TP R - Etude de modélisations autour du Problème du Problème du Voyageur de Commerve (TSP)

Florian Rascoussier

11-19/01/2022

Les objectif de ce TP sont les suivants: * Proposer une visualisation pour comparer des algorithmes, * Mettre en oeuvre des procédures de tests statistiques. * Introduire la notion de tests multiples. * Ajuster un modèle de régression linéaire. * Analyser un problème de régression (validité de l'ajustement et des hypothèses). * Réaliser une sélection de variable (méthode AIC).

0. Visualisation de chemins

Le but de cette section est de vérifier que votre installation est correcte, et de visualiser un problème du voyageur de commerce.

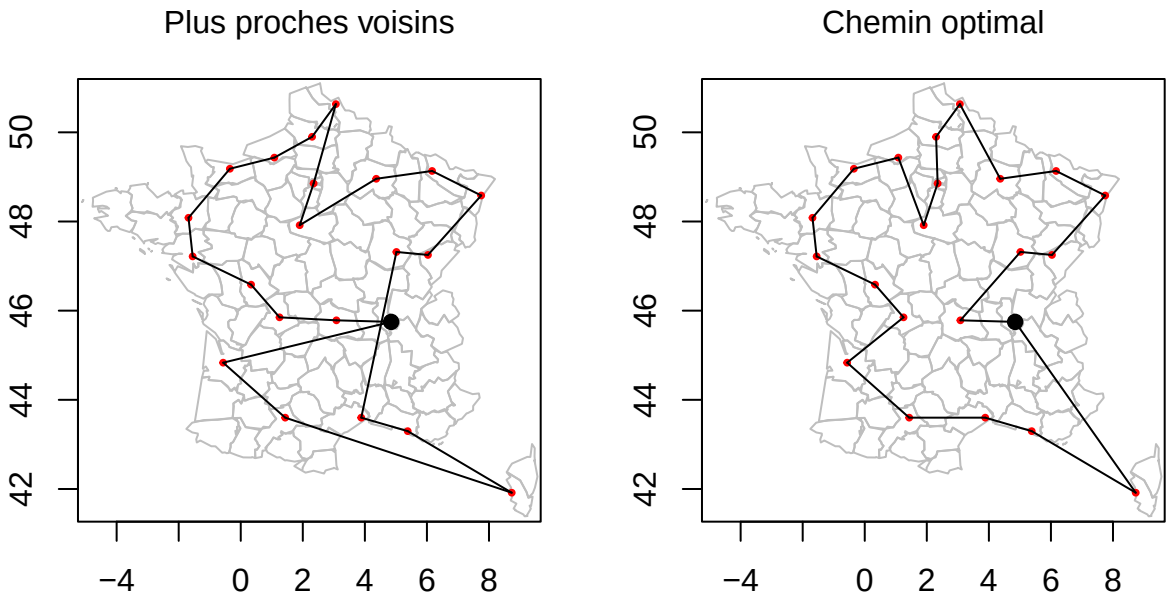
Lecture du fichier des villes :

```
# DonneesGPSvilles.csv contient les coordonnées GPS de 22 villes françaises
villes <- read.csv(
  '/home/onyr/Documents/code/R/4if_s1_stats_tp_tsp/data/DonneesGPSvilles.csv',
  header=TRUE,dec='.',sep=';',quote="\"")
str(villes)
```

```
## 'data.frame':    22 obs. of  5 variables:
##  $ EU_circo : chr  "Sud-Est" "Sud-Est" "Nord-Ouest" "Est" ...
##  $ region   : chr  "Rhône-Alpes" "Corse" "Picardie" "Franche-Comté" ...
##  $ ville    : chr  "Lyon" "Ajaccio" "Amiens" "Besançon" ...
##  $ latitude : num  45.7 41.9 49.9 47.2 44.8 ...
##  $ longitude: num  4.847 8.733 2.3 6.033 -0.567 ...
```

Représentation des chemins par plus proches voisins et du chemin optimal :

```
coord <- cbind(villes$longitude,villes$latitude)
dist <- distanceGPS(coord)
voisins <- TSPnearest(dist)

pathOpt <- c(1,8,9,4,21,13,7,10,3,17,16,20,6,19,15,18,11,5,22,14,12,2)

par(mfrow=c(1,2),mar=c(1,1,2,1))
plotTrace(coord[voisins$chemin,], title='Plus proches voisins')
plotTrace(coord[pathOpt,], title='Chemin optimal')
```



Les longueurs des trajets (à vol d'oiseau) valent respectivement, pour la *méthode des plus proches voisins* :

```
## [1] 4303.568
```

```
## [1] 4303.568
```

et pour la *méthode optimale* :

```
## [1] 3793.06
```

```
## [1] 3793.06
```

Ceci illustre bien l'intérêt d'un algorithme de voyageur de commerce. Nous allons dans la suite étudier les performances de cet algorithme.

1. Comparaison d'algorithmes

Comparaison de plusieurs algorithmes, donnant des solutions exactes et approchées.

Nombre de sommets fixes et graphes "identiques".

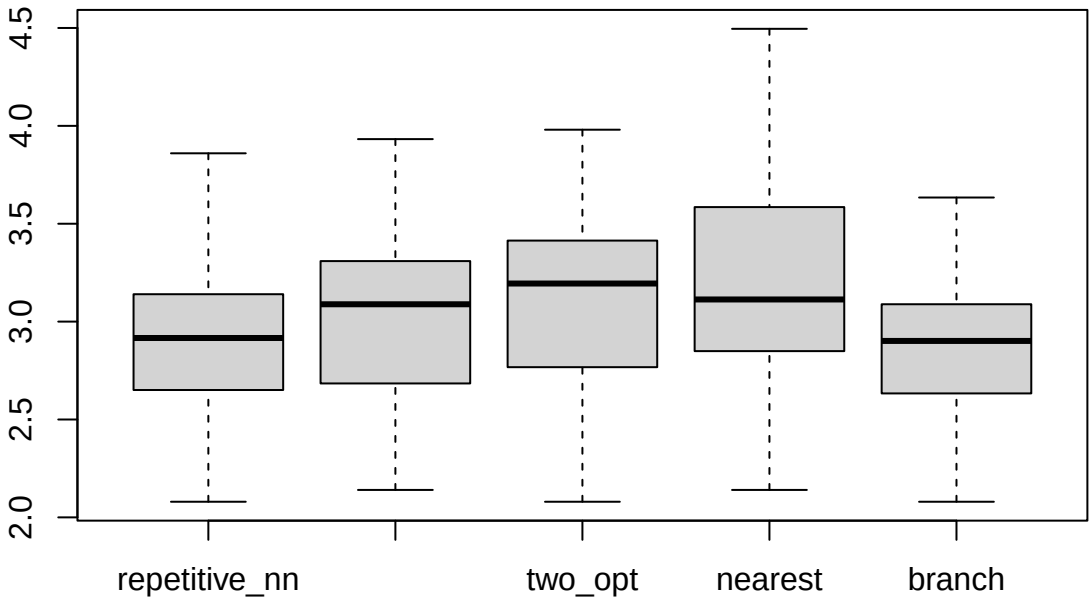
```
# generate a new graph couts
      n <- 10
sommets <- data.frame(
  x = runif(n),
  y = runif(n) # runif - get n points distributed along uniform law
)
  couts <- distance(sommets)
```

1.1. Longueur des chemins

Comparaison des longueurs de différentes méthodes :

- boxplots

```
compare_methods <- matrix(0,50,5)
method_names <- c("repetitive_nn", "nearest_insertion", "two_opt", "nearest", "branch")
colnames(compare_methods) <- method_names

# create 50 seeds for the 50 repetitions
seeds <- c(seq(from=0,by=5,length=50))
methods_used_for_each_value <- matrix("",50,5)
colnames(methods_used_for_each_value) <- method_names

# compute 50 times the 5 methods
for(i in 1:50) {
  # set current seed
  set.seed(seeds[i])

  # generate a new graph
  n <- 10
  sommets <- data.frame(x = runif(n), y = runif(n))
  couts <- distance(sommets) # the new graph

  # compute
  for(j in 1:(length(method_names))) {
    compare_methods[i, j] <- TSPsolve(couts, method_names[j])
    methods_used_for_each_value[i,j] <- method_names[j]
  }
}

boxplot(compare_methods)
```



Les boxplots nous indiquent que “branch” est le meilleur (on le savait), car les durées de transport sont les plus courtes. On peut se demander si “repetitive_nn” et “branch” sont similaires ou non.

- test entre ‘nearest’ et ‘branch’

```
# H1 avec > -> alternative = greater
pairwise.t.test(
  cbind(compare_methods[,"nearest"],compare_methods[,"branch"]),
  cbind(methods_used_for_each_value[,"nearest"],
  methods_used_for_each_value[,"branch"]),
  alternative = "greater",
  p.adjust.method = "bonferroni",
  paired = TRUE
)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  cbind(compare_methods[, "nearest"], compare_methods[, "branch"]) and cbind(methods_used_for_ea
##
##         branch
## nearest 5.7e-11
##
## P value adjustment method: bonferroni
```

- tests 2 à 2

```
# H1 avec > -> alternative = greater
pairwise.t.test(
  compare_methods,
  methods_used_for_each_value,
  alternative = "greater",
  p.adjust.method = "bonferroni",
  paired = TRUE
)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  compare_methods and methods_used_for_each_value
##
##                   branch  nearest nearest_insertion repetitive_nn
## nearest           5.7e-10 -       -                 -
## nearest_insertion 4.1e-09 1.00    -                 -
## repetitive_nn     3.8e-05 1.00    1.00              -
## two_opt           2.0e-09 1.00    0.37              2.4e-05
##
## P value adjustment method: bonferroni
```

Quand la p-valeur est petite, on peut conclure que (H0) est fausse donc que (H1) est vrai. Ici, rappel :

- (H0) : moyenne method 1 == moyenne methode 2

- (H1) : moyenne method 1 != moyenne methode 2

Donc ici, on peut déduire que “branch” a une moyenne différente des autres, de même entre “two_opt” et “repetitive_nn”. Pour le reste, on ne peut rien dire.

Cela invalide notre hypothèse que “branch” et “repetititve_nn” étaient similaires, puisqu'ils n'ont pas la même moyenne.

1.2. Temps de calcul

Comparaison des temps à l'aide du package microbenchmark.

Exemple d'application de microbenchmark :

```
microbenchmark(
  TSPsolve(couts, "repetitive_nn"),
  TSPsolve(couts, "nearest_insertion"),
  TSPsolve(couts, "two_opt"),
  TSPsolve(couts, "nearest"),
  TSPsolve(couts, "branch"),
  times=20,
  setup={
    n <- 10
    sommets <- data.frame(x = runif(n), y = runif(n)) # runif - get n points distributed along uniform
    couts <- distance(sommets)
  }
)
```

```
## Unit: microseconds
##                                   expr      min        lq      mean   median
##      TSPsolve(couts, "repetitive_nn") 4802.209 5129.0370 6260.4199 6009.619
##  TSPsolve(couts, "nearest_insertion")  834.600  910.0460 1363.5604 1092.704
##             TSPsolve(couts, "two_opt")  447.220  694.8620  987.0122  948.843
##             TSPsolve(couts, "nearest")   11.150   14.4475   20.5628   18.358
##              TSPsolve(couts, "branch") 1058.736 2480.4085 3662.9781 3567.229
##        uq      max neval  cld
##  7422.869 8414.931    20    d
##  1518.751 3947.135    20  b
##  1162.640 2106.829    20  b
##    22.404   38.401    20 a
##  4330.829 9002.563    20   c
```

On déduit que “nearest” est l'algorithme le plus rapide.

2. Etude de la complexité de l'algorithme Branch and Bound

2.1. Comportement par rapport au nombre de sommets : premier modèle

Récupération du temps sur 20 graphes pour différentes valeurs de n.

Ajustement du modèle linéaire de $\log(temps)^4$ en fonction de n.

Analyse de la validité du modèle :

- pertinence des coefficients et du modèle,
- étude des hypothèses sur les résidus.

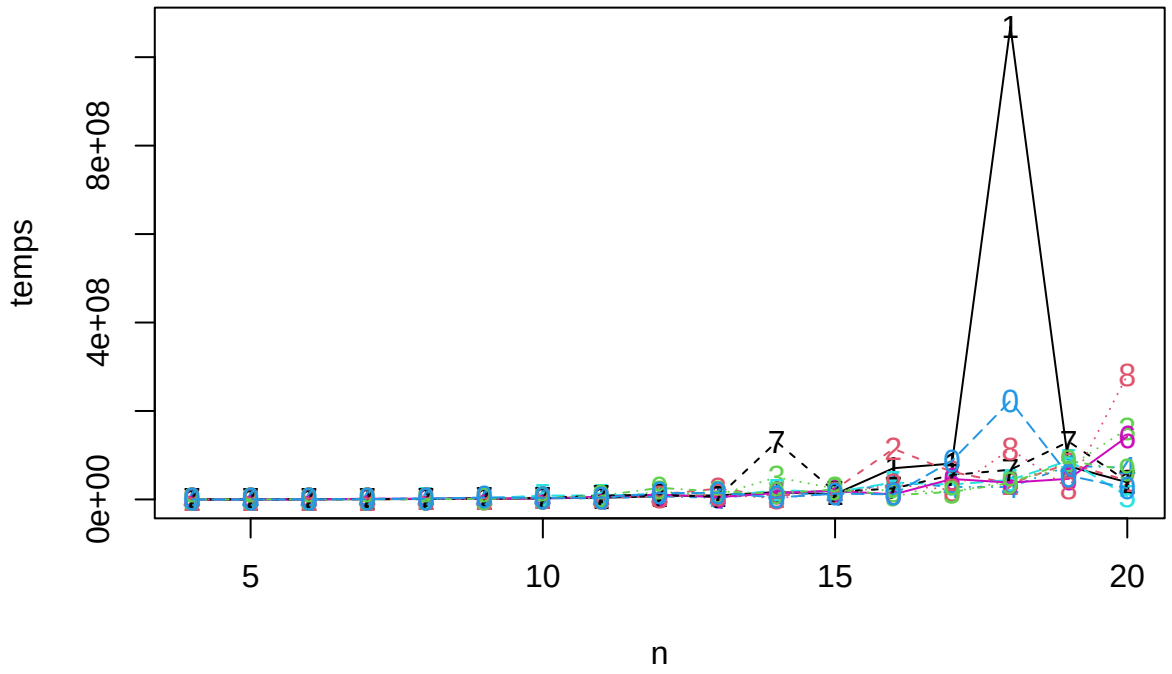
```
nb_of_times <- 10
seqn <- c(seq(4, 20, 1))

compute_time_per_n <- function(n) {
  return(
      microbenchmark(
      TSPsolve(couts, method = "branch"),
      times = nb_of_times,
      setup = {
        couts <- distance(cbind(x = runif(n), y = runif(n)))
      }
    )$time
  )
}
temps <- t( # t() transpose
  apply(X = as.array(seqn), MARGIN = 1, FUN = compute_time_per_n)
)
```

Représentation graphique de *temps* en fonction de n puis de sa régression linéaire $log(temps)^2$ en fonction de n :

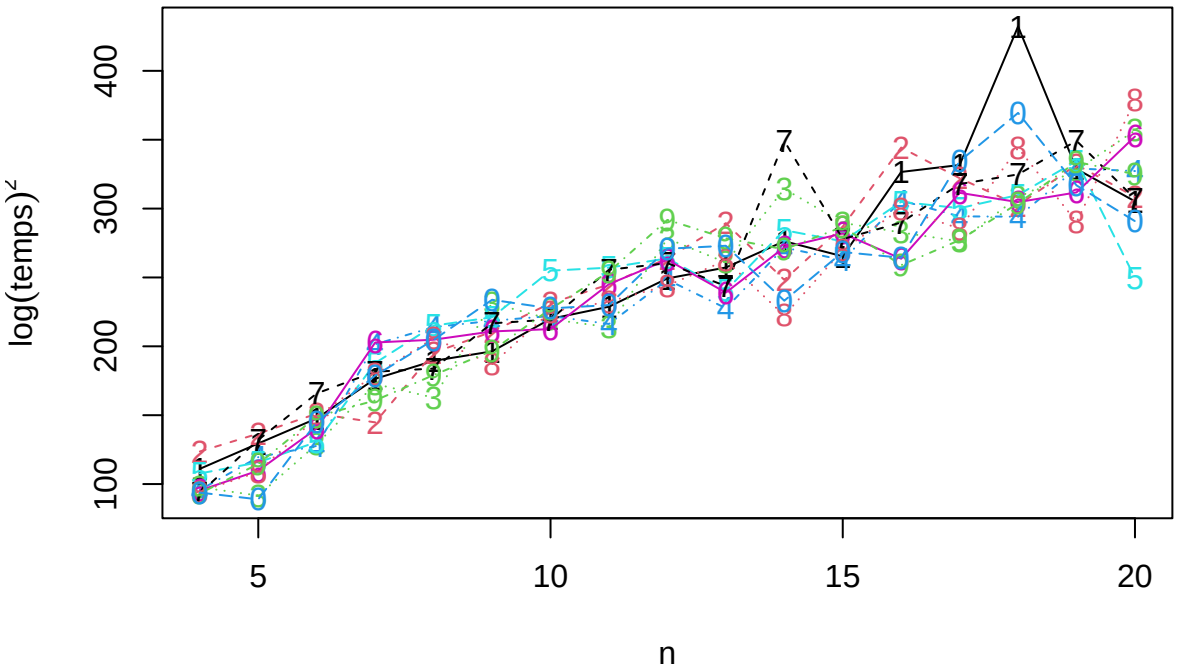
```
par(mflow=c(1,2)) # 2 graphiques en 1 ligne
```

```
## Warning in par(mflow = c(1, 2)): "mflow" is not a graphical parameter
```

```
matplot(seqn, temps, xlab="n", ylab="temps", type = "o")
```



```r
matplot(seqn, log(temps)^2, xlab="n", ylab=expression(log(temps)^2), type = "o")
```



Ajustement du modèle lineaire de $log(temps)^2$ en fonction de n puis récupération des principales statistiques :

```r
# ajustement du modèle lineaire de log(temps)^2
vect_temps <- log(as.vector(temps))^2
vect_dim <- rep(seqn, times=10)
temps.lm <- lm(vect_temps~vect_dim) # calcul des résidus
summary(temps.lm)
```

```
##
## Call:
## lm(formula = vect_temps ~ vect_dim)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -100.186  -18.003    1.778   16.034  110.187
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  71.3910     5.3830   13.26   <2e-16 ***
## vect_dim     13.9254     0.4153   33.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.53 on 168 degrees of freedom
## Multiple R-squared:   0.87,  Adjusted R-squared:  0.8692
## F-statistic:  1124 on 1 and 168 DF,  p-value: < 2.2e-16
```

On obtient un résumé des résultats du calcul des résidus.

- *Intercept* = constante
- *Estimate* = valeurs des coefficients correspondant à chaque variable
- *Pr(>|t|)* = p-valeur du test (H0) coefficient=0 contre (H1) coefficient 6 = 0. 't' car c'est un test de Student. Les * dans la marge servent à repérer les valeurs significatives
- *Residual standard error* = le sigma châpeau du cours
- *Multiple R-squared* = le R^2 du cours en dimension 2, En plus grande dimension, R^2 est le ratio entre la variance expliquée par le modèle et la variance des données : si le ratio est proche de 1, cela signifie que les observations s'éloignent peu du modèle.
- *F-statistic* = test de Fisher de pertinence du modèle. Un modèle pertinent est un modèle tel que R^2 est significativement supérieur à 0. F = $((R^{2)/(1-(R}2)))$*((N-K)/(K-1)) avec K le nombre de variables dans le modèle (hors constante). La p-valeur donne la probabilité de se trompe si on affirme que le modèle n'est pas pertinent.

S'il n'y a qu'une seule variable dans le modèle, tester (H0) coefficient=0 contre (H1) coefficient 6 = 0 ou faire le test de pertinence global de Fisher sont parfaitement équivalents.

plus la p-valeur (indiquée par `Pr(>|t|)`) est faible, plus le coef sert à quelque chose pour l'influence sur la "fitness" du modèle.

Ici, p-value: < 2.2e-16 la p-valeur du modèle. Donc ici un seul paramètre suffit largement.

Il faut vérifier les hypothèses sur les résidus. Il y en a 4 : * loi normale * espérance nulle * variance constante * indépendance

Si l'une de ces hypothèse est remise en cause, alors le modèle n'est plus valable (aucun des tests ci-dessus n'est valable et l'ajustement pas les moindres carrés est également discutable).

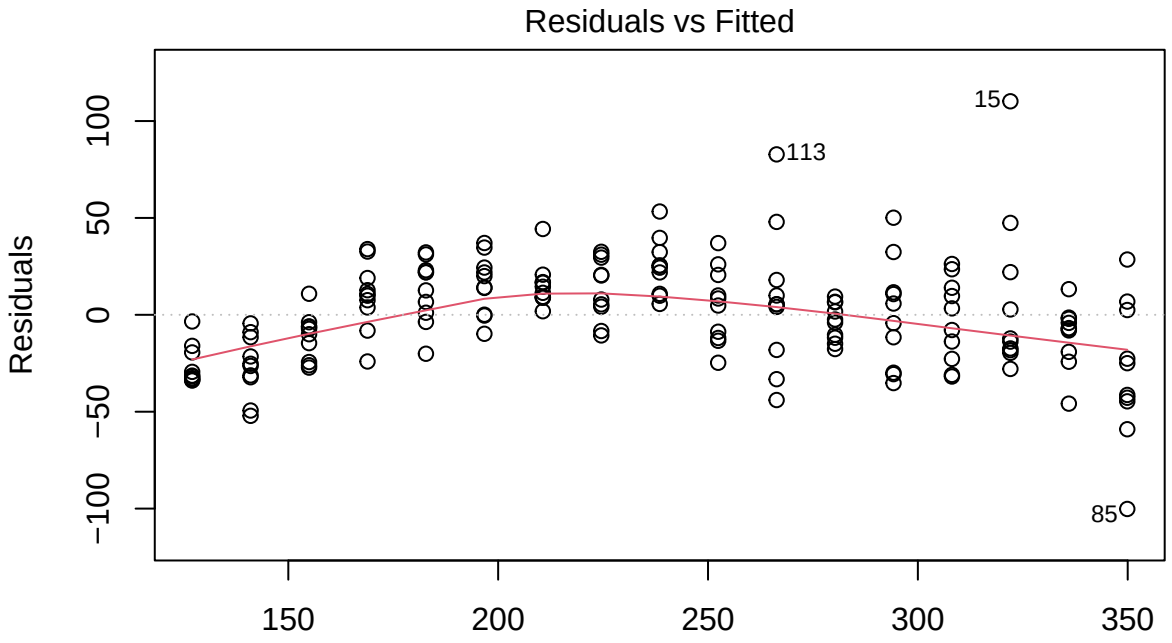
L'étude de ces hypothèses se fait par * une étude graphique * des tests

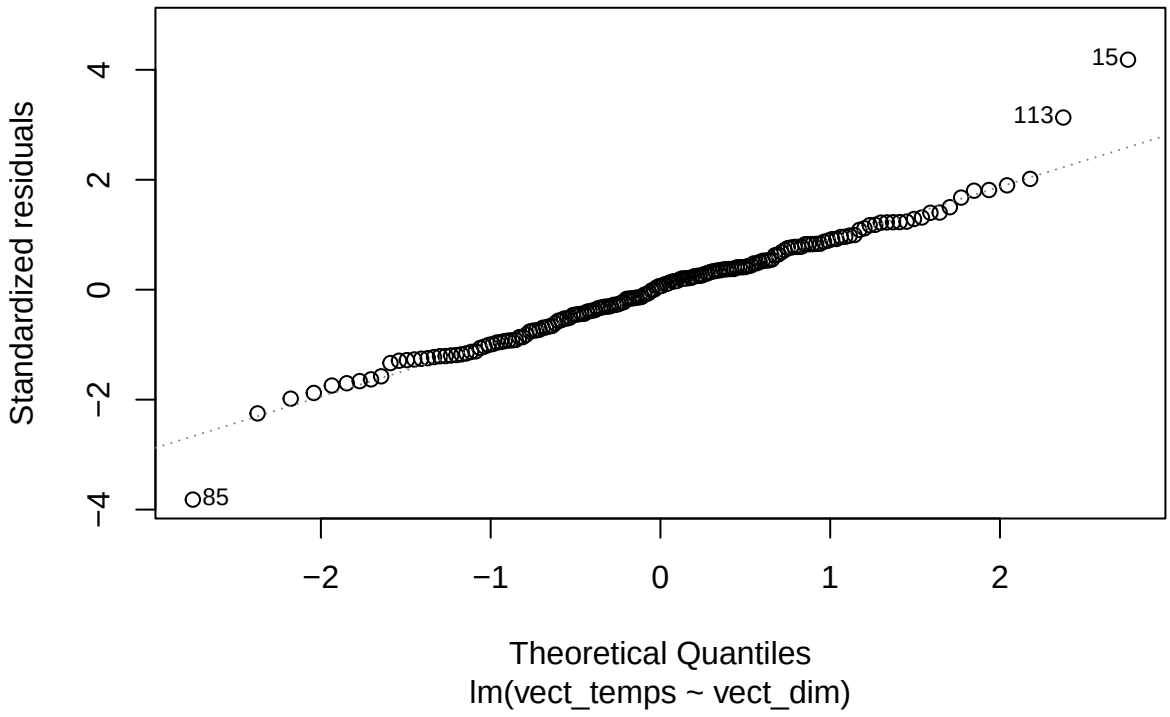
Etude graphique :

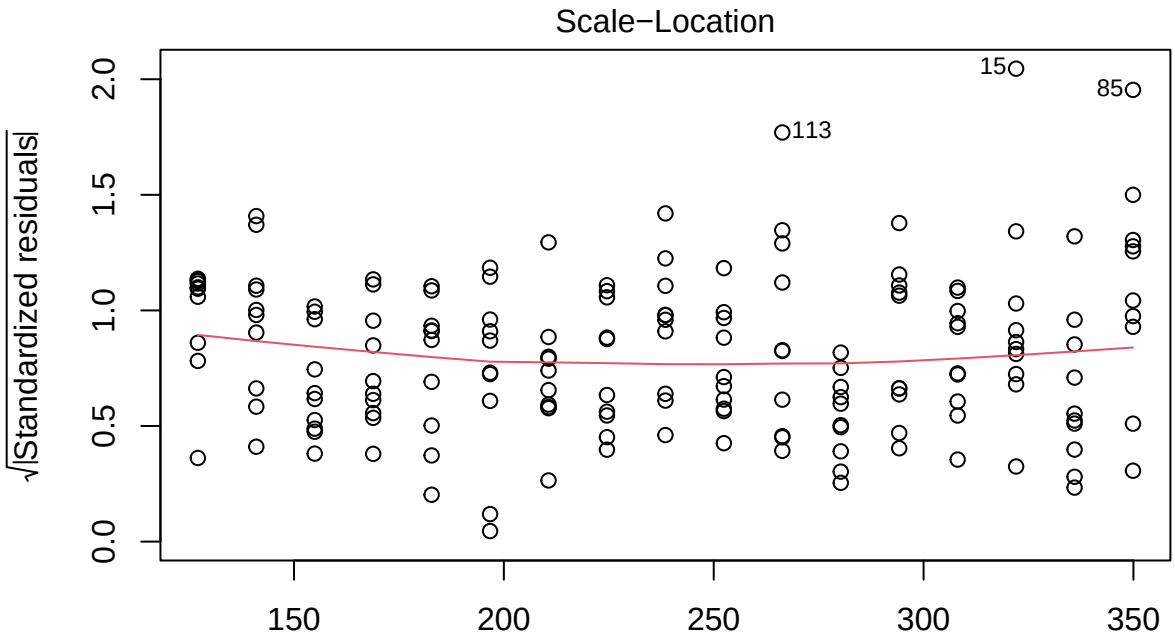
```
par(mflow=c(2,2)) # 4 graphiques
```

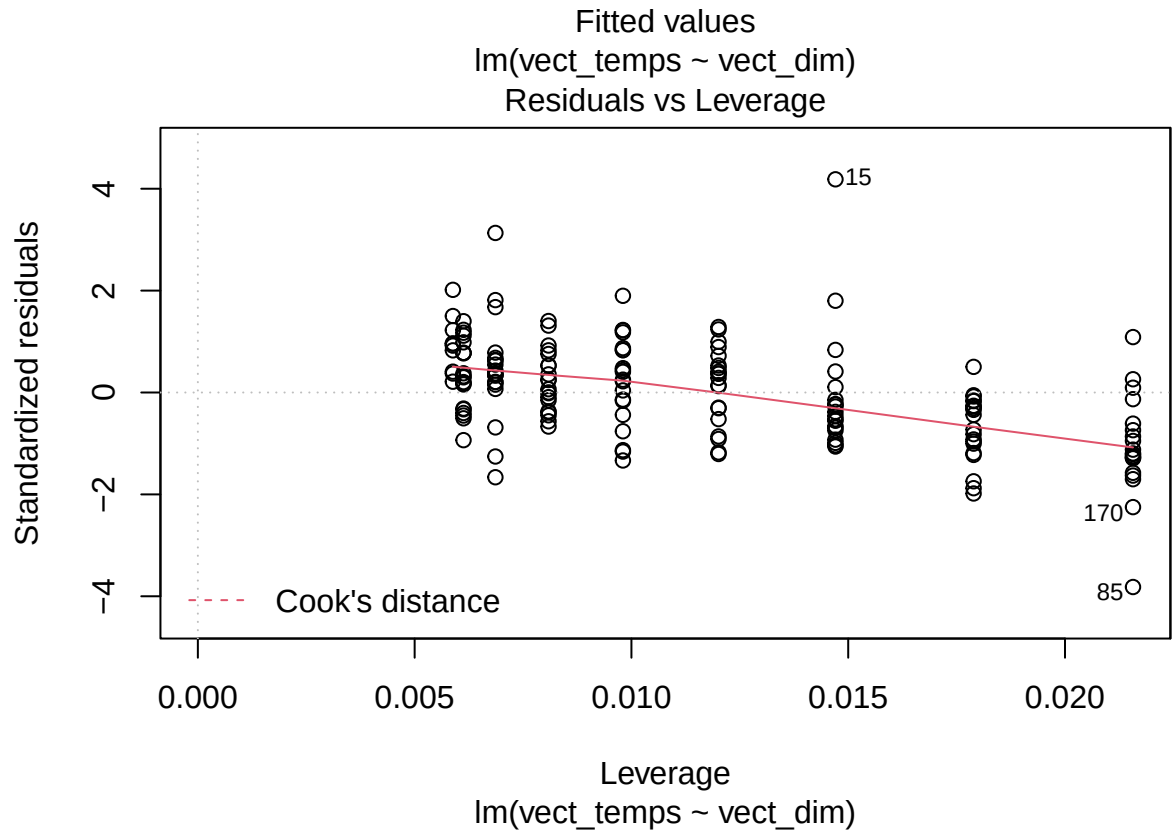
```
## Warning in par(mflow = c(2, 2)): "mflow" is not a graphical parameter
```

```
plot(temps.lm)
```


Residuals vs Fitted
Residuals
Fitted values
lm(vect_temps ~ vect_dim)
15
113
85
100
50
0
−50
−100
150
200
250
300
350


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(vect_temps ~ vect_dim)
15
113
85
4
2
0
−2
−4
−2
−1
0
1
2





comment interpréter les résultats des graphiques ?

- **Residuals vs Fitted** : Si on observe une tendance trop marquée des points sur le graphique, cela signifie que l'espérance des résidus n'est pas nulle, mais qu'elle est positive sur certaines sections et négatives sur d'autres. Ce problème peut souvent être corrigé avec un changement de variable. On reste assez "tolérant" sur les tendances et il faut qu'elles soient marquées pour rejeter le modèle.

– Si on observe que le nuage de point s'écarte (forme de trompette) la variance des résidus n'est pas constante. On dit que les résidus sont hétéroscédastiques.

Ici, même si les points ne semble pas former une droite mais plutôt une parabole, on peut quand même se dire que les observations graphiques ne permettent pas de rejeter le modèle. C'est encore suffisament plat. L'espérance des résidus est nulle.

- **Normal Q-Q** : Compare la distribution des résidus à une loi normale. En abscisse, les quantiles empiriques des résidus et en ordonnée les quantiles de la loi normale, avec estimation des paramètres sur les résidus. Si les distribution sont identiques ou presque alors l'ensemble des points sont sur la diagonale. Sinon on observera la plupart du temps des deviation aux extremité ce qui sous-entend que les queues de distribution sont différentes.

Ici, les points suivent relativement bien la droite donc pareil, c'est bien aussi.

- **Scale location** : Idem que Residuals vs Fitted mais avec des résidus normalisés.

Ici la droite est suffisament plate. Les résidus sont possiblement nuls.

- **Residuals vs Leverage** : Montre l'influence des echantillons (plus un point est à droite et plus il en a). Si un point est un outliers il apparaitra trés éloigné des autres et en dehors des bornes par rapport à la distance de Cook. Ces bornes sont représentées par des lignes rouge en pointillé. Il faut reprendre le modèle en enlevant les points concernés s'il y en a pour vérifier qu'ils ne déterminent pas le modèle à eux tout seuls.

Ici les points sont tous relativement proches de la courbe rouge, et surtout ne dépassent aucune distance de Cook (en pointillé). On peut donc être satisfait de l'influence des échantillons.

On peut faire un test du qui-2 ou de Shapiro pour vérifier les hypothèses suivantes :

- (H0) Les résidus suivent une loi normale
- (H1) Les résidus ne suivent pas une loi normale

```
shapiro.test(residuals(temps.lm))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(temps.lm)
## W = 0.97562, p-value = 0.004335
```

Rappel : Dans chaque cas les tests doivent être appliqués sur les résidus du modèle, et une p-valeur petite signifie un rejet de l'hypothèse, donc du modèle de régression.

Ici la p-valeur est relativement grande, donc on peut peut rien conclure, autrement dit, on ne peut pas réfuter (H0) et valider (H1), c'est-à-dire qu'on ne peut pas dire avec ce test que les données ne suivent pas une loi normale. ATTENTION : On ne peut cependant pas conclure pour autant que les données suivent bien une loi normale !

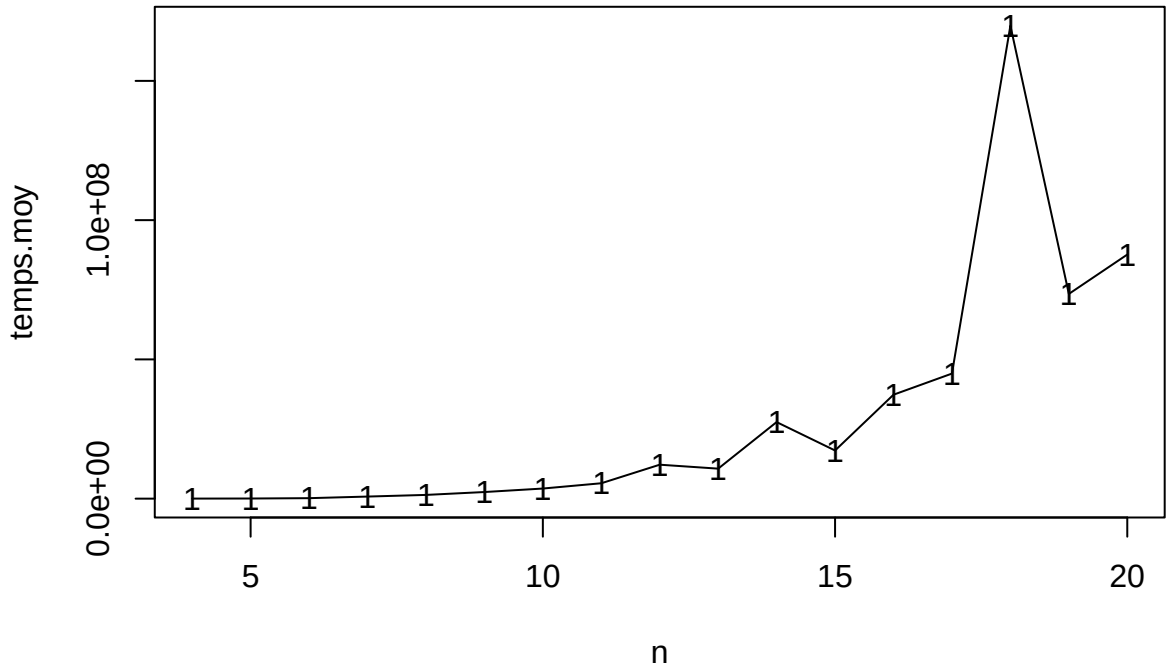
2.2. Comportement par rapport au nombre de sommets : étude du comportement moyen

Récupération du temps moyen : ajustement du modèle linéaire de $\log(temps.moy)^2$ en fonction de n.

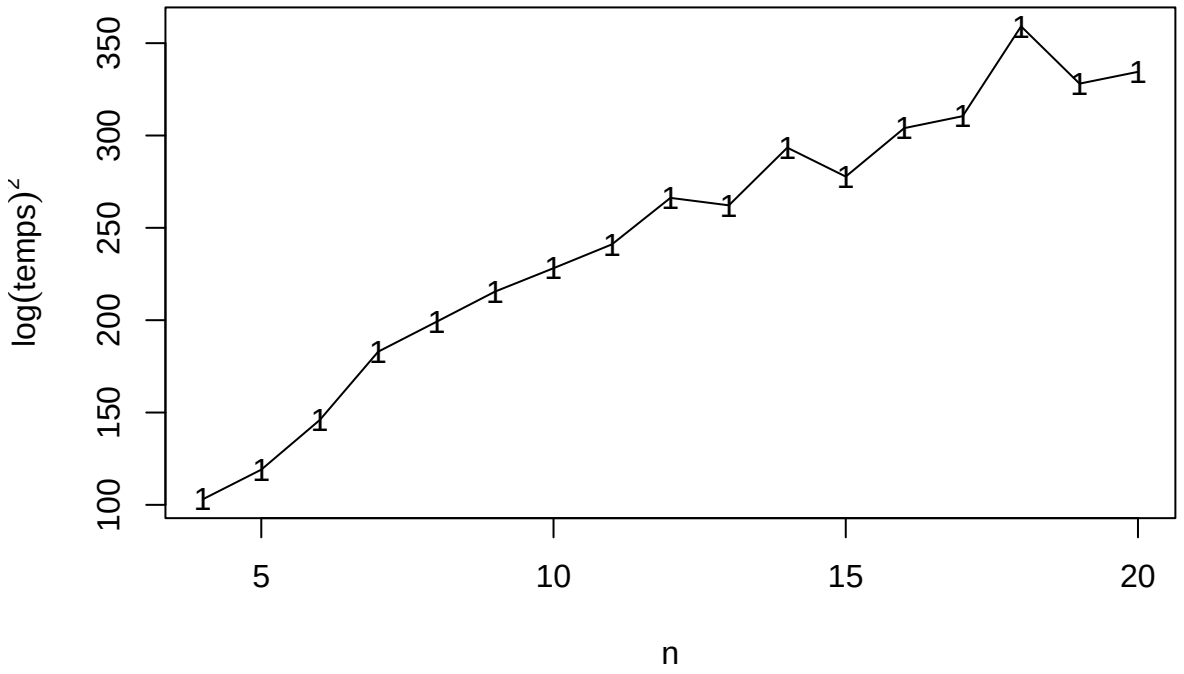
```r
temps.moy <- rowMeans(temps) # on remplace nos lignes de 10 valeurs par la moyenne des ses valeurs

# affichage graphique de temps.moy et de sa linéarisation
par(mflow=c(1,2)) # 2 graphiques en 1 ligne
```

```
## Warning in par(mflow = c(1, 2)): "mflow" is not a graphical parameter
```

```r
matplot(seqn, temps.moy, xlab="n", ylab="temps.moy", type = "o")
```




```r
matplot(seqn, log(temps.moy)^2, xlab="n", ylab=expression(log(temps)^2), type = "o")
```



Graphiquement, la régression linéaire à l'air correcte.

Ajustement du modèle de régression linéaire simple gaussien de $log(temps.moy)^2$, puis analyse de la validité du modèle :

- pertinence des coefficients et du modèle

```
# ajustement du modèle de régression linéaire simple gaussien de log(temps.moy)^2
vect_temps_moy <- log(as.vector(temps.moy))^2
# on a déjà vect_dim
temps.moy.lm <- lm(vect_temps_moy~seqn) # calcul des résidus
summary(temps.moy.lm) # afficher un résumé du calcul des résidus
```

```
##
## Call:
## lm(formula = vect_temps_moy ~ seqn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.282 -11.604   2.186  12.550  25.709
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  69.1785    11.5884    5.97 2.57e-05 ***
## seqn         14.6791     0.8941   16.42 5.39e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.06 on 15 degrees of freedom
## Multiple R-squared:  0.9473, Adjusted R-squared:  0.9438
## F-statistic: 269.6 on 1 and 15 DF,  p-value: 5.387e-11
```

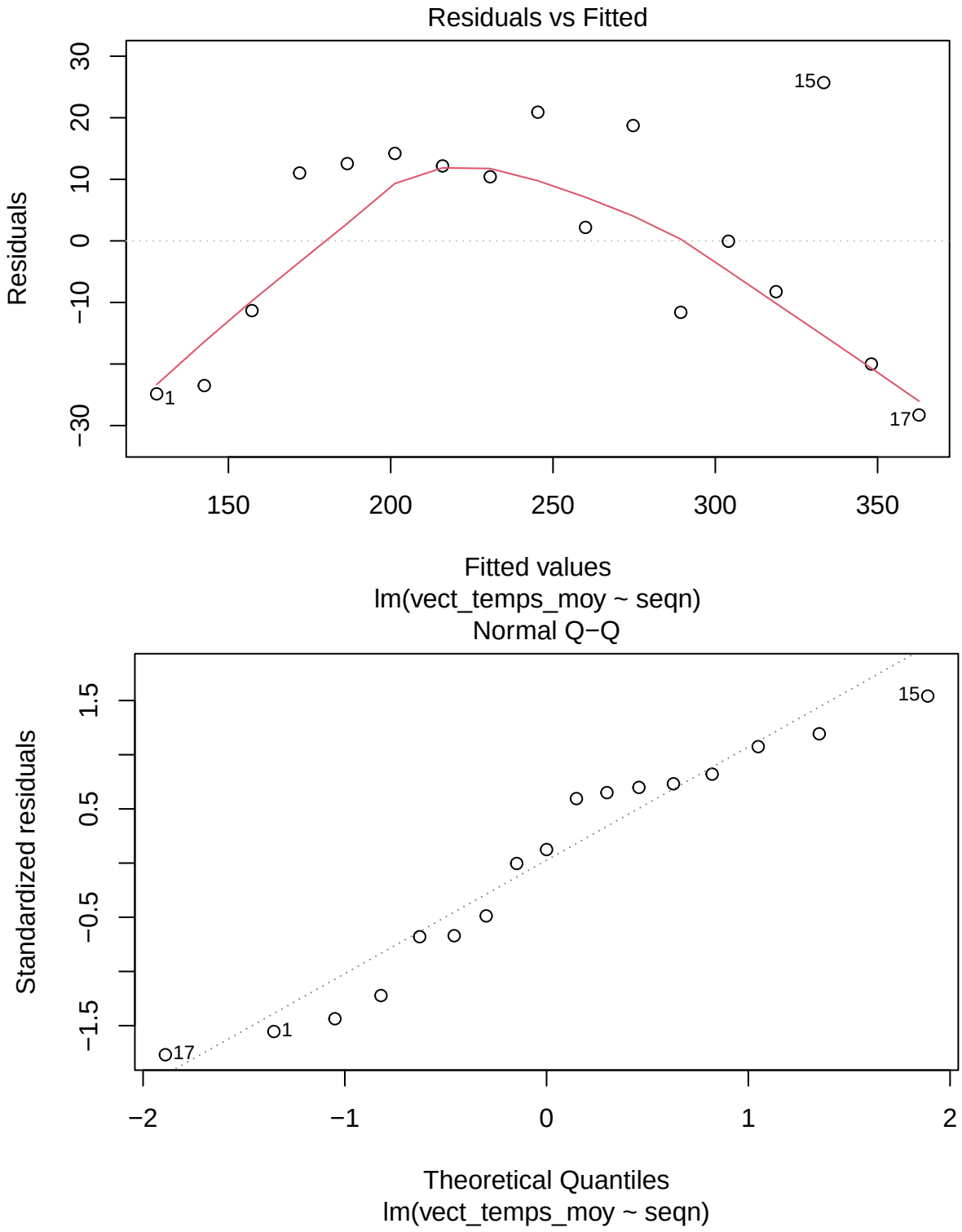
La p-valeur est très petite donc les coefs sont pertinents.

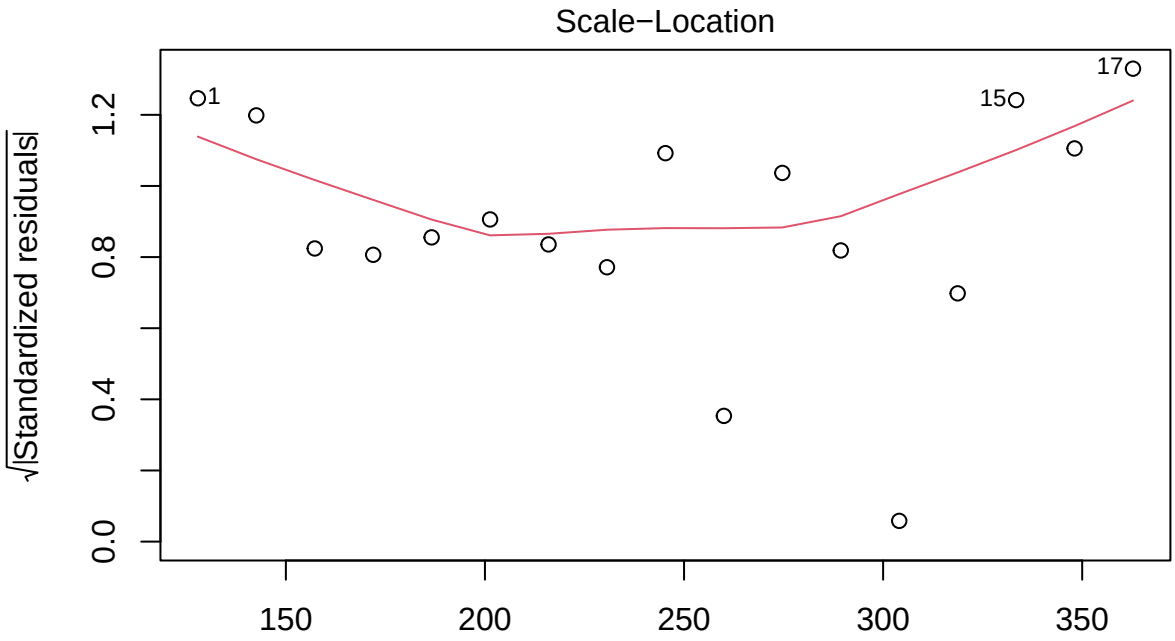
- étude des hypothèses sur les résidus.

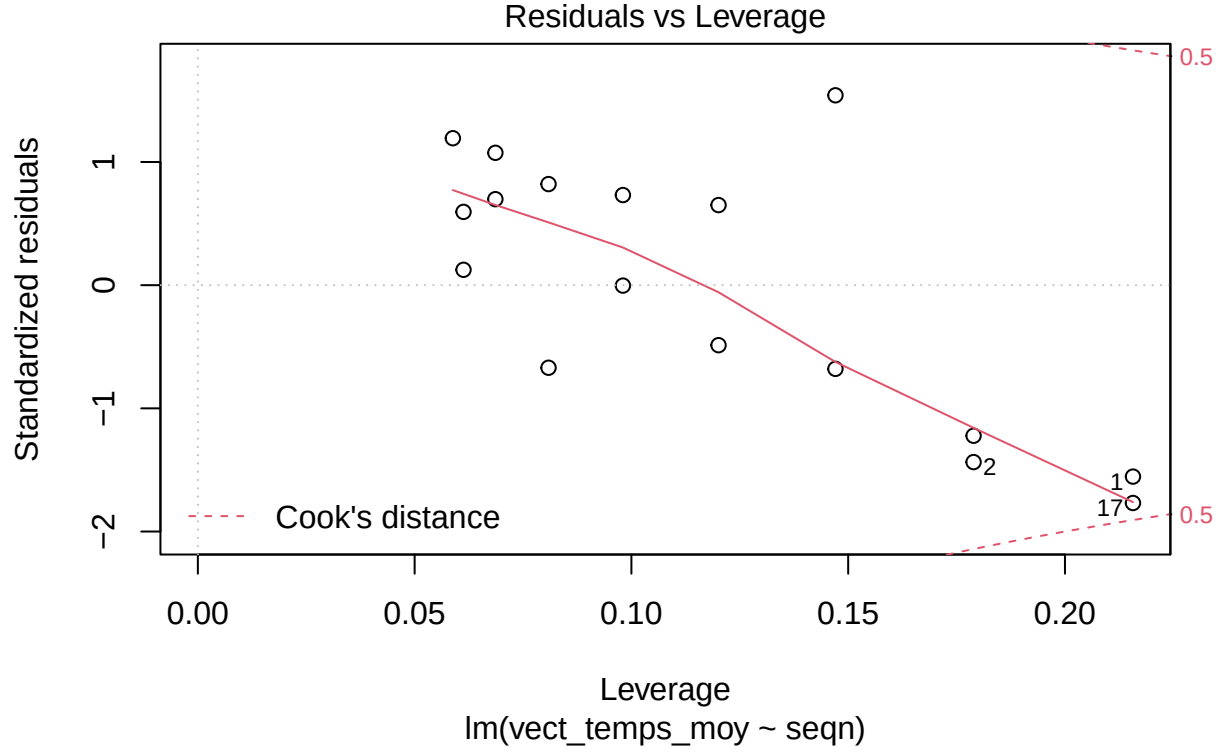
```
par(mflow=c(2,2)) # 4 graphiques
```

```
## Warning in par(mflow = c(2, 2)): "mflow" is not a graphical parameter
```

```
plot(temps.moy.lm)
```


Residuals vs Fitted
Residuals
30
20
10
0
−10
−30
15
1
17
150
200
250
300
350
Fitted values
lm(vect_temps_moy ~ seqn)
Normal Q−Q
Standardized residuals
1.5
0.5
−0.5
−1.5
15
1
17
−2
−1
0
1
2
Theoretical Quantiles
lm(vect_temps_moy ~ seqn)





On voit que Residuals vs Fitted et Scale-location sont pas vraiment plats donc c'est l'hypothèse linéaire n'est pas géniale. Il faurait faire une meilleur changement de variables.

Le résumé du calcul des résidus

2.3. Comportement par rapport à la structure du graphe

Lecture du fichier 'DonneesTSP.csv'. Ce fichier au format CSV contient une première ligne avec le nom des colonnes du dataframe, ainsi que les lignes des valeurs de ce dataframe. Attention à bien spécifier le séparateur !

```
# DonneesGPSvilles.csv contient les coordonnées GPS de 22 villes françaises
data.graph <- read.csv('/home/onyr/Documents/code/R/4if_s1_stats_tp_tsp/data/DonneesTSP.csv',header=TRU
class(data.graph)
```

```
## [1] "data.frame"
```

```
str(data.graph)
```

```
## 'data.frame':    70 obs. of  8 variables:
##  $ tps      : num  53692 144081 997803 2553322 6333009 ...
##  $ dim      : int  4 6 8 10 12 14 16 18 20 4 ...
##  $ mean.long: num  0.391 0.442 0.334 0.276 0.254 ...
##  $ mean.dist: num  0.665 0.592 0.537 0.506 0.502 ...
##  $ sd.dist  : num  0.276 0.259 0.246 0.238 0.227 ...
##  $ mean.deg : num  3 5 7 9 11 13 15 17 19 3 ...
##  $ sd.deg   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ diameter : num  1 1 1 1 1 1 1 1 1 1 ...
```

On veut construire le modèle de régression linéaire de `log(tps)` par rapport à `sqrt(dim)` et à toutes les autres variables de `data.graph`. Il faut donc créer un dataframe avec sans `log(tps)` et avec une colonne `sqrt(dim)`.

```
# extract and modify columns from data.graph dataframe
sqrt_dim <- apply(X = as.array(data.graph$dim), MARGIN = 1, FUN = sqrt) # build vector of sqrt(data.gra
log_tps <- apply(X = as.array(data.graph$tps), MARGIN = 1, FUN = sqrt) # build vector of log(data.graph

# create modified dataframe
data.graph_modified <- data.frame(sqrt_dim, data.graph$mean.long, data.graph$mean.dist, data.graph$sd.di
data_graph_modified_row_names <- c("sqrt(dim)", "mean.long", "mean.dist", "sd.dist", "mean.deg", "sd.deg
colnames(data.graph_modified) <- data_graph_modified_row_names

str(data.graph_modified)
```

```
## 'data.frame':    70 obs. of  7 variables:
##  $ sqrt(dim): num  2 2.45 2.83 3.16 3.46 ...
##  $ mean.long: num  0.391 0.442 0.334 0.276 0.254 ...
##  $ mean.dist: num  0.665 0.592 0.537 0.506 0.502 ...
##  $ sd.dist  : num  0.276 0.259 0.246 0.238 0.227 ...
##  $ mean.deg : num  3 5 7 9 11 13 15 17 19 3 ...
##  $ sd.deg   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ diameter : num  1 1 1 1 1 1 1 1 1 1 ...
```

Construction du modèle de régression linéaire de `log(tps)` par rapport à `sqrt(dim)` et à toutes les autres variables de `data.graph` (ici, donc de `data.graph_modified`).

```r
# Ajustement du modèle linéaire de $\log(temps.moy)$ en fonction de toutes les variables présentes

data_graph_row_names <- c("tps", "dim", "mean.long", "mean.dist", "sd.dist", "mean.deg", "sd.deg", "dia
log_tps.lm <- lm(log_tps~., data = data.graph_modified) # calcul des résidus
summary(log_tps.lm) # display summary of usefulness of variables to see which ones are interesting
```

```
##
## Call:
## lm(formula = log_tps ~ ., data = data.graph_modified)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1593.1  -720.9  -160.3   774.8  2176.4
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8455.903   1956.377  -4.322 5.70e-05 ***
## `sqrt(dim)`  3944.335    858.351   4.595 2.17e-05 ***
## mean.long    3008.210   2617.990   1.149  0.25495
## mean.dist       3.764      3.816   0.986  0.32783
## sd.dist        -1.564      2.388  -0.655  0.51494
## mean.deg      -13.114    152.706  -0.086  0.93184
## sd.deg        470.330    313.133   1.502  0.13817
## diameter    -1906.636    562.189  -3.391  0.00121 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1045 on 62 degrees of freedom
## Multiple R-squared:  0.8851, Adjusted R-squared:  0.8722
## F-statistic: 68.25 on 7 and 62 DF,  p-value: < 2.2e-16
```

Plus la p-valeur (indiquée par `Pr(>|t|)`) est faible, plus le coef sert à quelque chose pour l'influence sur la "fitness" du modèle.

Ici, on se rends compte que seul le paramètre `sqrt(dim)` a une p-valeur très faible ($<$ 1e-04) Donc ici un seul paramètre suffit largement. Les autres variables sont probablement peu pertinentes dans le modèle.

Afin de sélectionner les variables explicatives du modèle, on va utiliser un critère appelé AIC.

Mise en œuvre d'une sélection de variables pour ne garder que les variables pertinentes.

```r
# AIC criterion for selecting explanatory variables using a Stepwise Algorithm
slm <- step(log_tps.lm)
```

```
## Start:  AIC=980.77
## log_tps ~ `sqrt(dim)` + mean.long + mean.dist + sd.dist + mean.deg +
##     sd.deg + diameter
##
##               Df Sum of Sq      RSS    AIC
## - mean.deg     1      8055 67726375 978.77
## - sd.dist      1    468480 68186800 979.25
## - mean.dist    1   1062475 68780795 979.86
## - mean.long    1   1442098 69160418 980.24
## <none>                     67718320 980.77
```

```
## - sd.deg       1   2464116 70182436 981.27
## - diameter     1  12562784 80281104 990.68
## - `sqrt(dim)`  1  23063862 90782182 999.28
##
## Step:  AIC=978.77
## log_tps ~ `sqrt(dim)` + mean.long + mean.dist + sd.dist + sd.deg +
##     diameter
##
##               Df Sum of Sq       RSS     AIC
## - sd.dist      1    867829  68594204  977.67
## - mean.long    1   1763882  69490258  978.57
## <none>                      67726375  978.77
## - sd.deg       1   2460002  70186377  979.27
## - mean.dist    1   3082428  70808804  979.89
## - diameter     1  13437906  81164282  989.44
## - `sqrt(dim)`  1 160475906 228202282 1061.81
##
## Step:  AIC=977.67
## log_tps ~ `sqrt(dim)` + mean.long + mean.dist + sd.deg + diameter
##
##               Df Sum of Sq       RSS     AIC
## - mean.long    1    910957  69505161  976.59
## <none>                      68594204  977.67
## - mean.dist    1   2466706  71060911  978.14
## - sd.deg       1   2509984  71104188  978.18
## - diameter     1  12692840  81287044  987.55
## - `sqrt(dim)`  1 174338671 242932876 1064.19
##
## Step:  AIC=976.59
## log_tps ~ `sqrt(dim)` + mean.dist + sd.deg + diameter
##
##               Df Sum of Sq       RSS     AIC
## <none>                      69505161  976.59
## - sd.deg       1   3556659  73061821  978.08
## - mean.dist    1   8770739  78275900  982.91
## - diameter     1  13201543  82706704  986.76
## - `sqrt(dim)`  1 409297955 478803116 1109.68
```

```
summary(slm)
```

```
##
## Call:
## lm(formula = log_tps ~ `sqrt(dim)` + mean.dist + sd.deg + diameter,
##     data = data.graph_modified)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1568.95  -724.03   -72.19   810.35  2010.04
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6570.086    880.252  -7.464 2.62e-10 ***
## `sqrt(dim)`  3559.255    181.924  19.564  < 2e-16 ***
## mean.dist       4.887      1.706   2.864  0.00563 **
```

```
## sd.deg        544.144    298.363   1.824  0.07279 .
## diameter    -1872.433    532.900  -3.514  0.00081 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1034 on 65 degrees of freedom
## Multiple R-squared:  0.8821, Adjusted R-squared:  0.8748
## F-statistic: 121.6 on 4 and 65 DF,  p-value: < 2.2e-16
```

La méthode AIC a donc sélectionné les variables `sqrt(dim)`, `mean.dist`, `sd.deg` et `diameter` en tant que variables explicatives pertinente pour le modèle.

Variables exclues du modèle : * `sd.dist` * `mean.long` * `mean.deg`

Test de Fisher : rappels

- *F-statistic* = test de Fisher de pertinence du modèle. Un modèle pertinent est un modèle tel que R^2 est significativement supérieur à 0. F = ($R^{2/1-R}$2)(N-K)/(K-1) avec K le nombre de variables dans le modèle (hors constante). La p-valeur donne la probabilité de se trompe si on affirme que le modèle n'est pas pertinent.

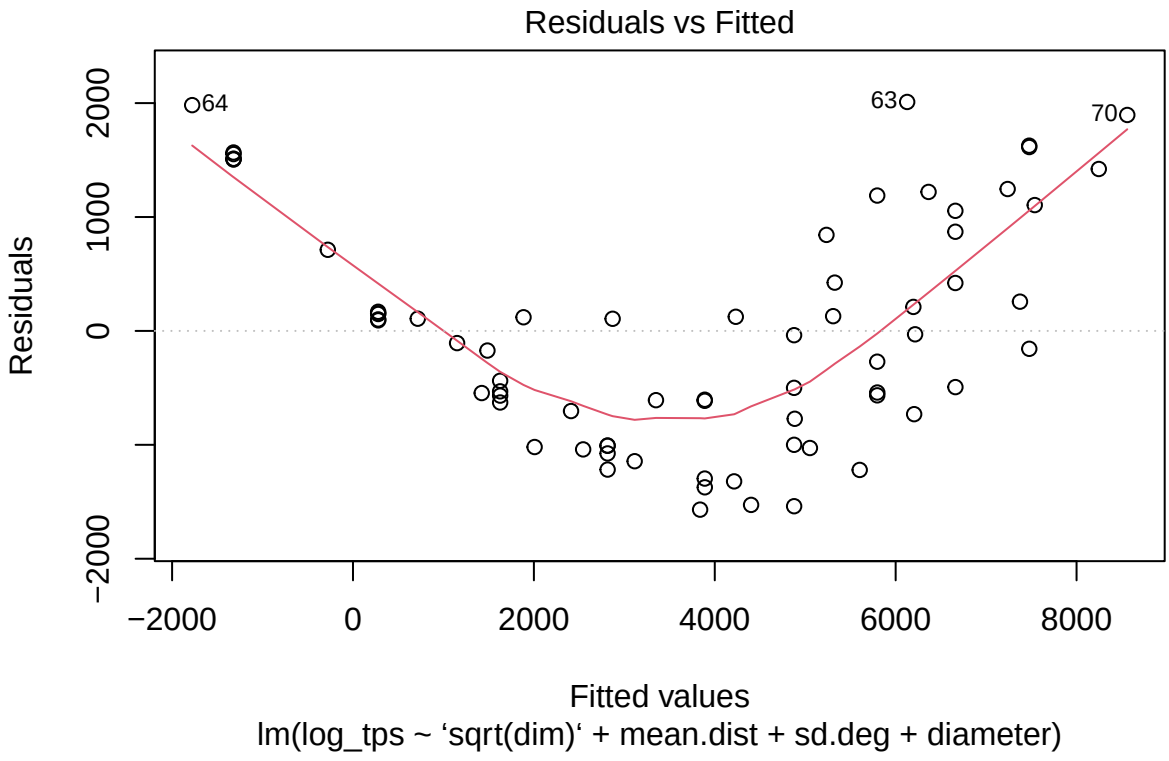
Ici, la p-valeur est très faible (< 1e-15), donc on peut réfuter (H0) et donc valider (H1), c'est-à-dire qu'on peut affirmer avec très peu de chances de se tromper, que la modélisation est fausse.

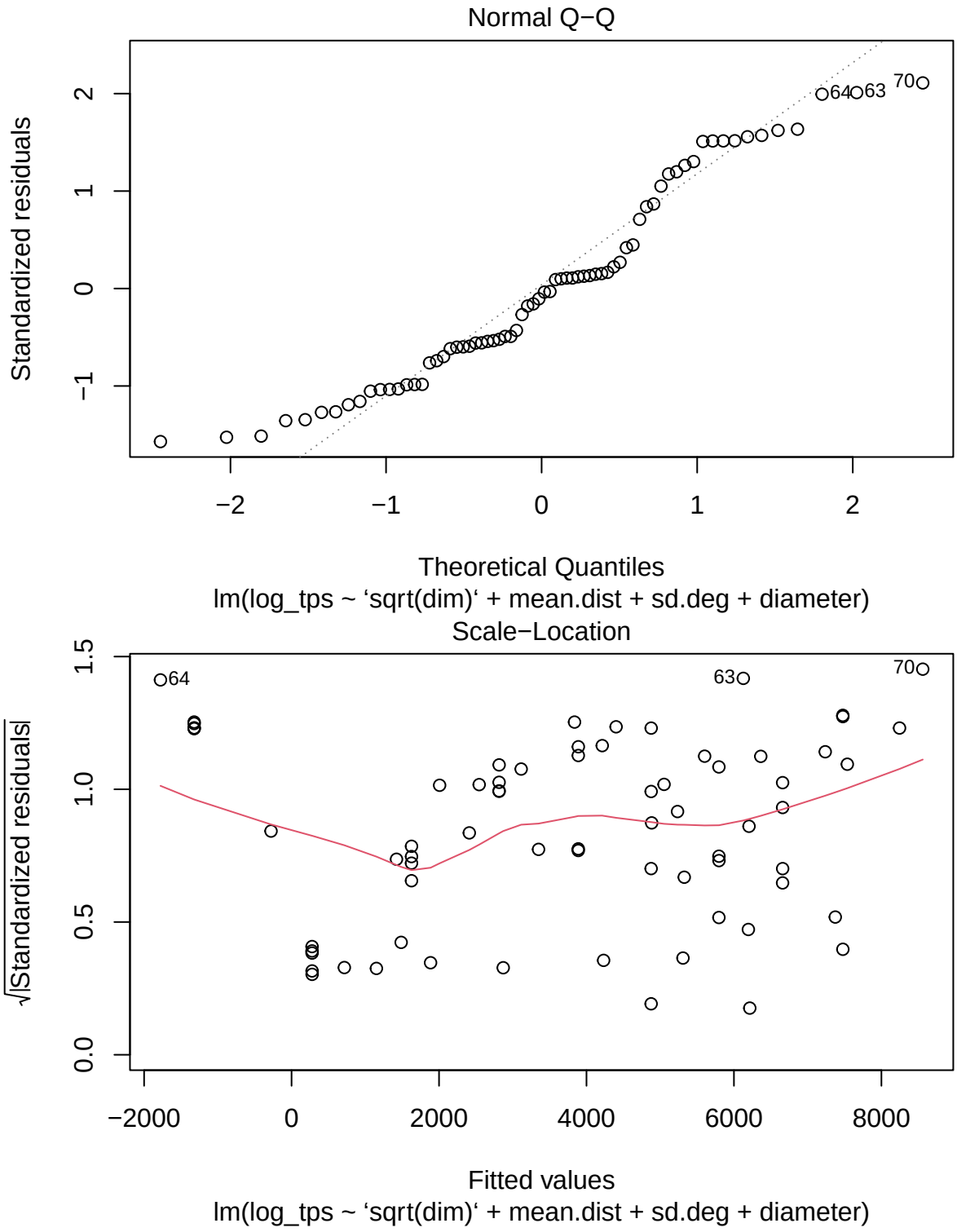
On va maintenant utiliser faire l'étude graphique des résidus, afin de vérifier la pertinence du modèle après sélection de variables par AIC.

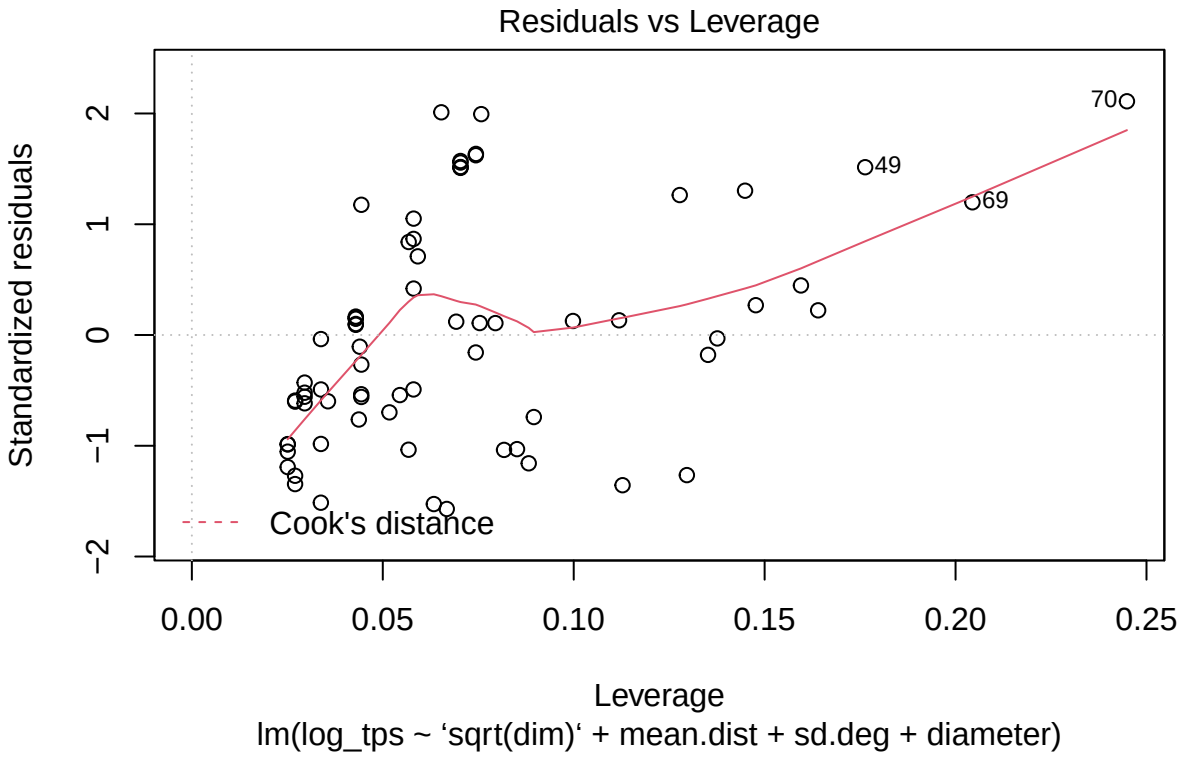
```
par(mflow=c(2,2)) # 4 graphiques
```

```
## Warning in par(mflow = c(2, 2)): "mflow" is not a graphical parameter
```

```
plot(slm)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(log_tps ~ ‘sqrt(dim)‘ + mean.dist + sd.deg + diameter)
Scale−Location
√|Standardized residuals|
Fitted values
lm(log_tps ~ ‘sqrt(dim)‘ + mean.dist + sd.deg + diameter)



- étude des hypothèses sur les résidus.

Interprétation des 4 courbes :

- **Residuals vs Fitted** : La courbe forme une hyperbole inversé visible, le modèle ne semble pas si bon que ça, un meilleur changement de variable pourrait permettre de faire mieux.
- **Normal Q-Q** : Ici, les points suivent relativement bien la droite la régression semble correcte.
- **Scale location** : Ici la droite semble suffisament plate. Les résidus sont possiblement nuls.
- **Residuals vs Leverage** : Si un point est un outliers il apparaitra trés éloigné des autres et en dehors des bornes par rapport à la distance de Cook. Ces bornes sont représentées par des lignes rouge en pointillé. Ici, tout semble correcte, aucun point ne dépasse les lignes de Cook.

L'analyse graphique des résidus ne permettant pas d'invalider le modèle avec sûreté, celui-ci semble potentiellement correct même si un meilleur modèle peut sans doute être trouvé.