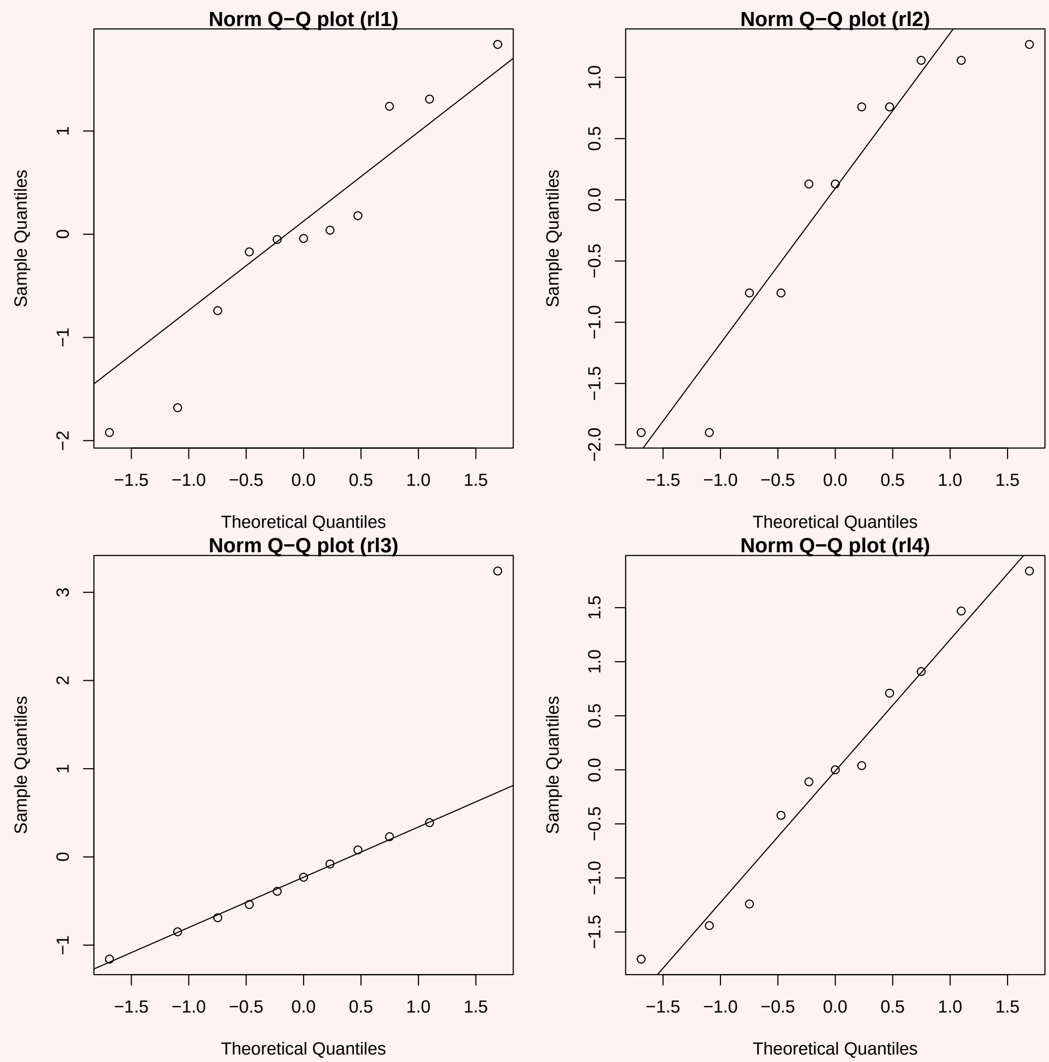
6 Faire une analyse des résidus et discuter de la validité des hypothèses Norm, Hom et éventuellement des hypothèses Lin et Ind pour l'ensemble des régressions linéaires proposées par le jeu de données Anscombe.

En particulier, on tracera

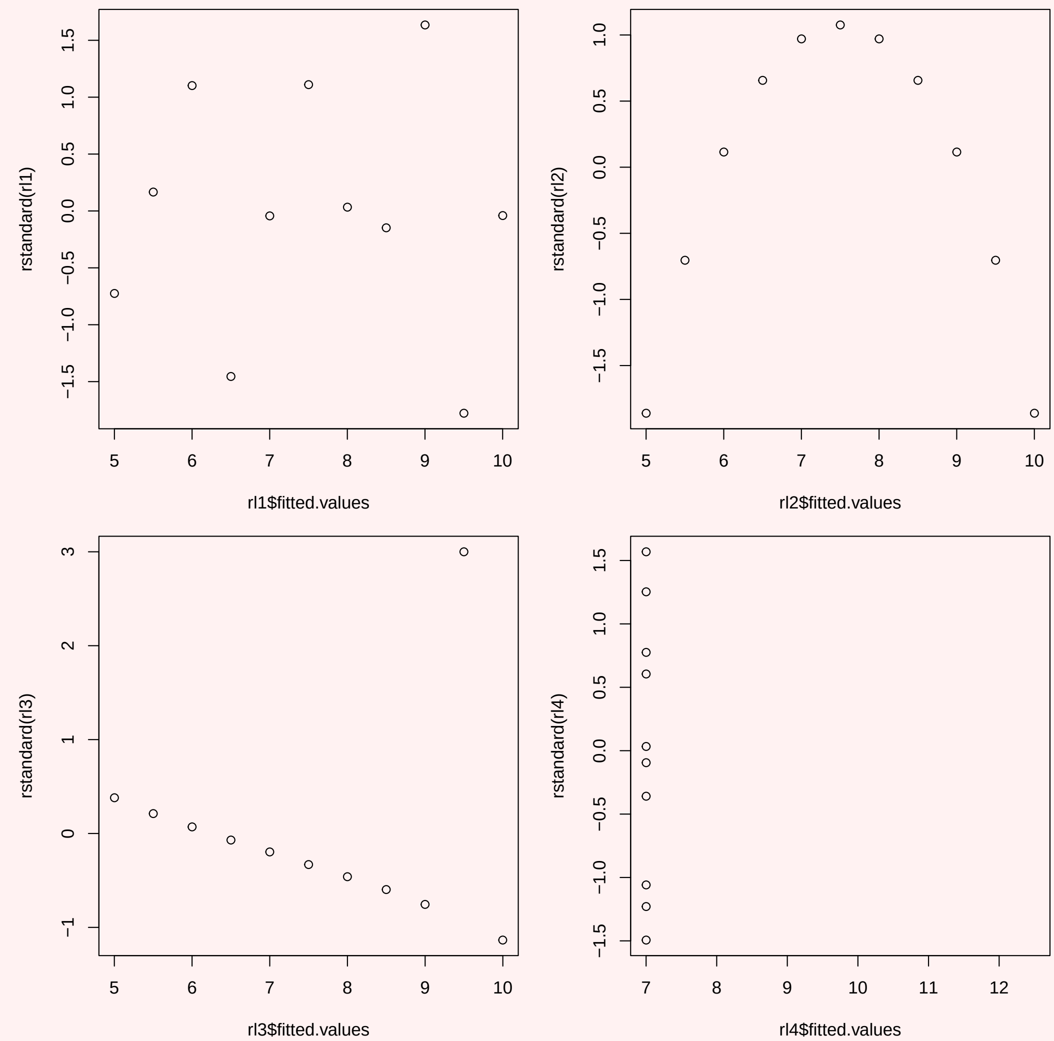
— Pour Norm, le diagramme quantile–quantile des résidus avec les fonctions `qqnorm` et `qqline` (on pourra également tracer l'histogramme des résidus corrigés et y superposer la densité d'une normale d'espérance leur moyenne empirique et d'écart-type, leur écart-type empirique) ;

— Pour Hom et éventuellement Ind et Lin), les résidus standardisés (`rstandard`) en fonction des prédictions (`fitted.values`) ou bien en fonction des valeurs de la variable explicative ;

```r
qqnorm(rl1$residuals, main = "Norm Q-Q plot (rl1)")
qqline(rl1$residuals)
qqnorm(rl2$residuals, main = "Norm Q-Q plot (rl2)")
qqline(rl2$residuals)
qqnorm(rl3$residuals, main = "Norm Q-Q plot (rl3)")
qqline(rl3$residuals)
qqnorm(rl4$residuals, main = "Norm Q-Q plot (rl4)")
qqline(rl4$residuals)
```




```
plot(rl1$fitted.values, rstandard(rl1))
plot(rl2$fitted.values, rstandard(rl2))
plot(rl3$fitted.values, rstandard(rl3))
plot(rl4$fitted.values, rstandard(rl4))
```



On validera graphiquement l'hypothèse d'homoscédasticité si les résidus corrigés se répartissent aléatoirement le long de la droite d'équation $y = 0$ et si les écarts des résidus par rapport à cette droite "$y = 0$" restent constants le long de l'axe des abscisses.

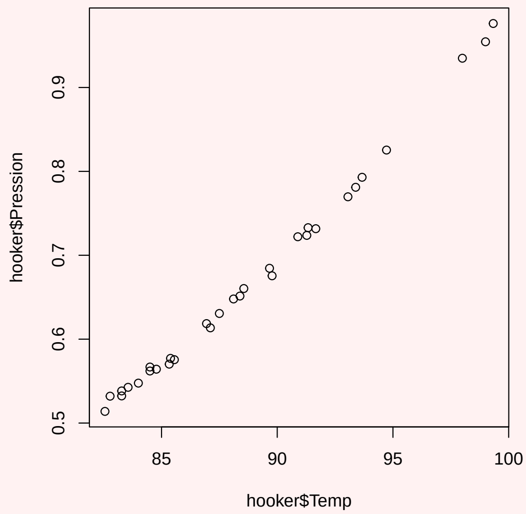
Ce graphe peut également être utile pour détecter une dépendance des résidus ou pour invalider l'hypothèse de linéarité ; par exemple, c'est le cas si visuellement, les résidus forment une structure particulière, comme pour les régressions 2, 3 et 4 du jeu de données.

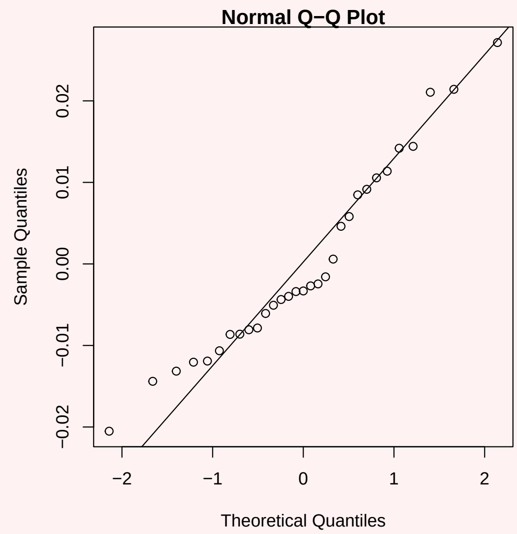
2 Prédiction

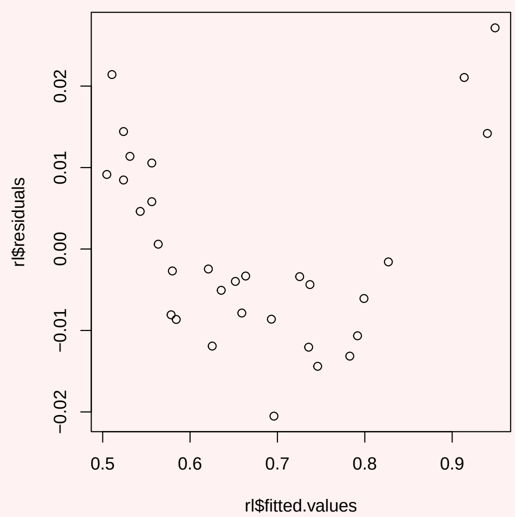
Le fichier `hooker-data.data` contient un jeu de données recueillies par le botaniste anglais Joseph Dalton Hooker. Il s'agit de températures d'ébullition de l'eau relevées pour différentes altitudes. Dans cette section, il est indispensable d'effectuer la régression linéaire en utilisant la syntaxe détaillée en début de TP (celle située avant la question 1).

7 Faire une étude de régression linéaire qui explique la pression atmosphérique.

```
hooker <- read.csv("data/hooker-data.data")
plot(hooker$Temp, hooker$Pression)
rl <- lm(Pression ~ Temp, data = hooker)
summary(rl)$r.squared
[1] 0.9917775
qqnorm(rl$residuals)
qqline(rl$residuals)
plot(rl$fitted.values, rl$residuals)
```







(8) À l'aide de la function `confint`, donner un intervalle de confiance sur les coefficients de la droite des moindres carrés au niveau de confiance $1 - \alpha = 0.99$.

```
confint(rl, level = 0.99)
                   0.5 %     99.5 %
(Intercept) -1.79160707 -1.5721316
Temp         0.02525198  0.0277208
```

(9) À l'aide de la fonction `predict`, calculer un intervalle de confiance sur la pression pour une température d'ébullition mesurée de 97 °C.

Pour plus d'informations sur les arguments à fournir à la fonction `predict`, on pourra utiliser l'instruction suivante

```
?predict.lm
```

Il faut fournir à la fonction `predict` l'objet retourné par la fonction `lm`, ainsi qu'un autre argument nommé `newdata` qui est un `data.frame` qui stocke les points où on désire faire une prédiction (attention, les noms de colonnes de `newdata` doivent coïncider avec les noms de colonnes du jeu de données de départ).

```
newdata <- data.frame(Temp = c(97, 100))
predict(rl, newdata, interval = "confidence")
        fit       lwr       upr
1 0.8873108 0.8785957 0.8960259
2 0.9667700 0.9555930 0.9779470
```

3 Étude de cas

Loi de Moore

La loi de Moore est une loi empirique qui dit que le nombre de transistors croit de manière exponentielle avec le temps. Autrement dit, on suppose que le nombre de transistors N_t au temps t est égal à

$$N_t = \alpha \exp(\beta t).$$

(10) À l'aide du fichier moore-data.data et en utilisant une régression linéaire, estimer les paramètres α et β et donner un intervalle de confiance et de prédiction sur N_{2018}. Retrouver le fait que le nombre de transistors double tous les 2 ans.

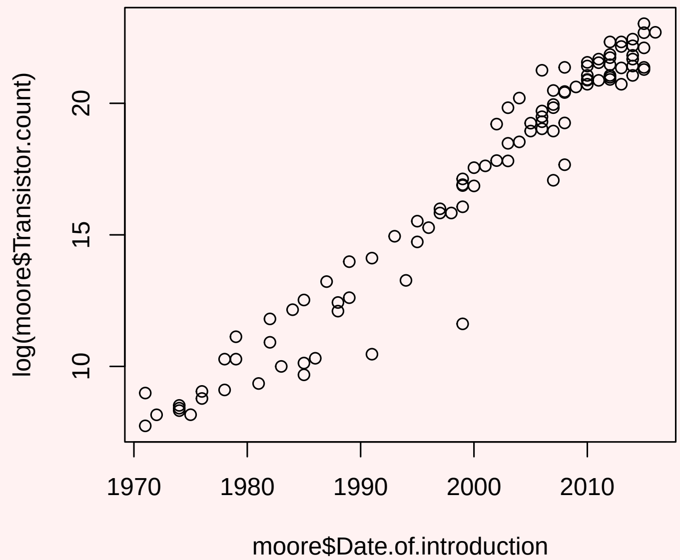
```
moore <- read.csv("data/moore-data.data")
head(moore)
            Processor Transistor.count Date.of.introduction          Designer
1            TMS 1000             8000                 1971 Texas Instruments
2          Intel 4004             2300                 1971             Intel
3          Intel 8008             3500                 1972             Intel
4 MOS Technology 6502             3510                 1975    MOS Technology
5       Motorola 6800             4100                 1974          Motorola
6          Intel 8080             4500                 1974             Intel
  Process Area
1    8000   NA
2   10000   12
3   10000   14
4    8000   21
5    6000   16
6    6000   20
```

Les colonnes intéressantes sont `Transistor.count` et `Date.of.introduction`.
Pour avoir une dépendance linéaire, on passe au logarithme. On obtient donc

$$\log N_t = \log \alpha + \beta t.$$

D'où la régression

```
plot(moore$Date.of.introduction, log(moore$Transistor.count))
```




```
rl.moore <- lm(log(Transistor.count) ~ Date.of.introduction, data = moore)
summary(rl.moore)
Call:
lm(formula = log(Transistor.count) ~ Date.of.introduction, data = moore)

Residuals:
    Min      1Q  Median      3Q     Max
-5.1301 -0.3279  0.1730  0.5220  2.0624

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)          -681.23873   15.75785  -43.23   <2e-16 ***
Date.of.introduction    0.34917    0.00788   44.31   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.049 on 100 degrees of freedom
Multiple R-squared:  0.9515,	Adjusted R-squared:  0.9511
F-statistic:  1963 on 1 and 100 DF,  p-value: < 2.2e-16
```

```
(IC <- confint(rl.moore, "Date.of.introduction"))
                         2.5 %    97.5 %
Date.of.introduction 0.3335336 0.3648017
exp(predict(rl.moore, newdata = data.frame(Date.of.introduction = c(2018)), interval =
↪  "confidence"))
          fit         lwr         upr
1 14271405422 10022931160 20320703543
exp(predict(rl.moore, newdata = data.frame(Date.of.introduction = c(2018)), interval =
↪  "prediction"))
          fit        lwr          upr
1 14271405422 1728651154 117821928510
```

Il faut trouver la durée T telle que $N_{t+T} = 2N_t$ c'est à dire

$$\frac{\exp(\beta(t+T))}{\exp(\beta t)} = 2,$$

d'où $T = \dfrac{\log 2}{\beta}$.

```
log(2)/IC
                        2.5 %   97.5 %
Date.of.introduction 2.078193 1.900066
```

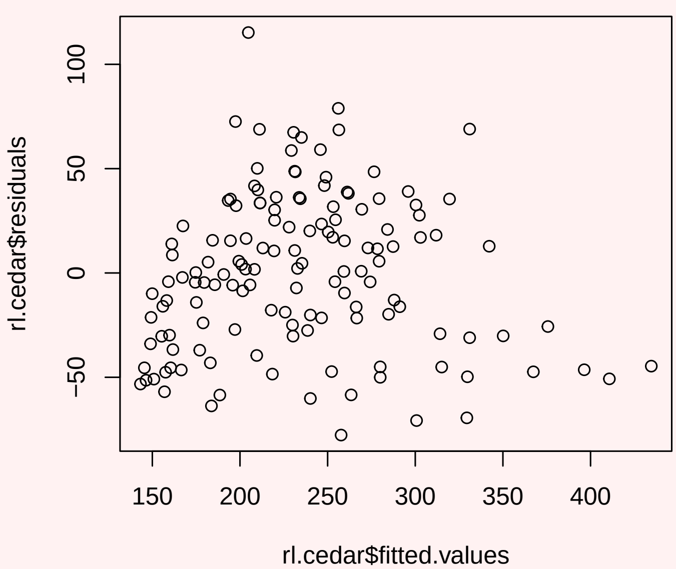
On trouve bien une période de 2 ans.

Hauteur et diamètre de cèdres

Le fichier cedar-data.data contient le diamètre et la hauteur de 139 cèdres. On cherche à prédire la hauteur d'un cèdre en fonction de leur diamètre.

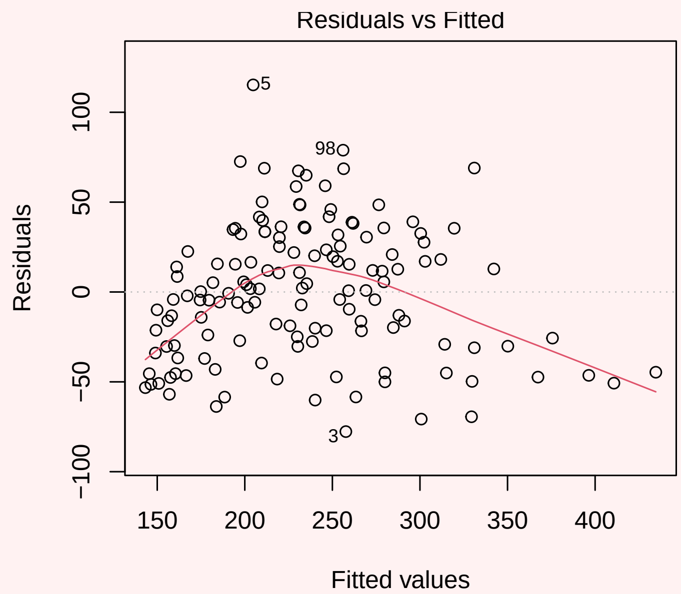
(11) Faites l'étude de la régression linéaire expliquant la hauteur en fonction du diamètre. Analyser le diagramme des résidus. Que remarquez-vous ?

```
cedar <- read.csv('data/cedar-data.data')
rl.cedar = lm(height ~ diameter, data = cedar)
summary(rl.cedar)
Call:
lm(formula = height ~ diameter, data = cedar)

Residuals:
    Min      1Q  Median      3Q     Max
-77.693 -29.467   0.713  28.959 115.237

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 111.02117    7.45869   14.88   <2e-16 ***
diameter      0.31885    0.01736   18.37   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 37.61 on 137 degrees of freedom
Multiple R-squared:  0.7113,	Adjusted R-squared:  0.7091
F-statistic: 337.5 on 1 and 137 DF,  p-value: < 2.2e-16
plot(rl.cedar$fitted.values, rl.cedar$residuals)
```



Les résidus semble être corrélés en fonction des valeurs prédites. En effet, si on trace l'évolution des résidus en fonction des valeurs prédites, on trouve un profil de type parabole concave alors qu'il faudrait avoir idéalement une fonction constante nulle.

```
plot(rl.cedar, which=1)
```



(12) On souhaite appliquer une transformation sur la variable explicative. On choisit la transformation de Box–Cox définie comme suit

$$f_\lambda(x) = \begin{cases} \dfrac{x^\lambda - 1}{\lambda} & \text{si } \lambda \neq 0 \\ \log(x) & \text{si } \lambda = 0, \end{cases}$$

de sorte que le problème de régression est maintenant

$$\mathbb{E}[Y \mid X] = \alpha + \beta f_\lambda(X).$$

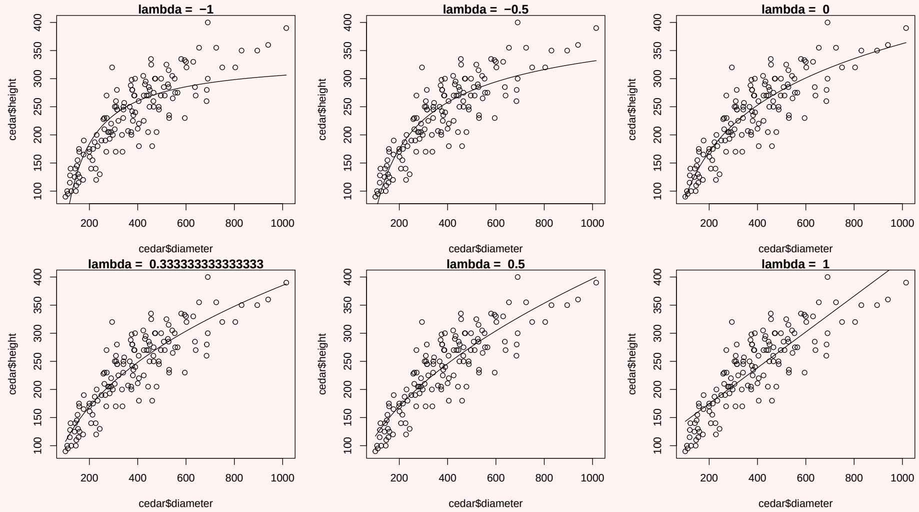
Créer une fonction qui réalise la transformation de Box–Cox.

```
boxcox <- function(x, lambda) {
    if (lambda == 0)
        log(x) else (x^lambda - 1)/lambda
}
```

(13) Parmi les valeurs -1, $-1/2$, 0, $1/3$, $1/2$, 1, trouver la valeur de λ qui semble le mieux expliquer la hauteur des cèdres.

```
for (lambda in c(-1, -1/2, 0, 1/3, 1/2, 1)) {
    plot(cedar$diameter, cedar$height, main = paste("lambda = ", lambda))
    cedar$logdiameter <- boxcox(cedar$diameter, lambda)
    rl.cedar <- lm(height ~ logdiameter, data = cedar)
    print(summary(rl.cedar)$r.squared)
    curve(rl.cedar$coefficients[1] + rl.cedar$coefficients[2] * boxcox(x, lambda), add = TRUE)
}
[1] 0.705868
[1] 0.7530177
```

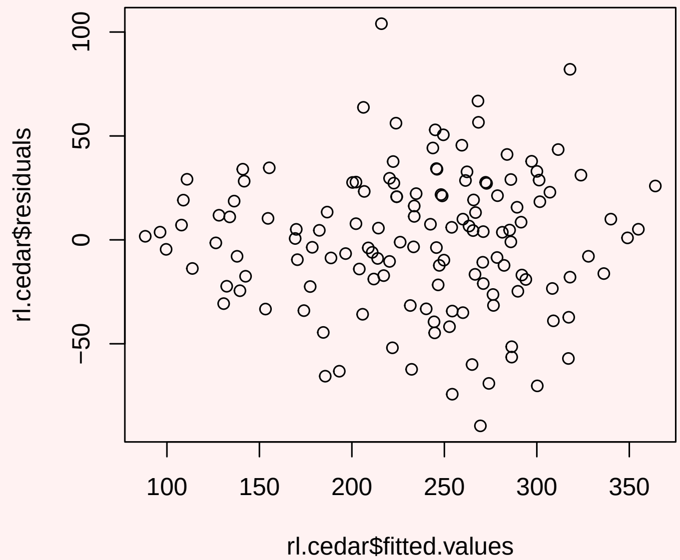
```
[1] 0.7731144
[1] 0.7674284
[1] 0.758663
[1] 0.7112522
```



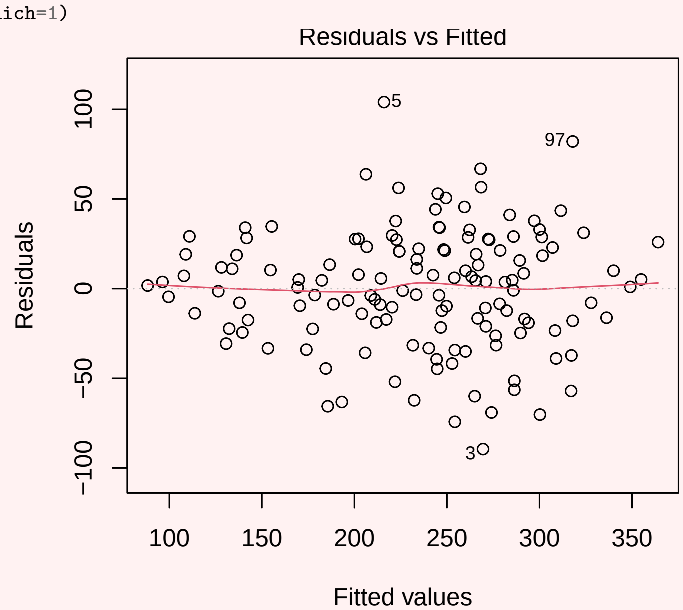
La valeur $\lambda = 0$ qui correspond à une transformation logarithmique donne le coefficient de détermination maximum.

14 Refaites la même étude en appliquant une transformation logarithmique à la variable explicative.

```
cedar <- read.csv("data/cedar-data.data")
rl.cedar <- lm(height ~ log(diameter), data = cedar)
summary(rl.cedar)
Call:
lm(formula = height ~ log(diameter), data = cedar)

Residuals:
    Min      1Q  Median      3Q     Max
-89.485 -20.046   3.652  22.586 104.017

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -463.314     32.438  -14.28   <2e-16 ***
log(diameter)  119.519      5.532   21.61   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 33.33 on 137 degrees of freedom
Multiple R-squared:  0.7731,	Adjusted R-squared:  0.7715
F-statistic: 466.8 on 1 and 137 DF,  p-value: < 2.2e-16
plot(rl.cedar$fitted.values, rl.cedar$residuals)
```




```r
plot(rl.cedar, which=1)
```



L'analyse des résidus est bien plus satisfaisante avec une allure de fonction constante nulle.