TP OUMOBIO 9: Le cas des relations non linéaires

Mathieu Brevet

2025-01-14

Bienvenue dans ce neuvième TP sur R. Après avoir vu comment modéliser une relation linéaire, nous allons maintenant explorer quelques solutions pour modéliser des réponses non linéaires dans R.

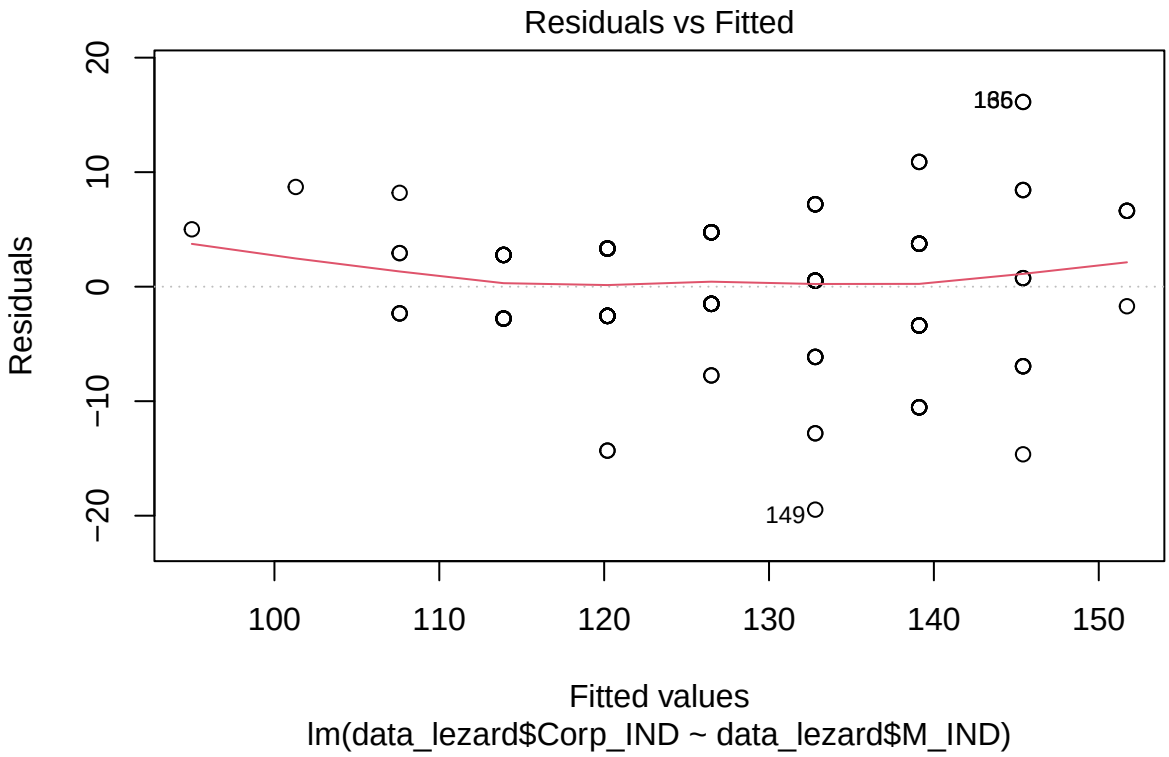
```
setwd("~/ATER PAU 2024/Cours modifiés/OUMOBIOS")


data_lezard = read.table("Suivi_lezard_vivipare.csv", header = T, sep = "\t", dec = ",")
```

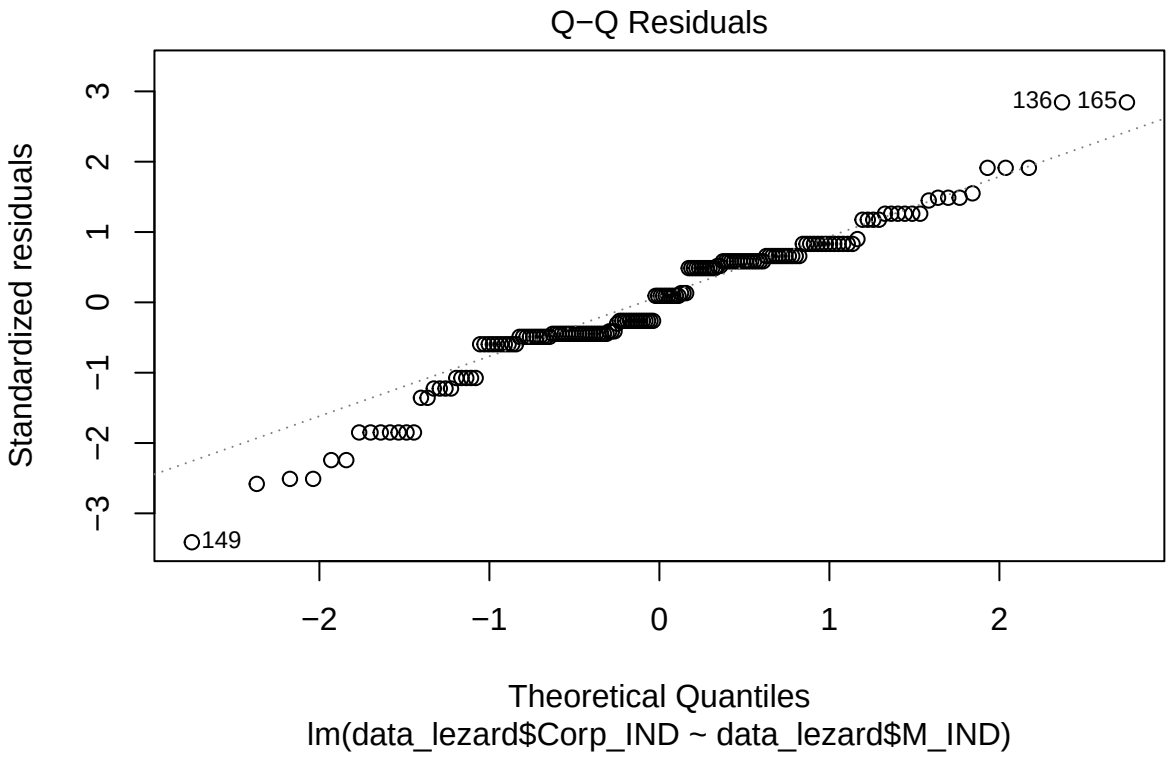
Distributions biaisées et transformation des données

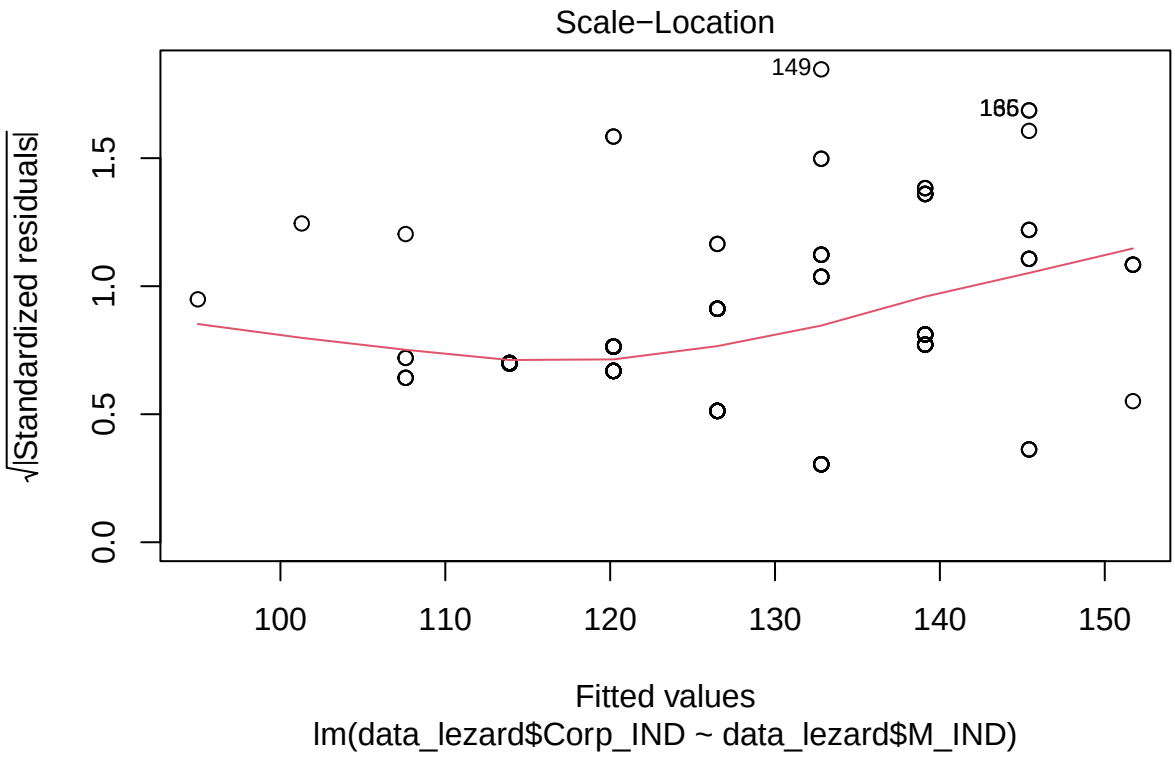
Il peut arriver que les prérequis de la régression linéaire ne soient pas respectés si la distribution de la variable réponse **dévie de la loi normale**, notamment du fait de **distributions asymétriques**, avec une forte queue de distribution sur un côté de la distribution (**biais positif** à droite de la distribution: positivement asymétrique; **biais négatif** à gauche de la distribution: négativement asymétrique). Ces problématiques peuvent facilement être résolues en **transformant la variable réponse**: on utilise usuellement une transformation racine carré pour un faible biais positif et carré pour un faible biais négatif; une transformation racine cubique (dans R: "x^(-3)") pour un biais positif modéré et cubique pour un biais négatif modéré; une transformation logarithmique pour un fort biais positif ; une transformation inverse ("1/x") pour un biais sévère (présence d'outliers). Dans la pratique, il peut être utile de tester les différentes transformations pour sélectionner celle qui permet de mieux s'approcher d'une distribution normale.

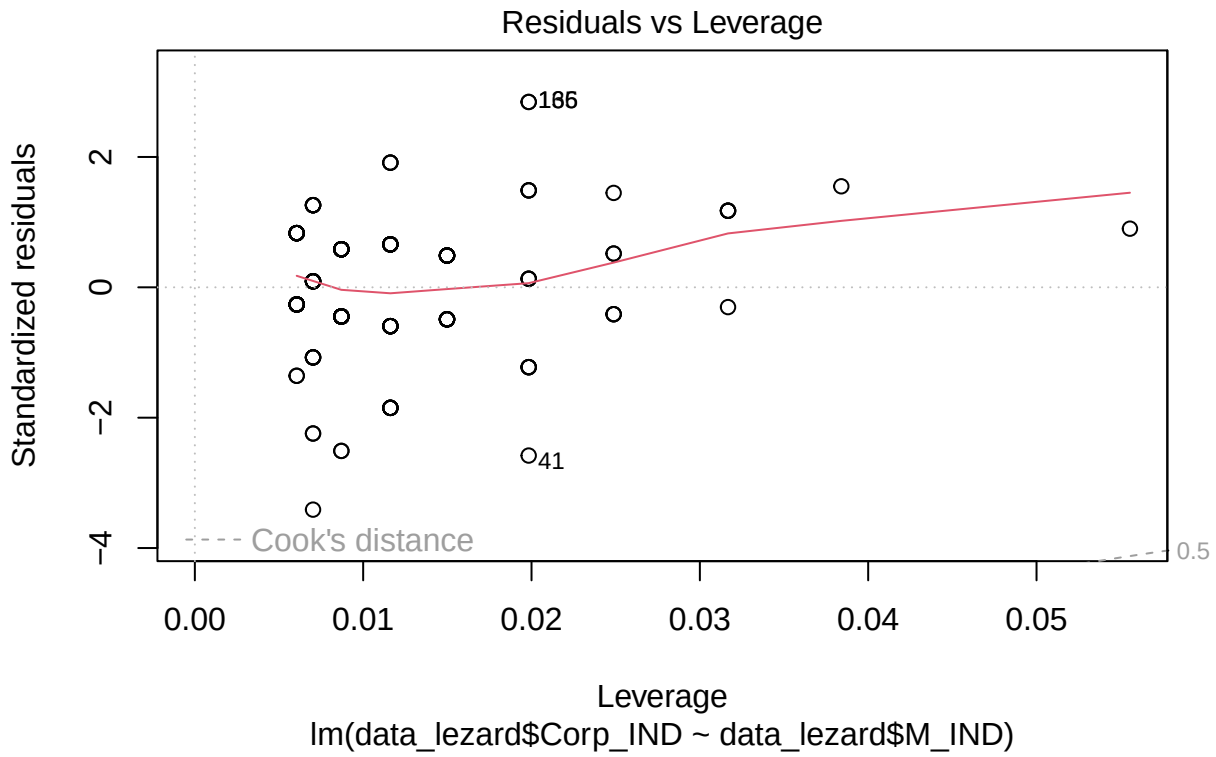
Prenons un exemple concret:

```
data_lezard$Corp_IND = data_lezard$SVL_IND/data_lezard$M_IND

reg_corp_poids_juv = lm(data_lezard$Corp_IND ~ data_lezard$M_IND)



plot(reg_corp_poids_juv)
```

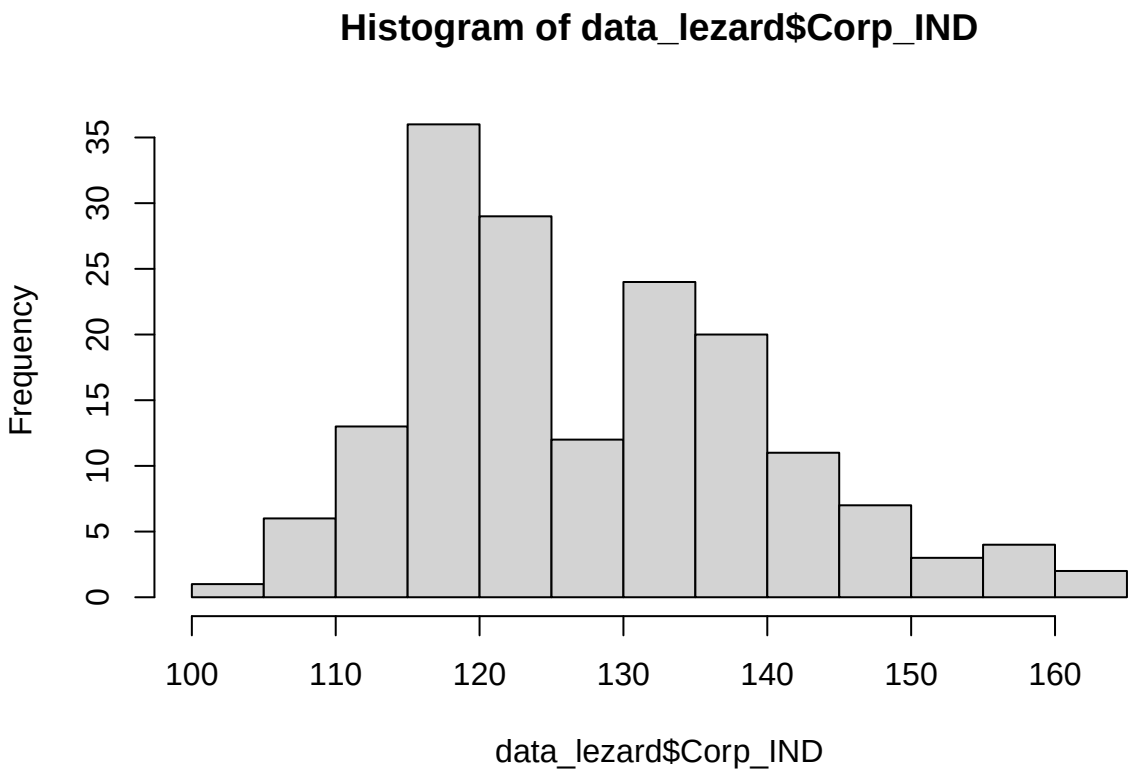

Residuals vs Fitted
Residuals
20
10
0
−10
−20
166
149
100
110
120
130
140
150
Fitted values
lm(data_lezard$Corp_IND ~ data_lezard$M_IND)


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(data_lezard$Corp_IND ~ data_lezard$M_IND)
136
165
149
−3
−2
−1
0
1
2
3

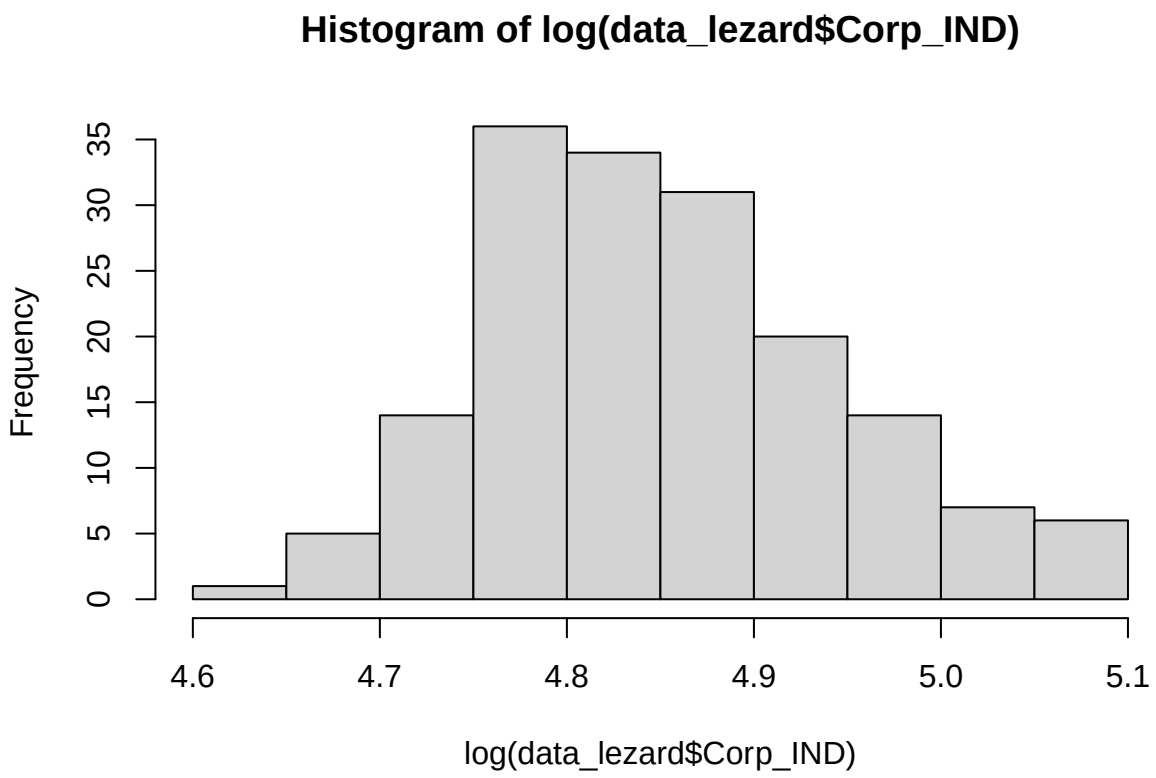

Scale−Location
149
166
√|Standardized residuals|
0.0
0.5
1.0
1.5
100
110
120
130
140
150
Fitted values
lm(data_lezard$Corp_IND ~ data_lezard$M_IND)



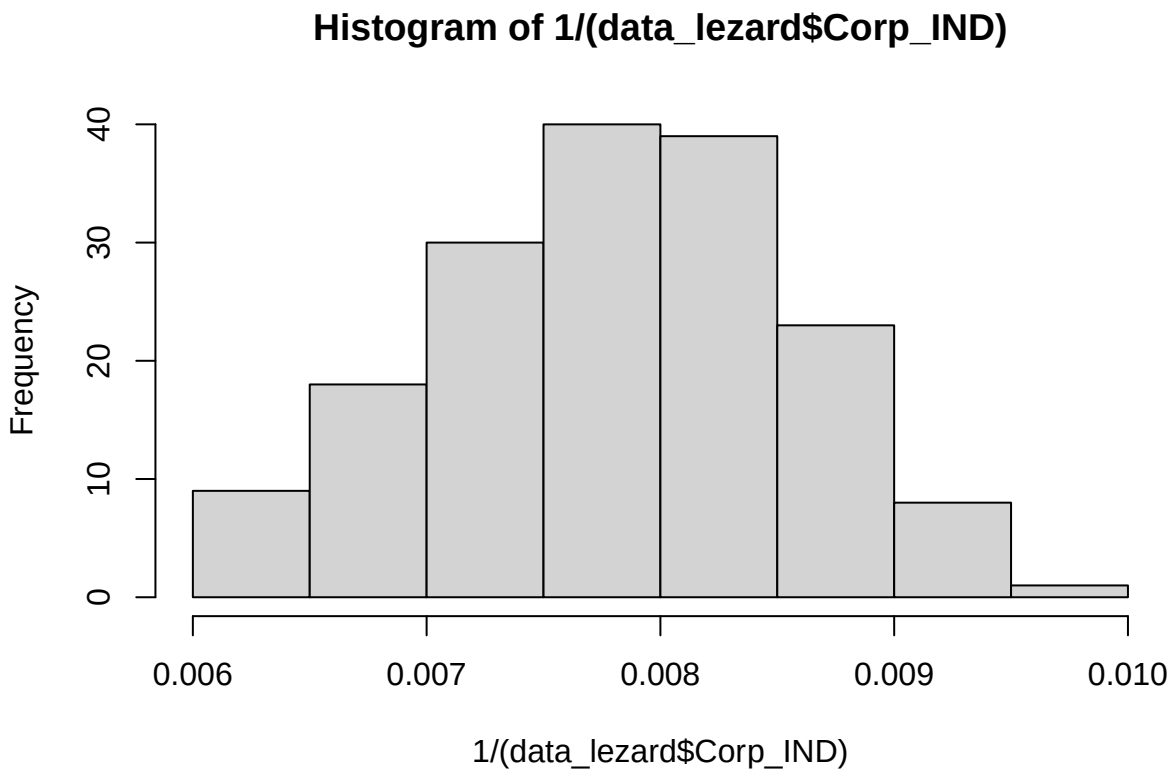

```
# problème assez visible d'homoscédasticité des résidus (variance augmente avec
# le poids)


hist(data_lezard$Corp_IND)
```



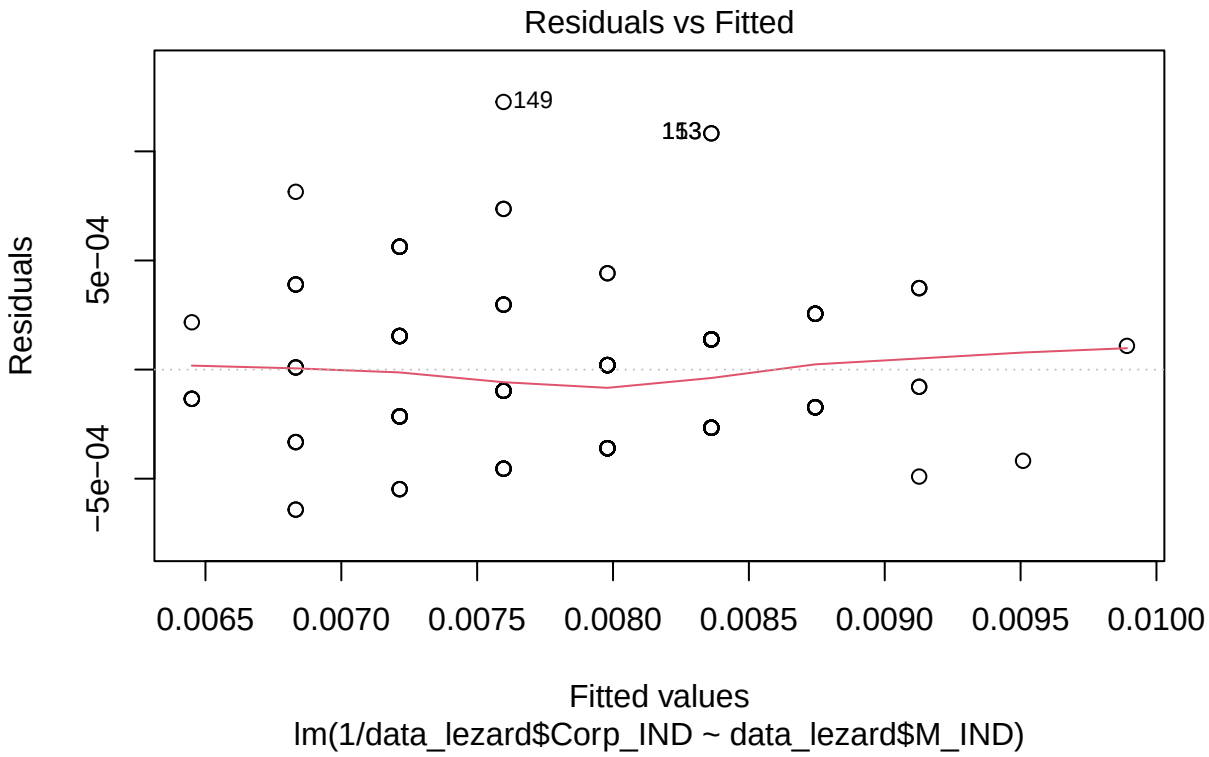

```
# biais positif très visible (longue queue de distribution à droite)

hist(log(data_lezard$Corp_IND))
```

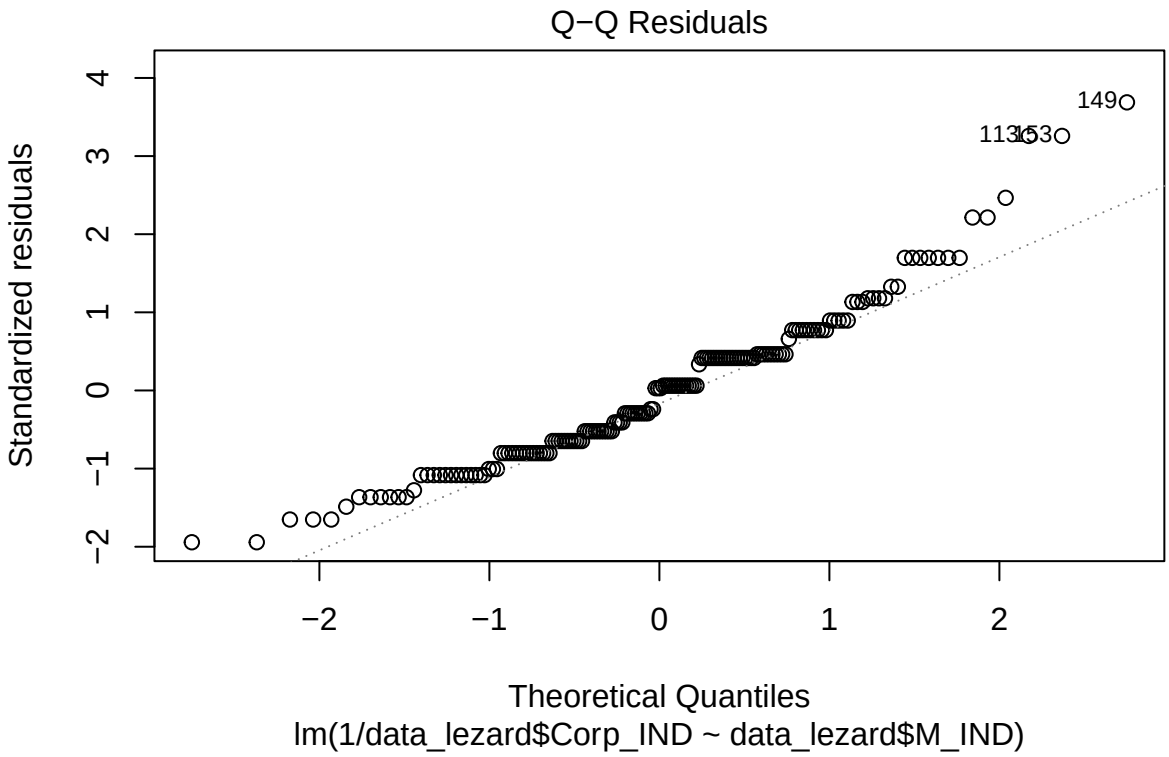


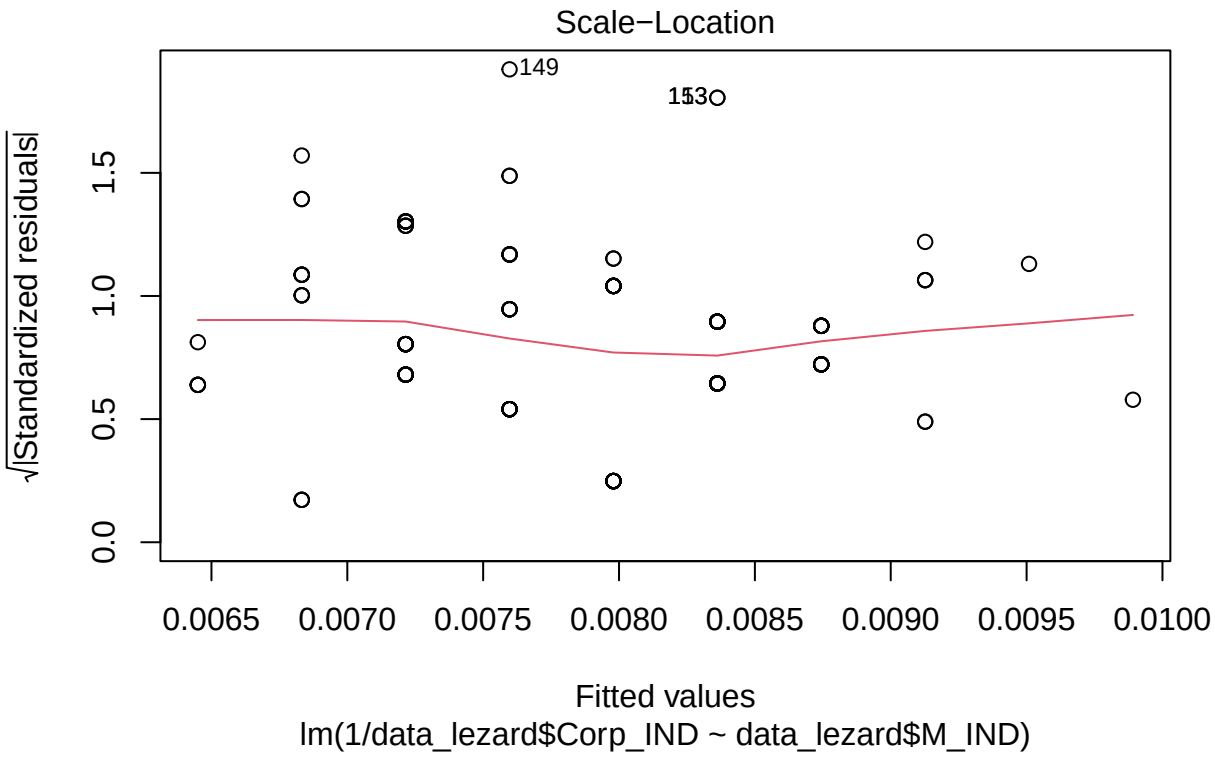

```
# la transformation logarithmique ne permet pas de complètement résorber le
# bais existant

hist(1/(data_lezard$Corp_IND))
```

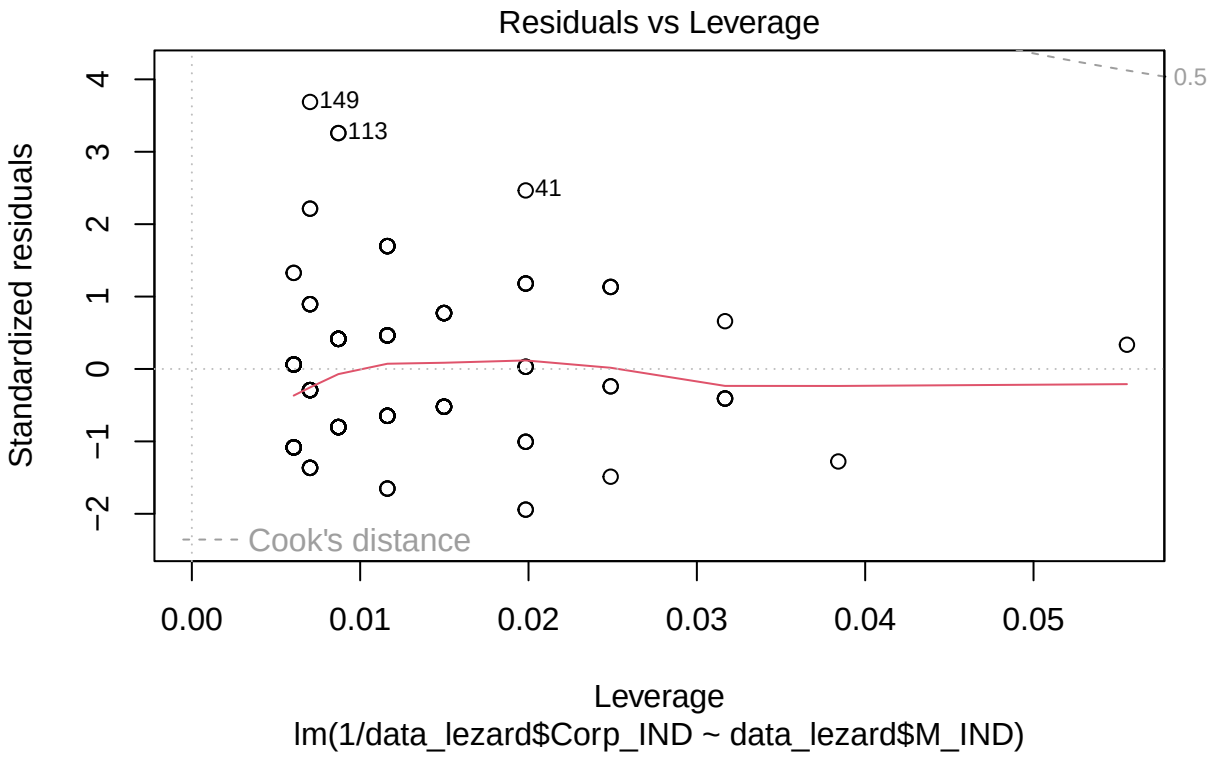



```
# le biais n'apparait plus après un transformation inverse


reg_corp_poids_juv_inv = lm(1/data_lezard$Corp_IND ~ data_lezard$M_IND)
plot(reg_corp_poids_juv_inv)
```


Residuals vs Fitted
149
153
Residuals
5e−04
−5e−04
0.0065
0.0070
0.0075
0.0080
0.0085
0.0090
0.0095
0.0100
Fitted values
lm(1/data_lezard$Corp_IND ~ data_lezard$M_IND)


Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(1/data_lezard$Corp_IND ~ data_lezard$M_IND)
149
113
153
−2
−1
0
1
2
3
4


Scale−Location
149
153
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.0065
0.0070
0.0075
0.0080
0.0085
0.0090
0.0095
0.0100
Fitted values
lm(1/data_lezard$Corp_IND ~ data_lezard$M_IND)




```
# le problème d'homoscédasticité des résidus est résolu (il n'apparaît plus) ce
# type de transformation permet d'éviter d'avoir des modèles biaisés dont les
# résultats peuvent ne pas être complètement fiable
```

Modéliser des relations non linéaires

Lorsque la relation entre deux variables quantitatives n'est pas linéaire, on peut envisager d'autres approches pour modéliser cette relation. La première solution est d'utiliser une **régression polynomiale**: on garde une structure linéaire mais on ajoute des termes d'ordre supérieur pour mieux représenter la nature de la relation entre les deux variables. La seconde solution est de **linéariser la relation** entre les deux variables, en utilisant une transformation de la relation en une relation linéaire (exemple: transformation logarithmique d'une relation exponentielle entre deux variables). La troisième solution est de directement modéliser la relation existante par une formule choisie et d'en estimer les paramètres par la méthode des moindres carrés.

Les régressions polynomiales

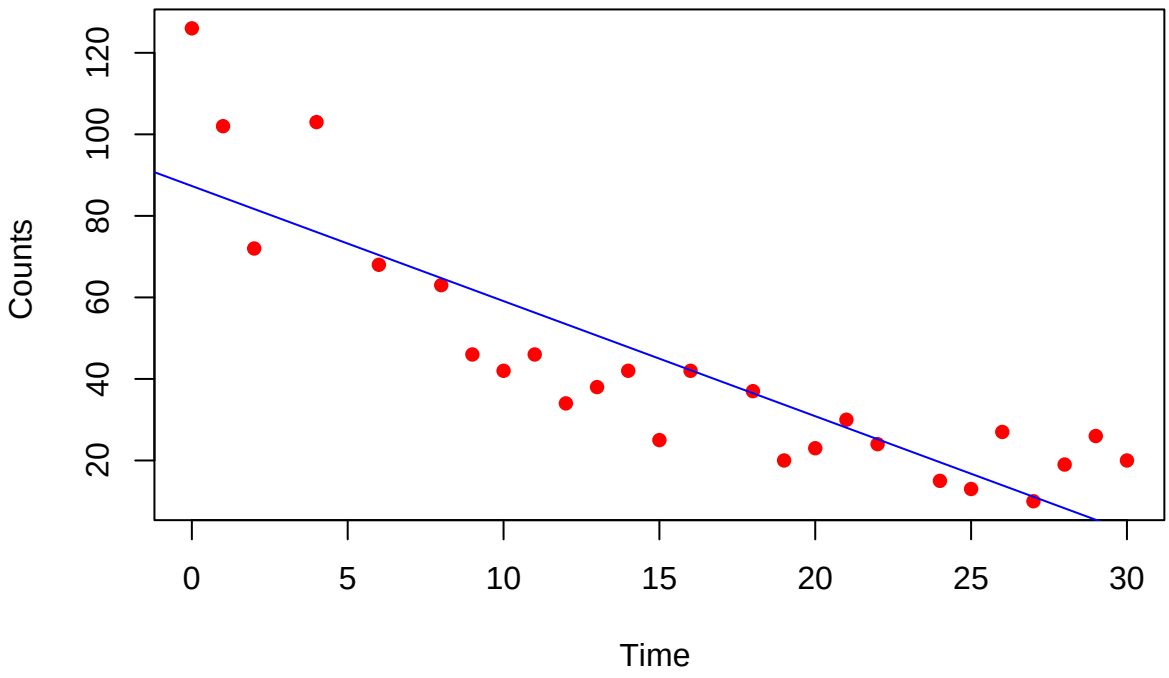
Les **régressions polynomiales** permettent de représenter une variable en fonction d'une combinaison linéaire des puissances d'une autre variable (de la forme $Y = aX + bX^2 + cX^3 + \ldots + dX^n$, pour un polynôme d'ordre ou de degré n). Ce type de régression est particulièrement adapté lorsque la forme de la relation est **incurvée** et plus globalement en forme de **"cloche"** (pour les polynômes d'ordre 2) ou de "vagues" (pour les polynômes d'ordre trois ou plus). Les polynômes de haut degré peuvent être utilisés pour approcher n'importe quelle relation "lisse".

Prenons un exemple ici en simulant le nombre de nouveaux individus capturés en fonction du nombre de jour depuis le premier échantillonnage:

```
Comptage_new_ind <- data.frame(
    Time = c(
        0, 1, 2, 4, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 24,
        25, 26, 27, 28, 29, 30
    ),
    Counts = c(
        126, 102, 72, 103, 68, 63, 46, 42, 46, 34, 38, 42, 25, 42, 37, 20, 23, 30,
        24, 15, 13, 27, 10, 19, 26, 20
    )
)
# Simulation de données: nombre de nouveaux individus capturés en fonction du
# nombre de jours écoulés


plot(
    Comptage_new_ind$Time, Comptage_new_ind$Counts, pch = 16, xlab = "Time ", ylab = "Counts ",
    col = "red"
)
# la relation ne semble pas linéaire (incurvée)


# modélisation par une régression linéaire:

linear.model <- lm(Counts ~ Time, data = Comptage_new_ind)

summary(linear.model)$r.squared
```

```
## [1] 0.7498847
```

```
plot(
    Comptage_new_ind$Time, Comptage_new_ind$Counts, pch = 16, xlab = "Time ", ylab = "Counts ",
    col = "red"
)

abline(linear.model, col = "blue")
```

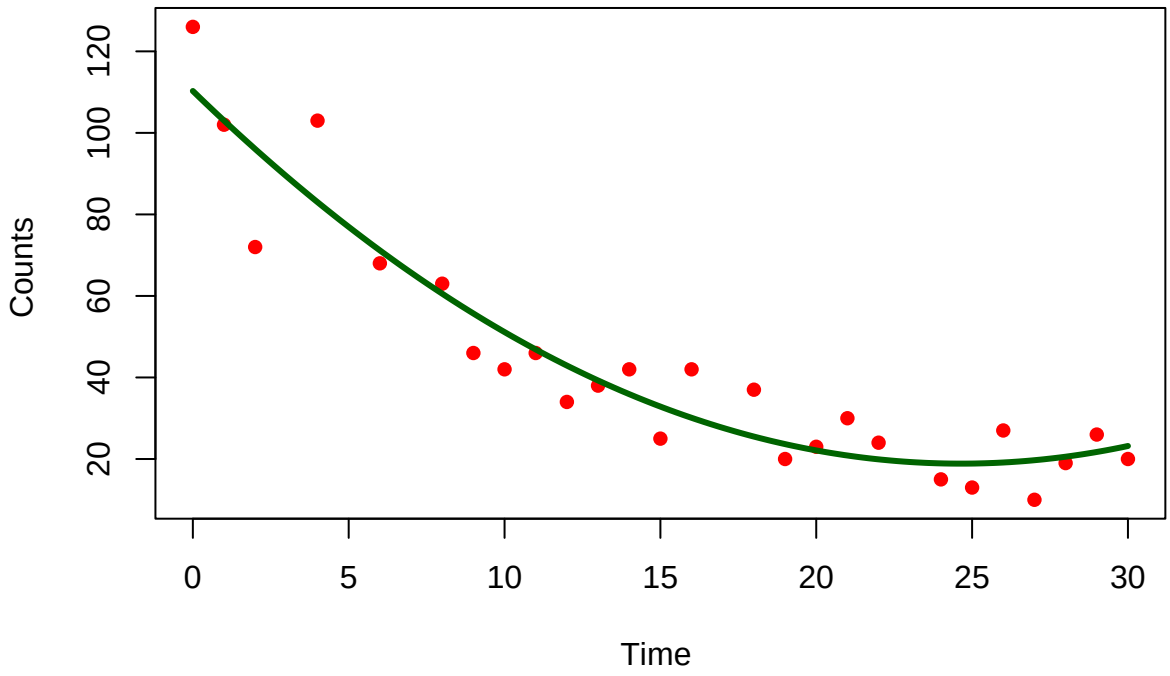



```
# modélisation par une régression polynomiale d'ordre deux:

quadratic.model <- lm(
    Counts ~ Time + I(Time^2),
    data = Comptage_new_ind
)
# 'I()' permet d'effectuer des opérations mathématiques au sein d'une formule

summary(quadratic.model)$r.squared
```

```
## [1] 0.8997921
```

```
# augmentation substantielle du coefficient de détermination: on a une
# meilleure adéquation aux données en utilisant une régression polynomiale
# plutôt qu'une régression linéaire


timevalues <- seq(0, 30, 0.1)

predictedcounts <- predict(quadratic.model, data.frame(Time = timevalues))
# prédictions du modèle polynomial sur l'intervalle de temps étudié

plot(
    Comptage_new_ind$Time, Comptage_new_ind$Counts, pch = 16, xlab = "Time ", ylab = "Counts ",
    col = "red"
```

```
)

lines(timevalues, predictedcounts, col = "darkgreen", lwd = 3)
```




```
# ajout de la courbe de régression au nuage de point
```

Linéarisation de relations non linéaires

Lorsque la relation observée a la forme caractéristique d'une **fonction connue donnée** et que celle-ci est **linéarisable** (par exemple "Y = b * exp(a * X)", "Y = a * ln(b + X)", "Y = 1 / (1 + exp(-X))"; i.e. respectivement des relations exponentielle, logarithmique ou sigmoïde qui sont linéarisables par des transformations logarithmique, exponentielle et logit), on peut **transformer la variable réponse** de manière à avoir une relation linéaire avec la variable explicative.

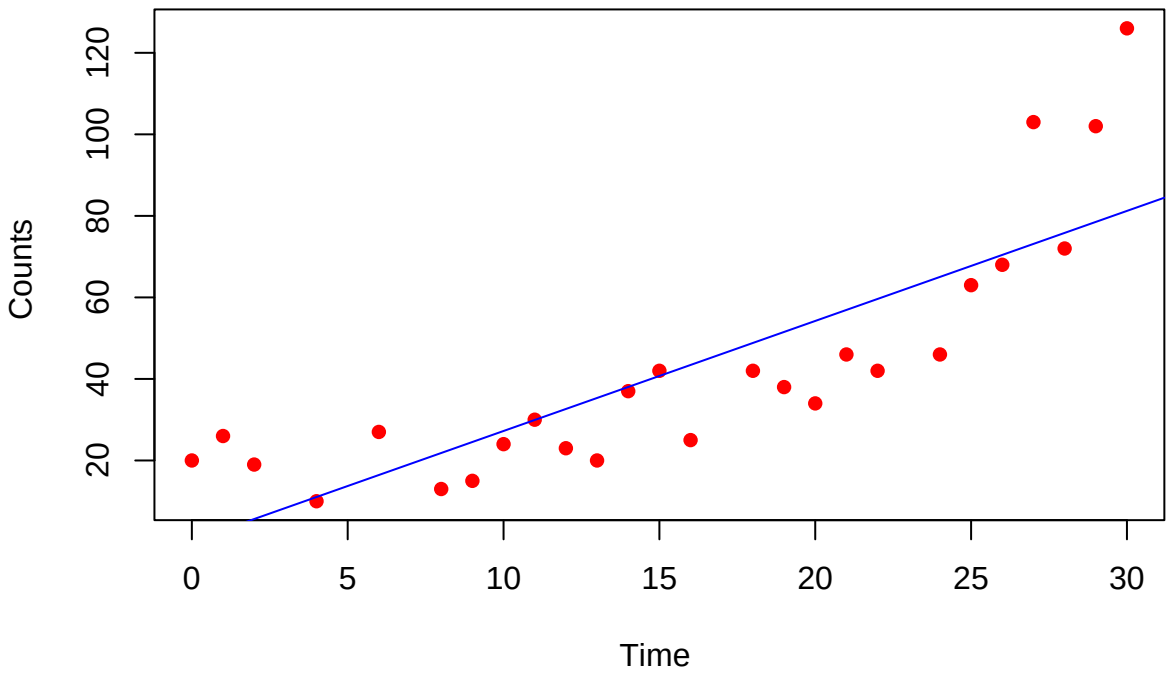
Prenons l'exemple d'une culture batérienne pour laquelle on étudie l'évolution de la densité en fonction du temps. Pendant la phase de croissance dite "exponentielle", cette évolution peut être décrite par une relation de type: "Densité = b x exp(a x temps)". Cette relation est linéarisable en utilisant une transormation logarithmique, on obtient alors: "log(Densité) = log(b) + a x temps". Etudions un exemple à partir de données simulées:

```
Growth <- data.frame(
    Time = c(
        0, 1, 2, 4, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 24,
        25, 26, 27, 28, 29, 30
```

```
    ),
    Counts = c(
        20, 26, 19, 10, 27, 13, 15, 24, 30, 23, 20, 37, 42, 25, 42, 38, 34, 46, 42,
        46, 63, 68, 103, 72, 102, 126
    )
)
# Simulation de données: nombre de bactéries en fonction du nombre de jours de
# culture



plot(
    Growth$Time, Growth$Counts, pch = 16, xlab = "Time ", ylab = "Counts ", col = "red"
)
# la relation ne semble pas linéaire (exponentielle)




# modélisation par une régression linéaire:


linear.model <- lm(Counts ~ Time, data = Growth)

summary(linear.model)$r.squared
```

```
## [1] 0.6853872
```

```
plot(
    Growth$Time, Growth$Counts, pch = 16, xlab = "Time ", ylab = "Counts ", col = "red"
)

abline(linear.model, col = "blue")
```

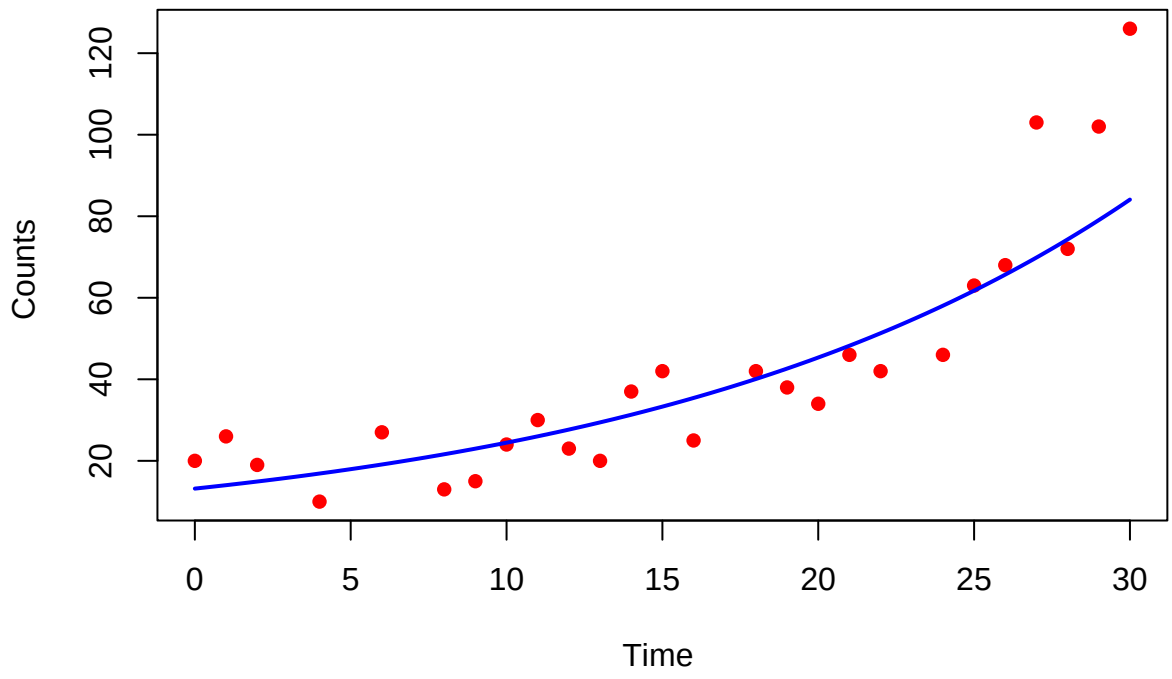



```
# modélisation par une régression linéaire après transformation logarithmique:


exponential.model <- lm(
    log(Counts) ~
        Time, data = Growth
)


summary(exponential.model)$r.squared
```

```
## [1] 0.7661261
```

```
# augmentation substantielle du coefficient de détermination: on a une
# meilleure adéquation aux données en utilisant une régression après
# tranformation logarithmique plutôt qu'une régression linéaire




timevalues <- seq(0, 30, 0.1)

Counts.exponential2 <- exp(predict(exponential.model, data.frame(Time = timevalues)))
# prédictions du modèle polynomial sur l'intervalle de temps étudié
```

```r
plot(
    Growth$Time, Growth$Counts, pch = 16, xlab = "Time ", ylab = "Counts ", col = "red"
)

lines(
    timevalues, Counts.exponential2, lwd = 2, col = "blue", xlab = "Time (h)", ylab = "Counts"
)
```




```r
# ajout de la courbe de régression au nuage de point
```

NOTE IMPORTANTE

La linéarisation de relations non linéaire permet de modéliser différentes réponses via une régression linéaire se basant sur la variable réponse transformée: on parle de modèle linéaire généralisés (ou GLM en anglais). La fonction permettant de linéariser la relation est appelée fonction de lien. Une fonction logarithmique est souvent utilisé pour modéliser une variable suivant une loi de Poisson (comptage d'événement rare), une fonction logit est souvent utilisé pour modéliser une variable suivant une loi binomiale (réponse catégorique binaire, on parle alors de régression logistique).

POUR ALLER PLUS LOIN

On peut également modéliser une variable réponse en fonction de plusieurs variables explicatives, on parle alors de modèle multivarié. Ce type de modèle est applicable aux régressions linéaire, mais également aux modèles linéaires généralisés ou aux régressions polynomiales. Ce dernier cas de figures est notamment fréquemment utilisé dans les modèles additifs pour lesquels on approche souvent la relation de chaque variable explicative à la variable réponse par des polynômes (avec de nombreux degrés).

Pour aller plus loin: les moindres carrés non-linéaires

Lorsque l'on connait le type de relation non-linéaire existant entre deux variables, on peut estimer les paramètres de cette relation par la méthode des moindres carrés. Cela peut nottament être utile pour des relations décrites par des fonctions non usuelles. Prenons l'exemple de la cinétique enzymatique (équation de Michaelis-Menten):

```
# Nouveau jeu de données vitesse de reaction d'une enzyme en fonction de la
# concentration en substrat:

vitesse_i <- c(0.043, 0.048, 0.052, 0.055, 0.057, 0.061)
# =densite optique (Do) / temps (=vitesse)

concentration <- c(1.5, 2, 2.5, 3.33, 5, 10)
# mmol.l-1


plot(
    vitesse_i ~ concentration, ylab = "Vitesse initiale", xlab = "Concentration en substrat",
    pch = 19, cex = 2, ylim = c(0.03, 0.07)
)




# Test avec la regression polynomiale

m_poly <- lm(vitesse_i ~ concentration + I(concentration^2))

coef(m_poly)
```

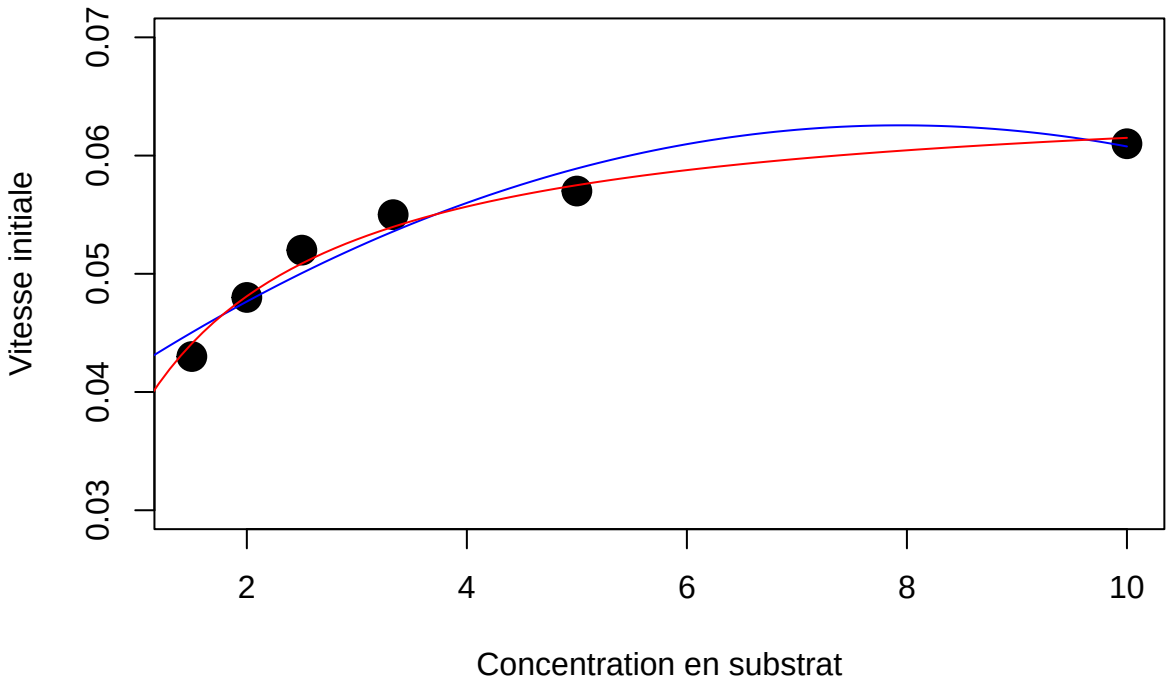
```
##        (Intercept)      concentration I(concentration^2)
##       0.0359294112       0.0067058146      -0.0004221644
```

```
m_poly <- lm(vitesse_i ~ concentration + I(concentration^2))
vec <- seq(0, 10, 0.1)
y100poly <- predict(m_poly, data.frame(concentration = vec))
lines(vec, y100poly, col = "blue")
```

```r
# bon résultat, mais on souhaiterait un plateau plutôt qu'une pente négative !


# Michaelis et Menten (1913) ont proposé un modèle non-lineaire permettant
# d'épouser correctement la presence d'un plateau après une certaine
# concentration

# Modele non-lineaire : La relation de Michaelis-Menten

# Le modele: Vi = (Vmax * [S]) / (Km + [S])

# Vi est la vitesse initiale de la reaction: c'est la vitesse de la reaction,
# la pente de la concentration en produit en fonction du temps

# Vmax est la vitesse initiale maximale. Km est la constante de Michaelis de
# l'enzyme: c'est la concentration en substrat pour laquelle Vi est égale à la
# moitie de Vmax.

# Ecriture de ce modele non lineaire:



modnl <- nls(
    vitesse_i ~ (a * concentration)/(b + concentration), start = list(a = 1, b = 1)
)
# On donne des points de départ pour l'algorithme

# Remarque: on peut écrire l'equation que l'on veut, créer son propre modele en
# fonction de nos données

# Visualisation
vec <- seq(1, 10, 0.1)
vec
```

```
##  [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4
## [16]  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9
## [31]  4.0  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4
## [46]  5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9
## [61]  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4
## [76]  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8  9.9
## [91] 10.0
```

```r
ypts <- predict(modnl, data.frame(concentration = vec))
lines(vec, ypts, col = "red")
```




```
coef(modnl)  #Vmax(=a), Km(=b) (Km = quand Vi=1/2Vmax)
```

```
##          a          b
## 0.06610005 0.74823165
```

```
# on a donc un meilleur modèle que la régression polynomiale, et avec des
# paramètres interprétables (avec un sens biologique : Vm et Km).
```

Bilan

Lorsque la **régression linéaire paraît inadaptée** (prérequis de la régression ou linéarité non vérifiés) il peut être utile de changer la forme de la régression pour obtenir une régression valide.
Si la relation semble linéaire mais que les prérequis ne sont pas vérifiés le problème peut provenir d'une **asymétrie** dans la distribution des variables modélisées. Dans ce cas une transformation de la variable réponse peut permettre de corriger le défaut de symétrie:

- transformation au **carré ou au cube**, si le **biais de symétrie est négatif** et modéré / fort
- transformation par la **racine carré ou cubique**, voire par le logarithme, si le **biais de symétrie est positif** et modéré / fort
- transformation par la **fonction inverse** pour un **biais de symétrie sévère** (notamment en présence de valeurs outliers)

Dans le cas où la relation est **non linéaire**, il convient d'adapter la modélisation en:

- utilisant une **régression polynomiale d'ordre 2** si la forme de la courbe est **incurvée ou en "U"**
- utilisant une **transformation de la variable réponse** si la relation existante est **linéarisable** (relation exponentielle linéarisée après transformation logarithmique, relation sigmoïde linéarisée après transformation logit)

On peut directement intégrer une transformation de variable dans une formule en positionnant la transformation dans l'expression **I()** (par exemple, "lm(I(Reponseˆ2) ~ Explicatif)" permet de faire une transformation de la variable réponse au carré).