devoir4

Lyes Heythem BETTACHE

14 novembre 2018

QUESTION N°1

a) Séparation des données.

Diviser les données en deux groupes qui serviront pour toutes les questions qui suivent. Pour cela, utiliser la fonction sample() de R afin de choisir un échantillon (sans remise) de taille 200 pour les données d'entraînement, et le reste des données (200) constituera les données de test.

```
remove(list=ls())

# On charge le packet ISLR et attache le dataframe Carseats

set.seed(1923715)
# On importe et prepare le dataframe.
remove(list=ls())
library(ISLR)
attach(Carseats)

# On selectionne un ensemble de donnees train et test.
# Par default est sans remise (replace=FALSE)
list=sample(seq_len(nrow(Carseats)), 200)

train=Carseats[list, ]
test=Carseats[-list, ]
```

b) Approche par meilleurs sous ensembles.

1. Avec les données d'entraînement, utiliser la fonction regsubset() de R pour déterminer les meilleurs modèles linéaires de l variables (selon R2 ajusté), notés Ml; ` l= 0;...; 11.

Déterminer ensuite le meilleur des 12 modèles par validation croisée.

```
#partie 1 Qb (cours chp6)
set.seed(1923715)

# On importe leaps et caret
library(leaps)
```

```r
#library(caret)
regfitfull.Carseats=regsubsets(Sales~.,data=train, nvmax=11)
reg.summary=summary(regfitfull.Carseats)
reg.summary
```

```
## Subset selection object
## Call: regsubsets.formula(Sales ~ ., data = train, nvmax = 11)
## 11 Variables  (and intercept)
##                 Forced in Forced out
## CompPrice           FALSE      FALSE
## Income              FALSE      FALSE
## Advertising         FALSE      FALSE
## Population          FALSE      FALSE
## Price               FALSE      FALSE
## ShelveLocGood       FALSE      FALSE
## ShelveLocMedium     FALSE      FALSE
## Age                 FALSE      FALSE
## Education           FALSE      FALSE
## UrbanYes            FALSE      FALSE
## USYes               FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: exhaustive
##          CompPrice Income Advertising Population Price ShelveLocGood
## 1  ( 1 ) " "       " "    " "         " "        " "   "*"
## 2  ( 1 ) " "       " "    " "         " "        "*"   "*"
## 3  ( 1 ) "*"       " "    " "         " "        "*"   "*"
## 4  ( 1 ) "*"       " "    " "         " "        "*"   "*"
## 5  ( 1 ) "*"       " "    " "         " "        "*"   "*"
## 6  ( 1 ) "*"       " "    "*"         " "        "*"   "*"
## 7  ( 1 ) "*"       "*"    "*"         " "        "*"   "*"
## 8  ( 1 ) "*"       "*"    "*"         " "        "*"   "*"
## 9  ( 1 ) "*"       "*"    "*"         " "        "*"   "*"
## 10  ( 1 ) "*"      "*"    "*"         "*"        "*"   "*"
## 11  ( 1 ) "*"      "*"    "*"         "*"        "*"   "*"
##          ShelveLocMedium Age Education UrbanYes USYes
## 1  ( 1 ) " "             " " " "       " "      " "
## 2  ( 1 ) " "             " " " "       " "      " "
## 3  ( 1 ) " "             " " " "       " "      " "
## 4  ( 1 ) " "             "*" " "       " "      " "
## 5  ( 1 ) "*"             "*" " "       " "      " "
## 6  ( 1 ) "*"             "*" " "       " "      " "
## 7  ( 1 ) "*"             "*" " "       " "      " "
## 8  ( 1 ) "*"             "*" "*"       " "      " "
## 9  ( 1 ) "*"             "*" "*"       " "      "*"
## 10  ( 1 ) "*"            "*" "*"       " "      "*"
## 11  ( 1 ) "*"            "*" "*"       "*"      "*"
```

```r
names(reg.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

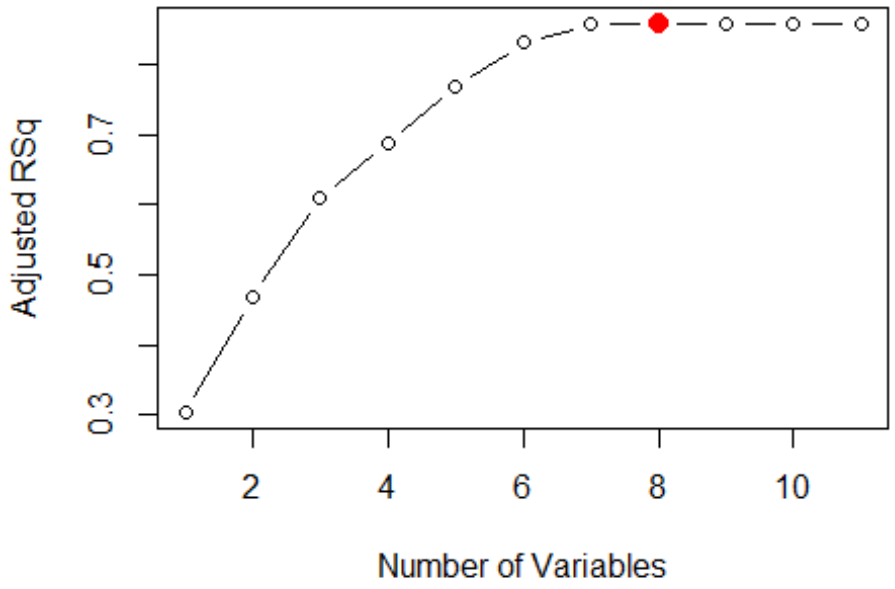
```
reg.summary$adjr2
```

```
##  [1] 0.3034154 0.4670303 0.6086889 0.6874679 0.7693322 0.8319403 0.8591644
##  [8] 0.8594593 0.8592152 0.8585707 0.8578213
```

```
which.max(reg.summary$adjr2)
```

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

```
#par(mfrow=c(2,2))
#plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="b
")
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.summary$a
djr2)], col="red",cex=2,pch=20)
```




```
#partie 2 Qb  (cours chp6)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(leaps)

set.seed(1923715)

#On cree la fonction pour faire la prédiction (cours chp6)
```

```r
predict.regsubsets<-function(object,newdata,id)
{
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi

}

k<-10
folds=createFolds(list,k=10)
cv.errors=matrix(NA,k,11)


 #on trouve l'erreur pour M0
for (i in 1:k){
predpourM0=mean(train$Sales[-folds[[i]]])
errorsM0=mean((train$Sales[folds[[i]]] - predpourM0)^2)
}


# on trouve l'erreur pour M1-->M11

for(j in 1:k){
        best.fit<-regsubsets(Sales~.,train[-folds[[j]], ] ,nvmax=11)
  for(i in 1:11){
            pred<-predict.regsubsets(best.fit,train[folds[[j]],],id=i)
          cv.errors[j,i]<-mean( (train$Sales[folds[[j]]]-pred)^2)
}
}

# On calcule l'erreur pour M1-->M11
mean.cv.errorsM1M11=apply(cv.errors,2,mean)

# l'erreur
mean.cv.errors=c(errorsM0,mean.cv.errorsM1M11)

min.errors=which.min(mean.cv.errors)-1
min.errors
```

```
## [1] 7
```

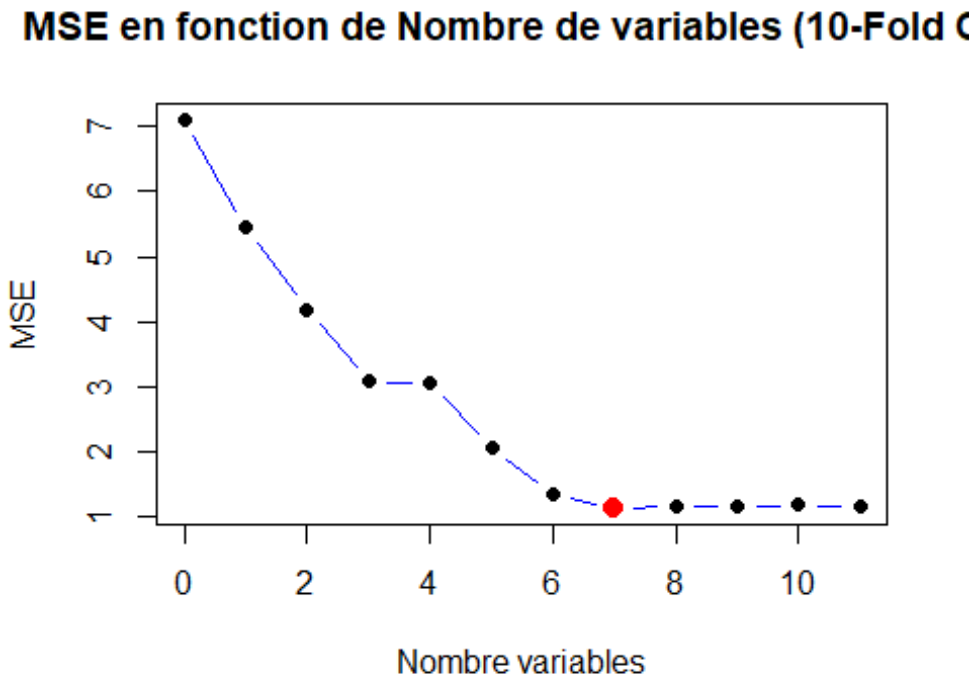
```r
regfit.best=regsubsets(Sales~.,data=train, nvmax=11)
cat("\n  le meilleur modèle retenu (avec l'approche du meilleur sous-ensemble
) 7 variable est: \n")
```

```
##
##   le meilleur modèle retenu (avec l'approche du meilleur sous-ensemble) 7
variable est:
```

```
coef(regfit.best,min.errors)
```

```
##      (Intercept)        CompPrice           Income      Advertising
##       5.51852723       0.09455821       0.01612939       0.10539079
##            Price    ShelveLocGood ShelveLocMedium              Age
##      -0.09607542       4.89862286       2.08640042      -0.05097069
```

```
plot(seq(0,11),mean.cv.errors,type="b",xlab="Nombre variables",ylab="MSE",mai
n="MSE en fonction de Nombre de variables (10-Fold CV)",col="blue")
points(seq(0,11),mean.cv.errors,col="black",pch=19)
points(min.errors,mean.cv.errors[min.errors+1],col="red",cex=2,pch=20)
```



2. Avec les données de test, évaluer l'erreur de test (i.e. la moyenne des carrés des erreurs de prévision) du meilleur modèle retenu.

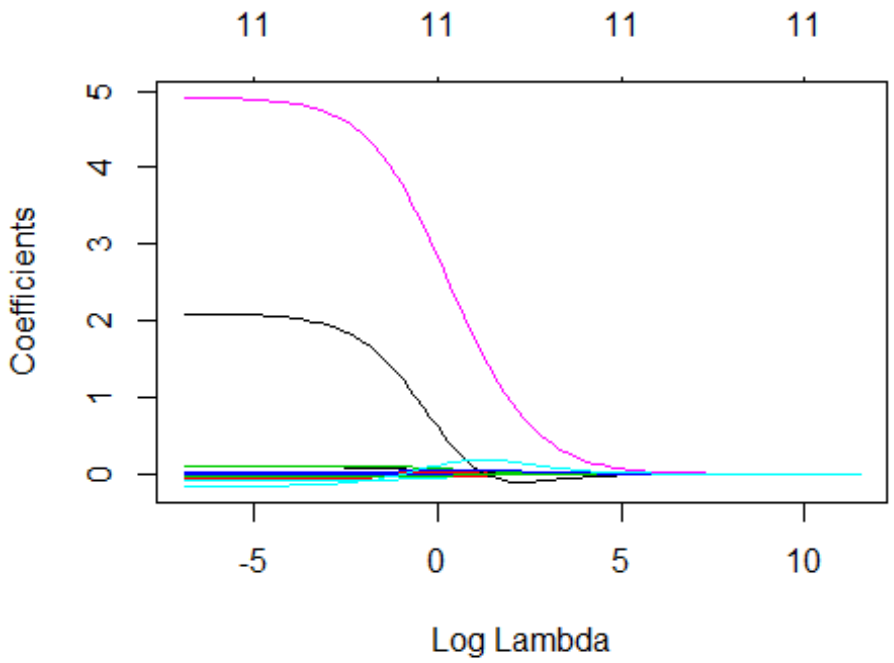
```
set.seed(1923715)

pred.test=predict.regsubsets(best.fit,test,id=min.errors)
erreur.sub=mean((test$Sales - pred.test)^2)

cat(sprintf("\n Erreur MSE test pour le meilleur modèle retenu (avec l'approc
he du meilleur sous-ensemble) 7 variable est: %s\n",erreur.sub))
```

```
##
##  Erreur MSE test pour le meilleur modèle retenu (avec l'approche du meille
ur sous-ensemble) 7 variable est: 1.06814784826757
```

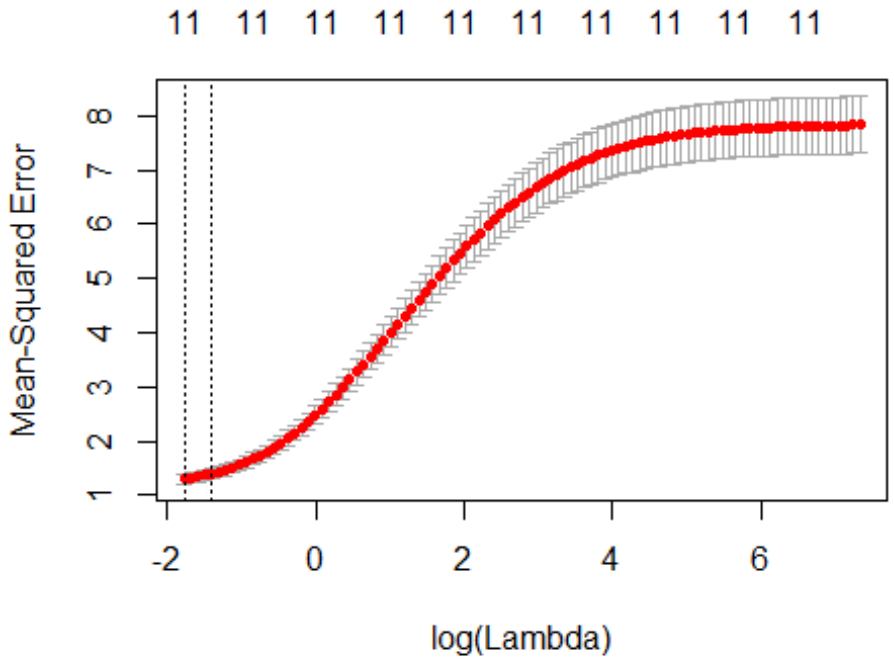
c) Approche Ridge.

Avec les données d'entraînement, ajuster un modèle de régression ridge (fonction glmnet() de R) et produire un graphique similaire à celui de la partie gauche de la figure 6.4 page 216 dans ISL. Déterminer ensuite la valeur appropriée de lamda (modèle optimal) par validation croisée.

```
set.seed(1923715)
library(glmnet)

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16

x.train=model.matrix(Sales~. ,train)[,-1]
y.train=train$Sales
grid=10^seq(5,-3,length=100)

# Modele Ridge
ridge.Carseats=glmnet(x.train,y.train,alpha=0,lambda=grid)

# Plot
plot(ridge.Carseats,xvar="lambda")
```



```r
# Validation croisse
cv.ridge=cv.glmnet(x.train,y.train,alpha=0,nfolds=10)
plot(cv.ridge)
```




```r
# On obtien l'erreur minimum de cv
erreur.cv.ridge=min(cv.ridge$cvm)

# On calcule le Lambda optimal
lambda.ridge=cv.ridge$lambda.min
cat(sprintf("\n Valeur optimale de lambda: %s",lambda.ridge))

##
##  Valeur optimale de lambda: 0.169353499734749
```

2. Avec les données de test, évaluer l'erreur de test (i.e. la moyenne des carrés des erreurs de prévision) du modèle optimal de la méthode ridge.

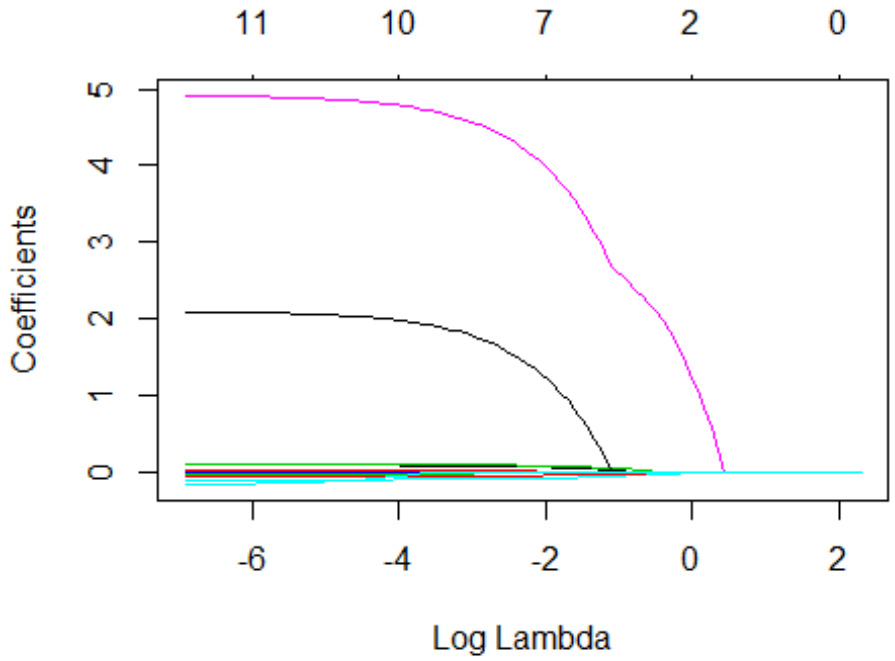
```r
x.test=model.matrix(Sales~. ,test)[,-1]
y.test=test$Sales

ridge.pred=predict(ridge.Carseats,s=lambda.ridge,newx=x.test)

erreur.ridge=mean((ridge.pred-y.test)^2)
cat(sprintf("\n l'erreur MSE test pour le modèle optimal de la méthode ridge(
lambda: 0.169353499734749): %s",erreur.ridge))
```

```
##
##  l'erreur MSE test pour le modèle optimal de la méthode ridge(lambda: 0.16
9353499734749): 1.08554989426633
```

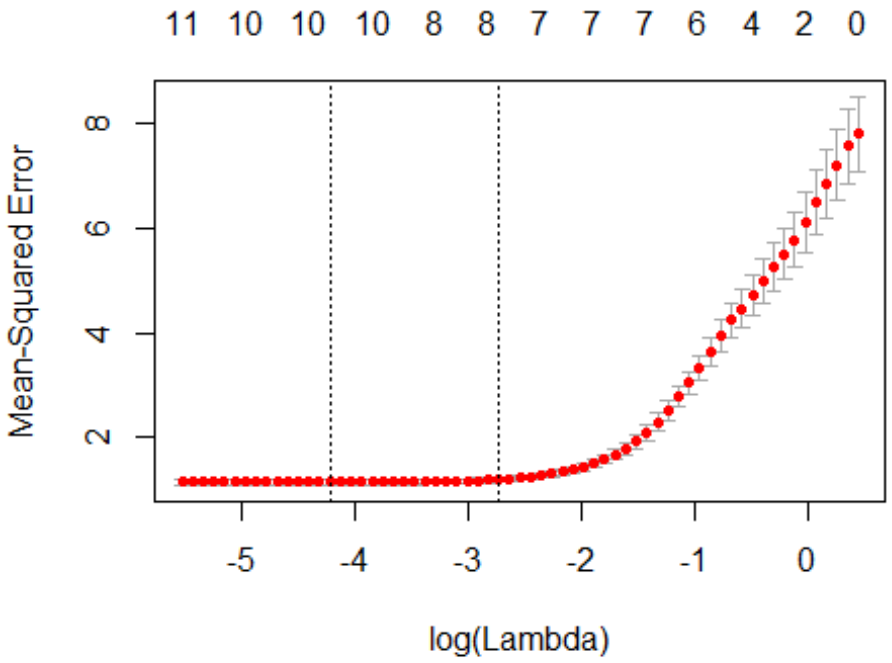
d) Approche Lasso.

1. Avec les données d'entraînement, ajuster un modèle de régression lasso (fonction glmnet() de R) et produire un graphique similaire à celui de la partie gauche de la figure 6.6 page 220 dans ISL. Déterminer ensuite la valeur appropriée de lamda (modèle optimal) par validation croisée.

```
# On le fait comme l'approche de Ridge
set.seed(1923715)

# On utilise les memes valeurs de 'x.train','y.train' crees pour Ridge
# Modele Lasso
grid=10^seq(1,-3,length=100)
lasso.Carseats=glmnet(x.train,y.train,alpha=1,lambda=grid)


# Plot
plot(lasso.Carseats,xvar="lambda")
```




```
# Validation croisse
cv.lasso=cv.glmnet(x.train,y.train,alpha=1,nfolds=5)
plot(cv.lasso)
```




```
# On obtien l'erreur minimum de cv
erreur.cv.lasso=min(cv.lasso$cvm)

# On fait le calcul pour le lambda optimale
lambda.lasso=cv.lasso$lambda.min
cat(sprintf("\n Valeur optimale de lambda: %s",lambda.lasso))

##
##  Valeur optimale de lambda: 0.0147295036871539
```

2. Avec les données de test, évaluer l'erreur de test (i.e. la moyenne des carrés des erreurs de prévision) du modèle optimal de la méthode ridge.

```
# On utilise les memes valeurs de 'x.test' et 'y.test' crees pour Ridge
lasso.pred=predict(lasso.Carseats,s=lambda.lasso,newx=x.test)

erreur.lasso=mean((lasso.pred-y.test)^2)
cat(sprintf("\n Taux d'erreur MSE pour les donnees de test avec Lasso (lambda
: 0.0147295036871539): %s",erreur.lasso))

##
##  Taux d'erreur MSE pour les donnees de test avec Lasso (lambda: 0.01472950
36871539): 1.00201336934613
```

e) Approche composantes principales.

1. Avec les données d'entraînement, utiliser la fonction pcr() de R pour déterminer les modèles linéaires avec M composantes principales. Déterminer ensuite la valeur optimale de M (meilleur modèle) par validation croisée.

```
set.seed(1923715)
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

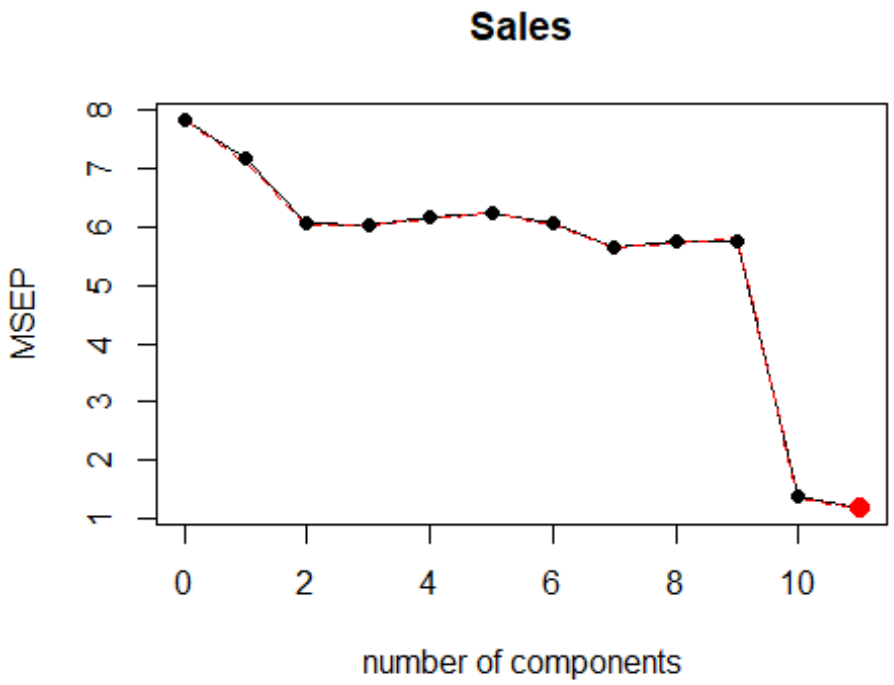
```
# Creation du modele
# On utilise segments=10 pour le 10-CV
pcr.Carseats=pcr(Sales~.,data=train,scale=TRUE,validation="CV")
summary(pcr.Carseats)
```

```
## Data:    X dimension: 200 11
##  Y dimension: 200 1
## Fit method: svdpc
## Number of components considered: 11
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           2.799    2.676    2.462    2.459    2.481    2.498    2.460
## adjCV        2.799    2.668    2.455    2.454    2.479    2.501    2.453
##        7 comps  8 comps  9 comps  10 comps  11 comps
## CV       2.381    2.398    2.397     1.182     1.092
## adjCV    2.375    2.393    2.410     1.160     1.088
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        17.64    33.48    46.73    57.01     66.5    75.64    83.90
## Sales    12.32    25.59    26.39    26.73     27.2    30.01    34.03
##        8 comps  9 comps  10 comps  11 comps
## X        90.95    94.46     97.45    100.00
## Sales    34.11    34.44     84.88     86.57
```

```
# Validation
validationplot(pcr.Carseats,val.type="MSEP")
```

```
pcr.p=MSEP(pcr.Carseats,estimate="CV")
min.erreur=which.min(pcr.p$val)
```

```
min.erreur=min.erreur-1

points(seq(0,11),pcr.p$val,col="black",pch=19)
points(min.erreur,min(pcr.p$val),col="red",cex=2,pch=20)
```




```
# Resultat
cat(sprintf("\n\n Le meilleur resultat est obtenu avec %s composants",min.err
eur))

##
##
##  Le meilleur resultat est obtenu avec 11 composants
```

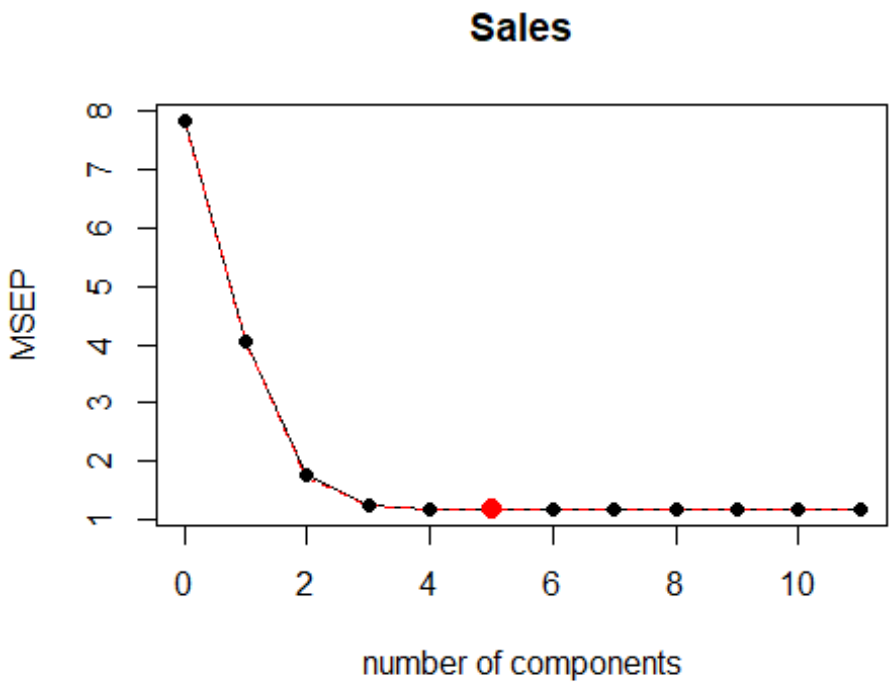
2. Avec les données de test, évaluer l'erreur de test (i.e. la moyenne des carrés des erreurs de prévision) du meilleur modèle retenu.

```
pcr.pred=predict(pcr.Carseats,x.test,ncomp=11)
erreur.pcr=mean((pcr.pred-y.test)^2)

cat(sprintf("\n Taux d'erreur MSE pour les donnees de test avec 11 composants
PCR: %s",erreur.pcr))

##
##  Taux d'erreur MSE pour les donnees de test avec 11 composants PCR: 1.0081
1149445217
```

f) Approche moindres carrés partiels.

1. Avec les données d'entraînement, utiliser la fonction plsr() de R pour ajuster les modèles linéaires de M composantes (directions). Déterminer ensuite la valeur optimale de M (meilleur modèle) par validation croisée.

```
set.seed(1923715)

# Creation du modÃ¨le et CV avec K=10
pls.Carseats=plsr(Sales~.,data=train,scale=TRUE,validation="CV")
summary(pls.Carseats)

## Data:    X dimension: 200 11
##  Y dimension: 200 1
## Fit method: kernelpls
## Number of components considered: 11
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           2.799    2.016    1.334    1.121    1.090    1.090    1.091
## adjCV        2.799    2.009    1.316    1.116    1.086    1.086    1.088
##        7 comps  8 comps  9 comps  10 comps  11 comps
## CV       1.092    1.092    1.092     1.092     1.092
## adjCV    1.088    1.088    1.088     1.088     1.088
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        15.41    22.13    32.97    45.07    56.12    63.37    70.14
## Sales    52.42    80.94    85.91    86.55    86.56    86.57    86.57
##        8 comps  9 comps  10 comps  11 comps
## X        78.09    83.87     90.64    100.00
## Sales    86.57    86.57     86.57     86.57

# Validation
validationplot(pls.Carseats,val.type="MSEP")

pls.p=MSEP(pls.Carseats,estimate="CV")
min.erreur=which.min(pls.p$val)
min.erreur=min.erreur-1

points(seq(0,11),pls.p$val,col="black",pch=19)
points(min.erreur,min(pls.p$val),col="red",cex=2,pch=20)
```



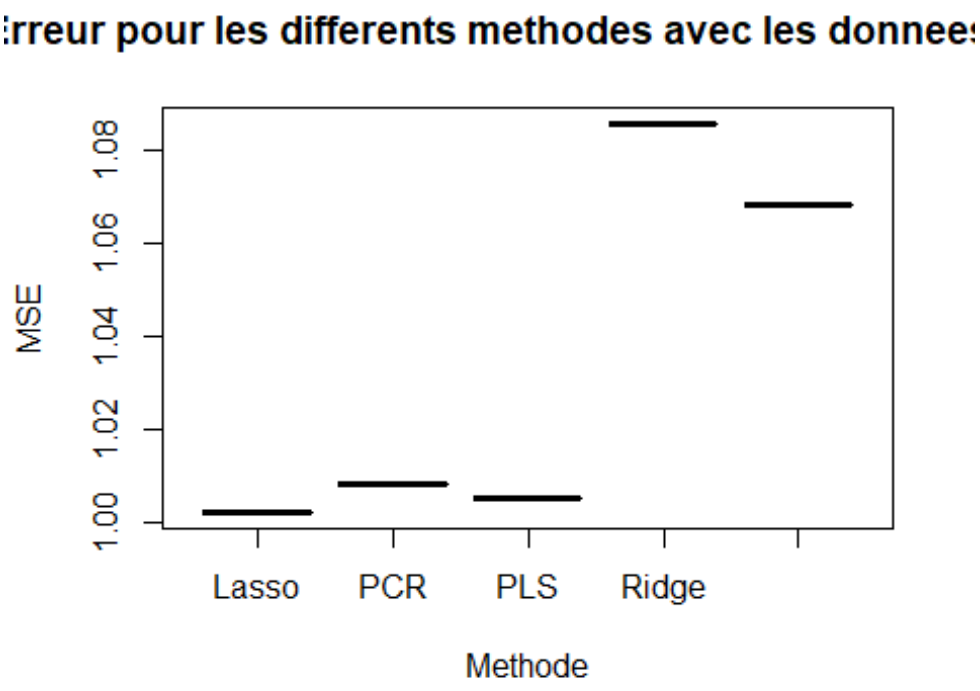

```
# Resultat
cat(sprintf("\n\n Le meilleur resultat est obtenu avec %s composants",min.err
eur))

##
##
##  Le meilleur resultat est obtenu avec 5 composants

pls.pred=predict(pls.Carseats,x.test,ncomp=5)
erreur.pls=mean((pls.pred-y.test)^2)

cat(sprintf("\n Taux d'erreur MSE pour les donnees de test avec 5 composants
PLS: %s",erreur.pls))

##
##  Taux d'erreur MSE pour les donnees de test avec 5 composants PLS: 1.00515
199588435
```

g) Commentaires et conclusion.

Commentaires et conclusion. Commenter sur les résultats obtenus. Y a-t-il une différence importante entre les taux d'erreur des 5 approches (ou méthodes d'apprentissage) ? Quelle modèle utiliseriez-vous pour calculer les prévisions des ventes ? Commenter brièvement.

```
set.seed(1923715)
```

```r
MSE=c(erreur.sub,erreur.ridge,erreur.lasso,erreur.pcr,erreur.pls)
axex<-as.factor(c("Sous-ensembles","Ridge","Lasso","PCR","PLS"))


# Donnees test
plot(axex,MSE,col="blue",xlab="Methode", ylab="MSE", main="Erreur pour les di
fferents methodes avec les donnees test")
```




```r
mse.min=which.min(MSE)
mse.max=which.max(MSE)

cat(sprintf("\n\n Pour les donnees test, l'erreur minimale est = %s et l'erre
ur maximale est = %s",MSE[mse.min], MSE[mse.max]))
```

```
##
##
##  Pour les donnees test, l'erreur minimale est = 1.00201336934613 et l'erre
ur maximale est = 1.08554989426633
```

La difference entre l'erreur minimale est = 1.00201336934613 et l'erreur maximale est = 1.08554989426633 est : 0.08353652 pour les données test. C'est un resultat acceptable, et on peut dire que les autres méthodes peuvent fonctionner, surtout Lasso, PCR et PLS ou il n'y a pas une grande difference entre l'erreur .

La méthode qui a fonctioné mieux pour les données test est le Lasso ou on a MSE=1.00201.

pour calculer les prévisions des ventes on utilise la méthode Lasso puisque on a trouvé l'erreur minimale est = 1.00201336934613 avec lasso

QUESTION N°2

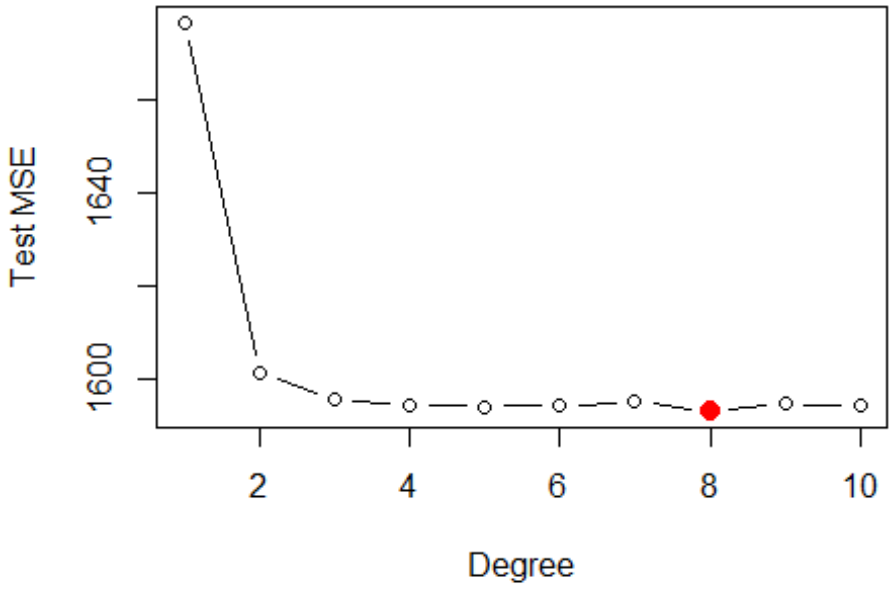
In this exercise, you will further analyze the "Wage" data set considered throughout this chapter.

a.

Perform polynomial regression to predict "wage" using "age". Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA ? Make a plot of the resulting polynomial fit to the data

Nous effectuerons une validation croisée K-fold avec K=10

```
set.seed(1923715)
library (ISLR)
library(boot)

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

deltas <- rep(NA, 10)
for (i in 1:10) {
    fit <- glm(wage ~ poly(age, i), data = Wage)
    deltas[i] <- cv.glm(Wage, fit, K = 10)$delta[1]
}
plot(1:10, deltas, xlab = "Degree", ylab = "Test MSE", type = "b")
d.min <- which.min(deltas)
points(which.min(deltas), deltas[which.min(deltas)], col = "red", cex = 2, pc
h = 20)
```



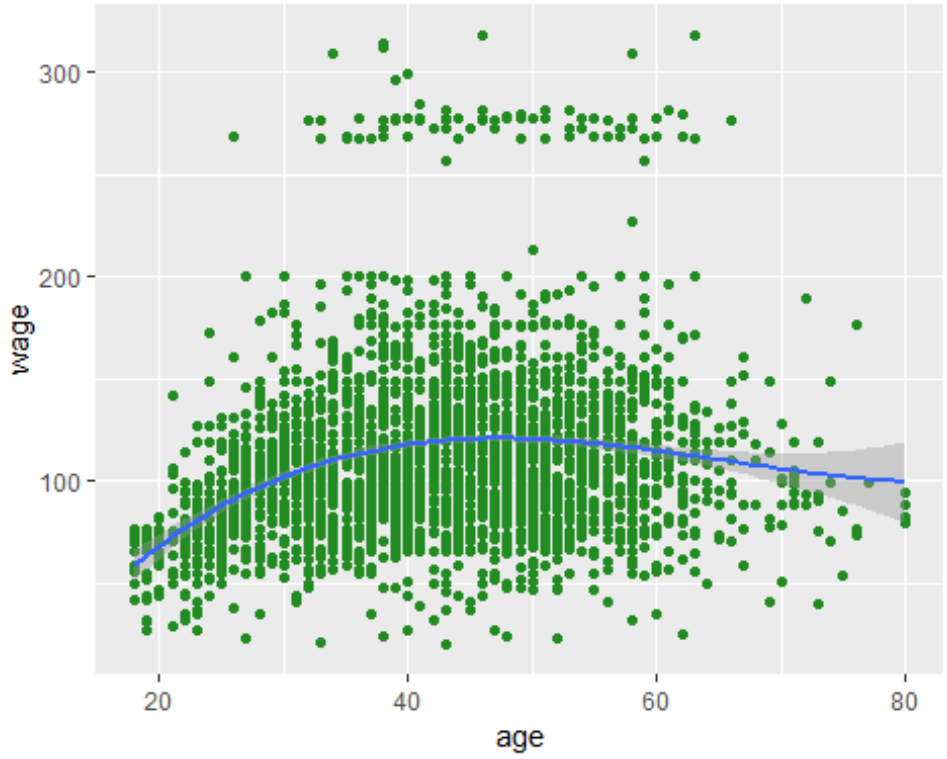
Nous pouvons voir que d = 8 est le degré optimal pour le polynôme. Nous utilisons maintenant ANOVA pour tester l'hypothèse nulle selon laquelle un modèle M1 est suffisant pour expliquer les données par rapport à l'hypothèse alternative selon laquelle un M2 plus complexe est requis.

```
fit1 <- lm(wage ~ age, data = Wage)
fit2 <- lm(wage ~ poly(age, 2), data = Wage)
fit3 <- lm(wage ~ poly(age, 3), data = Wage)
fit4 <- lm(wage ~ poly(age, 4), data = Wage)
fit5 <- lm(wage ~ poly(age, 5), data = Wage)
fit6 <- lm(wage ~ poly(age, 6), data = Wage)
fit7 <- lm(wage ~ poly(age, 7), data = Wage)
fit8 <- lm(wage ~ poly(age, 8), data = Wage)
fit9 <- lm(wage ~ poly(age, 9), data = Wage)
fit10 <- lm(wage ~ poly(age, 10), data = Wage)
anova(fit1, fit2, fit3, fit4, fit5, fit6, fit7, fit8, fit9, fit10)

## Analysis of Variance Table
##
## Model  1: wage ~ age
## Model  2: wage ~ poly(age, 2)
## Model  3: wage ~ poly(age, 3)
## Model  4: wage ~ poly(age, 4)
## Model  5: wage ~ poly(age, 5)
## Model  6: wage ~ poly(age, 6)
## Model  7: wage ~ poly(age, 7)
## Model  8: wage ~ poly(age, 8)
```

```
## Model  9: wage ~ poly(age, 9)
## Model 10: wage ~ poly(age, 10)
##    Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1    2998 5022216
## 2    2997 4793430  1    228786 143.7638 < 2.2e-16 ***
## 3    2996 4777674  1     15756   9.9005  0.001669 **
## 4    2995 4771604  1      6070   3.8143  0.050909 .
## 5    2994 4770322  1      1283   0.8059  0.369398
## 6    2993 4766389  1      3932   2.4709  0.116074
## 7    2992 4763834  1      2555   1.6057  0.205199
## 8    2991 4763707  1       127   0.0796  0.777865
## 9    2990 4756703  1      7004   4.4014  0.035994 *
## 10   2989 4756701  1         3   0.0017  0.967529
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pour une valeur de signifiance inferieure à 0.05 (le standard utilisé), ANOVA dit que seulement les polynômes de degrés 2,3 ou 9 (et presque 4) sont signifiants. on a choisi le polynôme de degré 3

```
ggplot(Wage, aes(age,wage))+
  geom_point(color="forestgreen")+
  stat_smooth(method = "lm", formula = y ~ poly(x, 3), size = 1)
```

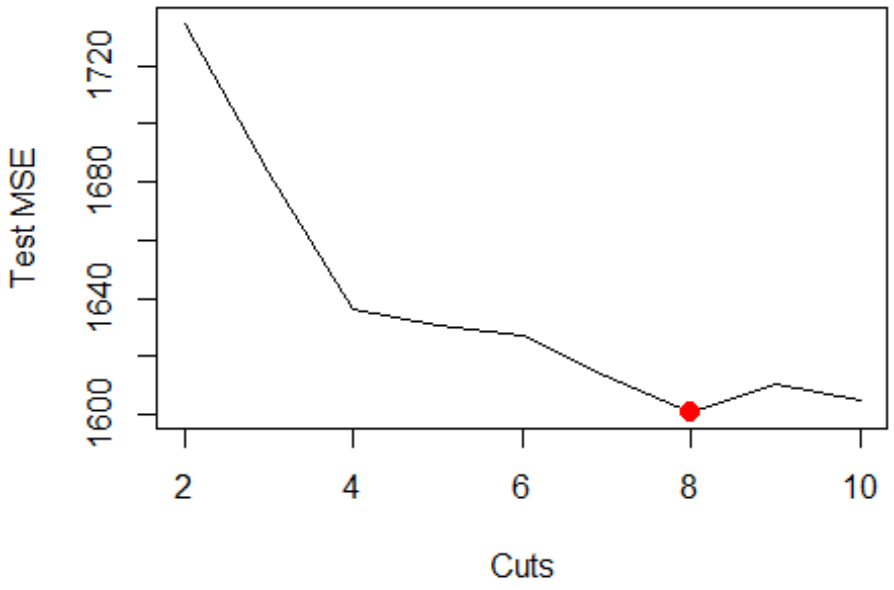


b.

Fit a step function to predict “wage” using “age”, and perform cross-validation to choose the optimal number of cuts. Make a plot of the fit obtained.

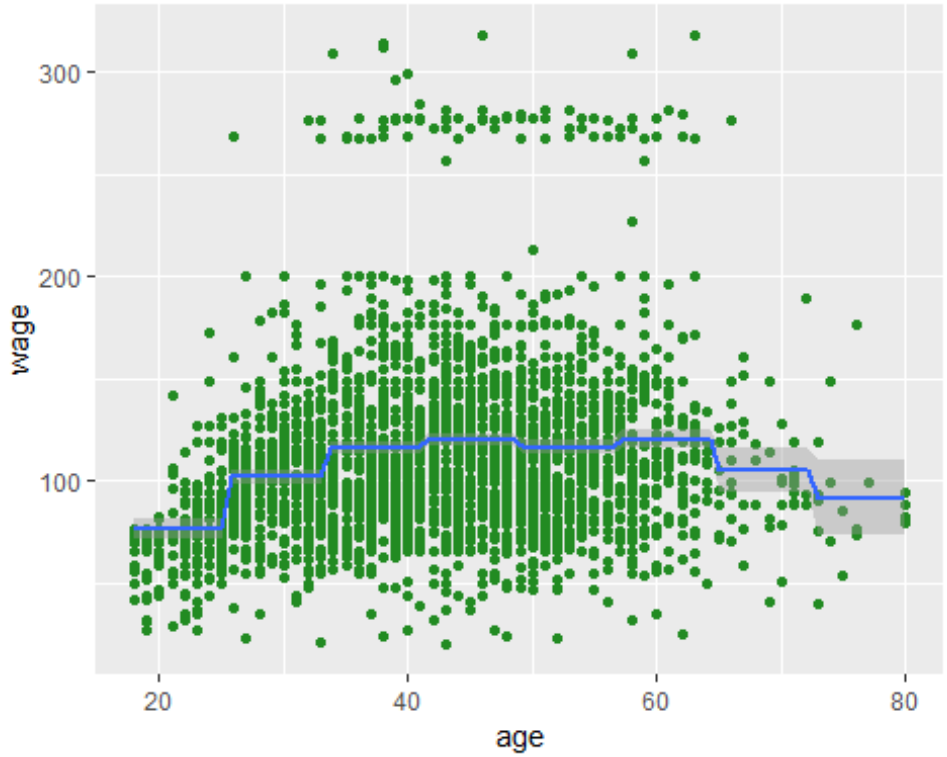
Nous effectuerons une validation croisée K-fold avec K=10

```
cvs <- rep(NA, 10)
for (i in 2:10) {
    Wage$age.cut <- cut(Wage$age, i)
    fit <- glm(wage ~ age.cut, data = Wage)
    cvs[i] <- cv.glm(Wage, fit, K = 10)$delta[1]
}
plot(2:10, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")
d.min <- which.min(cvs)
points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 2, pch = 20)
```



Nous pouvons voir que l’erreur est minimale pour 8 coupes. Maintenant, nous ajustons toutes les données avec une fonction pas à pas en utilisant 8 coupes et nous les représentons.

```
ggplot(Wage, aes(age,wage))+
  geom_point(color="forestgreen")+
  stat_smooth(method = "glm", formula = y ~ cut(x, 8), size = 1)
```


300
200
100
wage
20
40
60
80
age