TP 5 – SY02
Régression linéaire
Corrigé

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.

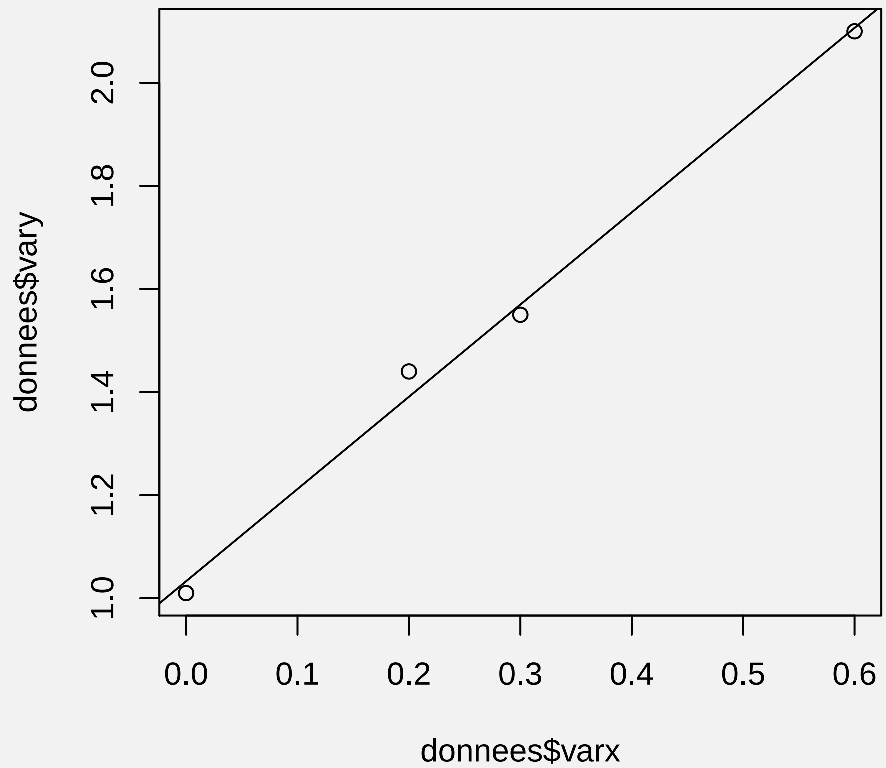
① Quelles sont les estimations de l'ordonnée à l'origine (*intercept*) $\widehat{a}$ et de la pente $\widehat{b}$?

```
>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$residuals)
[1] 0
>(a + b * mean(donnees$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 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
$$S_{\text{res}} = \frac{1}{n} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_i\right)^2.$$

Vérifier que la variance totale est la somme de la variance expliquée par la régression et de la variance résiduelle.

```
>(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 $S_Y^2 = S_{\text{reg}} + S_{\text{res}}$.

(5) On appelle **coefficient de détermination**, noté R^2 la proportion de la variance expliquée dans la variance totale soit :
$$R^2 = \frac{S_{\text{reg}}}{S_Y^2}.$$
Calculer cette quantité et vérifier que c'est en fait le carré du coefficient de corrélation de Pearson entre `x` et `y`.

```
>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
```

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

1.2 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 lorsqu'il sera nécessaire de faire de la prédiction.

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

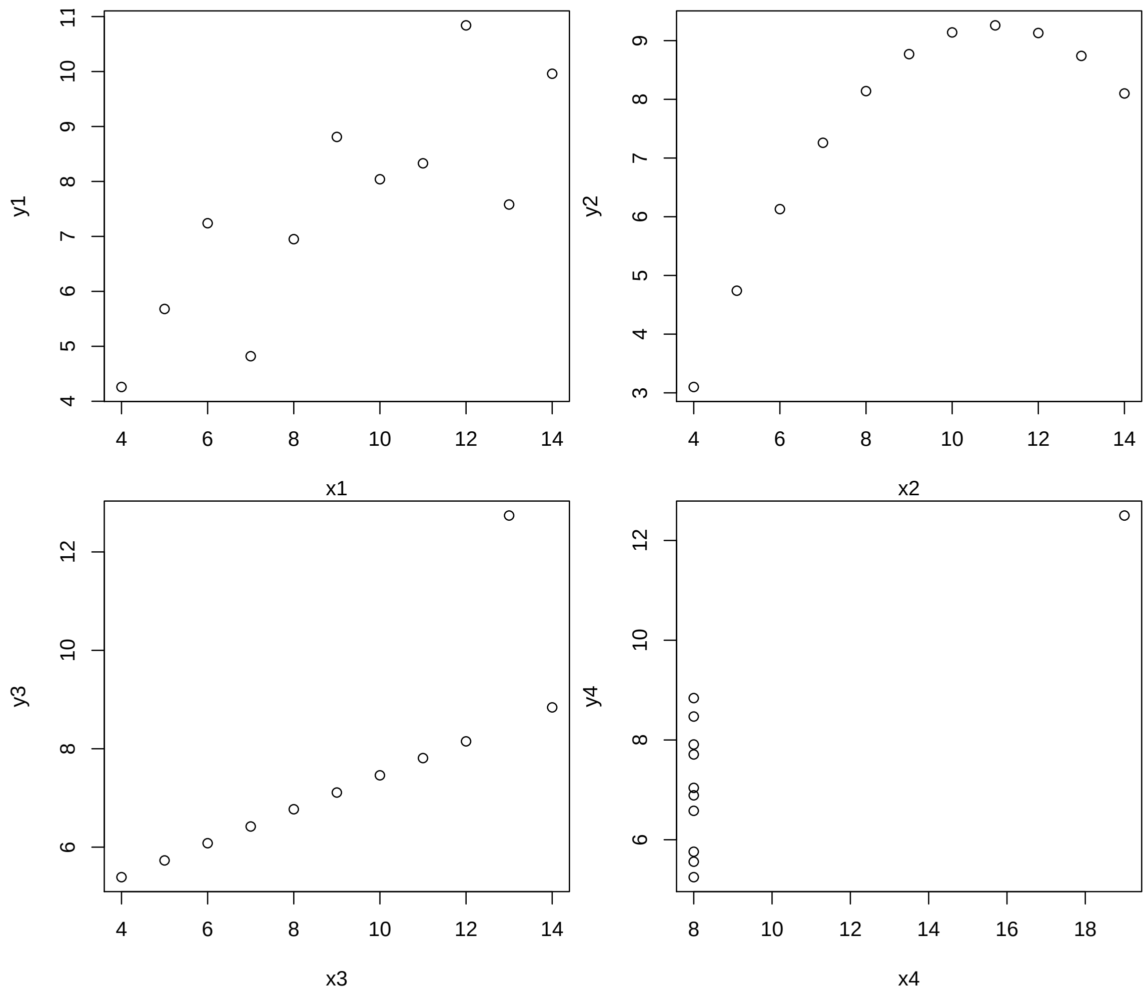


FIGURE 1 – Diagrammes de dispersion des 4 ensembles de 11 points du jeu de données d'Anscombe

```
>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
```

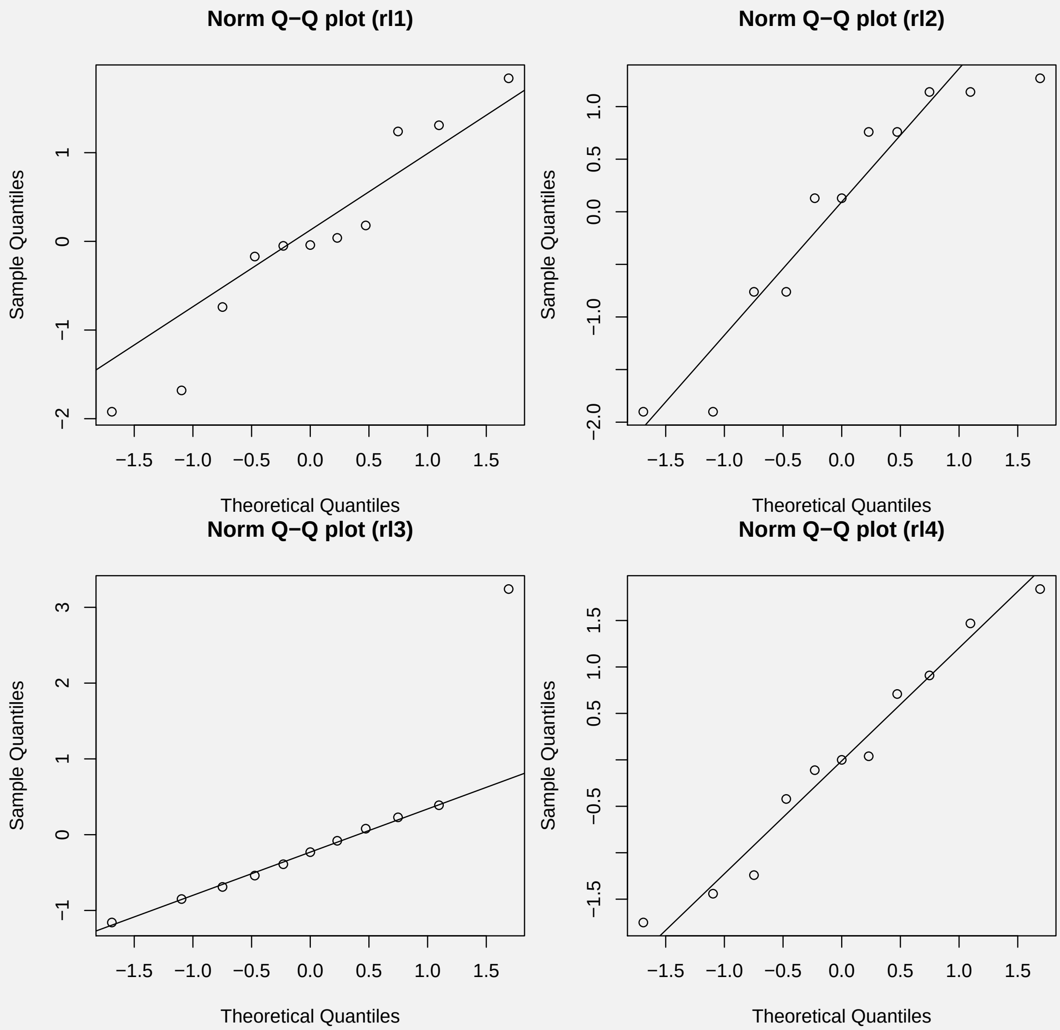
```
>rl4 <- lm(y4 ~ x4)
>summary(rl4)

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.

Pour estimer la qualité de l'ajustement, on utilise le fait que le modèle de régression linéaire suppose que les résidus suivent une loi normale et sont indépendants.

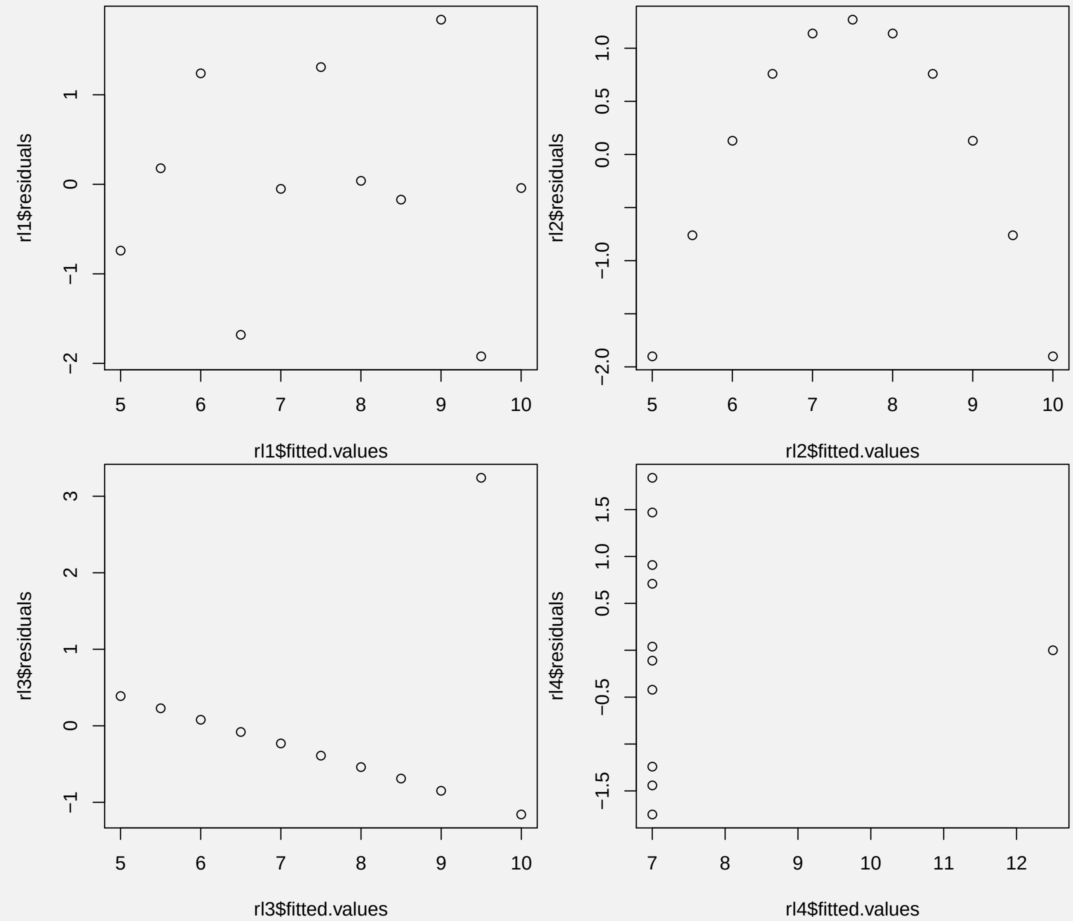
7 Évaluer visuellement la normalité des résidus à l'aide d'un diagramme quantile–quantile avec des fonctions `qqnorm` et `qqline`.

```
>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)
```



8 Évaluer visuellement l'indépendance des résidus en représentant les résidus en fonction des prédictions. (`fitted.values`)

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



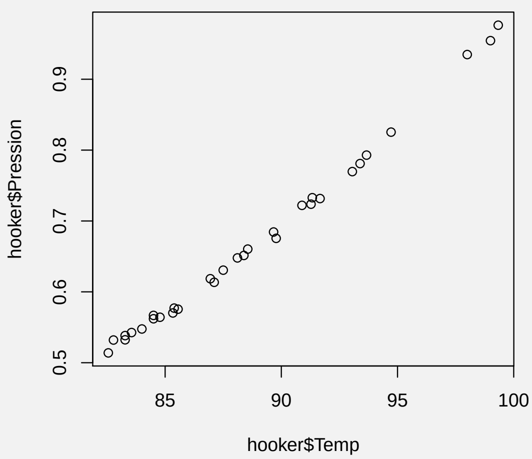
Il faut vérifier que les résidus sont indépendants. Visuellement, il faut vérifier que les résidus sont aléatoires et ne forment pas de structure prévisible. Seule la première régression ne présente pas de structure dans ses résidus.

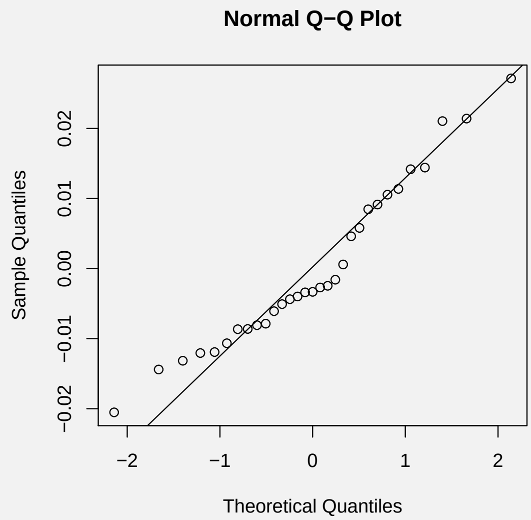
2 Prédiction

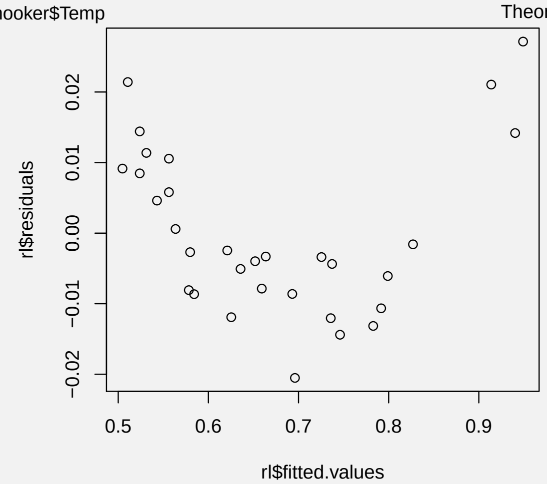
Le fichier hooker-data.data contient un jeu de données recueillies par le botaniste anglais Joseph Dalton Hooker. Il s'agit de températures d'ébullition de l'eau relevées pour différentes altitudes.

9 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)
```







(10) 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
```

(11) 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

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

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

(12) 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.

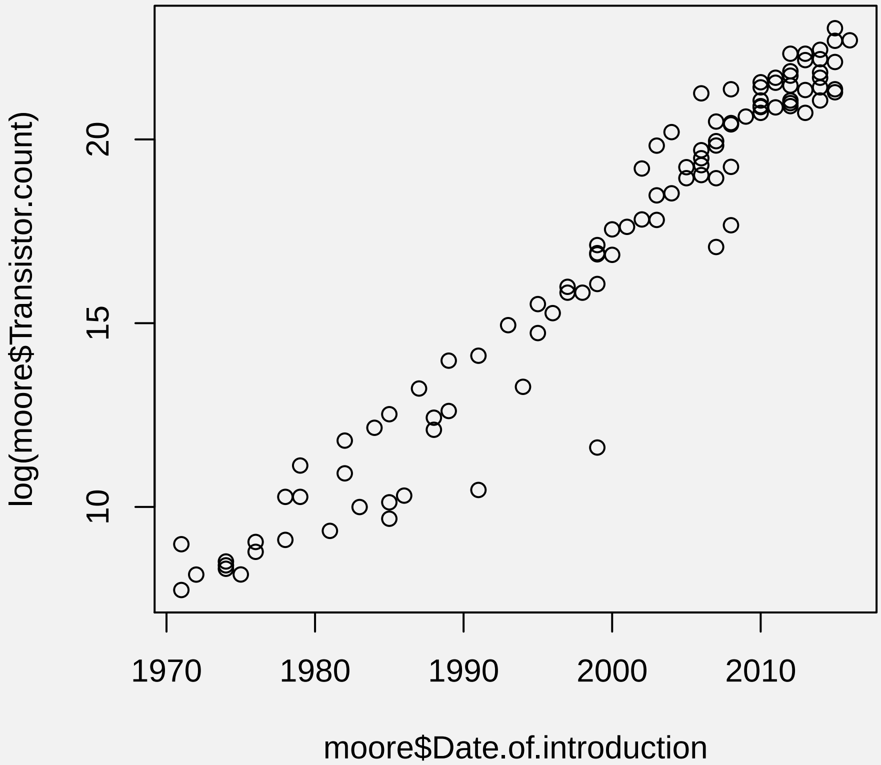
```
>moore <- read.csv("data/moore-data.data")
>head(moore)
            Processor Transistor.count Date.of.introduction
1            TMS 1000             8000                 1971
2          Intel 4004             2300                 1971
3          Intel 8008             3500                 1972
4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