Projet Machine Learning

Diamondra Rakotondraza

Année académique : 2020-2021

Introduction

Les données utilisées concernent les prix de biens immobilier au King County, aux Etats-Unis. Les variables sont essentiellement quantitatives, et portent sur les caractéristiques des biens et la variables Y à expliquer est leur prix.

Après une discussion avec le superviseur, on a donc décider d'employer les méthodes de régression par moindres carrés, regressions LASSO et Ridge dans le détail.

Nous comparerons les performances des modèles sur la base de la même métrique, le

Nettoyage et analyse descriptive des données

Suppression de variables et qualité des données

Nous supprimons dans un premier temps les variables jugées inutiles et inutilisables pour la construction de nos modèles de régression.

```
# Chargement des données
df <-
read.csv("C:/Users/Ascensio/Downloads/Machine_Learning/kc_house_data.csv")

# Suppression des variables id (identifiants de chaque bien immobilier), date
df = df[,-c(1,2)]
```

Il reste quelques variables de types qualititatives, mais les codages des catégories sont numériques. On peut donc les utiliser pour construires nos modèles de régression.

Vérifions maintenant si les données sont de qualité, selon le nombre de valeurs manquantes :

```
sum(is.na(df))

## [1] 0
```

Il n'y a aucune valeur manquante, donc les données sont bien utilisable pour analyse et construction de modèles de régression.

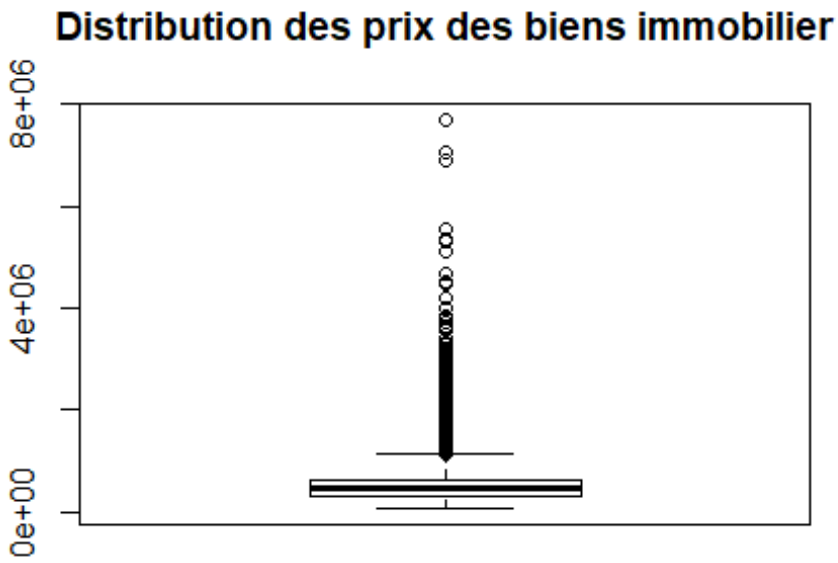
Analyse descriptive de la variable à expliquer

En rappelant que la variable à expliquer est le prix des biens immobilier, une analyse descriptive de cette dernière est utile afin d'avoir une idée des valeurs qu'elle peut prendre. Commençons par voir les caractéristiques générales de la distribution de cette variable :

```
summary(df$price)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   75000  321950  450000  540088  645000 7700000

# écart-type des prix
sd(df$price)

## [1] 367127.2
```

La moyenne du prix des biens immobiliers du King County est d'environ 540 000 dollars. En comparaison, l'écart-type valant près de 367 000 dollars, est très important. Représentons graphiquement la distribution des prix pour s'assurer du fait de cette importante disparité :

```
boxplot(df$price, main = "Distribution des prix des biens immobilier")
```



On remarque qu'un nombre non négligeable de biens sont excessivement plus chères que la plupart qui ont leur prix proche de la moyenne. Ce qui créer une grande disparité entre les prix en dollars des biens.

Régression par moindres carrés

A présent, appliquons la méthode par moindre carré, en vu de la création de notre premier modèle de régression. Cette méthode effectue la minimisation d'une fonction de risque empirique sous l'hypothèse d'un cout quadratique.

```
model1<-lm(price~.,data=df)
summary(model1)
```

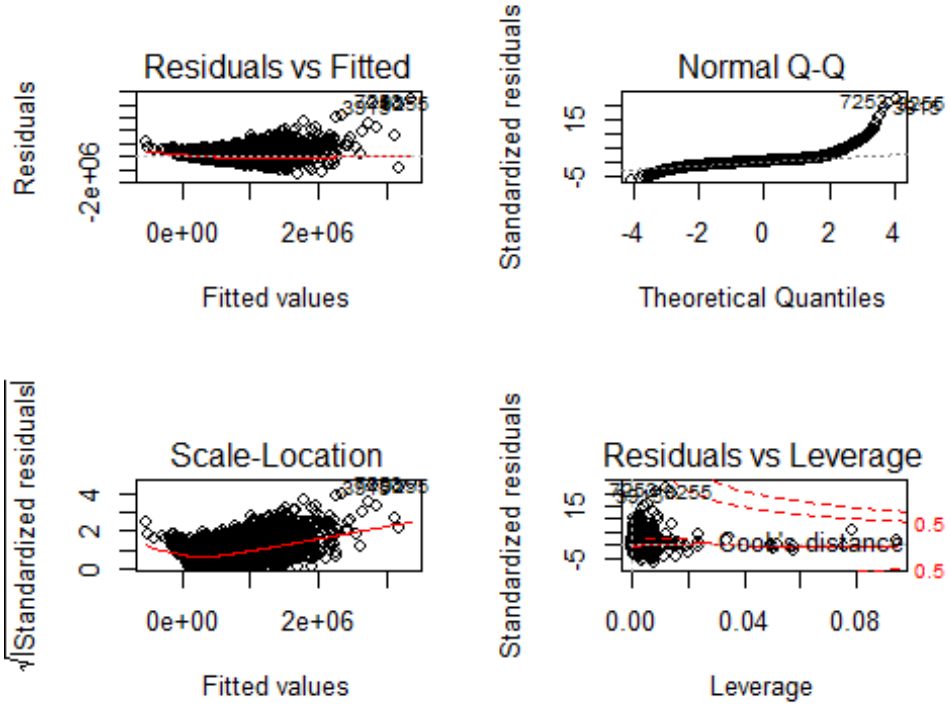
```
##
## Call:
## lm(formula = price ~ ., data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1291725   -99229    -9739    77583  4333222
##
## Coefficients: (1 not defined because of singularities)
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.690e+06  2.931e+06   2.282  0.02249 *
## bedrooms      -3.577e+04  1.892e+03 -18.906  < 2e-16 ***
## bathrooms      4.114e+04  3.254e+03  12.645  < 2e-16 ***
## sqft_living    1.501e+02  4.385e+00  34.227  < 2e-16 ***
## sqft_lot       1.286e-01  4.792e-02   2.683  0.00729 **
## floors         6.690e+03  3.596e+03   1.860  0.06285 .
## waterfront     5.830e+05  1.736e+04  33.580  < 2e-16 ***
## view           5.287e+04  2.140e+03  24.705  < 2e-16 ***
## condition      2.639e+04  2.351e+03  11.221  < 2e-16 ***
## grade          9.589e+04  2.153e+03  44.542  < 2e-16 ***
## sqft_above     3.113e+01  4.360e+00   7.139 9.71e-13 ***
## sqft_basement         NA         NA      NA       NA
## yr_built      -2.620e+03  7.266e+01 -36.062  < 2e-16 ***
## yr_renovated   1.981e+01  3.656e+00   5.420 6.03e-08 ***
## zipcode       -5.824e+02  3.299e+01 -17.657  < 2e-16 ***
## lat            6.027e+05  1.073e+04  56.149  < 2e-16 ***
## long          -2.147e+05  1.313e+04 -16.349  < 2e-16 ***
## sqft_living15  2.168e+01  3.448e+00   6.289 3.26e-10 ***
## sqft_lot15    -3.826e-01  7.327e-02  -5.222 1.78e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 201200 on 21595 degrees of freedom
## Multiple R-squared:  0.6997, Adjusted R-squared:  0.6995
## F-statistic:  2960 on 17 and 21595 DF,  p-value: < 2.2e-16
```

Avec toutes les variables, le modèle obtenu semble plutôt correct au vu de la valeur du coefficient de détermination. Il est d'une valeur proche de 0.7, donc l'équation de la droite de régression détermine 70 % de l'ensemble des données.

Nous n'obtenons pas d'estimation de coefficient pour la variable sqft_basement.

On accepte l'hypothèse globale d'une relation linaire car la p-value de la F-statistic est trés faible. On la compare à un seuil alpha valant 5%. Hormis floors, on rejète l'hypothèse de nullité du coefficient des autres variables. variables car les p-values des variables sont significatives. Si la valeur associée à un coefficient de régression vaut 0, la variable associée n'influe pas la variable à expliquer.

```
par(mfrow=c(2,2))
plot(model1)
```



Néanmoins d'après les graphiques précédents, il semble y avoir le phénomène d'hétéroscédasticité. La variance de la variable que l'on veut prédire n'est pas constante sur le domaine de la variable aléatoire que l'on utilise. Et une petites partie des observations, ne semblent pas venir d'une distribution normale.

Il va donc maintenant s'agir de sélectionner les variables qui constitueront notre modèle.

Sélection de variables

La sélection de variables s'est faites sur la base de la recherche exhaustive. Etant donné qu'on a une variable dont le modèle ne donne pas sa p-value (NA), les recherches pas-à-pas sont jugées moins pertinentes.

Recherche exhaustive

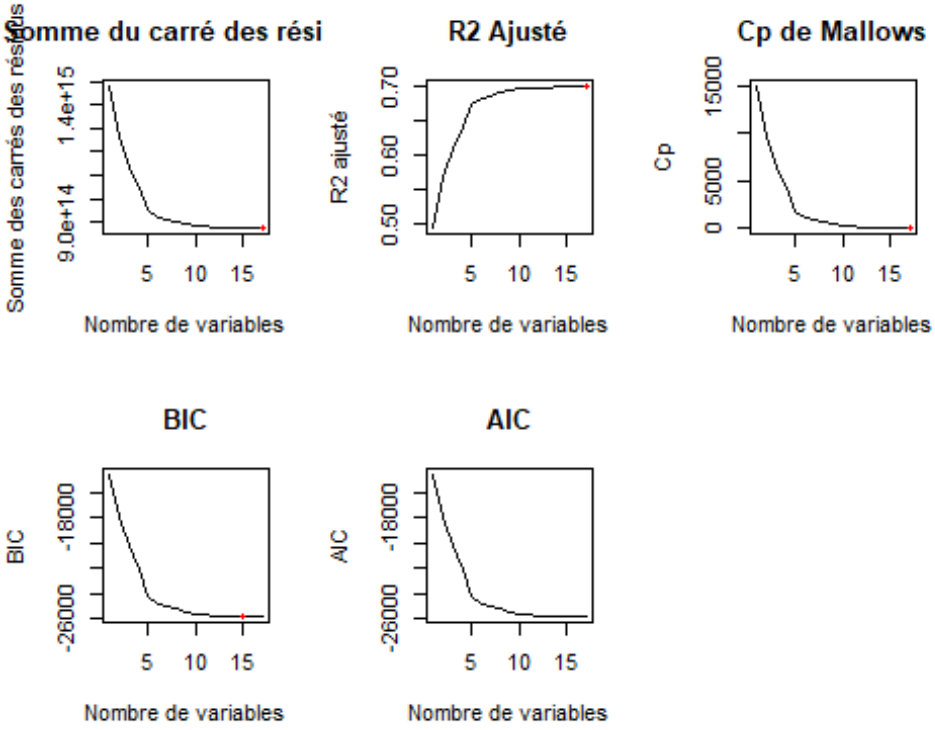
On considère ici toutes les combinaisons de sous-modèles possibles. Il y a 18 variables explicatives, donc 262 143 possibilités. Le meilleur modèle sera sélectionner selon le critère donné. On en comparera plusieurs, afin de voir si les résultats trouvés sont globalement

cohérents. A savoir la somme du carrés des résidus la plus faible, le plus grand R^2 ajusté, les critères BIC, AIC et le C_p de Mallow.

```
library(leaps)

## Warning: package 'leaps' was built under R version 3.6.2

comb_model1<-regsubsets(price~.,data= df, nvmax=18)

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

selection<-summary(comb_model1)
```

On représente graphiquement les résultats selon les critères choisis. Les optimums sont indiqués par un point rouge :

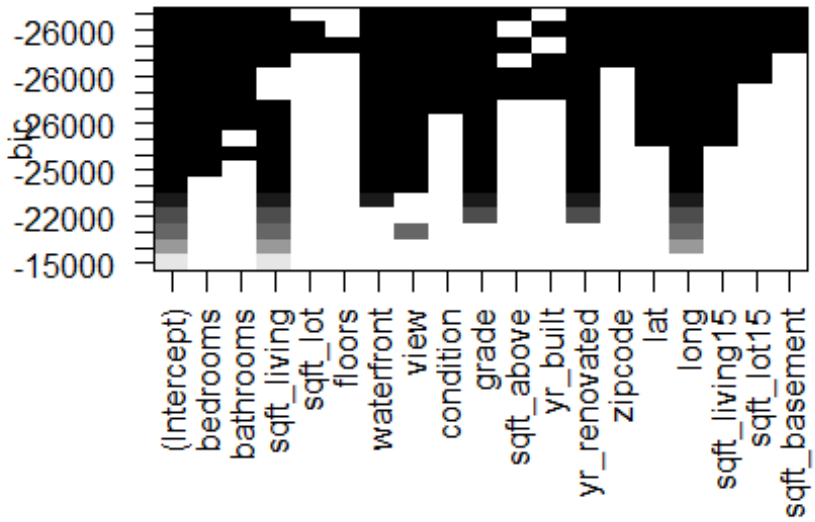
```
par(mfrow=c(2,3))
plot(selection$rss, xlab="Nombre de variables",ylab="Somme des carrés des
résidus",type="l", main ="Somme du carré des résidus")
u<-which.min(selection$rss)
points(u,selection$rss[u],pch=20,col="red")

plot(selection$adjr2, xlab="Nombre de variables",ylab="R2
ajust\U00E9",type="l", main="R2 Ajust\U00E9")
u<-which.max(selection$adjr2)
points(u,selection$adjr2[u],pch=20,col="red")

plot(selection$cp, xlab="Nombre de variables",ylab="Cp",type="l", main="Cp de
Mallows")
u<-which.min(selection$cp)
points(u,selection$cp[u],pch=20,col="red")

plot(selection$bic, xlab="Nombre de variables",ylab="BIC",type="l",
main="BIC")
u<-which.min(selection$bic)
points(u,selection$bic[u],pch=20,col="red")

plot(selection$bic, xlab="Nombre de variables",ylab="AIC",type="l",
main="AIC")
u<-which.min(selection$aic)
points(u,selection$bic[u],pch=20,col="red")
```



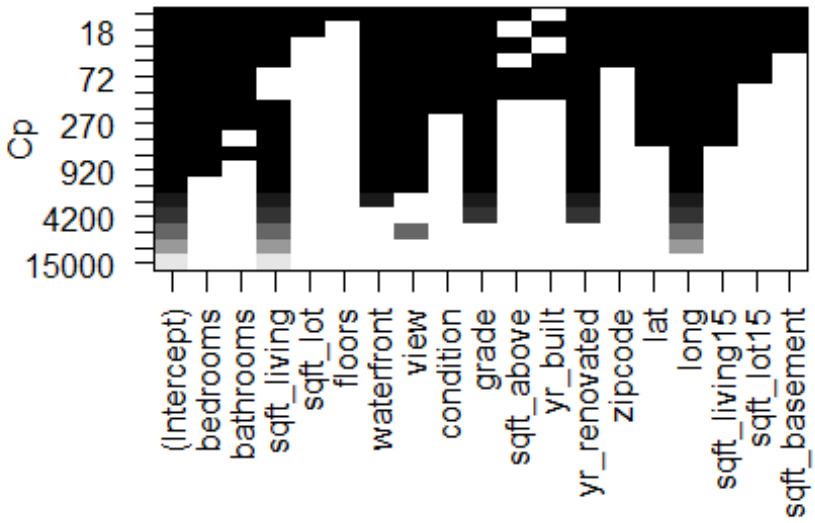
Le critère BIC indique que le modèle idéal comporte 15 variables et les autres 17.

Par conséquons, voyons quelles variables sont sélectionnées pour les modèles pour chaque critère.

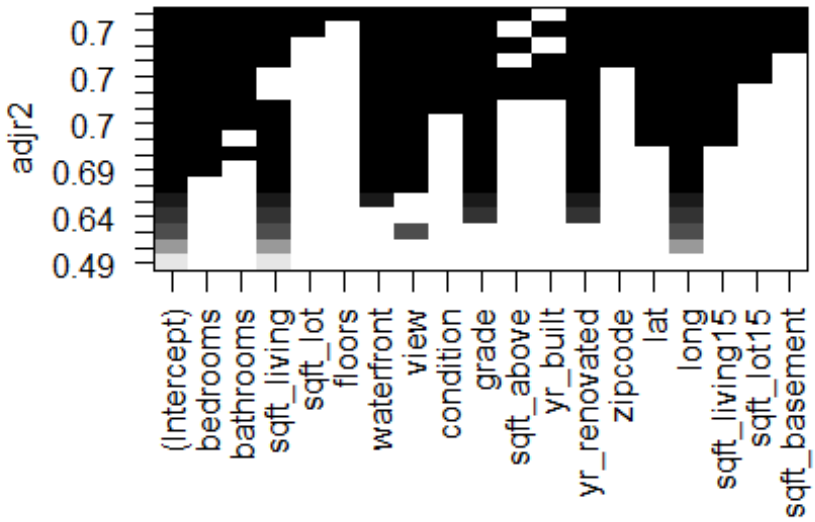
```
plot(comb_model1, scale="bic")
```




```
plot(comb_model1, scale="Cp")
```




```
plot(comb_model1, scale="adjr2")
```



La variable n'est pas incluse dans les modèles à 17 variables sous la base des critères BIC, du plus grand R^2 ajusté et C_p de Mallow.

Comparons les valeurs des coefficients de détermination des modèles idéals pour le critère BIC (15 variables) et les autres (17 variables) d'après la recherche exhaustive.

```
# modèle à 17 variables
model1_17<-
lm(price~bedrooms+bathrooms+sqft_living+sqft_lot+floors+waterfront+view+condi
tion+grade+sqft_above+sqft_basement+yr_renovated+zipcode+lat+long+sqft_living
15+sqft_lot15,data=df)
summary(model1_17)

##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_lot +
##     floors + waterfront + view + condition + grade + sqft_above +
##     sqft_basement + yr_renovated + zipcode + lat + long + sqft_living15 +
##     sqft_lot15, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1228795  -105112   -11333    79199  4494303
##
## Coefficients: (1 not defined because of singularities)
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -3.056e+07  2.825e+06 -10.818  < 2e-16 ***
```

```
## bedrooms      -3.119e+04  1.944e+03 -16.047  < 2e-16 ***
## bathrooms      2.671e+03  3.165e+03   0.844    0.399
## sqft_living    1.676e+02  4.488e+00  37.359  < 2e-16 ***
## sqft_lot       2.041e-01  4.930e-02   4.140 3.48e-05 ***
## floors        -2.154e+04  3.614e+03  -5.962 2.53e-09 ***
## waterfront     5.753e+05  1.787e+04  32.185  < 2e-16 ***
## view           5.914e+04  2.196e+03  26.926  < 2e-16 ***
## condition      5.371e+04  2.292e+03  23.433  < 2e-16 ***
## grade          7.868e+04  2.161e+03  36.401  < 2e-16 ***
## sqft_above     3.346e+01  4.489e+00   7.454 9.40e-14 ***
## sqft_basement         NA         NA      NA       NA
## yr_renovated   6.190e+01  3.567e+00  17.354  < 2e-16 ***
## zipcode       -4.304e+02  3.369e+01 -12.777  < 2e-16 ***
## lat            6.737e+05  1.087e+04  62.000  < 2e-16 ***
## long          -3.284e+05  1.313e+04 -25.017  < 2e-16 ***
## sqft_living15  2.527e+01  3.548e+00   7.122 1.10e-12 ***
## sqft_lot15    -4.218e-01  7.543e-02  -5.591 2.28e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 207200 on 21596 degrees of freedom
## Multiple R-squared:  0.6817, Adjusted R-squared:  0.6814
## F-statistic:  2890 on 16 and 21596 DF,  p-value: < 2.2e-16
```

```
# modèle à 15 variables
model1_15<-
lm(price~bedrooms+bathrooms+sqft_living+waterfront+view+condition+grade+sqft_
above+sqft_basement+yr_renovated+zipcode+lat+long+sqft_living15+sqft_lot15,da
ta=df)
summary(model1_15)
```

```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + waterfront +
##     view + condition + grade + sqft_above + sqft_basement + yr_renovated +
##     zipcode + lat + long + sqft_living15 + sqft_lot15, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1211742  -105712   -11701    78340  4503158
##
## Coefficients: (1 not defined because of singularities)
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -2.835e+07  2.811e+06 -10.087  < 2e-16 ***
## bedrooms      -3.088e+04  1.944e+03 -15.886  < 2e-16 ***
## bathrooms     -4.346e+03  2.957e+03  -1.470 0.141551
## sqft_living    1.775e+02  4.222e+00  42.040  < 2e-16 ***
## waterfront     5.732e+05  1.789e+04  32.037  < 2e-16 ***
## view           5.941e+04  2.197e+03  27.037  < 2e-16 ***
## condition      5.558e+04  2.273e+03  24.450  < 2e-16 ***
```

```
## grade          7.689e+04  2.142e+03  35.901  < 2e-16 ***
## sqft_above     2.175e+01  4.013e+00   5.421 5.98e-08 ***
## sqft_basement         NA         NA      NA       NA
## yr_renovated   6.229e+01  3.571e+00  17.446  < 2e-16 ***
## zipcode       -4.433e+02  3.365e+01 -13.175  < 2e-16 ***
## lat            6.695e+05  1.086e+04  61.624  < 2e-16 ***
## long          -3.222e+05  1.311e+04 -24.575  < 2e-16 ***
## sqft_living15  2.689e+01  3.520e+00   7.639 2.28e-14 ***
## sqft_lot15    -1.814e-01  5.460e-02  -3.323 0.000893 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 207500 on 21598 degrees of freedom
## Multiple R-squared:  0.6809, Adjusted R-squared:  0.6807
## F-statistic:  3291 on 14 and 21598 DF,  p-value: < 2.2e-16
```

On obtient un meilleur modèle avec 17 variables, avec une valeur supérieure de R^2. Toutefois le meilleur modèle est celui qui comporte toute les 18 variables.

Sélection par erreur de prédiction

```
set.seed(1)
appr<-sample(c(TRUE,FALSE), nrow(df), rep=TRUE)
test<-(!appr)

regappr<-regsubsets(price~.,data=df[appr,], nvmax=18)

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

X.test<-model.matrix(price~.,data=df[test,])

regappr<-regsubsets(price~.,data=df[appr,], nvmax=18)

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

X.test<-model.matrix(price~.,data=df[test,])
error<-rep(NA,18)
for (i in 1:17)
{
  coefi<-coef(regappr,id=i)
  pred<-X.test[,names(coefi)]%*%coefi
  error[i]<-mean((df$price[test]-pred)^2)
}
```

```r
which.min(error)
```

```
## [1] 16
```

```r
coef(regappr,16)
```

```
##   (Intercept)      bedrooms     bathrooms   sqft_living      sqft_lot
##  5.606162e+06 -4.055842e+04  4.334735e+04  1.803823e+02  1.570962e-01
##    waterfront          view     condition         grade      yr_built
##  6.308990e+05  4.970195e+04  2.632073e+04  9.570915e+04 -2.611463e+03
##  yr_renovated       zipcode           lat          long sqft_living15
##  1.370064e+01 -5.696490e+02  6.055942e+05 -2.122126e+05  2.780882e+01
##    sqft_lot15 sqft_basement
## -4.452664e-01 -2.989508e+01
```

```r
summary(lm(price~bedrooms+bathrooms+sqft_living+sqft_lot+waterfront+view+cond
ition+grade+yr_built+sqft_basement+yr_renovated+zipcode+lat+long+sqft_living1
5+sqft_lot15,data=df ))
```

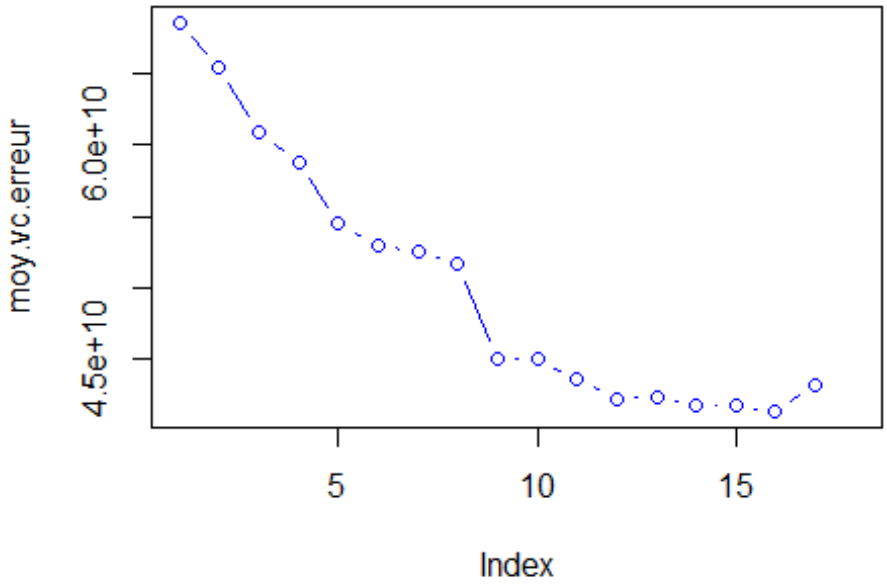
```
##
## Call:
## lm(formula = price ~ bedrooms + bathrooms + sqft_living + sqft_lot +
##     waterfront + view + condition + grade + yr_built + sqft_basement +
##     yr_renovated + zipcode + lat + long + sqft_living15 + sqft_lot15,
##     data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1292272   -99268    -9849    77605  4331513
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.741e+06  2.887e+06   1.989  0.04674 *
## bedrooms      -3.586e+04  1.891e+03 -18.962  < 2e-16 ***
## bathrooms      4.272e+04  3.142e+03  13.596  < 2e-16 ***
## sqft_living    1.823e+02  3.621e+00  50.353  < 2e-16 ***
## sqft_lot       1.266e-01  4.791e-02   2.643  0.00822 **
## waterfront     5.831e+05  1.736e+04  33.585  < 2e-16 ***
## view           5.297e+04  2.140e+03  24.756  < 2e-16 ***
## condition      2.614e+04  2.348e+03  11.133  < 2e-16 ***
## grade          9.624e+04  2.145e+03  44.878  < 2e-16 ***
## yr_built      -2.591e+03  7.092e+01 -36.531  < 2e-16 ***
## sqft_basement -3.472e+01  3.910e+00  -8.878  < 2e-16 ***
## yr_renovated   2.017e+01  3.651e+00   5.526 3.32e-08 ***
## zipcode       -5.767e+02  3.284e+01 -17.559  < 2e-16 ***
## lat            6.044e+05  1.070e+04  56.494  < 2e-16 ***
## long          -2.168e+05  1.309e+04 -16.568  < 2e-16 ***
## sqft_living15  2.097e+01  3.426e+00   6.119 9.57e-10 ***
## sqft_lot15    -3.874e-01  7.323e-02  -5.291 1.23e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 201300 on 21596 degrees of freedom
## Multiple R-squared:  0.6997, Adjusted R-squared:  0.6995
## F-statistic:  3145 on 16 and 21596 DF,  p-value: < 2.2e-16
```

Validation croisée

```
k<-10
set.seed(1)
plis<-sample(1:k, nrow(df),replace=TRUE)
vc.erreur<-matrix(NA,k,18, dimnames=list(NULL,paste(1:18)))

table(plis)

## plis
##    1    2    3    4    5    6    7    8    9   10
## 2106 2072 2147 2215 2122 2146 2326 2134 2208 2137

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
}

for(j in 1:k)
{
cv.reg<-regsubsets(price~.,data=df[plis!=j,], nvmax=18)
  for(i in 1:17)
    {
    pred<-predict(cv.reg, df[plis==j,],id=i)
    vc.erreur[j,i]<-mean((df$price[plis==j]-pred)^2)
    }
}

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

moy.vc.erreur<-apply(vc.erreur,2,mean)
plot(moy.vc.erreur, type='b', col="blue")
```



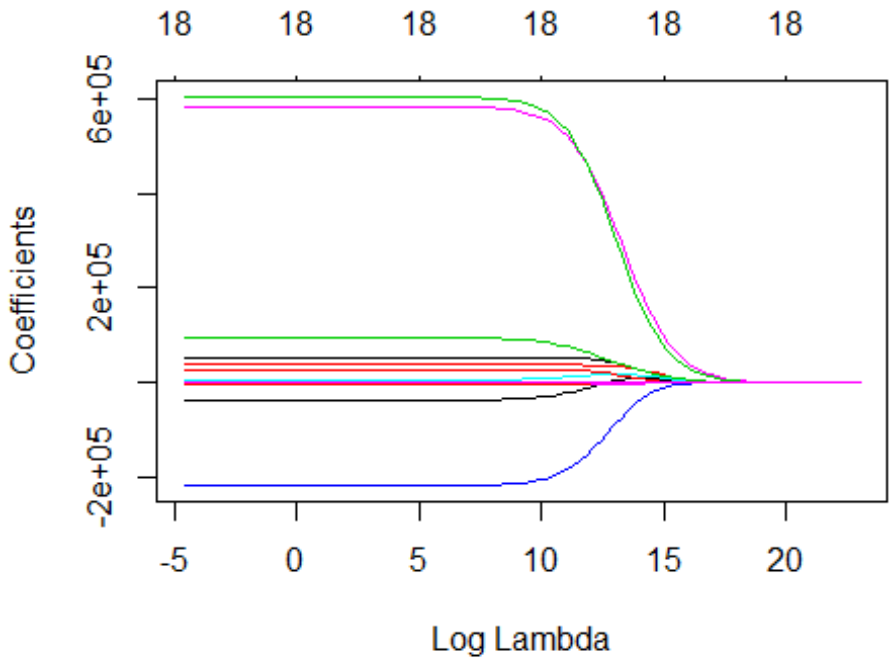

```
which.min(moy.vc.erreur)

## 16
## 16

reg.best<-regsubsets(price~.,data=df, nvmax=18)

## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
force.in =
## force.in, : 1 linear dependencies found

## Reordering variables and trying again:

names(coef(reg.best,16))

##  [1] "(Intercept)"   "bedrooms"      "bathrooms"     "sqft_living"
##  [5] "sqft_lot"      "waterfront"    "view"          "condition"
##  [9] "grade"         "yr_built"      "yr_renovated"  "zipcode"
## [13] "lat"           "long"          "sqft_living15" "sqft_lot15"
## [17] "sqft_basement"
```

Régression Ridge

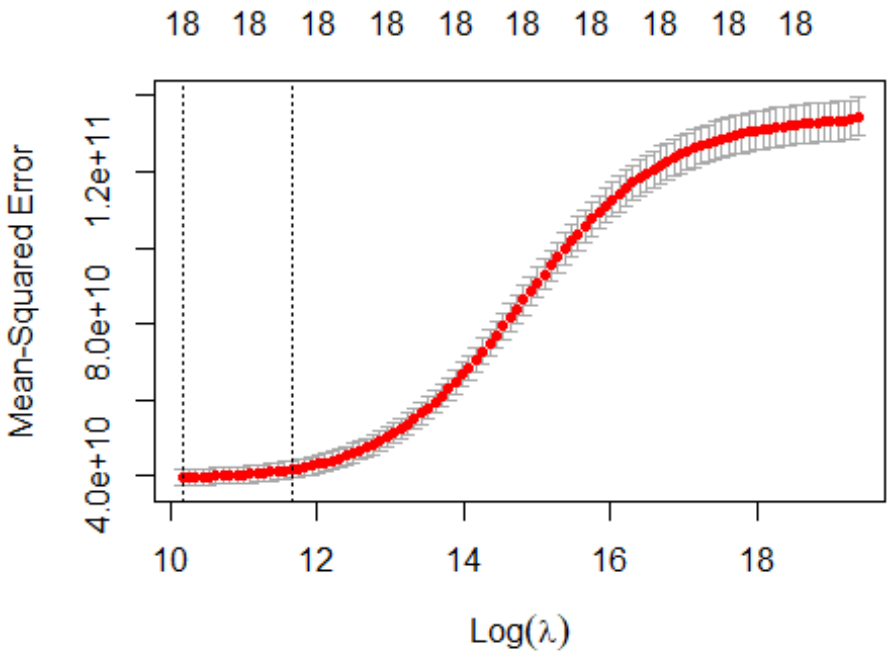
Contrairement à la régression par moindre carré, la regression Ridge garde toute les variables explicatives. Il n'y a donc pas d'étape de sélection de variable. La fonction de risque diffère, au niveau du fait qu'une pénalité est ajouté sur les coefficients de la régression. Selon la valeur d'un paramètre de réglage lambda (positif), on force les

coefficients à tendre vers 0 quand lambda croît. Donc le but est de trouver une valeur de lamdba

```
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.6.3

library(glmnet)

## Warning: package 'glmnet' was built under R version 3.6.3

## Loading required package: Matrix

## Loaded glmnet 4.1-1

x<-model.matrix(price~.,df) #matrice des variables explicatives
y<-df$price #vecteur de la variable à expliquer




# 1. Régression ridge

# La fonction glmnet effectue la régression de façon automatique
# sur un ensemble de 100 valeurs de lambda.
regridge<-glmnet(x,y,alpha=0)

grid<-10^seq(10,-2,length=100)

regridge<-glmnet(x,y,alpha=0,lambda=grid)# Effectue la régression ridge
# pour les 100 valeurs de lambda définies par grid.

plot(regridge,xvar="lambda")
```




```r
set.seed(1)
set.seed(1)
appr<-sample(1:nrow(x), nrow(x)/2) # on choisit 131.5 soit 132 indices donc joueurs aléatoirement.
test<-(-appr)
y.test<-y[test]
vc.out<-cv.glmnet(x[appr,],y[appr],alpha=0)
plot(vc.out)
```

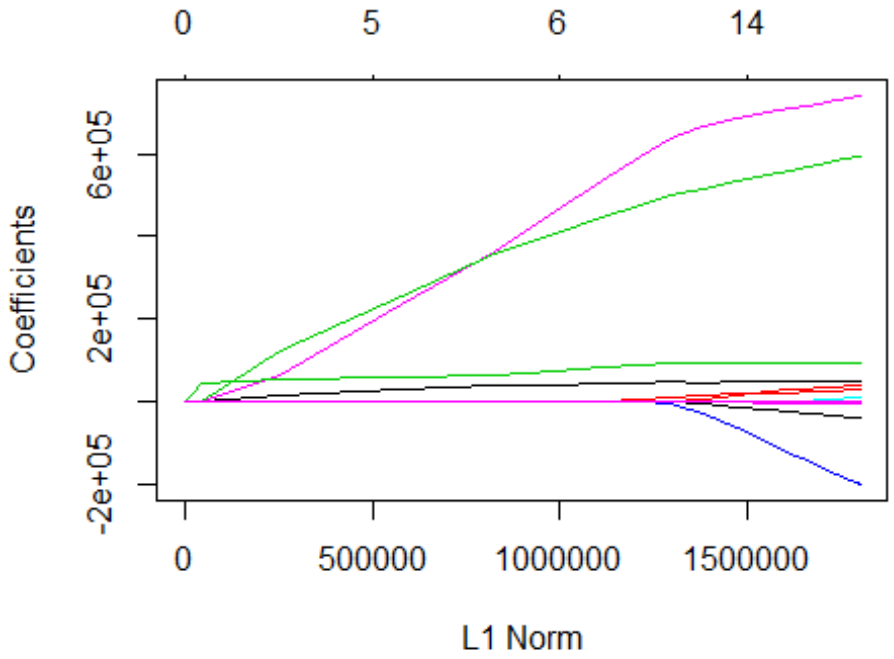



```
lamopti<-vc.out$lambda.min
lamopti

## [1] 25958.12

ridge<-glmnet(x[appr,],y[appr],alpha=0,lambda=lamopti, thresh=1e-12)

pred.ridge<-predict(ridge,s=lamopti,newx=x[test,])
mean((pred.ridge-y.test)^2)

## [1] 42394486414

out<-glmnet(x,y,alpha=0)
predict(out,type="coefficients",s=lamopti)

## 20 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)   -3.643485e+06
## (Intercept)    .
## bedrooms      -2.815513e+04
## bathrooms      3.979621e+04
## sqft_living    8.461738e+01
## sqft_lot       1.191650e-01
## floors         1.024622e+04
## waterfront     5.544089e+05
## view           5.299709e+04
## condition      2.811510e+04
## grade          8.563644e+04
```

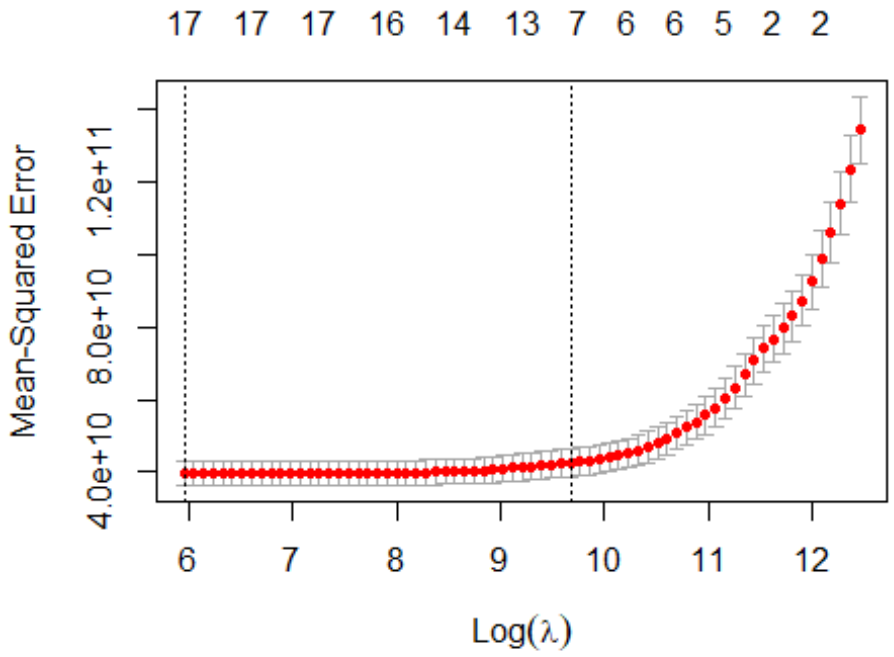
```
## sqft_above     8.363431e+01
## sqft_basement  6.013991e+01
## yr_built      -2.240462e+03
## yr_renovated   2.576191e+01
## zipcode       -4.519165e+02
## lat            5.710456e+05
## long          -2.010805e+05
## sqft_living15  3.674034e+01
## sqft_lot15    -3.135893e-01
```

Régression LASSO

```
reglasso<-glmnet(x[appr,],y[appr],alpha=1,lambda=grid)
plot(reglasso)

## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
unique
## 'x' values
```




```
set.seed(1)
vc.out=cv.glmnet(x[appr,],y[appr],alpha=1)
plot(vc.out)
```




```
lamopti=vc.out$lambda.min
lamopti
```

```
## [1] 385.4699
```

```
pred.lasso<-predict(reglasso,s=lamopti,newx=x[test,])
mean((pred.lasso-y.test)^2)
```

```
## [1] 42263200637
```

```
out<-glmnet(x,y,alpha=1,lambda=grid)
coef.lasso<-predict(out,type="coefficients",s=lamopti)[1:20,]
coef.lasso
```

```
##   (Intercept)   (Intercept)      bedrooms     bathrooms   sqft_living
##  5.429633e+06  0.000000e+00 -3.458319e+04  4.023805e+04  1.498564e+02
##      sqft_lot        floors    waterfront          view     condition
##  9.344604e-02  5.700823e+03  5.802226e+05  5.282436e+04  2.588833e+04
##         grade    sqft_above sqft_basement      yr_built  yr_renovated
##  9.611306e+04  3.080591e+01  0.000000e+00 -2.592093e+03  1.939748e+01
##       zipcode           lat          long sqft_living15    sqft_lot15
## -5.614706e+02  5.999189e+05 -2.088927e+05  2.113909e+01 -3.292644e-01
```

```
coef.lasso[coef.lasso!=0]
```

```
##   (Intercept)      bedrooms     bathrooms   sqft_living      sqft_lot
##  5.429633e+06 -3.458319e+04  4.023805e+04  1.498564e+02  9.344604e-02
##        floors    waterfront          view     condition         grade
```

```
##  5.700823e+03  5.802226e+05  5.282436e+04  2.588833e+04  9.611306e+04
##    sqft_above      yr_built  yr_renovated       zipcode           lat
##  3.080591e+01 -2.592093e+03  1.939748e+01 -5.614706e+02  5.999189e+05
##          long sqft_living15    sqft_lot15
## -2.088927e+05  2.113909e+01 -3.292644e-01
```