SAE Régression

Binh Minh TRAN

2024-05-17

Tableau de Matières

1.Comprendre l'ensemble de données

Dans ce jeu de données, nous avons 234 voitures. Voici la description des autres variables :

- cty et hwy donnent la consommation en carburant des voitures en miles par gallon respectivement pour la conduite en ville et sur l'autoroute.
- displ est la cylindrée du moteur en litres.

- cyl est le nombre de cylindres du moteur
- drv est le mode de transmission : traction avant (f comme front), propulsion (r comme rear) ou quatre roues motrices (4).
- modèle est le modèle de la voiture. Il y a 38 modèles, sélectionnés avec différentes versions entre 1999 et 2008.
- class est une variable catégorielle décrivant le « type » de voiture : deux places, SUV, compacte, etc.

La variable cible à modéliser est cty.

1.1 Chargement du jeu de données.

```
Consommations <- read.csv2(file = "Consommations.csv", header = TRUE,
    sep = ";", dec = ",", stringsAsFactors = TRUE)

# Transformation
Consommations$year <- as.factor(Consommations$year)
```

1.2 Analyse Exploratoire des Données (EDA).

```
head(Consommations)  #6 première observations
```

```
##   manufacturer model displl year cyl      trans drv cty hwy fl   class
## 1         audi    a4   1.8 1999   4   auto(l5)   f  18  29  p compact
## 2         audi    a4   1.8 1999   4 manual(m5)   f  21  29  p compact
## 3         audi    a4   2.0 2008   4 manual(m6)   f  20  31  p compact
## 4         audi    a4   2.0 2008   4   auto(av)   f  21  30  p compact
## 5         audi    a4   2.8 1999   6   auto(l5)   f  16  26  p compact
## 6         audi    a4   2.8 1999   6 manual(m5)   f  18  26  p compact
```

```
summary(Consommations)
```

```
##      manufacturer                  model         displ          year
##  dodge     :37    caravan 2wd        : 11   Min.   :1.600   1999:117
##  toyota    :34    ram 1500 pickup 4wd: 10   1st Qu.:2.400   2008:117
##  volkswagen:27    civic              :  9   Median :3.300
##  ford      :25    dakota pickup 4wd  :  9   Mean   :3.472
##  chevrolet :19    jetta              :  9   3rd Qu.:4.600
##  audi      :18    mustang            :  9   Max.   :7.000
##  (Other)   :74    (Other)            :177
##       cyl              trans    drv          cty             hwy
##  Min.   :4.000   auto(l4)  :83   4:103   Min.   : 9.00   Min.   :12.00
##  1st Qu.:4.000   manual(m5):58   f:106   1st Qu.:14.00   1st Qu.:18.00
##  Median :6.000   auto(l5)  :39   r: 25   Median :17.00   Median :24.00
##  Mean   :5.889   manual(m6):19           Mean   :16.86   Mean   :23.44
##  3rd Qu.:8.000   auto(s6)  :16           3rd Qu.:19.00   3rd Qu.:27.00
##  Max.   :8.000   auto(l6)  : 6           Max.   :35.00   Max.   :44.00
##                  (Other)   :13
```

```
##  fl             class
##  c:  1   2seater   : 5
##  d:  5   compact   :47
##  e:  8   midsize   :41
##  p: 52   minivan   :11
##  r:168   pickup    :33
##          subcompact:35
##          suv       :62
```

```
str(Consommations)
```

```
## 'data.frame':    234 obs. of  11 variables:
##  $ manufacturer: Factor w/ 15 levels "audi","chevrolet",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ model       : Factor w/ 38 levels "4runner 4wd",..: 2 2 2 2 2 2 2 3 3 3 ...
##  $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : Factor w/ 2 levels "1999","2008": 1 1 2 2 1 1 2 1 1 2 ...
##  $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : Factor w/ 10 levels "auto(av)","auto(l3)",..: 4 9 10 1 4 9 1 9 4 10 ...
##  $ drv         : Factor w/ 3 levels "4","f","r": 2 2 2 2 2 2 2 1 1 1 ...
##  $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : Factor w/ 5 levels "c","d","e","p",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ class       : Factor w/ 7 levels "2seater","compact",..: 2 2 2 2 2 2 2 2 2 2 ...
```

1.3 Gestion des données.

```
is.null(Consommations)
```

```
## [1] FALSE
```

```
# vérifier la valeur nulle
(col_has_na <- sapply(Consommations, function(x) any(is.na(x))))
```

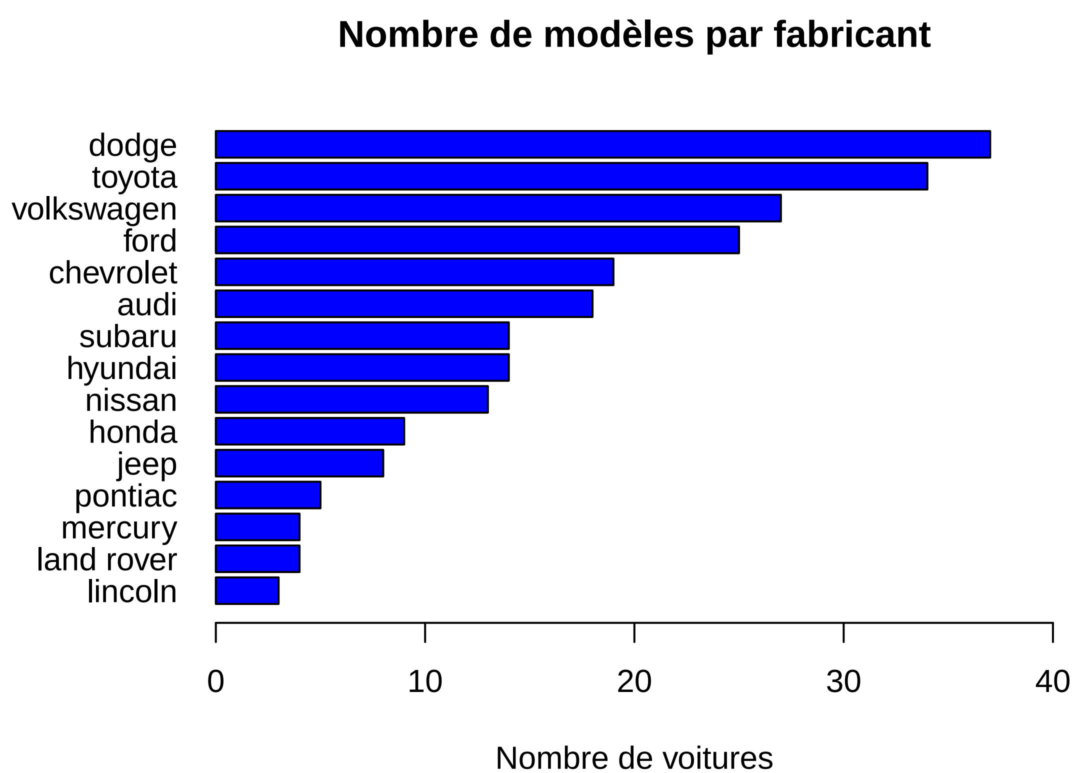
```
## manufacturer        model        displ         year          cyl        trans
##        FALSE        FALSE        FALSE        FALSE        FALSE        FALSE
##          drv          cty          hwy           fl        class
##        FALSE        FALSE        FALSE        FALSE        FALSE
```

1.4 Total de voitures.

```
# Table de l'effectif des modèles de manufacturer
(Eff_Manu <- table(Consommations$manufacturer))
```

```
##
##       audi  chevrolet      dodge       ford      honda    hyundai       jeep
##         18         19         37         25          9         14          8
## land rover    lincoln    mercury     nissan    pontiac     subaru     toyota
##          4          3          4         13          5         14         34
## volkswagen
##         27
```

```
EffManuSort <- sort(Eff_Manu, decreasing = FALSE)
par(mar = c(5, 10, 4, 0.5) + 0.1)
barplot(EffManuSort, xlab = "Nombre de voitures", main = "Nombre de modèles par fabricant",
    horiz = TRUE, xlim = c(0, 40), las = 1, col = "blue")
```



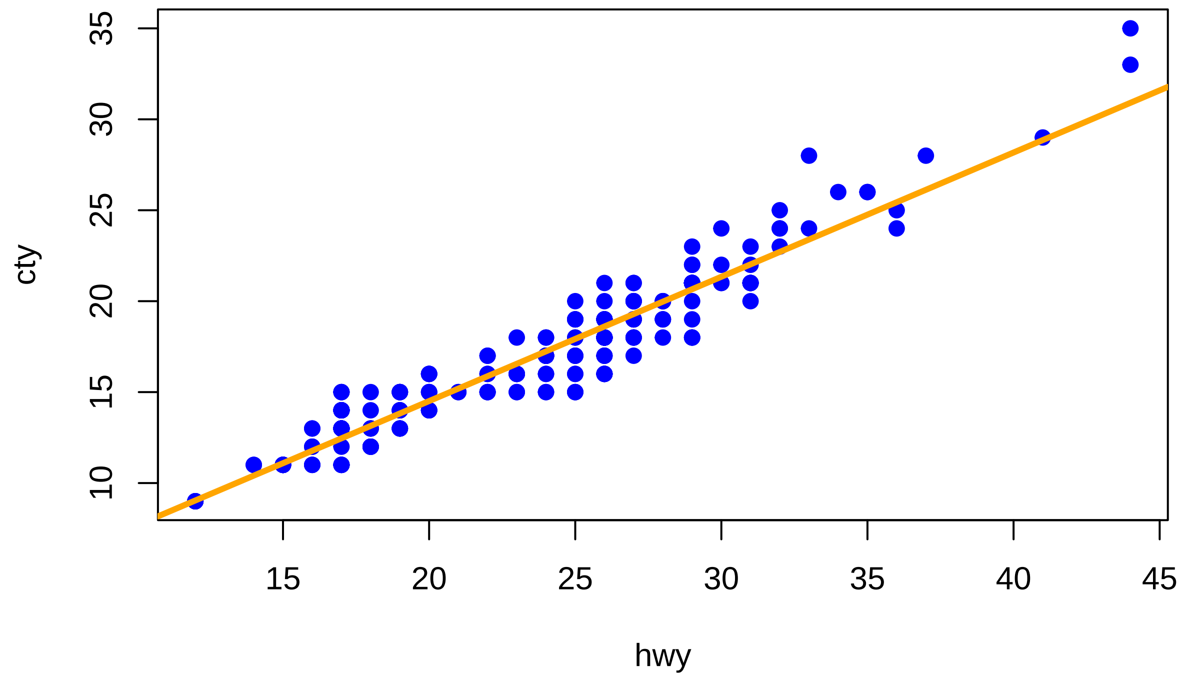
(*) Chaque modèle n'a pas qu'un seul type

2. La relation entre cty et hwy.

```
# la corrélation de coefficient
cor(Consommations$cty, Consommations$hwy)
```

```
## [1] 0.9559159
```

```
plot(cty ~ hwy, data = Consommations, col = "blue", bg = "blue",
    pch = 21)
reglin <- lm(cty ~ hwy, data = Consommations)
abline(reglin, col = "orange", lwd = 3)
```



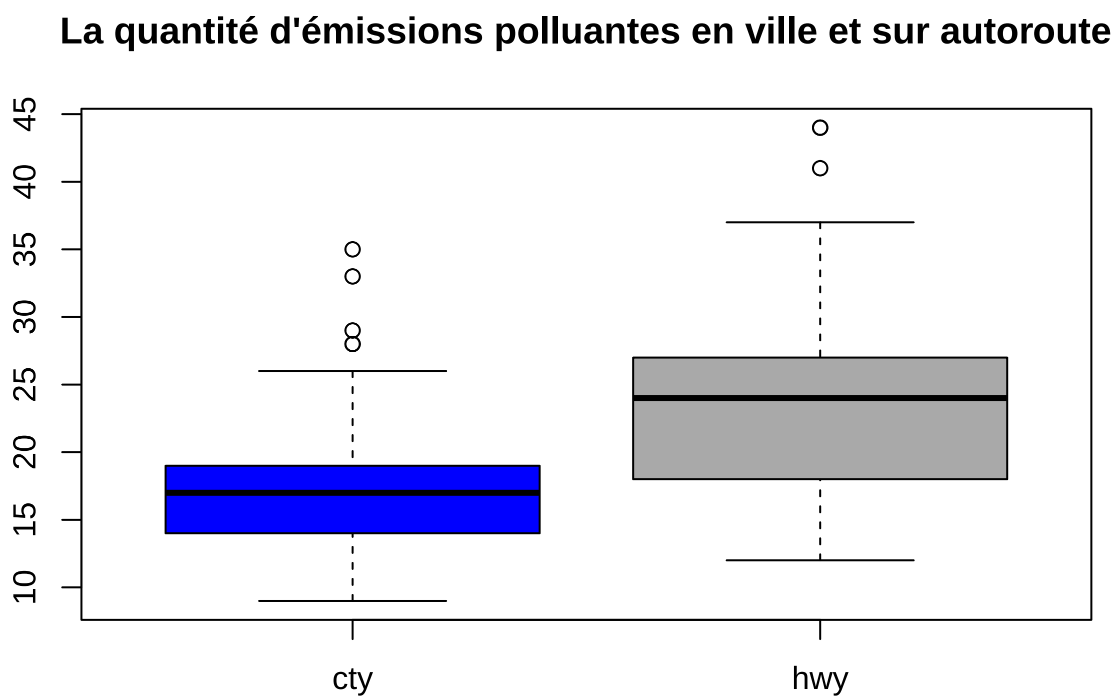

```
summary(reglin)
```

```
##
## Call:
## lm(formula = cty ~ hwy, data = Consommations)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9247 -0.7757 -0.0428  0.6965  4.6096
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.84420    0.33319   2.534   0.0119 *
## hwy          0.68322    0.01378  49.585   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.252 on 232 degrees of freedom
## Multiple R-squared:  0.9138, Adjusted R-squared:  0.9134
## F-statistic:  2459 on 1 and 232 DF,  p-value: < 2.2e-16
```

Commentaire: Les points sont répartis uniformément autour de la ligne de régression, montrant une relation claire et stable entre les deux variables avec une dispersion modérée. La tendance croissante et la ligne de régression indiquent une forte corrélation positive (R=95%) entre la consommation de carburant en ville et sur autoroute.

Conclusion : Cela indique qu'il existe une relation linéaire forte et positive entre les deux variables, ce qui signifie que les véhicules ayant une haute efficacité de carburant sur autoroute tendent également à avoir une haute efficacité de carburant en ville.

```
boxplot(Consommations$cty, Consommations$hwy, names = c("cty",
    "hwy"), main = "La quantité d'émissions polluantes en ville et sur autoroute",
    col = c("blue", "darkgrey"))
```



(*) Miles par gallon.

(*) Plus le nombre de miles par gallon (MPG) est faible, plus la consommation de carbone est élevée.

Conclusion: La consommation de carburant des voitures en milles par gallon a tendance à être plus faible en ville qu'en autoroute. Autrement dit, la consommation de carburant des voitures a tendance à être plus faible en autoroute qu'en ville.

3. Modélisation de la variable cty.

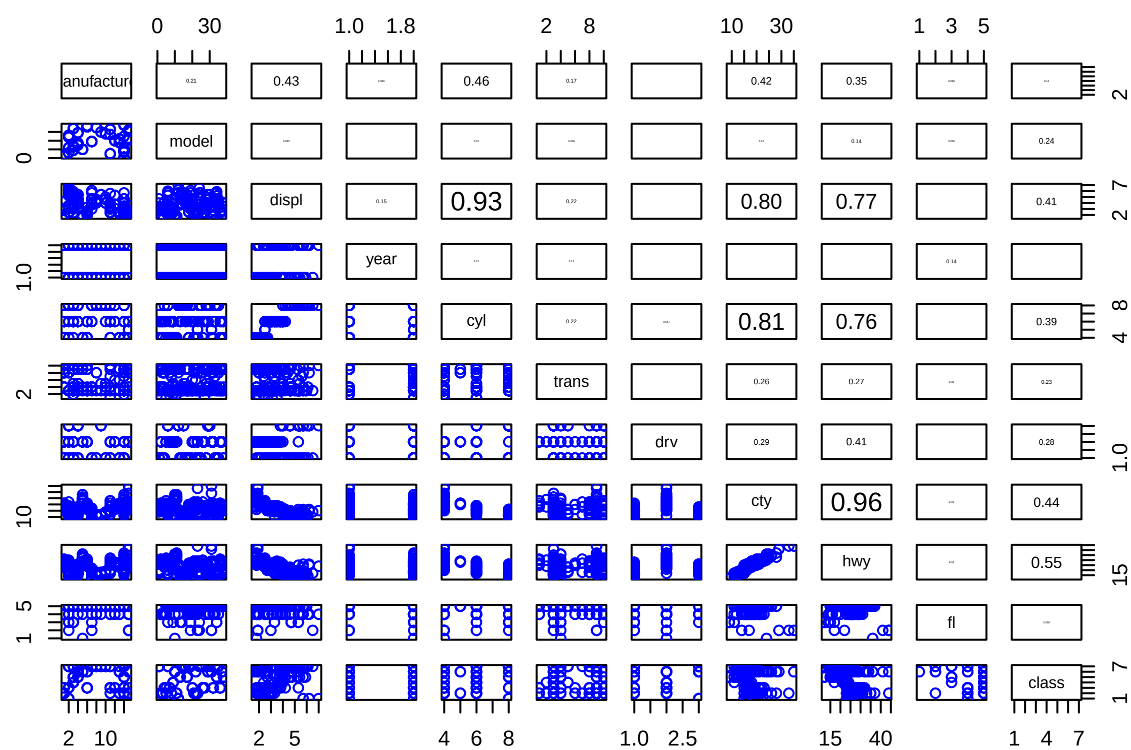
3 modèles ont été utilisés pour prédire la variable `cty` :

Tableau des formules de Régression

Modèle	Formule
Régression linéaire	$y = a + b \cdot x$
Régression exponentielle	$y = b \cdot a^x$
Régression logarithmique	$y = a \cdot \log(x) + b$

Tout d'abord, il est nécessaire de déterminer la variable indépendante (prédicteur) à inclure dans le modèle.

```
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor,
    ...) {
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if (missing(cex.cor))
        cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}
pairs(Consommations, upper.panel = panel.cor, col = "blue")
```



les variables qui sont fortement corrélée avec cty sont `displ` et `cyl`.

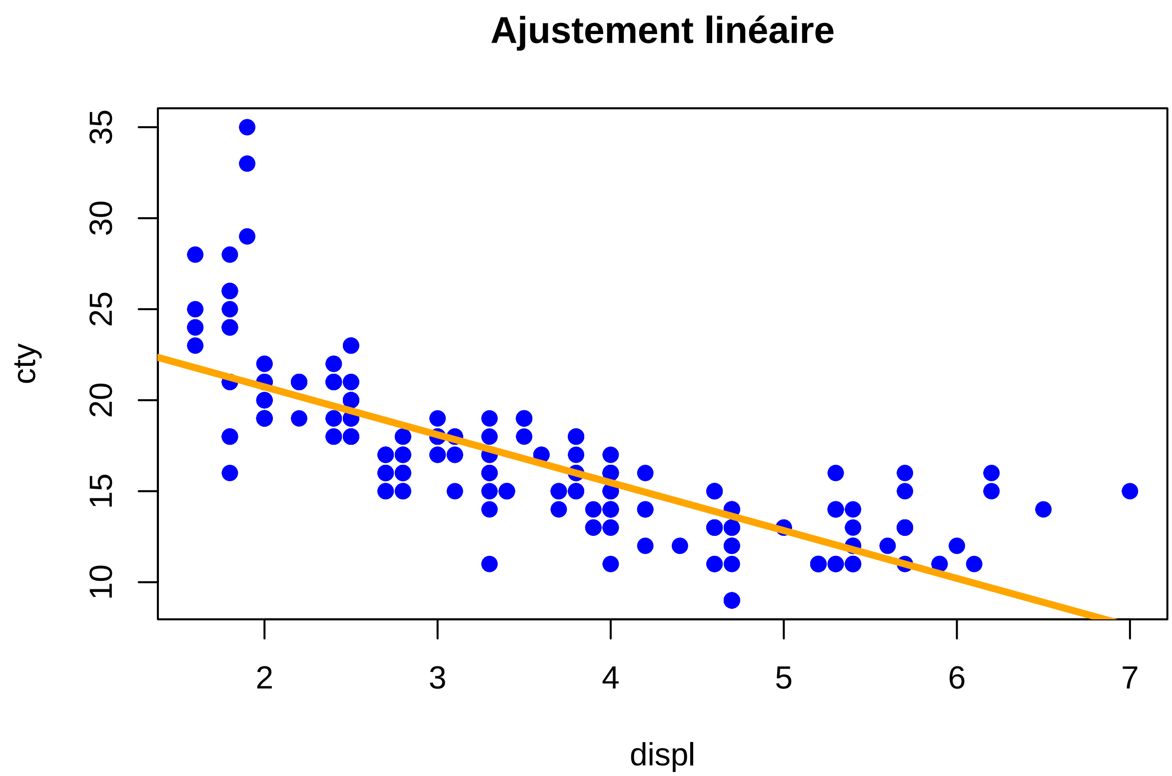
- `displ` est la cylindrée du moteur en litres.
- `cyl` est le nombre de cylindres du moteur

3.1 Le modèle linéaire.

```
# la corrélation de coéfficient
cor(Consommations$cty, Consommations$displ)
```

```
## [1] -0.798524
```

```
plot(cty ~ displ, data = Consommations, col = "blue", bg = "blue",
    pch = 21, main = "Ajustement linéaire")
reglinDispl <- lm(cty ~ displ, data = Consommations)
abline(reglinDispl, col = "orange", lwd = 3.5)
```



> **Commentaire** : Il existe une relation linéaire forte et négative entre la cylindrée et la consommation de carburant en ville. Cela signifie que les véhicules avec une plus grande cylindrée tendent à avoir une consommation de carburant plus élevée en ville.

3.2 Le modèle exponentiel.

3.2.1 Ajustement exponentiel avec modèle non-linéaire nls().

D'abord, il faut trouver les valeurs initiales de a et b pour le modèle non-linéaire `nls()`, afin d'atteindre plus rapidement le point de convergence. Ces valeurs de a et b ne sont pas encore le couple final qui minimise le RSS $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, mais elles peuvent aider à accélérer le processus de convergence vers la solution optimale (où le RSS est minimisé).

```
# Fonction pour essayer différentes valeurs initiales
finda_b <- function(data, formula, a_vals, b_vals) {
    for (a in a_vals) {
        for (b in b_vals) {
            try({
                model <- nls(formula, data = data, start = list(a = a,
                  b = b))
                return(list(a = a, b = b))
            }, silent = TRUE)
        }
    }
    stop("aucune valeur favorable")
}

# Initialiser les valeurs initiales de a et b
a_vals <- seq(1, 3, by = 1)
b_vals <- seq(1, 3, by = 1)


resultat <- finda_b(Consommations, cty ~ b * a^(displ), a_vals,
    b_vals)
print(resultat)
```

```
## $a
## [1] 1
##
## $b
## [1] 3
```

On utilise la fonction `nls` (moindres carrés non linéaires) avec a et b récupérés du résultat de la fonction `finda_b` pour exécuter l'équation $y = b \cdot a^x$.

```
(regExp <- nls(cty ~ b * a^(displ), data = Consommations, start = list(a = resultat$a,
    b = resultat$b)))
```

```
## Nonlinear regression model
##   model: cty ~ b * a^(displ)
##    data: Consommations
##       a       b
##  0.8402 30.0845
##  residual sum-of-squares: 1358
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 7.214e-06
```

```
summary(regExp)
```

```
##
## Formula: cty ~ b * a^(displ)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
```

```
## a  0.840189   0.006935  121.15   <2e-16 ***
## b 30.084529   0.785962   38.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.42 on 232 degrees of freedom
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 7.214e-06
```

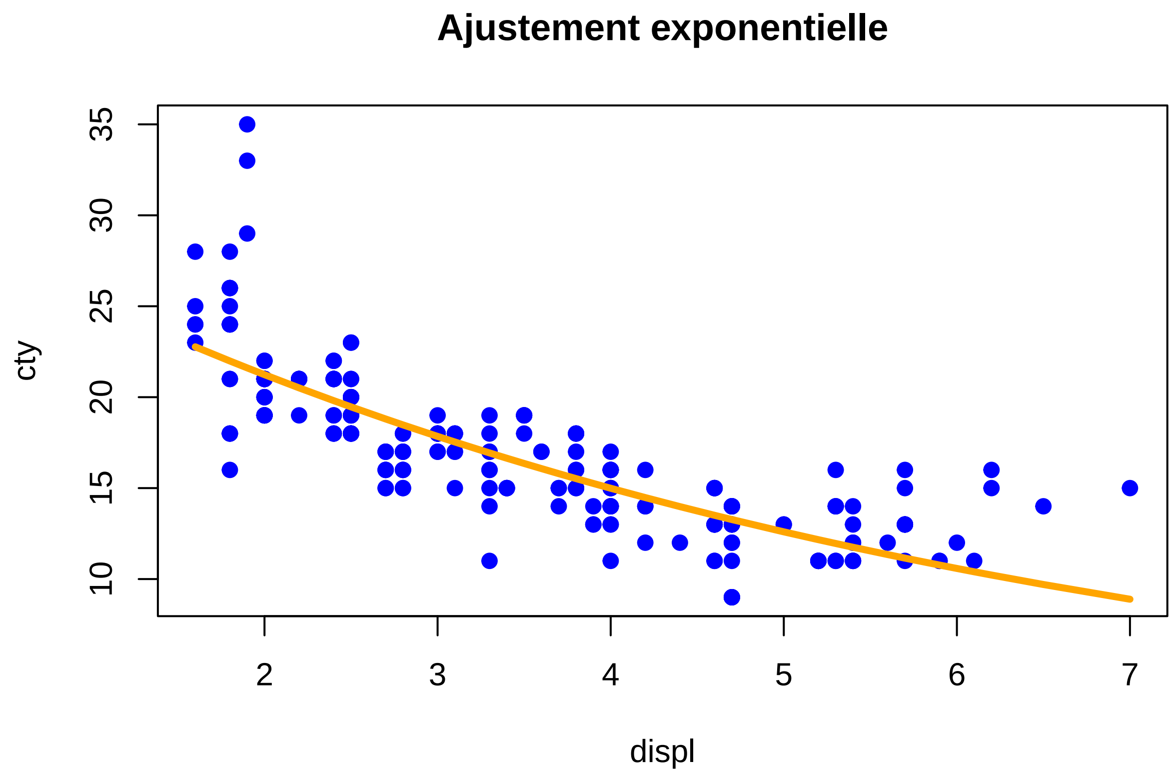
La fonction ajustée est : $y = 30.08 \cdot 0.84^x$

On doit après ordonner les valeurs de `displ` dans l'ordre croissant. Le but est d'éviter les discontinuités dans la courbe de prédiction car c'est un modèle non linéaire.

```
# sorted_index renvoie la permutation des indices des
# éléments de `displ` qui les trierait dans l'ordre
# croissant
sorted_index <- order(Consommations$displ)
sorted_displ <- Consommations$displ[sorted_index]
sorted_cty <- Consommations$cty[sorted_index]
sorted_pred <- predict(regExp)[sorted_index]
```

```
# trace un nuage de point
plot(cty ~ displ, data = Consommations, main = "Ajustement exponentielle",
    xlab = "displ", ylab = "cty", pch = 19, col = "blue")

# tracer une courbe de la fonction exponentielle
lines(sorted_displ, sorted_pred, col = "orange", lwd = 3.5)
```



3.2.2 Ajustement exponentiel avec modèle linéaire.

Une fonction exponentielle dans la forme $y = b \cdot a^x$.

Étape 1: Transformer l'équation en logarithme.

Logarithme népérien

$$\begin{aligned} y &= b \cdot a^x \\ \log(y) &= \log(b \cdot a^x) \\ &= \log(b) + \log(a^x) \\ &= \log(b) + x \cdot \log(a) \end{aligned}$$

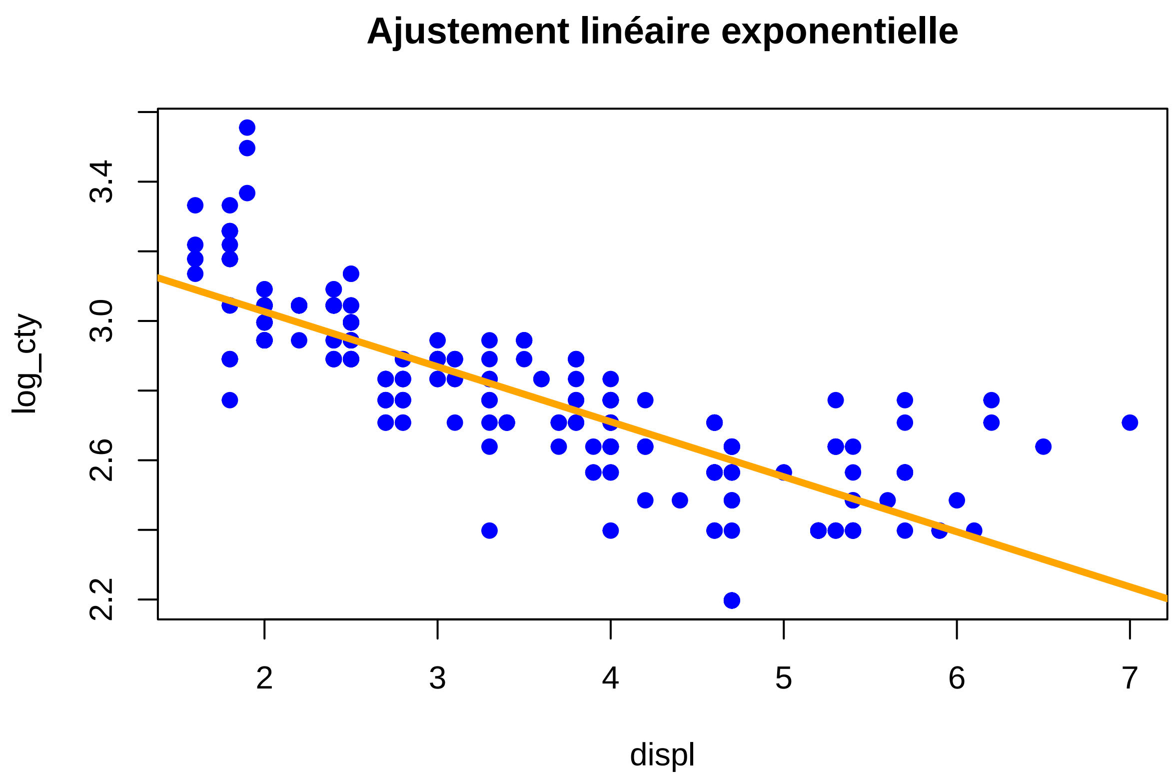
Ainsi, nous obtenons une nouvelle équation linéaire avec $\log(y)$ comme variable dépendante et x comme variable indépendante.

```
log_cty <- log(Consommations$cty)
plot(log_cty ~ displ, data = Consommations, main = "Ajustement linéaire exponentielle",
    xlab = "displ", ylab = "log_cty", col = "blue", bg = "blue",
    pch = 21)
(linExp <- lm(log_cty ~ displ, data = Consommations))
```

```
##
## Call:
```

```
## lm(formula = log_cty ~ displ, data = Consommations)
##
## Coefficients:
## (Intercept)        displ
##       3.343       -0.158
```

```
abline(linExp, col = "orange", lwd = 3.5)
```



On obtient l'équation suivante: $\log(y) = -0.158x + 3.343$. À présent, la relation de corrélation est entre `log_cty` et `displ`. **Étant donné que la variable cible est `cty`, nous devons continuer à transformer la relation.**

Étape 2: Trouver la courbe de prédiction en fonction de la relation entre `cty` et `displ`. Le but est d'assurer que la variable cible soit `cty` et non `log_cty`.

On transforme l'équation obtenue :

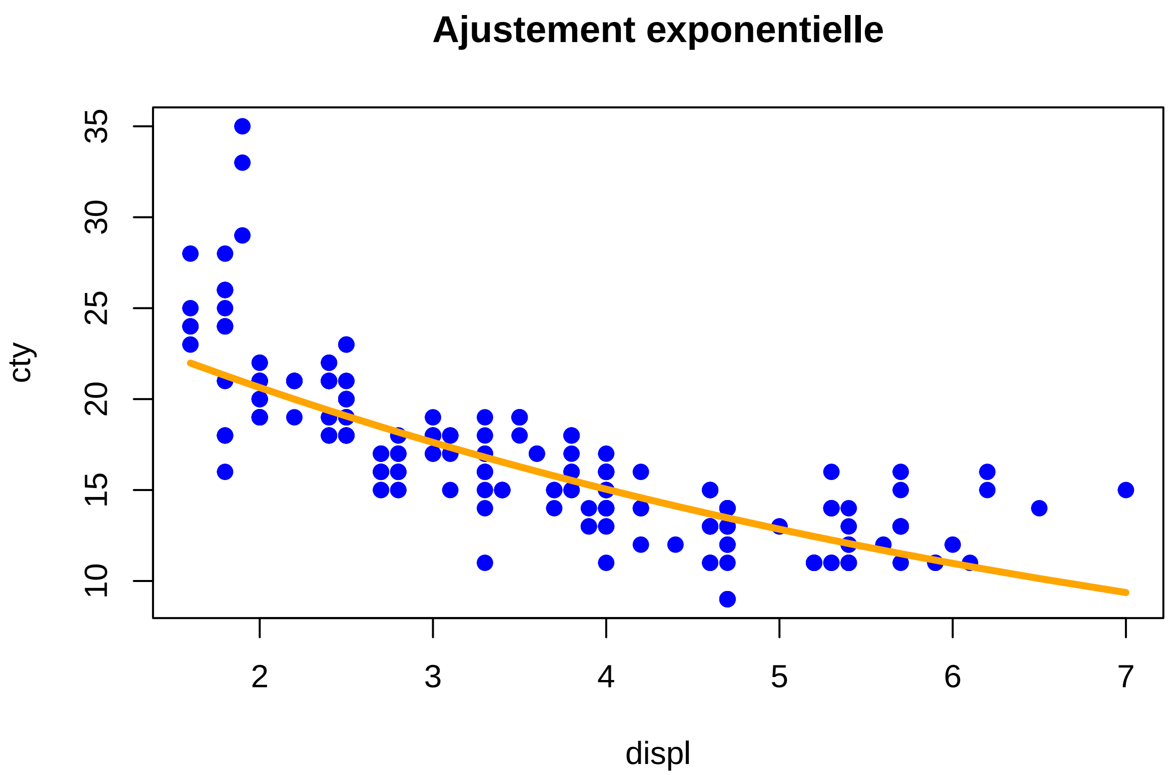
$$y = e^{-0.158 \cdot x + 3.343}$$

```
# y stocke les predictions de y par apport à l'équation au
# dessus.
y <- exp(-0.158 * Consommations$displ + 3.343)

# ordonner les paires de valeurs des variables
# indépendantes et dépendantes afin d'assurer la continuité
# de la courbe.
```

```
sorted_index <- order(Consommations$displ)
sorted_displ <- Consommations$displ[sorted_index]
sorted_y <- y[sorted_index]

# tracer un graphique
plot(cty ~ displ, data = Consommations, main = "Ajustement exponentielle",
    xlab = "displ", ylab = "cty", pch = 19, col = "blue")
lines(sorted_displ, sorted_y, col = "orange", lwd = 3.5)
```




```
# RSS
sum((sorted_cty - sorted_y)^2)
```

```
## [1] 1390.929
```

```
# R^2
(cov(log_cty, Consommations$displ)/(sd(log_cty) * sd(Consommations$displ)))^2
```

```
## [1] 0.6714293
```

Conclusion: À partir du RSS de ce méthode on peut dire que la premiere methode (retrouve dans 4.2.1) est plus optimise car le RSS de la premiere methode(1358) > celle de la deuxieme methode (1390).

3.3 Le modèle logarithmique.

3.3.1 Ajustement logarithmique avec modèle non-linéaire nls().

D'abord, on doit trouver les valeurs initiales de a et b pour le modèle non-linéaire `nls()`, afin d'atteindre plus rapidement le point de convergence. Ces valeurs de a et b ne sont pas encore le couple final qui minimise le RSS $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, mais elles peuvent aider à accélérer le processus de convergence vers la solution optimale (où le RSS est minimisé).

```
# Fonction pour essayer différentes valeurs initiales
finda_b <- function(data, formula, a_vals, b_vals) {
    for (a in a_vals) {
        for (b in b_vals) {
            try({
                model <- nls(formula, data = data, start = list(a = a,
                  b = b))
                return(list(a = a, b = b))
            }, silent = TRUE)
        }
    }
    stop("pas de valeur favorable")
}

# Initialiser les valeurs initiales de a et b
a_vals <- seq(1, 3, by = 1)
b_vals <- seq(1, 3, by = 1)


resultat <- finda_b(Consommations, cty ~ a * log(displ) + b,
    a_vals, b_vals)
print(resultat)
```

```
## $a
## [1] 1
##
## $b
## [1] 1
```

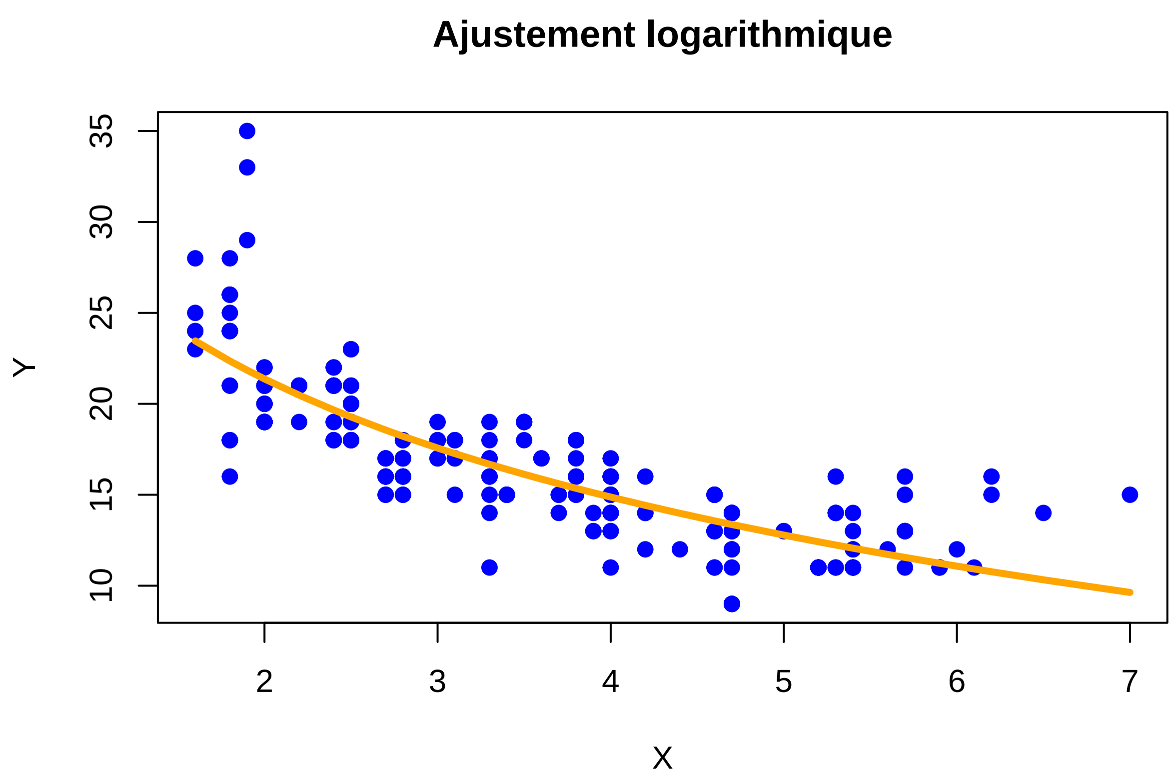
```
(RegLog <- nls(cty ~ a * log(displ) + b, data = Consommations,
    start = list(a = resultat$a, b = resultat$b)))
```

```
## Nonlinear regression model
##   model: cty ~ a * log(displ) + b
##    data: Consommations
##      a      b
## -9.369 27.861
##  residual sum-of-squares: 1281
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 7.306e-09
```

La fonction ajustée est : $y = -9.369 \cdot log(x) + 27.861$

```
# sorted_index renvoie la permutation des indices des
# éléments de `displ` qui les trierait dans l'ordre
# croissant
sorted_index <- order(Consommations$displ)
sorted_displ <- Consommations$displ[sorted_index]
sorted_cty <- Consommations$cty[sorted_index]
sorted_pred <- predict(RegLog)[sorted_index]
```

```
plot(Consommations$displ, Consommations$cty, main = "Ajustement logarithmique",
    xlab = "X", ylab = "Y", pch = 19, col = "blue")
# tracer un courbe du fonction logarithme
lines(sorted_displ, sorted_pred, col = "orange", lwd = 3.5)
```



3.3.2 Ajustement logarithmique avec modèle linéaire.

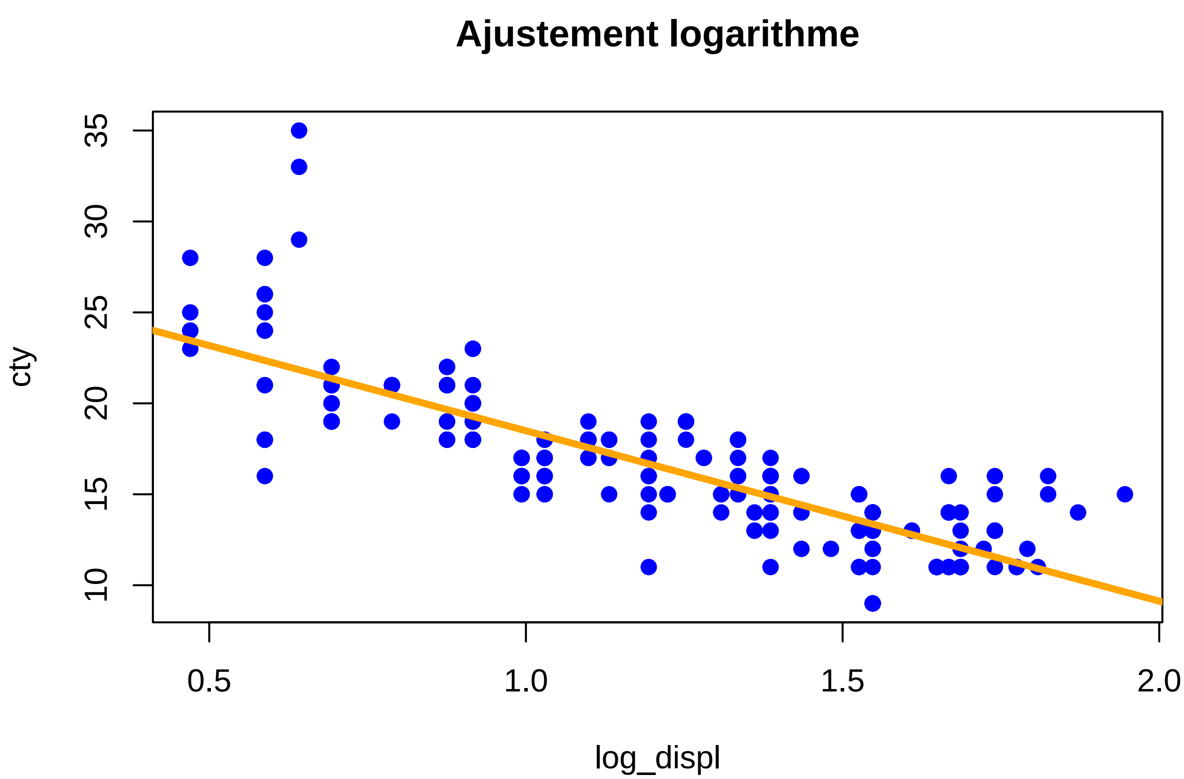
Une fonction logarithmique dans la forme $y = a \cdot \log(x) + b$:

Étape 1: nous obtenons une nouvelle équation linéaire avec $\log(x)$ comme variable indépendante et y comme variable dépendante.

```
# transform displ en log
log_displ <- log(Consommations$displ)
# trace un graphique
(ReglinLog <- lm(cty ~ log_displ, data = Consommations))
```

```
##
## Call:
## lm(formula = cty ~ log_displ, data = Consommations)
##
## Coefficients:
## (Intercept)    log_displ
##      27.861       -9.369
```

```
plot(cty ~ log_displ, data = Consommations, col = "blue", main = "Ajustement logarithme",
    bg = "blue", pch = 21)
abline(ReglinLog, col = "orange", lwd = 3.5)
```



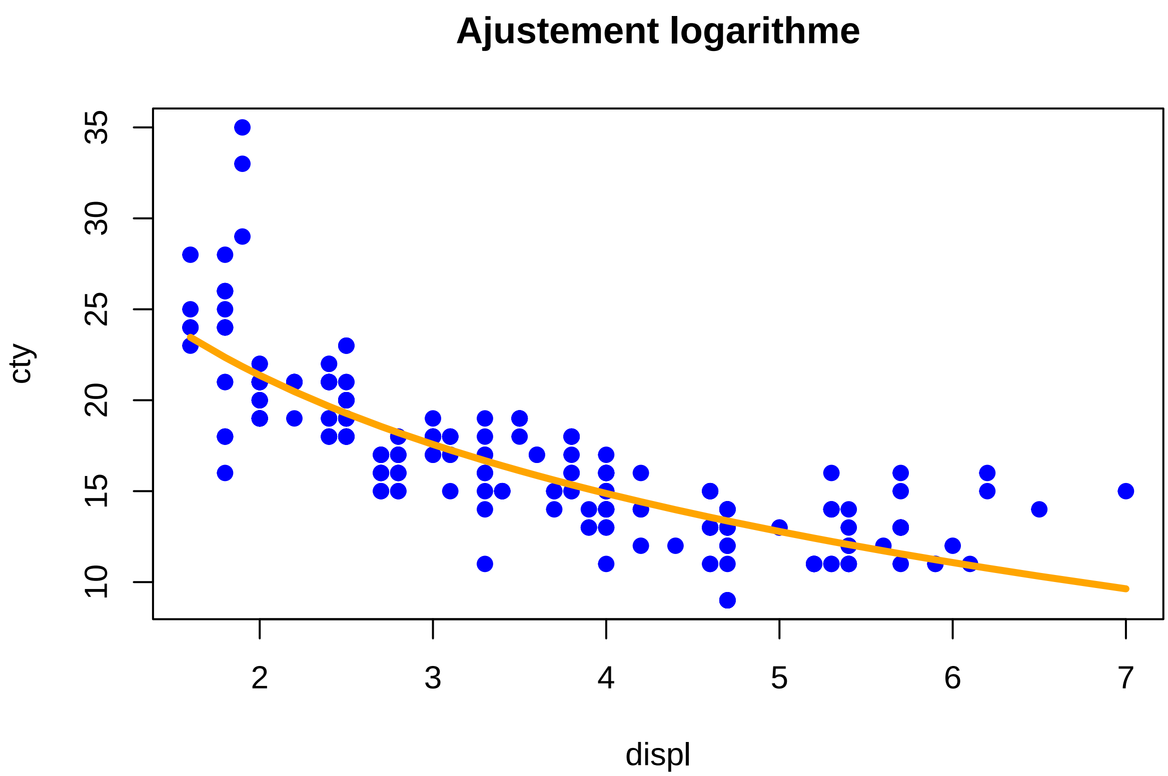
On obtient l'équation suivante : $y = -9.369 \cdot log(x) + 27.861$. À présent, la relation de corrélation est entre `cty` et `log_displ`.

Étape 2: Trouver la courbe de prédiction en fonction de la relation entre `cty` et `displ`.

On applique l'equation obtenue pour trouver la courbe de prediction.

```
# sorted_index renvoie la permutation des indices des
# éléments de `displ` qui les trierait dans l'ordre
# croissant
sorted_index <- order(Consommations$displ)
sorted_displ <- Consommations$displ[sorted_index]
sorted_cty <- Consommations$cty[sorted_index]
sorted_pred_lin_log <- predict(ReglinLog)[sorted_index]
```

```r
plot(cty ~ displ, data = Consommations, col = "blue", main = "Ajustement logarithme",
    bg = "blue", pch = 21)
lines(sorted_displ, sorted_pred_lin_log, col = "orange", lwd = 3.5)
```




```r
# RSS
sum((sorted_cty - sorted_pred_lin_log)^2)
```

```
## [1] 1281.351
```

> **Conclusion:** Les deux méthodes du modèle logarithmique renvoient un RSS de 1281. Cela signifie que l'on peut efficacement utiliser l'une des deux méthodes.

4. Évaluation de modèle.

Les critères d'évaluation utilisés sont les suivants : RSS, R², RMSE.

- RMSE (Root Mean Square Error): Erreur quadratique moyenne

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

- RSS (Residual Sum of Squares): Somme des carrés des résidus

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- R^2 (Coefficient of Determination): Coefficient de détermination

$$R^2 = 1 - \frac{RSS}{TSS}$$

où $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$ est la somme des carrés totaux.

4.1 Évaluation du modèle linéaire.

```
lin_model <- lm(cty ~ displ, data = Consommations)
(lin_rmse <- (sqrt(mean(resid(lin_model)^2))))  #Residus = prédiction - observation
```

```
## [1] 2.556442
```

```
(lin_rss <- (sum(resid(lin_model)^2)))
```

```
## [1] 1529.282
```

```
(lin_r2 <- summary(lin_model)$r.squared)
```

```
## [1] 0.6376405
```

> **Conclusion**: $R^2 \sim 65\%$, `displ` ne peut pas représenter la totalité de la variation de `cty`, seulement à 63%, le reste étant représenté par les résidus.

4.2 Évaluation du modèle exponentiel.

```
exp_model <- nlExp <- nls(cty ~ b * a^(displ), data = Consommations,
    start = list(a = 2, b = 1))
(exp_rmse <- (sqrt(mean(resid(exp_model)^2))))  #Residus = prédiction - observation
```

```
## [1] 2.409252
```

```
(exp_rss <- (sum(resid(exp_model)^2)))
```

```
## [1] 1358.252
```

```
(exp_r2 <- summary(linExp)$r.squared)
```

```
## [1] 0.6714293
```

> **Conclusion** : residual sum-of-squares (RSS): 1358, en moyenne, la différence entre les valeurs prédites et observées est d'environ 1.72, mais il y a plusieurs aberrantes. `displ` ne peut pas représenter la totalité de la variation de `cty`, seulement à 67%

4.3 Évaluation du modèle logarithmique.

```r
log_model <- nlExp <- nls(cty ~ a * log(displ) + b, data = Consommations,
    start = list(a = 1, b = 1))
(log_rmse <- (sqrt(mean(resid(log_model)^2))))  #Residus = prédiction - observation
```

```
## [1] 2.340056
```

```r
(log_rss <- (sum(resid(log_model)^2)))
```

```
## [1] 1281.351
```

```r
(log_r2 <- summary(ReglinLog)$r.squared)
```

```
## [1] 0.6963872
```

> **Conclusion** : R^2 explique qu'environ 70% de la variance de `cty` peut être expliquée par la variable `log_displ`, 30% étant représenté par les résidus.

5. Comparer les performances des modèles.

```r
# Représentation des résidus et des valeurs ajustées
par(mfrow = c(1, 3))
par(mar = c(4.5, 4.5, 4.5, 4.5))
# Lineaire
plot(fitted(reglinDispl), resid(reglinDispl), col = "blue", bg = "blue",
    pch = 21, main = "Résiduels vs Ajustés (LIN)", xlab = "Ajustés",
    ylab = "Résiduels")
abline(h = 0, col = "orange", lwd = 2)
# Exp
plot(fitted(regExp), resid(regExp), xlab = "Ajustés", ylab = "Résiduels",
    main = "Résiduels vs Ajustés (EXP)", col = "blue", bg = "blue",
    pch = 21)
abline(h = 0, col = "orange", lwd = 2)
# Log
plot(fitted(RegLog), resid(RegLog), xlab = "Ajustés", ylab = "Résiduels",
    main = "Résiduels vs Ajustés (LOG)", col = "blue", bg = "blue",
    pch = 21)
abline(h = 0, col = "orange", lwd = 2)
```

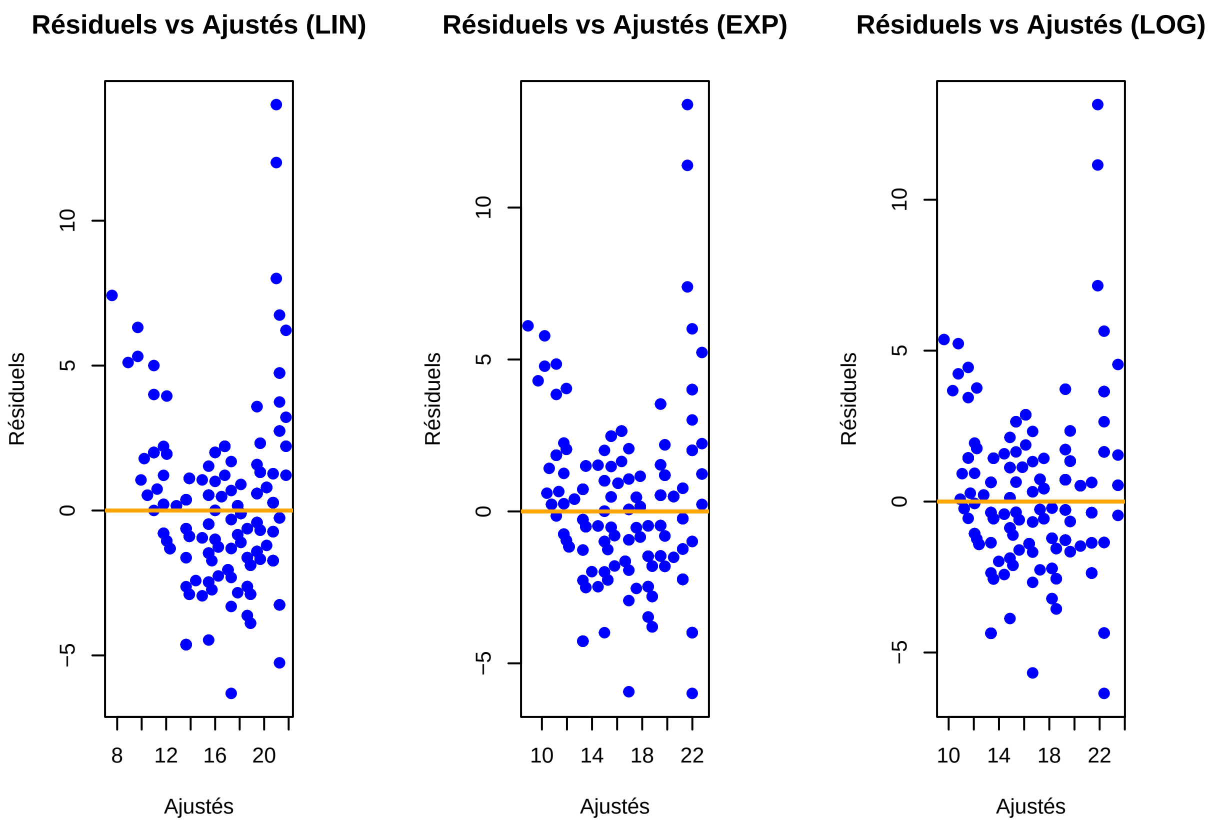


Tableau comparaison de l'efficacité des modèles:

Modèle	Régression Linéaire	Régression Exponentielle	Régression Logarithmique
RSS	1529.2823999	1358.2521972	1281.3512733
RMSE	2.5564418	2.4092523	2.3400556
R^2	0.6376405	0.6714293	0.6963872

Critiques :

1. Régression Logarithmique

- **Forces :**
  - Ce modèle présente les meilleures performances sur les trois métriques (RSS, RMSE et R-carré).
  - Cela suggère qu'une relation logarithmique entre les variables indépendantes et dépendantes offre le meilleur ajustement pour les données données.
- **Faiblesses :**
  - L'applicabilité d'un modèle logarithmique dépend de la nature des données.
  - Si les données contiennent des valeurs nulles ou négatives, une transformation logarithmique peut ne pas être appropriée.

2. Régression Exponentielle

- **Forces :**
  - Le modèle de régression exponentielle fonctionne mieux que le modèle linéaire sur toutes les métriques.
  - Il peut être particulièrement adapté aux données présentant une croissance ou une décroissance exponentielle.
- **Faiblesses :**
  - Comme le modèle logarithmique, le modèle exponentiel peut ne pas être approprié pour tous les types de données.
  - Surtout si les données ne suivent pas une tendance exponentielle.

3. Régression Linéaire

- **Forces :**
  - La régression linéaire est le modèle le plus simple et le plus interprétable.
  - Il fonctionne de manière décente mais pas aussi bien que les autres modèles.
- **Faiblesses :**
  - Le RSS et le RMSE plus élevés, ainsi que la valeur R-carré plus faible, indiquent que le modèle linéaire ne s'ajuste pas aux données aussi bien que les autres modèles.
  - Il peut être trop simpliste pour capturer les schémas sous-jacents dans les données.

Conclusion : Les trois modèles présentent des valeurs RSS assez similaires. Mais le modèle le plus efficace est le modèle logarithmique, avec le plus petit RMSE et le plus grand R^2. Sur la base des métriques fournies, le modèle de régression logarithmique est le plus approprié pour les données, suivi par la régression exponentielle et enfin par la régression linéaire. Cependant, le choix final du modèle doit également tenir compte du contexte des données, des hypothèses sous-jacentes de chaque modèle et de l'interprétabilité des résultats.

6. Propositions d'amélioration du modèle.

6.1 Amélioration du modèle logarithmique.:

`hwy` est une autre variable qui est aussi fortement corrélée avec `cty` comme `displ`. Mais la relation correlation entre `hwy` et `displ` n'est pas vraiment forte.

Le but est d'utiliser les deux variables `hwy` et `displ` pour améliorer la précision de la prédiction de la variable `cty`.

Le modèle utilisé est maintenant :

$$y = a \cdot \log(x1 + x2) + b$$

avec $x1$ et $x2$ étant `displ` et `hwy`.

```
# Fonction pour essayer différentes valeurs initiales
finda_b <- function(data, formula, a_vals, b_vals, c_vals) {
    for (a in a_vals) {
        for (b in b_vals) {
            try({
                model <- nls(formula, data = data, start = list(a = a,
                  b = b))
```

```
                return(list(a = a, b = b))
            }, silent = TRUE)
        }
    }
    stop("pas de valeur favorable")
}

# Initialiser les valeurs initiales de a, b et c
a_vals <- seq(1, 10, by = 1)
b_vals <- seq(1, 10, by = 1)

resultat <- finda_b(Consommations, cty ~ a * log(displ + hwy) +
    b, a_vals, b_vals)
print(resultat)
```

```
## $a
## [1] 1
##
## $b
## [1] 1
```

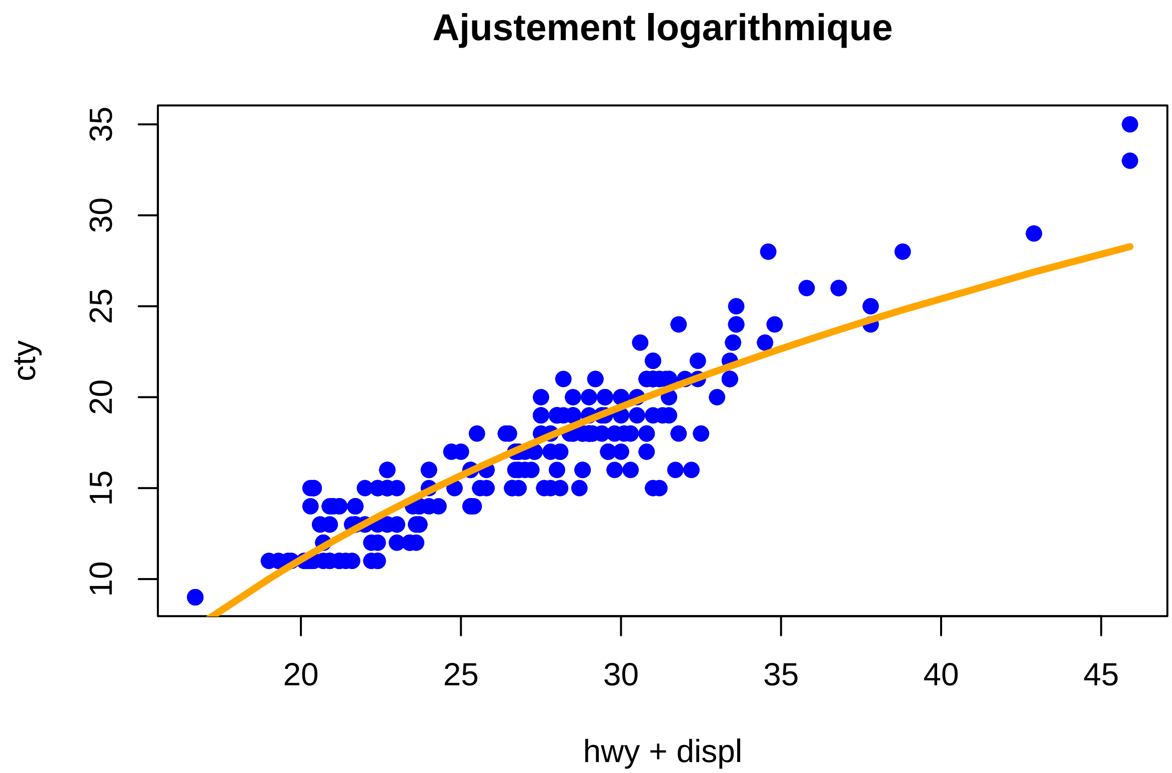
```
(RegLog <- nls(cty ~ a * log(displ + hwy) + b, data = Consommations,
    start = list(a = resultat$a, b = resultat$b)))
```

```
## Nonlinear regression model
##   model: cty ~ a * log(displ + hwy) + b
##    data: Consommations
##      a      b
##  20.72 -51.00
##  residual sum-of-squares: 730.2
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 5.275e-08
```

On obtient l'équation suivante: $y = 20.72 \cdot log(x1 + x2) - 51$. On a le RSS maintenant est 730.2 plus petit que le modele avec une seule variable independante `displ` est 1281.

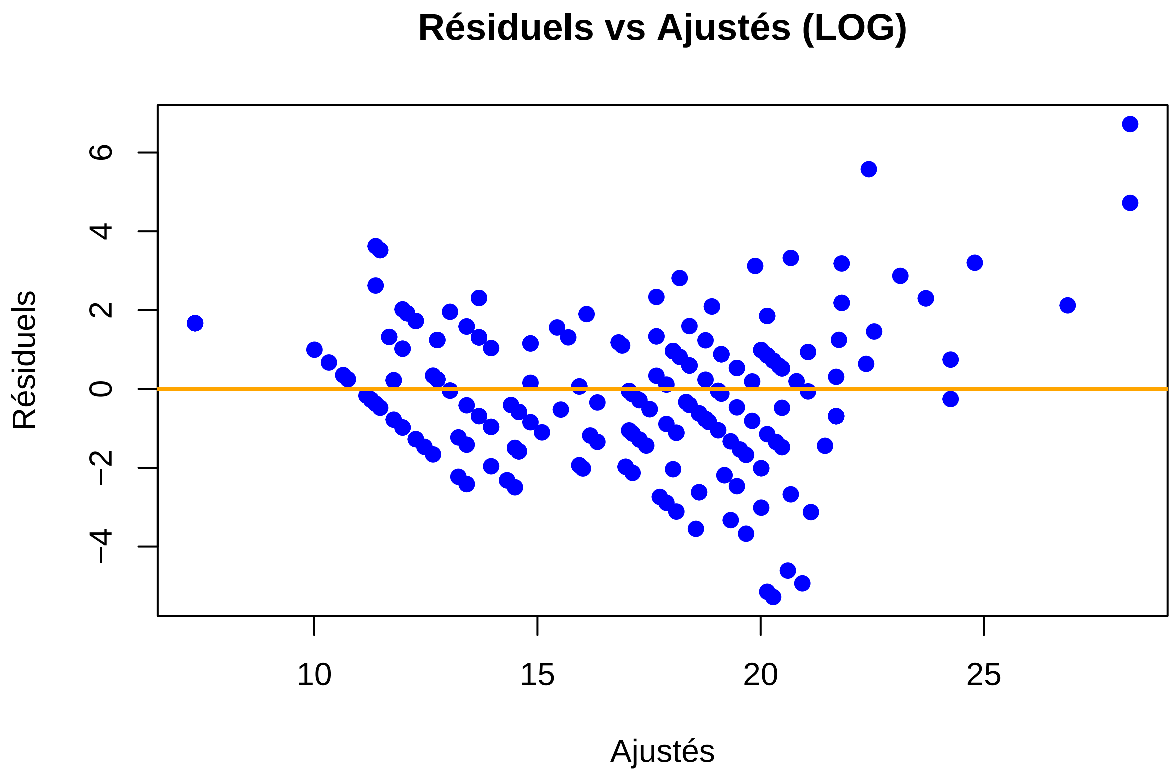
```
# sorted_index renvoie la permutation des indices des
# éléments de `displ` qui les trierait dans l'ordre
# croissant
Consommations_displ_hwy <- Consommations$displ + Consommations$hwy
sorted_index <- order(Consommations_displ_hwy)
sorted_displ_hwy <- Consommations_displ_hwy[sorted_index]
sorted_cty <- Consommations$cty[sorted_index]
sorted_pred <- predict(RegLog)[sorted_index]
```

```
plot(Consommations_displ_hwy, Consommations$cty, main = "Ajustement logarithmique",
    xlab = "hwy + displ", ylab = "cty", pch = 19, col = "blue")
# tracer un courbe du fonction logarithme
lines(sorted_displ_hwy, sorted_pred, col = "orange", lwd = 3.5)
```



(*) L'abscisse x est $hwy + displ$.

```
# residual vs fitted (modele amelioré)
plot(fitted(RegLog), resid(RegLog), xlab = "Ajustés", ylab = "Résiduels",
    main = "Résiduels vs Ajustés (LOG)", col = "blue", bg = "blue",
    pch = 21)
abline(h = 0, col = "orange", lwd = 2)
```



6.2 Amélioration du modèle linéaire.:

On ajoute une autre variable `hwy`.

`hwy` est une autre variable qui est aussi fortement corrélée avec `cty` comme `displ`. Mais la relation correlation entre `hwy` et `displ` n'est pas vraiment forte.

Le but est d'utiliser les deux variables `hwy` et `displ` pour améliorer la précision de la prédiction de la variable `cty`.

Le modèle utilisé est maintenant :

$$y = a \cdot x1 + c \cdot x2 + b$$

avec $x1$ et $x2$ étant `displ` et `hwy`.

```
displ <- Consommations$displ
hwy <- Consommations$hwy
cty <- Consommations$cty

# modele lineaire
ReglinLog <- lm(cty ~ displ + hwy, data = Consommations)
summary(ReglinLog)
```

```
##
## Call:
## lm(formula = cty ~ displ + hwy, data = Consommations)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.1426 -0.6324 -0.0081  0.6989  5.0691
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.73676    0.75117   6.306 1.44e-09 ***
## displ       -0.52834    0.09270  -5.699 3.65e-08 ***
## hwy          0.59541    0.02011  29.602  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.175 on 231 degrees of freedom
## Multiple R-squared:  0.9244, Adjusted R-squared:  0.9237
## F-statistic:  1412 on 2 and 231 DF,  p-value: < 2.2e-16
```

On obtient l'équation suivante :

$$y = -0.528 \cdot x1 + 0.595 \cdot x2 + 4.736$$

On prédit `cty` avec `displ(la cylindrée du moteur en litres) = 5` et `hwy(en miles par gallon pour la conduite sur l'autoroute) = 25`

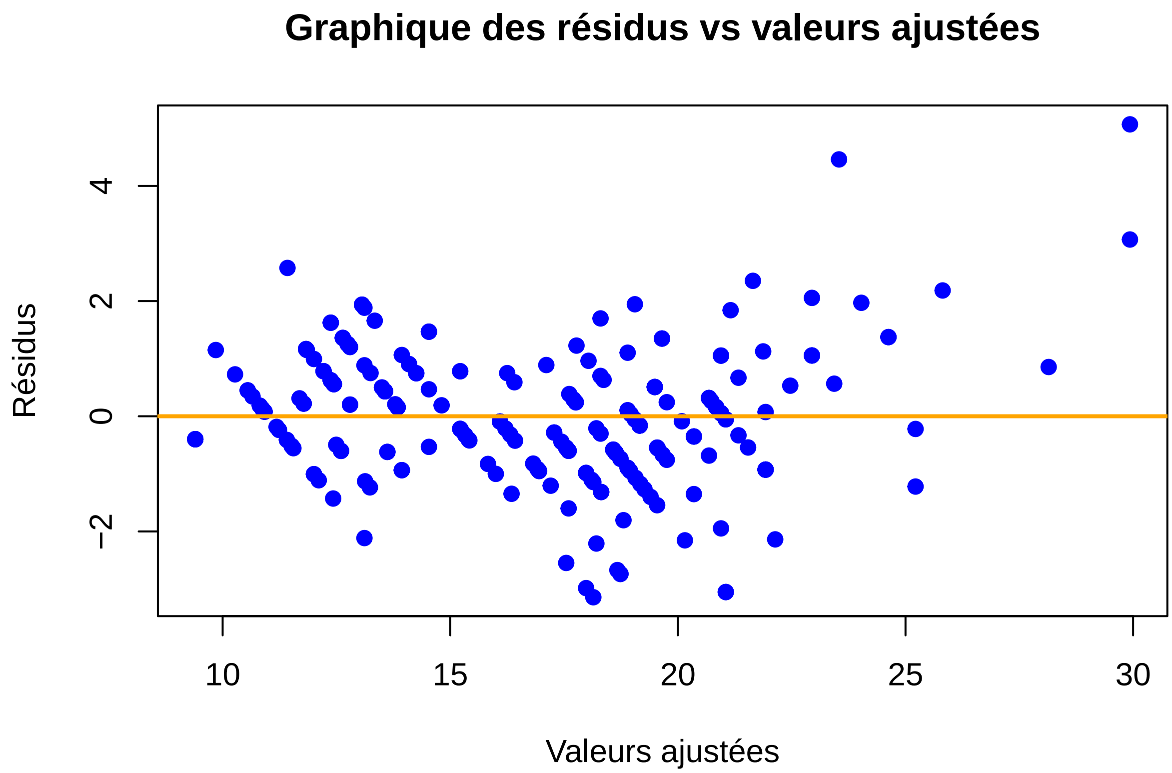
```
predict(ReglinLog, newdata = data.frame(displ = 5, hwy = 25))
```

```
##       1
## 16.9803
```

Évaluation de modèle:

```
# Calculer les valeurs ajustées et les résidus
fitted_values <- fitted(ReglinLog)
residuals <- resid(ReglinLog)

# Tracer le graphique des résidus contre les valeurs
# ajustées
plot(fitted_values, residuals, main = "Graphique des résidus vs valeurs ajustées",
    xlab = "Valeurs ajustées", ylab = "Résidus", col = "blue",
    pch = 21, bg = "blue")
abline(h = 0, col = "orange", lwd = 2)
```




```
# RSS
(lin_rssOpti <- (sum(resid(ReglinLog)^2)))
```

```
## [1] 319.0393
```

```
# R^2
(lin_r2 <- summary(ReglinLog)$r.squared)
```

```
## [1] 0.9244045
```

```
ModeleLinAvant <- lin_rss
ModeleLinApres <- lin_rssOpti
```

Tableau des RSS des Modèles Linéaires.

RSS Avant Optimisation	RSS Après Optimisation
1529.2823999	319.0392996

Conclusion: Le modèle linéaire optimisé a réduit le RSS de ~ 5 fois par rapport au modèle linéaire précédent.