Complément au guide d'utilisation de **R** concernant le modèle linéaire

Marie Laure Delignette-Muller

12 octobre 2020

Ce guide est un complément au guide de base d'utilisation de R.

Table des matières

1 Régression linéaire simple **2**
1.1 Estimation des moindres carrés . . . 2
1.2 Analyse des résidus . . . 3
1.3 Inférence . . . 5
1.4 Cas particulier du modèle avec ordonnée à l'origine nulle . . . 6

2 Régression linéaire multiple **7**
2.1 Estimation des moindres carrés . . . 7
2.2 Analyse des résidus . . . 8
2.3 Analyse des données influentes . . . 10
2.4 Inférence . . . 11
2.5 Comparaison de modèles et sélection de variables . . . 11
2.6 Cas particulier du modèle polynomial . . . 13

3 Lorsque certaines variables explicatives sont qualitatives **15**
3.1 Analyse de variance à 1 facteur . . . 15
3.2 Analyse de variance à 2 facteurs . . . 20
3.3 Analyse de covariance . . . 25

1 Régression linéaire simple

1.1 Estimation des moindres carrés

Les données seront importées sous la forme d'un tableau contenant les deux colonnes correspondant à la variable de contrôle (`Tcoeur` dans cet exemple), et à la variable observée (`log10Nsurv` dans cet exemple).

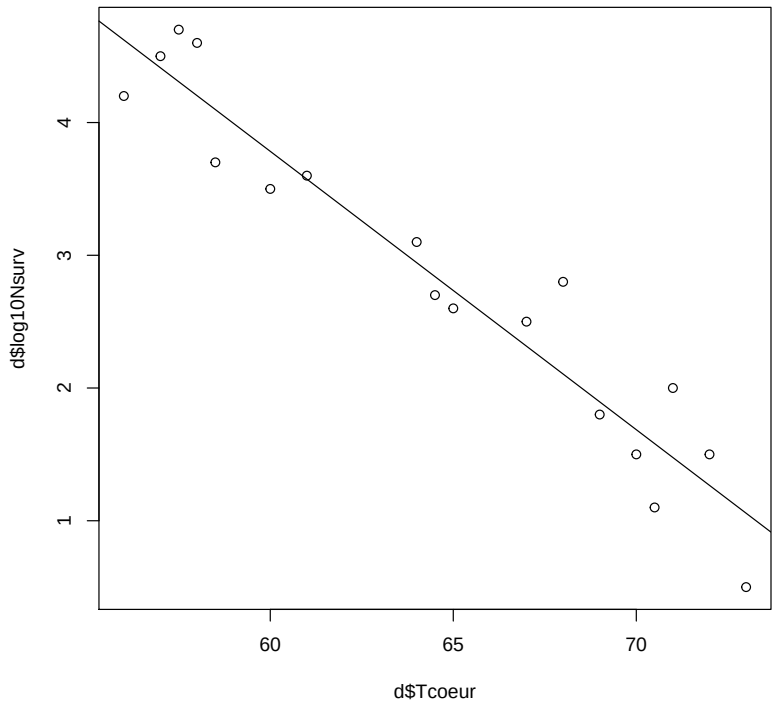
```
d <- read.table("cuissonsteaks.txt", header = TRUE, stringsAsFactors = TRUE)
str(d)

## 'data.frame': 18 obs. of  2 variables:
##  $ Tcoeur    : num  56 57 57.5 58 58.5 60 61 64 64.5 65 ...
##  $ log10Nsurv: num  4.2 4.5 4.7 4.6 3.7 3.5 3.6 3.1 2.7 2.6 ...
```

L'ajustement d'un modèle linéaire aux données par régression peut être réalisé à l'aide de la fonction `lm` (pour "linear model"). Il convient de lui mettre en argument la formule du modèle, composée du nom de la variable observée, suivi du signe $\sim$ et du nom de la variable de contrôle :

```
m <- lm(log10Nsurv ~ Tcoeur, data = d)
summary(m)

##
## Call:
## lm(formula = log10Nsurv ~ Tcoeur, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5568 -0.2589 -0.0346  0.2213  0.6946
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  16.3667     1.0181    16.1  2.7e-11
## Tcoeur       -0.2097     0.0157   -13.3  4.3e-10
##
## Residual standard error: 0.373 on 16 degrees of freedom
## Multiple R-squared:  0.918,Adjusted R-squared:  0.912
## F-statistic:  178 on 1 and 16 DF,  p-value: 4.34e-10
```

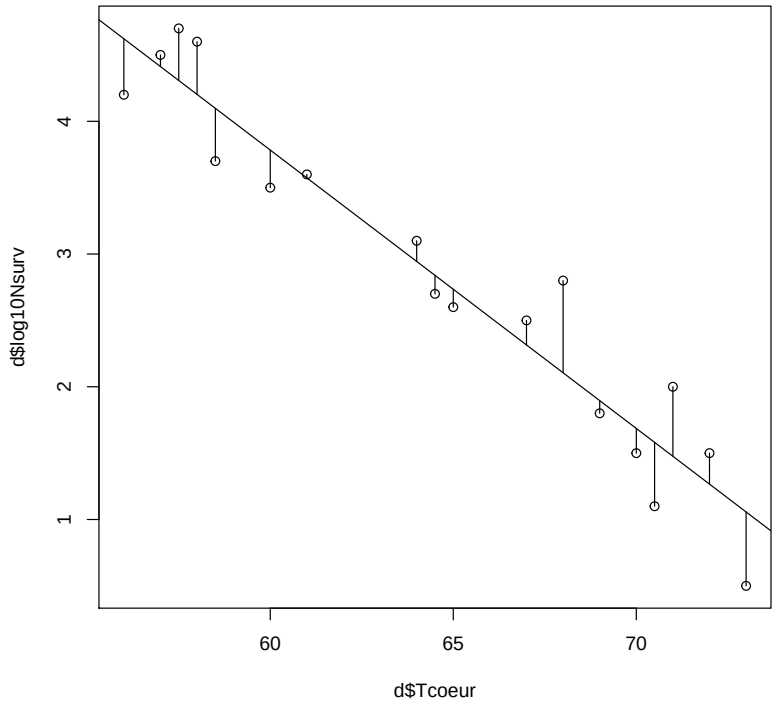
Les données peuvent être représentées sous forme de diagramme de dispersion (ou nuage de points) avec la fonction `plot` en indiquant bien la variable contrôlée comme premier argument. On peut y ajouter la droite ajustée à l'aide de la fonction `abline` comme ci-dessous.

```
plot(d$Tcoeur,d$log10Nsurv)
abline(m)
```



On peut ajouter à ce graphe des segments matérialisant les écarts au modèle, ou résidus, ceux dont la somme des carrés est minimisée.

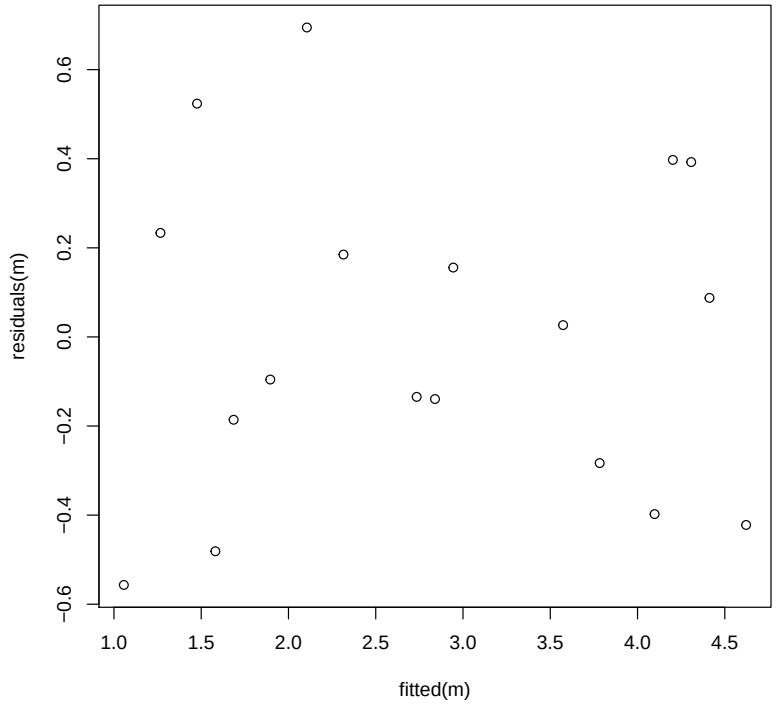
```
plot(d$Tcoeur, d$log10Nsurv)
abline(m)
segments(d$Tcoeur, fitted(m), d$Tcoeur, d$log10Nsurv)
```



1.2 Analyse des résidus

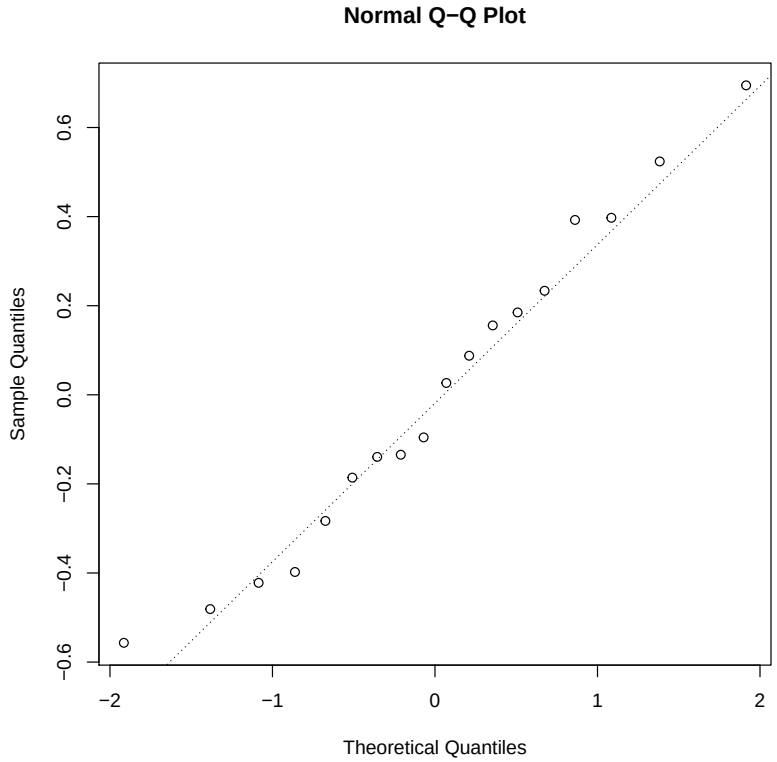
Pour encore mieux visualiser les résidus afin de vérifier les conditions d'utilisation de la régression linéaire, il est d'usage de représenter les résidus en fonction des valeurs prédites par le modèle.

```
plot(residuals(m) ~ fitted(m))
```



Il est fort utile pour vérifier la normalité des résidus de représenter leurs quantiles en fonction des quantiles d'une loi normale.

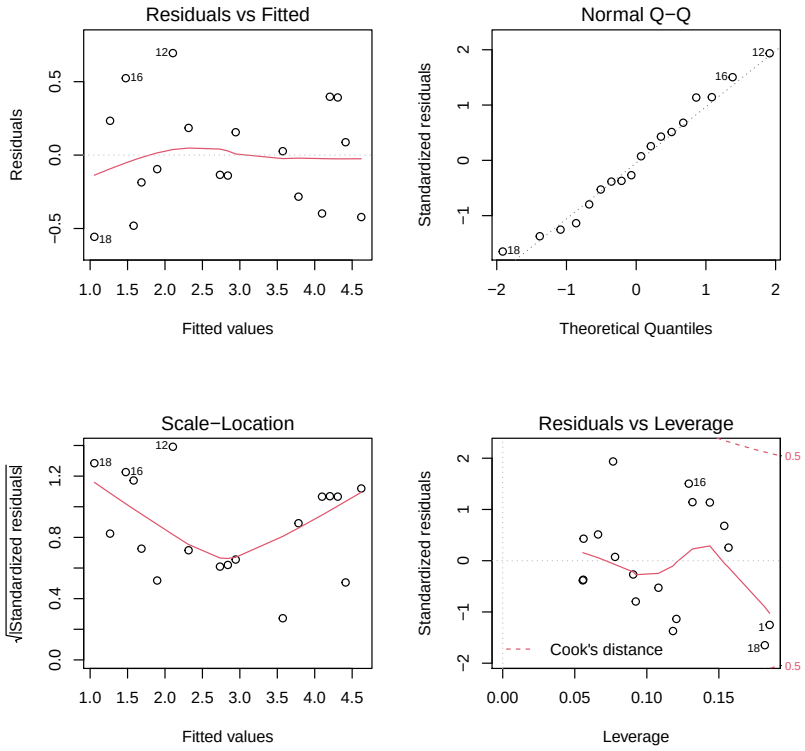
```
qqnorm(residuals(m))
qqline(residuals(m), lty = 3)
```



Si on applique la fonction `plot` à l'objet créé par la fonction `lm` il propose automatiquement 4 graphes de résidus dont les deux premiers correspondant à peu près à ceux présentés ci-dessus. Le troisième permet de mettre en évidence une éventuelle hétéroscédasticité (amplitude des résidus en fonction de la valeur prédite : si la variance est constante la ligne rouge est proche de l'horizontale) et le dernier permet de voir à quel point les données extrêmes sont influentes

(si des points extrêmes sortent des contours des distances de Cook en haut ou en bas à droite, c'est qu'ils sont très influents - pas le cas du tout ici).

```
par(mfrow = c(2,2))
plot(m)
```



On peut éventuellement en complément réaliser sur les résidus le test de normalité de Shapiro-Wilk, **tout en gardant à l'esprit que ce test ne peut à lui seul ni valider les hypothèses de normalité des résidus** (par manque de puissance en cas d'effectif petit) **ni invalider l'utilisation du modèle gaussien** (excès de puissance en cas d'effectif très élevé).

```
shapiro.test(residuals(m))

##
##  Shapiro-Wilk normality test
##
## data:  residuals(m)
## W = 1, p-value = 0.8
```

1.3 Inférence

Si l'on souhaite accéder aux seuls coefficients du modèle ajusté, on peut utiliser la fonction `coef`. De plus la fonction `confint` donne accès directement aux intervalles de confiance sur ces coefficients.

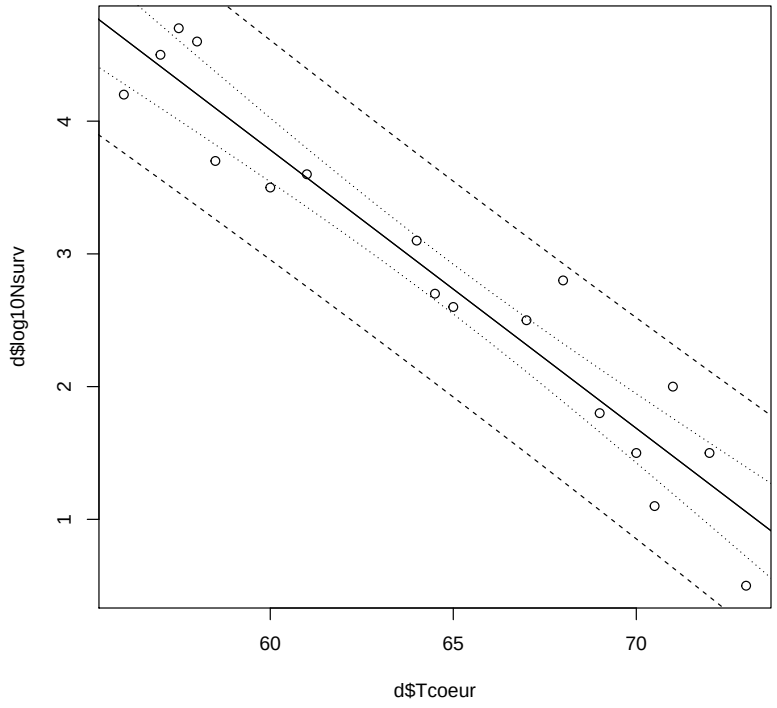
```
coef(m)

## (Intercept)      Tcoeur
##       16.37       -0.21

confint(m)

##               2.5 % 97.5 %
## (Intercept) 14.208 18.525
## Tcoeur      -0.243 -0.176
```

La fonction `predict` peut être utilisée pour prédire la variable dépendante pour une ou une série de valeurs de la variables explicative. Elle permet aussi de calculer les intervalles de confiance ou de prédiction associés à ces prédictions.

```r
predict(m, data.frame(Tcoeur = c(60, 65, 70)))

##    1    2    3
## 3.78 2.73 1.69

predict(m, data.frame(Tcoeur = c(60, 65, 70)), interval = "confidence")

##    fit  lwr  upr
## 1 3.78 3.54 4.02
## 2 2.73 2.55 2.92
## 3 1.69 1.43 1.95

predict(m, data.frame(Tcoeur = c(60, 65, 70)), interval = "prediction")

##    fit   lwr  upr
## 1 3.78 2.956 4.61
## 2 2.73 1.921 3.55
## 3 1.69 0.853 2.52
```

Dans cet exemple on peut représenter les bandes de confiance associées à ces deux types d'intervalles en utilisant le code suivant.

```r
val.des.x <- data.frame(Tcoeur = seq(from = 50, to = 75, by = 1))
ipred <- predict(m, val.des.x, interval = "prediction")
iconf <- predict(m, val.des.x, interval = "confidence")
plot(d$Tcoeur, d$log10Nsurv)
matlines(val.des.x, ipred, lty = c(1, 2, 2), col = "black")
matlines(val.des.x, iconf, lty = c(1, 3, 3), col = "black")
```



1.4 Cas particulier du modèle avec ordonnée à l'origine nulle

Dans certains cas on souhaite forcer la droite à passer par l'origine, c'est-à-dire fixer *a priori* l'ordonnée à l'origine à 0. Il convient d'utiliser le code suivant :

```r
lm(y ~ x - 1, data = nomdujeudedonnees)
```

2 Régression linéaire multiple

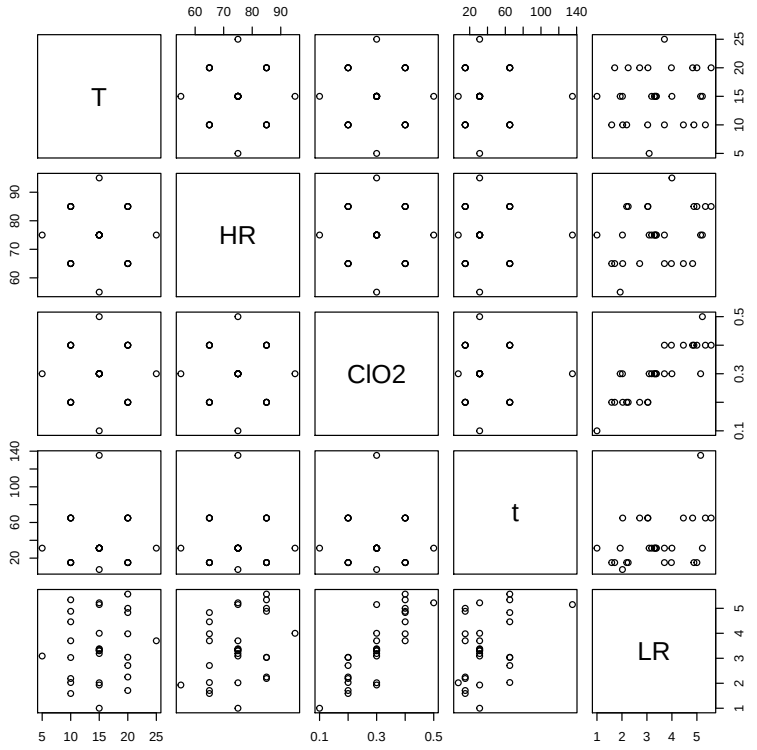
2.1 Estimation des moindres carrés

Les données seront importées sous la forme d'un tableau contenant plusieurs colonnes correspondant aux variables explicatives et à la variable observée.

```
d <- read.table("han2001.txt", header = TRUE, stringsAsFactors = TRUE)
str(d)

## 'data.frame': 29 obs. of  5 variables:
##  $ T   : int  10 20 10 20 10 20 10 20 10 20 ...
##  $ HR  : int  65 65 85 85 65 65 85 85 65 65 ...
##  $ ClO2: num  0.2 0.2 0.2 0.2 0.4 0.4 0.4 0.4 0.2 0.2 ...
##  $ t   : num  15 15 15 15 15 15 15 15 65 65 ...
##  $ LR  : num  1.59 1.71 2.19 2.25 3.7 3.98 4.88 5 2.03 2.71 ...
```

Il peut être intéressant, avant d'entamer la régression linéaire, notamment lorsque l'on dispose d'un grand nombre de variables potentiellement explicatives, d'utiliser la fonction `pairs` pour visualiser toutes les corrélations 2 à 2 entre les variables du jeu de données.

```
pairs(d)
```



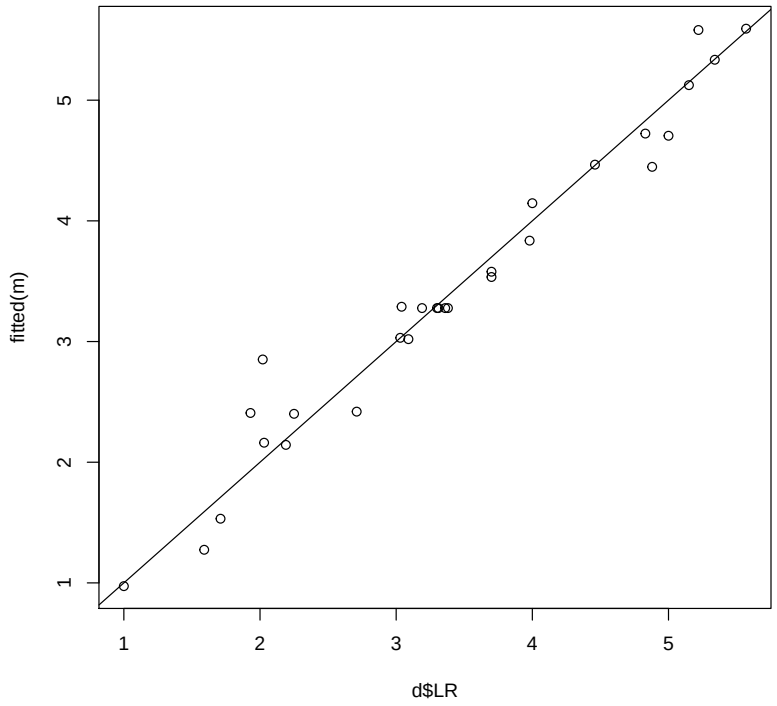
L'ajustement d'un modèle linéaire aux données par régression est réalisé comme en régression simple :

```
m <- lm(LR ~ T + HR + ClO2 + t, data = d)
summary(m)

##
## Call:
## lm(formula = LR ~ T + HR + ClO2 + t, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8313 -0.0872  0.0269  0.1215  0.4323
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.37830    0.48624   -9.00  3.7e-09
```

```
## T            0.02575    0.01110    2.32    0.029
## HR           0.04346    0.00555    7.83  4.6e-08
## ClO2        11.52083    0.55483   20.76  < 2e-16
## t            0.01775    0.00188    9.45  1.5e-09
##
## Residual standard error: 0.272 on 24 degrees of freedom
## Multiple R-squared:  0.961,Adjusted R-squared:  0.954
## F-statistic:  147 on 4 and 24 DF,  p-value: <2e-16
```

Il n'est plus possible de faire un graphe d'ajustement sous forme de droite en régression multiple. Dans certains cas il peut être intéressant de faire un nuage de points avec les données prédites en fonction des données observées et d'y ajouter la bissectrice ($y = x$).

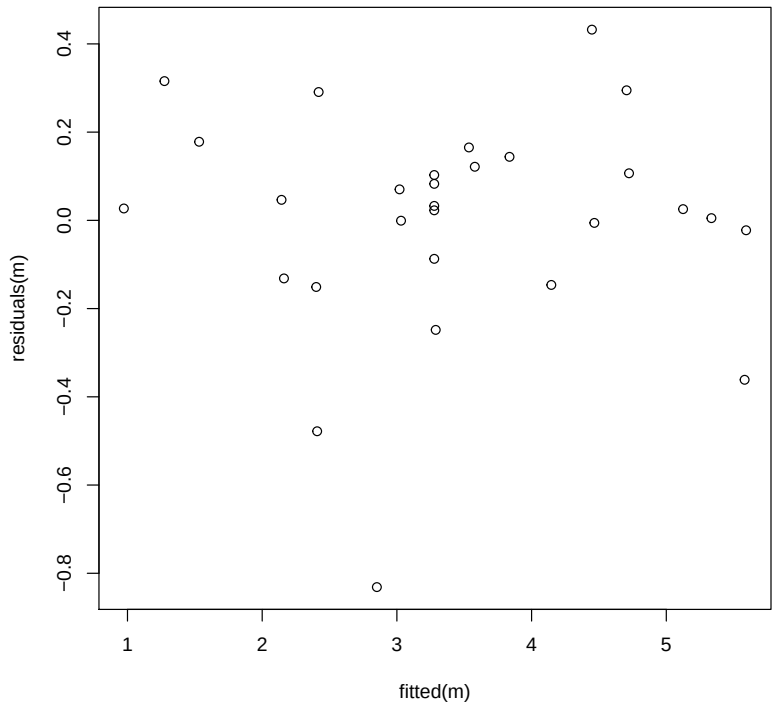
```
plot(fitted(m) ~ d$LR)
abline(a = 0, b = 1)
```



2.2 Analyse des résidus

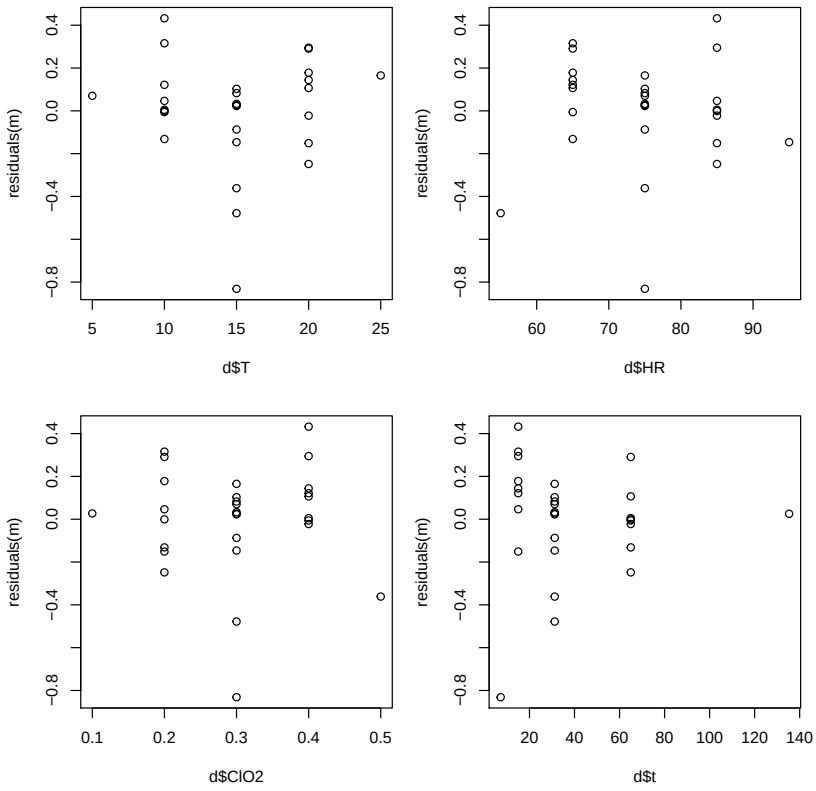
Comme en régression simple on peut représenter les résidus en fonction des valeurs prédites.

```
plot(residuals(m) ~ fitted(m))
```



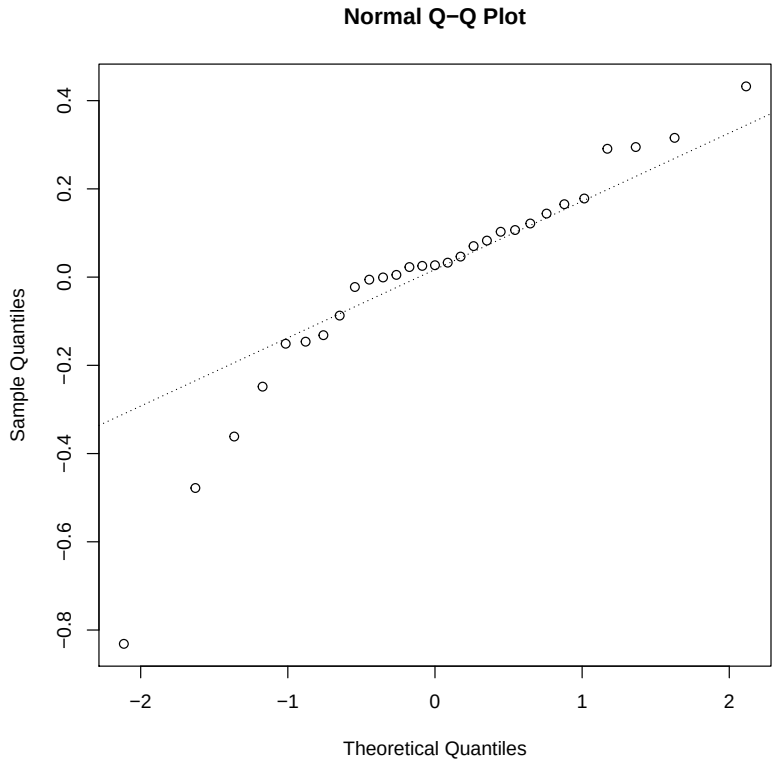
En complément il peut être intéressant de représenter toutes les résidus en fonction de chaque variable explicative.

```
par(mfrow=c(2, 2))
par(mar=c(5, 4, 1, 1))
plot(residuals(m) ~ d$T)
plot(residuals(m) ~ d$HR)
plot(residuals(m) ~ d$ClO2)
plot(residuals(m) ~ d$t)
```



Le diagramme Quantile-Quantile des résidus (et éventuellement le test de leur normalité avec toutes les réserves décrites précédemment) peut aussi être réalisé comme en régression simple.

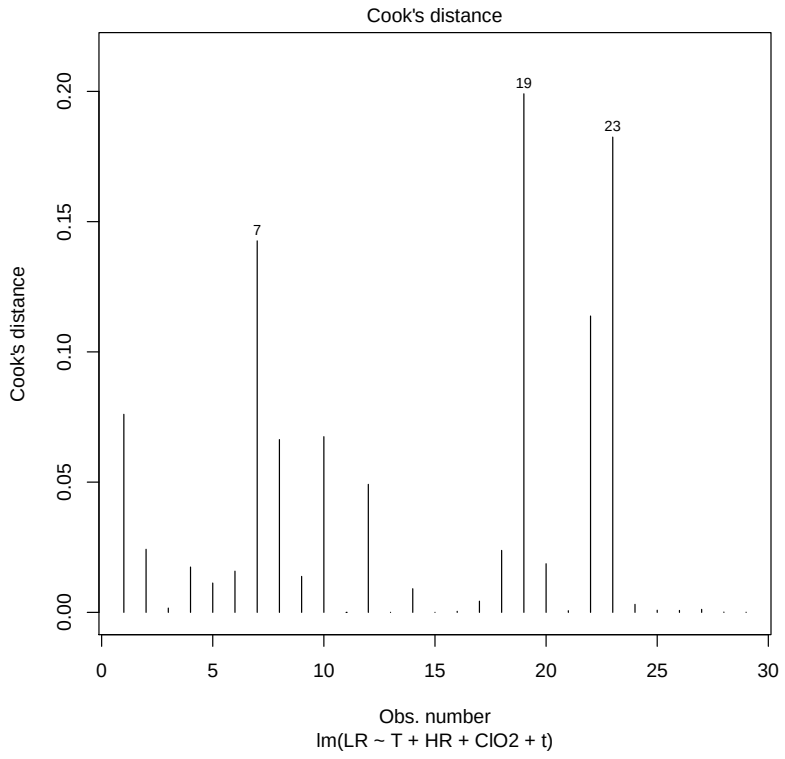
```r
qqnorm(residuals(m))
qqline(residuals(m), lty = 3)
```



2.3 Analyse des données influentes

Le graphe des distances de Cook permettant d'évaluer l'influence de chaque observation sur l'ensemble des valeurs prédites par le modèle peut être réalisé de la fonçon suivante.

```r
plot(m, which = 4)
```



2.4 Inférence

Comme en régression simple on peut accéder aux coefficients du modèle ajusté et aux intervalles de confiance sur ces coefficients.

```
coef(m)
```

```
## (Intercept)           T          HR        ClO2           t
##     -4.3783      0.0258      0.0435     11.5208      0.0177
```

```
confint(m)
```

```
##                2.5 %  97.5 %
## (Intercept) -5.38184 -3.3747
## T            0.00285  0.0487
## HR           0.03201  0.0549
## ClO2        10.37571 12.6660
## t            0.01387  0.0216
```

On peut aussi obtenir des valeurs prédites avec les deux types d'intervalles associés.

```
predict(m,data.frame(T = 15, HR = 70, ClO2 = 0.4, t = 50))
```

```
##    1
## 4.55
```

```
predict(m,data.frame(T = 15, HR = 70, ClO2 = 0.4, t = 50), interval = "confidence")
```

```
##    fit  lwr  upr
## 1 4.55 4.38 4.72
```

```
predict(m,data.frame(T = 15, HR = 70, ClO2 = 0.4, t = 50), interval = "prediction")
```

```
##    fit  lwr  upr
## 1 4.55 3.96 5.13
```

2.5 Comparaison de modèles et sélection de variables

La fonction `anova` permet très simplement de réaliser le test F des modèles emboîtés.

```
mcomplet <- m
msimplifie <- lm(LR ~ HR + ClO2 + t, data = d)
anova(msimplifie, mcomplet)
```

```
## Analysis of Variance Table
##
## Model 1: LR ~ HR + ClO2 + t
## Model 2: LR ~ T + HR + ClO2 + t
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1     25 2.17
## 2     24 1.77  1     0.398 5.38  0.029
```

La fonction `AIC` permet de calculer l'AIC (critère d'information d'Akaïké) d'un ou plusieurs modèles.

```
AIC(msimplifie, mcomplet)
```

```
##            df  AIC
## msimplifie  5 17.1
## mcomplet    6 13.3
```

Elle permet aussi de calculer le BIC (critère d'information de Schwarz), en changeant comme ci-dessous l'argument `k`.

```
AIC(msimplifie, mcomplet, k = log(nrow(d)))
```

```
##            df  AIC
## msimplifie  5 24.0
## mcomplet    6 21.5
```

Il existe aussi une fonction spécifique que l'on peut utiliser directement pour calculer le BIC :

```
BIC(msimplifie, mcomplet)
```

```
##            df  BIC
## msimplifie  5 24.0
## mcomplet    6 21.5
```

Enfin la fonction `step` met en oeuvre une méthode de sélection automatique de variables sur la base de l'AIC et renvoit le modèle sélectionné. Par défaut si on lui donne un modèle complet, une sélection descendante est appliquée. *Dans cet exemple le modèle complet est retenu.*

```
mselectAIC <- step(m)
```

```
## Start:  AIC=-71
## LR ~ T + HR + ClO2 + t
##
##        Df Sum of Sq  RSS   AIC
## <none>               1.8 -71.0
## - T     1       0.4  2.2 -67.2
## - HR    1       4.5  6.3 -36.2
## - t     1       6.6  8.4 -28.0
## - ClO2  1      31.9 33.6  12.3
```

```
mselectAIC
```

```
##
## Call:
## lm(formula = LR ~ T + HR + ClO2 + t, data = d)
##
## Coefficients:
## (Intercept)            T           HR         ClO2            t
##     -4.3783       0.0258       0.0435      11.5208       0.0177
```

Le BIC peut être utilisé dans la sélection en modifiant l'argument `k` de la fonction `step` comme précédemment dans la fonction `AIC`. Dans cet exemple cela aboutit à la même sélection de modèle, ce qui n'est bien entendu pas systématique, le BIC ayant généralement tendance à sélectionner des modèles plus parcimonieux (avec moins de variables) que l'AIC.

```
mselectBIC <- step(m, k = log(nrow(d)))
```

```
## Start:  AIC=-64.2
## LR ~ T + HR + ClO2 + t
##
##        Df Sum of Sq  RSS   AIC
## <none>               1.8 -64.2
## - T     1       0.4  2.2 -61.7
## - HR    1       4.5  6.3 -30.8
## - t     1       6.6  8.4 -22.5
## - ClO2  1      31.9 33.6  17.8
```

```
mselectBIC
```

```
##
## Call:
## lm(formula = LR ~ T + HR + ClO2 + t, data = d)
##
## Coefficients:
## (Intercept)            T           HR         ClO2            t
##     -4.3783       0.0258       0.0435      11.5208       0.0177
```

Enfin on peut faire de la sélection ascendante avec la fonction `step` mais il faut alors indiquer le modèle de départ (le plus simple) comme premier argument ET la formule du modèle le plus complexe envisagé dans l'argument `scope`. Voici un exemple de sélection ascendante basée sur l'AIC (une méthode plus sophistiquée, mélange d'ascendante et descendante est proposée par défaut correspondant à l'argument `direction` fixé à `"both"`).

```
mdepart <- lm(LR ~ 1, data = d)
mselectAICasc <- step(mdepart, scope = LR ~ T + HR + ClO2 + t, direction = "forward")

## Start:  AIC=14.8
## LR ~ 1
##
##        Df Sum of Sq  RSS   AIC
## + ClO2  1      31.9 13.3 -18.6
## + t     1       6.6 38.6  12.3
## + HR    1       4.5 40.6  13.8
## <none>              45.2  14.8
## + T     1       0.4 44.8  16.6
##
## Step:  AIC=-18.6
## LR ~ ClO2
##
##        Df Sum of Sq   RSS   AIC
## + t     1      6.60  6.70 -36.5
## + HR    1      4.53  8.77 -28.7
## <none>              13.31 -18.6
## + T     1      0.40 12.91 -17.5
##
## Step:  AIC=-36.5
## LR ~ ClO2 + t
##
##        Df Sum of Sq  RSS   AIC
## + HR    1      4.53 2.17 -67.2
## <none>              6.70 -36.5
## + T     1      0.40 6.31 -36.2
##
## Step:  AIC=-67.2
## LR ~ ClO2 + t + HR
##
##        Df Sum of Sq  RSS   AIC
## + T     1     0.398 1.77 -71.0
## <none>              2.17 -67.2
##
## Step:  AIC=-71
## LR ~ ClO2 + t + HR + T

mselectAICasc

##
## Call:
## lm(formula = LR ~ ClO2 + t + HR + T, data = d)
##
## Coefficients:
## (Intercept)         ClO2            t           HR            T
##     -4.3783      11.5208       0.0177       0.0435       0.0258
```

On peut remarquer que l'AIC donné par la fonction AIC n'a pas la même valeur que celui affiché dans la fonction step. Ceci est lié au fait que l'AIC n'est défini qu'à une constante près et que le choix de la constante n'est pas le même dans les deux fonctions.

2.6 Cas particulier du modèle polynomial

Un modèle polynomial est un modèle linéaire que l'on peut spécifier très facilement avec **R**. Voici l'exemple de l'ajustement d'un modèle quadratique (polynomial d'ordre 2) sur le logarithme népérien de la variable `lag` en fonction des variables explicatives `T1` et `T2`.

```
d <- read.table("lagT1T2.txt", header = TRUE, stringsAsFactors = TRUE)
str(d)

## 'data.frame': 13 obs. of  3 variables:
```

```
##  $ T1 : int  4 4 4 4 4 15 15 15 15 15 ...
##  $ T2 : int  4 8 15 28 37 4 8 15 28 37 ...
##  $ lag: num  23.2 10 4.4 1.9 1.4 31.6 12.7 4.3 1.6 1.4 ...
```

```
m <- lm(log(lag) ~ T1 + T2 + T1:T2 + I(T1^2) + I(T2^2), data = d)
summary(m)
```

```
##
## Call:
## lm(formula = log(lag) ~ T1 + T2 + T1:T2 + I(T1^2) + I(T2^2),
##     data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.14817 -0.07970 -0.00742  0.09456  0.13862
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.683325   0.157468   23.39  6.6e-08
## T1           0.027355   0.013899    1.97  0.08973
## T2          -0.191449   0.014929  -12.82  4.1e-06
## I(T1^2)      0.000118   0.000313    0.38  0.71620
## I(T2^2)      0.002852   0.000350    8.16  8.0e-05
## T1:T2       -0.001260   0.000222   -5.68  0.00075
##
## Residual standard error: 0.132 on 7 degrees of freedom
## Multiple R-squared:  0.994,Adjusted R-squared:  0.99
## F-statistic:  236 on 5 and 7 DF,  p-value: 1.21e-07
```

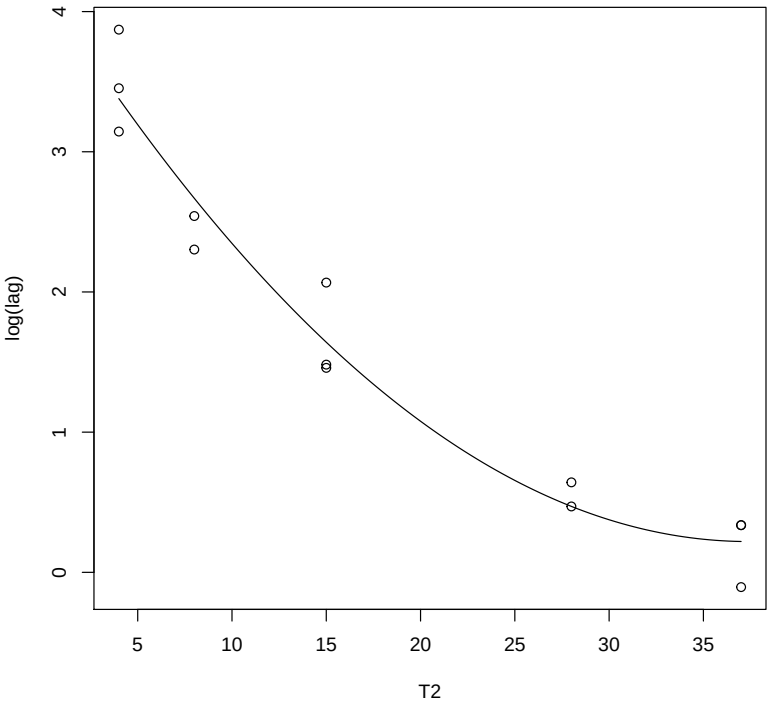
Là encore il est difficile de faire un graphe d'ajustement simple mais on analyse les résidus comme pour tout autre modèle linéaire.

Dans le cas d'un modèle polynomial avec une seule variable explicative, on peut tracer le polynome avec les données observées en récupérant les coefficients du modèle comme ci-dessous.

```
msansT1 <- lm(log(lag) ~ T2 + I(T2^2), data = d)
(coefm <- coef(msansT1))
```

```
## (Intercept)          T2     I(T2^2)
##     4.18122    -0.21178     0.00283
```

```
# on fait une séquence de 100 valeurs de T2 dans le domaine étudié
T2forplot <- seq(min(d$T2), max(d$T2), length.out = 100)
# on calcule les valeurs de lag en log associées, prédites par le modèle
Llagpred <- coefm[1] + coefm[2] * T2forplot + coefm[3] * T2forplot^2
# et on trace les données et le modèle
plot(log(lag) ~ T2, data = d)
lines(T2forplot, Llagpred)
```

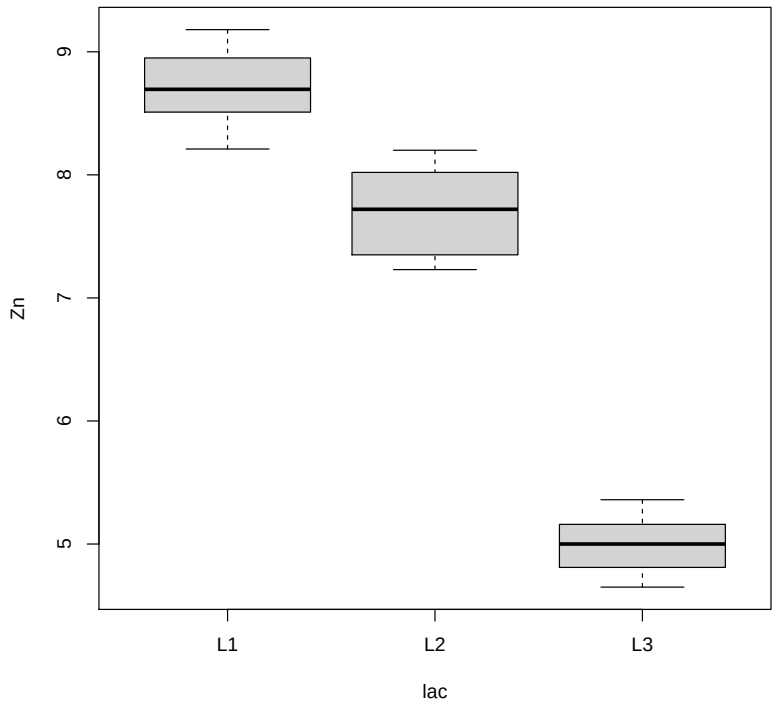


3 Lorsque certaines variables explicatives sont qualitatives

3.1 Analyse de variance à 1 facteur

Une analyse de variance peut être réalisée à l'aide de la fonction `lm` de façon équivalente à l'utilisation de la fonction `aov` lorsque le facteur étudié est un **facteur fixe**. La fonction `lm` suppose le facteur fixe et ne permet donc pas de modéliser l'effet d'un facteur aléatoire.

```
d <- read.table("Znmoules.txt", header = TRUE, stringsAsFactors = TRUE)
str(d)

## 'data.frame': 30 obs. of  2 variables:
##  $ lac: Factor w/ 3 levels "L1","L2","L3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Zn : num  8.51 8.75 8.42 8.95 8.64 9.18 9.17 8.9 8.51 8.21 ...

plot(Zn ~ lac,data = d)
```




```
(m <- lm(Zn ~ lac, data = d))

##
## Call:
## lm(formula = Zn ~ lac, data = d)
##
## Coefficients:
## (Intercept)         lacL2         lacL3
##        8.72         -1.04         -3.73

anova(m)

## Analysis of Variance Table
##
## Response: Zn
##           Df Sum Sq Mean Sq F value Pr(>F)
## lac        2   74.1    37.1     378 <2e-16
## Residuals 27    2.6     0.1
```

L'interprétation des coefficients du modèle linéaire dépend du type de contrainte utilisé dans le codage de l'ANOVA en modèle linéaire. Par défaut la fonction `lm` utilise une contrainte de type cellule de référence. Dans ce cas l'intercept correspond à la moyenne du groupe de référence (première modalité du facteur) et les autres coefficients à la différence entre la moyenne du groupe considéré et cette moyenne de référence.

```
options(contrasts = c("contr.treatment", "contr.treatment"))
m <- lm(Zn ~ lac, data = d)
summary(m)

##
## Call:
## lm(formula = Zn ~ lac, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5140 -0.2815 -0.0085  0.2220  0.5200
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)    8.724      0.099   88.15   <2e-16
## lacL2         -1.044      0.140   -7.46    5e-08
## lacL3         -3.731      0.140  -26.66   <2e-16
##
## Residual standard error: 0.313 on 27 degrees of freedom
## Multiple R-squared:  0.966,Adjusted R-squared:  0.963
## F-statistic:  378 on 2 and 27 DF,  p-value: <2e-16
```

Une autre contrainte moins classiquement utilisée mais correspondant à l'écriture courante du modèle théorique de l'ANOVA est la contrainte de type somme, pour laquelle l'intercept correspond à la moyenne globale et chaque coefficient à l'écart entre la moyenne du groupe et cette moyenne globale. Le coefficient du dernier groupe n'apparaît pas mais peut facilement être calculé à partir des autres, la somme des coefficients étant supposée nulle avec cette contrainte.

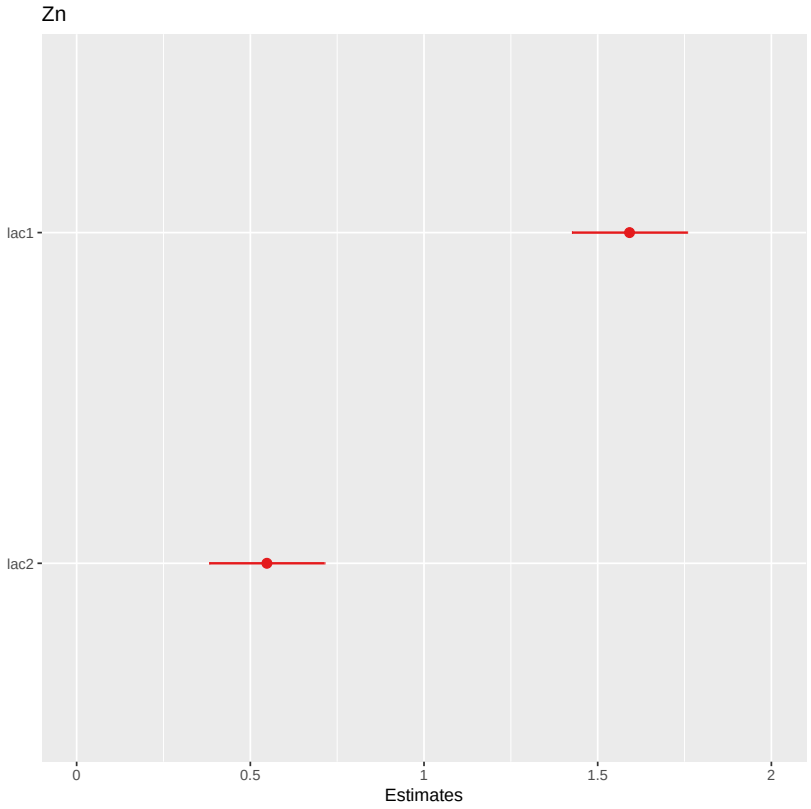
```
options(contrasts = c("contr.sum","contr.sum"))
m <- lm(Zn ~lac, data = d)
summary(m)

##
## Call:
## lm(formula = Zn ~ lac, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5140 -0.2815 -0.0085  0.2220  0.5200
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.1323     0.0571  124.83  < 2e-16
## lac1          1.5917     0.0808   19.70  < 2e-16
## lac2          0.5477     0.0808    6.78  2.8e-07
##
## Residual standard error: 0.313 on 27 degrees of freedom
## Multiple R-squared:  0.966,Adjusted R-squared:  0.963
## F-statistic:  378 on 2 and 27 DF,  p-value: <2e-16
```

Lors de l'utilisation de la contrainte de type cellule de référence, il est parfois nécessaire de changer le groupe de référence. Une façon de modifier le groupe de référence est de modifier l'ordre des facteurs de la façon suivante (ci-dessous on redéfinit les modalités dans l'ordre L3, L2, L1 et donc le groupe de référence devient L3 le moins contaminé)

```
d$lac <- factor(d$lac, levels = c("L3", "L2", "L1"))
```

Les intervalles de confiance sur les effets estimés (donc ici différence à la moyenne du groupe de référence) peuvent être calculés et représentés à l'aide du package `sjPlot` par exemple.

```
confint(m)

##             2.5 % 97.5 %
## (Intercept) 7.015  7.250
## lac1        1.426  1.757
## lac2        0.382  0.713

library(sjPlot)
plot_model(m)
```

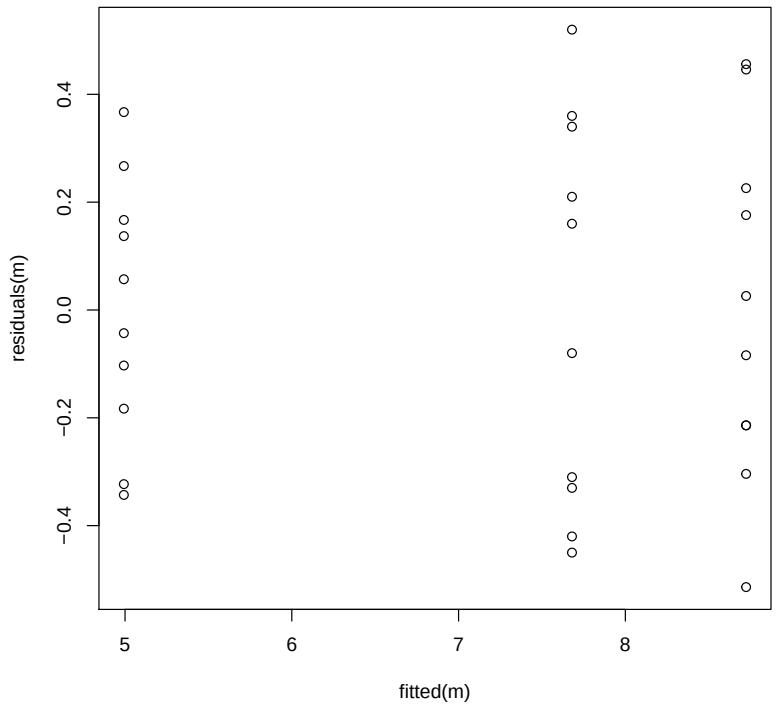


Il est parfois plus facile d'interpréter les coefficients en supprimant le paramètre redondant dans l'écriture classique d'un modèle d'ANOVA 1, c'est-à-dire en enlevant la constante comme ci-dessous. Néanmoins il faut bien avoir à l'esprit que dans ce cas les valeurs de p associées à chaque coefficient (donc chaque moyenne de groupe) n'ont plus aucun intérêt.

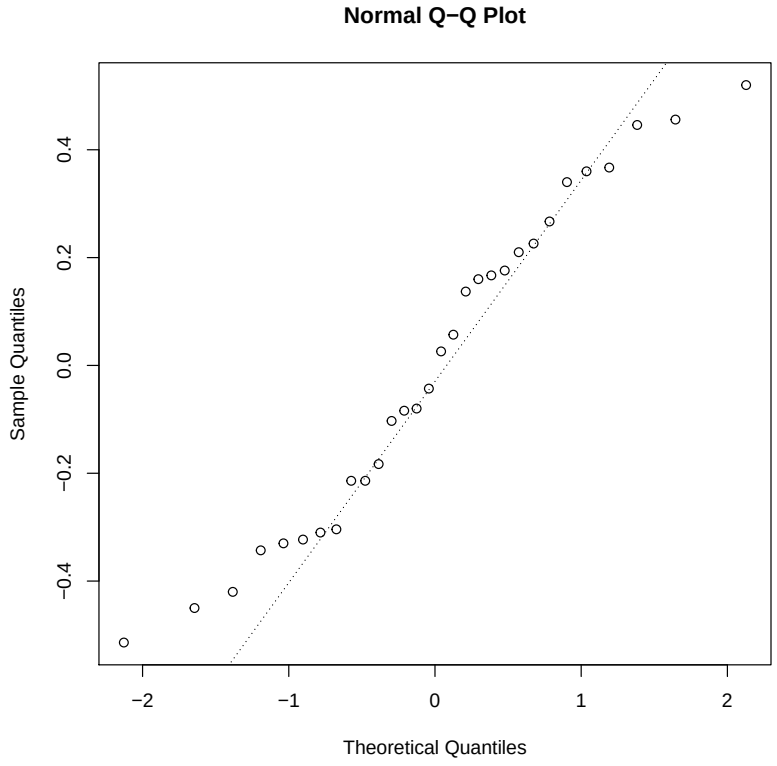
```
msansconst <- lm(Zn ~ lac - 1, data = d)
summary(msansconst)

##
## Call:
## lm(formula = Zn ~ lac - 1, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5140 -0.2815 -0.0085  0.2220  0.5200
##
## Coefficients:
##       Estimate Std. Error t value Pr(>|t|)
## lacL3    4.993      0.099    50.5   <2e-16
## lacL2    7.680      0.099    77.6   <2e-16
## lacL1    8.724      0.099    88.2   <2e-16
##
## Residual standard error: 0.313 on 27 degrees of freedom
## Multiple R-squared:  0.998,Adjusted R-squared:  0.998
## F-statistic: 5.45e+03 on 3 and 27 DF,  p-value: <2e-16
```

Les résidus peuvent être représentés en fonction de la variable prédite et leur normalité vérifiée comme pour un autre modèle linéaire.

```
plot(fitted(m), residuals(m))
```




```
qqnorm(residuals(m))
qqline(residuals(m), lty = 3)
```



On peut aussi tester l'égalité des variances entre groupes par un test de Levene réalisé sur les résidus, en appliquant à nouveau l'analyse de variance sur la valeur absolue des résidus. La mise en évidence d'un effet impliquerait le non respect de la condition d'égalité des variances. ATTENTION, là encore il faut bien garder à l'esprit qu'un tel test ne permet pas à lui seul de valider l'hypothèse d'égalité des variances.

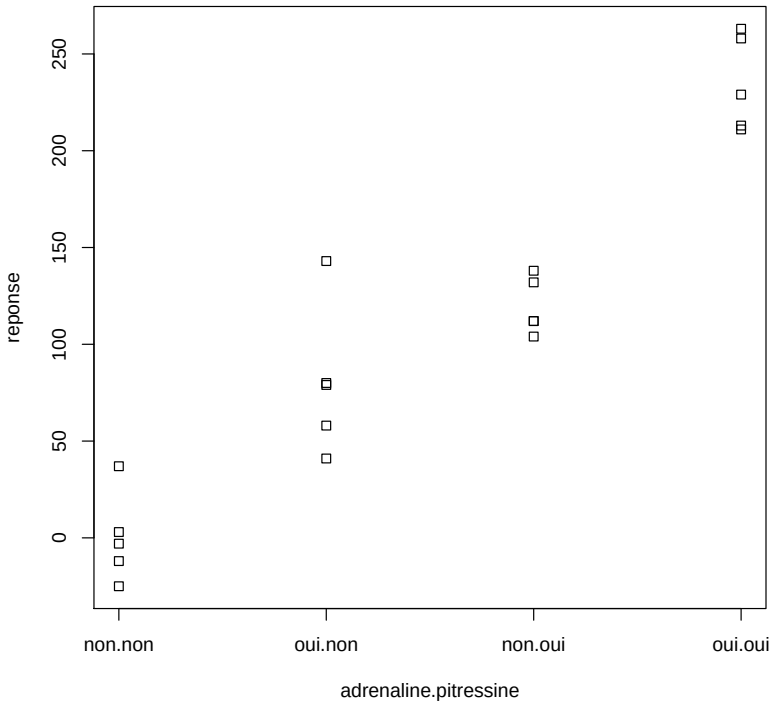
```
mres <- lm(abs(residuals(m)) ~ d$lac)
anova(mres)

## Analysis of Variance Table
```

```
##
## Response: abs(residuals(m))
##           Df Sum Sq Mean Sq F value Pr(>F)
## d$lac      2  0.071  0.0356    1.81   0.18
## Residuals 27  0.529  0.0196
```

3.2 Analyse de variance à 2 facteurs

Une analyse de variance à 2 facteurs peut être réalisée à l'aide de la fonction `lm` de façon équivalente à l'utilisation de la fonction `aov` lorsque les 2 **facteurs étudiés sont fixes et croisés**. La fonction `lm` ne permet pas, par contre, de prendre en compte des facteurs aléatoires et de modèles hiérarchisés.

```
d <- read.table("ADREPITRE.txt", header = TRUE, stringsAsFactors = TRUE)
str(d)

## 'data.frame': 20 obs. of  3 variables:
##  $ adrenaline: Factor w/ 2 levels "non","oui": 1 1 1 1 1 2 2 2 2 2 ...
##  $ pitressine: Factor w/ 2 levels "non","oui": 1 1 1 1 1 1 1 1 1 1 ...
##  $ reponse   : int  -25 -3 -12 37 3 143 41 58 80 79 ...

stripchart(reponse ~ adrenaline:pitressine, vertical = TRUE,
           xlab = "adrenaline.pitressine", data = d)
```




```
options(contrasts=c("contr.treatment","contr.treatment"))
m <- lm(reponse ~ adrenaline + pitressine + adrenaline:pitressine, data = d)
summary(m)

##
## Call:
## lm(formula = reponse ~ adrenaline + pitressine + adrenaline:pitressine,
##     data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -39.2  -17.1   -4.4   13.9   62.8
##
## Coefficients:
```

```
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       0.0       11.9    0.00  1.00000
## adrenalineoui                    80.2       16.9    4.75  0.00022
## pitressineoui                   119.6       16.9    7.09  2.6e-06
## adrenalineoui:pitressineoui      35.0       23.9    1.47  0.16187
##
## Residual standard error: 26.7 on 16 degrees of freedom
## Multiple R-squared:  0.926,Adjusted R-squared:  0.913
## F-statistic: 67.1 on 3 and 16 DF,  p-value: 2.8e-09
```

Une écriture équivalente du modèle est :

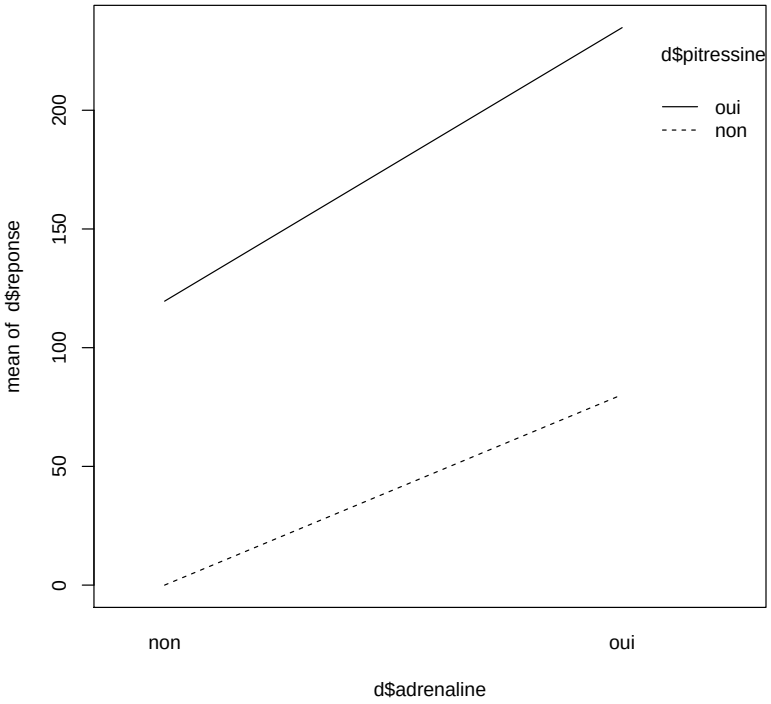
```
m <- lm(reponse ~ adrenaline * pitressine, data = d)
```

Le **tableau d'analyse de variance classique** associé aux données peut être réalisé. ATTENTION, dans le cas d'un plan d'expérience déséquilibré, le résultat donné par la fonction `anova` dépendra de l'ordre d'introduction des facteurs dans l'écriture du modèle et sera donc beaucoup parfois plus difficile à interpréter.

```
# Vérification de l'équilibre du plan d'expérience
xtabs(~ adrenaline + pitressine, data = d)

##           pitressine
## adrenaline non oui
##        non   5   5
##        oui   5   5

anova(m)

## Analysis of Variance Table
##
## Response: reponse
##                       Df Sum Sq Mean Sq F value  Pr(>F)
## adrenaline             1  47726   47726   67.04 4.1e-07
## pitressine             1  93982   93982  132.01 3.9e-09
## adrenaline:pitressine  1   1531    1531    2.15    0.16
## Residuals             16  11391     712
```

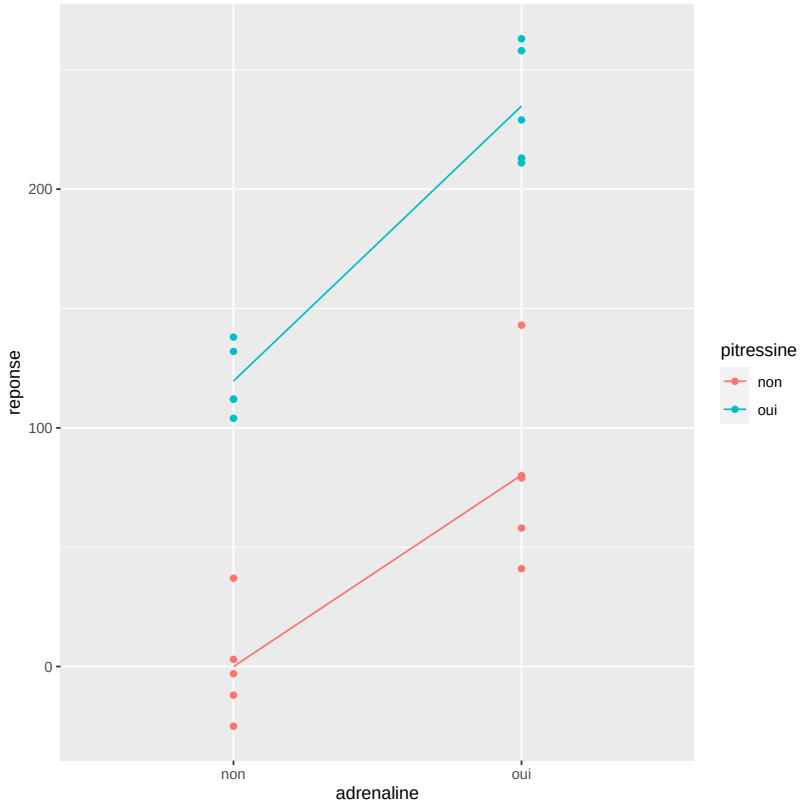
Un graphe classique et utile dans le cas d'une ANOVA 2 de ce type est le graphe d'interaction, qui représente les moyennes observées pour chaque croisement des modalités des deux facteurs, en les reliant de façon à ce qu'une interaction fasse apparaître des segments non parallèles.

```
interaction.plot(d$adrenaline, d$pitressine, d$reponse)
```



En utilisant `ggplot2` on peut faire apparaître les points observées sur ce type de graphe d'interaction :

```
require(ggplot2)
qplot(adrenaline, reponse, data = d, colour = pitressine) +
  stat_summary(fun = mean, geom = "line", aes(group = pitressine))
```



L'interprétation des effets principaux de chacun des facteurs ne peut être réalisée simplement que lorsque l'interaction n'est pas significative. Dans ce cas il est parfois intéressant (si l'on pense pouvoir négliger une éventuelle interaction) de simplifier le modèle en enlevant le terme d'interaction pour donner une estimation de l'effet de chacun des deux facteurs.

```r
ms <- lm(reponse ~ adrenaline + pitressine, data = d)
summary(ms)
```

```
##
## Call:
## lm(formula = reponse ~ adrenaline + pitressine, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -48.0  -16.3   -6.1   10.2   54.0
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -8.75      10.68   -0.82     0.42
## adrenalineoui    97.70      12.33    7.92  4.2e-07
## pitressineoui   137.10      12.33   11.12  3.2e-09
##
## Residual standard error: 27.6 on 17 degrees of freedom
## Multiple R-squared:  0.916,Adjusted R-squared:  0.907
## F-statistic: 93.2 on 2 and 17 DF,  p-value: 6.88e-10
```

```r
anova(ms)
```

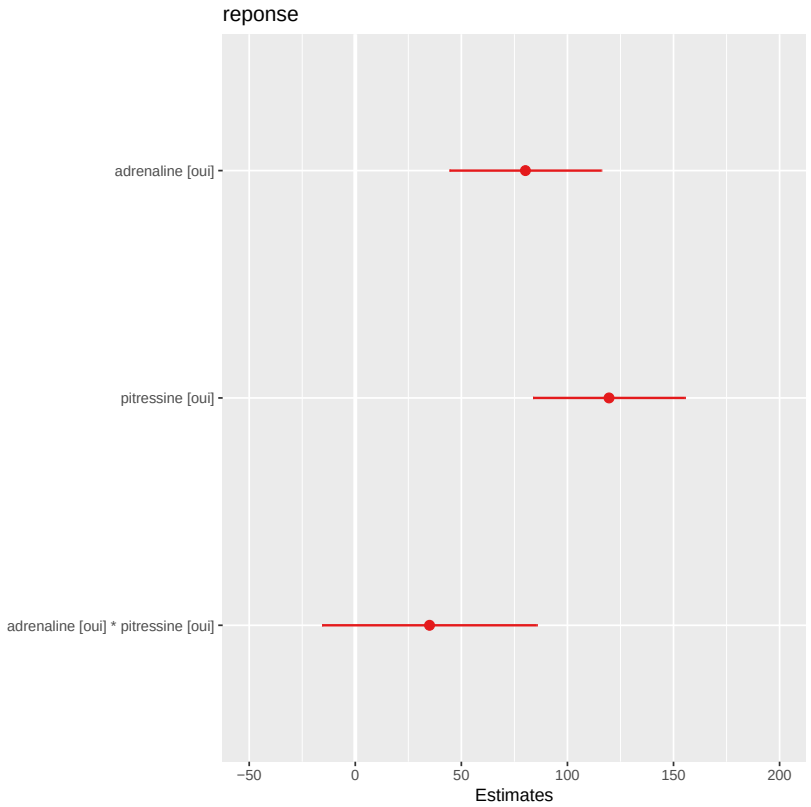
```
## Analysis of Variance Table
##
## Response: reponse
##             Df Sum Sq Mean Sq F value  Pr(>F)
## adrenaline   1  47726   47726    62.8 4.2e-07
## pitressine   1  93982   93982   123.6 3.2e-09
## Residuals   17  12922     760
```

L'interprétation des coefficients du modèle linéaire suit les mêmes règles qu'en ANOVA 1, en fonction du type de contraintes spécifié. Avec la contrainte par défaut de type cellule de référence, l'intercept correspond à la moyenne prédite pour le groupe correspondant au croisement des premières modalités de chacun des facteurs. Et l'on peut calculer les intervalles de confiance sur les effets estimés et les représenter avec le package `sjPlot` par exemple. Voici ci-dessous les intervalles de confiance du modèle complet avec l'interaction qui apparaît non significative (intervalle de confiance comprenant 0).

```r
confint(m)
```

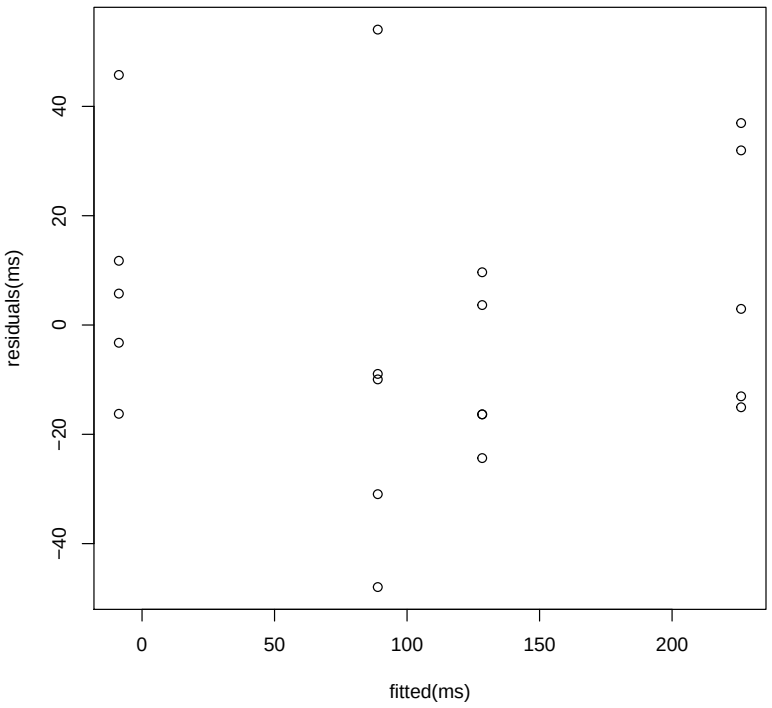
```
##                               2.5 % 97.5 %
## (Intercept)                   -25.3   25.3
## adrenalineoui                  44.4  116.0
## pitressineoui                  83.8  155.4
## adrenalineoui:pitressineoui -15.6   85.6
```

```r
plot_model(m)
```

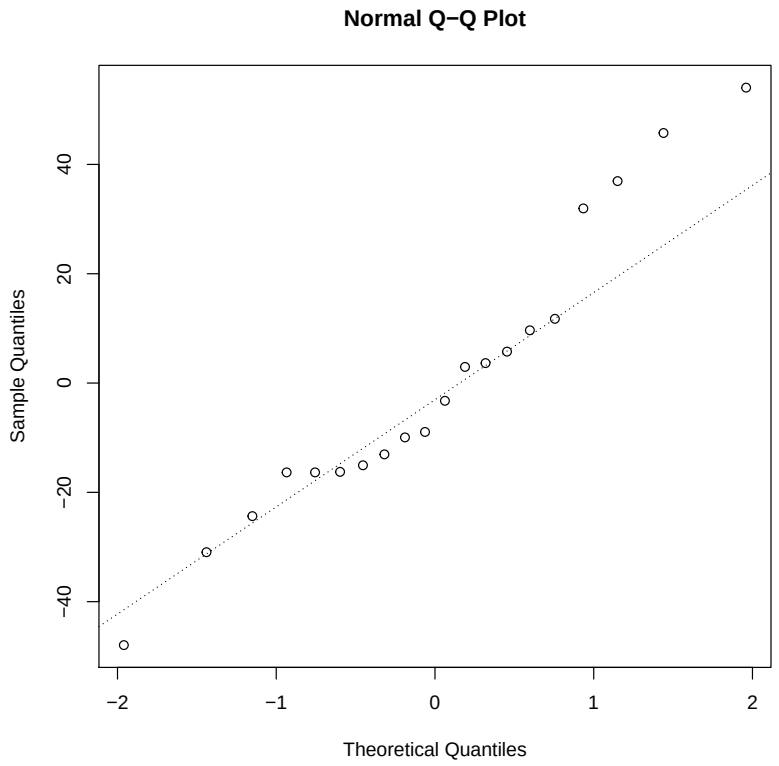


Les résidus peuvent être représentés en fonction de la variable prédite et leur normalité vérifiée comme pour un autre modèle linéaire.

```
plot(fitted(ms), residuals(ms))
```




```
qqnorm(residuals(ms))
qqline(residuals(ms), lty = 3)
```



Le test de Levene peut aussi être réalisé, toujours avec les mêmes réserves quant aux conclusions qu'on peut en tirer :

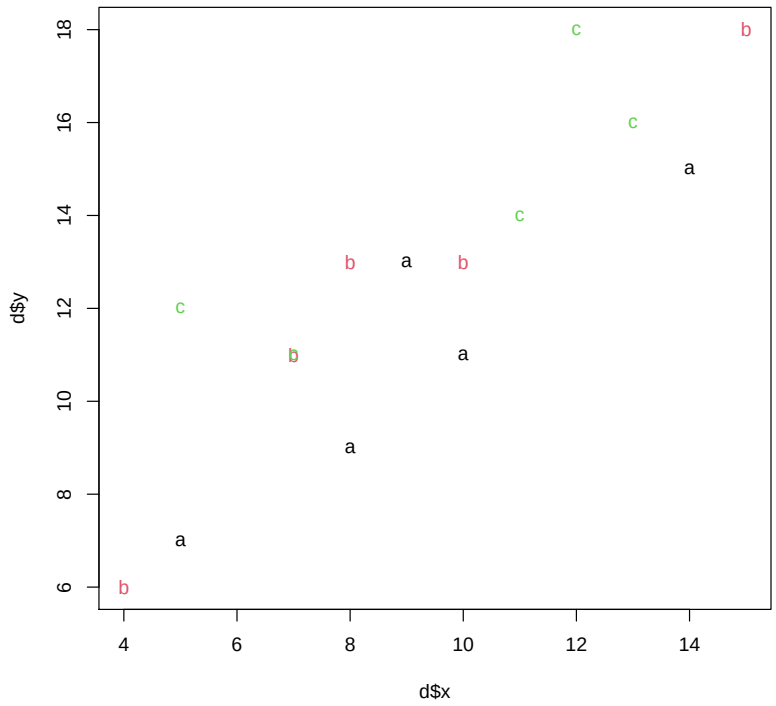
```
mres <- lm(abs(residuals(ms)) ~ d$adrenaline + d$pitressine)
anova(mres)

## Analysis of Variance Table
##
## Response: abs(residuals(ms))
##               Df Sum Sq Mean Sq F value Pr(>F)
## d$adrenaline   1    487     487    2.05   0.17
## d$pitressine   1    207     207    0.87   0.36
## Residuals     17   4031     237
```

3.3 Analyse de covariance

La fonction `lm` permet de modéliser l'effet de deux facteurs croisés fixes dont l'un serait quantitatif.

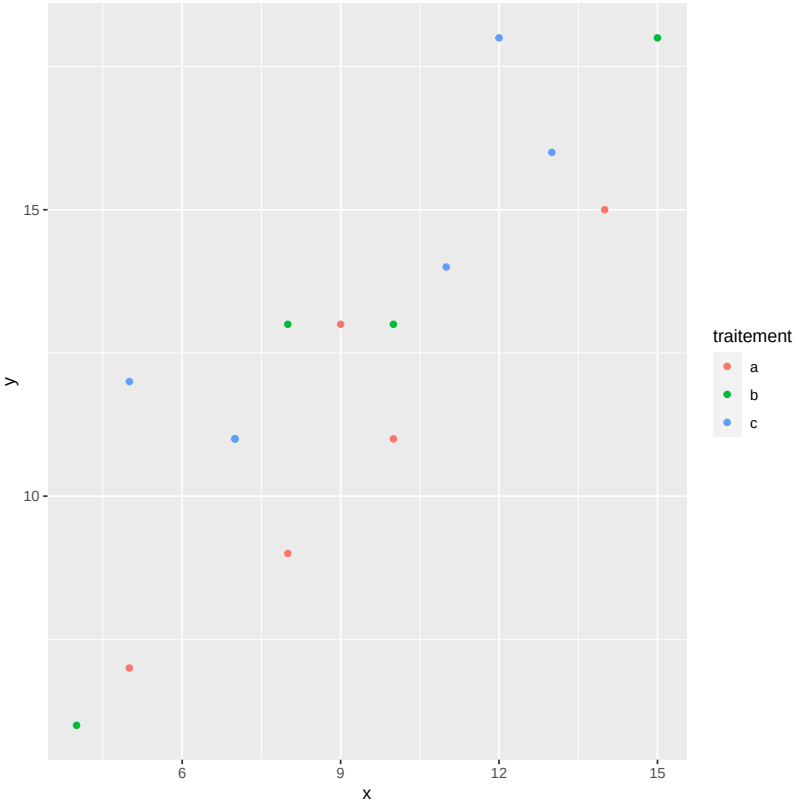
Les données peuvent être visualisées par un nuage de points en adoptant des codes de couleur et/ou des caractères différents pour les groupes comparés.

```
d <- read.table("dosageavap.txt", header = TRUE, stringsAsFactors = TRUE)
str(d)

## 'data.frame': 15 obs. of  3 variables:
##  $ x         : int  5 8 9 10 14 4 7 8 10 15 ...
##  $ y         : int  7 9 13 11 15 6 11 13 13 18 ...
##  $ traitement: Factor w/ 3 levels "a","b","c": 1 1 1 1 1 2 2 2 2 2 ...

plot(d$x, d$y, type = "n")
text(d$x, d$y, as.character(d$traitement),
     col = as.numeric(d$traitement))
```



C'est bien entendu plus simple à écrire si l'on utilise `ggplot2` :

```
require(ggplot2)
qplot(x, y, colour = traitement, data = d)
```



— **Modèle complet avec interaction (pentes différentes)**

```
m1 <- lm(y ~ x + traitement + x:traitement, data = d)
summary(m1)
##
## Call:
## lm(formula = y ~ x + traitement + x:traitement, data = d)
##
```

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.360 -0.822 -0.425  0.846  2.178
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2.832      2.232    1.27    0.236
## x                0.888      0.231    3.84    0.004
## traitementb      0.384      2.845    0.13    0.896
## traitementc      4.575      3.148    1.45    0.180
## x:traitementb    0.133      0.296    0.45    0.664
## x:traitementc   -0.180      0.319   -0.56    0.586
##
## Residual standard error: 1.51 on 9 degrees of freedom
## Multiple R-squared:  0.881,Adjusted R-squared:  0.816
## F-statistic: 13.4 on 5 and 9 DF,  p-value: 0.000604
```

On peut aussi écrire plus simplement ce modèle :

```
m1 <- lm(y ~ x * traitement, data = d)
```

ATTENTION, comme dans le cas d'une ANOVA sur plan d'expérience déséquilibré, le résultat donné par la fonction `anova` *appliqué sur un modèle avec covariable (variable explicative quantitative) dépendra de l'ordre d'introduction des variables explicatives dans l'écriture du modèle et sera donc difficile à interpréter (non recommandé).*

— **Modèle simplifié sans interaction (supposant les pentes égales)**
Si cela semble raisonnable on peut simplifier le modèle (modèle à pente unique) pour estimer l'effet du facteur qualitatif sur l'ordonnée à l'origine.

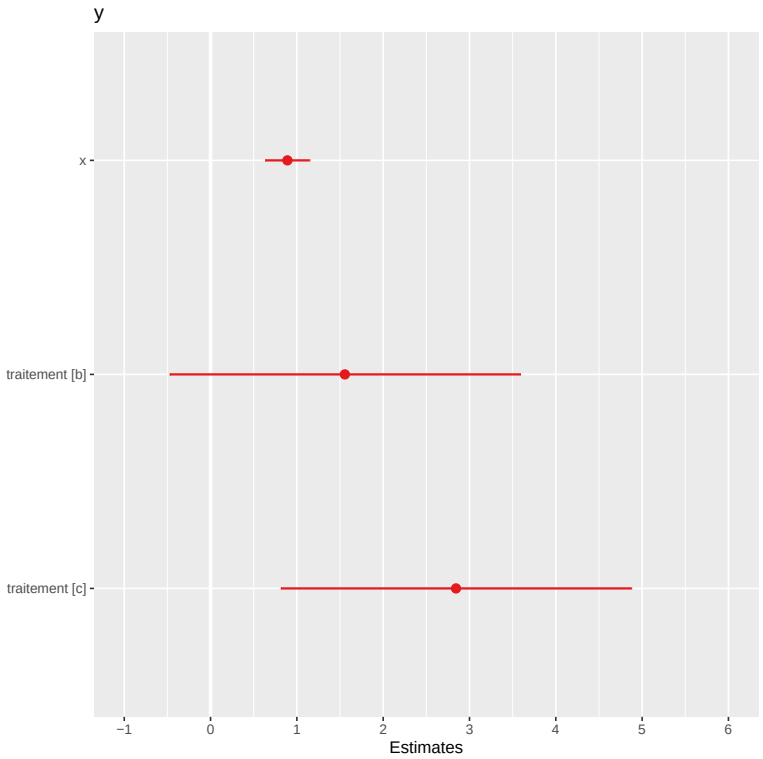
```
m2 <- lm(y ~ x + traitement, data = d)
summary(m2)
##
## Call:
## lm(formula = y ~ x + traitement, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.927 -0.908 -0.268  0.957  2.178
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.809      1.252    2.24    0.046
## x              0.890      0.116    7.66  9.9e-06
## traitementb    1.556      0.922    1.69    0.120
## traitementc    2.844      0.922    3.08    0.010
##
## Residual standard error: 1.46 on 11 degrees of freedom
## Multiple R-squared:  0.866,Adjusted R-squared:  0.829
## F-statistic: 23.7 on 3 and 11 DF,  p-value: 4.23e-05
```

— **Test de l'égalité des pentes par comparaison des deux modèles emboîtés**

```
anova(m1, m2)
## Analysis of Variance Table
##
## Model 1: y ~ x * traitement
## Model 2: y ~ x + traitement
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1      9 20.6
## 2     11 23.3 -2     -2.72 0.59   0.57
```

— **Interprétation des coefficients des modèles** L'interprétation des coefficients du modèle linéaire suit les mêmes règles qu'en ANOVA, en fonction du type de contraintes spécifié. Dans le modèle complet avec la contrainte par défaut de type cellule de référence, l'intercept et la pente correspondent aux valeurs prédites pour le groupe de référence (première modalité du facteur étudié) et les autres coefficients aux écarts sur l'ordonnée à l'origine et la pente par rapport à ce groupe de référence.
Voici les intervalles de confiance associés au modèle sans interaction.

```
confint(m2)
##                2.5 % 97.5 %
## (Intercept)  0.0532   5.57
## x            0.6344   1.15
## traitementb -0.4730   3.59
## traitementc  0.8148   4.87
plot_model(m2)
```



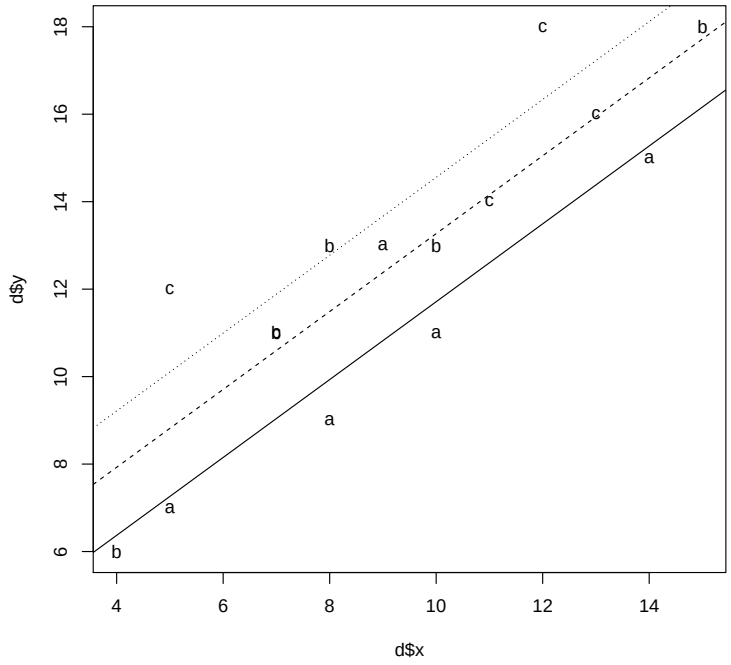
Il faut bien avoir en tête en interprétant ce type de graphe que les coefficients associés à des modalités de variables qualitatives (ici modalités b et c du traitement) correspondent à des différences à la moyenne du groupe de référence (traitement a) alors que le coefficient associé à x correspondant à une différence moyenne pour la variable observée lorsque x augmente d'une unité.

```
coef(m2)
## (Intercept)           x traitementb traitementc
##        2.81        0.89        1.56        2.84
```

— **Visualisation de l'ajustement**

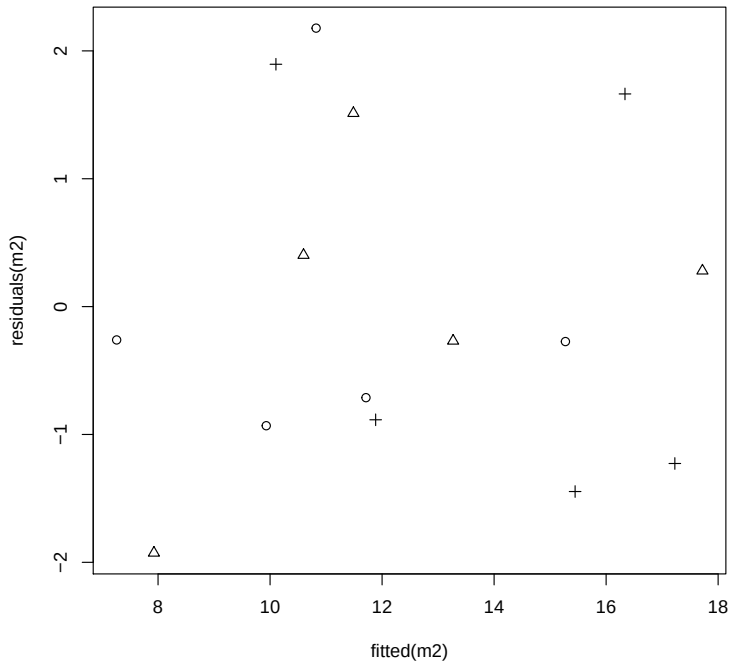
Si l'on veut visualiser les droites ajustées correspondant aux différents groupes, on peut récupérer les valeurs des coefficients à l'aide de la fonction `coef` et utiliser la fonction `abline` pour ajouter les droite aux points observés de la façon suivante :

```
plot(d$x, d$y, type = "n")
text(d$x, d$y, as.character(d$traitement))
abline(a= coef(m2)[1], b = coef(m2)[2],lty = 1)
abline(a= coef(m2)[1] + coef(m2)[3], b = coef(m2)[2], lty = 2)
abline(a= coef(m2)[1] + coef(m2)[4], b = coef(m2)[2], lty = 3)
```

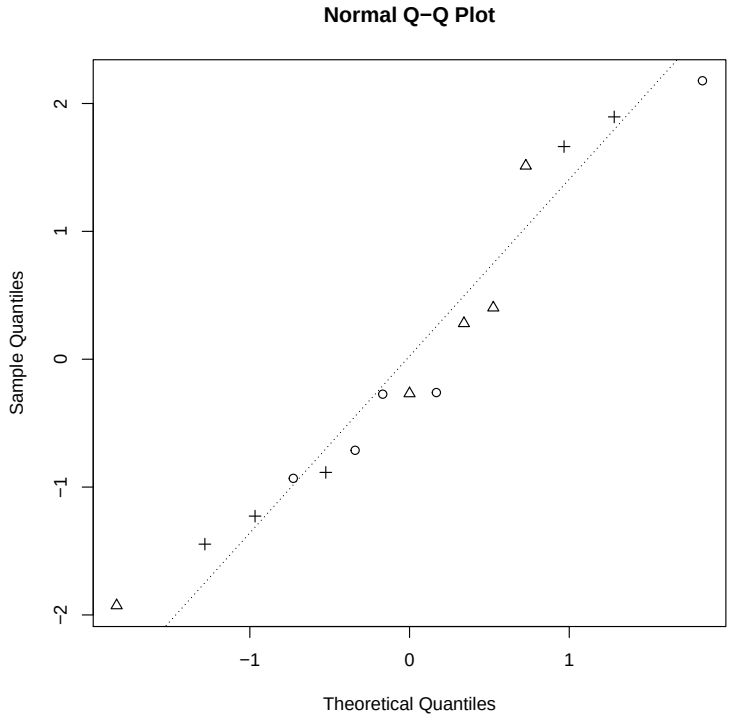


— **Examen des résidus**
La vérification du modèle d'erreur peut être réalisé à partir des résidus comme pour tout modèle linéaire gaussien. On peut aussi coder le groupe par un type de points (cf. ex. ci-dessous) ou une couleur pour voir aussi s'il y a un effet groupe sur les résidus :

```
plot(fitted(m2), residuals(m2), pch = as.numeric(d$traitement))
```




```
qqnorm(residuals(m2), pch = as.numeric(d$traitement))
qqline(residuals(m2), pch = as.numeric(d$traitement), lty = 3)
```


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−2
−1
0
1
2
−1
0
1