Prédiction des prix des maisons

Techniques de régression basiques en R

EDY DIEHL TD

Contents

1. Introduction

Le but de ce projet est de manipuler un échantillon de données afin d'entrainer un modèle qui prédit le prix de vente des maisons à Ames, dans l'état d'Iowa aux USA. Nous pouvons sélectionner parmi 75 variables, celles qui ont un pouvoir d'influence important sur le prix de vente. Le jeu de données contient les prix de vente observés pour 1095 maisons construites entre 1872 et 2010.

La variable à prédire est "SalePrice" qui correspond au prix de vente d'une maison.

Nous importons les fichiers contenant les jeux de données d'entrainement et de test ("Train.csv" et "Test.csv") et fixons la graine aléatoire afin de pouvoir comparer nos résultats.

```
set.seed(2010)
train <- read.csv("Train.csv", header = TRUE)
test <- read.csv("Test.csv", header = TRUE)
dim(train)
```

```
## [1] 1095   75
```

```
dim(test)
```

```
## [1] 365  75
```

```
str(train)
```

```
## 'data.frame':    1095 obs. of  75 variables:
##  $ MSSubClass   : int  60 190 20 90 20 50 90 20 60 20 ...
##  $ MSZoning     : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : int  63 62 90 92 65 86 75 71 74 63 ...
##  $ LotArea      : int  7875 10106 17217 12108 8450 11500 9825 9230 7472 8487 ...
##  $ Street       : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ LotShape     : chr  "Reg" "Reg" "Reg" "Reg" ...
##  $ LandContour  : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr  "Inside" "Inside" "Inside" "Inside" ...
##  $ LandSlope    : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr  "Gilbert" "Edwards" "Mitchel" "Edwards" ...
##  $ Condition1   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ Condition2   : chr  "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr  "1Fam" "2fmCon" "1Fam" "Duplex" ...
##  $ HouseStyle   : chr  "2Story" "1.5Fin" "1Story" "1Story" ...
##  $ OverallQual  : int  7 5 5 4 7 7 5 5 7 7 ...
##  $ OverallCond  : int  5 7 5 4 5 7 5 8 9 5 ...
##  $ YearBuilt    : int  2003 1940 2006 1955 2000 1936 1965 1965 1972 2004 ...
##  $ YearRemodAdd : int  2003 1999 2006 1955 2001 1987 1965 1998 2004 2004 ...
##  $ RoofStyle    : chr  "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr  "VinylSd" "Wd Sdng" "VinylSd" "VinylSd" ...
##  $ Exterior2nd  : chr  "VinylSd" "Wd Sdng" "VinylSd" "VinylSd" ...
##  $ MasVnrType   : chr  "None" "None" "None" "BrkFace" ...
##  $ MasVnrArea   : int  0 0 0 270 108 0 0 166 138 210 ...
##  $ ExterQual    : chr  "Gd" "TA" "TA" "TA" ...
##  $ ExterCond    : chr  "TA" "Gd" "TA" "TA" ...
##  $ Foundation   : chr  "PConc" "BrkTil" "PConc" "CBlock" ...
```

```
##  $ BsmtQual     : chr  "Gd" "TA" "Gd" "TA" ...
##  $ BsmtCond     : chr  "TA" "TA" "TA" "TA" ...
##  $ BsmtExposure : chr  "No" "No" "No" "No" ...
##  $ BsmtFinType1 : chr  "Unf" "ALQ" "Unf" "ALQ" ...
##  $ BsmtFinSF1   : int  0 351 0 133 0 223 0 661 626 20 ...
##  $ BsmtFinType2 : chr  "Unf" "Rec" "Unf" "Unf" ...
##  $ BsmtFinSF2   : int  0 181 0 0 0 0 0 0 0 0 ...
##  $ BsmtUnfSF    : int  783 112 1140 1307 1349 794 0 203 99 1480 ...
##  $ TotalBsmtSF  : int  783 644 1140 1440 1349 1017 0 864 725 1500 ...
##  $ Heating      : chr  "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr  "Ex" "Gd" "Ex" "TA" ...
##  $ CentralAir   : chr  "Y" "Y" "Y" "N" ...
##  $ Electrical   : chr  "SBrkr" "SBrkr" "SBrkr" "FuseF" ...
##  $ X1stFlrSF    : int  807 808 1140 1440 1349 1020 1664 1200 725 1500 ...
##  $ X2ndFlrSF    : int  702 547 0 0 0 1037 0 0 754 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1509 1355 1140 1440 1349 2057 1664 1200 1479 1500 ...
##  $ BsmtFullBath : int  0 1 0 0 0 0 0 1 1 0 ...
##  $ BsmtHalfBath : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 1 2 2 1 2 1 1 2 ...
##  $ HalfBath     : int  1 0 0 0 0 1 0 1 1 0 ...
##  $ BedroomAbvGr : int  3 4 3 4 3 3 4 1 4 3 ...
##  $ KitchenAbvGr : int  1 2 1 2 1 1 2 1 1 1 ...
##  $ KitchenQual  : chr  "Gd" "TA" "TA" "Fa" ...
##  $ TotRmsAbvGrd : int  8 6 6 8 6 6 8 6 7 6 ...
##  $ Functional   : chr  "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : int  1 0 0 0 0 1 0 0 0 0 ...
##  $ GarageType   : chr  "Attchd" "Attchd" "Attchd" "Attchd" ...
##  $ GarageYrBlt  : int  2003 1954 1999 1961 2000 1936 1967 1977 1972 2004 ...
##  $ GarageFinish : chr  "Fin" "Unf" "Unf" "Unf" ...
##  $ GarageCars   : int  2 0 0 0 2 1 0 2 2 2 ...
##  $ GarageArea   : int  393 0 0 0 539 180 0 884 484 570 ...
##  $ GarageQual   : chr  "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr  "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr  "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : int  0 140 36 0 120 0 0 0 0 192 ...
##  $ OpenPorchSF  : int  75 0 56 0 55 0 0 64 32 36 ...
##  $ EnclosedPorch: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 322 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MiscVal      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MoSold       : int  7 9 7 9 12 6 5 10 6 8 ...
##  $ YrSold       : int  2006 2008 2006 2008 2007 2006 2010 2006 2007 2009 ...
##  $ SaleType     : chr  "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr  "Normal" "Normal" "Abnorml" "Normal" ...
##  $ SalePrice    : int  180000 127500 84500 118000 179000 250000 100000 146000 184000 190000 ...
```

Nous nous assurons que les jeux de données d'entrainement et de test contiennent les mêmes variables.

```
train_columns <- names(train)
test_columns <- names(test)

dif_train_test <- train_columns[which(!(train_columns %in% test_columns))]
dif_test_train <- test_columns[which(!(test_columns %in% train_columns))]
```

```
dif_train_test
```

```
## character(0)
```

```
dif_test_train
```

```
## character(0)
```

```
rm(dif_train_test, dif_test_train)
```

2. Analyse exploratoire des données et modélisation

2.1. Etude de la distribution de la variable cible

Affichons les statistiques sommaires de la variable SalePrice.

Variance et écart type :

```
print(var(train$SalePrice))
```

```
## [1] 6071338311
```

```
print(sd(train$SalePrice))
```

```
## [1] 77918.79
```

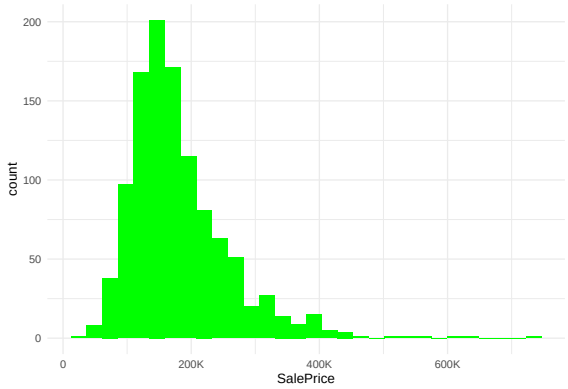
Min,max, médiane, quantiles, moyenne:

```
pander(summary(train$SalePrice))
```

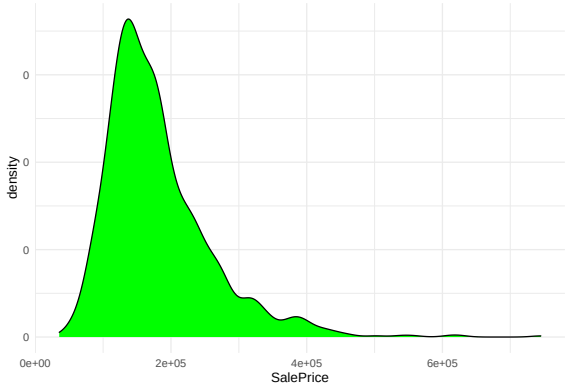
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
35311	129250	164000	180934	214250	745000

Affichons des graphiques descriptives de la variable SalePrice.

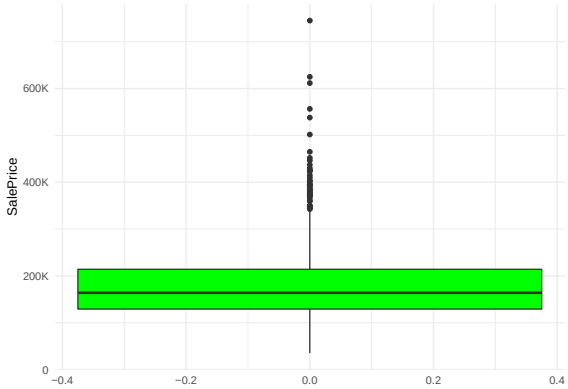
```
# Histogramme
ggplot(train) +
  aes(x = SalePrice) +
  geom_histogram(bins = 30L, fill = "green") +
  theme_minimal() +
  scale_x_continuous(labels = scales::label_number_si())
```




```
# Densité de probabilté
ggplot(train) +
  aes(x = SalePrice) +
  geom_density(fill = "green") +
  theme_minimal() +
  scale_y_continuous(labels = scales::label_number_si())
```

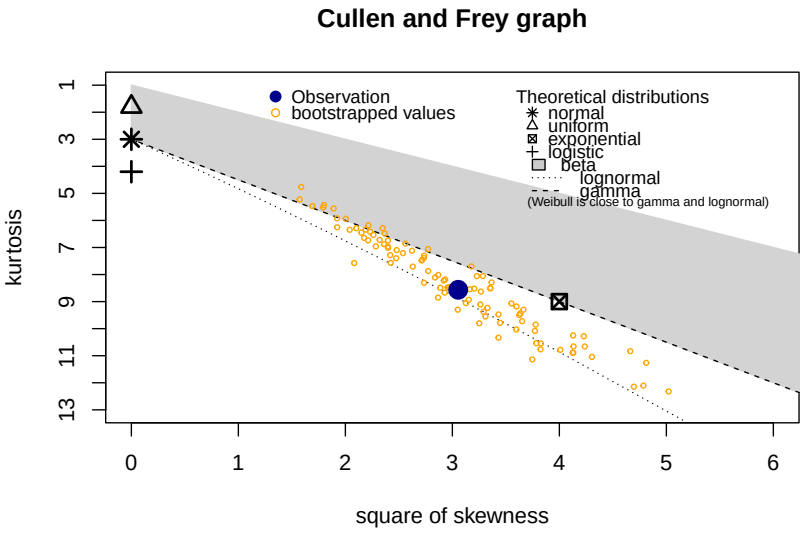



```
# Boite à moustache
ggplot(train) +
  aes(y = SalePrice) +
  geom_boxplot(fill = "green") +
  theme_minimal() +
  scale_y_continuous(labels = scales::label_number_si())
```



La distribution de la variable SalePrice semble asymétrique, ce qui pourrait réduire les effets d'une regression. Nous pouvons utiliser le diagramme de Cullen-Frey pour mieux étudier cette distribution, en la comparant à d'autres lois usuelles.

```
descdist(train$SalePrice, discrete = FALSE, boot = 100)
```




```
## summary statistics
## ------
## min:  35311   max:  745000
## median:  164000
## mean:  180933.9
## estimated sd:  77918.79
## estimated skewness:  1.747786
## estimated kurtosis:  8.564094
```

Le point bleu du graphique ci-dessous représente la distribution de la variable SalePrice qui est très proche de la loi log-normale. Une variable suivant une loi log-normale suit une loi normale par application du logarithme. La loi normale étant parfaitement symétrique, l'utilisation du logarithme de notre variable nous garantirait de meilleurs effets sur une regression.

Démontrons que notre variable SalePrice ne suit pas une loi normale par un test de Shapiro-Wilk.

```
shapiro.test(train$SalePrice)
```

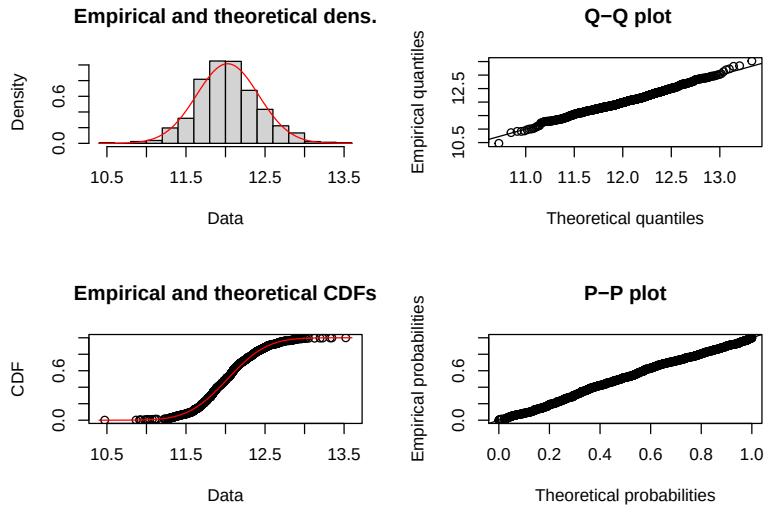
```
##
##  Shapiro-Wilk normality test
##
## data:  train$SalePrice
## W = 0.88059, p-value < 2.2e-16
```

La p-value de ce test est très faible (2.2e-16 < 5%), nous rejetons ainsi le caractère gaussien de la distribution de la variable SalePrice.

Nous pouvons appliquer le logarithme sur notre variable SalePrice pour la rapprocher de la loi normale.

Comparons graphiquement la loi normale à la distribution du logarithme de notre variable cible.

```
plot(fitdist(log(train$SalePrice), "norm"))
```



Ainsi, le logarithme de la variable SalePrice est très proche d'une loi log-normale. Nous considérerons donc le logarithme de cette variable dans notre regression.

2.2. Etude de la distribution des variables indépendantes

2.2.1. Analyse des données quantitatives

Extrayons les variables quantitatives de notre jeu de données.

```
train_numerical <- train %>% select_if(is.numeric)
test_numerical <- test %>% select_if(is.numeric)

dim(train_numerical)
```

```
## [1] 1095   37
```

```
dim(test_numerical)
```

```
## [1] 365  37
```

Observons les histogrammes des variables quantitatives pour avoir l'allure de leur distribution.

Histogrammes pour le train :

```
p <- list()
i <- 1
for (col in names(train_numerical)) {
  p[[i]] <- ggplot(train_numerical, aes_string(x = col)) + geom_histogram(bins = 30L, fill = "green'
  i <- i + 1
}
do.call("grid.arrange", c(p, nrow = 10, ncol = 5))
```


```
rm(p, i, col)
```

Histogrammes pour le test :

```
p <- list()
i <- 1
```

```
for (col in names(test_numerical)) {
  p[[i]] <- ggplot(test_numerical, aes_string(x = col)) + geom_histogram(bins = 30L, fill = "purple"
  i <- i + 1
}
do.call("grid.arrange", c(p, nrow = 10, ncol = 5))
```

```
rm(p, i, col)
```

Les résultats des graphiques ci-dessus indiquent que les deux échantillons (train et test) proviennent de la même population étant donné que leurs distributions suivent la même allure. Cependant, certaines variables semblent très constantes par rapport à la variable cible, examinons-les plus en détail.

2.2.1.a. Mise à l'écart de variable quantitative constante Sélectionnons les variables quantitatives avec une variance très faible.

```
variances <- apply(train_numerical, 2, var)
constant_var_list <- names(variances[which(variances <= 0.1)])
constant_var_list
```

```
## [1] "BsmtHalfBath" "KitchenAbvGr"
```

Compte tenu de la variance élevée de notre variable cible (variance, min max, mean). Nous supprimons ces variables quantitatives presque constante des variables à considérer pour notre regression.

```
train_numerical <- train_numerical[, !(names(train_numerical) %in% constant_var_list)]
test_numerical <- test_numerical[, !(names(test_numerical) %in% constant_var_list)]
dim(train_numerical)
```

```
## [1] 1095   35
```

```
dim(test_numerical)
```

```
## [1] 365  35
```

2.2.1.b. Etude des relations entre les variables indépendantes et la variable cible À partir de l'année de vente d'une maison (variable YrSold), nous créons les nouvelles variables qui définissent l'age de construction de la maison, l'age de la renovation et lage de construction du garage en fonction des années spécifiques.

```
train_numerical$Age_Maison <- with(train_numerical, train_numerical$YrSold - YearBuilt)
train_numerical$Age_Depuis_Reno <- with(train_numerical, train_numerical$YrSold - YearRemodAdd)
train_numerical$Age_Garage <- with(train_numerical, train_numerical$YrSold - GarageYrBlt)

test_numerical$Age_Maison <- with(test_numerical, test_numerical$YrSold - YearBuilt)
test_numerical$Age_Depuis_Reno <- with(test_numerical, test_numerical$YrSold - YearRemodAdd)
test_numerical$Age_Garage <- with(test_numerical, test_numerical$YrSold - GarageYrBlt)

train_numerical <- select(train_numerical, -YearBuilt)
train_numerical <- select(train_numerical, -YearRemodAdd)
train_numerical <- select(train_numerical, -GarageYrBlt)

test_numerical <- select(test_numerical, -YearBuilt)
test_numerical <- select(test_numerical, -YearRemodAdd)
test_numerical <- select(test_numerical, -GarageYrBlt)

#Deplacement de la colonne SalePrice à la fin de chaque data frame
train_numerical <- select(train_numerical, -SalePrice)
```

```
test_numerical <- select(test_numerical, -SalePrice)
train_numerical$SalePrice <- train$SalePrice
test_numerical$SalePrice <- test$SalePrice

print("YrSold" %in% names(train_numerical))
```

```
## [1] TRUE
```

```
print("YearBuilt" %in% names(train_numerical))
```

```
## [1] FALSE
```

```
print("YearRemodAdd" %in% names(train_numerical))
```

```
## [1] FALSE
```

```
print("GarageYrBlt" %in% names(train_numerical))
```

```
## [1] FALSE
```

```
print("YrSold" %in% names(test_numerical))
```

```
## [1] TRUE
```

```
print("YearBuilt" %in% names(test_numerical))
```

```
## [1] FALSE
```

```
print("YearRemodAdd" %in% names(test_numerical))
```

```
## [1] FALSE
```

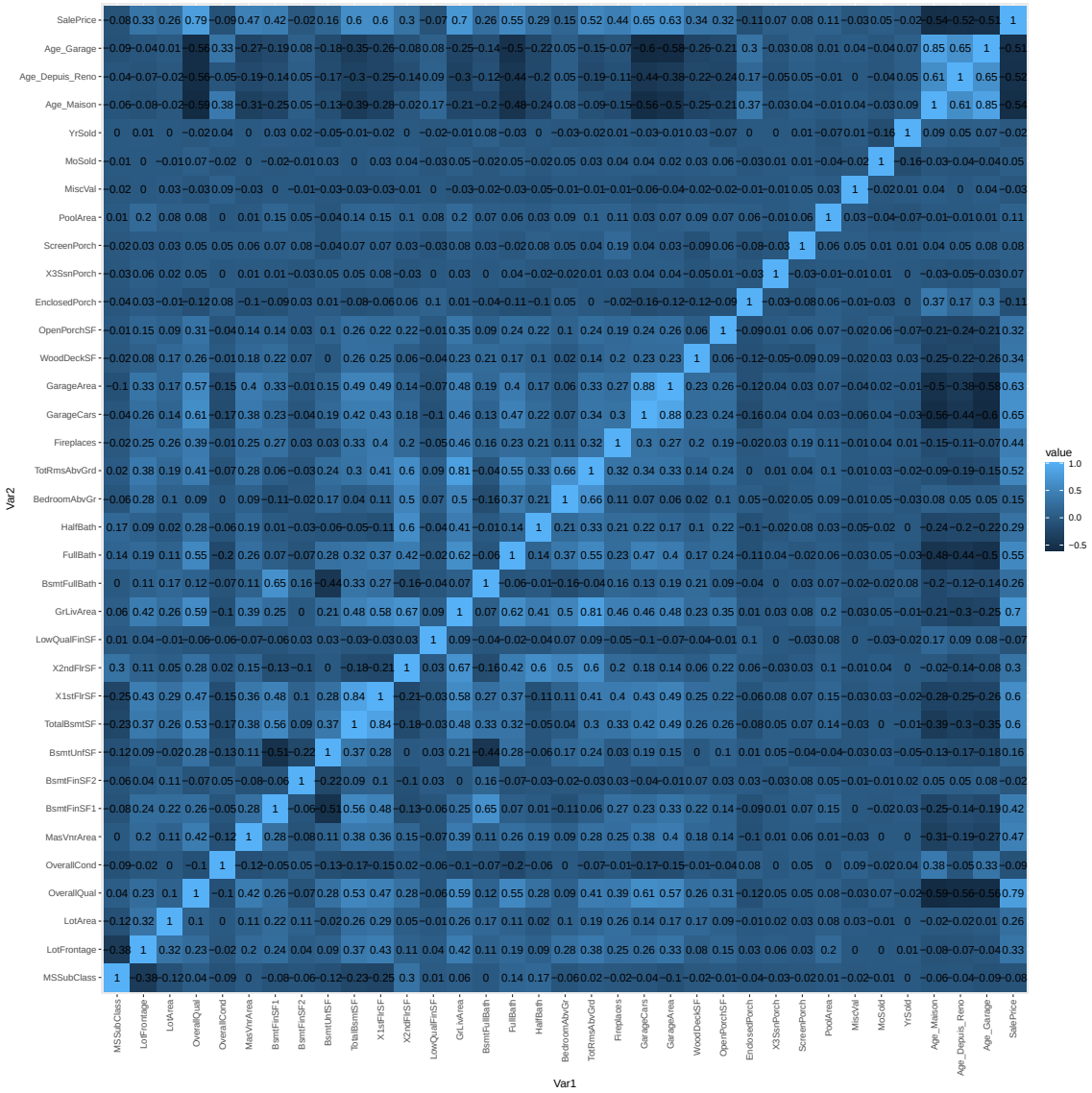
```
print("GarageYrBlt" %in% names(test_numerical))
```

```
## [1] FALSE
```

Evaluons la corrélation des variables indépendantes par rapport à la variable cible à partir d'une matrice de corrélation. Ce tableau nous donne un aperçu des variables fortement corrélées à la variable cible et qui seront importantes pour notre regression.

```
correlation <- list()
abs_correlation <- list()
for (col in names(train_numerical)) {
  correlation[col] <- cor(train_numerical[col], train_numerical$SalePrice)
  abs_correlation[col] <- abs(cor(train_numerical[col], train_numerical$SalePrice))
}
# considered_correlation <- names(abs_correlation[which(abs_correlation >= 0.3)])
considered_correlation <- names(abs_correlation)
```

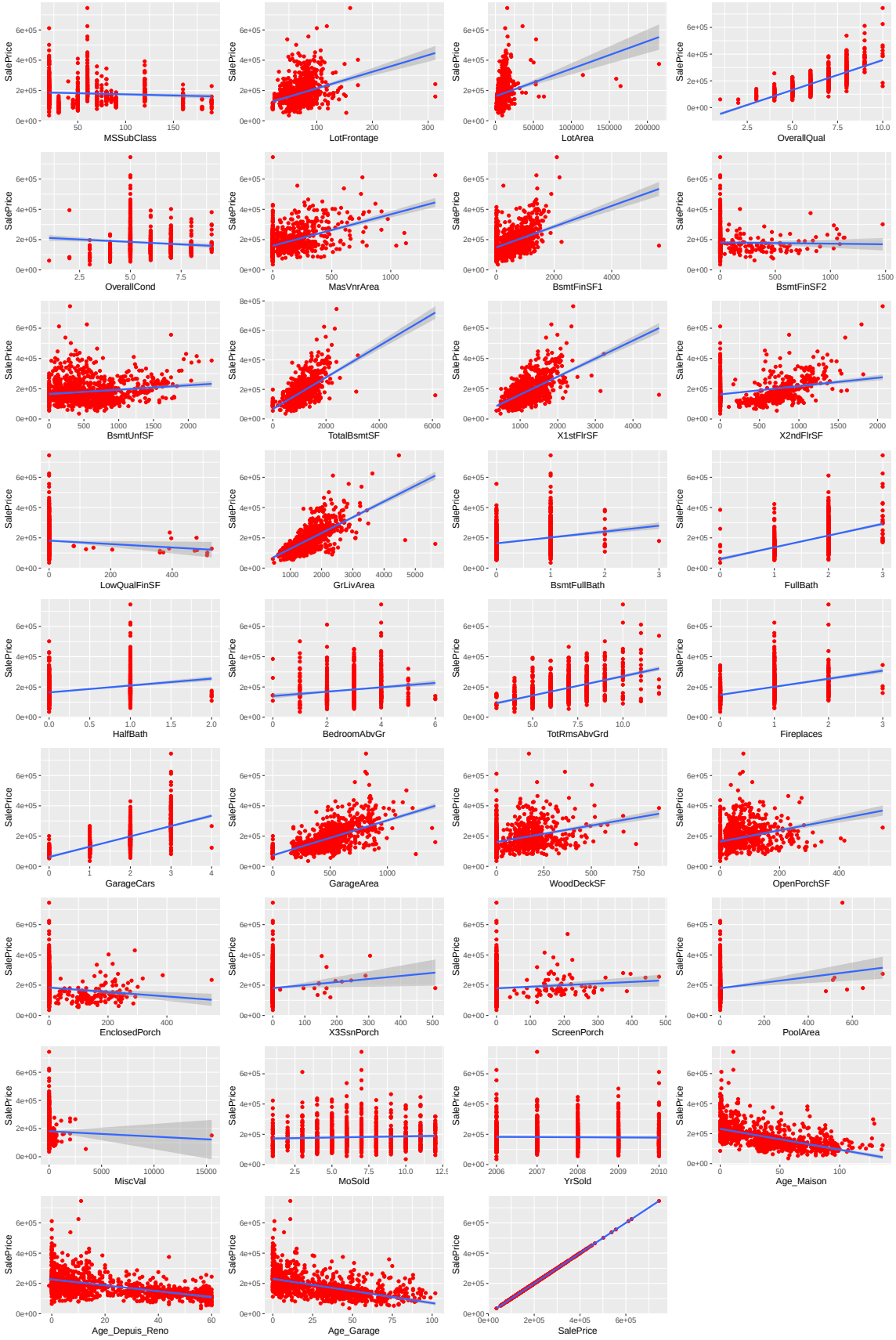
```
considered_train_numerical <- train_numerical[, (names(train_numerical) %in% considered_correlation)]

corr_mat <- round(cor(considered_train_numerical), 2)
# reduce the size of correlation matrix
melted_corr_mat <- melt(corr_mat)

ggplot(data = melted_corr_mat, aes(x = Var1, y = Var2, fill = value)) +
  geom_tile() +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```




```
rm(corr_mat, melted_corr_mat)
```

Evaluons la corrélation des variables indépendantes par rapport à la variable cible à partir d'un nuage de point et une droite de corrélation linéaire.

```r
p <- list()
i <- 1
for (col in names(train_numerical)) {
  p[[i]] <- ggplot(train_numerical, aes_string(x = col, y = 'SalePrice')) +
    geom_point(color = 'red') +
    stat_smooth(method = lm, formula = 'y ~ x')
  i <- i + 1
}
do.call("grid.arrange", c(p, nrow = 10, ncol = 4))
```


SalePrice
MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
X1stFlrSF
X2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
FullBath
HalfBath
BedroomAbvGr
TotRmsAbvGrd
Fireplaces
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
X3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
Age_Maison
Age_Depuis_Reno
Age_Garage
SalePrice

```
rm(p, i, col)
```

Procédons à la mise à l'échelle des variables indépendantes. La mise à l'échelle est un moyen de comparer des données qui ne sont pas mesurées de la même manière. Dans la mise à l'échelle standard, également connue sous le nom de normalisation des valeurs, nous mettons à l'échelle les valeurs des données de sorte que le résumé statistique global de chaque variable ait une valeur moyenne de zéro et une valeur de variance unitaire. Nous écartons la variable cible à cette étape pour la normaliser à partir d'un logarithme.

```
train_numeric <- select(train_numerical, -SalePrice)
test_numeric <- select(test_numerical, -SalePrice)

train_numeric <- as.data.frame(scale(train_numeric))
test_numeric <- as.data.frame(scale(test_numeric))

train_numeric$SalePrice <- train$SalePrice
test_numeric$SalePrice <- test$SalePrice

dim(train_numeric)
```

```
## [1] 1095   35
```

```
dim(test_numeric)
```

```
## [1] 365  35
```

```
print("SalePrice" %in% names(train_numeric))
```

```
## [1] TRUE
```

```
print("SalePrice" %in% names(test_numeric))
```

```
## [1] TRUE
```

2.2.2. Analyse des données qualitatives

Extrayons les variables qualitatives de notre jeu de données.

```
train_cathegorical <- train %>% select_if(Negate(is.numeric))
test_cathegorical <- test %>% select_if(Negate(is.numeric))

dim(train_cathegorical)
```

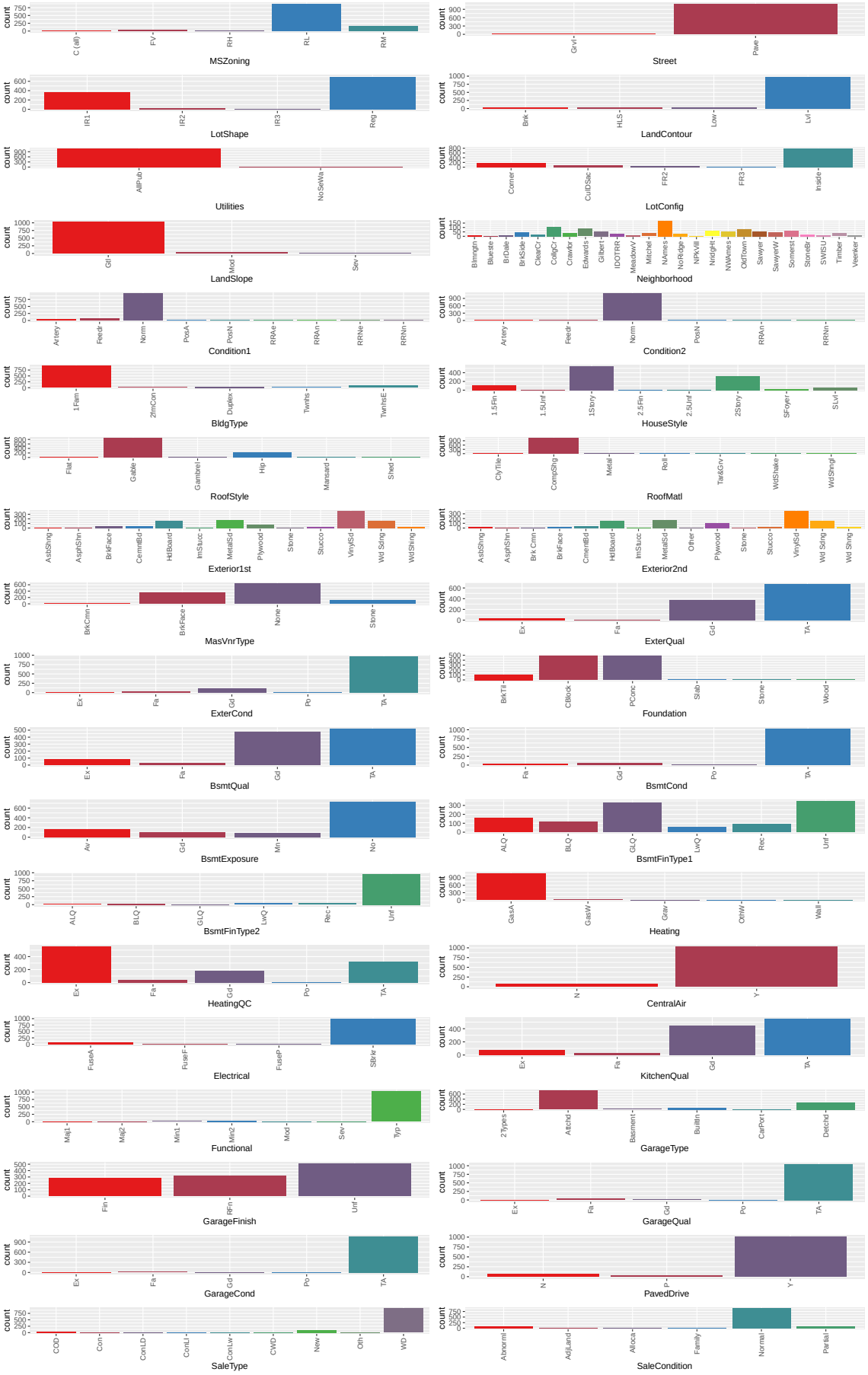
```
## [1] 1095   38
```

```
dim(test_cathegorical)
```

```
## [1] 365  38
```

Observons les variables qualitatives à partir de diagramme en baton et vérifions si certaines de ces variables sont quasi constantes (dominées par l'une de leurs modalités)

```
p <- list()
i <- 1
mycolors <- colorRampPalette(brewer.pal(9, "Set1"))(25)
for (col in names(train_cathegorical)) {
  p[[i]] <- ggplot(train_cathegorical, aes_string(x = col, fill = col)) +
    geom_bar() +
    scale_fill_manual(values = mycolors) +
    theme(legend.position = "none") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
  i <- i + 1
}
do.call("grid.arrange", c(p, nrow = 20, ncol = 2))
```


count
C (all)
FV
RH
RL
RM
MSZoning
Grvl
Pave
Street
IR1
IR2
IR3
Reg
LotShape
Bnk
HLS
Low
Lvl
LandContour
AllPub
NoSeWa
Utilities
Corner
CulDSac
FR2
FR3
Inside
LotConfig
Gtl
Mod
Sev
LandSlope
Blmngtn
Blueste
BrDale
BrkSide
ClearCr
CollgCr
Crawfor
Edwards
Gilbert
IDOTRR
MeadowV
Mitchel
NAmes
NoRidge
NPkVill
NridgHt
NWAmes
OldTown
Sawyer
SawyerW
Somerst
StoneBr
SWISU
Timber
Veenker
Neighborhood
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNe
RRNn
Condition1
Condition2
1Fam
2fmCon
Duplex
Twnhs
TwnhsE
BldgType
1.5Fin
1.5Unf
1Story
2.5Fin
2.5Unf
2Story
SFoyer
SLvl
HouseStyle
Flat
Gable
Gambrel
Hip
Mansard
Shed
RoofStyle
ClyTile
CompShg
Metal
Roll
Tar&Grv
WdShake
WdShngl
RoofMatl
AsbShng
AsphShn
BrkFace
CemntBd
HdBoard
ImStucc
MetalSd
Plywood
Stone
Stucco
VinylSd
Wd Sdng
WdShing
Exterior1st
Brk Cmn
BrkFace
CmentBd
Other
Exterior2nd
BrkCmn
None
MasVnrType
Ex
Fa
Gd
TA
ExterQual
Po
ExterCond
BrkTil
CBlock
PConc
Slab
Wood
Foundation
BsmtQual
BsmtCond
Av
Mn
No
BsmtExposure
ALQ
BLQ
GLQ
LwQ
Rec
Unf
BsmtFinType1
BsmtFinType2
GasA
GasW
Grav
OthW
Wall
Heating
HeatingQC
N
Y
CentralAir
FuseA
FuseF
FuseP
SBrkr
Electrical
KitchenQual
Maj1
Maj2
Min1
Min2
Mod
Sev
Typ
Functional
2Types
Attchd
Basment
BuiltIn
CarPort
Detchd
GarageType
Fin
RFn
Unf
GarageFinish
GarageQual
GarageCond
P
PavedDrive
COD
Con
ConLD
ConLI
ConLw
CWD
New
Oth
WD
SaleType
Abnorml
AdjLand
Alloca
Family
Normal
Partial
SaleCondition

```
rm(p, i, col, mycolors)
```

D'après le graphique ci-dessus, on remarque facilement que certaines des variables sont quasi constantes, nous les sélectionnons pour les supprimer des jeux de données.

```
to_drop <- c('MSZoning', 'Street', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
             'Condition1', 'Condition2', 'BldgType', 'RoofMatl', 'RoofStyle', 'ExterCond',
             'BsmtCond', 'BsmtFinType2', 'Heating', 'CentralAir', 'Electrical', 'Functional',
             'GarageQual', 'GarageCond', 'PavedDrive', 'SaleType', 'SaleCondition')

train_cathegorical <- train_cathegorical[, !(names(train_cathegorical) %in% to_drop)]
test_cathegorical <- test_cathegorical[, !(names(test_cathegorical) %in% to_drop)]

dim(train_cathegorical)
```

```
## [1] 1095   15
```

```
dim(test_cathegorical)
```

```
## [1] 365  15
```

Transformons les variables catégorielles en facteur pour faciliter la regression en permettant la conversion implicite en valeur numérique par les modules de regression.

```
factor_names <- names(train_cathegorical)
train_cathegorical[, factor_names] <- lapply(train_cathegorical[, factor_names], as.factor)
test_cathegorical[, factor_names] <- lapply(test_cathegorical[, factor_names], as.factor)
factor_names
```

```
##  [1] "LotShape"     "Neighborhood" "HouseStyle"   "Exterior1st"  "Exterior2nd"
##  [6] "MasVnrType"   "ExterQual"    "Foundation"   "BsmtQual"     "BsmtExposure"
## [11] "BsmtFinType1" "HeatingQC"    "KitchenQual"  "GarageType"   "GarageFinish"
```

3. Validité des modèles

3.1. Implémentation d'un modèle de base

Nous utilisons les variables numériques et les variables catégorielles pour former un seul jeu de données.

```
train_final1 <- cbind(train_numeric, train_cathegorical)
dim(train_final1)
```

```
## [1] 1095   50
```

```
test_final1 <- cbind(test_numeric, test_cathegorical)
dim(test_final1)
```

```
## [1] 365  50
```

Implémentons un modèle de base et résumons-le.

```r
model0 <- lm(log(SalePrice) ~ ., data = train_final1)
summary(model0)
```

```
##
## Call:
## lm(formula = log(SalePrice) ~ ., data = train_final1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.03285 -0.06019  0.00320  0.07069  0.41855
##
## Coefficients: (2 not defined because of singularities)
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)          11.7936886  0.1268979  92.938  < 2e-16 ***
## MSSubClass           -0.0319185  0.0062648  -5.095 4.20e-07 ***
## LotFrontage          -0.0175635  0.0057413  -3.059 0.002281 **
## LotArea               0.0181266  0.0052311   3.465 0.000553 ***
## OverallQual           0.0747287  0.0085464   8.744  < 2e-16 ***
## OverallCond           0.0440854  0.0055290   7.973 4.34e-15 ***
## MasVnrArea           -0.0072706  0.0066812  -1.088 0.276772
## BsmtFinSF1            0.0014634  0.0127660   0.115 0.908761
## BsmtFinSF2            0.0008039  0.0059069   0.136 0.891772
## BsmtUnfSF             0.0071208  0.0107476   0.663 0.507780
## TotalBsmtSF                  NA         NA      NA       NA
## X1stFlrSF             0.0669473  0.0123516   5.420 7.52e-08 ***
## X2ndFlrSF             0.0804530  0.0145070   5.546 3.78e-08 ***
## LowQualFinSF          0.0074341  0.0048737   1.525 0.127494
## GrLivArea                    NA         NA      NA       NA
## BsmtFullBath          0.0260095  0.0061395   4.236 2.49e-05 ***
## FullBath              0.0209416  0.0074652   2.805 0.005129 **
## HalfBath              0.0223297  0.0064182   3.479 0.000525 ***
## BedroomAbvGr          0.0107036  0.0065607   1.631 0.103118
## TotRmsAbvGrd          0.0184974  0.0087844   2.106 0.035488 *
## Fireplaces            0.0199271  0.0053424   3.730 0.000203 ***
## GarageCars            0.0489886  0.0099030   4.947 8.89e-07 ***
## GarageArea            0.0071493  0.0102855   0.695 0.487171
## WoodDeckSF            0.0181154  0.0046438   3.901 0.000102 ***
## OpenPorchSF           0.0004712  0.0047118   0.100 0.920354
## EnclosedPorch         0.0104941  0.0047263   2.220 0.026627 *
## X3SsnPorch            0.0103468  0.0040691   2.543 0.011154 *
## ScreenPorch           0.0172716  0.0042772   4.038 5.82e-05 ***
## PoolArea             -0.0026329  0.0044508  -0.592 0.554278
## MiscVal               0.0005352  0.0040867   0.131 0.895838
## MoSold                0.0002382  0.0041465   0.057 0.954201
## YrSold               -0.0042525  0.0043199  -0.984 0.325164
## Age_Maison           -0.0284263  0.0136372  -2.084 0.037380 *
## Age_Depuis_Reno      -0.0168188  0.0071328  -2.358 0.018575 *
## Age_Garage            0.0054442  0.0097178   0.560 0.575456
## LotShapeIR2           0.0464739  0.0269126   1.727 0.084516 .
## LotShapeIR3          -0.1466896  0.0559061  -2.624 0.008831 **
## LotShapeReg          -0.0008348  0.0097466  -0.086 0.931763
## NeighborhoodBlueste  -0.1530210  0.1394293  -1.097 0.272705
## NeighborhoodBrDale   -0.1657872  0.0647152  -2.562 0.010564 *
## NeighborhoodBrkSide  -0.0001372  0.0516489  -0.003 0.997880
## NeighborhoodClearCr   0.0269025  0.0525899   0.512 0.609082
```

```
## NeighborhoodCollgCr  0.0047287  0.0422834   0.112 0.910980
## NeighborhoodCrawfor  0.1304102  0.0497599   2.621 0.008910 **
## NeighborhoodEdwards -0.1142153  0.0469974  -2.430 0.015270 *
## NeighborhoodGilbert -0.0249011  0.0453553  -0.549 0.583117
## NeighborhoodIDOTRR  -0.1334189  0.0545140  -2.447 0.014565 *
## NeighborhoodMeadowV -0.2496705  0.0656239  -3.805 0.000151 ***
## NeighborhoodMitchel -0.0607114  0.0485914  -1.249 0.211811
## NeighborhoodNAmes   -0.0454498  0.0458746  -0.991 0.322061
## NeighborhoodNoRidge  0.1394508  0.0510804   2.730 0.006448 **
## NeighborhoodNPkVill -0.0490101  0.0898551  -0.545 0.585581
## NeighborhoodNridgHt  0.1502282  0.0451629   3.326 0.000913 ***
## NeighborhoodNWAmes  -0.0627510  0.0470531  -1.334 0.182642
## NeighborhoodOldTown -0.1169169  0.0491259  -2.380 0.017508 *
## NeighborhoodSawyer  -0.0575257  0.0485214  -1.186 0.236082
## NeighborhoodSawyerW  0.0083334  0.0461370   0.181 0.856701
## NeighborhoodSomerst  0.0882843  0.0444679   1.985 0.047388 *
## NeighborhoodStoneBr  0.2136254  0.0519058   4.116 4.19e-05 ***
## NeighborhoodSWISU   -0.0396916  0.0612104  -0.648 0.516851
## NeighborhoodTimber   0.0100190  0.0475168   0.211 0.833047
## NeighborhoodVeenker  0.0287870  0.0624534   0.461 0.644949
## HouseStyle1.5Unf     0.0120985  0.0455046   0.266 0.790393
## HouseStyle1Story     0.0171782  0.0236957   0.725 0.468657
## HouseStyle2.5Fin    -0.0483513  0.0809014  -0.598 0.550209
## HouseStyle2.5Unf     0.0445263  0.0553876   0.804 0.421651
## HouseStyle2Story    -0.0256069  0.0203944  -1.256 0.209570
## HouseStyleSFoyer     0.0496113  0.0354836   1.398 0.162389
## HouseStyleSLvl       0.0384217  0.0290200   1.324 0.185825
## Exterior1stAsphShn  -0.0522271  0.1853041  -0.282 0.778123
## Exterior1stBrkFace   0.1997528  0.0791869   2.523 0.011810 *
## Exterior1stCemntBd   0.0742551  0.1261807   0.588 0.556346
## Exterior1stHdBoard   0.0583529  0.0803870   0.726 0.468076
## Exterior1stImStucc   0.0187508  0.1605628   0.117 0.907057
## Exterior1stMetalSd   0.0614468  0.0899169   0.683 0.494535
## Exterior1stPlywood   0.1073566  0.0796255   1.348 0.177888
## Exterior1stStone     0.1051982  0.1326158   0.793 0.427824
## Exterior1stStucco    0.0764350  0.0965142   0.792 0.428580
## Exterior1stVinylSd   0.0749832  0.0814440   0.921 0.357451
## Exterior1stWd Sdng   0.0571246  0.0777839   0.734 0.462882
## Exterior1stWdShing   0.0776097  0.0842990   0.921 0.357464
## Exterior2ndAsphShn   0.0740555  0.1219641   0.607 0.543867
## Exterior2ndBrk Cmn  -0.0524490  0.1310585  -0.400 0.689101
## Exterior2ndBrkFace  -0.0291024  0.0823532  -0.353 0.723877
## Exterior2ndCmentBd   0.0348024  0.1248567   0.279 0.780505
## Exterior2ndHdBoard   0.0009992  0.0761503   0.013 0.989534
## Exterior2ndImStucc   0.0583027  0.0879007   0.663 0.507310
## Exterior2ndMetalSd   0.0279311  0.0866688   0.322 0.747315
## Exterior2ndOther    -0.0910162  0.1569922  -0.580 0.562218
## Exterior2ndPlywood  -0.0077827  0.0739461  -0.105 0.916200
## Exterior2ndStone    -0.0960448  0.1027799  -0.934 0.350294
## Exterior2ndStucco   -0.0493873  0.0948549  -0.521 0.602722
## Exterior2ndVinylSd   0.0297301  0.0781893   0.380 0.703856
## Exterior2ndWd Sdng   0.0048971  0.0740553   0.066 0.947289
## Exterior2ndWd Shng  -0.0351262  0.0786110  -0.447 0.655094
## MasVnrTypeBrkFace    0.0549568  0.0429208   1.280 0.200703
## MasVnrTypeNone       0.0481760  0.0431975   1.115 0.265021
## MasVnrTypeStone      0.0489681  0.0451085   1.086 0.277943
```

```
## ExterQualFa         -0.0351047  0.0585765  -0.599 0.549116
## ExterQualGd          0.0460201  0.0304964   1.509 0.131617
## ExterQualTA          0.0512155  0.0333616   1.535 0.125071
## FoundationCBlock     0.0343825  0.0191835   1.792 0.073398 .
## FoundationPConc      0.0422993  0.0215798   1.960 0.050268 .
## FoundationSlab      -0.0259048  0.0476605  -0.544 0.586892
## FoundationStone      0.0880310  0.0576903   1.526 0.127356
## FoundationWood       0.0013327  0.1016054   0.013 0.989537
## BsmtQualFa          -0.1033979  0.0397901  -2.599 0.009504 **
## BsmtQualGd          -0.0743516  0.0208759  -3.562 0.000387 ***
## BsmtQualTA          -0.0809069  0.0258168  -3.134 0.001777 **
## BsmtExposureGd       0.0495611  0.0185111   2.677 0.007546 **
## BsmtExposureMn      -0.0094772  0.0188312  -0.503 0.614888
## BsmtExposureNo      -0.0193018  0.0135831  -1.421 0.155636
## BsmtFinType1BLQ     -0.0073426  0.0173934  -0.422 0.673013
## BsmtFinType1GLQ     -0.0072068  0.0158974  -0.453 0.650410
## BsmtFinType1LwQ     -0.0234985  0.0232879  -1.009 0.313208
## BsmtFinType1Rec     -0.0158632  0.0191614  -0.828 0.407946
## BsmtFinType1Unf     -0.0637002  0.0178567  -3.567 0.000378 ***
## HeatingQCFa         -0.0291212  0.0255300  -1.141 0.254291
## HeatingQCGd         -0.0193182  0.0130865  -1.476 0.140216
## HeatingQCPo         -0.2733936  0.1410799  -1.938 0.052931 .
## HeatingQCTA         -0.0213451  0.0127795  -1.670 0.095191 .
## KitchenQualFa       -0.1045464  0.0373190  -2.801 0.005189 **
## KitchenQualGd       -0.0521068  0.0214545  -2.429 0.015334 *
## KitchenQualTA       -0.0593116  0.0241738  -2.454 0.014321 *
## GarageTypeAttchd     0.2216289  0.0977553   2.267 0.023600 *
## GarageTypeBasment    0.1643111  0.1032832   1.591 0.111964
## GarageTypeBuiltIn    0.1960721  0.0995248   1.970 0.049114 *
## GarageTypeCarPort    0.1270874  0.1098533   1.157 0.247606
## GarageTypeDetchd     0.1944289  0.0970664   2.003 0.045451 *
## GarageFinishRFn     -0.0141894  0.0124575  -1.139 0.254977
## GarageFinishUnf     -0.0222061  0.0153217  -1.449 0.147572
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1285 on 966 degrees of freedom
## Multiple R-squared:  0.9058, Adjusted R-squared:  0.8933
## F-statistic: 72.56 on 128 and 966 DF,  p-value: < 2.2e-16
```

Le modèle de base créé contient 2 variables non définies à cause des singularités. Ce qui signifie que ces variables ont des relations linéaires exactes avec d'autres variables. Ces variables non définies pourraient prédire des variables non définies. Nous devons donc améliorer le modèle pour éliminer ces variables.

3.1.1. Amélioration du modèle de base par selection de variable pertinente

Implémentons un modèle Random Forest pour sélectionner les variables pertinentes pour notre regression.

```
model_tree0 <- randomForest(log(SalePrice) ~ ., data = train_final1)
print(model_tree0)
```

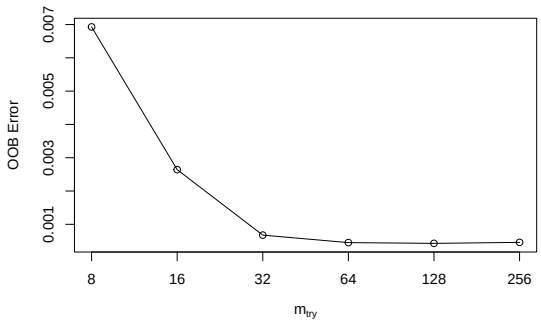
```
##
## Call:
##  randomForest(formula = log(SalePrice) ~ ., data = train_final1)
```

```
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 16
##
##           Mean of squared residuals: 0.01877307
##                     % Var explained: 87.85
```

Optimisons le modèle Random Forest en recherchant le paramètre optimale mtry

```
t <- tuneRF(train_final1[, -50], log(train_final1$SalePrice), stepFactor = 0.5, plot = TRUE, ntreeTr
```

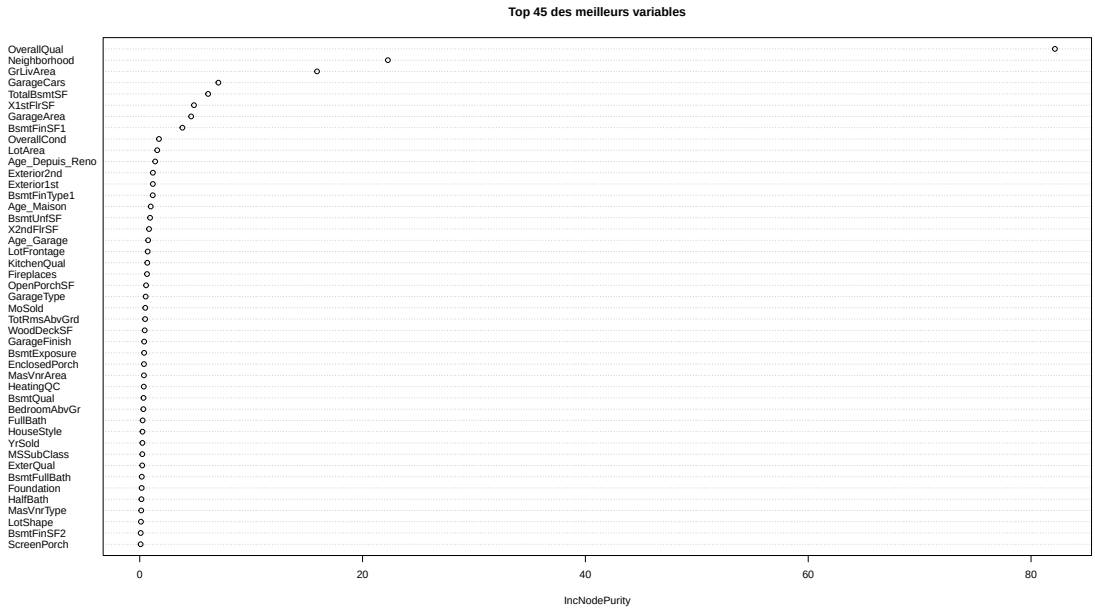
```
## 0.7442812 0.05
## 0.332913 0.05
## 0.05044845 0.05
## -0.06998692 0.05
## -15.1857 0.05
```



D'après le graphique ci-dessus, l'erreur OOB minimal est obtenu en testant environ 64 variables, soit 50 variables pour le maximum de notre dataset.

Observons les variables les plus importantes pour notre prédiction et conservons-les dans un dataframe par odre d'importance.

```
model_tree1 <- randomForest(log(SalePrice) ~ ., data = train_final1, ntree = 500, mtry = 50)
varImpPlot(model_tree1, sort = T, n.var = 45, main = "Top 45 des meilleurs variables")
```




```
train_result <- data.frame(variables = colnames(train_final1[1:49]), IncNodePurity = format(model_tr
train_result <- train_result[order(train_result$IncNodePurity, decreasing = TRUE),]
```

Nous sélectionnons les 42 meilleurs variables pour notre prédiction.

```
selected_var <- train_result[1:42, 1]
train_final11 <- train_final1[, c(selected_var, 'SalePrice')]
test_final2 <- test_final1[, c(selected_var, 'SalePrice')]
names(train_final11)
```

```
##  [1] "OverallQual"     "LotShape"        "GrLivArea"       "GarageCars"
##  [5] "TotalBsmtSF"     "X1stFlrSF"       "GarageArea"      "BsmtFinSF1"
##  [9] "OverallCond"     "LotArea"         "Age_Depuis_Reno" "Exterior1st"
## [13] "HouseStyle"      "BsmtExposure"    "Age_Maison"      "BsmtUnfSF"
## [17] "X2ndFlrSF"       "Age_Garage"      "LotFrontage"     "HeatingQC"
## [21] "Fireplaces"      "OpenPorchSF"     "KitchenQual"     "MoSold"
## [25] "TotRmsAbvGrd"    "WoodDeckSF"      "GarageType"      "BsmtQual"
## [29] "EnclosedPorch"   "MasVnrArea"      "BsmtFinType1"    "Foundation"
## [33] "BedroomAbvGr"    "FullBath"        "Neighborhood"    "YrSold"
## [37] "MSSubClass"      "MasVnrType"      "BsmtFullBath"    "ExterQual"
## [41] "HalfBath"        "Exterior2nd"     "SalePrice"
```

3.1.2. Amélioration du modèle de base par suppression de valeur aberrante

Nous pouvons utiliser la distance de Cook pour identifier les valeurs aberrantes. La distance de Cook est une estimation de l'influence d'un point de données. Il prend en compte à la fois l'effet de levier et le résidu de chaque observation. La distance de Cook est un résumé de la variation d'un modèle de régression lorsque la ième observation est supprimée.

Lorsque nous cherchons à voir quelles observations peuvent être des valeurs aberrantes, une règle générale consiste à étudier tout point supérieur à un seuil. Nous choisissons comme seuil les points supérieurs à 4 fois la moyenne de toutes les distances.

Identifions les valeurs aberrantes.

```
model1 <- lm(log(SalePrice) ~ ., data = train_final11)
cooksD <- cooks.distance(model1)
influential <- cooksD[(cooksD > (4 * mean(cooksD, na.rm = TRUE)))]
influential
```

```
##       <NA>         21        217        332        362        378        379
##         NA 0.02326700 0.01497603 0.04002391 0.01751054 0.02312401 0.01214823
##        388       <NA>        585        641        661       <NA>        720
## 0.02326700         NA 0.03069743 0.01103592 0.04626887         NA 1.35172600
##        743        749        783        794        814        819        829
## 0.01439464 0.01164247 0.01313846 0.01210350 0.01943942 0.14976018 0.04751487
##       <NA>        884       <NA>        960       1079
##         NA 0.01313846         NA 0.01186436 0.01377455
```

Implémentons notre modèle de regression de base avec nos variables finales sans les valeurs aberrantes, puis résumons-le.

```
names_of_influential <- names(influential)
outliers <- train_final11[names_of_influential,]
train_final2 <- train_final11 %>% anti_join(outliers)
```

```
## Joining, by = c("OverallQual", "LotShape", "GrLivArea", "GarageCars",
## "TotalBsmtSF", "X1stFlrSF", "GarageArea", "BsmtFinSF1", "OverallCond",
## "LotArea", "Age_Depuis_Reno", "Exterior1st", "HouseStyle", "BsmtExposure",
## "Age_Maison", "BsmtUnfSF", "X2ndFlrSF", "Age_Garage", "LotFrontage",
## "HeatingQC", "Fireplaces", "OpenPorchSF", "KitchenQual", "MoSold",
## "TotRmsAbvGrd", "WoodDeckSF", "GarageType", "BsmtQual", "EnclosedPorch",
## "MasVnrArea", "BsmtFinType1", "Foundation", "BedroomAbvGr", "FullBath",
## "Neighborhood", "YrSold", "MSSubClass", "MasVnrType", "BsmtFullBath",
## "ExterQual", "HalfBath", "Exterior2nd", "SalePrice")
```

```
model2 <- lm(log(SalePrice) ~ ., data = train_final2)
summary(model2)
```

```
##
## Call:
## lm(formula = log(SalePrice) ~ ., data = train_final2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43166 -0.04955  0.00396  0.05465  0.34047
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.207e+01  6.266e-02 192.667  < 2e-16 ***
## OverallQual          6.664e-02  6.545e-03  10.182  < 2e-16 ***
## LotShapeIR2          7.803e-03  2.112e-02   0.369 0.711916
## LotShapeIR3         -5.545e-03  4.545e-02  -0.122 0.902924
## LotShapeReg         -3.258e-03  7.380e-03  -0.442 0.658947
## GrLivArea            4.664e-02  4.277e-02   1.090 0.275783
## GarageCars           2.943e-02  7.635e-03   3.855 0.000124 ***
## TotalBsmtSF          5.659e-02  1.238e-02   4.570 5.51e-06 ***
## X1stFlrSF            5.689e-02  3.305e-02   1.721 0.085486 .
```

```
## GarageArea           1.472e-02  7.818e-03   1.883 0.060069 .
## BsmtFinSF1           1.230e-02  1.039e-02   1.184 0.236644
## OverallCond          5.032e-02  4.243e-03  11.859  < 2e-16 ***
## LotArea              2.075e-02  4.802e-03   4.322 1.71e-05 ***
## Age_Depuis_Reno     -1.180e-02  5.437e-03  -2.170 0.030251 *
## Exterior1stAsphShn   2.210e-02  1.395e-01   0.158 0.874150
## Exterior1stBrkFace   1.823e-01  6.011e-02   3.033 0.002486 **
## Exterior1stCemntBd   6.323e-02  9.537e-02   0.663 0.507487
## Exterior1stHdBoard   6.700e-02  6.130e-02   1.093 0.274685
## Exterior1stImStucc   8.128e-02  1.210e-01   0.672 0.502053
## Exterior1stMetalSd   8.749e-02  6.847e-02   1.278 0.201664
## Exterior1stPlywood   8.809e-02  6.054e-02   1.455 0.145954
## Exterior1stStone     3.196e-02  9.991e-02   0.320 0.749133
## Exterior1stStucco    1.977e-01  7.561e-02   2.615 0.009071 **
## Exterior1stVinylSd   5.223e-02  6.209e-02   0.841 0.400419
## Exterior1stWd Sdng   5.000e-02  5.929e-02   0.843 0.399324
## Exterior1stWdShing   1.112e-01  6.410e-02   1.734 0.083231 .
## HouseStyle1.5Unf     4.517e-03  3.423e-02   0.132 0.895066
## HouseStyle1Story     1.238e-02  1.803e-02   0.687 0.492433
## HouseStyle2.5Fin    -2.302e-02  6.106e-02  -0.377 0.706264
## HouseStyle2.5Unf     3.465e-02  4.470e-02   0.775 0.438493
## HouseStyle2Story    -1.709e-03  1.552e-02  -0.110 0.912341
## HouseStyleSFoyer     2.409e-02  2.692e-02   0.895 0.371009
## HouseStyleSLvl       5.049e-02  2.191e-02   2.304 0.021417 *
## BsmtExposureGd       3.049e-02  1.398e-02   2.181 0.029442 *
## BsmtExposureMn      -1.929e-02  1.417e-02  -1.361 0.173835
## BsmtExposureNo      -2.505e-02  1.036e-02  -2.419 0.015758 *
## Age_Maison          -6.420e-02  1.044e-02  -6.147 1.16e-09 ***
## BsmtUnfSF           -1.343e-02  1.018e-02  -1.319 0.187596
## X2ndFlrSF            8.024e-02  3.456e-02   2.322 0.020440 *
## Age_Garage          -2.930e-03  7.368e-03  -0.398 0.691003
## LotFrontage          3.410e-03  4.540e-03   0.751 0.452728
## HeatingQCFa         -2.857e-02  1.918e-02  -1.489 0.136745
## HeatingQCGd         -8.659e-03  9.910e-03  -0.874 0.382480
## HeatingQCPo         -2.471e-01  1.066e-01  -2.318 0.020639 *
## HeatingQCTA         -1.680e-02  9.619e-03  -1.746 0.081047 .
## Fireplaces           1.784e-02  4.054e-03   4.401 1.20e-05 ***
## OpenPorchSF          7.339e-03  3.598e-03   2.040 0.041665 *
## KitchenQualFa       -1.167e-01  2.890e-02  -4.039 5.81e-05 ***
## KitchenQualGd       -6.135e-02  1.637e-02  -3.748 0.000189 ***
## KitchenQualTA       -6.403e-02  1.843e-02  -3.475 0.000534 ***
## MoSold              -1.031e-03  3.157e-03  -0.327 0.744061
## TotRmsAbvGrd         2.566e-03  6.687e-03   0.384 0.701209
## WoodDeckSF           1.099e-02  3.482e-03   3.155 0.001657 **
## GarageTypeBasment   -3.974e-02  3.035e-02  -1.309 0.190717
## GarageTypeBuiltIn   -5.411e-03  1.598e-02  -0.339 0.735052
## GarageTypeCarPort   -6.610e-02  4.168e-02  -1.586 0.113120
## GarageTypeDetchd    -1.487e-02  9.971e-03  -1.491 0.136195
## BsmtQualFa          -8.154e-02  3.044e-02  -2.679 0.007517 **
## BsmtQualGd          -5.323e-02  1.590e-02  -3.348 0.000845 ***
## BsmtQualTA          -5.075e-02  1.962e-02  -2.587 0.009822 **
## EnclosedPorch        6.741e-03  3.589e-03   1.878 0.060663 .
## MasVnrArea           1.541e-03  5.042e-03   0.306 0.759879
## BsmtFinType1BLQ      7.272e-03  1.314e-02   0.553 0.580205
## BsmtFinType1GLQ     -2.790e-03  1.202e-02  -0.232 0.816449
## BsmtFinType1LwQ     -2.139e-02  1.761e-02  -1.215 0.224833
```

```
## BsmtFinType1Rec     -1.443e-02  1.463e-02  -0.987 0.324104
## BsmtFinType1Unf     -2.439e-02  1.375e-02  -1.774 0.076334 .
## FoundationCBlock     1.126e-02  1.449e-02   0.777 0.437394
## FoundationPConc      3.044e-02  1.629e-02   1.869 0.061978 .
## FoundationSlab       1.794e-02  3.610e-02   0.497 0.619438
## FoundationStone      3.741e-02  4.722e-02   0.792 0.428445
## BedroomAbvGr        -5.662e-03  4.982e-03  -1.137 0.256011
## FullBath             6.559e-03  5.666e-03   1.158 0.247288
## NeighborhoodBlueste -9.446e-02  1.049e-01  -0.900 0.368260
## NeighborhoodBrDale  -1.678e-01  4.841e-02  -3.467 0.000550 ***
## NeighborhoodBrkSide -1.061e-02  3.885e-02  -0.273 0.784909
## NeighborhoodClearCr -1.029e-02  4.000e-02  -0.257 0.797050
## NeighborhoodCollgCr -4.186e-02  3.135e-02  -1.335 0.182125
## NeighborhoodCrawfor  9.563e-02  3.781e-02   2.529 0.011587 *
## NeighborhoodEdwards -1.033e-01  3.526e-02  -2.931 0.003463 **
## NeighborhoodGilbert -4.229e-02  3.420e-02  -1.237 0.216539
## NeighborhoodIDOTRR  -9.214e-02  4.144e-02  -2.224 0.026403 *
## NeighborhoodMeadowV -2.323e-01  4.920e-02  -4.720 2.71e-06 ***
## NeighborhoodMitchel -9.423e-02  3.632e-02  -2.594 0.009625 **
## NeighborhoodNAmes   -7.852e-02  3.442e-02  -2.281 0.022773 *
## NeighborhoodNoRidge  1.238e-02  3.860e-02   0.321 0.748467
## NeighborhoodNPkVill -2.733e-02  6.720e-02  -0.407 0.684284
## NeighborhoodNridgHt  3.290e-02  3.372e-02   0.976 0.329461
## NeighborhoodNWAmes  -7.885e-02  3.531e-02  -2.233 0.025771 *
## NeighborhoodOldTown -1.191e-01  3.694e-02  -3.224 0.001309 **
## NeighborhoodSawyer  -7.278e-02  3.640e-02  -1.999 0.045876 *
## NeighborhoodSawyerW -4.355e-02  3.441e-02  -1.266 0.205921
## NeighborhoodSomerst  3.326e-02  3.330e-02   0.999 0.318261
## NeighborhoodStoneBr  1.198e-01  3.888e-02   3.082 0.002115 **
## NeighborhoodSWISU   -5.137e-02  4.690e-02  -1.095 0.273734
## NeighborhoodTimber  -5.500e-02  3.577e-02  -1.537 0.124506
## NeighborhoodVeenker -8.805e-03  4.691e-02  -0.188 0.851153
## YrSold               7.593e-05  3.268e-03   0.023 0.981468
## MSSubClass          -2.707e-02  4.834e-03  -5.599 2.82e-08 ***
## MasVnrTypeBrkFace    3.526e-02  3.333e-02   1.058 0.290429
## MasVnrTypeNone       3.353e-02  3.361e-02   0.998 0.318727
## MasVnrTypeStone      4.341e-02  3.497e-02   1.241 0.214741
## BsmtFullBath         1.255e-02  4.722e-03   2.658 0.007994 **
## ExterQualFa         -5.733e-02  4.649e-02  -1.233 0.217797
## ExterQualGd          9.420e-03  2.347e-02   0.401 0.688237
## ExterQualTA          1.446e-02  2.566e-02   0.563 0.573242
## HalfBath             1.206e-02  4.870e-03   2.476 0.013462 *
## Exterior2ndAsphShn   3.420e-02  9.141e-02   0.374 0.708425
## Exterior2ndBrk Cmn  -2.476e-02  9.809e-02  -0.252 0.800792
## Exterior2ndBrkFace  -2.681e-02  6.239e-02  -0.430 0.667532
## Exterior2ndCmentBd   2.568e-02  9.409e-02   0.273 0.785006
## Exterior2ndHdBoard  -7.286e-03  5.754e-02  -0.127 0.899268
## Exterior2ndImStucc  -1.948e-02  6.629e-02  -0.294 0.768988
## Exterior2ndMetalSd   5.449e-03  6.545e-02   0.083 0.933665
## Exterior2ndOther    -3.985e-02  1.182e-01  -0.337 0.736046
## Exterior2ndPlywood  -7.217e-03  5.585e-02  -0.129 0.897207
## Exterior2ndStone    -5.672e-02  7.770e-02  -0.730 0.465603
## Exterior2ndStucco   -1.261e-01  7.492e-02  -1.683 0.092671 .
## Exterior2ndVinylSd   3.670e-02  5.928e-02   0.619 0.535951
## Exterior2ndWd Sdng   2.578e-02  5.587e-02   0.461 0.644672
## Exterior2ndWd Shng  -5.186e-02  5.930e-02  -0.874 0.382068
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09661 on 953 degrees of freedom
## Multiple R-squared:  0.945,  Adjusted R-squared:  0.9381
## F-statistic: 136.4 on 120 and 953 DF,  p-value: < 2.2e-16
```

Le modèle de base obtenu a un coefficient de determination ajusté de 0.9381 ce qui semble correct pour une prédiction. Toutefois, il est nécessaire de valider les hypothèses d'une regression.

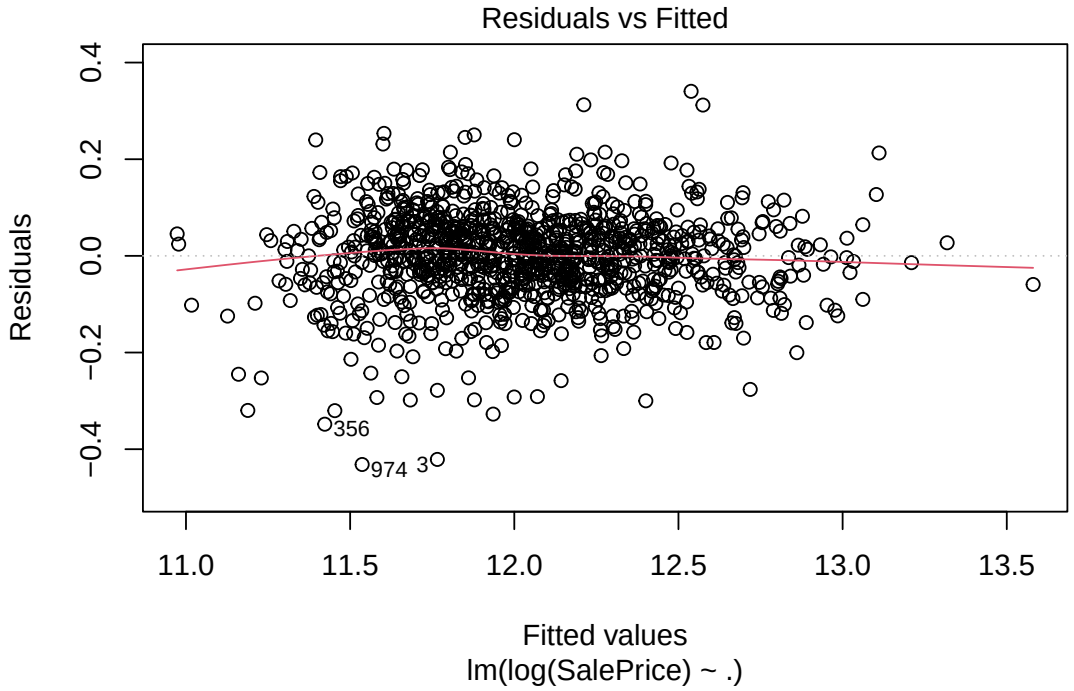
3.1.3. Validation du modèle de base : Analyse des résidus

Le modèle de base est validé s'il respecte 4 hypothèses :

P1 : Les erreurs sont centrées / le modèle est linéaire ; P2 : Les erreurs ont une variance homoscédastique ; P3 : Les erreurs sont non corrélées ; P4 : Les erreurs sont gaussiennes.

- Vérification de P1 :

```
plot(model2, 1)
```

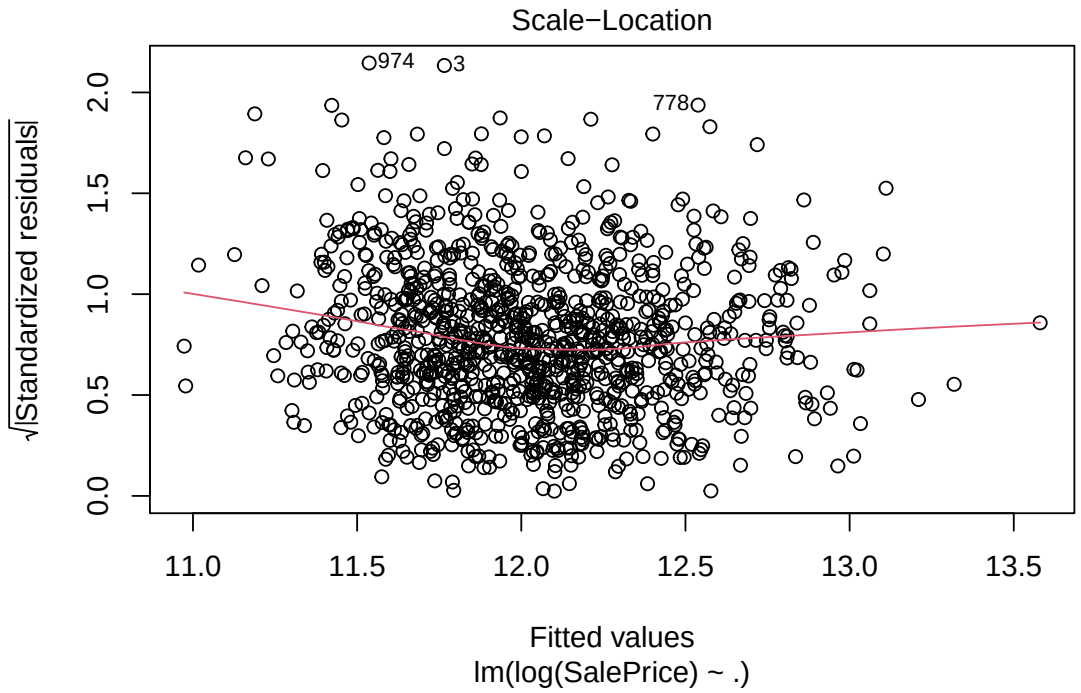


P1 est vérifiée si les résidus restent globalement uniformément répartis des deux côtés de 0. La ligne rouge est approximativement horizontale à zéro. Les résidus restent globalement uniformément répartis des deux côtés de 0. Le modèle vérifie donc P1.

- Vérification de P2 :

```
plot(model2, 3)
```

```
## Warning: not plotting observations with leverage one:
##   7, 407, 668, 833, 883
```



P2 est vérifiée si la courbe rouge est horizontale et les points sont uniformément répartis autour d'elle (Absence de tendance). Il est difficile de vérifier P2 à partir du graphique ci-dessus. Nous pouvons effectuer un test de Breush-Pagan pour vérifier l'homoscédasticité des résidus du modèle.

Hypothèse H0 du test : "Homoscédasticité"

```
ncvTest(model2)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 19.09688, Df = 1, p = 1.2425e-05
```

P-value = 1.2425e-05 < 0.05, donc P2 n'est pas vérifiée pour un niveau de confiance de 95%.

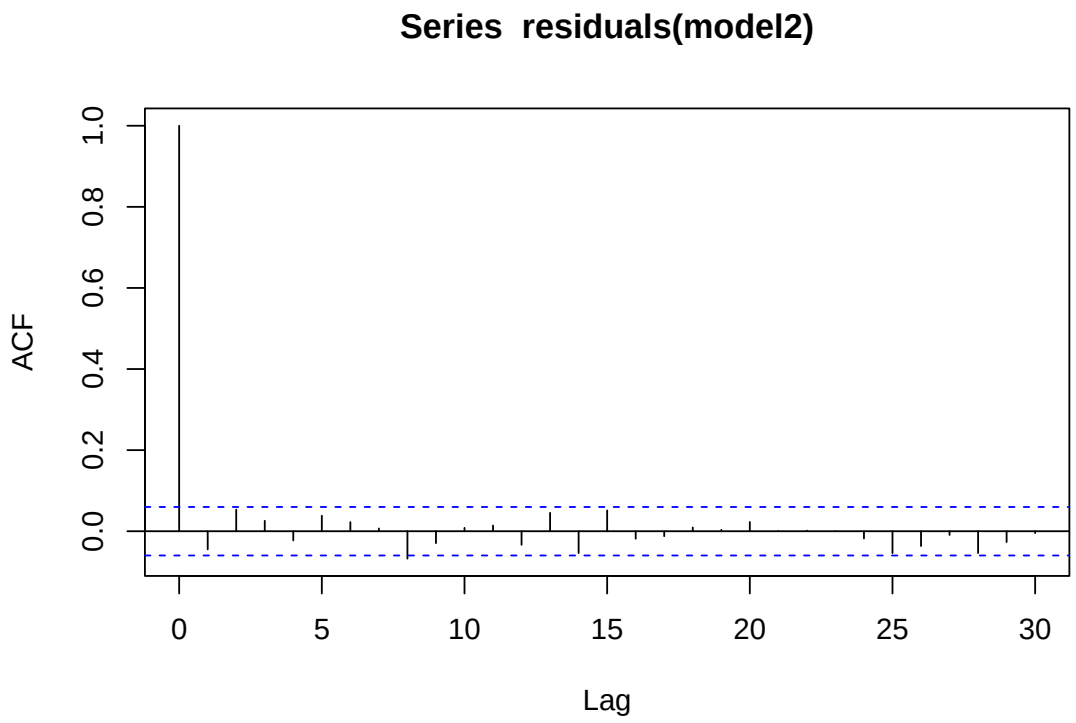
- Vérification de P3 :

```
acf(residuals(model2), mail = "Plot Auto-correlation")
```

```
## Warning in plot.window(...): "mail" n'est pas un paramètre graphique
```

```
## Warning in plot.xy(xy, type, ...): "mail" n'est pas un paramètre graphique
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "mail" n'est pas un
## paramètre graphique

## Warning in axis(side = side, at = at, labels = labels, ...): "mail" n'est pas un
## paramètre graphique

## Warning in box(...): "mail" n'est pas un paramètre graphique

## Warning in title(...): "mail" n'est pas un paramètre graphique
```



P3 est vérifiée si tous les traits verticaux, sauf le premier, ne dépasse pas les seuils en pointillés. P3 semble être verifier à partir du graphique ci-dessus. Nous pouvons effectuer un test de Durbin-Watson pour vérifier la non-corrélation des résidus du modèle.

Hypothèse H0 du test : "Non-corrélation"

```
durbinWatsonTest(model2)
```

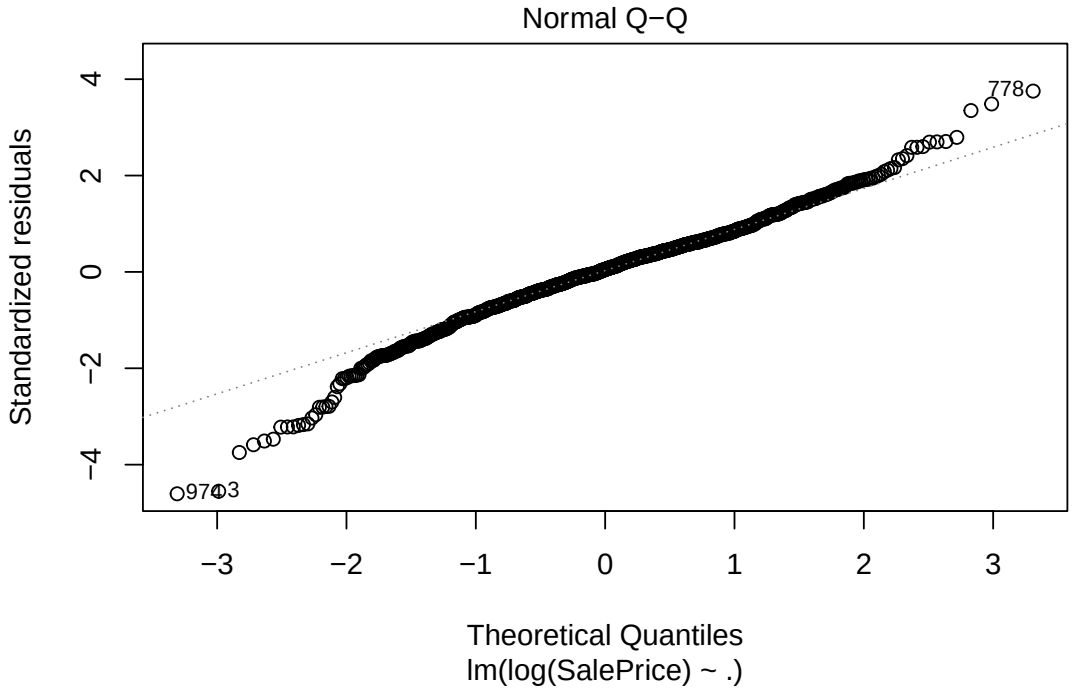
```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.04473948       2.08938   0.126
##  Alternative hypothesis: rho != 0
```

P-value = 0.126 > 0.05, donc P3 est vérifiée pour un niveau de confiance de 95%.

- Vérification de P4 :

```
plot(model2, 2)
```

```
## Warning: not plotting observations with leverage one:
##   7, 407, 668, 833, 883
```



P4 est vérifiée si les erreurs sont gaussiennes. Les points doivent être alignés autour de la bissectrice. P4 ne semple pas être vérifié à partir du graphique ci-dessus. Nous pouvons effectuer un test de Shapiro pour vérifier le caractère gaussien des résidus du modèle.

Hypothèse H0 du test : “Distribution gaussienne des résidus”

```
shapiro.test(residuals((model2)))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals((model2))
## W = 0.97576, p-value = 2.042e-12
```

P-value = 2.042e-12 < 0.05, donc P4 n’est pas vérifiée pour un niveau de confiance de 95%.

Les erreurs de notre modèle de base n’ont pas une variance homoscédastique et ne suivent pas une distribution gaussienne (P2 et P4 non validées). Notre modèle n’est donc pas validé pour une prédiction.

3.2. Implémentation de nouveaux modèles

- Validation croisée :

Dans la plupart des applications, nous sommes moins préoccupés par la performance d’un modèle sur les données utilisées pour l’entraîner, et plus préoccupés par la façon dont il se généralisera à un ensemble de données indépendant. En d’autres termes, nous ne voulons généralement pas de modèles trop spécifiques

à un échantillon particulier (sur-ajustement) car ces modèles ne nous aident pas réellement à expliquer le phénomène d'intérêt plus largement.

La validation croisée est un outil permettant d'estimer les performances d'un modèle sur un échantillon indépendant. En R, le package caret nous permet d'entrainer des modèles en considérant des étapes de validation croisée. Définissons donc un objet indiquant que nos modèles seront évalués à l'aide de 5 répétitions de validation croisée 10 fois.

```
custom <- trainControl(method = "repeatedcv", number = 10, repeats = 5, verboseIter = FALSE)
```

3.2.1. Régression pas à pas

La régression pas à pas (ou sélection pas à pas) consiste à ajouter et à supprimer de manière itérative des prédicteurs, dans le modèle prédictif, afin de trouver le sous-ensemble de variables dans l'ensemble de données résultant en le modèle le plus performant, c'est-à-dire un modèle qui réduit l'erreur de prédiction.

Il existe trois stratégies de régression pas à pas : ** La régression Forward, qui commence sans aucun prédicteur dans le modèle, ajoute de manière itérative les prédicteurs les plus contributifs et s'arrête lorsque l'amélioration n'est plus statistiquement significative ; ** La régression Backward (ou élimination en amont), qui commence avec tous les prédicteurs du modèle (modèle complet), supprime de manière itérative les prédicteurs les moins contributifs et s'arrête lorsque nous disposons d'un modèle dans lequel tous les prédicteurs sont statistiquement significatifs ; ** La régression Both (ou remplacement séquentiel), qui est une combinaison de sélections avant et arrière.

- Implémentons un modèle de regression Forward.

```
tuneGrid <- data.frame(nvmax = 1:42)
model_forward <- train(log(SalePrice) ~ ., data = train_final2, method = "leapForward", tuneGrid = t
                       trControl = custom, verbose = FALSE)
```

```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

```
model_forward$bestTune
```

```
##    nvmax
## 42    42
```

- Implémentons un modèle de regression Backward.

```
model_backward <- train(log(SalePrice) ~ ., data = train_final2, method = "leapBackward", tuneGrid =
                        trControl = custom, verbose = FALSE)
```

```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

```
model_backward$bestTune
```

```
##    nvmax
## 42    42
```

- Implémentons un modèle de regression Both.

```
model_both <- train(log(SalePrice) ~ ., data = train_final2, method = "leapSeq",
                    tuneGrid = tuneGrid, trControl = custom, verbose = FALSE)
```

```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

```
model_both$bestTune
```

```
##    nvmax
## 41    41
```

3.2.2. Regression par régularisation

L'algorithme de régression linéaire fonctionne en sélectionnant des coefficients pour chaque variable indépendante qui minimise une fonction de perte. Cependant, si les coefficients sont grands, ils peuvent conduire à un sur-ajustement sur l'ensemble de données d'apprentissage, un tel modèle ne se généralisera pas bien sur les données de test invisibles. Pour pallier cette lacune, nous pouvons faire de la régularisation, qui pénalise les gros coefficients. Il existe 3 familles de regression régularisé : Ridge, Lasso et Elastic Net.

Pour estimer les modèles dans R, nous pouvons utiliser le package caret qui nous permet de configurer une grille de valeurs de régularisation alpha et lambda et effectuer une validation croisée pour trouver les valeurs de paramètre optimales.

- Implémentons un modèle de regression Ridge :

```
ridge <- train(log(SalePrice) ~ ., train_final2, method = 'glmnet', tuneGrid = expand.grid(alpha = 0
               trControl = custom)
ridge$bestTune
```

```
##   alpha lambda
## 1     0  1e-04
```

- Implémentons un modèle de regression Lasso :

```
lasso <- train(log(SalePrice) ~ ., train_final2, method = 'glmnet',
               tuneGrid = expand.grid(alpha = 1, lambda = seq(0.0001, 1, length = 5)), trControl = c
lasso$bestTune
```

```
##   alpha lambda
## 1     1  1e-04
```

- Implémentons un modèle de regression Elastic Net :

```
en <- train(log(SalePrice) ~ ., train_final2, method = 'glmnet',
            tuneGrid = expand.grid(alpha = seq(0, 1, length = 10), lambda = seq(0.0001, 1, length =
en$bestTune
```

```
##    alpha lambda
## 46     1  1e-04
```

4. Modèle finale

4.1. Comparaison des modèles

```
model_list <- list(ModelForward = model_forward, ModelBackward = model_backward,
                   ModelBoth = model_both, Ridge = ridge, Lasso = lasso, ElasticNet = en)
res <- resamples(model_list)
```

```
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: ModelForward, ModelBackward, ModelBoth, Ridge, Lasso, ElasticNet
## Number of resamples: 50
##
## MAE
```

```
##                      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## ModelForward  0.07125300 0.08011954 0.08385703 0.08367991 0.08729494 0.10098964
## ModelBackward 0.06779376 0.08086604 0.08602690 0.08487662 0.08968602 0.10419325
## ModelBoth     0.07182329 0.07959480 0.08461768 0.08460173 0.09031161 0.09843638
## Ridge         0.06453290 0.07309898 0.07687480 0.07721687 0.08087697 0.09212950
## Lasso         0.06267790 0.07345791 0.07647102 0.07783978 0.08315249 0.09296316
## ElasticNet    0.06818545 0.07514625 0.07812678 0.07777853 0.08057577 0.08713615
##               NA's
## ModelForward     0
## ModelBackward    0
## ModelBoth        0
## Ridge            0
## Lasso            0
## ElasticNet       0
##
## RMSE
##                      Min.    1st Qu.    Median      Mean   3rd Qu.      Max.
## ModelForward  0.09039889 0.10547273 0.1130520 0.1118816 0.1177936 0.1329221
## ModelBackward 0.09176152 0.10510791 0.1159319 0.1137910 0.1202662 0.1449822
## ModelBoth     0.09432499 0.10384113 0.1122896 0.1126915 0.1202490 0.1340842
## Ridge         0.08444176 0.09708814 0.1037998 0.1030749 0.1084914 0.1200295
## Lasso         0.07810861 0.09682922 0.1046273 0.1036439 0.1093173 0.1277068
## ElasticNet    0.08502160 0.09721223 0.1039503 0.1034606 0.1093082 0.1242480
##               NA's
## ModelForward     0
## ModelBackward    0
## ModelBoth        0
## Ridge            0
## Lasso            0
## ElasticNet       0
##
## Rsquared
##                       Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## ModelForward  0.8909257 0.9014837 0.9224370 0.9178326 0.9286861 0.9444740    0
## ModelBackward 0.8796984 0.9044269 0.9166732 0.9144502 0.9263445 0.9447320    0
## ModelBoth     0.8661398 0.9049552 0.9184778 0.9158250 0.9277643 0.9422606    0
## Ridge         0.9076841 0.9198191 0.9305738 0.9302162 0.9383251 0.9491562    0
## Lasso         0.8908356 0.9184471 0.9290289 0.9289724 0.9413144 0.9632488    0
## ElasticNet    0.8985490 0.9216188 0.9285088 0.9296925 0.9383586 0.9555550    0
```

4.1. Selection du modèle final

Les différents modèles ayant des statistiques très proches, nous choisissons comme meilleur modèle, celui donc le RMSE minium est le plus bas. Il s'agit du modèle Lasso. Nous pouvons résumer les coefficients estimés comme suit :

```
model_final <- lasso
coef(model_final$finalModel, s = model_final$bestTune$lambda)
```

```
## 123 x 1 sparse Matrix of class "dgCMatrix"
##                                 s1
## (Intercept)          1.210200e+01
## OverallQual          6.777944e-02
## LotShapeIR2          7.360180e-03
```

```
## LotShapeIR3         -3.483037e-03
## LotShapeReg         -1.910264e-03
## GrLivArea            9.583750e-02
## GarageCars           2.992501e-02
## TotalBsmtSF          5.438278e-02
## X1stFlrSF            1.961717e-02
## GarageArea           1.542497e-02
## BsmtFinSF1           1.463849e-02
## OverallCond          5.043743e-02
## LotArea              1.979355e-02
## Age_Depuis_Reno     -1.234105e-02
## Exterior1stAsphShn  -1.658301e-02
## Exterior1stBrkFace   1.297077e-01
## Exterior1stCemntBd   1.260149e-02
## Exterior1stHdBoard   1.467193e-02
## Exterior1stImStucc   1.746972e-02
## Exterior1stMetalSd   3.700724e-02
## Exterior1stPlywood   3.416024e-02
## Exterior1stStone    -1.197295e-02
## Exterior1stStucco    1.269925e-01
## Exterior1stVinylSd   1.680103e-04
## Exterior1stWd Sdng   .
## Exterior1stWdShing   5.417677e-02
## HouseStyle1.5Unf     6.133982e-03
## HouseStyle1Story     1.458758e-02
## HouseStyle2.5Fin    -4.238503e-02
## HouseStyle2.5Unf     2.522869e-02
## HouseStyle2Story     .
## HouseStyleSFoyer     2.149217e-02
## HouseStyleSLvl       4.710569e-02
## BsmtExposureGd       3.089034e-02
## BsmtExposureMn      -1.854375e-02
## BsmtExposureNo      -2.581702e-02
## Age_Maison          -6.584337e-02
## BsmtUnfSF           -1.107545e-02
## X2ndFlrSF            3.862827e-02
## Age_Garage          -2.146528e-03
## LotFrontage          3.102544e-03
## HeatingQCFa         -3.065357e-02
## HeatingQCGd         -8.873749e-03
## HeatingQCPo         -2.365165e-01
## HeatingQCTA         -1.664581e-02
## Fireplaces           1.885045e-02
## OpenPorchSF          7.351604e-03
## KitchenQualFa       -1.056353e-01
## KitchenQualGd       -5.606275e-02
## KitchenQualTA       -5.818880e-02
## MoSold              -6.150478e-04
## TotRmsAbvGrd         2.593968e-03
## WoodDeckSF           1.141083e-02
## GarageTypeAttchd     6.595564e-03
## GarageTypeBasment   -3.445999e-02
## GarageTypeBuiltIn    .
## GarageTypeCarPort   -5.752770e-02
## GarageTypeDetchd    -8.008497e-03
## BsmtQualFa          -7.900357e-02
```

```
## BsmtQualGd          -4.953816e-02
## BsmtQualTA          -4.709086e-02
## EnclosedPorch        6.043697e-03
## MasVnrArea           .
## BsmtFinType1BLQ      6.779279e-03
## BsmtFinType1GLQ     -3.634863e-05
## BsmtFinType1LwQ     -1.968736e-02
## BsmtFinType1Rec     -1.380320e-02
## BsmtFinType1Unf     -2.296956e-02
## FoundationCBlock     6.490842e-03
## FoundationPConc      2.394332e-02
## FoundationSlab       7.877343e-03
## FoundationStone      2.997930e-02
## FoundationWood       .
## BedroomAbvGr        -5.763928e-03
## FullBath             6.600410e-03
## NeighborhoodBlueste -4.956578e-02
## NeighborhoodBrDale  -1.273448e-01
## NeighborhoodBrkSide  2.513498e-02
## NeighborhoodClearCr  2.920663e-02
## NeighborhoodCollgCr -3.890204e-03
## NeighborhoodCrawfor  1.355523e-01
## NeighborhoodEdwards -5.874601e-02
## NeighborhoodGilbert -1.943212e-03
## NeighborhoodIDOTRR  -5.098839e-02
## NeighborhoodMeadowV -1.889658e-01
## NeighborhoodMitchel -5.073768e-02
## NeighborhoodNAmes   -3.539289e-02
## NeighborhoodNoRidge  5.140617e-02
## NeighborhoodNPkVill  1.391970e-03
## NeighborhoodNridgHt  6.798461e-02
## NeighborhoodNWAmes  -3.516592e-02
## NeighborhoodOldTown -7.863299e-02
## NeighborhoodSawyer  -2.943439e-02
## NeighborhoodSawyerW -3.108736e-03
## NeighborhoodSomerst  6.845530e-02
## NeighborhoodStoneBr  1.545941e-01
## NeighborhoodSWISU   -1.728262e-02
## NeighborhoodTimber  -1.353934e-02
## NeighborhoodVeenker  2.850297e-02
## YrSold               .
## MSSubClass          -2.524019e-02
## MasVnrTypeBrkFace    4.473568e-03
## MasVnrTypeNone       4.836274e-04
## MasVnrTypeStone      9.959244e-03
## BsmtFullBath         1.183985e-02
## ExterQualFa         -6.759169e-02
## ExterQualGd          .
## ExterQualTA          2.641553e-03
## HalfBath             1.175087e-02
## Exterior2ndAsphShn   3.673553e-02
## Exterior2ndBrk Cmn   .
## Exterior2ndBrkFace  -1.154738e-02
## Exterior2ndCmentBd   3.482688e-02
## Exterior2ndHdBoard   3.823823e-03
## Exterior2ndImStucc  -4.259796e-03
```

```
## Exterior2ndMetalSd   1.465532e-02
## Exterior2ndOther    -2.323658e-02
## Exterior2ndPlywood   5.439795e-03
## Exterior2ndStone    -3.461747e-02
## Exterior2ndStucco   -9.567650e-02
## Exterior2ndVinylSd   4.978796e-02
## Exterior2ndWd Sdng   3.580019e-02
## Exterior2ndWd Shng  -3.320054e-02
```

5. Calcul du RMSE

Pour pouvoir prédire sur les données de test, nous devons remplacer les modalités absentes des données de test par les modalités les plus fréquentes des variables qualitatives.

```
exterior1st_mode <- names(which.max(table(train_final2$Exterior1st)))
exterior2nd_mode <- names(which.max(table(train_final2$Exterior2nd)))
garageType_mode <- names(which.max(table(train_final2$GarageType)))
foundation_mode <- names(which.max(table(train_final2$Foundation)))
lotShape_mode <- names(which.max(table(train_final2$LotShape)))

test_final2$Exterior1st[test_final2$Exterior1st == "BrkComm"] <- exterior1st_mode
test_final2$Exterior1st[test_final2$Exterior1st == "CBlock"] <- exterior1st_mode
test_final2$Exterior2nd[test_final2$Exterior2nd == "CBlock"] <- exterior2nd_mode
test_final2$GarageType[test_final2$GarageType == "2Types"] <- garageType_mode
test_final2$Foundation[test_final2$Foundation == "Wood"] <- foundation_mode
test_final2$LotShape[test_final2$LotShape == "IR3"] <- lotShape_mode
```

Calculons les valeurs du RMSE sur le train et le test avec l'unité de base du SalePrice:

- Prediction et calcul du RMSE sur les données de train

```
RMSE_train <- rmse(train_final2$SalePrice, exp(predict(model_final, train_final2)))
RMSE_train
```

```
## [1] 17166.93
```

- Prediction et calcul du RMSE sur les données de test

```
RMSE_test <- rmse(test_final2$SalePrice, exp(predict(model_final, test_final2)))
RMSE_test
```

```
## [1] 21534.28
```

.