

# PROJET INFO0806 - RÉGRESSION LINÉAIRE MULTIPLE SUR LE JEU DE DONNÉES << CPUS >>

réalisé par LIEPO BRICE - KEVIN & ROA SERRANO WALTER

12 mai, 2020

## Contents

<b>INTRODUCTION</b>	<b>3</b>
<b>ENVIRONNEMENT DE TRAVAIL</b>	<b>3</b>
<b>LANGAGE R</b> . . . . .	3
<b>R STUDIO</b> . . . . .	3
<b>MARKDOWN</b> . . . . .	3
<b>PACKAGES UTILISÉS</b> . . . . .	3
<b>I. PRÉSENTATION ET PRÉPARATION DES DONNÉES</b>	<b>4</b>
<b>PRÉPARATION DES DONNÉES</b> . . . . .	4
<b>PRÉSENTATION DES DONNÉES</b> . . . . .	4
Resume jeu de données . . . . .	4
Table des premiers individus . . . . .	5
Suppression de la variable “estperf” . . . . .	5
<b>CHOIX DE VARIABLES</b> . . . . .	5
Table des individus et graphe de corrélation . . . . .	5
<b>II. RÉGRESSION LINÉAIRE MULTIPLE</b>	<b>6</b>
Régression linéaire avec toutes les variables et critère d’AIC . . . . .	6
Régression linéaire avec toutes les variables et critère d’AIC sans la variable “chmin” . . . . .	7
Régression linéaire avec toutes les variables et critère d’AIC sans la variable “syct” . . . . .	8
Résumé et Coefficient du model calibré . . . . .	9
<b>ÉTUDE DES RÉSIDUS</b> . . . . .	9
Graphe de représentation de l’indépendance ou l’autocorrélation des résidus . . . . .	12
<b>PRÉDICTION</b> . . . . .	13
Graph des valeurs prédites . . . . .	13
Intervale de confiance sur la valeur prédite . . . . .	14

<b>III. TESTS STATISTIQUES</b>	<b>14</b>
TEST D'ADÉQUATION OU DE LINÉARITÉ . . . . .	15
TEST DE NORMALITÉ DE SHAPIRO . . . . .	15
TEST D'HOMOSCÉDASTICITÉ DES VARIANCES DE GOLDFELD-QUANDT . . .	15
TESTS D'INDÉPENDANCE ET D'AUTOCORELATION DES RÉSIDUS DE DURBIN WATSON . . . . .	16
TEST DE COLINÉARITÉ FORTE DU MODELE . . . . .	16
<b>CONCLUSION</b>	<b>16</b>

# INTRODUCTION

En statistiques, un modèle de régression linéaire est un modèle de régression qui cherche à établir une relation linéaire entre une variable, dite expliquée, et une ou plusieurs variables, dites explicatives.

La régression linéaire multiple est, quant à elle, une méthode de régression mathématique étendant la régression linéaire simple pour décrire les variations d’une variable endogène (expliquée) associée aux variations de plusieurs variables exogènes (explicatives). Par exemple, une analyse de régression multiple peut révéler une relation positive entre la demande de lunettes de soleil et différents caractères démographiques (âge, salaire) des acheteurs de ce produit. La demande augmente et baisse avec les variations de ces caractéristiques.

Dans le cadre du module info0806, nous sommes amenés à réaliser une régression linéaire sur un jeu de données du nom de “cpus”, afin de trouver le modèle de régression adéquat.

## ENVIRONNEMENT DE TRAVAIL

### LANGAGE R

Dans le cadre de ce projet nous avons travaillé avec R qui est un langage de programmation open source destiné aux statistiques et à la science des données. Ce langage est largement utilisé par les statisticiens, les data miners, data scientists pour l’analyse de données et la représentation graphique de ces analyses. R dispose d’un très grand nombre de bibliothèques développées par une communauté de contributeurs qui permettent des diverses analyses et d’une grande variété de possibilités pour la représentation graphique.

### R STUDIO

RStudio est un environnement de développement intégré multiplateforme, qui permet le développement d’applications sous R. Il dispose de nombreux outils qui vous permettent le traitement de données et l’analyse statistique comme la création des scripts, la création des graphes et la compilation du code.

### MARKDOWN

Markdown est un langage de balisage léger, une syntaxe facile qui permet de mélanger du code R avec du text pour générer des documents au format HTML, PDF, Word et d’autres. On va l’utiliser pour l’exportation des résultats d’analyse en PDF.

### PACKAGES UTILISÉS

- **library(MASS)** : Contient des fonctions et jeux de données pour prendre en charge Venables et Ripley. Notre jeu de données “cpus” appartient à cette librairie.
- **library(knitr)** : Fournit un outil polyvalent pour la génération de rapports dynamiques en R à l’aide de techniques de programmation littéraire.
- **library(lmtest)** : Une collection de tests, d’ensembles de données et d’exemples pour la vérification diagnostique dans les modèles de régression linéaire. En outre, certains outils génériques pour l’inférence dans les modèles paramétriques sont fournis.
- **library(car)** : Utilisée pour appliquer une régression linéaire et des tests statistiques
- **library(tidyverse)** : C’est une collection d’extensions qui concernent à la data science et qui permet la manipulation des tableaux, la manipulation de variables et la visualisation de données.

# I. PRÉSENTATION ET PRÉPARATION DES DONNÉES

## PRÉPARATION DES DONNÉES

Comme annoncé précédemment, nous allons utiliser le jeu de données “cpus”, présent dans un des packages R, du nom de MASS. Nous allons effectuer la récupération de ces données et leur mise en forme afin de les utiliser.

```
data(cpus)
cpus <- data.frame(cpus, row.names="name")
attach(cpus)
```

## PRÉSENTATION DES DONNÉES

Ce jeu de données resume les performances relatives et les caractéristiques de 209 processeurs. Les différentes variables sont les suivantes :

- syct : temps de cycle en nanosecondes;
- mmin : mémoire principale minimale en kilo-octets;
- mmax : mémoire principale maximale en kilo-octets;
- cach : taille du cache en kilo-octets;
- chmin : nombre minimum de canaux;
- chmax : nombre maximum de canaux;
- perf : performances publiées sur un mix de référence par rapport à un IBM 370 / 158-3;
- estperf : performances estimées (par Ein-Dor & Feldmesser).

Le contenu de ces colonnes se resume comme suit :

### Resume jeu de données

```
summary(cpus)
```

##	syct	mmin	mmax	cach
##	Min. : 17.0	Min. : 64	Min. : 64	Min. : 0.00
##	1st Qu.: 50.0	1st Qu.: 768	1st Qu.: 4000	1st Qu.: 0.00
##	Median : 110.0	Median : 2000	Median : 8000	Median : 8.00
##	Mean : 203.8	Mean : 2868	Mean : 11796	Mean : 25.21
##	3rd Qu.: 225.0	3rd Qu.: 4000	3rd Qu.: 16000	3rd Qu.: 32.00
##	Max. : 1500.0	Max. : 32000	Max. : 64000	Max. : 256.00
##	chmin	chmax	perf	estperf
##	Min. : 0.000	Min. : 0.00	Min. : 6.0	Min. : 15.00
##	1st Qu.: 1.000	1st Qu.: 5.00	1st Qu.: 27.0	1st Qu.: 28.00
##	Median : 2.000	Median : 8.00	Median : 50.0	Median : 45.00
##	Mean : 4.699	Mean : 18.27	Mean : 105.6	Mean : 99.33
##	3rd Qu.: 6.000	3rd Qu.: 24.00	3rd Qu.: 113.0	3rd Qu.: 101.00
##	Max. : 52.000	Max. : 176.00	Max. : 1150.0	Max. : 1238.00

les dix premiers individus sont les suivants :

## Table des premiers individus

```
kable(head(cpus,10))
```

	syst	mmin	mmax	cach	chmin	chmax	perf	estperf
ADVISOR 32/60	125	256	6000	256	16	128	198	199
AMDAHL 470V/7	29	8000	32000	32	8	32	269	253
AMDAHL 470/7A	29	8000	32000	32	8	32	220	253
AMDAHL 470V/7B	29	8000	32000	32	8	32	172	253
AMDAHL 470V/7C	29	8000	16000	32	8	16	132	132
AMDAHL 470V/8	26	8000	32000	64	8	32	318	290
AMDAHL 580-5840	23	16000	32000	64	16	32	367	381
AMDAHL 580-5850	23	16000	32000	64	16	32	489	381
AMDAHL 580-5860	23	16000	64000	64	16	32	636	749
AMDAHL 580 5880	23	32000	64000	128	32	64	1144	1238

Nous allons supprimer la dernière variable de notre jeu de données car, après recherche, nous avons constaté que cette variable représente un critère de prédiction déterminé par le travail de Ein-Dor & Feldmesser.

## Suppression de la variable “estperf”

```
cpus$estperf <- NULL
```

## CHOIX DE VARIABLES

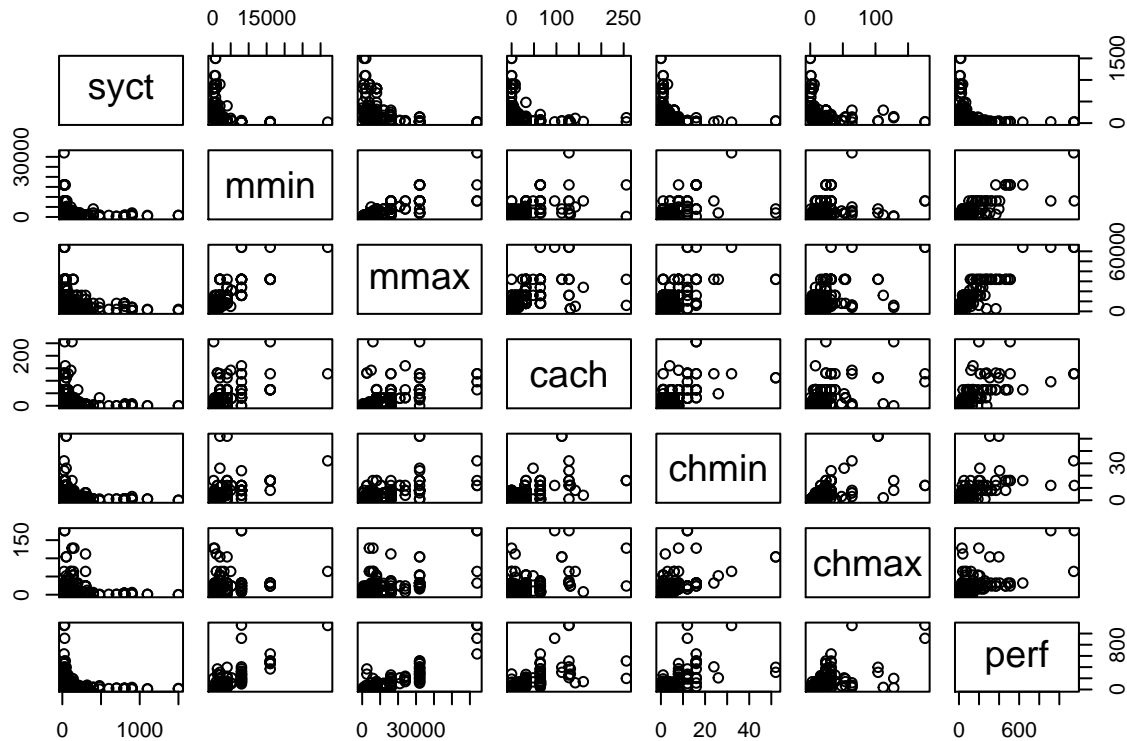
Pour notre régression linéaire multiple nous avons choisi la variable “perf” comme variable expliquée. En effet nous souhaitons expliquer la performance des processeurs en fonction des autres variables. Le tableau et le graphe de corrélation suivants nous montreront que, la variable “perf” est la plus corrélée aux autres variables :

## Table des individus et graphe de corrélation

```
kable(cor(cpus))
```

	syst	mmin	mmax	cach	chmin	chmax	perf
syst	1.0000000	-0.3356422	-0.3785606	-0.3209998	-0.3010897	-0.2505023	-0.3070821
mmin	-0.3356422	1.0000000	0.7581573	0.5347291	0.5171892	0.2669074	0.7949233
mmax	-0.3785606	0.7581573	1.0000000	0.5379898	0.5605134	0.5272462	0.8629942
cach	-0.3209998	0.5347291	0.5379898	1.0000000	0.5822455	0.4878458	0.6626135
chmin	-0.3010897	0.5171892	0.5605134	0.5822455	1.0000000	0.5482812	0.6089025
chmax	-0.2505023	0.2669074	0.5272462	0.4878458	0.5482812	1.0000000	0.6052193
perf	-0.3070821	0.7949233	0.8629942	0.6626135	0.6089025	0.6052193	1.0000000

```
plot(cpus)
```



## II. RÉGRESSION LINÉAIRE MULTIPLE

Nous allons utiliser une méthode descendante en se servant de la fonction `lm()`, comme en régression linéaire simple. Dans la fonction `lm`, le point, placé après le symbole “~”, indique qu’on souhaite régresser “perf” sur toutes les autres variables du jeu de données.

Le critère utilisé par défaut dans R est le critère AIC (pour “An Information Criterion”, proposé par Akaike). La fonction “`extractAIC`” permet de minimiser ce critère.

La fonction `summary()` permet de produire les sorties pour notre régression.

### Régression linéaire avec toutes les variables et critère d’AIC

```
summary(perf.lm <- lm(formula=perf~.,data=cpus))
```

```
##
## Call:
## lm(formula = perf ~ ., data = cpus)
##
## Residuals:
```

```
##      Min      1Q  Median      3Q      Max
## -195.84 -25.17   5.41   26.53  385.75
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.590e+01  8.045e+00 -6.948 4.99e-11 ***
## syct         4.886e-02  1.752e-02   2.789  0.00579 **
## mmin         1.529e-02  1.827e-03   8.371 9.42e-15 ***
## mmax         5.571e-03  6.418e-04   8.680 1.33e-15 ***
## cach         6.412e-01  1.396e-01   4.594 7.64e-06 ***
## chmin        -2.701e-01  8.557e-01  -0.316  0.75263
## chmax         1.483e+00  2.201e-01   6.738 1.64e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 59.99 on 202 degrees of freedom
## Multiple R-squared:  0.8649, Adjusted R-squared:  0.8609
## F-statistic: 215.5 on 6 and 202 DF,  p-value: < 2.2e-16
```

```
extractAIC(perf.lm)
```

```
## [1]      7.000 1718.259
```

Les informations données par la fonction `summary()` concernent :

- Les résidus (maximum, minimum, moyenne, quartiles);
- Les coefficients estimés : Les estimations (Estimate), l'écart-type estimé des estimateurs correspondants (Std. Error), la valeur de la statistique de test (t value) et la p-value ( $\text{Pr}(>|t|)$ ) associées aux tests (probabilité que le coefficient soit significativement différent de zéro);
- La qualité d'adéquation du modèle : une estimation de l'écart-type du terme d'erreur (Residual standard error), la valeur du R2 (Multiple R-squared) et celle du R2 ajusté (Adjusted R-squared), et enfin la valeur de la statistique de test (F-statistic : testant la significativité globales des variables), son degré de liberté et la p-value du test de Fisher de significativité du modèle.

On soustrait à présent la variable “chmin” car son coefficient n'est pas significativement différent de zéro et sa p-valeur est supérieur à 0.3.

### Régression linéaire avec toutes les variables et critère d'AIC sans la variable “chmin”

```
summary(perf.lm <- update(perf.lm,.-chmin))
```

```
##
## Call:
## lm(formula = perf ~ syct + mmin + mmax + cach + chmax, data = cpus)
##
## Residuals:
##      Min      1Q  Median      3Q      Max
## -193.39 -24.94   5.77   26.65  389.66
##
## Coefficients:
```

```
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.608e+01  8.007e+00 -7.004 3.58e-11 ***
## syct        4.912e-02  1.746e-02  2.813  0.00539 **
## mmin        1.518e-02  1.788e-03  8.490 4.34e-15 ***
## mmax        5.561e-03  6.397e-04  8.694 1.18e-15 ***
## cach        6.296e-01  1.344e-01  4.686 5.11e-06 ***
## chmax       1.460e+00  2.076e-01  7.032 3.05e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 59.86 on 203 degrees of freedom
## Multiple R-squared:  0.8648, Adjusted R-squared:  0.8615
## F-statistic: 259.7 on 5 and 203 DF,  p-value: < 2.2e-16
```

```
extractAIC(perf.lm)
```

```
## [1]      6.000 1716.362
```

Nous constatons que l'AIC et l'écart type ont légèrement diminué. Ce qui signifie que ce modèle est plus efficace que le précédent. Nous conservons donc ce model pour l'instant.

Nous allons retirer ensuite la variable dont le coefficient est le moins significatif, à savoir la variable “syct”.

### Régression linéaire avec toutes les variables et critère d'AIC sans la variable “syct”

```
summary(perf.lm<-update(perf.lm,~.-syct))
```

```
##
## Call:
## lm(formula = perf ~ mmin + mmax + cach + chmax, data = cpus)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -186.73  -26.08    8.27   26.99  402.87
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept) -40.982864   6.041898  -6.783 1.25e-10 ***
## mmin         0.014887   0.001815   8.202 2.61e-14 ***
## mmax         0.005330   0.000645   8.263 1.79e-14 ***
## cach        0.587097   0.135764   4.324 2.39e-05 ***
## chmax       1.436179   0.210954   6.808 1.08e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60.86 on 204 degrees of freedom
## Multiple R-squared:  0.8595, Adjusted R-squared:  0.8568
## F-statistic: 312.1 on 4 and 204 DF,  p-value: < 2.2e-16
```



```
extractAIC(perf.lm)
```

```
## [1] 5.000 1722.354
```

Nous constatons que l'AIC a augmenté et l'écart-type estimé des résidus est passé de 59.86 à 60.86. e qui signifie que ce modèle est moins efficace que le précédent. Nous allons par conséquent conserver le modèle contenant la variable "syct" et afficher par la meme occasion les coefficient de la droite.

## Résumé et Coefficient du model calibré

```
summary(perf.lm<-lm(formula = perf ~ syct + mmin + mmax + cach + chmax, data = cpus))
```

```
##
## Call:
## lm(formula = perf ~ syct + mmin + mmax + cach + chmax, data = cpus)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -193.39  -24.94    5.77   26.65  389.66
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.608e+01  8.007e+00  -7.004 3.58e-11 ***
## syct         4.912e-02  1.746e-02   2.813 0.00539 **
## mmin         1.518e-02  1.788e-03   8.490 4.34e-15 ***
## mmax         5.561e-03  6.397e-04   8.694 1.18e-15 ***
## cach         6.296e-01  1.344e-01   4.686 5.11e-06 ***
## chmax        1.460e+00  2.076e-01   7.032 3.05e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 59.86 on 203 degrees of freedom
## Multiple R-squared:  0.8648, Adjusted R-squared:  0.8615
## F-statistic: 259.7 on 5 and 203 DF, p-value: < 2.2e-16
```

```
coef(perf.lm)
```

```
##      (Intercept)          syct          mmin          mmax          cach
## -56.081005071    0.049121395    0.015181451    0.005561306    0.629642098
##           chmax
##    1.460122823
```

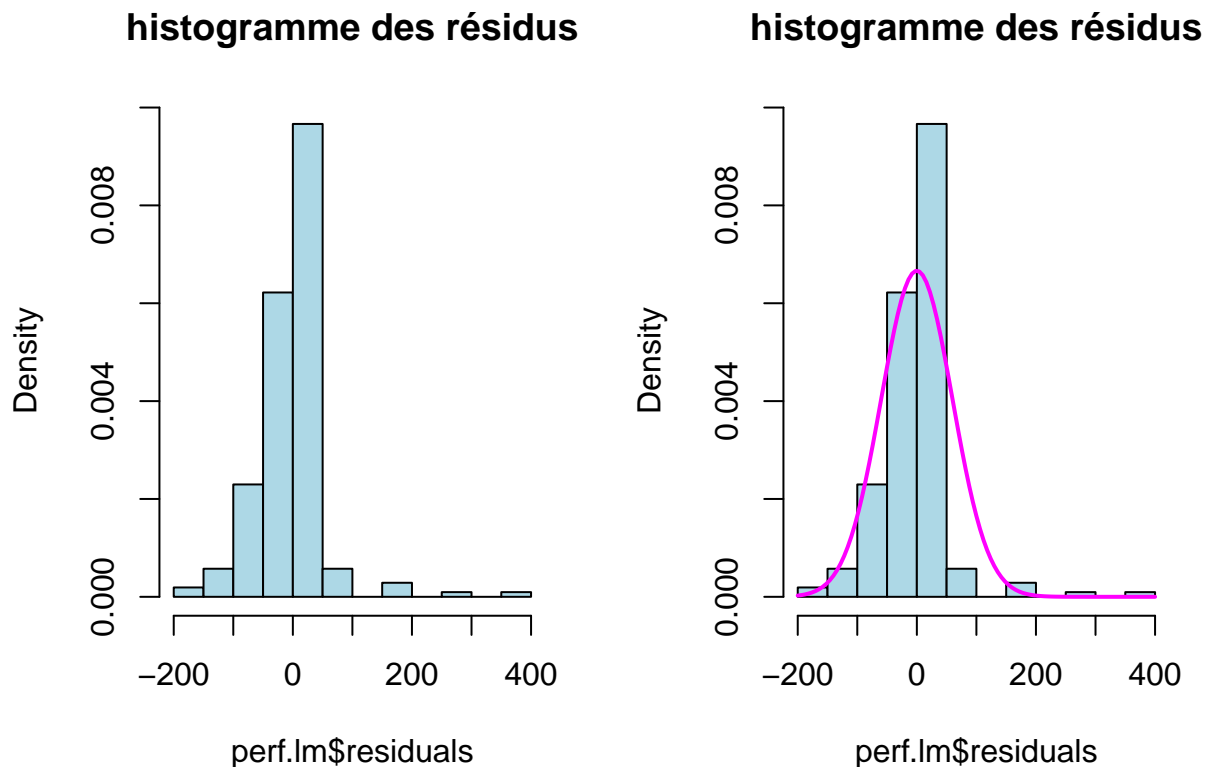
## ÉTUDE DES RÉSIDUS

Nous allons récupérer et afficher sous forme d'histogramme et de graphes l'ensemble des résidus, afin d'observer si les conditions d'indépendance, d'homoscédasticité (même variance) et de normalité sont respectés. Mais aussi vérifier que ces residus peuvent suivre la loi normale.

Les représentations sont, respectivement, les suivantes :

- Un histogramme des résidus;
- Un histogramme des résidus avec une courbe pour vérifier si ces résidus peuvent suivre la loi normale;

```
par(mfrow=c(1,2))
hist(perf.lm$residuals,freq=FALSE,nclass=10, col="lightblue",main="histogramme des résidus")
histo <- hist(perf.lm$residuals, col="lightblue", main="histogramme des résidus", probability = TRUE)
ec_typ <- summary(perf.lm)$sigma
curve(dnorm(x, 0, ec_typ), from = min(histo$breaks), to = max(histo$breaks),
      add = TRUE, type = "l", col = "magenta", lwd = 2)
```



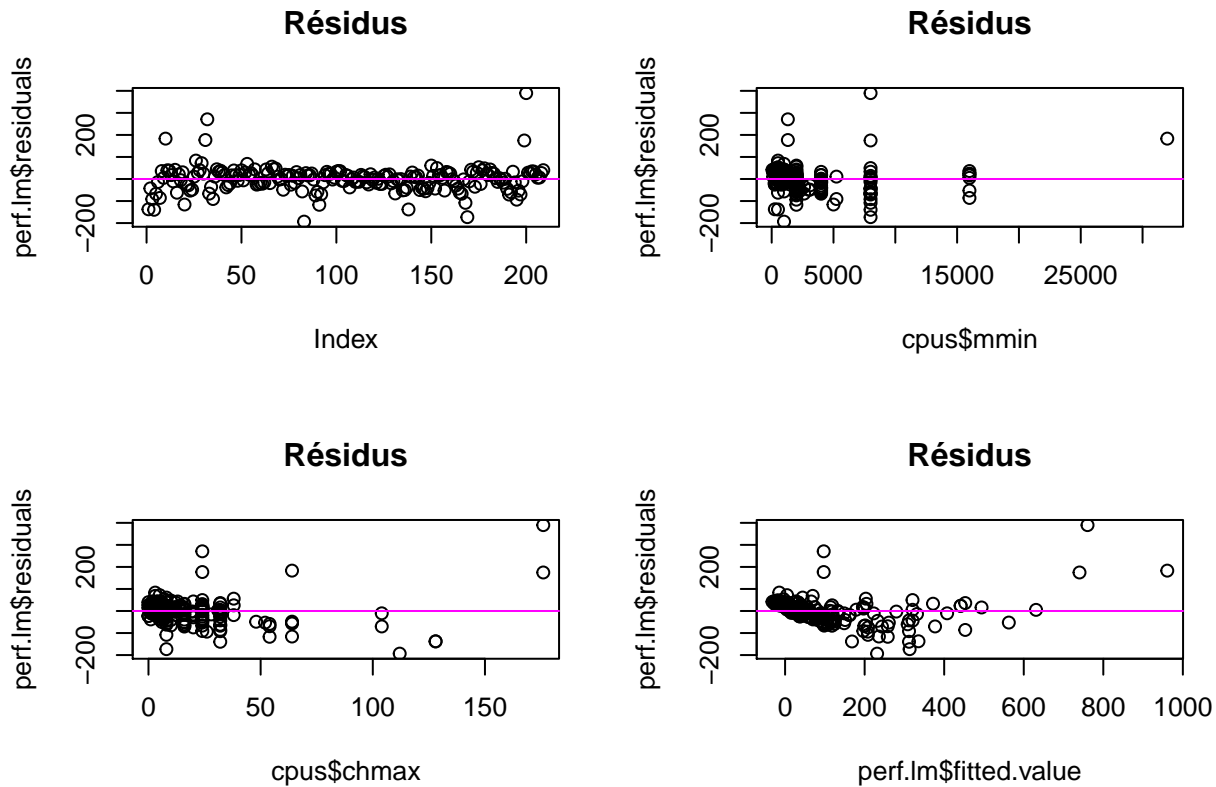
- La répartition des résidus;
- La répartition des résidus par rapport à la mémoire principale minimale en kilo-octets (mmin);
- La répartition des résidus par rapport au nombre maximum de canaux (chmax);
- La répartition des résidus par rapport aux valeurs prédites (fitted value);

```
par(mfrow=c(2,2))
plot(perf.lm$residuals,main="Résidus")
abline(h=0,col="magenta")

plot(cpus$mmin,perf.lm$residuals,main="Résidus")
abline(h=0,col="magenta")

plot(cpus$chmax,perf.lm$residuals,main="Résidus")
abline(h=0,col="magenta")
```

```
plot(perf.lm$fitted.value,perf.lm$residuals,main="Résidus")
abline(h=0,col="magenta")
```



Nous constatons dans un premier temps, grace aux deux premier histogramme, que les résidus suivent bel et bien la loi normale.

Nous remarquons aussi que les résidus sont repartis, de manière relativement indépendantes, de part et d'autre de la droite d'équation  $y=0$ .

Si les résidus n'était pas repartis de manière relativement indépendantes, de part-et d'autre de la droite d'équation  $y=0$ , il y aurait eu un problème (corrélations, hétéroscédasticité, ...) que l'on allait devoir corriger avant de pouvoir interpréter un résultat.

De plus, nous remarquons une diminution des résidus (négatifs) lorsque la mémoire principale minimale en kilo-octets (mmin), le nombre maximum de canaux (chmax), ou la valeur prédite augmentent.

Pour finir nous allons afficher :

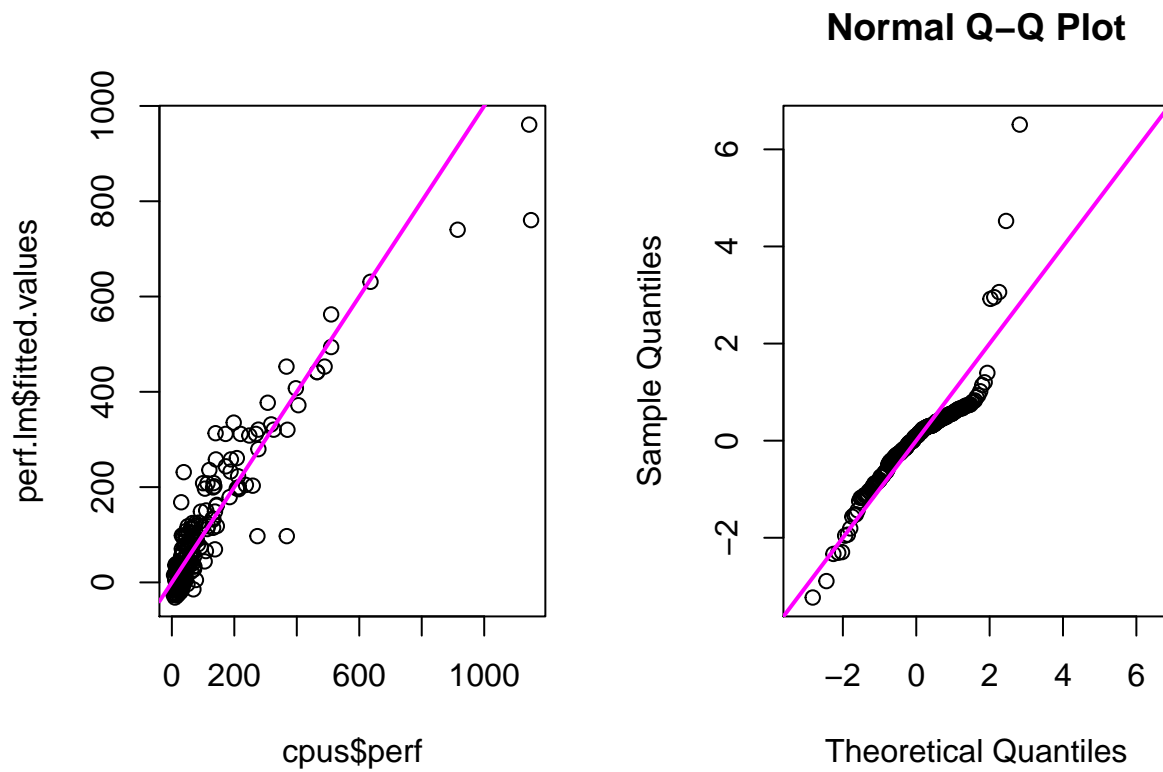
- La repartition des valeurs prédites (en ordonnée) et des valeurs de "perf" (en abscisse);
- La repartition des quantiles.

```
par(mfrow=c(1,2))

plot(cpus$perf, perf.lm$fitted.values)
abline(0, 1, col = "magenta", lwd = 2)

ec_typ <- summary(perf.lm)$sigma
```

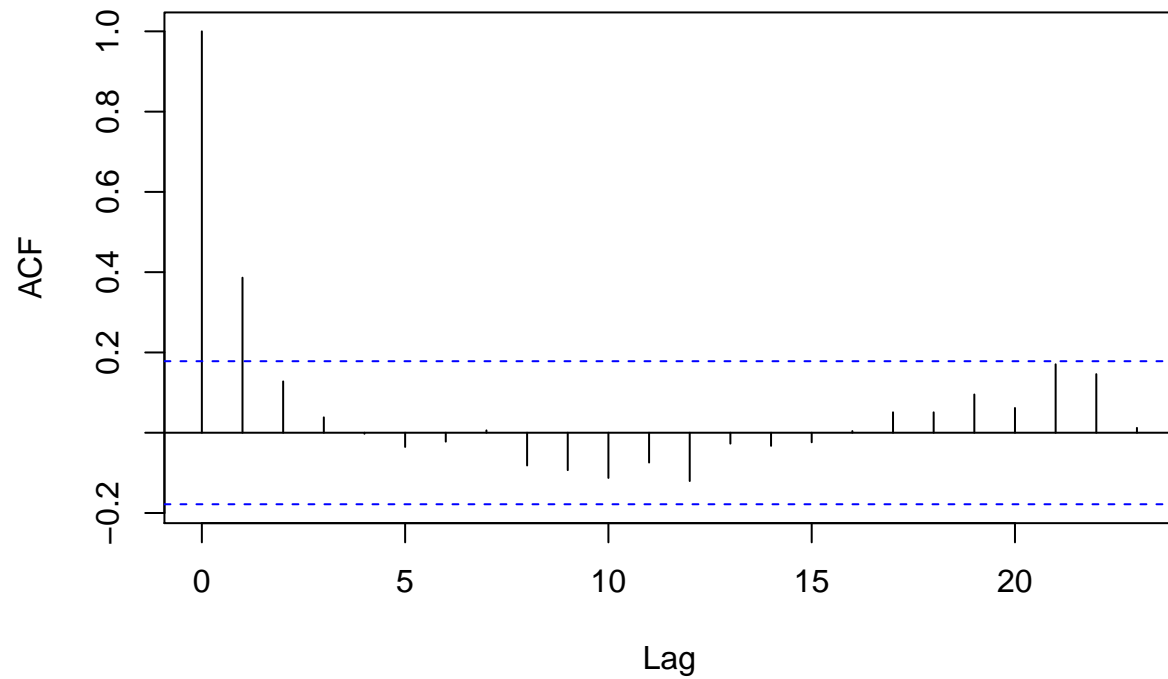
```
normed_res <- perf.lm$residuals/ec_typ
qqnorm(normed_res, xlim = range(normed_res), ylim = range(normed_res))
abline(0, 1, col = "magenta", lwd = 2)
```



Graphique de représentation de l'indépendance ou l'autocorrélation des résidus

```
acf(perf.lm$residuals, ci=0.99)
```

### Series perf.lm\$residuals

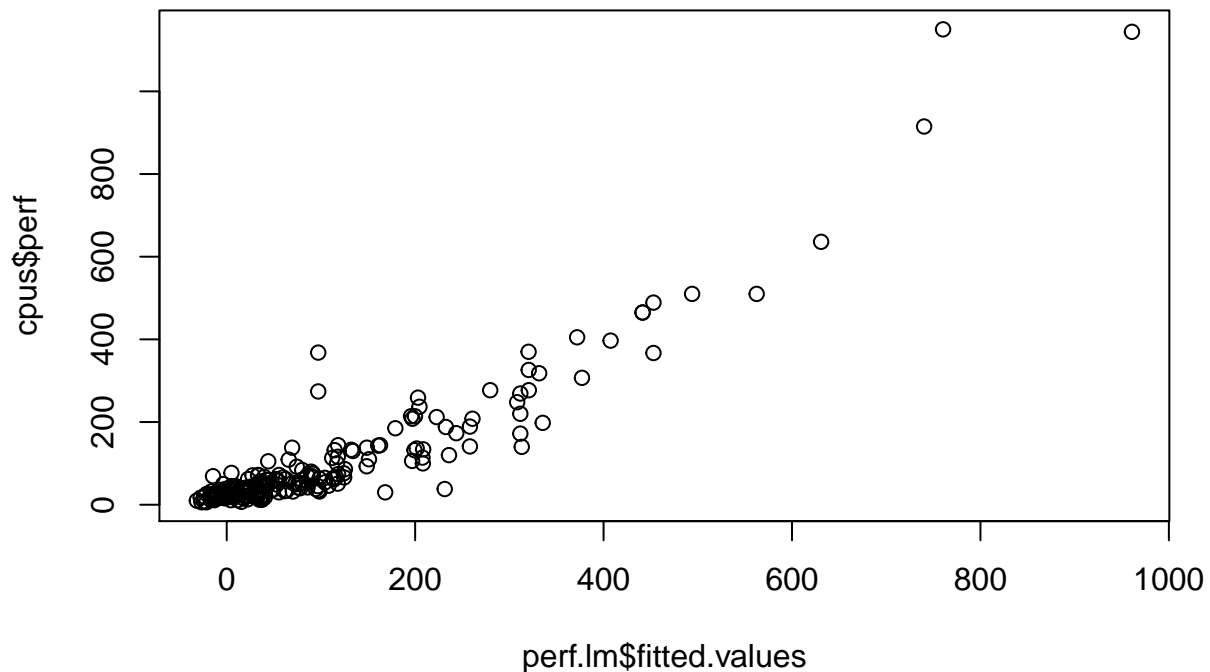


## PRÉDICTION

Nous allons afficher les valeurs prédites par rapport aux valeurs normales:

Graph des valeurs prédites

```
plot(perf.lm$fitted.values, cpus$perf)
```



Partons du principe que nous avons un nouvel individu. L'intervalle de confiance sur la valeur prédite est donné par l'instruction suivante :

**Intervale de confiance sur la valeur prédite**

```
predict(perf.lm,data.frame(syct=125, mmin=256, mmax=6000, cach=256, chmax=128),interval="prediction",lev
```

```
##          fit      lwr      upr
## 1 335.3976 200.4104 470.3847
```

Par conséquent, si la performance (perf) est comprise dans l'intervalle [191.76; 466.11], il n'y a pas de contradiction avec le modèle.

### III. TESTS STATISTIQUES

Nous allons réaliser des tests statistiques afin de valider les hypothèses :

La p-value est souvent utilisée dans les tests d'hypothèses, ce test nous permet de rejeter, ou non, une hypothèse nulle. Elle représente la probabilité de faire une erreur de type 1 ou « faux positif », ou de rejeter l'hypothèse nulle si elle est vrai.

Plus la valeur p-value est petite, plus la probabilité de faire une erreur en rejetant l'hypothèse nulle est faible. Une valeur limite de 0,05 est souvent utilisée. Autrement dit, vous pouvez rejeter l'hypothèse nulle si la p-valeur est inférieure à 0,05.

## TEST D'ADÉQUATION OU DE LINÉARITÉ

Pour conclure à la non-linéarité du modèle de régression, on préconise le test de Rainbow : si p-valeur  $< 0.05$ , on rejette la linéarité du modèle.

```
raintest(perf.lm)
```

```
##  
## Rainbow test  
##  
## data: perf.lm  
## Rain = 7.6764, df1 = 105, df2 = 98, p-value < 2.2e-16
```

On a obtenu une p-value  $< 2.2e-16$  donc on peut rejeter l'hypothèse nulle.

Le résultat donne une p-valeur  $< 0.05$  donc, nous pouvons en conclure qu'un modèle de régression non-linéaire est plus adapté aux données.

## TEST DE NORMALITÉ DE SHAPIRO

En statistique, le test de Shapiro-Wilk teste l'hypothèse nulle selon laquelle un échantillon est issu d'une population normalement distribuée. En d'autre terme, le test de Shapiro-Wilk permet de savoir si une série de données suit une loi normale. Il a été publié en 1965 par Samuel Sanford Shapiro et Martin.

```
shapiro.test(perf.lm$residuals)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: perf.lm$residuals  
## W = 0.83008, p-value = 2.295e-14
```

Ci-dessus, la p-value est significative, donc les résidus ne suivent pas une loi normale.

## TEST D'HOMOSCÉDASTICITÉ DES VARIANCES DE GOLDFELD-QUANDT

Le test de Goldfeld et Quandt (formulé en 1965) est un test statistique, très utilisé en économétrie dans le cadre d'un modèle linéaire multiple estimé par la méthode des moindres carrés afin de savoir si les perturbations sont hétéroscédastiques ou homoscedastiques. Ce test s'appuie sur la loi de Fisher.

```
gqtest(perf.lm)
```

```
##  
## Goldfeld-Quandt test  
##  
## data: perf.lm  
## GQ = 1.3787, df1 = 99, df2 = 98, p-value = 0.05642  
## alternative hypothesis: variance increases from segment 1 to 2
```

L'application du test de Goldfeld-Quandt permet de vérifier l'homogénéité des variances des résidus.

Les résultats donnent une p-value  $> 0,05$  par conséquent l'homogénéité des résidus est respectée.

## TESTS D'INDÉPENDANCE ET D'AUTOCORRELATION DES RÉSIDUS DE DURBIN WATSON

Le test de Durbin-Watson est un test statistique destiné à tester l'autocorrélation des résidus dans un modèle de régression linéaire. Il a été proposé en 1950 et 1951 par James Durbin et Geoffrey Watson.

Il permet de détecter une auto-corrélation des erreurs d'ordre un et il repose sur l'estimation d'un modèle autorégressif de premier ordre pour les résidus estimés.

```
dwtest(perf.lm)
```

```
##
## Durbin-Watson test
##
## data: perf.lm
## DW = 1.1993, p-value = 1.126e-09
## alternative hypothesis: true autocorrelation is greater than 0
```

Dans l'application du test de Durbin-Watson si la p-value est  $> 0,05$  l'indépendance des résidus est respectée. Nos résultats donnent une p-value de  $1.126e-09$  qui est beaucoup plus inférieur à  $0,05$ . Par conséquent l'indépendance des résidus est rejetée.

Nous constatons enfin qu'il y a colinéarité forte dans le modèle, c'est-à-dire que les variables explicatives sont linéairement dépendantes.

## TEST DE COLINÉARITÉ FORTE DU MODELE

Dans une régression, la multicollinéarité est un problème qui survient lorsque certaines variables de prévision du modèle mesurent le même phénomène.

On dit que des variables sont liées s'il existe une relation linéaire entre elles. Une erreur fréquente est de confondre multicollinéarité et corrélation. Si des variables colinéaires sont de facto fortement corrélées entre elles, deux variables corrélées ne sont pas forcément colinéaires. En termes non statistiques, il y a colinéarité lorsque deux ou plusieurs variables mesurent la même chose.

Pour étudier la colinéarité des variables explicatives l'approche la plus classique consiste à examiner les facteurs d'inflation de la variance (FIV) ou variance inflation factor (VIF). La fonction `vif()` de la librairie `car` permet de comparer la colinéarité des variables. Les valeurs renvoyées par `vif` doivent être basses sinon les variables ont tendance à évoluer dans la même direction.

```
vif(perf.lm)
```

```
##      syct      mmin      mmax      cach      chmax
## 1.199030 2.792328 3.266378 1.730247 1.691608
```

Dans nos résultats, les valeurs des VIF (Variance Inflation Factors) des variables explicatives sont inférieures à 10 par conséquent les variables ne présentent pas de forte colinéarité.

## CONCLUSION

Pour conclure, ce projet, réalisé dans le cadre du module info0806, qui avait pour objectif de réaliser une régression linéaire sur un jeu de données de notre choix nous a permis de :



- Déterminer l'impact des variables explicatives sur les prédictions de la variable expliquée;
- Appliquer les différentes fonctions relatives à la réalisation d'une régression linéaire multiple;
- Comprendre les différents paramètres associés au résultat des différentes fonctions.