Modèle Linéaire : séléction de modèle

Rémi TANIEL

12/18/2019

Contents

Introduction **2**

Question 1 **2**

Statistiques descriptives univariées 2

Statistiques descriptives bivariées 5

Question 2 **8**

Question 3 **9**

Analyse en composantes principales 9

Conclusion 11

Question 4 **11**

Séléction de variables 12

Qualité du modèle 12

Validation du modèle 13

Performance du modèle 15

Nouveau modèle avec 2 variables 16

Création du modèle 17

Qualité du modèle 17

Validation du modèle 17

Question 5 **20**

Critère du Cp 20

Critère du R^2 ajusté 20

Critère du R^2 21

Question 6 **22**

Introduction

On commence par importer nos données grâce au code suivant :

```
data = read.table('cornell.csv', header = TRUE, sep = ',', dec = '.')
```

On décide de regarder quelle est la taille de nos données :

```
dim(data)
```

```
## [1] 12  8
```

On a donc 12 observations pour 8 variables, on peut décider de visualiser nos données grâce à la fonction head :

```
head(data)
```

```
##   X1   X2 X3   X4   X5   X6   X7    Y
## 1  0 0.23  0 0.00 0.00 0.74 0.03 98.7
## 2  0 0.10  0 0.00 0.12 0.74 0.04 97.8
## 3  0 0.00  0 0.10 0.12 0.74 0.04 96.6
## 4  0 0.49  0 0.00 0.12 0.37 0.02 92.0
## 5  0 0.00  0 0.62 0.12 0.18 0.08 86.6
## 6  0 0.62  0 0.00 0.00 0.37 0.01 91.2
```

Question 1

Statistiques descriptives univariées

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

```
summary(data)
```

```
##        X1                X2               X3                X4
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.0750   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.2000   Median :0.00000   Median :0.3150
##  Mean   :0.07417   Mean   :0.2183   Mean   :0.04333   Mean   :0.2533
##  3rd Qu.:0.17000   3rd Qu.:0.2925   3rd Qu.:0.10000   3rd Qu.:0.3800
##  Max.   :0.21000   Max.   :0.6200   Max.   :0.12000   Max.   :0.6200
##        X5                X6               X7                Y
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.01000   Min.   :81.40
##  1st Qu.:0.00000   1st Qu.:0.0600   1st Qu.:0.03750   1st Qu.:82.92
##  Median :0.01000   Median :0.2750   Median :0.07000   Median :87.35
##  Mean   :0.04333   Mean   :0.3108   Mean   :0.05667   Mean   :88.58
##  3rd Qu.:0.12000   3rd Qu.:0.4625   3rd Qu.:0.08000   3rd Qu.:93.15
##  Max.   :0.12000   Max.   :0.7400   Max.   :0.08000   Max.   :98.70
```

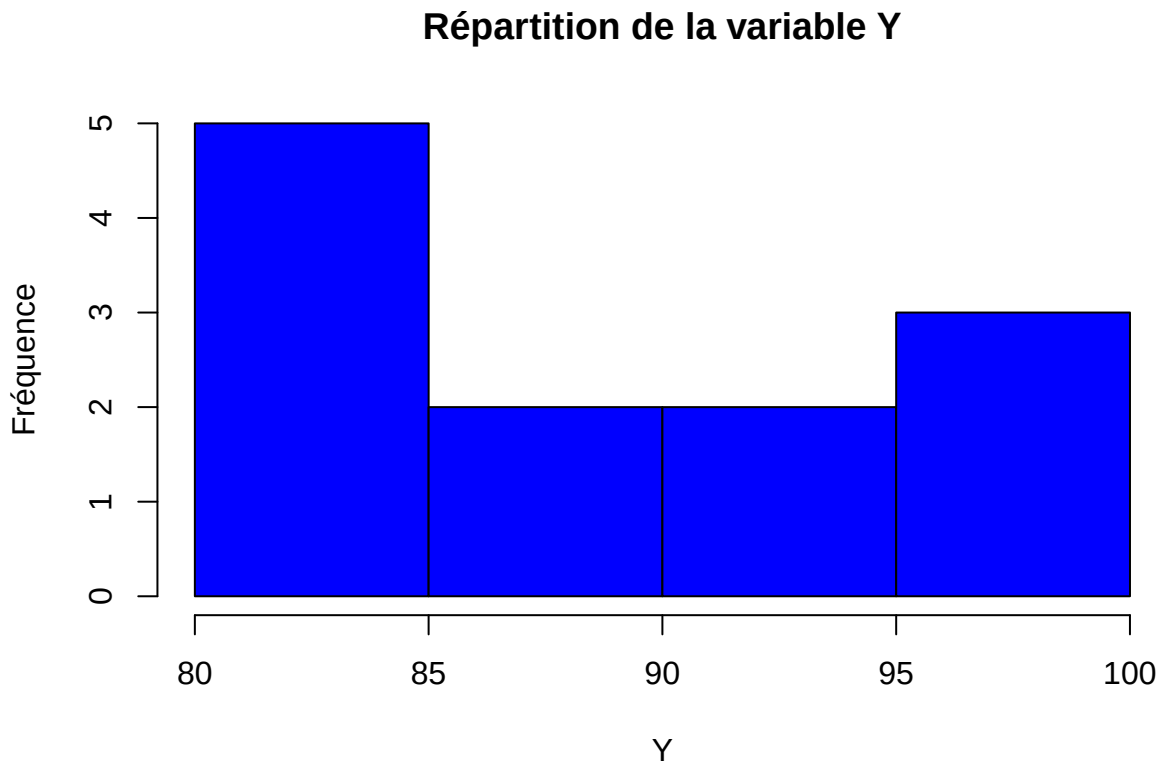
On remarque dans un premier temps que nous n'avons pas de données manquantes dans notre jeu de donnée, et on remarque les choses suivantes :

- Pour les variables X1, X3, X5 et X7, la plage entre la valeur minium et la valeur maximum est faible
- Tandis que pour les variables X2, X4 et X6, cette même plage est très grande

Répartition des variables

On décide également d'afficher la répartition de la variable Y grâce au graphique suivant :

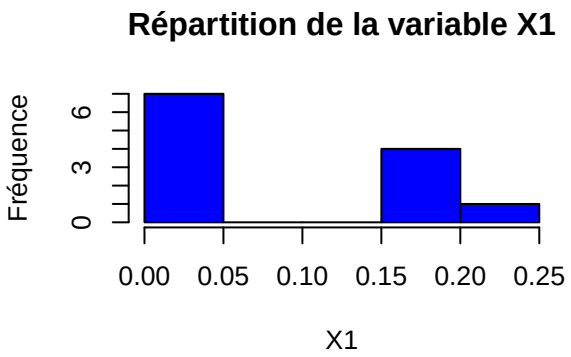
```
hist(data$Y, xlab = 'Y', ylab = 'Fréquence', main = 'Répartition de la variable Y', col = 'blue')
```

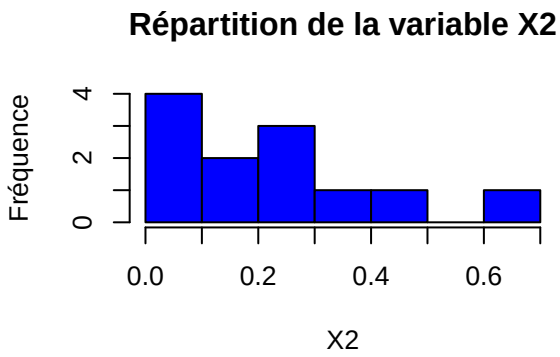


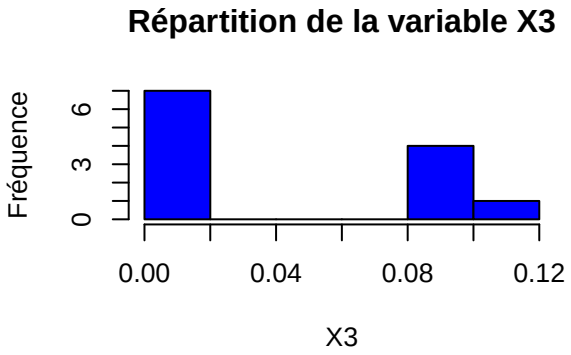
La majorité de nos variables sont donc comprises entre les intervalles [80;85] et [95;100]

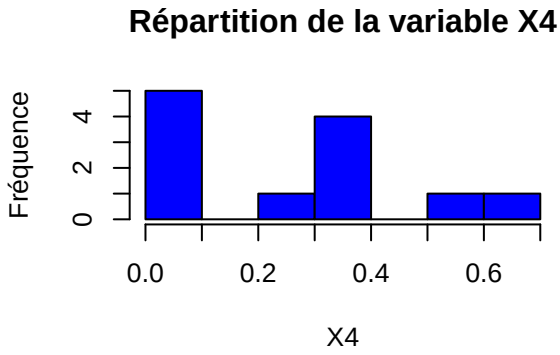
On décide de faire pareil pour le reste des variabes de notre jeu de donnée :

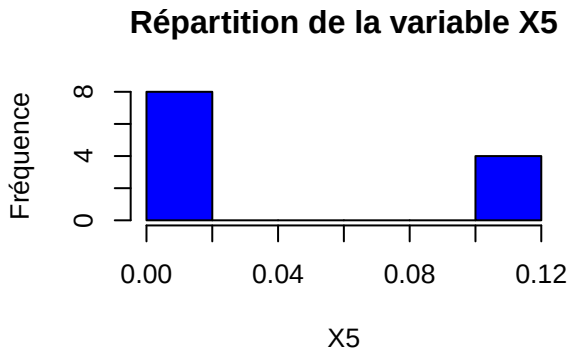
```
par(mfrow = c(2,2))
for(i in 1:(ncol(data)-1)) {
  hist(data[,i], xlab = names(data)[i], ylab = 'Fréquence', main = paste('Répartition de la variable', n
}
```

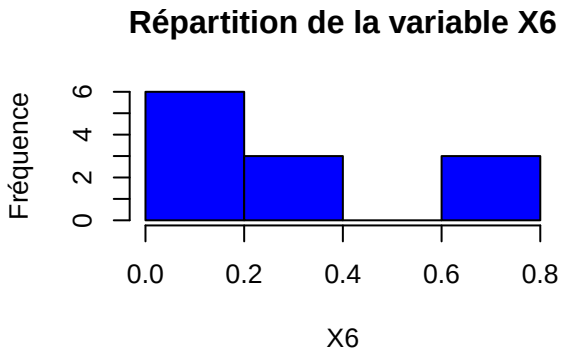

Répartition de la variable X1
Fréquence
0
3
6
0.00
0.05
0.10
0.15
0.20
0.25
X1

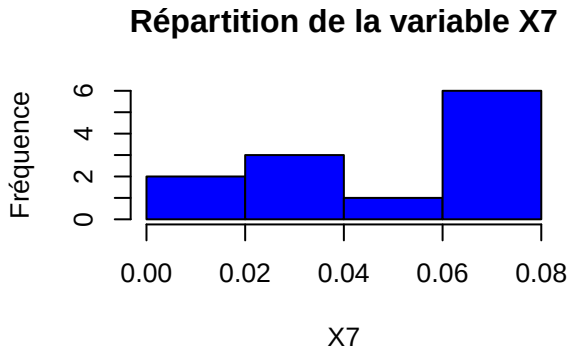

Répartition de la variable X2
Fréquence
0
2
4
0.0
0.2
0.4
0.6
X2


Répartition de la variable X3
Fréquence
0
3
6
0.00
0.04
0.08
0.12
X3


Répartition de la variable X4
Fréquence
0
2
4
0.0
0.2
0.4
0.6
X4







On remarque que la réparition des variables X1, X3 et X5 sont très disparates, pour chacunes de ces variables il n'y aucune valeur au milieu de l'intervalle, les valeurs sont situés au minimum et à l'extremum de l'intervalle

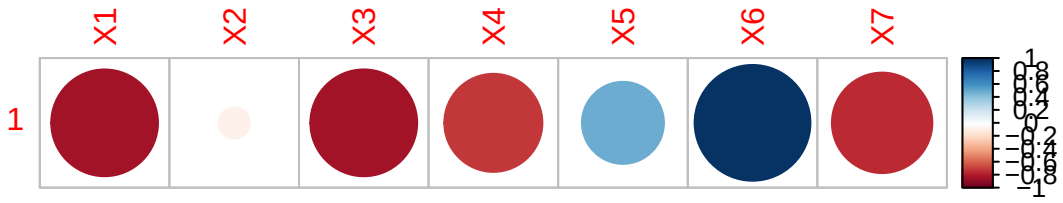
Statistiques descriptives bivariées

On va maintenant comparer la variable Y par rapport aux autres variables de notre problème

Corrélation

On va dans un premier temps regarder quelles sont les variables qui sont les plus corrélées avec la variable Y, pour cela, on trace le graphique suivant :

```
library(corrplot)
corrplot::corrplot(cor(data$Y, data[,-8], use = 'pairwise.complete.obs'))
```



Grâce à ce graphique on remarque tout de suite les affirmations suivantes :

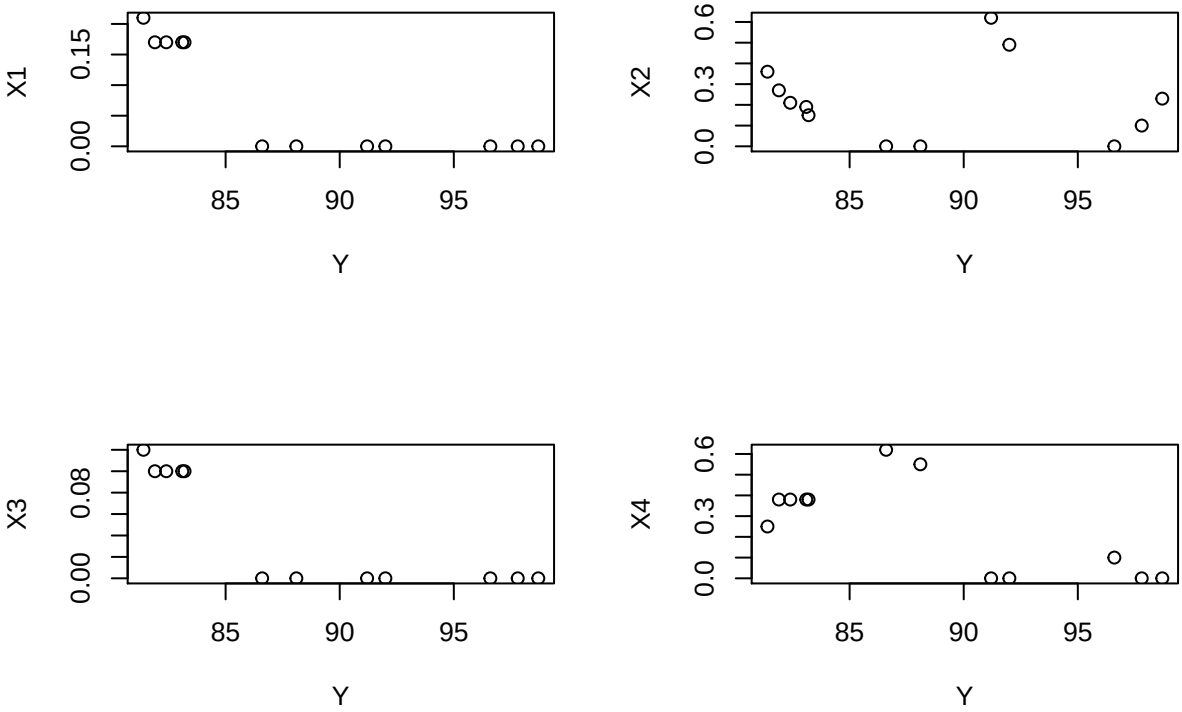
- La variable X6 est très fortement positivement corrélée avec la variable Y
- Les variables X1, X3, X4 et X7 sont très fortement négativement corrélées avec la variable Y
- La variable X2 n'est quasiment pas corrélée avec la variable Y

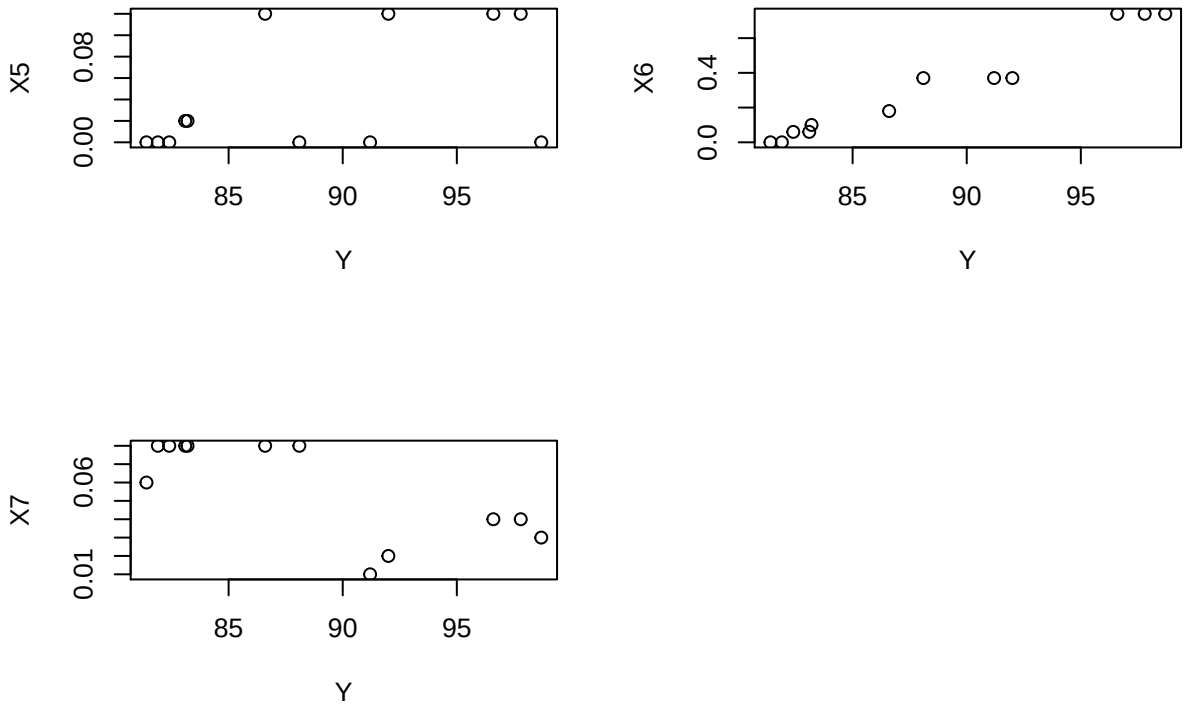
On peut obtenir les valeurs précises grâce au bout de code suivant :

```
library(knitr)
kable(cor(data$Y, data[,-8], use = 'pairwise.complete.obs'))
```

X1	X2	X3	X4	X5	X6	X7
-0.8372958	-0.0708189	-0.8379578	-0.7067135	0.493799	0.9850704	-0.7411162

```
par(mfrow = c(2,2))
for(i in 1:(ncol(data) - 1)) {
  plot(data[,8], data[,i], xlab = 'Y', ylab = names(data)[i])
}
```


X1
0.00
0.15
85
90
95
Y
X2
0.0
0.3
0.6
85
90
95
Y
X3
0.00
0.08
85
90
95
Y
X4
0.0
0.3
0.6
85
90
95
Y



Question 2

On réalise le modèle de régression entre Y et les autres variables en utilisant la fonction lm :

```
m0 = lm(Y~., data = data)
```

Et on affiche les informations relatives à celui-ci :

```
summary(m0)
```

```
##
## Call:
## lm(formula = Y ~ ., data = data)
##
## Residuals:
##          1          2          3          4          5          6          7
##  1.207e+00 -2.218e-01 -5.475e-01 -8.195e-02  8.105e-01 -3.527e-01  5.906e-02
##          8          9         10         11         12
##  3.598e-01 -3.036e-01 -1.153e-01  4.718e-15 -8.141e-01
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   116.92      81.10   1.442    0.209
```

```
## X1            -82.60     173.22  -0.477    0.654
## X2            -31.00      80.84  -0.383    0.717
## X3             24.33     431.97   0.056    0.957
## X4            -39.74      90.25  -0.440    0.678
## X5            -29.17      84.02  -0.347    0.743
## X6            -16.62      84.45  -0.197    0.852
## X7                NA         NA      NA       NA
##
## Residual standard error: 0.8362 on 5 degrees of freedom
## Multiple R-squared:  0.9925, Adjusted R-squared:  0.9836
## F-statistic: 110.7 on 6 and 5 DF,  p-value: 3.762e-05
```

On remarque plusieurs choses :

- Le r^2 de notre modèle est égale à 0,9925, ce qui signifie que 99,25% de l'information portée par Y est expliquée par les autres variables, la qualité de notre modèle est donc excellente.
- La ligne pour la variable X7 est NA pour tout les paramètres, ce qui signifie que cette variable est linéarement reliée à une variable.

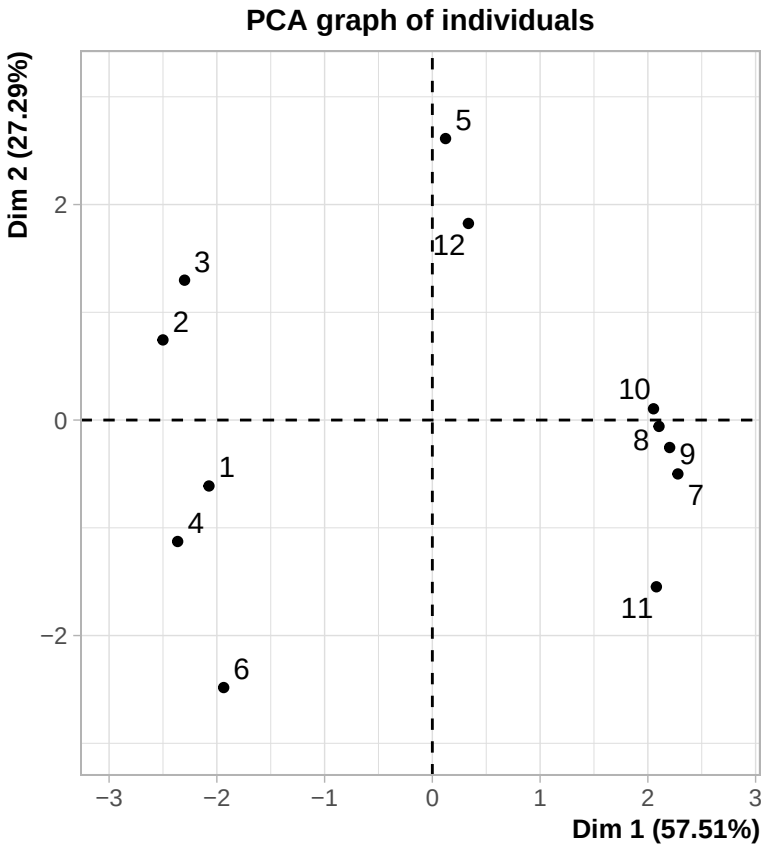
Question 3

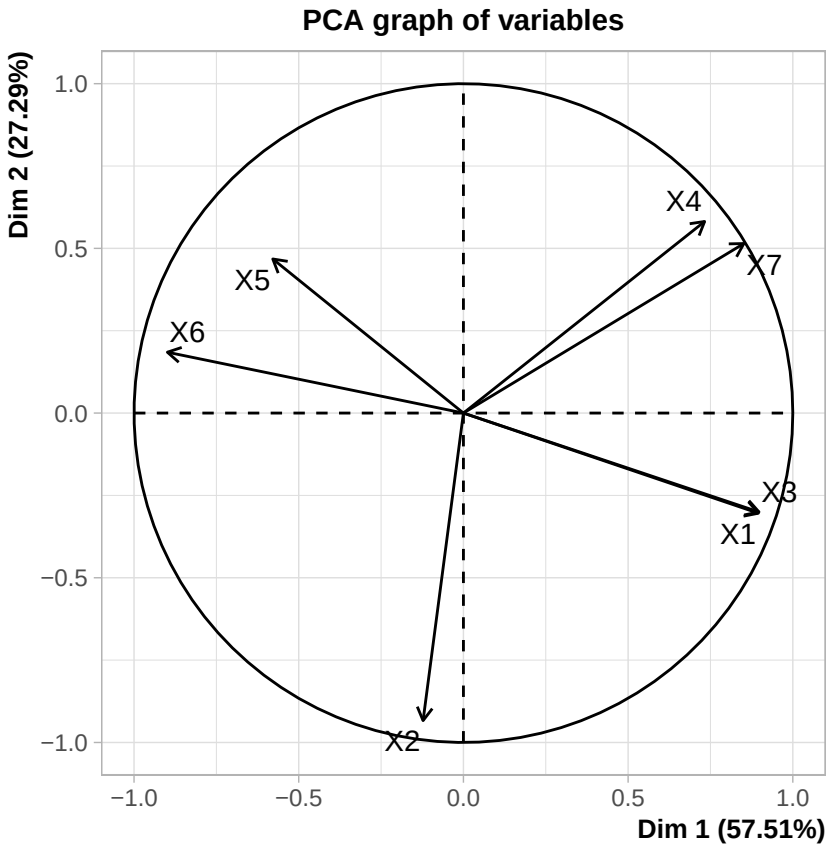
On va vérifier maintenant que nos variables ne sont pas corrélées entre elles, c'est à dire qu'il n'y a pas de relation entre 2 d'entre elles.

Analyse en composantes principales

Pour cela, nous avons plusieurs solutions qui s'offrent à nous, la première solution est de réaliser une ACP sur nos données :

```r
library(FactoMineR)
pca = FactoMineR::PCA(data[,-8], scale.unit = TRUE, ncp = 7, graph = TRUE)
```


PCA graph of individuals
Dim 2 (27.29%)
Dim 1 (57.51%)
5
12
3
2
10
8
9
7
1
4
11
6
2
0
−2
−3
−2
−1
0
1
2
3



On remarque que la variable X7 est fortement positivement correlée avec la variable X4, tout comme les variables X1 et X3.

Conclusion

Or nous avons déjà vu précédemment que la variable X7 est linéarement reliée à une autre variable de notre problème, on peut vérifier cela en calculant le determinant de la matrice XTX :

```
X = as.matrix(data[,-8])
det(t(X) %*% X)
```

```
## [1] 2.510764e-12
```

Etant donné que le déterminant de la matrice peut être considéré comme nul, il existe donc des relations entre certaines variables

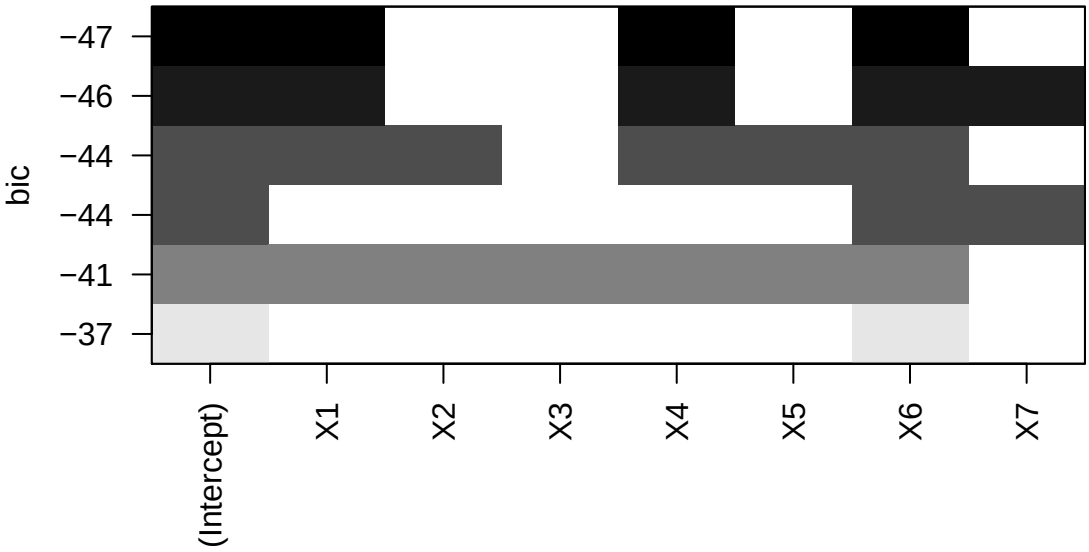
Question 4

On ne peut donc pas faire de modèle prenant en compte toutes les variables, mais quelles sont les variables que nous devons enlever ?

Séléction de variables

On procède dans un premier temps à une selection de variables en utilisant la fonction regsubsets contenue dans la librairie leaps, on choisira le meilleur modèle selon le critère BIC

```
library(leaps)
choix = regsubsets(Y~., int = T, nbest = 1, nvmax = 7, method = 'exh', data = data)
plot(choix, scale = 'bic')
```



Le meilleur modèle est donc celui qui fait rentrer les 3 variables suivantes :

- X1
- X4
- X6

```
model = lm(Y~X1+X4+X6, data = data)
```

Qualité du modèle

On obtient les informations du modèle grâce à la fonction summary :

```
summary(model)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X4 + X6, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.00154 -0.41198  0.02205  0.29286  1.00148
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  85.9435     0.9964  86.255 3.64e-13 ***
## X1          -14.0924     4.1175  -3.423  0.00905 **
## X4           -4.9445     1.3018  -3.798  0.00525 **
## X6           15.8852     1.5779  10.067 8.07e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.707 on 8 degrees of freedom
## Multiple R-squared:  0.9915, Adjusted R-squared:  0.9882
## F-statistic: 309.3 on 3 and 8 DF,  p-value: 1.31e-08
```

Le r^2 de ce modèle est de 0,9882, ce qui signifie que 98,82% de l'information portée par la variable Y est expliquée par les variables X1, X4 et X6

Validation 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 :

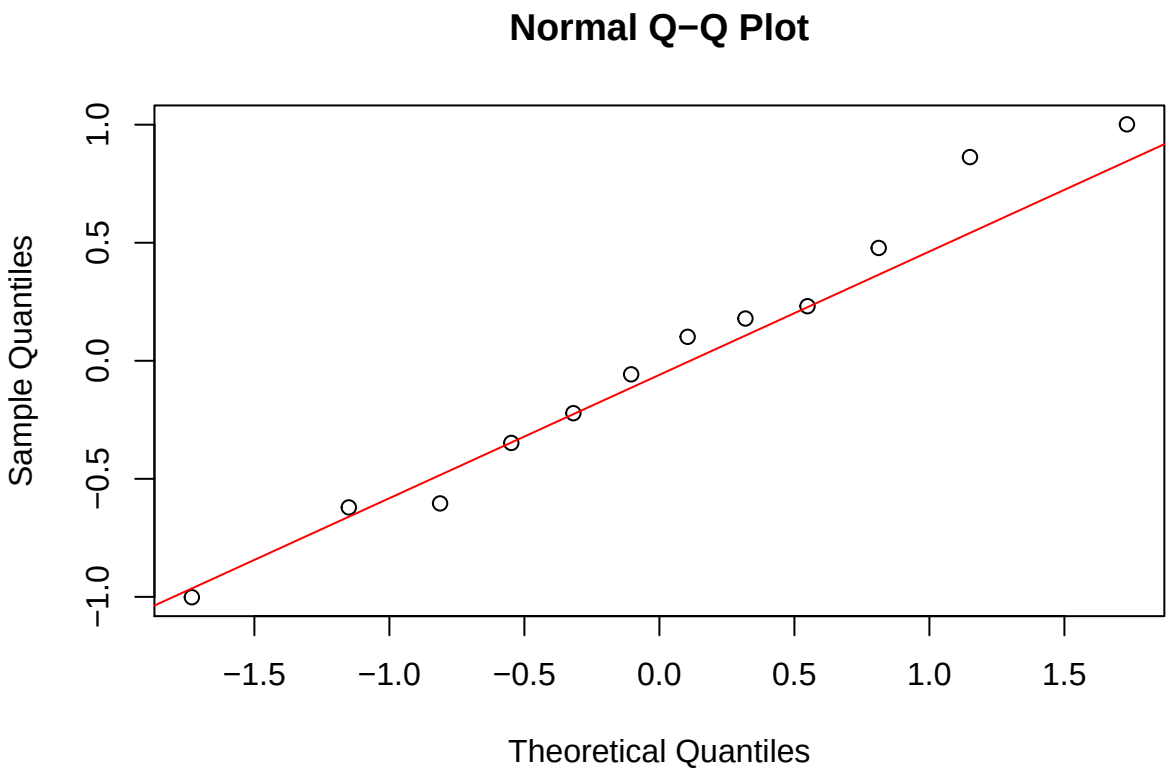
```
shapiro.test(model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.97806, p-value = 0.9747
```

La p-value étant de 0.9747, 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')
```



On remarque que les points suivent la courbe donc les résidus sont donc normales.

Indépendance des résidus

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

```
library(lmtest)
dwtest(model)
```

```
##
##  Durbin-Watson test
##
## data:  model
## DW = 1.6019, p-value = 0.1137
## 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 :

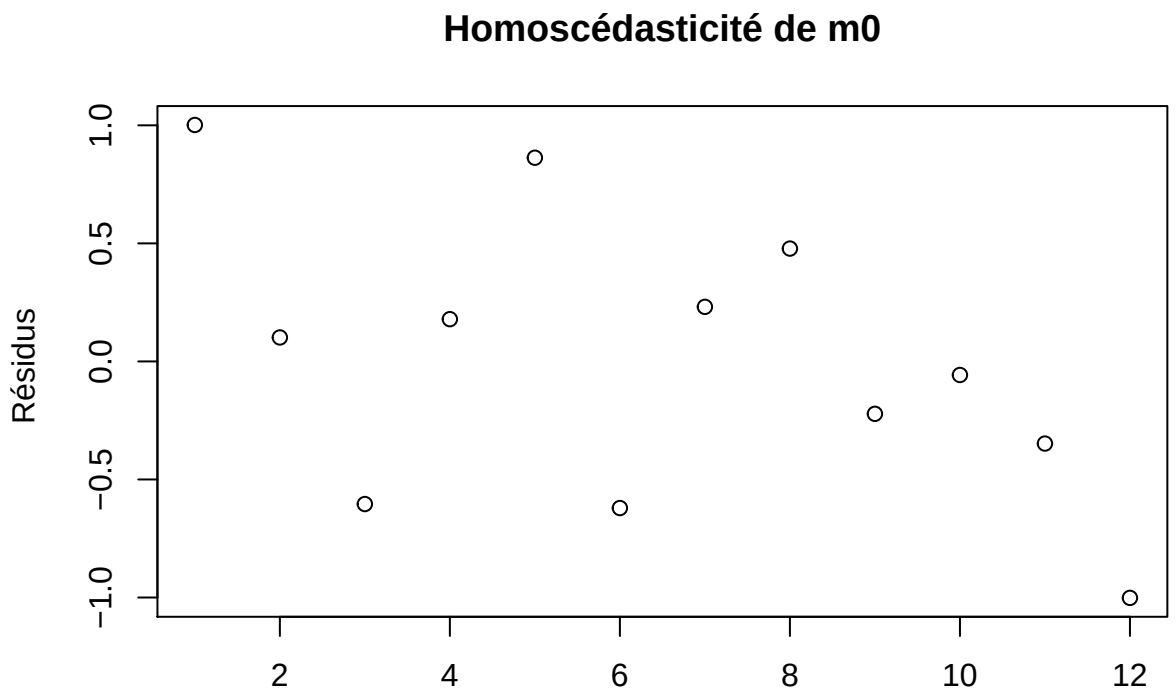
```
bptest(model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model
## BP = 6.4889, df = 3, p-value = 0.0901
```

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 = "", ylab = "Résidus ", main = "Homoscédasticité de m0")
```



Performance du modèle

On rappelle les informations de notre modèle grâce à la fonction suivante :

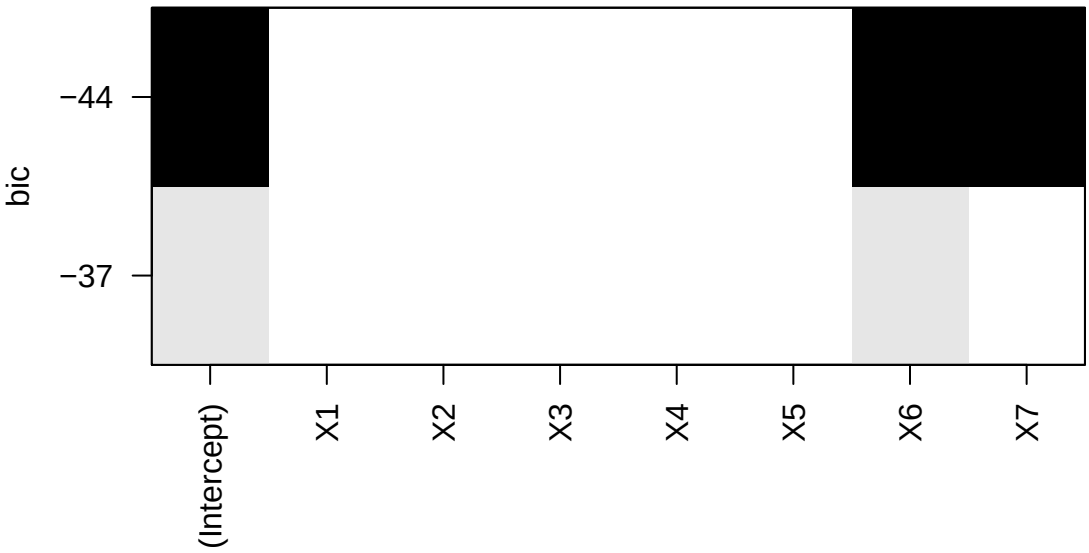
```
summary(model)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X4 + X6, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -1.00154 -0.41198  0.02205  0.29286  1.00148
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  85.9435     0.9964  86.255 3.64e-13 ***
## X1          -14.0924     4.1175  -3.423  0.00905 **
## X4           -4.9445     1.3018  -3.798  0.00525 **
## X6           15.8852     1.5779  10.067 8.07e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.707 on 8 degrees of freedom
## Multiple R-squared:  0.9915, Adjusted R-squared:  0.9882
## F-statistic: 309.3 on 3 and 8 DF,  p-value: 1.31e-08
```

Nouveau modèle avec 2 variables

```
choix_2 = regsubsets(Y~., int = T, nbest = 1, nvmax = 2, method = 'exh', data = data)
plot(choix_2, scale = 'bic')
```



Le meilleur modèle avec 2 variables est le modèle en fonction des variables X6 et X7

Création du modèle

On crée donc le modèle en utilisant le résultat précédent :

```
model_2 = lm(Y~X6+X7, data = data)
```

Comme précédemment on cherche à estimer le modèle et à analyser la validité de celui-ci

Qualité du modèle

On obtient les informations du modèle grâce à la fonction summary :

```
summary(model_2)
```

```
##
## Call:
## lm(formula = Y ~ X6 + X7, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02075 -0.47918 -0.07922  0.42667  1.50196
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   84.788      1.017  83.388  2.6e-14 ***
## X6            19.504      1.161  16.801  4.2e-08 ***
## X7           -40.006     12.556  -3.186   0.0111 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8508 on 9 degrees of freedom
## Multiple R-squared:  0.9861, Adjusted R-squared:  0.983
## F-statistic: 318.6 on 2 and 9 DF,  p-value: 4.44e-09
```

Le r^2 de ce modèle est de 0,983, ce qui signifie que 98,30% de l'information portée par la variable Y est expliquée par les variables X6 et X7.

Validation 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 :

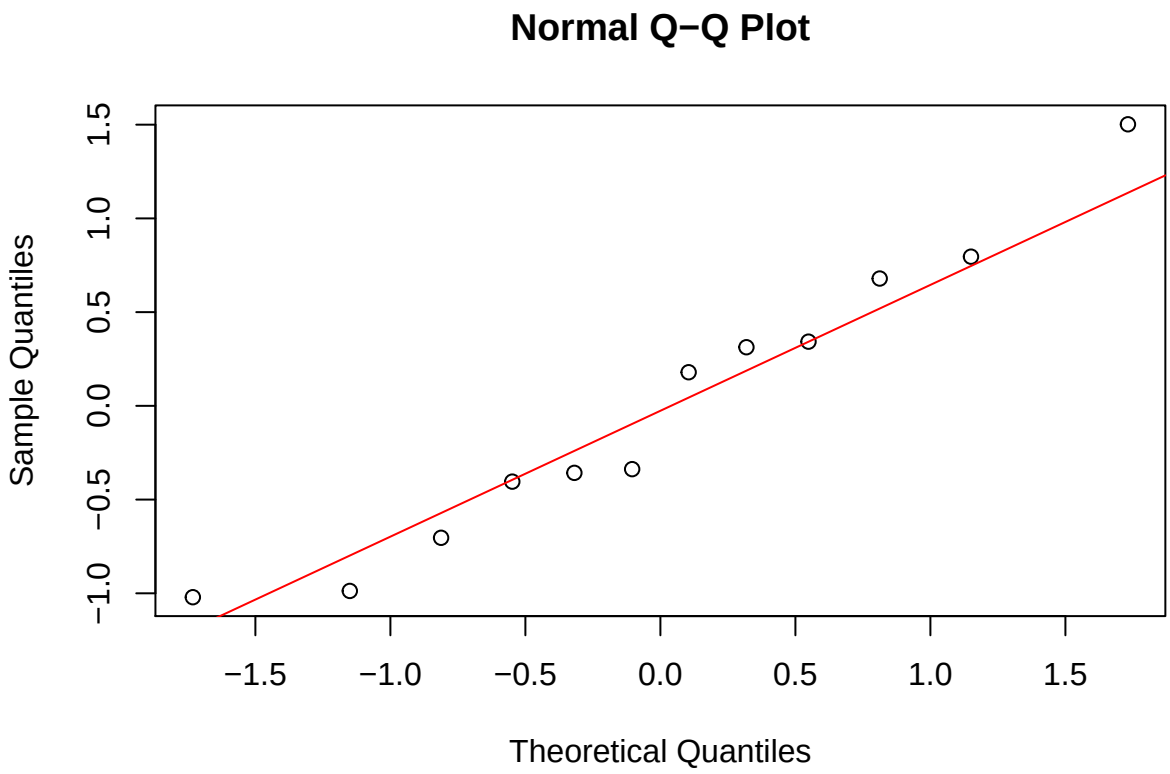
```r
shapiro.test(model_2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model_2$residuals
## W = 0.95523, p-value = 0.7142
```

La p-value étant de 0.7142, 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 :

```r
qqnorm(model_2$residuals)
qqline(model_2$residuals, col = 'red')
```



On remarque que les points suivent la courbe donc les résidus sont donc normales.

Indépendance des résidus

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

```r
library(lmtest)
dwtest(model_2)
```

```
##
##  Durbin-Watson test
##
## data:  model_2
## DW = 1.6316, p-value = 0.1838
## 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

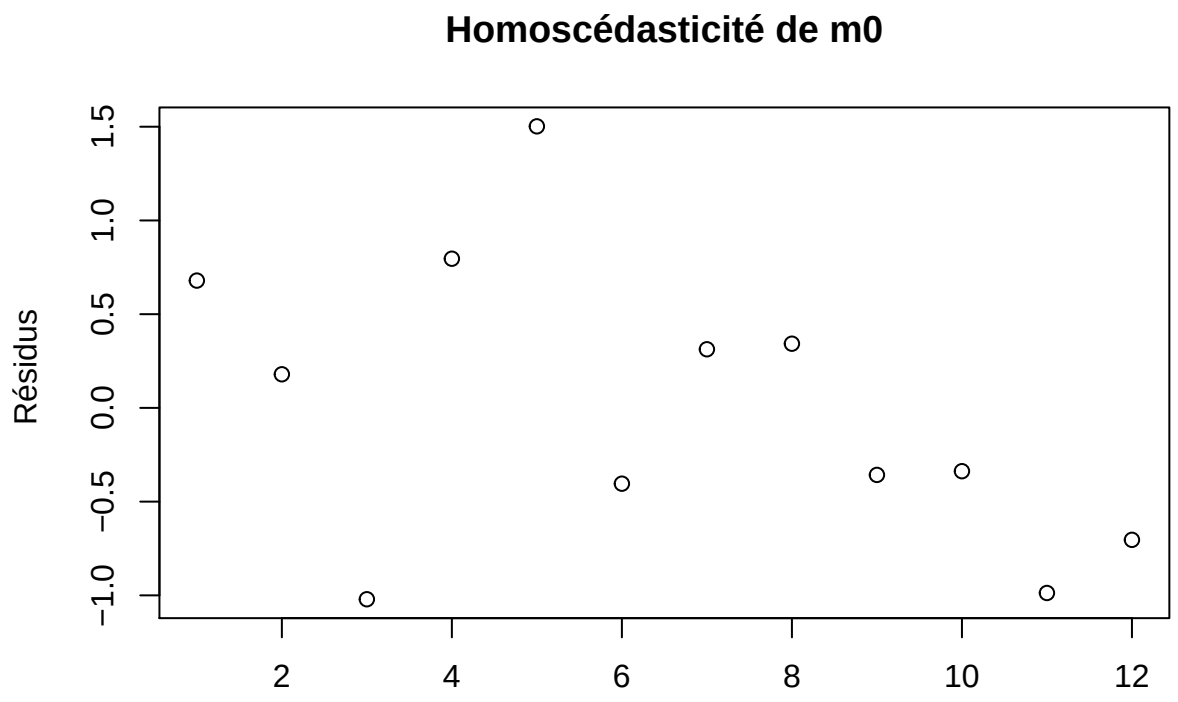
Afin de vérifier l'homoscédasticité des résidus on utilise le test de Breusch-Pagan :

```
bptest(model_2)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model_2
## BP = 0.1957, df = 2, p-value = 0.9068
```

Comme les 2 précédents tests, la p-value est supérieure à 0.05 (0.9068) donc on ne rejette pas l'hypothèse
Ceci est confirmé par le graphique suivant :

```
plot(model_2$residuals, xlab = "", ylab = "Résidus ", main = "Homoscédasticité de m0")
```



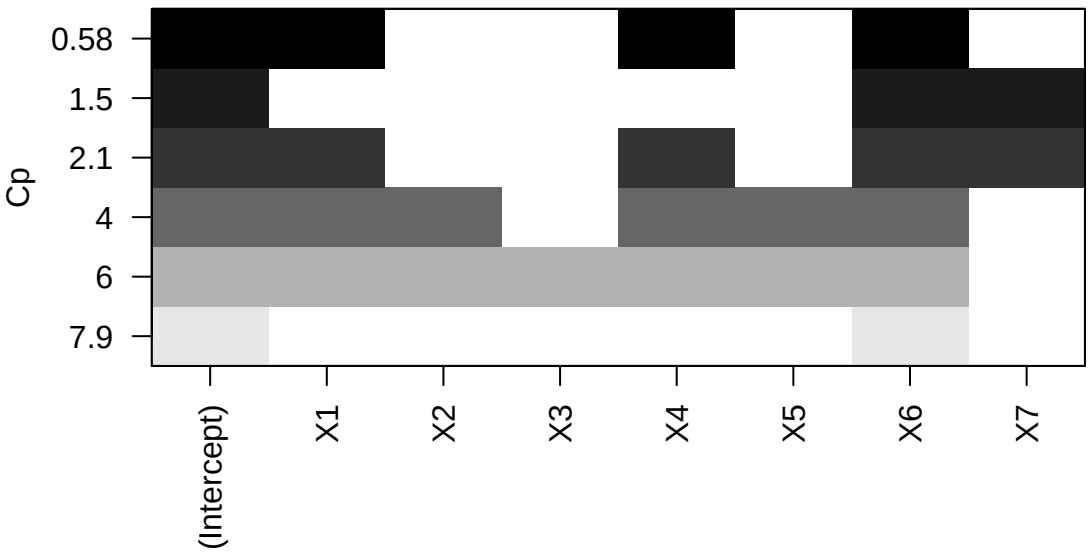
Question 5

Dans la question précédente, on a utilisé le critère BIC, nous allons donc rechoisir le meilleur modèle en utilisant 2 nouveaux critères :

- critère Cp
- critère du R^2 ajuste

Critère du Cp

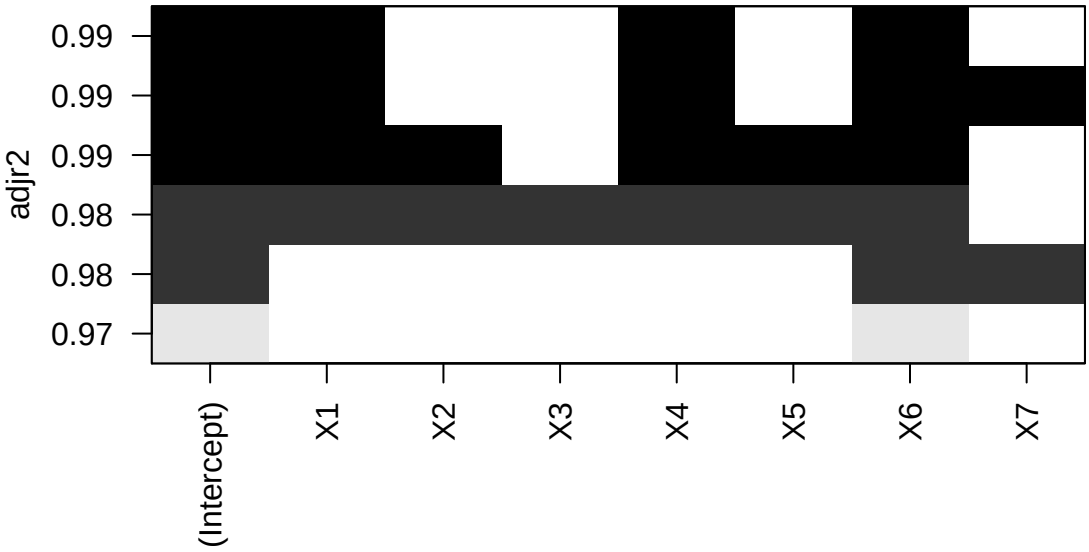
```
plot(choix, scale = 'Cp')
```



Suivant ce critère, on retrouve le même modèle que précédemment, c'est à dire un modèle en fonction de X1, X4 et X6

Critère du R^2 ajusté

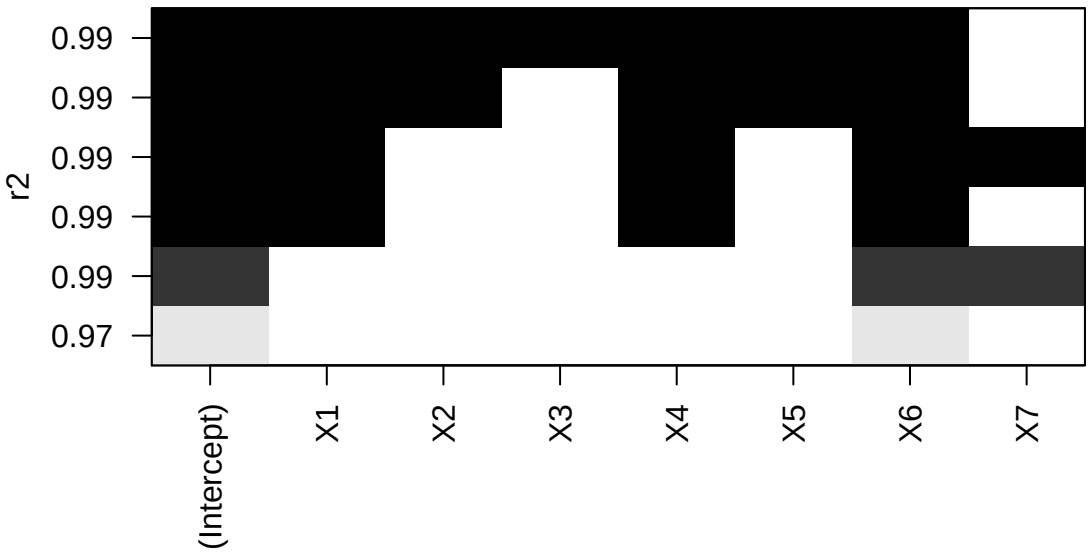
```
plot(choix, scale = 'adjr2')
```



Comme précédemment, on retrouve également le même modèle qu'obtenu avec les 2 derniers critères, c'est à dire un modèle avec X1, X4 et X6.

Critère du R^2

```
plot(choix, scale = 'r2')
```



Contrairement aux 3 critères précédents, le meilleur modèle avec le critère du R^2 est en fonction de toutes les variables sauf X7.

Question 6

Les recheches précédentes étaient exhaustives, or cela pose problème lorsque nous avons un jeu de donnée avec de nombreuses variables, pour éviter ce problème, on utilise une séléction pas-à-pas, nous allons donc créer 3 modèles en utilisant des directions différentes :

- descendante
- ascendante
- les 2

```
library(MASS)
m_0 = lm(Y~1, data = data)
m_all = lm(Y~., data = data)

m_back = stepAIC(m_all, direction = "backward")
m_forw = stepAIC(m_0, direction = "forward", scope = list(upper = m_all, lower = m_0))
m_stepwise = stepAIC(m_0, direction = "both", scope = list(upper = m_all, lower = m_0))
```

Ensuite, nous allons comparer les 3 modèles en utilisant le PRESS, calculer grâce à la fonction suivante :

```r
press = function(fit) {
  h = lm.influence(fit)$h
  return(sqrt(mean((residuals(fit) / (1 - h))^2)))
}
```

On calcule donc le PRESS pour chacun de nos modèles précédents :

```r
c(press(m_back), press(m_forw), press(m_stepwise))
```

```
## [1] 1.287098 1.012127 1.017572
```

Le modèle avec le pouvoir prédictif le plus élevé est donc le modèle formé avec la méthode pas-à-pas AIC avec une direction ascendante