TP2

Rémi Taniel

13/12/2019

## Introduction

On commence tout d’abord par charger les données dans une variable data :

data = as.data.frame(readxl::read\_xls('debitmetrie.xls'))

On propose de regarder à quoi ressemble nos données :

head(data)

## id resul fig fid fpg fpd tpg tpd carg card  
## 1 1 2.00 74.9 76.7 91.0 93.6 87.2 91.3 78.0 88.0  
## 2 2 2.50 80.3 80.8 82.3 90.3 80.8 89.6 76.9 82.5  
## 3 3 1.70 77.8 77.4 79.8 90.0 84.3 83.4 77.3 69.4  
## 4 4 3.75 74.8 72.9 78.8 77.1 93.4 85.8 89.3 81.2  
## 5 5 2.00 81.2 82.0 80.6 79.4 79.6 91.1 75.4 78.3  
## 6 6 4.00 88.8 90.2 93.3 95.1 87.7 88.4 89.8 95.8

On remarque que toutes nos variables sont des variables qualitatives, seule la variable id n’est pas à prendre à compte dans notre analyse, pour l’enlever on décide de l’utiliser pour nommer nos lignes :

data <- data[,-1]  
head(data)

## resul fig fid fpg fpd tpg tpd carg card  
## 1 2.00 74.9 76.7 91.0 93.6 87.2 91.3 78.0 88.0  
## 2 2.50 80.3 80.8 82.3 90.3 80.8 89.6 76.9 82.5  
## 3 1.70 77.8 77.4 79.8 90.0 84.3 83.4 77.3 69.4  
## 4 3.75 74.8 72.9 78.8 77.1 93.4 85.8 89.3 81.2  
## 5 2.00 81.2 82.0 80.6 79.4 79.6 91.1 75.4 78.3  
## 6 4.00 88.8 90.2 93.3 95.1 87.7 88.4 89.8 95.8

## Question 1

On décide de proposer des statistiques univariées pour chacune de nos variables, dans un premier temps, on utilise la fonction suivante :

summary(data)

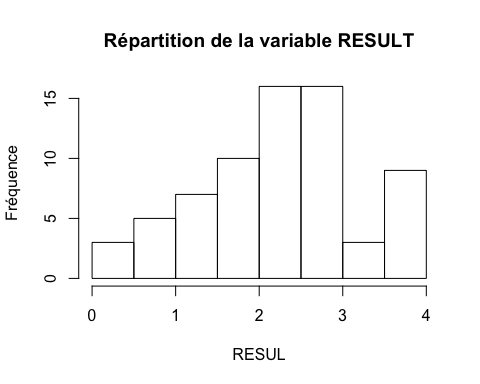
## resul fig fid fpg   
## Min. :0.000 Min. :45.00 Min. :54.00 Min. :47.00   
## 1st Qu.:1.800 1st Qu.:74.80 1st Qu.:75.90 1st Qu.:77.75   
## Median :2.500 Median :81.50 Median :82.00 Median :82.00   
## Mean :2.349 Mean :79.57 Mean :80.96 Mean :80.68   
## 3rd Qu.:2.900 3rd Qu.:87.50 3rd Qu.:88.50 3rd Qu.:86.40   
## Max. :4.000 Max. :93.90 Max. :98.20 Max. :93.70   
## NA's :2   
## fpd tpg tpd carg   
## Min. :59.00 Min. :53.00 Min. :61.00 Min. :45.00   
## 1st Qu.:79.20 1st Qu.:76.20 1st Qu.:80.53 1st Qu.:69.00   
## Median :83.20 Median :83.80 Median :85.90 Median :75.30   
## Mean :83.37 Mean :81.21 Mean :84.19 Mean :74.79   
## 3rd Qu.:90.05 3rd Qu.:87.45 3rd Qu.:89.88 3rd Qu.:82.40   
## Max. :95.30 Max. :94.70 Max. :99.80 Max. :98.90   
## NA's :2 NA's :2 NA's :3   
## card   
## Min. :50.20   
## 1st Qu.:71.45   
## Median :79.60   
## Mean :77.60   
## 3rd Qu.:84.00   
## Max. :99.00   
## NA's :2

On remarque que pour plusieurs variables (fpg, fpd, tpg, tpd, card) nous avons des données manquantes, il faudra y faire attention lors de nos prochaines analyses, Pour chacune de nos variables on décide de

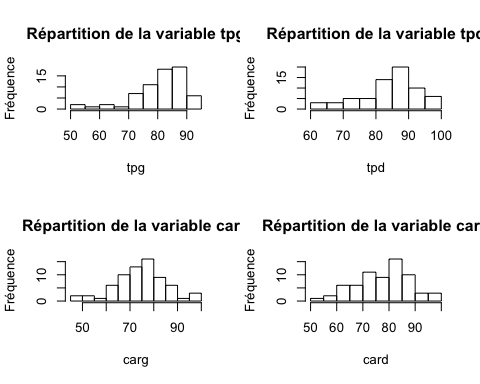
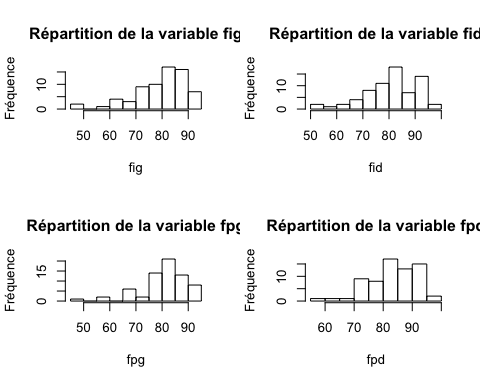
stats = apply(data, 2, function(col) {  
 nb\_obs = length(na.omit(col))  
 nb\_na = length(col[is.na(col)])  
 min = min(col, na.rm = TRUE)  
 max = max(col, na.rm = TRUE)  
 mean = mean(col, na.rm = TRUE)  
 sd = sd(col, na.rm = TRUE)  
   
 return(c(nb\_obs, nb\_na, min, max, mean, sd))  
})  
row.names(stats) = c('Nb obs.', 'Nb NA', 'Min', 'Max', 'Moyenne', 'Ecart type')  
knitr::kable(round(t(stats), 3), format = 'markdown', align = 'r')

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Nb obs. | Nb NA | Min | Max | Moyenne | Ecart type |
| resul | 69 | 0 | 0.0 | 4.0 | 2.349 | 0.979 |
| fig | 69 | 0 | 45.0 | 93.9 | 79.565 | 10.015 |
| fid | 69 | 0 | 54.0 | 98.2 | 80.964 | 9.745 |
| fpg | 67 | 2 | 47.0 | 93.7 | 80.676 | 8.876 |
| fpd | 67 | 2 | 59.0 | 95.3 | 83.370 | 8.006 |
| tpg | 67 | 2 | 53.0 | 94.7 | 81.207 | 8.890 |
| tpd | 66 | 3 | 61.0 | 99.8 | 84.192 | 8.811 |
| carg | 69 | 0 | 45.0 | 98.9 | 74.794 | 10.817 |
| card | 67 | 2 | 50.2 | 99.0 | 77.604 | 10.093 |

hist(data$resul, xlab = 'RESUL', ylab = 'Fréquence', main = 'Répartition de la variable RESULT')



par(mfrow = c(2,2))  
for(i in 2:ncol(data)) {  
 hist(data[,i], xlab = names(data)[i], ylab = 'Fréquence', main = paste('Répartition de la variable', names(data)[i]))  
}



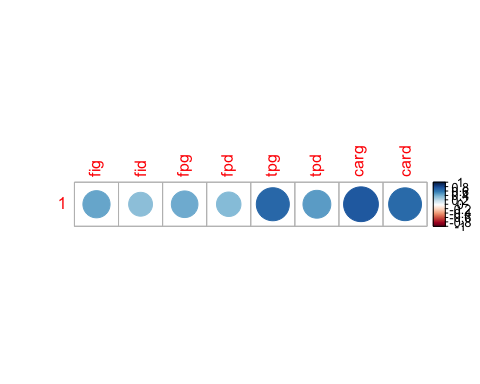
## Question 2

On décide ensuite de trouver les variables les plus corrélées avec la variable RESUL qui represente le résultat du test cognitif, pour cela nous allons utiliser un nuage de point :

library(corrplot)

## corrplot 0.84 loaded

corrplot::corrplot(cor(data$resul, data[,-1], use = 'pairwise.complete.obs'))



Les variables les plus corrélées avec la variable RESUL sont les variables :

* tpg représentant le débit dans
* carg représentant le débit dans la carotide gauche
* card représentant le débit dans la carotide droite

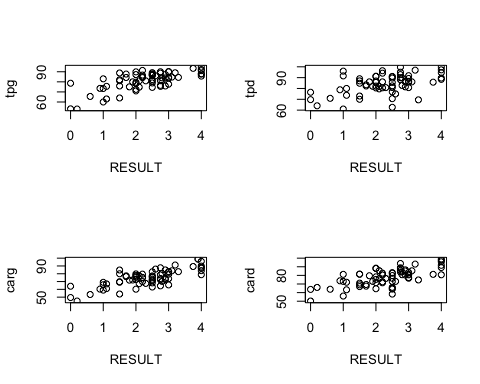
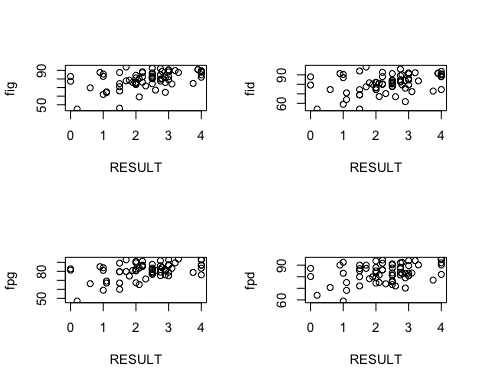
Les valeurs précises sont obtenus grâce à la fonction suivante :

cor(data$resul, data[,-1], use = 'pairwise.complete.obs')

## fig fid fpg fpd tpg tpd carg  
## [1,] 0.4683541 0.3607721 0.4494765 0.3800304 0.6920652 0.4924288 0.7681647  
## card  
## [1,] 0.6844961

Maintenant, on décide de présenter les statistiques bivariées de la variable RESUL par rapport aux autres variables :

par(mfrow = c(2,2))  
for(i in 2:ncol(data)) {  
 plot(data[,1], data[,i], xlab = 'RESULT', ylab = names(data)[i])  
}



## Question 3

On réalise maintenant le modèle de régression linéaire simple entre la variable RESUL et la variable card, pour cela, on utilise la fonction lm :

model = lm(resul~card, data = data)

### Qualité du modèle

On explore les propriétés de notre modèle linéaire en appliquant la fonction summary :

summary(model)

##   
## Call:  
## lm(formula = resul ~ card, data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.53579 -0.36168 0.03938 0.43687 1.49658   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.721895 0.669305 -4.067 0.000131 \*\*\*  
## card 0.064750 0.008554 7.570 1.7e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7014 on 65 degrees of freedom  
## (2 observations deleted due to missingness)  
## Multiple R-squared: 0.4685, Adjusted R-squared: 0.4604   
## F-statistic: 57.3 on 1 and 65 DF, p-value: 1.695e-10

On a un r^2 égal à 46,85%, c’est à dire que la variable card explique 46.85% de l’information de la variable RESUL, notre modèle est donc de bonne qualité, ce qui n’est pas illogique vu que le coefficient de corrélation entre ces 2 variables est de 0.6844961

### Validité du modèle

On décide maintenant de vérifier la validité de notre modèle, pour cela on dispose de trois critères :

* La normalité des résidus
* L’indépendance des résidus
* L’homoscédasticité des résidus

#### Normalité des résidus

La normalité des résidus est vérifié par le test de Shapiro grâce à la fonction :

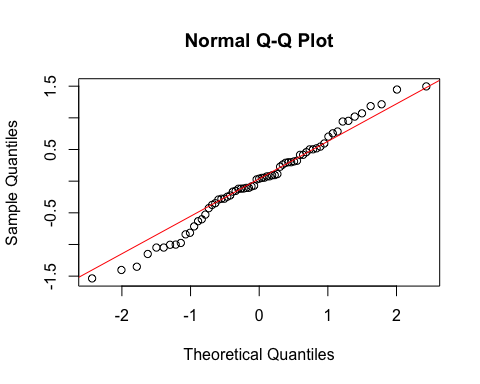
shapiro.test(model$residuals)

##   
## Shapiro-Wilk normality test  
##   
## data: model$residuals  
## W = 0.98626, p-value = 0.6702

La p-value étant de 0.6702, elle est donc supérieure à 0.05, donc on ne rejette pas l’hypothèse

On peut également tracer le graphique suivant pour vérifier la normalité des résidus :

qqnorm(model$residuals)  
qqline(model$residuals, col = 'red')



#### Indépendance des résidus

Pour tester l’indépendance des résidus, on utilise cette fois le test de Durbin-Watson :

library(lmtest)

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

dwtest(model)

##   
## Durbin-Watson test  
##   
## data: model  
## DW = 1.6466, p-value = 0.06891  
## alternative hypothesis: true autocorrelation is greater than 0

Tout comme le test précédent, la p-value est supérieure à 0.05 donc on ne rejette pas l’hypothèse

#### Homoscédasticité des résidus

Afin de vérifier l’homoscédasticité des résidus on utilise le test de Breusch-Pagan :

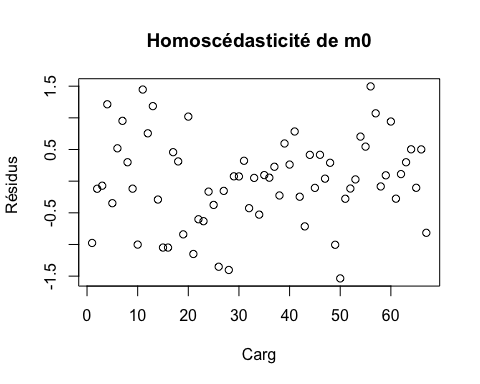
bptest(model)

##   
## studentized Breusch-Pagan test  
##   
## data: model  
## BP = 2.3314, df = 1, p-value = 0.1268

Comme les 2 précédents tests, la p-value est supérieure à 0.05 donc on ne rejette pas l’hypothèse

Ceci est confirmé par le graphique suivant :

plot(model$residuals, xlab = "Carg", ylab = "Résidus ", main = "Homoscédasticité de m0")



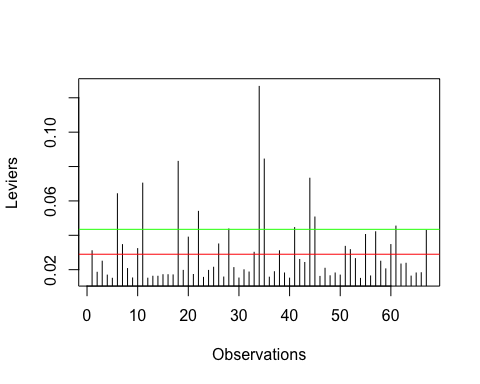
### Influence des observations

On décide d’étudier l’influence des observations sur l’estimation du modèle, pour cela on utilise le code suivant :

influence = lm.influence(model)  
str(influence)

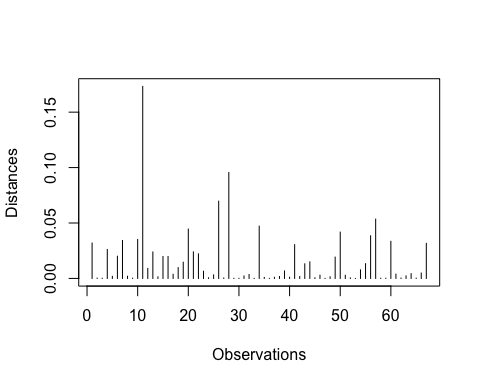
## List of 4  
## $ hat : Named num [1:67] 0.031 0.0185 0.0249 0.0168 0.015 ...  
## ..- attr(\*, "names")= chr [1:67] "1" "2" "3" "4" ...  
## $ coefficients: num [1:67, 1:2] 0.10583 0.00508 -0.00807 -0.03282 -0.00244 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ : chr [1:67] "1" "2" "3" "4" ...  
## .. ..$ : chr [1:2] "(Intercept)" "card"  
## $ sigma : Named num [1:67] 0.696 0.707 0.707 0.69 0.705 ...  
## ..- attr(\*, "names")= chr [1:67] "1" "2" "3" "4" ...  
## $ wt.res : Named num [1:67] -0.9761 -0.12 -0.0717 1.2142 -0.348 ...  
## ..- attr(\*, "names")= chr [1:67] "1" "2" "3" "4" ...

plot(influence$hat, xlab = 'Observations', ylab = 'Leviers', type = 'h')  
abline(h = 2/nrow(data), col = 'red')  
abline(h = 3/nrow(data), col = 'green')



Puis un graphique représentant les distances de Cook :

plot(cooks.distance(model), xlab = 'Observations', ylab = 'Distances', type = 'h')



### Validation croisée

Pour calculer le PRESS, on utilise la fonction suivante :

press = sum((model$residuals / (1 - influence$hat))^2)  
press

## [1] 34.05903

On peut également calculer le REMSEP :

sqrt((1 / length(model$residuals)) \* press)

## [1] 0.7129823

### Autres

Réaliser la même analyse sur la base us\_crime.txt afin d’expliquer la variable R (taux de criminalité) a partir de la variable Ed (education)

## Question 4