Devoir N2

MTH6312 - METHODES STATISTIQUES D'APPRENTISSAGE

11 octobre 2018

Lyes Heythem Bettache -1923715

# Devoir 2 MTH6312

Lyes Heythem BETTACHE 1923715

11 octobre 2018

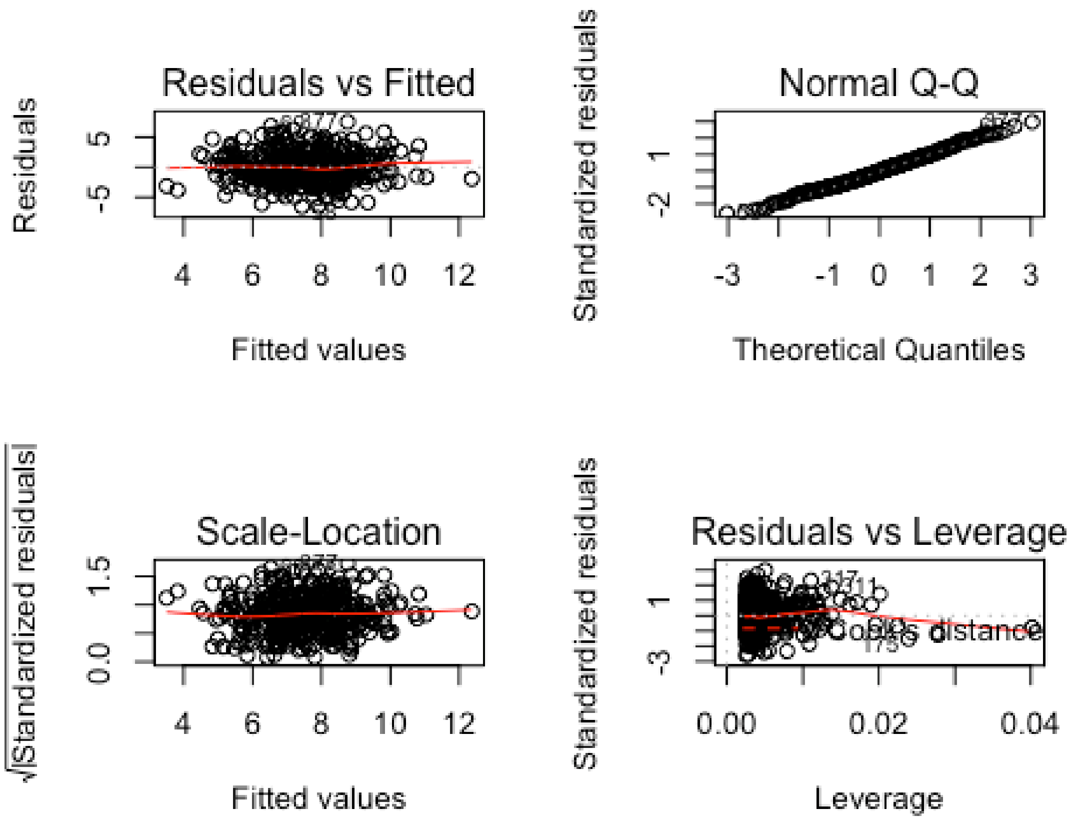
Question N°1

**Prtie a**

```
# On charge le packet ISLR et attache les données Carseats
remove(list=ls())
library(ISLR)
attach(Carseats)

#******************************(a)************************************

#On crée le modèle de régression linéaire

amodel.RL=lm(Sales ~ Price)

summary(amodel.RL)

##
## Call:
## lm(formula = Sales ~ Price)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5224 -1.8442 -0.1459  1.6503  7.5108
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.641915   0.632812  21.558   <2e-16 ***
## Price       -0.053073   0.005354  -9.912   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.532 on 398 degrees of freedom
## Multiple R-squared:  0.198,  Adjusted R-squared:  0.196
## F-statistic: 98.25 on 1 and 398 DF,  p-value: < 2.2e-16
```

On remarque que le p-value: < 2.2e-16 qui implique que notre variable Price est significatif R-squared:0.198, et Adjusted R-squared:0.196 (notre système prédit n'est pas bon)

```r
#les graphiques diagnostiques des résidus
par(mfrow=c(2,2))
plot(amodel.RL)
```




a_1)

D’après le coefficient de détermination R-squared = 0.198 < 0,5 (R=0,44 < 0,7), on remarque qu’on a une mauvaise corrélation entre X1(Price) et Y(Sales).

On conclure que notre modèle n’est pas bien a expliqué le lien entre Y et X1 (notre modèles est mauvais)

a_2)

```r
# On prédit la réponse et les intervalles
X_pred=data.frame(Price=117.5)

pred_modelRLa = predict(amodel.RL,X_pred,type="response",interval = "pred",le
vel = .95)
conf_modelRLa = predict(amodel.RL,X_pred,type="response",interval = "conf",le
vel = .95)
```

```r
# les résultats.
cat(sprintf("Intervale de prévision (0.95) :\n"))

## Intervale de prévision (0.95) :

pred_modelRLa

##        fit      lwr      upr
## 1 7.405836 2.421178 12.39049

cat(sprintf("\n\nIntervale de confiance (0.95) :\n"))

##
##
## Intervale de confiance (0.95) :

conf_modelRLa

##        fit      lwr      upr
## 1 7.405836 7.156269 7.655402
```

Intervale de prévision: [2.421178   12.39049] ,  la valeur de l'output Y prédite :7.405836

Intervale de confiance: [7.156269   7.655402] ,  la valeur de l'output Y prédite :7.405836

a_3)

<u>Residuals vs Fitted</u>

On remarque que les valeurs sont distribuées uniformément dans les deux cotés de la ligne horizontale, c'est un bon indice qu'on n'a pas la relation non linéaire.

<u>Normal Q-Q</u>

On remarque que les résidus sont bien alignés sur la ligne pointillée droite (les résidus sont bien distribués)

<u>Scale-Location</u>

On remarque qu'il y a une ligne horizontale avec des points également (aléatoirement) répartis. les résidus sont répartis de manière égale le long des plages de prédicteurs

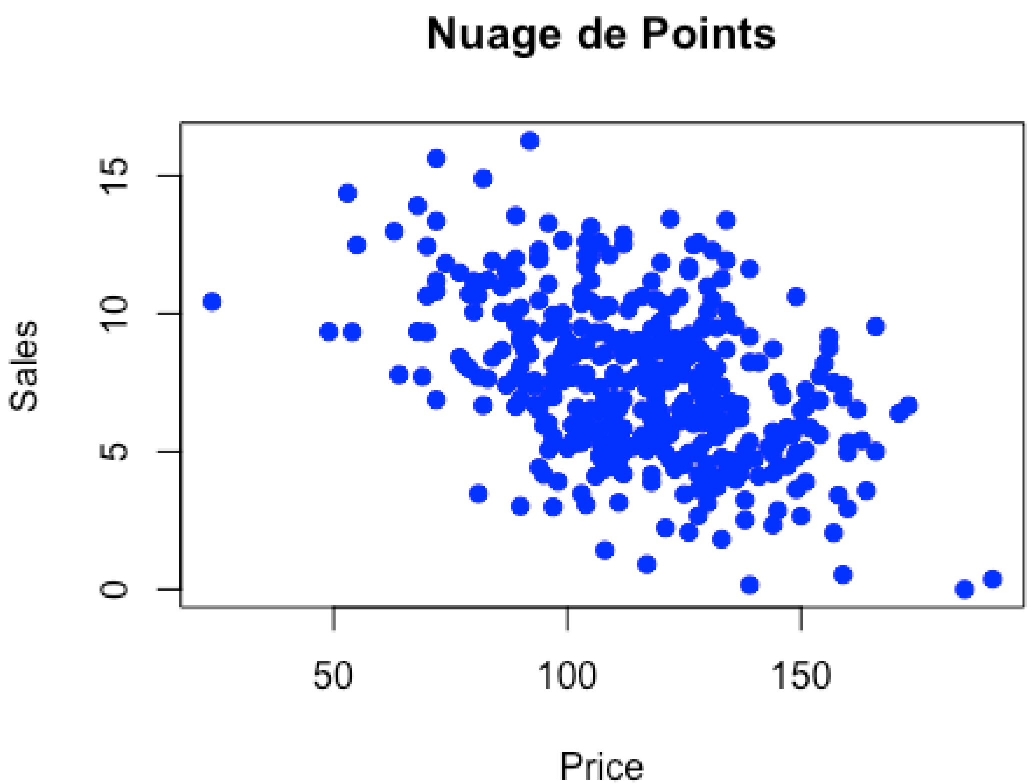
D'après les graphes précédents on remarques qu'il y a des points aberrantes ce qui montre que les graphiques des résidus indiquent une anomalie

Residuals vs Leverage

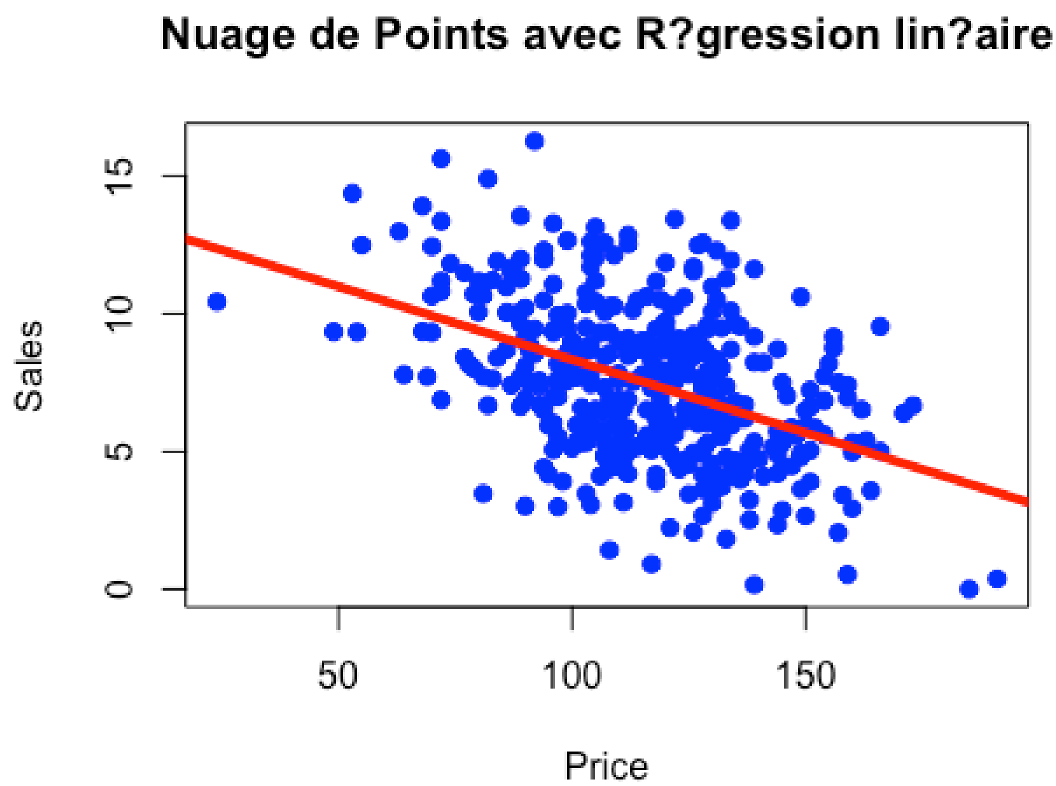
On remarque qu'il n'y a pas des points hors la 'Cookés distance' ce qui montre qu'il n'y a pas des points influents

**Partie b**

```
#*********************************b*********************************

# Nuage de points
plot(Price,Sales,main='Nuage de Points',xlab="Price",ylab="Sales",pch=19, col
="blue")
```



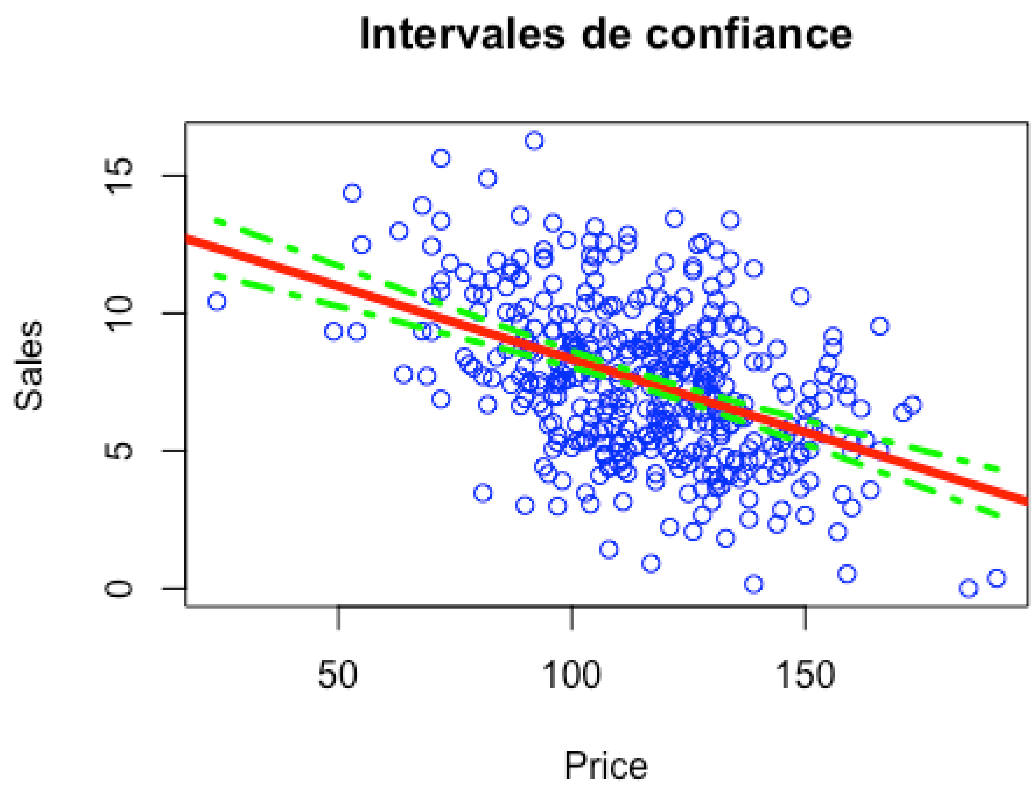

```
# Droite de régression
plot(Price,Sales,main='Nuage de Points avec Régression linéaire',xlab="Price"
,ylab="Sales",pch=19, col="blue")
abline(amodel.RL, col="red", lwd=4)
```




b_2)

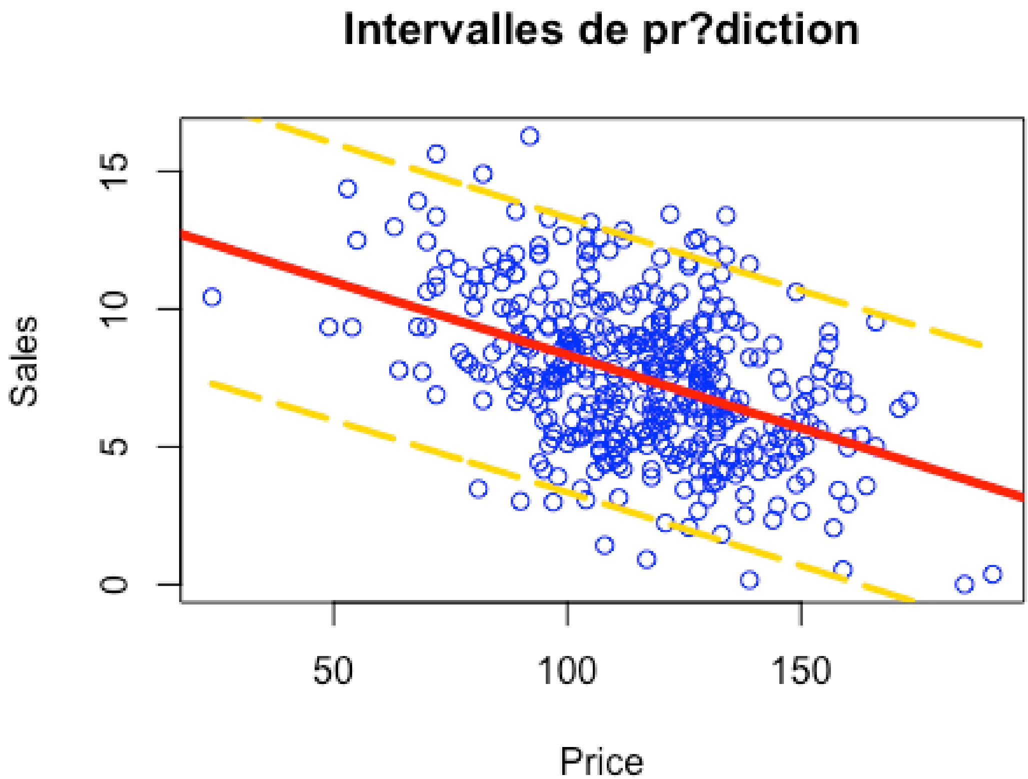
```
# On divise l'espace en 300 parties et un on crée deux matrices pour les inte
rvalles de "pred" et "conf"
Price_divise=seq(from=min(Price), to=max(Price), length.out=300)

pred_Sales_mat=matrix(0,nrow=300,ncol=3,byrow=FALSE)
conf_Sales_mat=matrix(0,nrow=300,ncol=3,byrow=FALSE)
for (n in 1:300) {
  pred_Sales_mat[n,] = predict(amodel.RL,data.frame(Price=Price_divise[n]),ty
pe="response",interval = "pred",level = .95)

  conf_Sales_mat[n,] = predict(amodel.RL,data.frame(Price=Price_divise[n]),ty
pe="response",interval = "conf",level = .95)

}

# On plot
#Intervalles de confiance.
plot(Price,Sales,main='Intervales de confiance',xlab="Price",ylab="Sales",col
="blue")
abline(amodel.RL, col="red", lwd=4)
```

```
lines(Price_divise, conf_Sales_mat[,3], lty =4, lwd=3, col="green")
lines(Price_divise, conf_Sales_mat[,2], lty =4, lwd=3, col="green")
```

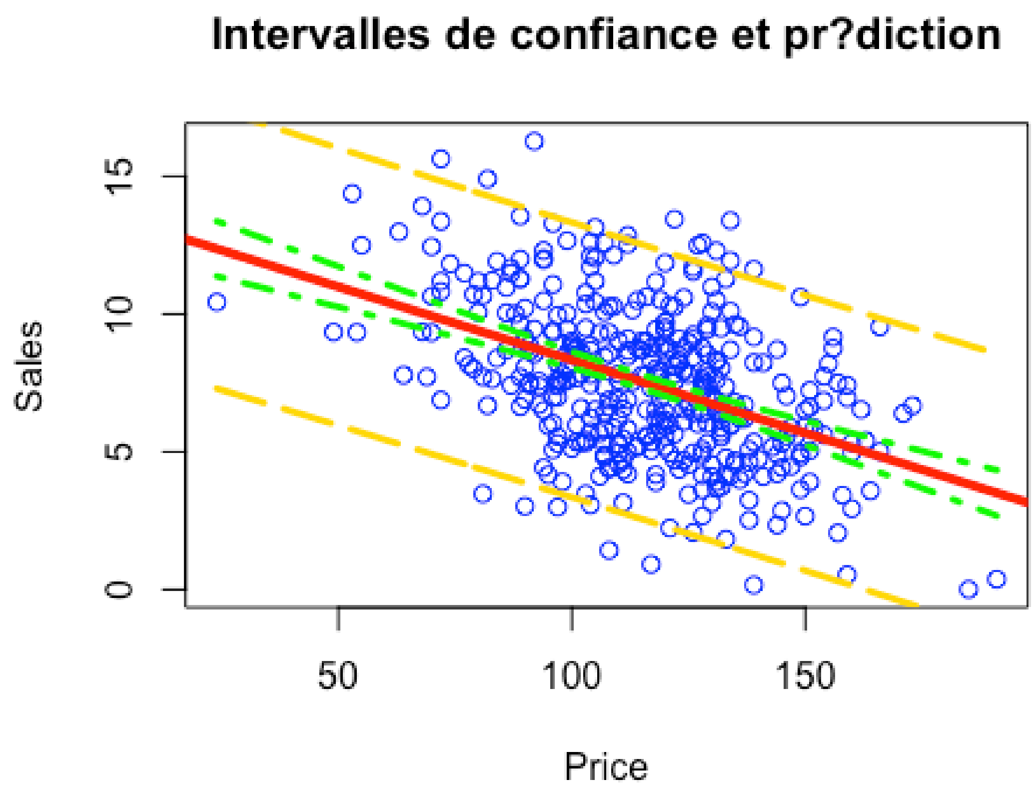



b_3)

```
# Intervalles de prédiction.
plot(Price,Sales,main='Intervalles de prédiction',xlab="Price",ylab="Sales",c
ol="blue")
abline(amodel.RL, col="red", lwd=4)
lines(Price_divise, pred_Sales_mat[,3], lty =5, lwd=3, col="gold")
lines(Price_divise, pred_Sales_mat[,2], lty =5, lwd=3, col="gold")
```




```r
# Intervalles de prédiction et de confiance
plot(Price,Sales,main='Intervalles de confiance et prédiction',xlab="Price",y
lab="Sales",col="blue")
abline(amodel.RL, col="red", lwd=4)
lines(Price_divise, pred_Sales_mat[,3], lty =5, lwd=3, col="gold")
lines(Price_divise, pred_Sales_mat[,2], lty =5, lwd=3, col="gold")
lines(Price_divise, conf_Sales_mat[,3], lty =4, lwd=3, col="green")
lines(Price_divise, conf_Sales_mat[,2], lty =4, lwd=3, col="green")
```




**Partie c**

```
#**************************************C***********************************
Cnewmodel.RL=lm(Sales ~ Price + US + Price*US)
#on peut utiliser Cnewmodel.RL=lm(Sales ~ Price:US)
summary(Cnewmodel.RL)

##
## Call:
## lm(formula = Sales ~ Price + US + Price * US)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9299 -1.6375 -0.0492  1.5765  7.0430
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.974798   0.953079  13.614  < 2e-16 ***
## Price       -0.053986   0.008163  -6.613 1.22e-10 ***
## USYes        1.295775   1.252146   1.035    0.301
## Price:USYes -0.000835   0.010641  -0.078    0.937
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 2.472 on 396 degrees of freedom
## Multiple R-squared:  0.2393, Adjusted R-squared:  0.2335
## F-statistic: 41.52 on 3 and 396 DF,  p-value: < 2.2e-16
```

D'après le tableau (p-value) on remarque que la variable Prise est significatif ((p-value < 0,05)) par contre les variables US et Price:US ne sont pas Significatives (p-value > 0,05) donc on peut conclure que les variables US et Price*US n'influent pas sur le système donc on peut représenter les ventes (Sales) en fonction du prix (Price) par une seule équation (Sales=fct(Price))

**Partie d**

```
#**************d**************
# le modéle
dmodel.RL_7var=lm(Sales ~ Advertising + Age + CompPrice + Education + Income
+ Population + Price)
summary(dmodel.RL_7var)

##
## Call:
## lm(formula = Sales ~ Advertising + Age + CompPrice + Education +
##     Income + Population + Price)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0598 -1.3515 -0.1739  1.1331  4.8304
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.7076934  1.1176260   6.896 2.15e-11 ***
## Advertising  0.1308637  0.0151219   8.654  < 2e-16 ***
## Age         -0.0449743  0.0060083  -7.485 4.75e-13 ***
## CompPrice    0.0939149  0.0078395  11.980  < 2e-16 ***
## Education   -0.0399844  0.0371257  -1.077 0.282142
## Income       0.0128717  0.0034757   3.703 0.000243 ***
## Population  -0.0001239  0.0006877  -0.180 0.857092
## Price       -0.0925226  0.0050521 -18.314  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.929 on 392 degrees of freedom
## Multiple R-squared:  0.5417, Adjusted R-squared:  0.5335
## F-statistic: 66.18 on 7 and 392 DF,  p-value: < 2.2e-16
```

d_1)

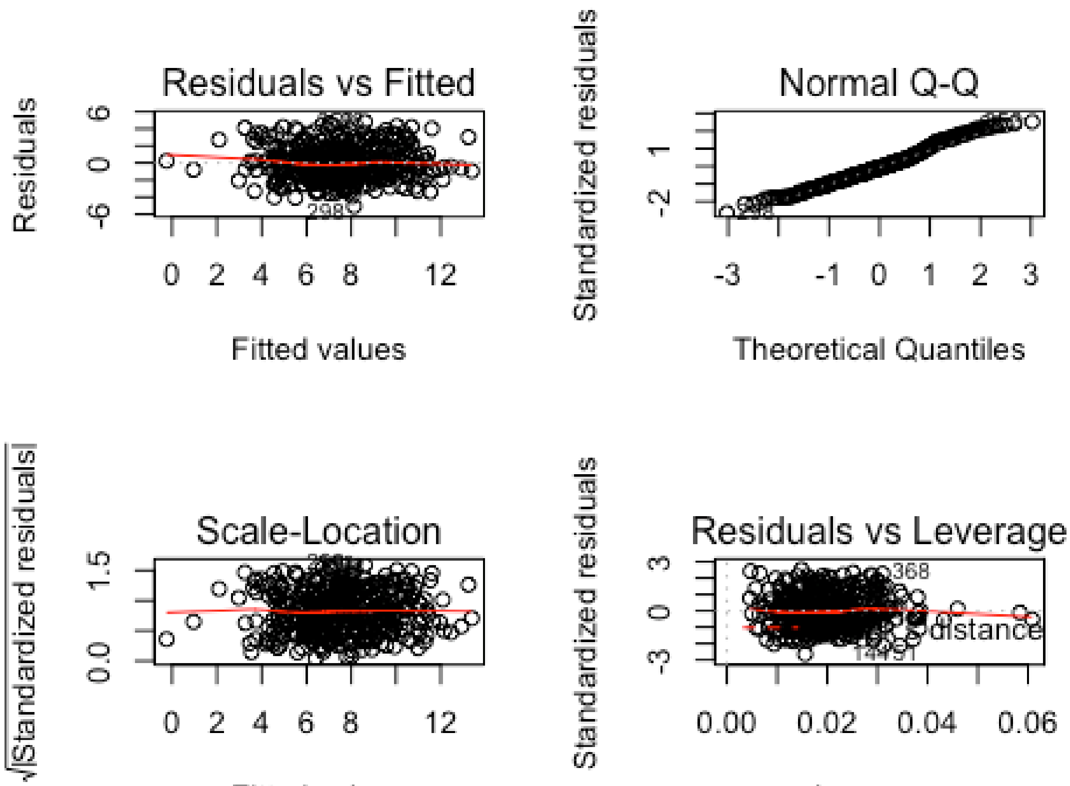
Les variables qui ne contribuent pas significativement au modèle (Pr(>|t|) > 0,5):

- Education

- Population

d_2)

```
# les graphiques diagnostiques des résidus
par(mfrow=c(2,2))
plot(dmodel.RL_7var)
```




Residuals vs Fitted

On remarque que les valeurs sont distribuées uniformément dans les deux cotés de la ligne horizontale, c'est un bon indice qu'on n'a pas la relation non linéaire.

Normal Q-Q

On remarque que les résidus sont bien alignés sur la ligne pointillée droite (les résidus sont bien distribués)

Scale-Location

On remarque qu'il y a une ligne horizontale avec des points également (aléatoirement) répartis. les résidus sont répartis de manière égale le long des plages de prédicteurs

D'après les graphes précédents on remarque qu'il y a des points aberrants ce qui montre que les graphiques des résidus indiquent une anomalie

Residuals vs Leverage

On remarque qu'il n'y a pas des points hors la 'Cookés distance' ce qui montre qu'il n'y a pas des points influents

d_3)

méthode 1 avec Adjusted R-squared

Pour trouver le meilleur modèle nous avons essayé les différents cas possibles et nous avons essayé de pris le modèle qui a la plus grande valeur de Adjusted R-squared

| | R-squared | Adjusted R-squared |
|---|---|---|
| avec Population et sans Education | 0,5403 | 0.5333 |
| sans Population et avec Education | 0.5416 | 0.5346 |
| sans Population et sans Education | 0.5403 | 0.5345 |
| avec Population et avec Education | 0,5417 | 0,5335 |

D'après le tableau, on voit que les  Adjusted R-squared sont très proches donc c'est difficiles a conclure le meilleur modèle. Donc on utilise anova

Methode 2 avec ANOVA

```
# model 0: just avec Price
 dmodel.RL_0=lm(Sales ~  Price)
# model 1: avec  Population et avec Education
 dmodel.RL_1=lm(Sales ~ Advertising + Age + CompPrice + Education + Income +
Population + Price)
# Model 2: sans  Population et sans Education
 dmodel.RL_2=lm(Sales ~ Advertising + Age + CompPrice  + Income  + Price)
# Model 3: sans  Population et avec Education
 dmodel.RL_3=lm(Sales ~ Advertising + Age + CompPrice + Education + Income  +
Price)
# Model 4: avec  Population et sans Education
 dmodel.RL_4=lm(Sales ~ Advertising + Age + CompPrice + Income + Population +
Price)

# Compare model 0 to model 1
 anova( dmodel.RL_0,  dmodel.RL_1)

## Analysis of Variance Table
##
## Model 1: Sales ~ Price
## Model 2: Sales ~ Advertising + Age + CompPrice + Education + Income +
##     Population + Price
```

```
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    398 2552.2
## 2    392 1458.6  6    1093.7 48.989 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Compare model 0 to model 2
 anova( dmodel.RL_0,  dmodel.RL_2)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ Price
## Model 2: Sales ~ Advertising + Age + CompPrice + Income + Price
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    398 2552.2
## 2    394 1462.9  4    1089.3 73.348 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
 # Compare model 0 to model 3
 anova( dmodel.RL_0,  dmodel.RL_3)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ Price
## Model 2: Sales ~ Advertising + Age + CompPrice + Education + Income +
##     Price
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    398 2552.2
## 2    393 1458.7  5    1093.6 58.926 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
 # Compare model 0 to model 4
 anova( dmodel.RL_0,  dmodel.RL_4)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ Price
## Model 2: Sales ~ Advertising + Age + CompPrice + Income + Population +
##     Price
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    398 2552.2
## 2    393 1462.9  5    1089.4 58.531 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Compare model 2 to model 3
 anova( dmodel.RL_2,  dmodel.RL_3)
```

```
## Analysis of Variance Table
##
```

```
## Model 1: Sales ~ Advertising + Age + CompPrice + Income + Price
## Model 2: Sales ~ Advertising + Age + CompPrice + Education + Income +
##     Price
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    394 1462.9
## 2    393 1458.7  1    4.2143 1.1354 0.2873
```

```
# Compare model 2 to model 4
 anova( dmodel.RL_2,  dmodel.RL_4)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ Advertising + Age + CompPrice + Income + Price
## Model 2: Sales ~ Advertising + Age + CompPrice + Income + Population +
##     Price
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    394 1462.9
## 2    393 1462.9  1  0.019196 0.0052 0.9428
```

Au début on a comparé le modèle 0 (Sales=fct(Price)) avec les autres 4 modèles présenté dans le tableau précédant et on a trouvé (p-valeu=`2.2e-16`) donc on peut remplacer le modèle 0 par l'un les autres modèles. Et on a aussi faire la comparaison entre le modèle 'sans Population et sans Education' avec 'avec Population et sans Education ' et avec 'sans Population et sans Education'

Et on trouvé (p-valeu1=`0.2873` , p=valeu2=`0.9428`) `donc le` 'sans Population et sans Education' mieux

Et on a trouvé aussi la plus grande valeur de F= 73.348 pour le modèle 'sans Population et sans Education'

D'après les résultat précédant on conclure que le meilleur modèle est « *`sans Population et sans Education`* »

On crée le nouveau model

```
# le modéle sans  Population et sans Education
dmodel.RL_7var=lm(Sales ~ Advertising + Age + CompPrice  +  Income + Price)
summary(dmodel.RL_7var)
```

```
##
## Call:
## lm(formula = Sales ~ Advertising + Age + CompPrice + Income +
##     Price)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9071 -1.3081 -0.1892  1.1495  4.6980
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.109190   0.943940   7.531 3.46e-13 ***
## Advertising  0.130611   0.014572   8.963  < 2e-16 ***
```

```
## Age          -0.044971   0.005994  -7.503 4.20e-13 ***
## CompPrice     0.093904   0.007792  12.051  < 2e-16 ***
## Income        0.013092   0.003465   3.779 0.000182 ***
## Price        -0.092543   0.005044 -18.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.927 on 394 degrees of freedom
## Multiple R-squared:  0.5403, Adjusted R-squared:  0.5345
## F-statistic: 92.62 on 5 and 394 DF,  p-value: < 2.2e-16
```

Question N°2

**Partie a**

```
remove(list=ls())
# On importe les données
data=read.csv("C:/Users/defaultuser0.DESKTOP-I1A8N1U.000/Desktop/Auto18/MTH63
12/d2/Q2/Equipement.csv")

#*******************a*******************

# Convertir "Y" à numeric (1 , 0)
data$Y=as.numeric(data$Y == 'D')

# On sépare les données en deux : d'entrainement et de test
training_d=data[1:170,1:3]
test_d=data[171:250,1:3]

#modéle de régression linéaire

attach(training_d)
model.RL=lm( Y~ X1 + X2)
summary(model.RL)

##
## Call:
## lm(formula = Y ~ X1 + X2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.62134 -0.20504 -0.01665  0.19418  0.63501
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.06195    0.23703  -0.261    0.794
## X1          -0.11199    0.01152  -9.720   <2e-16 ***
## X2           0.12072    0.01195  10.099   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2763 on 167 degrees of freedom
## Multiple R-squared:  0.6999, Adjusted R-squared:  0.6963
## F-statistic: 194.8 on 2 and 167 DF,  p-value: < 2.2e-16

detach(training_d)



#Predict
predictions=predict(model.RL,newdata=test_d,type="response")
# Fit to 0 or 1 (1=D, 0=N)
predictions = ifelse(predictions > 0.5,1,0)
# Taux d'erreur
error_RL<- mean(predictions != test_d$Y)
print(paste('le taux d’erreur pour la régression linéaire =',error_RL))

## [1] "le taux d’erreur pour la régression linéaire = 0.1125"
```

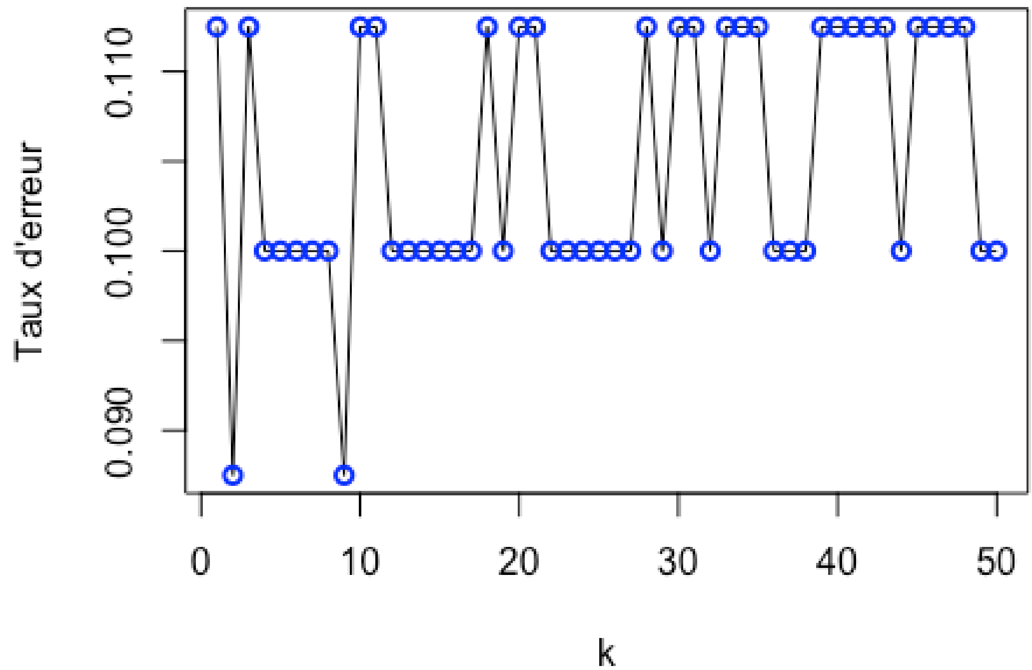
**Partie b**

```
#************************************b******************************
#************************************KNN*******************************

# #####ici nous ne normalisons pas les données puisque celles-ci ont la méme
unité

data_entr<-data[1:170,c(1,2)]
data_test<-data[171:250,c(1,2)]

# On extrait les classes (étiquettes) des données d'entraénement
# (nécessaires pour knn)
# et celles des données de test
class_entr<-data[1:170,3]
class_test<-data[171:250,3]

# L'algorithme knn fait partie de la librarie "class"
library(class)

######################################error_knn_mat=matrix(nrow=1, ncol=50)
set.seed(50)
#vv=round(runif(50, min=1 , max=70), digits=0)
k=50

taux_err <- rep(0,k)

for (h in 1:k) {
```

```r
   model_knn=knn(train=data_entr,test=data_test,cl=class_entr,k=h)

   taux_err[h] = mean(model_knn != class_test)

}

#print(taux_err)
cat(sprintf("Taux d'erreur (k = %s): %s\n",1:k,taux_err))
```

```
## Taux d'erreur (k = 1): 0.1125
##  Taux d'erreur (k = 2): 0.0875
##  Taux d'erreur (k = 3): 0.1125
##  Taux d'erreur (k = 4): 0.1
##  Taux d'erreur (k = 5): 0.1
##  Taux d'erreur (k = 6): 0.1
##  Taux d'erreur (k = 7): 0.1
##  Taux d'erreur (k = 8): 0.1
##  Taux d'erreur (k = 9): 0.0875
##  Taux d'erreur (k = 10): 0.1125
##  Taux d'erreur (k = 11): 0.1125
##  Taux d'erreur (k = 12): 0.1
##  Taux d'erreur (k = 13): 0.1
##  Taux d'erreur (k = 14): 0.1
##  Taux d'erreur (k = 15): 0.1
##  Taux d'erreur (k = 16): 0.1
##  Taux d'erreur (k = 17): 0.1
##  Taux d'erreur (k = 18): 0.1125
##  Taux d'erreur (k = 19): 0.1
##  Taux d'erreur (k = 20): 0.1125
##  Taux d'erreur (k = 21): 0.1125
##  Taux d'erreur (k = 22): 0.1
##  Taux d'erreur (k = 23): 0.1
##  Taux d'erreur (k = 24): 0.1
##  Taux d'erreur (k = 25): 0.1
##  Taux d'erreur (k = 26): 0.1
##  Taux d'erreur (k = 27): 0.1
##  Taux d'erreur (k = 28): 0.1125
##  Taux d'erreur (k = 29): 0.1
##  Taux d'erreur (k = 30): 0.1125
##  Taux d'erreur (k = 31): 0.1125
##  Taux d'erreur (k = 32): 0.1
##  Taux d'erreur (k = 33): 0.1125
##  Taux d'erreur (k = 34): 0.1125
##  Taux d'erreur (k = 35): 0.1125
##  Taux d'erreur (k = 36): 0.1
##  Taux d'erreur (k = 37): 0.1
##  Taux d'erreur (k = 38): 0.1
##  Taux d'erreur (k = 39): 0.1125
##  Taux d'erreur (k = 40): 0.1125
##  Taux d'erreur (k = 41): 0.1125
##  Taux d'erreur (k = 42): 0.1125
##  Taux d'erreur (k = 43): 0.1125
##  Taux d'erreur (k = 44): 0.1
##  Taux d'erreur (k = 45): 0.1125
##  Taux d'erreur (k = 46): 0.1125
##  Taux d'erreur (k = 47): 0.1125
##  Taux d'erreur (k = 48): 0.1125
##  Taux d'erreur (k = 49): 0.1
##  Taux d'erreur (k = 50): 0.1
```

```
#plot(taux_err)
plot (taux_err,type="l",xlab="k",ylab="Taux d'erreur")
points (1:k,taux_err,lwd=2,col="blue")
```




D’après la courbe du taux d’erreur en fonction k on a trouvé que k optimal égal 2 #k optimal égal 2 le taux d’erreur pour KNN (k=2) 0.0875

**Partie c**

```
#*******************C*********************************
#  la régression logistique

attach(training_d)
model.log<-glm (Y ~ X1 + X2, family=binomial,data=data_entr)
summary(model.log)

##
## Call:
## glm(formula = Y ~ X1 + X2, family = binomial, data = data_entr)
##
## Deviance Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -2.28872  -0.08710  -0.00031   0.05736   2.08848
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.4020     5.5064  -1.889   0.0589 .
## X1           -1.9516     0.4589  -4.253 2.11e-05 ***
## X2            2.1710     0.4943   4.392 1.12e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 235.670  on 169  degrees of freedom
## Residual deviance:  43.711  on 167  degrees of freedom
## AIC: 49.711
##
## Number of Fisher Scoring iterations: 8
```

```
detach(training_d)
```

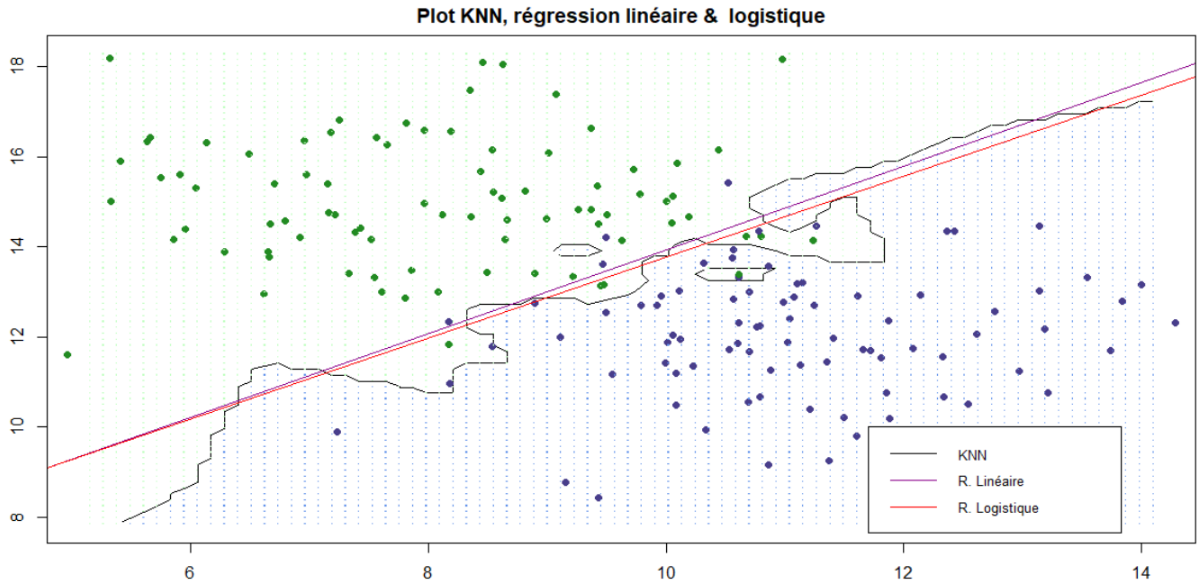
```
#Predict
predictions_log=predict(model.log,newdata=test_d,type="response")
# Fit to 0 or 1 (1=D, 0=N)
predictions_log = ifelse(predictions_log > 0.5,1,0)
# Taux d'erreur
error_log<- mean(predictions_log != test_d$Y)
print(paste('le taux d’erreur pour la régression logistique=',error_log))
```

```
## [1] "le taux d’erreur pour la régression logistique= 0.1125"
```

## Partie d

```
#*************************************d*****************************************
***
####### paramétres de l'equation RL #######
attach(test_d)

# pour trouver l'équation X2=fct(X1), on a resoudre l'équation Y=S => f(X1,X2
)=0,5
# b0+b1*X1+b2*X2=0,5  =>  X2=(0,5-b0)/b2 +(-b1/b2)*X1 = C_RL + A_RL*X1
C_RL <- -(model.RL$coef[1] - 0.5) / model.RL$coef[3]
A_RL <- - model.RL$coef[2] / model.RL$coef[3]


#######paramétres de l'equation log#######
# pour trouver l'équation X2=fct(X1), on a resoudre l'équation Y=S => f(X1,X2
)=0
# b0+b1*X1+b2*X2=0  =>  X2=(-b0)/b2 +(-b1/b2)*X1 = C_log + A_log*X1
C_log <- -(model.log$coef[1]) / model.log$coef[3]
```

```r
A_log <- - model.log$coef[2] / model.log$coef[3]

detach(test_d)

#####  knn ######
library(class)
#on trace la plateforme de knn
minx1= min(data_test[,1])
maxx1= max(data_test[,1])
x1 <- seq(from = minx1, to = maxx1,length.out=80)

minx2=min(data_test[,2])
maxx2=max(data_test[,2])
x2 <- seq(from = minx2, to = maxx2,length.out=80)

gd <- expand.grid(x1 = x1, x2 = x2)

#On calcule le modéle knn avec K optimal 2.
model.knn_opt=knn(train=data_entr,test=gd,cl=class_entr,k=2,prob=TRUE)
probabilite <- attr(model.knn_opt, "prob")
probabilite <- ifelse(model.knn_opt=="1", probabilite, 1-probabilite)
probabiliteopt <- matrix(probabilite, length(x1), length(x2))
#############

##### plot knn & RL &  Logic ######


# plot knn
contour(x1, x2, probabiliteopt, levels=0.5, labels="", xlab="", ylab="", main
=
          "Plot KNN, régression linéaire &  logistique")

points(data_entr,pch=19, col=ifelse(class_entr==1, "forestgreen", "darkslateb
lue"))
points(gd, pch=".", cex=1.2, col=ifelse(probabiliteopt >0, "darkseagreen1", "
cornflowerblue"))


# plot RL
abline(C_RL, A_RL, col="darkmagenta" )

# plot log
abline(C_log, A_log, col="red" )

legend(11.7, 10, legend=c("KNN", "R. Linéaire", "R. Logistique"), col=c("blac
k", "darkmagenta","red"), lty=1, cex=0.8)

box()
```




**Partie e**

```
#*************************e***********************

# On introduit les valeurs.
class_c=data.frame(matrix(c(9.5,13.5),nrow=1,ncol=2))
colnames(class_c)=c("X1","X2")

# On classifie avec régression lineaire.
predictionLR=predict(model.RL,class_c,type="response")
predictionLR=ifelse(predictionLR > 0.5,"D","N")
cat(sprintf("Le prédiction par régression lineaire est: %s\n",predictionLR))

## Le prédiction par régression lineaire est: D

# On fait la classification avec KNN.
predictionKNN=knn(data_entr,class_c,class_entr,k=8)
predictionKNN=ifelse(predictionKNN==1,"D","N")
cat(sprintf("Le prédiction par KNN est: %s",predictionKNN))

## Le prédiction par KNN est: D

# On fait la classification avec log.
predictionslog=predict(model.log,class_c,type="response")

predictionslog = ifelse(predictionslog > 0.5,"D","N")
cat(sprintf("Le prédiction par glm est: %s",predictionslog))

## Le prédiction par glm est: D
```

On remarque que la précision lorsqu'on a utilisé KNN, est supérieure à la précision de régression linéaire et régression logistique. Le taux d'erreur de la 1er est de 0.0875 et le taux d'erreur de la 2eme et la 3eme sont 0,1125.

On remarque aussi que la valeur obtenue est la même: “D” c'est a dire on a la « présence d'anomalies ».