Projet MRR 2017/2018

Un modèle prédictif : Parkinson Telemonitoring

Ghassen Jlassi - Walid Tlili

21 janvier 2018

Table des matières

1 Analyse préliminaire **2**
1.1 Introduction . . . 2
1.2 Analyse des observations . . . 2
1.3 Analyse des variables . . . 2
1.4 Analyse des corrélations . . . 2
1.5 Étude du modèle linéaire avec toute les variables explicatives . . . 3

2 Établissement du modèle prédictif **4**
2.1 Introduction . . . 4
2.2 Mise en place des régressions Backward, Forward et Stepwise . . . 4
2.3 Une méthode d'échantillonage : La méthode k-fold . . . 4
2.4 Régression RIDGE . . . 5
2.5 Régression LASSO . . . 6
2.6 Elastic-Net . . . 7
2.7 Choix du modèle . . . 7

3 Conclusion **8**

4 Annexes **9**
4.1 Code R . . . 9
4.2 Graphiques . . . 13

1 Analyse préliminaire

1.1 Introduction

Dans le cadre de notre projet, nous avons choisi la base de données Pakinsons Telemonitoring. Nous nous intéresserons ici à l'analyse des observation de la base de donnée et des variables explicatives et étudiées du modèle.

1.2 Analyse des observations

Notre support de travail contient 5875 observations et 26 variables. Il s'agit d'une base de donnée dont une étude servirait à établir un score spécifique respectant certains critères qui permettrait d'évaluer le degré d'avancement des symptômes de la maladie de Parkinson. Il serait donc judicieux de commencer par repérer les variables cibles et les variables explicatives.

1.3 Analyse des variables

Cet ensemble de données est composé d'une gamme de mesures vocales biomédicales de 42 personnes atteintes de la maladie de Parkinson au stade précoce recrutées pour un essai de six mois d'un dispositif de télésurveillance pour la surveillance à distance de la progression des symptômes.
Les colonnes du tableau contiennent le numéro du sujet, l'âge du sujet, le sexe du sujet, l'intervalle de temps entre la date de recrutement initiale, l'UPDRS moteur, l'UPDRS total et 16 mesures vocales biomédicales. Chaque rangée correspond à l'un des 5 875 enregistrements vocaux de ces individus.

L'objectif principal de ces données est de prédire les scores UPDRS moteur et total ('motor_UPDRS' et 'total_UPDRS') parmi les 16 mesures vocales.
Par ailleurs, cette base de données suggère une étude multi-variée au vu du nombre de variable mesurées sur les unités statistiques.

1.4 Analyse des corrélations

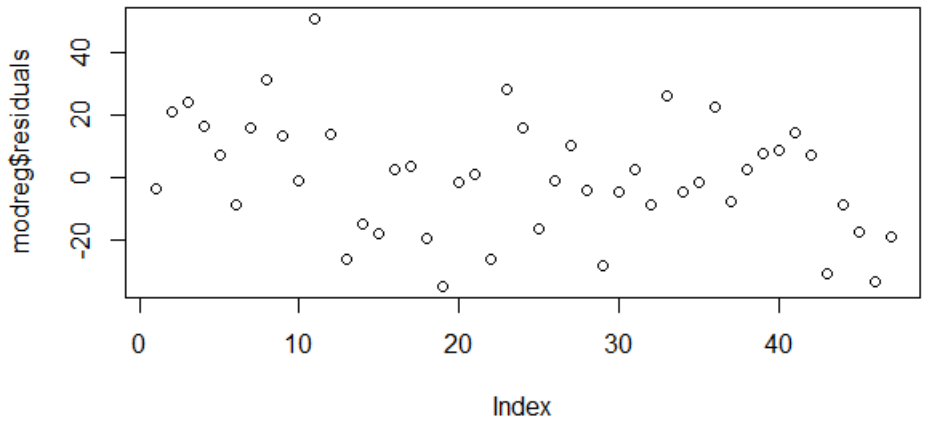
Nous avons tout d'abord standardiser les données avant de calculer la matrice de corrélation pour éviter la présence d'erreurs sur les coefficients de la matrice de corrélation.

Nous remarquons que l'on a une forte corrélation entre les variables explicatives du modèle. Ainsi, on ne sait pas encore quelle variable va plus influer sur les autres dans le cadre de notre étude.
Concernant les variables à trouver, elles possèdent toute deux une forte corrélation avec l'âge et le type de sujet.

1.5 Étude du modèle linéaire avec toute les variables explicatives

L'étape suivante de l'étude est donc de visualiser le modèle de régression linéaire simple en prenant en compte toute les variables explicatives modèle.

Nous établissons donc le modèle linéaire simple via la fonction lm puis nous analysons le graphique des résidus. Nous obtenons le graphique suivant :



Nous voyons donc qu'il y a une valeur qui possède une très forte variance. Ceci peut fausser les résultats finaux de la régression. Une solution serait de retirer cette observation lié cette valeur de résidu, de recalculer la régression puis de comparer et de voir s'il y a des différences significatives.

Ensuite, via le summary de ce modèle linéaire, nous localisons les variables significatives avec une faible p-value pour les garder dans le cadre de cette étude.

2 Établissement du modèle prédictif

2.1 Introduction

Nous allons dans la suite du projet, faire une sélection de variables et de modèles à l'aide des méthodes de régressions Lasso, Ridge, Backward, Forward, Stepwise.

2.2 Mise en place des régressions Backward, Forward et Stepwise

Après avoir localisé la variable cible et avoir étudié la matrice de corrélation, nous passons à présent à la mise en oeuvre de la regression Backward. Cette dernière appliquée à notre jeu de données nous donne au final un AIC égal à l'AIC du modèle linéaire de départ, égal à 2759.81, ce qui n'est pas intéressant car nous cherchons l'AIC le plus faible permettant de pénaliser les modèles en fonction du nombre de paramètres afin de satisfaire le critère de Parcimonie.

Puis nous faisons ensuite une régression Backward qui va prendre en considération toute les variables au départ puis va supprimer les variables qui sont le moins significatives du modèle, dans le but d'avoir un AIC inférieur à celui du modèle de base. En effet, on obtient un AIC égal à 27450.17, ce qui est déja inférieur à l'AIC antérieur. Cependant, l'inconvénient d'une telle méthode est qu'une variable introduite dans le modèle ne peut plus être éliminée. Le modèle finale peut donc contenir des variables non significatives, d'où l'usage de la méthode Stepwise.

Enfin, on effectue une régression Stepwise, qui est une combinaison des deux précédents type de regression, qui va tester à chaque itération si une variable doit être conservée ou éliminée. L'avantage de cette méthode est qu'elle prend en compte les corrélations présentes entre les variables. En effet, une variable considéré comme la plus significative à une étape de l'algorithme peut à une étape antérieure devenir non significative.

La régression Stepwise donne comme AIC un AIC égal à celui obtenu par la régression Backward, mais une meilleure sélection de variable.

2.3 Une méthode d'échantillonage : La méthode k-fold

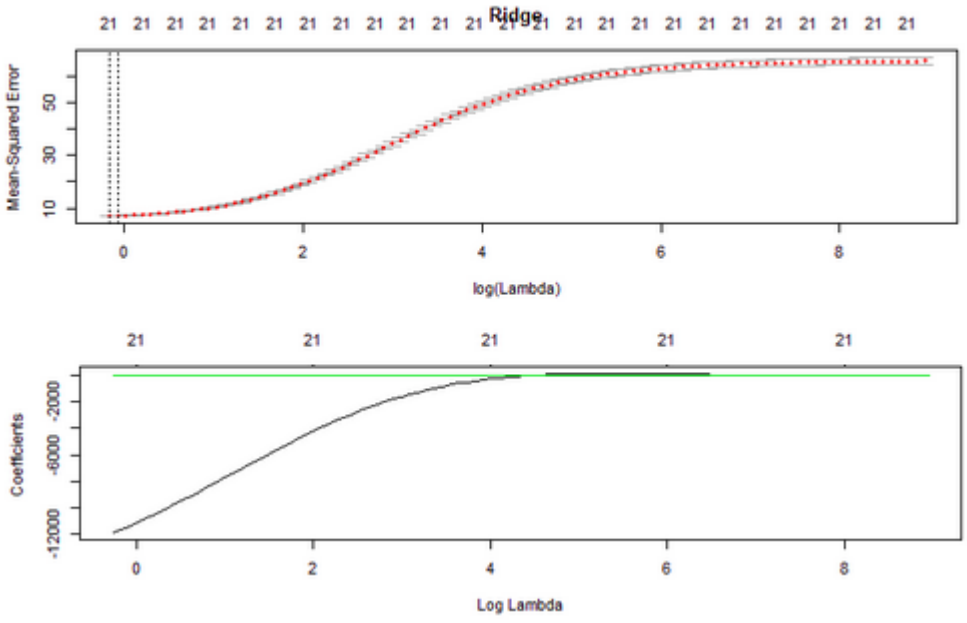
Pour augmenter la précision de nos résultats et sélectionner un modèle prédictif efficace et fiable, la méthode du k-fold est idéal dans la mesure où

elle permet de vérifier le critère de validation croisée. Pour notre étude, nous avons choisi un k=10 pour avoir plus de précision sur le modèle que l'on va choisir. Nous utilisons la fonctions glmnet pour établir nos modèles de régressions Ridge,Lasso et Elastic-net en fonction des valeurs de la variable alpha.

Après l'étude de ces trois derniers modèles de régression, nous passons à présent à des méthodes impliquant l'emploi de pénalités pour une meilleurs sélection de modèle : La régression Lasso et la régression Ridge.

2.4 Régression RIDGE

On importe tout d'abord la librairie Ridge dans RStudio, puis on va effectuer une régression Ridge. Cette dernière permet de réduire l'amplitude des coefficients d'une régression linéaire et d'éviter le sur-apprentissage. Pour cela, la régression Ridge fait tendre les coefficients du modèle non-significatif vers 0, réduisant ainsi au maximum leur impact.
Ce qui est intéressant ici est le choix du paramètre de régularisation lambda. En pratique, on constate que le chemin de régularisation de la régression Ridge est continue, ne permettant pas un ajustement aisé de lambda. On choisi donc notre paramètre par validation croisée sur une grille de valeurs de lambda supérieurs à 0.
Nous représentons dans ces graphiques le tracé des coefficients et des erreurs quadratiques moyennes en fonction du log de lambda.

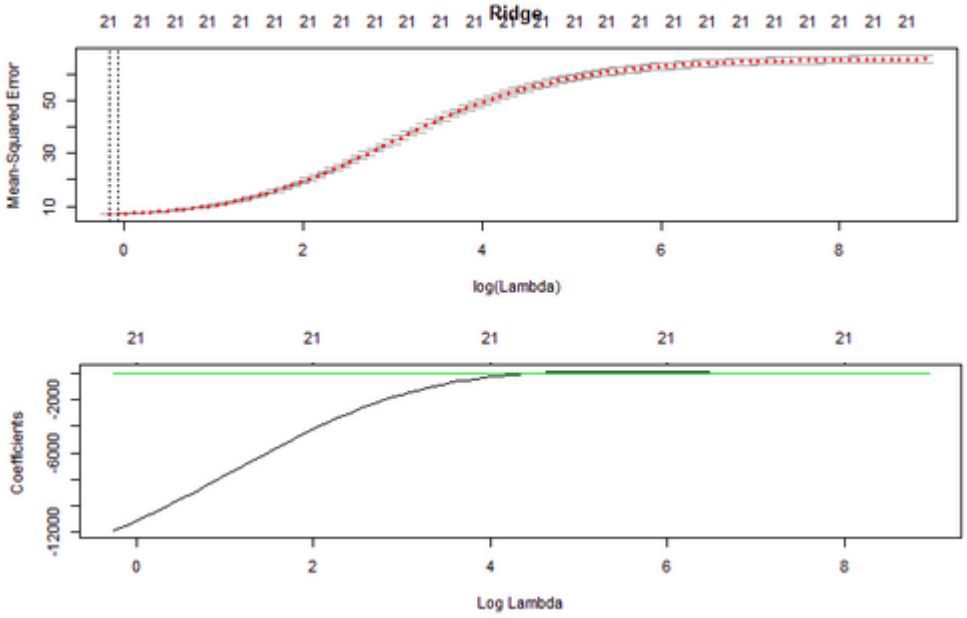


Nous avons choisi comme abscisse le logarithme de notre paramètre car le log est plus sensible à la variation et nous permet d'avoir une meilleur vision sur les éléments recherchés.

La régression Ridge permet de faire tendre les variables non-significatives vers 0. Nous cherchons à présent à aller plus loin et à annuler complètement les variables non-significatives. Pour cela, nous établissons un modèle via la méthode de régression Lasso.

2.5 Régression LASSO

L'utilité de la méthode LASSO est qu'elle impose comme valeur 0 aux variables qui sont non-significative. Elles ne feront donc plus partie du modèle.
Une telle méthode est dite parcimonieuse car le modèle possèdera beaucoup de coefficients nuls.
Pour un alpha=1, nous obtenons les courbes suivantes :

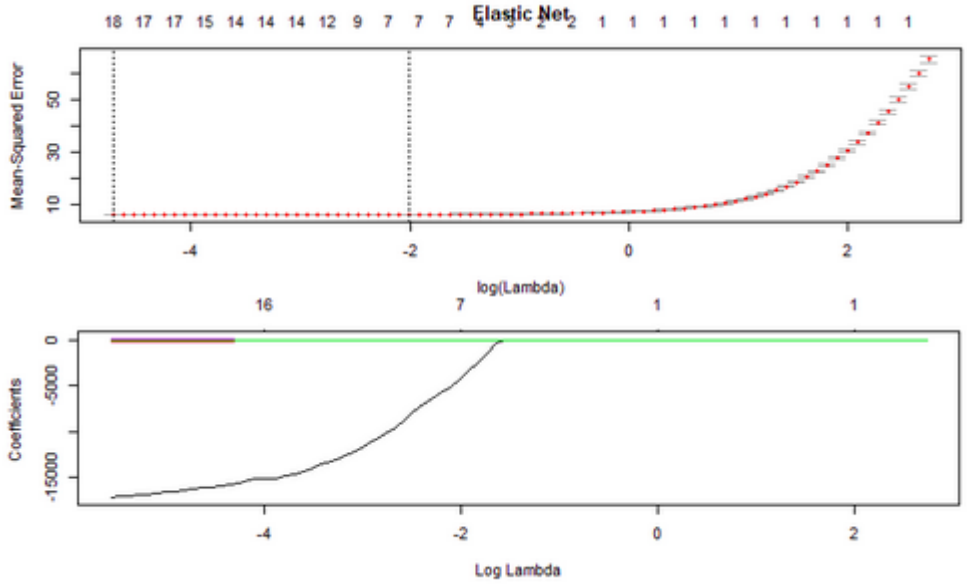


On voit que pour la régression Lasso, contrairement à la régression Ridge, le lambda minimum et le lambda optimale sont assez éloigné.
Cependant, la régression Lasso n'est pas très fiable lorsque certaines variables sont fortement corrélées. C'est pour cela que nous adoptons à présent un modèle Elastic-Net, qui combine les atouts des méthodes Ridge et Lasso et permet de pallier le défault de l'estimation Lasso lorsque les variables sont

fortement corrélées.

2.6 Elastic-Net

Le modèle Elastic-Net nous donne les courbes suivantes :



On reconnaît ici les mêmes formes de courbes que celles obtenus pour la régression Lasso, avec des valeurs de lambda minimum et lambda optimale différentes mais proches.

Comparons à présent l'erreur quadratique moyenne de chacun des 10 modèles obtenus par la méthode du k-fold.

2.7 Choix du modèle

On calcule à présent les erreurs quadratiques moyennes de chacun de nos modèles et on va sélectionner le modèle pour lequel l'erreur quadratique la plus faible. Il s'agira du modèle prédictif le plus fiable et le plus précis.

Le modèle avec l'erreur quadratique moyenne la plus faible dans notre cas est le modèle elastic-net. Les valeurs de notre variable cible se situent entre 20 et 80 et le modèle retenu possède une erreur quadratique de 1.76 environ, ce qui est acceptable.

3 Conclusion

En conclusion, ce projet nous a permis de plus se documenter sur les différents types de régression, leur utilité, leur default ainsi que les différents critères de validation de modèle.

Dans notre cas, dans le but de prédire un score reflétant l'avancement des symptômes de la maladie de Parkinson, nous retenons certaines variables tel que l'âge du sujet et son sexe.
Malheureusement, indépendamment de notre volonté, nous n'avons pu afficher la courbe finale prédictive car nous avons eu des problèmes techniques que nous n'avons pu résoudre.

4 Annexes

4.1 Code R

```r
# Projet MRR

# Lecture de la data
setwd("C:/Users/Walid/Desktop/MRR")
tab <- read.csv(file="parkinsons_updrs.data",header=TRUE, sep=",")

library("lars")
library("corrplot")
library("ridge")
library("MASS")
library("glmnet")
library("caret")


mat_cor <- cor(scale(as.matrix(tab)))
mat_cor

modreg <- lm(motor_UPDRS~.,data=tab)
summary(modreg)
plot(modreg$residuals)

# Regression Backward pour selection des variables explicatives :

regbackward = step(modreg,direction='backward')
summary(regbackward)

# Regression Forward :
modreg2=lm(motor_UPDRS~1,data=tab)

regforward = step(modreg2,list(upper=modreg),direction='forward')
summary(regforward)

AIC(regforward)
AIC(regbackward)
AIC(modreg)

```

```r
# Regression Stepwise :

regboth=step(modreg,direction='both')
summary(regboth)
AIC(regboth)


# On enlève de la tab la variable cible pour la suite.
data = tab
dataY= tab$motor_UPDRS

dim(data)  # 5875


#Séparation aléatoire pour base d'apprentissage et de test
indexes = sample(1:nrow(data), size=0.3*nrow(data))

test = data[indexes,]
dim(test)  # 1762
train = data[-indexes,]
dim(train) # 4113

# Nécessité de mettre les éléments en matrice pour les manipuler dans la suite
Xtest<-test[,-c(5)]
x.test=as.matrix(Xtest)
Ytest=test$motor_UPDRS
y.test=as.matrix(Ytest)



Xtrain<-train[,-c(5)]
x.train=as.matrix(Xtrain)
Ytrain=train$motor_UPDRS
y.train=as.matrix(Ytrain)
nrow(y.train)
nrow(Xtrain)

# Nos modèles particuliers en fonction de la valeur de alpha
fit.lasso <- glmnet(x.train, y.train, family="gaussian", alpha=1)
fit.ridge <- glmnet(x.train, y.train, family="gaussian", alpha=0)
fit.elnet <- glmnet(x.train, y.train, family="gaussian", alpha=.5)
```

```
78
79
80
81  # k-fold avec k=10 pour le critère de validation croisée.
82  for (i in 0:10) {
83    assign(paste("fit", i, sep=""), cv.glmnet(x.train, y.train, type.measure="mse",
84                                              alpha=i/10,family="gaussian"))
85  }
86  # Affichage des plot des coefficients et des erreurs quadratiques moyennes
87  # en fonction de log(lambda).
88  par(mfrow=c(3,2))
89
90  plot(fit.lasso, xvar="lambda")
91  plot(fit10, main="LASSO")
92
93  plot(fit.ridge, xvar="lambda")
94  plot(fit0, main="Ridge")
95
96  plot(fit.elnet, xvar="lambda")
97  plot(fit5, main="Elastic Net")
98
99
100 # Calcul de l'erreur quadratique moyenne pour chacun de nos modèles, permettant
101 # un choix de modèle selon le critère de validation croisée.
102
103 yhat0 <- predict(fit0, s=fit0$lambda.1se, newx=x.test)
104 yhat1 <- predict(fit1, s=fit1$lambda.1se, newx=x.test)
105 yhat2 <- predict(fit2, s=fit2$lambda.1se, newx=x.test)
106 yhat3 <- predict(fit3, s=fit3$lambda.1se, newx=x.test)
107 yhat4 <- predict(fit4, s=fit4$lambda.1se, newx=x.test)
108 yhat5 <- predict(fit5, s=fit5$lambda.1se, newx=x.test)
109 yhat6 <- predict(fit6, s=fit6$lambda.1se, newx=x.test)
110 yhat7 <- predict(fit7, s=fit7$lambda.1se, newx=x.test)
111 yhat8 <- predict(fit8, s=fit8$lambda.1se, newx=x.test)
112 yhat9 <- predict(fit9, s=fit9$lambda.1se, newx=x.test)
113 yhat10 <- predict(fit10, s=fit10$lambda.1se, newx=x.test)
114
115 mse0 <- mean((y.test - yhat0)^2)
116 mse1 <- mean((y.test - yhat1)^2)
117 mse2 <- mean((y.test - yhat2)^2)
118 mse3 <- mean((y.test - yhat3)^2)
```

```
mse4 <- mean((y.test - yhat4)^2)
mse5 <- mean((y.test - yhat5)^2)
mse6 <- mean((y.test - yhat6)^2)
mse7 <- mean((y.test - yhat7)^2)
mse8 <- mean((y.test - yhat8)^2)
mse9 <- mean((y.test - yhat9)^2)
mse10 <- mean((y.test - yhat10)^2)

mse0
mse1
mse2
mse3
mse4
mse5
mse6
mse7
mse8
mse9
mse10
```

4.2 Graphiques

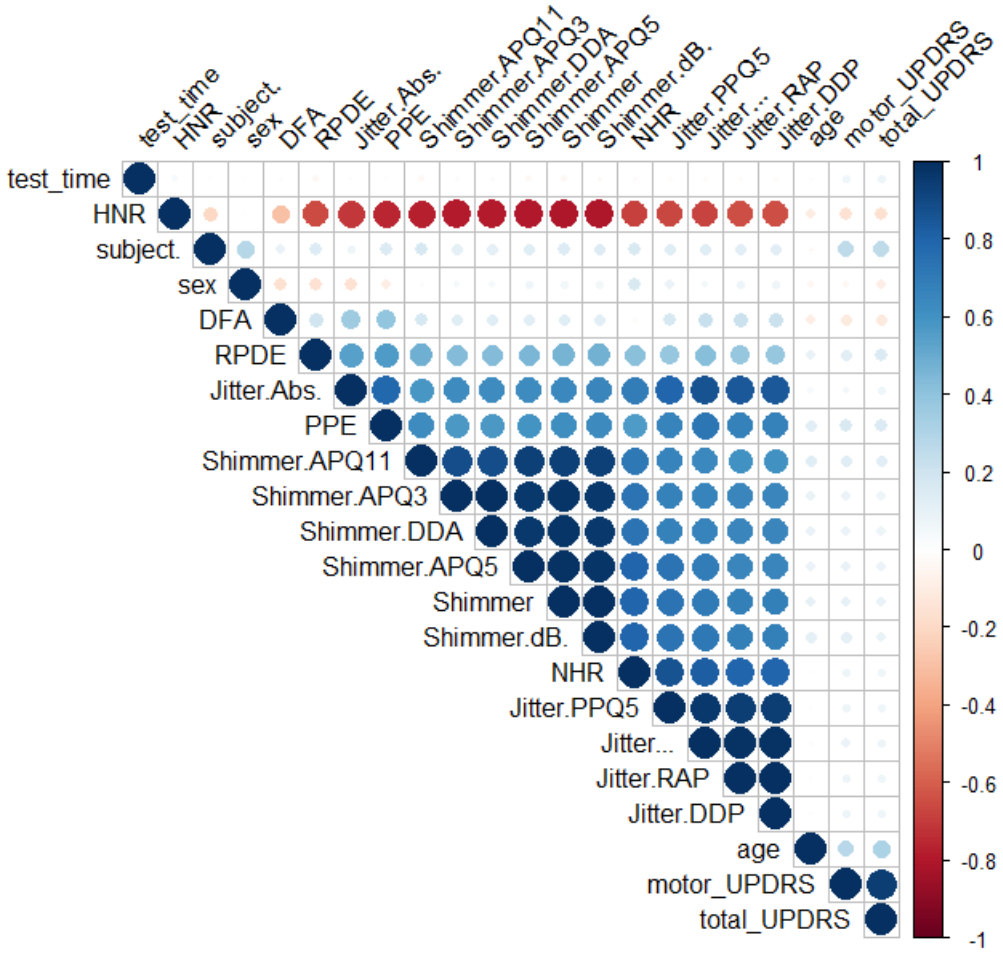


FIGURE 1 – Matrice de corrélation

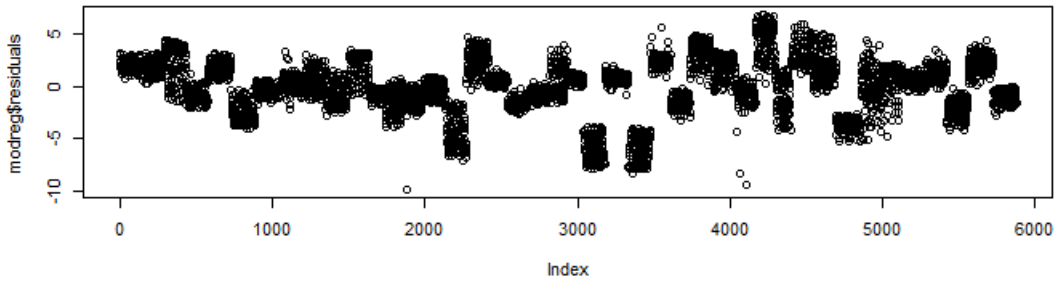


FIGURE 2 – Graphique des résidus

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.499 on 5853 degrees of freedom
Multiple R-squared:  0.9058,    Adjusted R-squared:  0.9055
F-statistic:  2680 on 21 and 5853 DF,  p-value: < 2.2e-16
```

FIGURE 3 – Résultats pour le modèle linéaire simple