TP Statistique

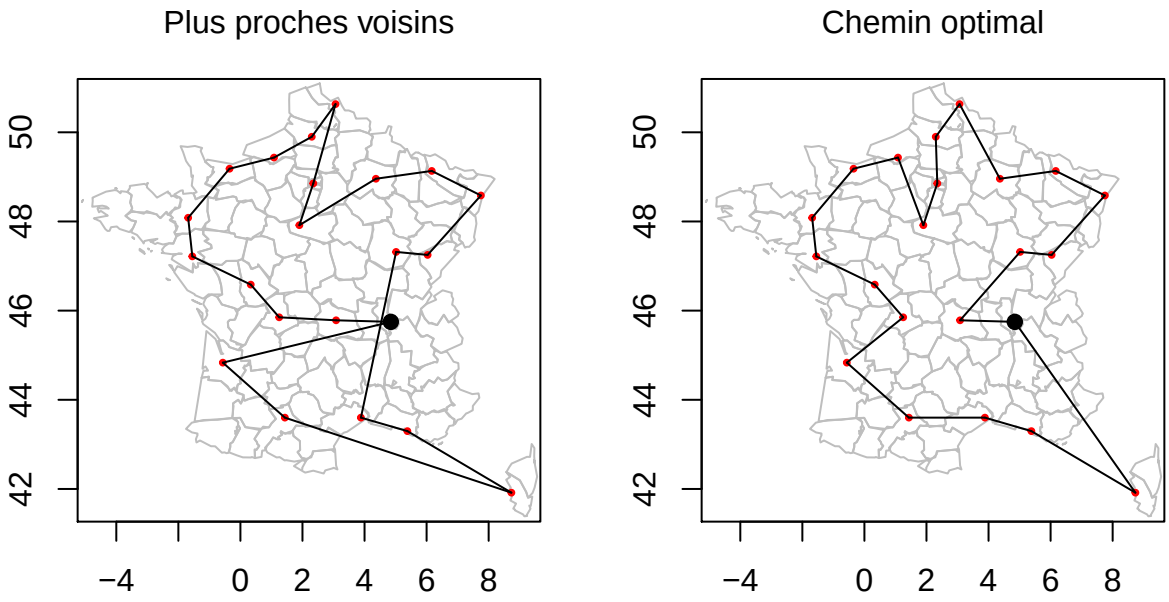
Emmy LERANDY et Louis UNG

31 mars 2020

0. Visualisation de chemins

On nous fournit dans ce TP des données de villes françaises stockées dans un fichier CSV et des algorithmes de calcul du chemin optimal.

Voici ci-dessous une représentation des chemins par la méthode des plus proches voisins et la méthode du chemin optimal :



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

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

et pour la méthode optimale :

```
## [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.

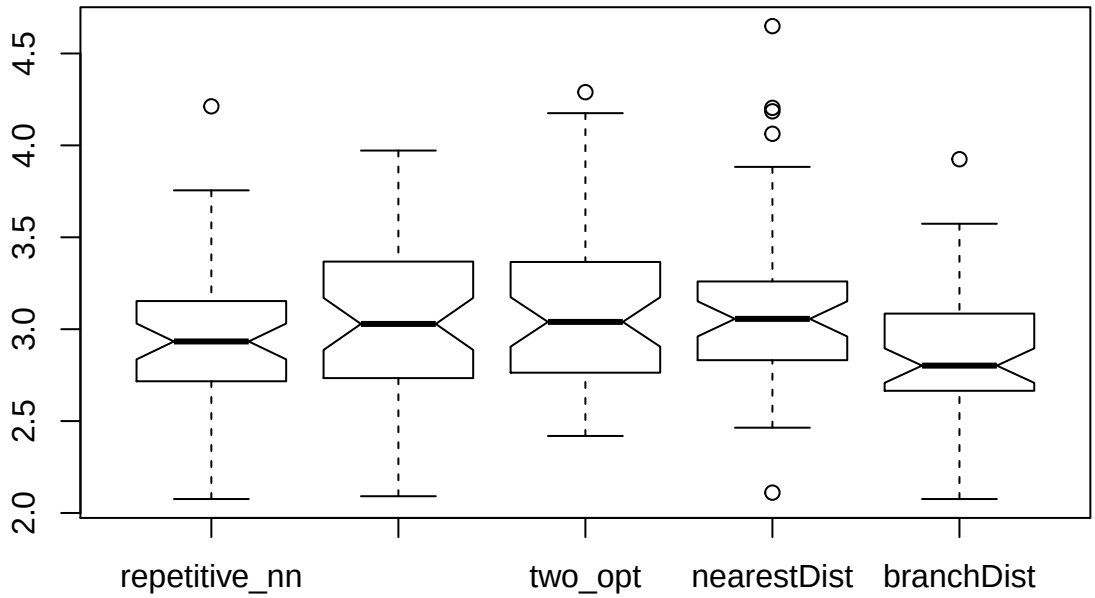
---

1. Comparaison d'algorithmes

1.1. Longueur des chemins

Dans cette partie, il s'agit de comparer 5 différentes méthodes de calcul du chemin optimal : repetitive_nn, nearest_insertion, two_opt, nearest et branch.

Boxplots des 5 méthodes



Commentaire : On constate que Branch&Bound a la plus petite médiane, et a donc calculé le meilleur chemin optimal. En ce qui concerne les autres algorithmes, comme le résultat n'est pas stable entre chaque exécution du code, il est difficile d'en déduire d'autres informations supplémentaires.

Test entre nearest et branch

```
##
##  One Sample t-test
##
## data:  nearestDist - branchDist
## t = 7.2937, df = 49, p-value = 2.33e-09
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.1788553 0.3148939
## sample estimates:
## mean of x
## 0.2468746
```

Commentaire : L'espérance obtenue appartient à l'intervalle de confiance à 95%, nous ne rejettons donc l'hypothèse H0, l'espérance de l'algorithme Branch&Bound est donc inférieur à celui des plus proches voisins.

Tests 2 à 2

```
results <- vector(length = 250)
methods <- vector(length = 250)

results <- c(nearestDist, branchDist, nearest_insertion,repetitive_nn, two_opt)

for (i in 1:250){
  if (i < 51){
    methods[i] <- 'nearest'
  }
  else if (i < 101){
    methods[i] <- 'Branch&Bound'
  }
  else if (i < 151) {
    methods[i] <- 'nearest_insertion'
  }
  else if (i < 201) {
    methods[i] <- 'repetitive_nn'
  }
  else {
    methods[i] <- 'two_opt'
  }
}

pairwise.t.test(results,methods,adjust.method='bonferroni')
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  results and methods
##
##                  Branch&Bound nearest nearest_insertion repetitive_nn
## nearest          0.030        -       -                 -
```

```
## nearest_insertion 0.131        1.000   -                -
## repetitive_nn     1.000        0.411   0.945            -
## two_opt           0.051        1.000   1.000            0.551
##
## P value adjustment method: holm
```

Les résultats obtenus du tests de la correction multiple de Bonferroni sont présents ci-dessus.

Commentaire : Nous observons que la difference entre Branch&Bound et nearest_insertion, Branch&Bound et repetitive_nn, nearest_insertion et repetitive_nn, nearest et two_opt est supérieur ou égal 10^-5. Nous pouvons donc conclure pour ces tests que l'hypothèse H0 est rejetée, ils n'ont donc pas la même espérance 2 à 2. A l'inverse, pour les autres pairs de tests, les résultats sont tous inférieurs ou égaux à 10^-12, valeur qu'on peut considérer suffisamment négligeable pour être approximée à 0 Nous pouvons donc pour ces pairs valider l'hypothèse H0, ils ont donc la même espérance 2 à 2.

En résumé, les pairs de tests validant l'hypothèse H0 sont: – nearest et Branch&Bound – nearest et insertion – nearest et repetitive_nn – Branch&Bound et two_opt – nearest_insertion et two_opt – repetitive_nn et two_opt

1.2. Temps de calcul

Comparaison des temps à l'aide du package microbenchmark.

Exemple d'application de microbenchmark :

```
microbenchmark(sqrt(x),x^0.5, times=100, setup={x <- runif(1)})
```

```
## Unit: nanoseconds
##     expr min  lq mean median  uq  max neval cld
##  sqrt(x) 100 100  222    200 200 3800   100  a
##    x^0.5 200 300  458    300 400 9800   100   b
```

Microbenchmark

```
microbenchmark(TSPsolve(couts, 'repetitive_nn'), TSPsolve(couts,'nearest_insertion'),
TSPsolve(couts, 'two_opt'), TSPnearest(couts)$longueur, TSPbranch(couts), times=1,
setup={
  n <- 10
  sommets <- data.frame(x = runif(n), y = runif(n))
  couts <- distance(sommets)
})
```

```
## Unit: microseconds
##                                 expr    min     lq   mean median     uq
##      TSPsolve(couts, "repetitive_nn") 5696.8 5696.8 5696.8 5696.8 5696.8
##  TSPsolve(couts, "nearest_insertion")  917.1  917.1  917.1  917.1  917.1
##            TSPsolve(couts, "two_opt")  553.6  553.6  553.6  553.6  553.6
##            TSPnearest(couts)$longueur   25.1   25.1   25.1   25.1   25.1
##                      TSPbranch(couts) 1368.5 1368.5 1368.5 1368.5 1368.5
##     max neval
##  5696.8     1
##   917.1     1
```

```
##   553.6     1
##    25.1     1
##  1368.5     1
```

Nous avons décidé d'augmenter le nombre de graphes étudiés car les résultats qu'on obtenait étaient trop instables avec 20 graphes. Après plusieurs essais, sur 400 graphes, nous obtenons dans la majorité des cas pour la dernière colonne le classement suivant : d pour repetitive_nn, b pour nearest_insertion, b pour two_opt, a pour nearest et c pour Branch&Bound.

Commentaire: Le plus rapide en temps d'exécution parmi les 5 méthodes d'après microbenchmark est donc la méthode nearest, suivie de nearest_insertion et two_opt, puis par Branch&Bound et enfin repetitive_nn qui est la méthode la plus lente. On note que nearest_insertion et two_opt ont la même classe b, ce qui signifie que le test entre ces 2 méthodes valide l'hypothèse H0, nous avons donc une égalité entre l'espérance des temps de l'algorithme nearest_insertion et two_opt avec un risque de se tromper de 5%.

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

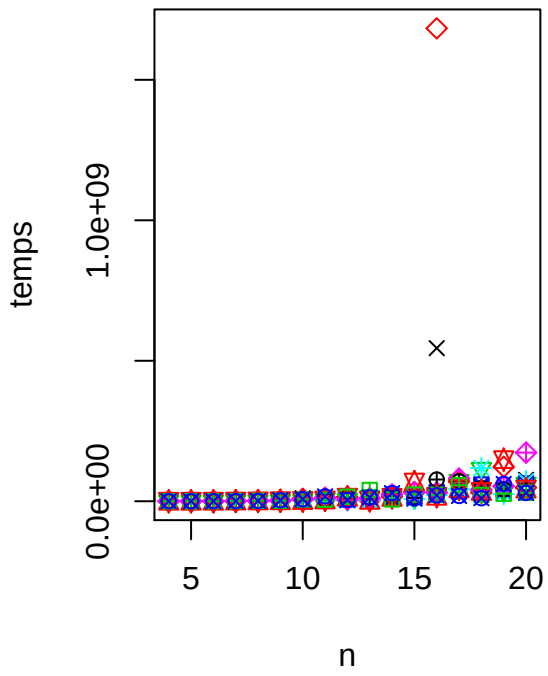
2.1. Comportement par rapport au nombre de sommets : premier modèle

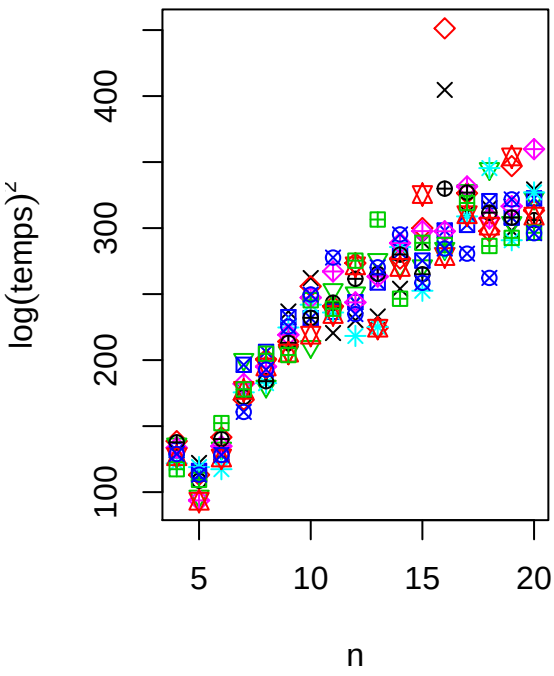
Récupération du temps sur 10 graphes pour différentes valeurs de n.

```
seqn <- seq(4,20,1)
temps <- matrix(ncol=10, nrow=length(seqn))
for (i in 1:length(seqn)){
  temps[i,] <- microbenchmark(TSPsolve(couts, method = 'branch'),
                 times = 10,
                 setup = { n <- seqn[i]
                 couts <- distance(cbind(x = runif(n), y = runif(n)))}
  )$time
}
```

Ajustement du modèle linéaire de $\log(temps)^2$ en fonction de n.

```
par(mfrow=c(1,2)) # 2 graphiques sur 1 ligne
matplot(seqn, temps, xlab='n', ylab='temps', pch=seqn)
matplot(seqn, log(temps)^2, xlab='n', ylab=expression(log(temps)^2), pch=seqn)
```






```
vect_temps <- log(as.vector(temps))^2
vect_dim <- rep(seqn,times=10)
```

Après ajustement, on obtient une courbe plus proche d'une droite.

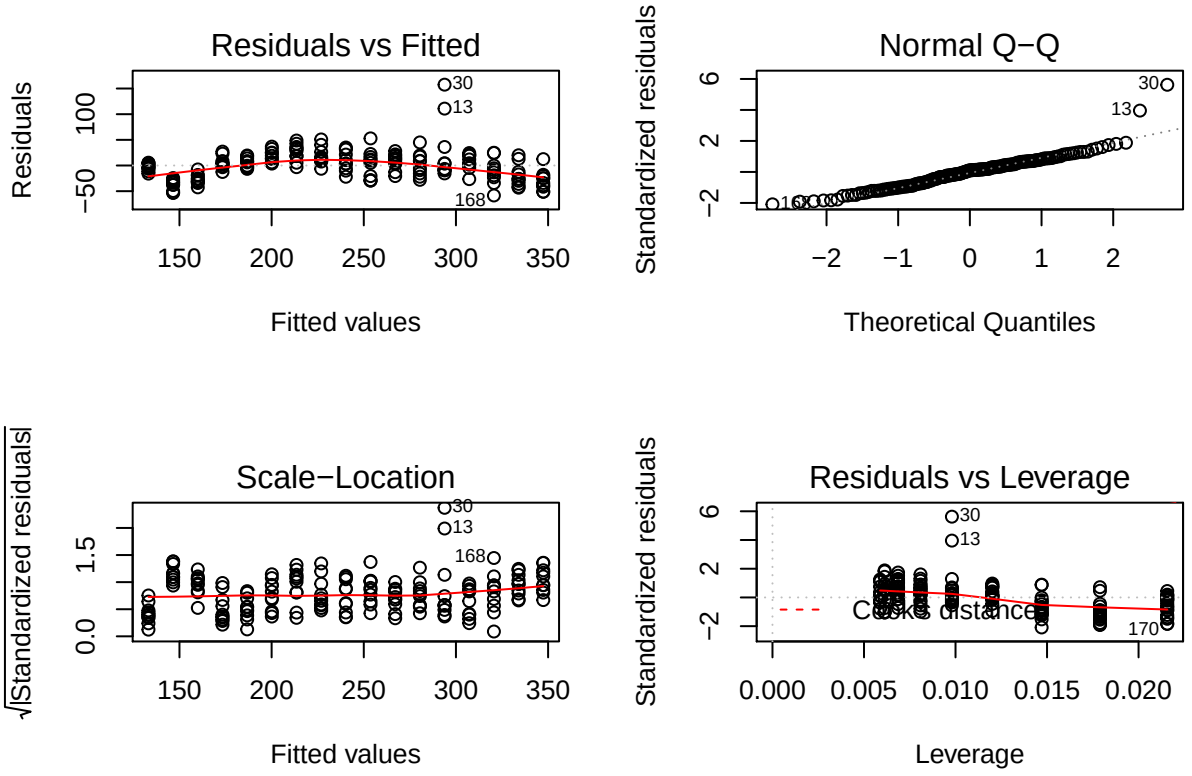
```
temps.lm <- lm(vect_temps ~ vect_dim)
summary(temps.lm)
```

```
##
## Call:
## lm(formula = vect_temps ~ vect_dim)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.355 -20.200   2.777  16.618 157.470
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  79.6322     5.7095   13.95   <2e-16 ***
## vect_dim     13.3877     0.4405   30.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.14 on 168 degrees of freedom
## Multiple R-squared:  0.8461, Adjusted R-squared:  0.8452
## F-statistic: 923.7 on 1 and 168 DF,  p-value: < 2.2e-16
```

Commentaire : On observe qu'on obtient un coefficient R^2 ajusté valant 0.8758, on est donc proche de 1. Or plus ce coefficient est proche de 1, plus la courbe obtenue sera proche d'une droite. Nous avons donc une relation de proportionnalité entre le logarithme au carré du temps d'exécution et le nombre de sommets. Par ailleurs, on obtient une p-value inférieure à 10^-16, donc très négligeable et approximable à la valeur 0, on ne rejette donc pas l'hypothèse H0 et le test de Fisher est validé. On en conclut que ce modèle est donc pertinent.

Etude graphique

```
par(mfrow=c(2,2))
plot(temps.lm)
```



Commentaire : Dans les graphes Residuals vs Fitted et Scale-Location, on observe que les points ne sont pas homogènes et une droite non-horizontale, on a donc pas de linéarité. Dans le graphe Normal Q-Q, on obtient une droite très semblable d'une diagonale, la distribution des résidus peut donc être assimilée à une distribution normale. Dans le graphe Residuals vs Leverage, on est dans les bornes de la distance de Cook donc OK

Test de Shapiro-Wilk

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

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  residuals(temps.lm)
## W = 0.92401, p-value = 8.995e-08
```

Commentaire : On obtient une p-value grande, qui est largement supérieure au risque alpha qui est de 0.05. Ainsi, on ne rejette pas l'hyphothèse H0, les résidus du modèles suivent donc possiblement une distribution normale.

Conclusion : le modèle est valide.

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

Ajustement du modèle de régression linéaire simple gaussien de log(temps.moy)^2 en fonction de seqn

// rajouter un plot pour montrer qu'on a une droite

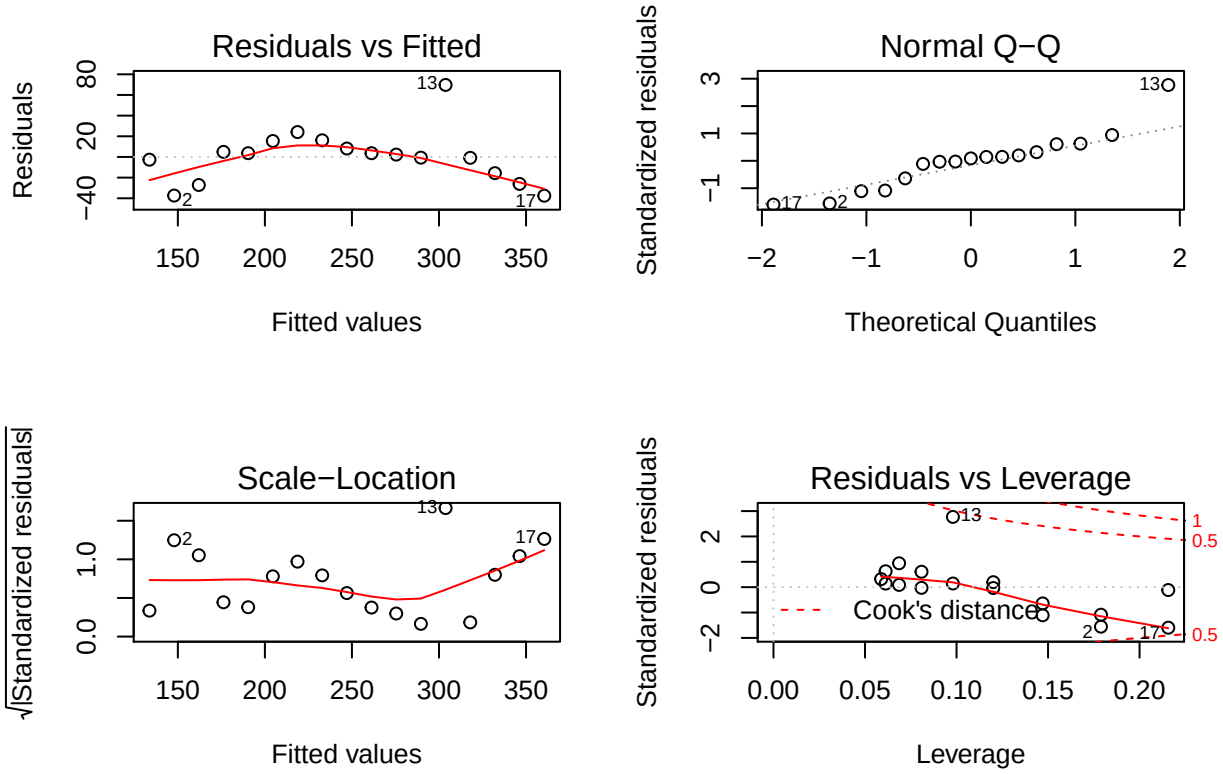
```
temps.moy <- rowMeans(temps)
vect_temps.moy <- log(as.vector(temps.moy))^2
temps.moy.lm <- lm(vect_temps.moy ~ seqn)
summary(temps.moy.lm)
```

```
##
## Call:
## lm(formula = vect_temps.moy ~ seqn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.515 -15.672   2.281   8.162  69.705
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   77.000     17.008   4.527 0.000401 ***
## seqn          14.174      1.312  10.802 1.79e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.5 on 15 degrees of freedom
## Multiple R-squared:  0.8861, Adjusted R-squared:  0.8785
## F-statistic: 116.7 on 1 and 15 DF,  p-value: 1.795e-08
```

Commentaire : On observe qu'on obtient un coefficient R^2 ajusté valant 0.9385, on est donc proche de 1. Or plus ce coefficient est proche de 1, plus la courbe obtenue sera proche d'une droite. Nous avons donc une relation de proportionnalité entre le logarithme au carré du temps d'exécution et le nombre de sommets. Par ailleurs, on obtient une p-value inférieure à 1^-10, donc très négligeable et approximable à la valeur 0, on ne rejette donc pas l'hypothèse H0 et le test de Fisher est validé. On en conclut que la statistique F = 0, donc la modèle n'apporte aucune information utile.

Etude graphique

```r
par(mfrow=c(2,2))
plot(temps.moy.lm)
```



Dans les graphes Residuals vs Fitted et Scale-Location, on observe que les points ne sont pas homogènes et une droite non-horizontale, on n'a donc pas de linéarité dans ce cas aussi. Dans le graphe Normal Q-Q, on obtient des points qui sont éparpillés autour de la diagonale, mais si on cherche à relier ces points on obtient une courbe ressemblant à une diagonale, on en déduit donc que la distribution des résidus peut être assimilée à une distribution normale. Dans le graphe Residuals vs Leverage, on observe un outlier (le numéro 1), la qualité de l'échantillon en est donc impactée. Il faudrait supprimer cet outlier pour améliorer la qualité de l'échantillon des données et augmenter leur pertinence. Cependant, l'écart est relativement faible et nous pouvons donc quand même affirmer que nous avons un échantillon de qualité acceptable.

Test de Shapiro-Wilk

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

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(temps.moy.lm)
## W = 0.90371, p-value = 0.07837
```

Commentaire : On obtient à l'issue du test une p-value de valeur élevée. Cette p-value est largement

supérieure au risque alpha qui est de 5%, nous ne rejettons pas l'hypothèse H0. Les résidus du modèle peuvent donc suivre une distribution normale.

Conclusion : le modèle est valide

2.3. Comportement par rapport à la structure du graphe

Construction du modèle de régression

Nous allons construire le modèle de régression de log(tps)^2 par rapport à sqrt(dim) et toutes les autres variables, exceptées la variable tps. Une fois le modèle construit, nous mettons en oeuvre la sélection de variables à l'aide de la fonction step. On obtient le résultat ci-dessous :

```
data.graph <- read.csv('DonneesTSP.csv',header=TRUE,dec='.',sep=',',quote="\"")

dataset <- data.frame(matrix(ncol = 7, nrow = 70))
colnames(dataset) <- c("sqrtdim", "mean.long","mean.dist",
                       "sd.dist","mean.deg", "sd.deg", "diameter")
dataset$sqrtdim = sqrt(data.graph$dim)
dataset$mean.long = data.graph$mean.long
dataset$mean.dist = data.graph$mean.dist
dataset$sd.dist = data.graph$sd.dist
dataset$mean.deg = data.graph$mean.deg
dataset$sd.deg = data.graph$sd.deg
dataset$diameter = data.graph$diameter

vect_tps <- log(as.vector(data.graph$tps))^2
tps.lm <- lm(vect_tps ~., data = dataset)
step(tps.lm)
```

```
## Start:  AIC=272.28
## vect_tps ~ sqrtdim + mean.long + mean.dist + sd.dist + mean.deg +
##     sd.deg + diameter
##
##             Df Sum of Sq     RSS    AIC
## - mean.dist  1      75.2  2798.5 272.18
## <none>                    2723.2 272.28
## - diameter   1     230.7  2953.9 275.97
## - sd.deg     1     269.1  2992.3 276.87
## - mean.deg   1     552.5  3275.7 283.21
## - mean.long  1    2122.7  4846.0 310.62
## - sd.dist    1    2483.9  5207.2 315.65
## - sqrtdim    1   14671.8 17395.0 400.08
##
## Step:  AIC=272.18
## vect_tps ~ sqrtdim + mean.long + sd.dist + mean.deg + sd.deg +
##     diameter
##
##             Df Sum of Sq   RSS    AIC
## <none>                    2798 272.18
## - sd.deg     1       299  3098 277.29
## - mean.deg   1       714  3512 286.08
## - diameter   1       718  3516 286.17
```

```
## - mean.long  1      2068  4867 308.92
## - sd.dist    1      3116  5914 322.56
## - sqrtdim    1     44208 47007 467.67
```

```
##
## Call:
## lm(formula = vect_tps ~ sqrtdim + mean.long + sd.dist + mean.deg +
##     sd.deg + diameter, data = dataset)
##
## Coefficients:
## (Intercept)      sqrtdim    mean.long      sd.dist     mean.deg
##     -10.118       93.493     -113.470        0.102       -2.447
##      sd.deg     diameter
##       5.159      -11.133
```

Résultat de la sélection de variables

Nous observons qu'après la mise en oeuvre de la sélection de variables, la variable mean.dist a été exclue du modèle par l'algorithme de la fonction step(). Par ailleurs, nous pouvons constater qu'individuellement toutes les variables conservées ne sont pas autant pertinentes. En effet, on cherche à maximiser l'AIC et la variable sqrtdim semble ainsi être largement plus pertinente que les autres.

Test de Fisher

Nous effectuons le test de Fisher avec le nouveau modèle sans la variable mean.dist, comme elle a été déterminée comme non pertinente précédemment.

```
datasetAfterAIC <- data.frame(matrix(ncol = 6, nrow = 70))
colnames(datasetAfterAIC) <- c("sqrtdim", "mean.long",
                        "sd.dist","mean.deg", "sd.deg", "diameter")
datasetAfterAIC$sqrtdim = sqrt(data.graph$dim)
datasetAfterAIC$mean.long = data.graph$mean.long
datasetAfterAIC$sd.dist = data.graph$sd.dist
datasetAfterAIC$mean.deg = data.graph$mean.deg
datasetAfterAIC$sd.deg = data.graph$sd.deg
datasetAfterAIC$diameter = data.graph$diameter

tps.lm_aic <- lm(vect_tps ~., data = datasetAfterAIC)
summary(tps.lm_aic)
```

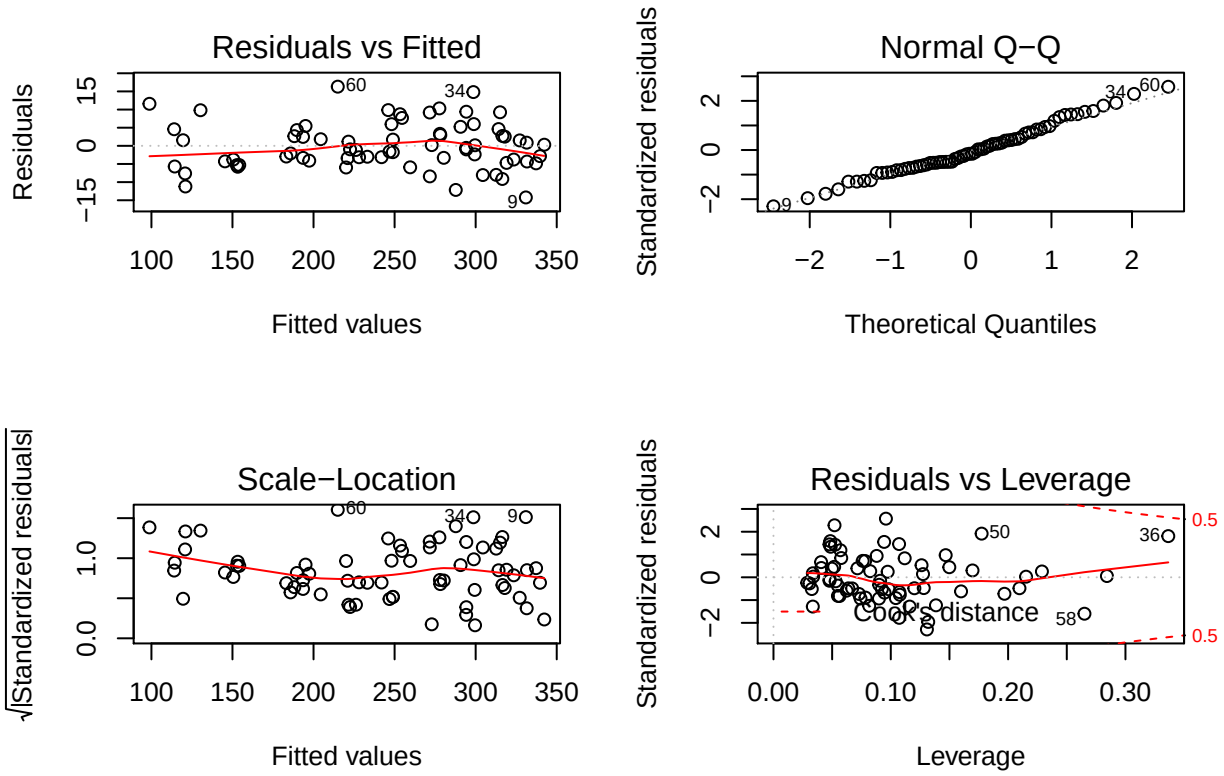
```
##
## Call:
## lm(formula = vect_tps ~ ., data = datasetAfterAIC)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2246  -4.2312  -0.9755   4.0830  16.3081
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -10.11788    8.50435  -1.190 0.238617
## sqrtdim       93.49324    2.96359  31.547  < 2e-16 ***
```

```
## mean.long   -113.47013   16.62867  -6.824 4.05e-09 ***
## sd.dist        0.10196    0.01217   8.375 7.90e-12 ***
## mean.deg      -2.44693    0.61053  -4.008 0.000165 ***
## sd.deg         5.15950    1.98806   2.595 0.011745 *
## diameter     -11.13317    2.76963  -4.020 0.000158 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.665 on 63 degrees of freedom
## Multiple R-squared:  0.9913, Adjusted R-squared:  0.9905
## F-statistic:  1199 on 6 and 63 DF,  p-value: < 2.2e-16
```

Commentaire : On observe qu'on obtient un coefficient R^2 ajusté valant 0.9905, on est donc très proche de 1. Nous avons donc une relation de proportionnalité entre le logarithme au carré du temps moyen d'exécution de l'algorithme Branch&Bound et le nombre de sommets. Par ailleurs, on obtient une p-value inférieure à 10^-16, donc très négligeable et approximable à la valeur 0, on ne rejette donc pas l'hypothèse H0 et le test de Fisher est validé. On en conclut que ce modèle est donc pertinent.

Etude des résidus

```
par(mfrow=c(2,2))
plot(tps.lm_aic)
```



Commentaire : Dans les graphes Residuals vs Fitted et Scale-Location, on observe que les points ne sont pas homogènes et une droite non-horizontale, on n'a donc pas de linéarité dans ce cas aussi. Dans le graphe

Normal Q-Q, on obtient une courbe ressemblant à une diagonale, on en déduit donc que la distribution des résidus peut être assimilée à une distribution normale. Dans le graphe Residuals vs Leverage, on observe que tous les points dans compris dans la zone délimitée par les pointillés rouge, calculée à partir de la distance de Cook. Nous n'avons donc pas de valeurs aberrantes présentes dans cet échantillon, nous pouvons donc en déduire que nous avons un échantillon de bonne qualité.

Test de Shapiro-Wilk

```
shapiro.test(residuals(tps.lm_aic))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(tps.lm_aic)
## W = 0.98303, p-value = 0.4621
```

Commentaire : On obtient une p-value grande, qui est largement supérieure au risque alpha qui est de 0.05. Ainsi, on ne rejette pas l'hyphothèse H0, les résidus du modèles suivent donc possiblement une distribution normale.

Conclusion : le modèle est valide.