SIM Project 1. Ames Housing dataset

Adrià Casanova, Víctor Garcia, Zhengyong Ji

November, 19th 2023

Table of Contents

In this work, we will study the data set called "Ames Housing dataset", collected by Dean De Cock for the purpose to analyze the correlation about house prices and different features that describe the house condition, and then to build a regression model that will allow us to predict the sale price.

All members have contributed equally to all parts of the project.

The data set has two parts, the training part and testing part, with 1460 and 1459 observations each other, and 81 variables (including the id variable).

```
# Delete any existing object
if(!is.null(dev.list())) dev.off()
rm(list = ls())

library(car)
library(mice)
library(dplyr)
```

```r
library(missMDA)
library(FactoMineR)
library(chemometrics)
library(DataExplorer)
library(corrplot)
library(DataExplorer)
library(MASS)
library(effects)

train = read.csv("train.csv")
test = read.csv("test.csv")

#Create EDA report before doing any data preparation
create_report(train, output_format = "pdf_document", output_file = "tra
in.pdf")
create_report(test, output_format = "pdf_document", output_file = "test
.pdf")
```

0. Data preparation and data cleaning

After loading the datasets we defined the types of the variables (categorical, numerical or dates). Some of them required further transformation, based on some assumptions, that are detailed below.

```r
Categorical_val = c("MSSubClass","MSZoning","Street","Alley","LotShape"
,"LandContour","Utilities","LotConfig","LandSlope","Neighborhood","Cond
ition1","Condition2","BldgType","HouseStyle","OverallQual","OverallCond
","RoofStyle","RoofMatl","Exterior1st","Exterior2nd","MasVnrType","Exte
rQual","ExterCond","Foundation","BsmtQual","BsmtCond","BsmtExposure","B
smtFinType1","BsmtFinType2","Heating","HeatingQC","CentralAir","Electri
cal","KitchenQual","Functional","FireplaceQu","GarageType","GarageFinis
h","GarageQual","GarageCond","PavedDrive","PoolQC","Fence","MiscFeature
","SaleType","SaleCondition", "MoSold")

Numerical_val = c("LotFrontage","LotArea","MasVnrArea","BsmtFinSF1","Bs
mtFinSF2","BsmtUnfSF","TotalBsmtSF","X1stFlrSF","X2ndFlrSF","GrLivArea"
,"BsmtFullBath","BsmtHalfBath","FullBath","HalfBath","BedroomAbvGr","Ki
tchenAbvGr","TotRmsAbvGrd","Fireplaces","GarageCars","GarageArea","Wood
DeckSF","OpenPorchSF","EnclosedPorch","X3SsnPorch","ScreenPorch","MiscV
al","YearBuilt","YearRemodAdd","GarageYrBlt","YrSold")

Date_val = c("YearBuilt","YearRemodAdd","GarageYrBlt","MoSold","YrSold"
)

# Identify variables susceptible to be transformed into categorical
sapply(dplyr::select(train, Numerical_val), table)
sapply(dplyr::select(train, Categorical_val), table)
sapply(dplyr::select(train, Date_val), table)
```

1) Non applicable NaN's: There were 3 variables with an important number of missing (aprox 90%) because the measure was not applicable. This happened, firstly, in PoolArea because the pool area can not be computed for houses without a pool. It was also the case of LowQualFinSF because it is only refered to surfaces finished with low quality, and with BsmtFinSF2, that is only applicable for basement of type 2. Our solution was to define those three variables as binary variables.

```
# As we can see there are an important number of Nan
# PoolArea: 99% missings
length(which(train$PoolArea > 0))/dim(train)[1]*100
## [1] 0.4794521

length(which(test$PoolArea > 0))/dim(test)[1]*100
## [1] 0.4112406

# LowQualFinSF: 98% missings
length(which(train$LowQualFinSF > 0))/dim(train)[1]*100
## [1] 1.780822

length(which(test$LowQualFinSF > 0))/dim(test)[1]*100
## [1] 0.9595613

#BsmtFinSF2: 89% missings
length(which(train$BsmtFinSF2 > 0))/dim(train)[1]*100
## [1] 11.43836

length(which(test$BsmtFinSF2 > 0))/dim(test)[1]*100
## [1] 12.33722

# Under the assumption 1, we transform the variables to binary
test <- test %>% mutate(PoolArea = ifelse(PoolArea > 0, "Yes", "No"))
test$PoolArea = as.factor(test$PoolArea)
train <- train %>% mutate(PoolArea = ifelse(PoolArea > 0, "Yes", "No"))
train$PoolArea = as.factor(train$PoolArea)

test <- test %>% mutate(LowQualFinSF = ifelse(LowQualFinSF > 0, "Yes",
"No"))
test$LowQualFinSF = as.factor(test$LowQualFinSF)
train <- train %>% mutate(LowQualFinSF = ifelse(LowQualFinSF > 0, "Yes"
, "No"))
train$LowQualFinSF = as.factor(train$LowQualFinSF)

test <- test %>%mutate(BsmtFinSF2 = ifelse(BsmtFinSF2 > 0, "Yes","No"))
test$BsmtFinSF2 = as.factor(test$BsmtFinSF2)
train <- train %>% mutate(BsmtFinSF2 = ifelse(BsmtFinSF2 > 0, "Yes", "N
o"))
train$BsmtFinSF2 = as.factor(train$BsmtFinSF2)
```

2) LotFrontage, which represents the distance from the property to the street, has a high percentage of missing values, 18% in "train" and 16% in "test". A

quick look at the summary in both datasets shows there is not any house with a value of 0 for this variable. However, in the real world there exist houses whose entrance is right next to the street, with no separation from it. Hence, we deduce that missing values correspond to a distance of 0 and we impute LotFrontage like so.

```
#Analysis of the percentage of missings
percent_miss <- function(data) {
  return (length(which(is.na(data)))/length(data)*100)}
percent_miss(train$LotFrontage)
## [1] 17.73973

percent_miss(test$LotFrontage)
## [1] 15.5586

# Transformation Na'n to 0
lltrain <- which(is.na(train$LotFrontage))
lltest <- which(is.na(test$LotFrontage))
train$LotFrontage[lltrain] <- 0
test$LotFrontage[lltest] <- 0
```

3) Only few values possible: Variables BsmtHalfBath KitchenAbvGr have only 3 and 4 values possible, so we transform them into categorical

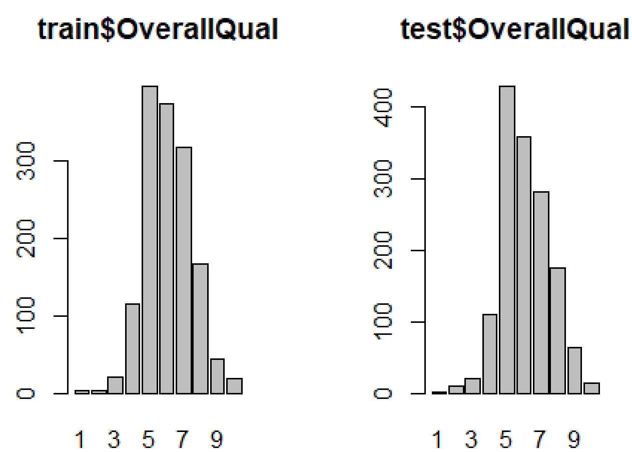
```
# BsmtHalfBath is numerical but it can only be 0, 1 or 2
length(which(train$BsmtHalfBath > 0))/dim(train)[1]*100
## [1] 5.616438

length(which(test$BsmtHalfBath > 0))/dim(test)[1]*100
## [1] 6.374229

#KitchenAbvGr can only be 0, 1, 2 or 3
length(which(train$KitchenAbvGr != 1))/dim(train)[1]*100
## [1] 4.657534

length(which(test$KitchenAbvGr != 1))/dim(test)[1]*100
## [1] 4.523646

#Transformation into categorical
train$BsmtHalfBath <- as.factor(train$BsmtHalfBath)
test$BsmtHalfBath <- as.factor(test$BsmtHalfBath)
train$KitchenAbvGr <- as.factor(train$KitchenAbvGr)
test$KitchenAbvGr <- as.factor(test$KitchenAbvGr)
levels(test$KitchenAbvGr) = c(levels(test$KitchenAbvGr),"3")
```

4) Variables with too many categories: OverallQual, Neighborhood and MSSubClass have too many levels to study their interactions in the models we will create later. Hence, we aggregate their categories following logical criterias. Even though, these will create a bias in the model, it will allow us to study their effect on the target. That being said, OverallQual will have 5 ordered levels.

```
t.train <- table(train$OverallQual); t.train
##
##   1   2   3   4   5   6   7   8   9  10
##   2   3  20 116 397 374 319 168  43  18
```

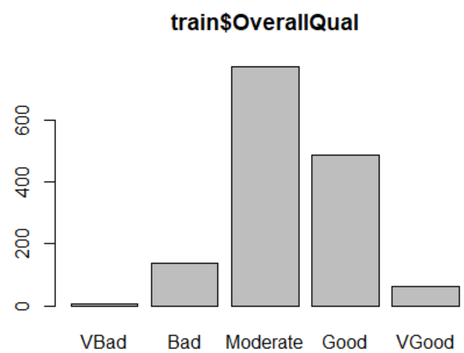
```
t.test <- table(test$OverallQual); t.test
##
##   1   2   3   4   5   6   7   8   9  10
##   2  10  20 110 428 357 281 174  64  13
```

```
par(mfrow=c(1,2))
barplot(t.train, main = "train$OverallQual")
barplot(t.test, main = "test$OverallQual")
```

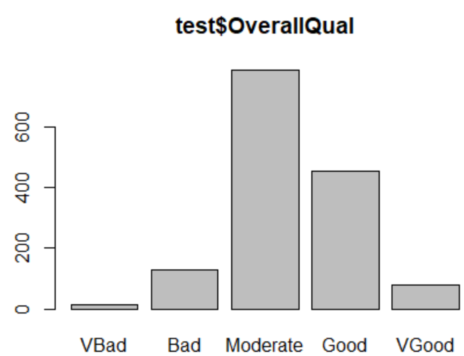



```
par(mfrow=c(1,1))

train$OverallQual <- replace(train$OverallQual, train$OverallQual %in%
1:2, "VBad")
train$OverallQual <- replace(train$OverallQual, train$OverallQual %in%
3:4, "Bad")
train$OverallQual <- replace(train$OverallQual, train$OverallQual %in%
5:6, "Moderate")
train$OverallQual <- replace(train$OverallQual, train$OverallQual %in%
7:8, "Good")
train$OverallQual <- replace(train$OverallQual, train$OverallQual %in%
9:10, "VGood")

test$OverallQual <- replace(test$OverallQual, test$OverallQual %in% 1:2
, "VBad")
test$OverallQual <- replace(test$OverallQual, test$OverallQual %in% 3:4
, "Bad")
test$OverallQual <- replace(test$OverallQual, test$OverallQual %in% 5:6
, "Moderate")
test$OverallQual <- replace(test$OverallQual, test$OverallQual %in% 7:8
, "Good")
```

```r
test$OverallQual <- replace(test$OverallQual, test$OverallQual %in% 9:1
0, "VGood")

train$OverallQual <- factor(train$OverallQual, levels = c("VBad", "Bad"
, "Moderate", "Good", "VGood"))
test$OverallQual <- factor(test$OverallQual, levels = c("VBad", "Bad","
Moderate", "Good","VGood"))

t.train2 <- table(train$OverallQual); t.train2
##
##     VBad      Bad Moderate     Good    VGood
##        5      136      771      487       61

t.test2 <- table(test$OverallQual); t.test2
##
##     VBad      Bad Moderate     Good    VGood
##       12      130      785      455       77

barplot(t.train2, main = "train$OverallQual")
```

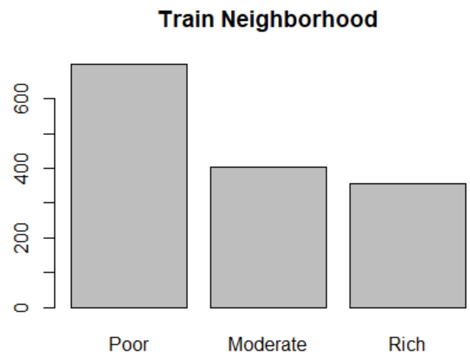



```r
barplot(t.test2, main = "test$OverallQual")
```

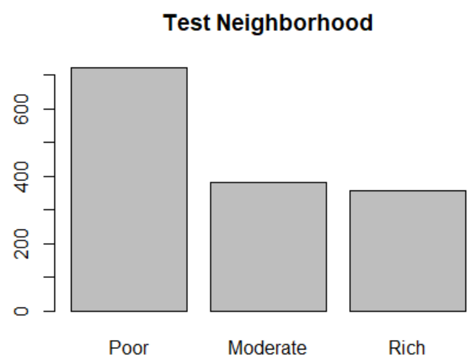


Neighborhood will have 3 ordered levels ("Poor", "Moderate" or "Rich") following the real-estate order found in https://www.neighborhoodscout.com/ia/ames/real-estate.

```
t.train <- table(train$Neighborhood)

t.test <- table(test$Neighborhood)

Rich = c("NoRidge", "NridgHt", "StoneBr", "Timber", "Veenker", "Somerst
", "ClearCr", "Crawfor")
Moderate = c("SWISU", "CollgCr", "Blueste", "Blmngtn", "Gilbert", "Mitc
hel", "NWAmes", "NPkVill")
Poor = c("Edwards", "BrDale", "BrkSide", "IDOTRR", "MeadowV", "NAmes",
"OldTown", "Sawyer", "SawyerW")

train$Neighborhood <- replace(train$Neighborhood, train$Neighborhood %i
n% Poor, "Poor")
train$Neighborhood <- replace(train$Neighborhood, train$Neighborhood %i
n% Moderate, "Moderate")
train$Neighborhood <- replace(train$Neighborhood, train$Neighborhood %i
n% Rich, "Rich")

test$Neighborhood <- replace(test$Neighborhood, test$Neighborhood %in%
Poor, "Poor")
test$Neighborhood <- replace(test$Neighborhood, test$Neighborhood %in%
Moderate, "Moderate")
test$Neighborhood <- replace(test$Neighborhood, test$Neighborhood %in%
Rich, "Rich")

train$Neighborhood <- factor(train$Neighborhood, levels = c("Poor", "Mo
derate", "Rich"))
test$Neighborhood <- factor(test$Neighborhood, levels = c("Poor", "Mode
rate", "Rich"))

t.train2 <- table(train$Neighborhood); t.train2

##
##     Poor Moderate     Rich
##      699      404      357

t.test2 <- table(test$Neighborhood); t.test2

##
##     Poor Moderate     Rich
##      721      382      356

barplot(t.train2, main = "Train Neighborhood")
```




```
barplot(t.test2, main = "Test Neighborhood")
```



4) Non applicable 0's: There are three variables that represent the area of different types of porches (EnclosedPorch, X3SsnPorch and ScreenPorch). In all of them, there is an important percentatge of 0's (about 90%). As a consequence, we consider that it is more efficient to treat those variables as binary to have a more balanced variable and because the univariate analysis of those variables, like outlier detection, of those variables would be very complicated, as their IQR was 0.

```
# Calculation of the % of non 0's
length(which(train$EnclosedPorch > 0))/dim(train)[1]*100

## [1] 14.24658

length(which(test$EnclosedPorch > 0))/dim(test)[1]*100

## [1] 17.20356

length(which(train$X3SsnPorch > 0))/dim(train)[1]*100

## [1] 1.643836

length(which(test$X3SsnPorch > 0))/dim(test)[1]*100
```

```
## [1] 0.8910212

length(which(train$ScreenPorch > 0))/dim(train)[1]*100

## [1] 7.945205

length(which(test$ScreenPorch > 0))/dim(test)[1]*100

## [1] 9.595613

#Transformation of the variables into binary
test <- test %>%mutate(EnclosedPorch = ifelse(EnclosedPorch > 0, "Yes",
 "No"))
test$EnclosedPorch = as.factor(test$EnclosedPorch)
train <- train %>%mutate(EnclosedPorch = ifelse(EnclosedPorch > 0, "Yes
", "No"))
train$EnclosedPorch = as.factor(train$EnclosedPorch)

test <- test %>%mutate(X3SsnPorch = ifelse(X3SsnPorch > 0, "Yes","No"))
test$X3SsnPorch = as.factor(test$X3SsnPorch)
train <- train %>%mutate(X3SsnPorch = ifelse(X3SsnPorch > 0, "Yes", "No
"))
train$X3SsnPorch = as.factor(train$X3SsnPorch)

test <- test %>%mutate(ScreenPorch = ifelse(ScreenPorch > 0, "Yes", "No
"))
test$ScreenPorch = as.factor(test$ScreenPorch)
train <- train %>%mutate(ScreenPorch = ifelse(ScreenPorch > 0, "Yes", "
No"))
train$ScreenPorch = as.factor(train$ScreenPorch)
```

5) Redundant variable: MiscVal, that measures the price of a miscellaneous feature (like having an elevator) has a lot of 0's (96%) as it is only applicable for some properties. Moreover, the information of the properties that have a miscellaneous feature can be also optained in "MiscFeature" variable. Consequently, we decided to remove this variable from the analysis.

```
# Analysis of non 0's
length(which(train$MiscVal > 0))/dim(train)[1]*100

## [1] 3.561644

length(which(test$MiscVal > 0))/dim(test)[1]*100

## [1] 3.495545

miscVal_train <- train$MiscVal
miscVal_test <- test$MiscVal
train$MiscVal <- NULL
test$MiscVal <- NULL
```

6) Creation of a new level for categorical: Because we do not know if all the Nan's in categorical variables are at random we decided that we will not impute any categorical. Consequently, we created a new level for all the missings.

```
# Declaration of a categorical as factor variables with a new level, "N
an"
levels(train$Alley) <- c(levels(train$Alley), "NAlley")
train$Alley[which(is.na(train$Alley))] <- "NAlley"
levels(test$Alley) <- c(levels(test$Alley), "NAlley")
test$Alley[which(is.na(test$Alley))] <- "NAlley"

levels(train$BsmtQual) <- c(levels(train$BsmtQual), "NBsmt")
train$BsmtQual[which(is.na(train$BsmtQual))] <- "NBsmt"
levels(test$BsmtQual) <- c(levels(test$BsmtQual), "NBsmt")
test$BsmtQual[which(is.na(test$BsmtQual))] <- "NBsmt"

levels(train$BsmtCond) <- c(levels(train$BsmtCond), "NBsmt")
train$BsmtCond[which(is.na(train$BsmtCond))] <- "NBsmt"
levels(test$BsmtCond) <- c(levels(test$BsmtCond), "NBsmt")
test$BsmtCond[which(is.na(test$BsmtCond))] <- "NBsmt"

levels(train$BsmtExposure) <- c(levels(train$BsmtExposure), "NBsmt")
train$BsmtExposure[which(is.na(train$BsmtExposure))] <- "NBsmt"
levels(test$BsmtExposure) <- c(levels(test$BsmtExposure), "NBsmt")
test$BsmtExposure[which(is.na(test$BsmtExposure))] <- "NBsmt"

levels(train$BsmtFinType1) <- c(levels(train$BsmtFinType1), "NBsmt")
train$BsmtFinType1[which(is.na(train$BsmtFinType1))] <- "NBsmt"
levels(test$BsmtFinType1) <- c(levels(test$BsmtFinType1), "NBsmt")
test$BsmtFinType1[which(is.na(test$BsmtFinType1))] <- "NBsmt"

levels(train$BsmtFinType2) <- c(levels(train$BsmtFinType2), "NBsmt")
train$BsmtFinType2[which(is.na(train$BsmtFinType2))] <- "NBsmt"
levels(test$BsmtFinType2) <- c(levels(test$BsmtFinType2), "NBsmt")
test$BsmtFinType2[which(is.na(test$BsmtFinType2))] <- "NBsmt"

levels(train$FireplaceQu) <- c(levels(train$FireplaceQu), "NFp")
train$FireplaceQu[which(is.na(train$FireplaceQu))] <- "NFp"
levels(test$FireplaceQu) <- c(levels(test$FireplaceQu), "NFp")
test$FireplaceQu[which(is.na(test$FireplaceQu))] <- "NFp"

levels(train$GarageType) <- c(levels(train$GarageType), "NGar")
train$GarageType[which(is.na(train$GarageType))] <- "NGar"
levels(test$GarageType) <- c(levels(test$GarageType), "NGar")
test$GarageType[which(is.na(test$GarageType))] <- "NGar"

levels(train$GarageFinish) <- c(levels(train$GarageFinish), "NGar")
train$GarageFinish[which(is.na(train$GarageFinish))] <- "NGar"
levels(test$GarageFinish) <- c(levels(test$GarageFinish), "NGar")
```

```
test$GarageFinish[which(is.na(test$GarageFinish))] <- "NGar"

levels(train$GarageQual) <- c(levels(train$GarageQual), "NGar")
train$GarageQual[which(is.na(train$GarageQual))] <- "NGar"
levels(test$GarageQual) <- c(levels(test$GarageQual), "NGar")
test$GarageQual[which(is.na(test$GarageQual))] <- "NGar"

levels(train$GarageCond) <- c(levels(train$GarageCond), "NGar")
train$GarageCond[which(is.na(train$GarageCond))] <- "NGar"
levels(test$GarageCond) <- c(levels(test$GarageCond), "NGar")
test$GarageCond[which(is.na(test$GarageCond))] <- "NGar"

levels(train$PoolQC) <- c(levels(train$PoolQC), "NPool")
train$PoolQC[which(is.na(train$PoolQC))] <- "NPool"
levels(test) <- c(levels(test$PoolQC), "NPool")
test$PoolQC[which(is.na(test$PoolQC))] <- "NPool"

levels(train$Fence) <- c(levels(train$Fence), "NFen")
train$Fence[which(is.na(train$Fence))] <- "NFen"
levels(test$Fence) <- c(levels(test$Fence), "NFen")
test$Fence[which(is.na(test$Fence))] <- "NFen"

levels(train$MiscFeature) <- c(levels(train$MiscFeature), "N")
train$MiscFeature[which(is.na(train$MiscFeature))] <- "N"
levels(test$MiscFeature) <- c(levels(test$MiscFeature), "N")
test$MiscFeature[which(is.na(test$MiscFeature))] <- "N"
```

7) Missing in KitchenQual: there is a single missing value in test$KitchenQual, so we impute it with the mode of the variable, TA.

```
test$KitchenQual <- replace(test$KitchenQual, is.na(test$KitchenQual),
"TA")
```

8) Transformations into categorical: In some variables, like Month, we decided to transform them into categorical as only some values are possible

```
# Transformation of other variables into categorical
test <- test %>% mutate_if(is.character, as.factor)
train <- train %>% mutate_if(is.character, as.factor)

test$MSSubClass = as.factor(test$MSSubClass)
test$OverallQual = as.factor(test$OverallQual)
test$OverallCond = as.factor(test$OverallCond)

train$MSSubClass = as.factor(train$MSSubClass)
train$OverallQual = as.factor(train$OverallQual)
train$OverallCond = as.factor(train$OverallCond)

test$MoSold = month.name[test$MoSold]
test$MoSold = as.factor(test$MoSold)
```

```
train$MoSold = month.name[train$MoSold]
train$MoSold = as.factor(train$MoSold)
```

9) Correction of errors: we found that “Exterior2nd” has a record of “Brk Cmn”, which does not match with the data description “BrkComm”. So we rename it (in order to match with “Exterior1st”)

```
names(test)[names(test) == "Brk Cmn"] <- "BrkComm"
```

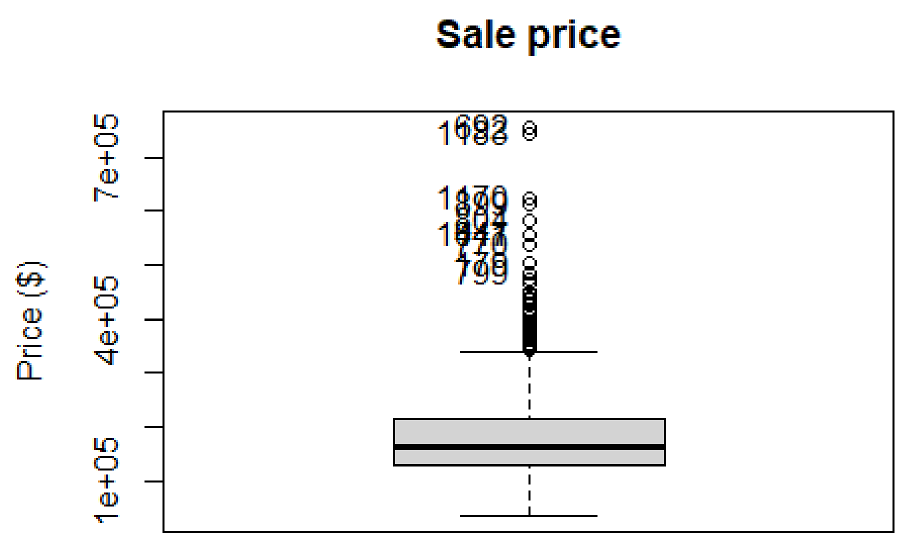
Lastly, we define the new indexes of all types of variables after transformation.

```
# Find numerical, categorical and date variables after the imputation
id_num_val = which(sapply(test, is.numeric)==TRUE)

# We won't analyze the id variable
id_num_val = as.numeric(id_num_val)[-1];
id_cat_val = which(sapply(test, is.factor)==TRUE)
id_cat_val = as.numeric(id_cat_val); id_cat_val
id_date_val = c(20,21,60,77,78)

# In our datasets, categorical variables are:
Categorical_val = c("MSSubClass","MSZoning","Street","Alley","LotShape"
,"LandContour","Utilities","LotConfig","LandSlope","Neighborhood","Cond
ition1","Condition2","BldgType","HouseStyle","OverallQual","OverallCond
","RoofStyle","RoofMatl","Exterior1st","Exterior2nd","MasVnrType","Exte
rQual","ExterCond","Foundation","BsmtQual","BsmtCond","BsmtExposure","B
smtFinType1","BsmtFinType2","BsmtFinSF2","Heating","HeatingQC","Central
Air","Electrical","LowQualFinSF","BsmtHalfBath","KitchenAbvGr","Kitchen
Qual","Functional","FireplaceQu","GarageType","GarageFinish","GarageQua
l","GarageCond","PavedDrive","EnclosedPorch","X3SsnPorch","ScreenPorch"
,"PoolArea","PoolQC","Fence","MiscFeature","SaleType","SaleCondition","
MoSold")

# The numerical variables, except the target are
Numerical_val = c("LotFrontage","LotArea","YearBuilt","YearRemodAdd","M
asVnrArea","BsmtFinSF1","BsmtUnfSF","TotalBsmtSF","X1stFlrSF","X2ndFlrS
F","GrLivArea","BsmtFullBath","FullBath","HalfBath","BedroomAbvGr","Tot
RmsAbvGrd","Fireplaces","GarageYrBlt","GarageCars","GarageArea","WoodDe
ckSF","OpenPorchSF","YrSold")

train_num <- dplyr::select(train, Numerical_val)
train_cat <- dplyr::select(train, Categorical_val)
```

1. Univariate outliers detection

First we analysed the target variable, where we found 12 severe outliers as this variable. Because the target variable can not be imputed we decided to keep those observacions and, in the remove them in the model creation in case they are influents points. You can see all the outliers in the following plot

```
sevout <- quantile(train$SalePrice,0.75, na.rm=TRUE)+3*(quantile(train$
SalePrice, 0.75,na.rm=TRUE)-quantile(train$SalePrice,0.25,na.rm=TRUE))
target_outlier <- which(train$SalePrice > sevout)

Boxplot(train$SalePrice, main = "Sale price", ylab = "Price ($)")
```




```
##  [1]  692 1183 1170  899  804 1047  441  770  179  799

severe_outliers <- function(data) {
  ss <- summary(data)
  # Upper/lower severe thresholds
  utso <- as.numeric(ss[5]+3*(ss[5]-ss[2]))
  ltso <- as.numeric(ss[2]-3*(ss[5]-ss[2]))
  return (which((data>utso)|(data<ltso)))}
```

Secondly, for all remaining numerical variables (26), we detected outliers and, for severe outliers, we set them to NA to impute them. This process was done automatically with a loop.

```
# Function to detect outliers
severe_outliers <- function(data) {
  ss <- summary(data)
  # Upper/lower severe thresholds
  utso <- as.numeric(ss[5]+3*(ss[5]-ss[2]))
  ltso <- as.numeric(ss[2]-3*(ss[5]-ss[2]))
  return (which((data>utso)|(data<ltso)))}

# Set them to NA'n and visualize them
par(mfrow=c(1,2))

for (var in id_num_val) {
```

```
  train[severe_outliers(train[,var]),var] <- NA
  Boxplot(train[,var], ylab = names(test)[var], main = "Train")
  test[severe_outliers(test[,var]),var] <- NA
  Boxplot(test[,var], ylab = names(test)[var], main = "Test")}

par(mfrow=c(1,1))
```

2. PCA imputation

Before the detection of outliers there was arround 1% of missing in some numerical variables (see the profiling at the annexes for more datail). After this detection, the variables that contained most missings were "GarageYrBlt" (6% in train and 5% in test), "MasVnrArea" (2% in train and 3% in test), and "OpenPorchSF"(1% in both).

To impute, we assumed that all numerical variables had NA's that were at random and used a PCA to impute both "test" and "train" datasets. As the quartile distributions for all imputed variables are similar, as we can see in the box-plot, we conclude that the imputation was successful for all variables and created a new dataframe with the imputed values. However, for train, we found that for OpenPorchSF feature, there is a negative record. As this is the square feet for open porch area, and it cannot be negative. We suspect that it could be 0, and transformed it.

```
# Impute
res.PCA = imputePCA (train[,id_num_val])
str (res.PCA)
str(res.PCA$completeObs)

res.PCA.test = imputePCA (test[,id_num_val])   # impute numeric variabl
es
str (res.PCA.test)
str(res.PCA.test$completeObs)

# Create a new dataframe
train_impute <- data.frame(res.PCA$completeObs)
train_impute$SalePrice <- train$SaleP
test_impute <- data.frame(res.PCA.test$completeObs)

# Check if the imputation was successful or not: TRAIN
before_imputation = summary(train[,id_num_val])
after_imputation = summary(train_impute)

label = c('Before imputation', 'After imputation')

for (x in c(1,2,5,6,8,9,11,15,16,18,20,21,22)) {
d = data.frame(A = train[,id_num_val][x], B = train_impute[,x])
b = boxplot(d, names=label, main = names(train[,id_num_val][x]));b}
```

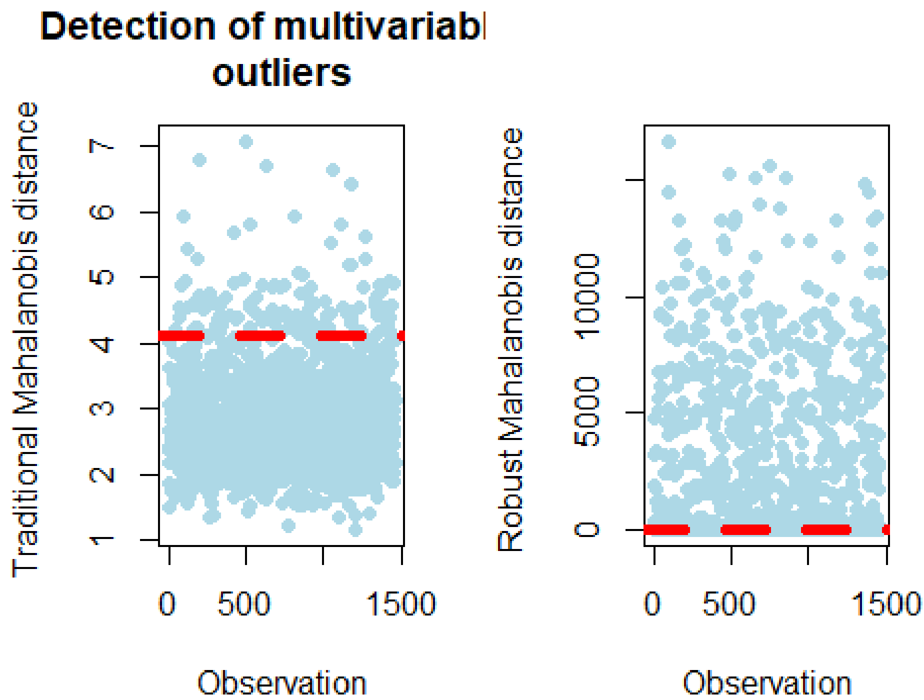
```
# Transform all negative values of "OpenPorchSF" to 0's
train_impute[which(train_impute$OpenPorchSF < 0),"OpenPorchSF"] = 0

# Check if the imputation was successful or not: TEST
before_imputation_test = summary(test[,id_num_val])
after_imputation_test = summary(test_impute)

label = c('Before imputation', 'After imputation')

for (x in c(1,2,5,6,8,9,11,15,16,18,20,21,22)) {
d = data.frame(A = test[,id_num_val][x], B = test_impute[,x])
b = boxplot(d, names=label, main = names(test[,id_num_val][x]));b}
```

3. Multivariate outliers detection

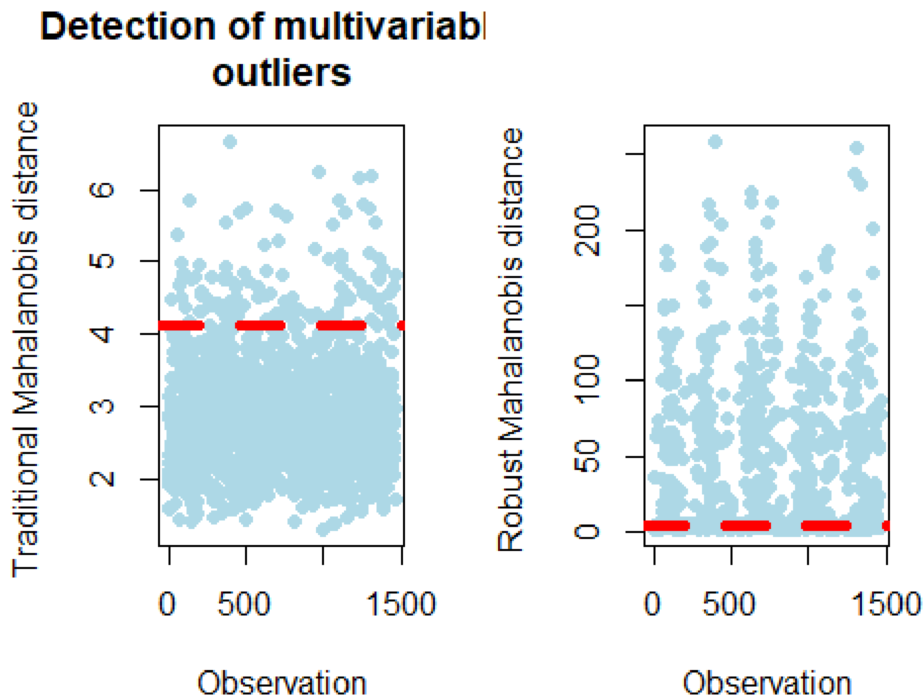
After the imputation, we decided to perform a Moutlier analysis to detect multivariate outliers. As using all numerical variables returns a singular matrix we decided to make the analysis with only the following variables: “LotFrontage”, “LotArea”, “YearRemodAdd”, “BsmtFinSF1”, “BsmtUnfSF”, “GrLivArea”, “Fireplaces”, “GarageYrBlt”, “GarageArea”.

The analysis showed that there are 112 multivariate outliers in the train dataset and 115 in the test datset.

```
set.seed(123) #ensure that we always get the same result in Moutlier
# Best combination of variables
id_num_val_not_corr = c(1, 2, 4, 6, 7, 11, 17,18, 20)

# Analysis for train
res.mout <- Moutlier(train_impute[,id_num_val_not_corr], quantile = 0.9
5, plot= FALSE)

par(mfrow=c(1,2))
plot(res.mout$md, col="lightblue", pch = 19, main = 'Detection of multi
variable outliers', xlab= 'Observation', ylab ='Traditional Mahalanobis
 distance ')
abline(h = res.mout$cutoff, col = "red", lwd = 5, lty = 2)

plot(res.mout$rd, col="lightblue", pch = 19, xlab= 'Observation', ylab
='Robust Mahalanobis distance ')
abline(h = res.mout$cutoff, col = "red", lwd = 5, lty = 2)
```




```
par(mfrow=c(1,1))

outliers = which(res.mout$md>res.mout$cutoff & res.mout$rd > res.mout$c
utoff)
length(outliers)

## [1] 116

set.seed(123) #ensure that we always get the same result in Moutlier
# Analysis for test
res.mout.test <- Moutlier(test_impute[,id_num_val_not_corr], quantile =
 0.95, plot= FALSE)
par(mfrow=c(1,2))

plot(res.mout.test$md, col="lightblue", pch = 19, main = 'Detection of
multivariable outliers', xlab= 'Observation', ylab ='Traditional Mahala
nobis distance ')
abline(h = res.mout.test$cutoff, col = "red", lwd = 5, lty = 2)

plot(res.mout.test$rd, col="lightblue", pch = 19, xlab= 'Observation',
ylab ='Robust Mahalanobis distance ')
abline(h = res.mout.test$cutoff, col = "red", lwd = 5, lty = 2)
```




```
par(mfrow=c(1,1))

outliers.test = which(res.mout.test$md>res.mout.test$cutoff & res.mout.
test$rd > res.mout.test$cutoff)
length(outliers.test)

## [1] 115
```

4. EDA

The last step of the preprocessing was the exploratory data analysis. This step was done automatically using the reports generated with the “SmartEDA” library that you can find in the annexes. The reports were generated considering “train” and “test” files after imputation and just after loading them, without any transformation.

The most relevant conclusions of EDA, considering all numerical values are:

1 - “Train” and “test” datasets contains observations that follows a similar distribution for all variables, numerical and categorical. There are also similarities in the % of missings and all the other summaries.

2 - Both datasets are highly unbalanced in almost all categories. This is specially relevant in variables like “ExterQual” or “Foundation”, where only 2 out of 6 categories retains 86% of the accumulative probability.

3 - Numerical variables have a non normal distribution according to Shapiro–Wilk and Kolmogorov-Smirnov tests. This is specially relevant when modelling as linear models requires normality.

```
# Tests for normality (done in all numerical variables)
ks.test(train$LotArea, y = 'pnorm')
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  train$LotArea
## D = 1, p-value < 2.2e-16
## alternative hypothesis: two-sided

shapiro.test(train$LotArea)

##
##  Shapiro-Wilk normality test
##
## data:  train$LotArea
## W = 0.97518, p-value = 5.751e-15
```

5. Profiling and selection of categorical features

Once we have the data clean and preprocesed, we have selected the 10 most relevant categories using the profiling of FactoMiner. More precisely, we alinised the relationship between variables in “train” datasets with “SalePrice” and selected the categorical variables with an smaller p-value.

The variables that we selected, sorted starting with the smallest p-value, are: OverallQual, ExterQual, BsmtQual, KitchenQual, Neighborhood, GarageFinish FireplaceQu, Foundation, GarageType and MSSubClass.

```
# Profiling: selecting only the 10 more significative qualitative varia
bles
res.con = condes(train, 80)
res.con$quali[1:10,]

##                    R2       p.value
## OverallQual  0.6062371 1.509421e-292
## ExterQual    0.4773878 1.439551e-204
## BsmtQual     0.4649938 8.158548e-196
## KitchenQual  0.4565986 3.032213e-192
## Neighborhood 0.4083529 8.865334e-167
## GarageFinish 0.3058737 6.228747e-115
## FireplaceQu  0.2939608 2.971217e-107
## Foundation   0.2563684  5.791895e-91
## GarageType   0.2492042  6.117026e-87
## MSSubClass   0.2463160  8.662166e-79
```

Additionally, we analysed the correlation of numerical variables with the target. According to the profile all numerical variables have a R^2 of p < 0.05 except for "YrSold". Furthermore, we have used cor.test() to test against H0="correlation between"YrSold" and "SalePrice" is 0" and we have failed to reject H0. Therefore, "YrSold" cannot be used to model "SalePrice".

```
res.con$quanti
res.con$category

# Test the correlation between the target and YrSold
cor.test(train$YrSold, train$SalePrice)
```

6. Analysis of correlation of numerical variables

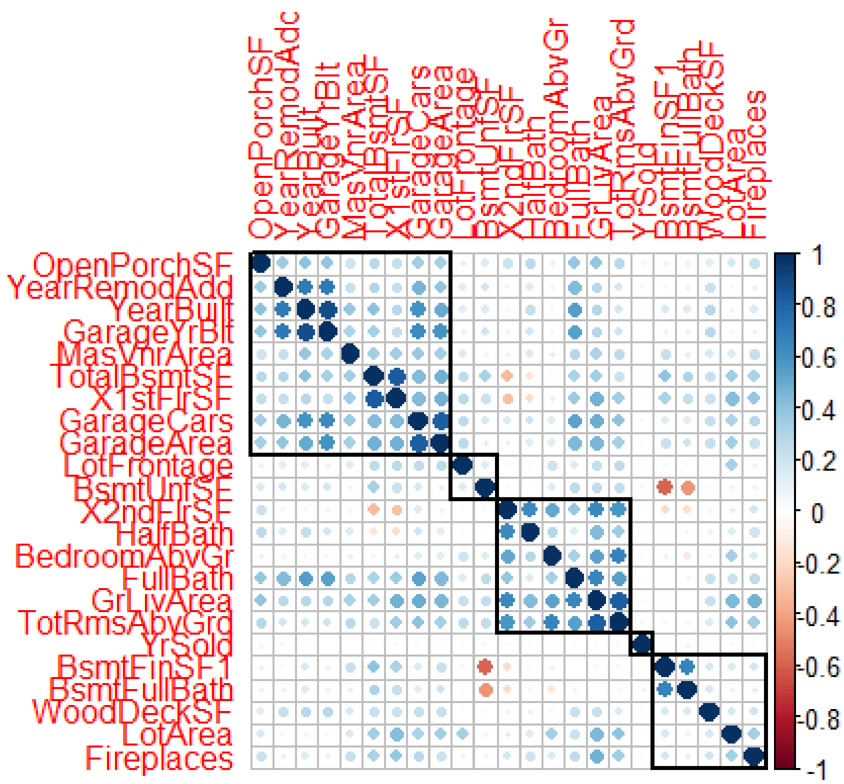
Using the basic profiling of Factominer we discover that the most correlated numerical variables with the target, with more than 50 % of R^2 are: GrLivArea, GarageCars, GarageArea, TotalBsmtSF, X1stFlrSF, YearBuilt, FullBath, YearRemodAdd, GarageYrBlt and TotRmsAbvGrd.

```
res.con = condes(train, 80)
res.con$quanti

##                correlation       p.value
## GrLivArea        0.7205163 1.939850e-233
## TotalBsmtSF      0.6442410 2.113140e-171
## GarageCars       0.6404092 2.498644e-169
## GarageArea       0.6342993 7.769971e-165
## X1stFlrSF        0.6283722 6.703198e-161
## FullBath         0.5606638 1.236470e-121
## TotRmsAbvGrd     0.5368913 1.021117e-109
## YearBuilt        0.5228973 2.990229e-103
## YearRemodAdd     0.5071010  3.164948e-96
## GarageYrBlt      0.4863617  8.705128e-83
## Fireplaces       0.4669288  6.141487e-80
## LotArea          0.4235805  3.553955e-63
## MasVnrArea       0.4104427  4.224392e-59
## BsmtFinSF1       0.4070598  2.472431e-59
## OpenPorchSF      0.3665666  4.320047e-47
## WoodDeckSF       0.3273428  9.843914e-38
## X2ndFlrSF        0.3193338  5.764335e-36
## HalfBath         0.2841077  1.650473e-28
## BsmtFullBath     0.2271222  1.550344e-18
## LotFrontage      0.2152898  9.476093e-17
## BsmtUnfSF        0.2144791  1.182976e-16
## BedroomAbvGr     0.1694976  7.224235e-11
```

As variables are not normally distributed, we created the correlation matrix of all numerical variables using spearman. The result is ploted in a correlation plot, where we performed a cluster analysis to sort the variables, so that variables that are more correlated are placed closer to each other. Additionally, we decided to create 5 clusters, as we do not expect to work with a model with more than 5 numerical variables. Also, note that in this plot the target variable is not included as this analysis was already done.

The interpretation of this plot suggest that positive correlations are more common than negative, where the most important is between BsmtFullBath and BsmtFin with BsmtUnfSF. Also, there are some important positve correlarions that must be considered when making the model, for example, GarageArea is hightly correlated with GarageCars, so both variables should not be included in the same model.

```
# Calculate the correlation matrix and then plot it
corr_mat = cor(train_num, method = 'spearman', use = "complete.obs")
corrplot(corr_mat, order = 'hclust', addrect = 5)
```



7. Preparation of data for modelling

The last step of the preprocessing was to create a new file with all the variables that we will use to make our model. To do so, we added the 10 categorical variables to the imputed dataframe. The same process was done with "test" to predict the target variable using the model that we will create.

```
write.csv(train_impute, file='train_impute.csv', row.names = FALSE)
write.csv(test_impute, file='test_impute.csv', row.names = FALSE)
```

8. First model building

We create a first model with all the numerical variables that we selected previously.

```
df_num <- df[, which(sapply(df, is.numeric))]
m0 = lm(SalePrice ~ ., data=df_num)

summary(m0)

##
## Call:
## lm(formula = SalePrice ~ ., data = df_num)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -284830  -16703    -822   16153  227181
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.729e+06  1.307e+05 -13.231  < 2e-16 ***
## LotFrontage    1.060e+02  2.944e+01   3.602 0.000327 ***
## LotArea        1.706e+00  3.137e-01   5.437 6.36e-08 ***
## YearBuilt      3.881e+02  6.148e+01   6.313 3.64e-10 ***
## YearRemodAdd   4.972e+02  6.108e+01   8.139 8.55e-16 ***
## MasVnrArea     2.017e+01  7.339e+00   2.749 0.006061 **
## BsmtFinSF1     3.190e+01  5.585e+00   5.712 1.36e-08 ***
## BsmtUnfSF      1.009e+01  5.642e+00   1.788 0.074017 .
## TotalBsmtSF    2.907e+01  6.713e+00   4.330 1.59e-05 ***
## X1stFlrSF      6.815e+01  1.120e+01   6.086 1.48e-09 ***
## X2ndFlrSF      8.081e+01  1.043e+01   7.744 1.81e-14 ***
## GrLivArea     -1.388e+00  1.049e+01  -0.132 0.894698
## BsmtFullBath -1.966e+03  2.485e+03  -0.791 0.429079
## FullBath      -2.732e+03  2.793e+03  -0.978 0.328218
## HalfBath      -4.013e+03  2.677e+03  -1.499 0.134096
## BedroomAbvGr -1.367e+04  1.664e+03  -8.212 4.81e-16 ***
## TotRmsAbvGrd  1.917e+03  1.193e+03   1.607 0.108168
## Fireplaces     7.637e+03  1.732e+03   4.410 1.11e-05 ***
## GarageYrBlt   -8.597e+00  7.634e+01  -0.113 0.910357
## GarageCars     4.923e+03  2.858e+03   1.723 0.085186 .
## GarageArea     2.454e+01  1.026e+01   2.393 0.016845 *
## WoodDeckSF     2.125e+01  8.173e+00   2.600 0.009413 **
## OpenPorchSF    4.559e+01  1.812e+01   2.516 0.011978 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 34930 on 1437 degrees of freedom
## Multiple R-squared:  0.8096, Adjusted R-squared:  0.8067
## F-statistic: 277.7 on 22 and 1437 DF,  p-value: < 2.2e-16

vif(m0)
```

```
##   LotFrontage       LotArea     YearBuilt YearRemodAdd    MasVnrArea    B
smtFinSF1
##      1.152603      1.497042      4.123365      1.901884      1.320233
 7.068653
##     BsmtUnfSF   TotalBsmtSF     X1stFlrSF     X2ndFlrSF     GrLivArea Bsm
tFullBath
##      7.431487      8.834991     20.368811     24.811318     32.823752
 1.988987
##      FullBath      HalfBath  BedroomAbvGr  TotRmsAbvGrd    Fireplaces   Ga
rageYrBlt
##      2.832002      2.167450      2.147125      4.433981      1.490638
 4.237834
##    GarageCars    GarageArea    WoodDeckSF   OpenPorchSF
##      5.455675      5.549918      1.178558      1.233308
```

There are a lot of features with a vif correlation larger than 5. So, in order to reduce the amount of workload, we decided to keep those that are less than 5 and are highly correlated with our target.

```r
# Let's store the indices of the variables with at least one star in th
e lm and vif<5
id_num_star1 = c(1:5,15,17,21:23)
df_num1 <- df_num[, id_num_star1]
# And build a new model only with significance features
m1 = lm(SalePrice ~., data=df_num1)
summary(m1)
```

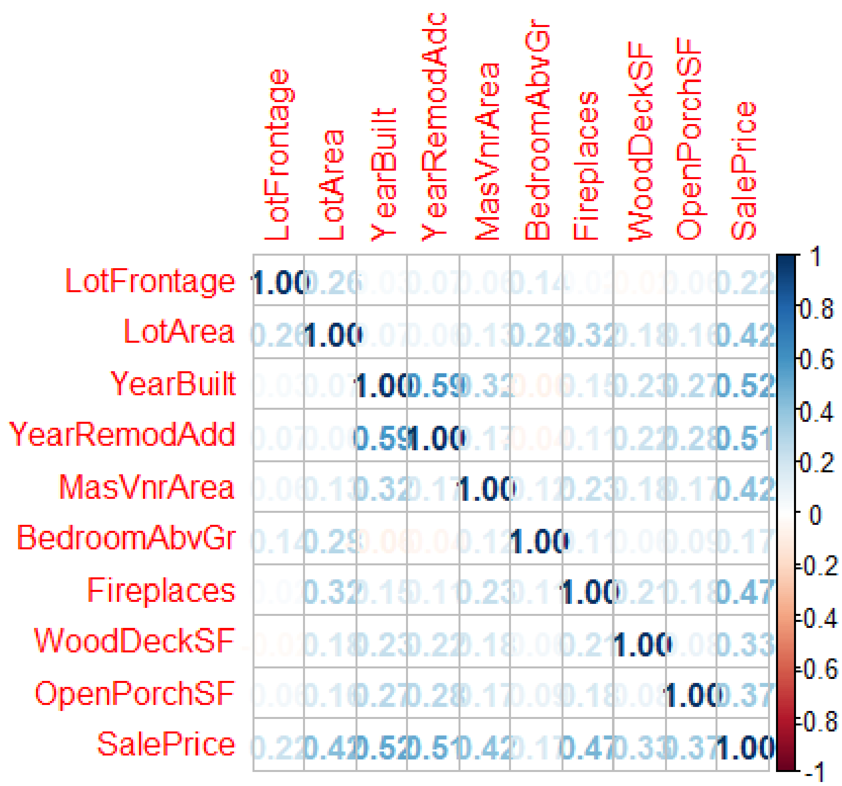
```
##
## Call:
## lm(formula = SalePrice ~ ., data = df_num1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -189053  -27080   -4750   19843  434071
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.976e+06  1.355e+05 -21.960  < 2e-16 ***
## LotFrontage   2.530e+02  4.024e+01   6.287 4.27e-10 ***
## LotArea       4.696e+00  4.103e-01  11.446  < 2e-16 ***
## YearBuilt     5.591e+02  5.581e+01  10.018  < 2e-16 ***
## YearRemodAdd  9.775e+02  7.879e+01  12.405  < 2e-16 ***
## MasVnrArea    9.444e+01  9.795e+00   9.642  < 2e-16 ***
## BedroomAbvGr  5.369e+03  1.691e+03   3.175  0.00153 **
## Fireplaces    3.096e+04  2.183e+03  14.184  < 2e-16 ***
## WoodDeckSF    6.130e+01  1.124e+01   5.455 5.75e-08 ***
## OpenPorchSF   1.598e+02  2.458e+01   6.499 1.11e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 48990 on 1450 degrees of freedom
## Multiple R-squared:  0.622,	Adjusted R-squared:  0.6197
## F-statistic: 265.1 on 9 and 1450 DF,  p-value: < 2.2e-16
```

```r
vif(m1)
```

```
##  LotFrontage      LotArea    YearBuilt YearRemodAdd   MasVnrArea Bed
roomAbvGr
##     1.094625     1.301887     1.727062     1.608513     1.195294
 1.126313
##   Fireplaces   WoodDeckSF  OpenPorchSF
##     1.203511     1.132628     1.153802
```

```r
# As we can observe, vif correlations are much better, all values are l
ess than 2.
# So the next step is to check the correlation between predictors.
corr_mat <- cor(df_num1)
corrplot(corr_mat, method = "number")
```



Feature “YearBuilt” and “YearRemodAdd” are highly correlated between them, and “YearBuilt” is more correlated to our target SalePrice. Hence, we remove YearRemodAdd in the next model.

```r
# Building the model without "YearRemodAdd"
id_num_star2 = c(1:3,5,15,17,21:23)
df_num2 <- df_num[, id_num_star2]
```

```
m2 = lm(SalePrice ~., data=df_num2)
summary(m2)
```

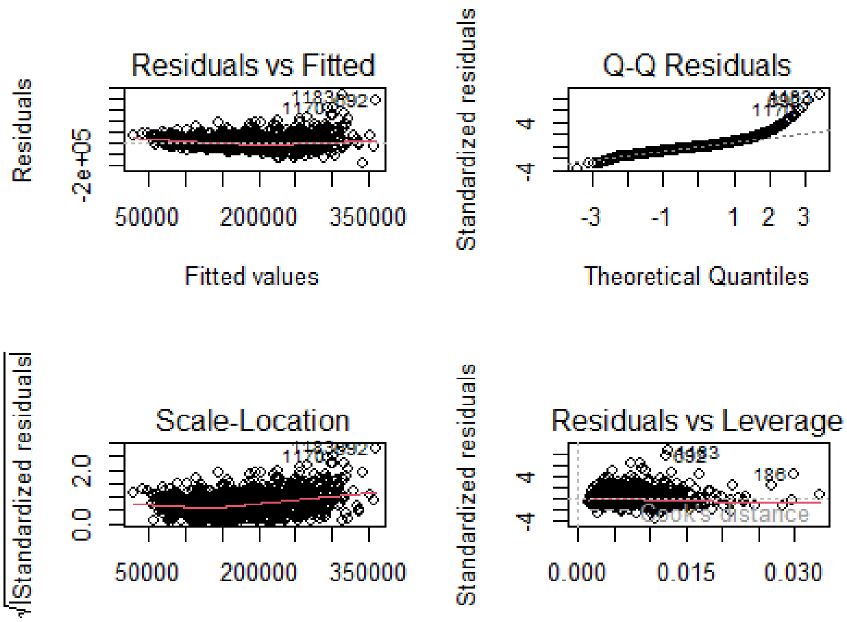
```
##
## Call:
## lm(formula = SalePrice ~ ., data = df_num2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -183896  -29373   -5718   21404  429918
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.759e+06  9.831e+04 -17.894  < 2e-16 ***
## LotFrontage   2.879e+02  4.220e+01   6.823 1.30e-11 ***
## LotArea       4.570e+00  4.312e-01  10.598  < 2e-16 ***
## YearBuilt     9.247e+02  4.983e+01  18.559  < 2e-16 ***
## MasVnrArea    8.807e+01  1.028e+01   8.564  < 2e-16 ***
## BedroomAbvGr  4.819e+03  1.777e+03   2.712  0.00677 **
## Fireplaces    3.121e+04  2.295e+03  13.599  < 2e-16 ***
## WoodDeckSF    7.673e+01  1.174e+01   6.534 8.83e-11 ***
## OpenPorchSF   2.072e+02  2.553e+01   8.116 1.02e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51510 on 1451 degrees of freedom
## Multiple R-squared:  0.5819, Adjusted R-squared:  0.5796
## F-statistic: 252.4 on 8 and 1451 DF,  p-value: < 2.2e-16
```

Now, the most correlated variables in our model have at most a coefficient of correlation of 0.315, which in the context of real estate it is weak. We have obtained this information from https://37parallel.com/real-estate-correlation/.

9. Model analysis and iteration

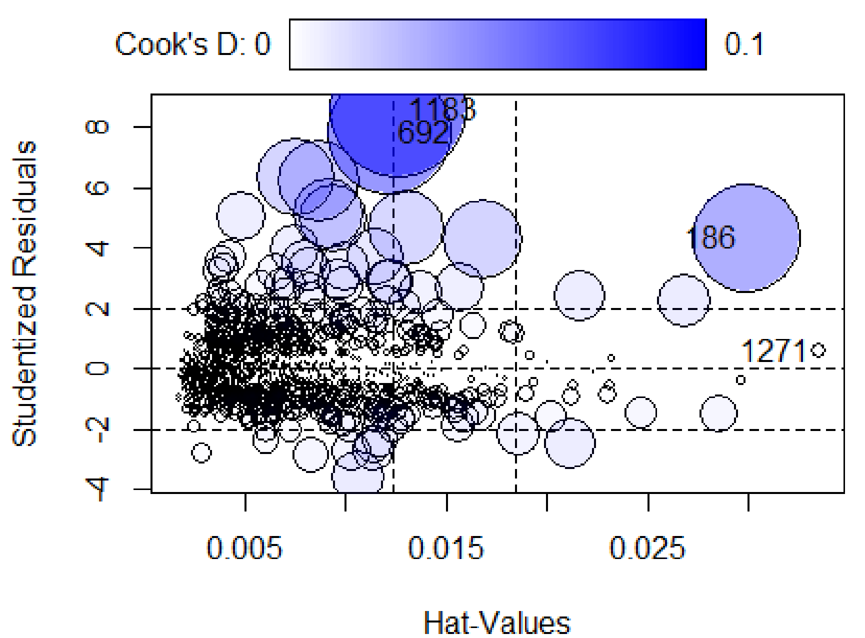
First, let us plot the residuals of m2 to be able to compare them with the next iterations of the model.

```
par(mfrow=c(2,2))
plot(m2)
```



We analysed if there were influential data and found 3 observations with a bigger Cook's distance than the threshold (considered as 2/sqrt(n)). Consequently, we decided to remove those observations.

```
# Check the influential plot before removing the influential observatio
n.
influencePlot(m2)
```

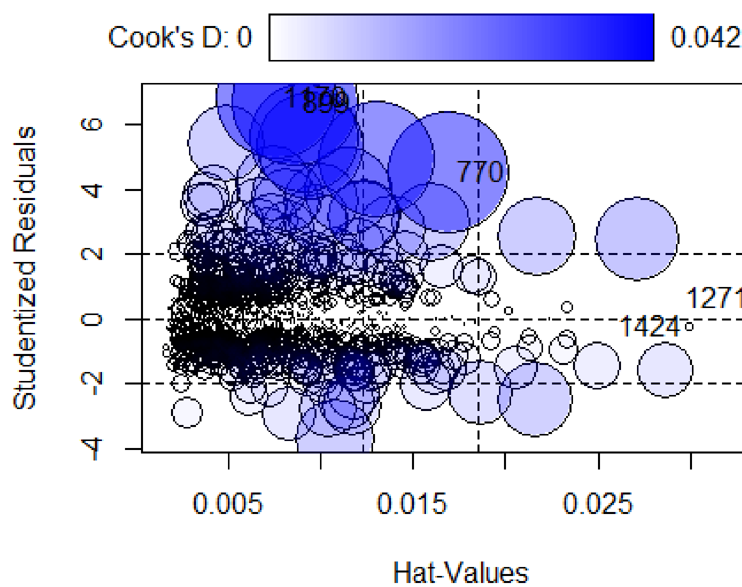


```
##          StudRes        Hat       CookD
## 186  4.3264547 0.02991534 0.063362899
## 692  7.8641613 0.01219078 0.081391863
## 1183 8.6087259 0.01264601 0.100407718
## 1271 0.5819775 0.03351270 0.001305513
```

```r
# Calculate D's threshold
D_thresh <- 2/sqrt(dim(df_num2)[1]); D_thresh
```

```
## [1] 0.05234239
```

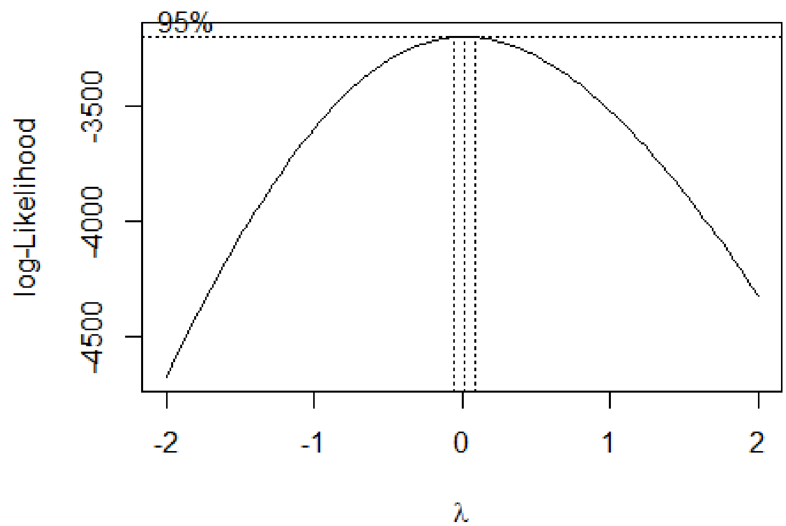
```r
#Remove the points and fit the model again
influent <- c(1183, 692, 186)

df <- df[-influent,]
df_num <- df[, which(sapply(df, is.numeric))]
df_num2 <- df_num[, id_num_star2]
m2 = lm(SalePrice ~., data=df_num2)

influencePlot(m2)
```




```
##           StudRes         Hat         CookD
## 770   4.5443947 0.016903181 0.0389248857
## 899   6.6815842 0.008642834 0.0419803351
## 1170  6.8469664 0.007587455 0.0386020474
## 1271  0.6859165 0.033627929 0.0018197619
## 1424 -0.2309732 0.029944826 0.0001831007
```

Firstly, we check if there is any needed transformation with boxcox().

```r
boxcox(m2)
```




```r
# As the lambda is greater than 0, we should apply a logarithmic transf
ormation to SalePrice
m3 = lm(log(SalePrice)~., data=df_num2)
summary(m3)

##
## Call:
## lm(formula = log(SalePrice) ~ ., data = df_num2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.09825 -0.14427 -0.00563  0.12877  0.94288
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.120e-02  4.511e-01  -0.025     0.98
## LotFrontage   9.145e-04  1.934e-04   4.728 2.48e-06 ***
## LotArea       2.117e-05  1.980e-06  10.697  < 2e-16 ***
## YearBuilt     5.794e-03  2.287e-04  25.337  < 2e-16 ***
## MasVnrArea    3.251e-04  4.705e-05   6.910 7.25e-12 ***
## BedroomAbvGr  5.230e-02  8.124e-03   6.438 1.64e-10 ***
## Fireplaces    1.657e-01  1.051e-02  15.766  < 2e-16 ***
## WoodDeckSF    3.610e-04  5.372e-05   6.720 2.61e-11 ***
## OpenPorchSF   9.952e-04  1.174e-04   8.478  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2354 on 1448 degrees of freedom
## Multiple R-squared:  0.6468, Adjusted R-squared:  0.6449
## F-statistic: 331.5 on 8 and 1448 DF,  p-value: < 2.2e-16
```

Compared with m2, adjusted R-squared has increased about 4%.

We will proceed now with the study of possible variable transformations. We'll assign 10^(-6) to all cells equal to 0 to be able to use boxTidwell() without altering too much the model

```
df_num2 = replace(df_num2, df_num2 == 0, 1e-6)
summary(df_num2)

##   LotFrontage        LotArea        YearBuilt      MasVnrArea
##  Min.   :  0.00   Min.   : 1300   Min.   :1872   Min.   :  0.00
##  1st Qu.: 42.00   1st Qu.: 7540   1st Qu.:1954   1st Qu.:  0.00
##  Median : 63.00   Median : 9416   Median :1973   Median :  0.00
##  Mean   : 57.16   Mean   : 9524   Mean   :1971   Mean   : 91.67
##  3rd Qu.: 79.00   3rd Qu.:11345   3rd Qu.:2000   3rd Qu.:162.00
##  Max.   :182.00   Max.   :23595   Max.   :2010   Max.   :664.00
##   BedroomAbvGr        Fireplaces         WoodDeckSF      OpenPorchSF

##  Min.   :0.000001   Min.   :0.000001   Min.   :  0.0   Min.   :  0.0
0
##  1st Qu.:2.000000   1st Qu.:0.000001   1st Qu.:  0.0   1st Qu.:  0.0
0
##  Median :3.000000   Median :1.000000   Median :  0.0   Median : 24.0
0
##  Mean   :2.861290   Mean   :0.610158   Mean   : 92.7   Mean   : 42.7
8
##  3rd Qu.:3.000000   3rd Qu.:1.000000   3rd Qu.:168.0   3rd Qu.: 66.0
0
##  Max.   :6.000000   Max.   :3.000000   Max.   :670.0   Max.   :267.0
0
##    SalePrice
##  Min.   : 34900
##  1st Qu.:129900
##  Median :163000
##  Mean   :179938
##  3rd Qu.:213500
##  Max.   :625000

boxTidwell(log(SalePrice) ~ LotArea+YearBuilt+MasVnrArea, data = df_num
2)

##            MLE of lambda Score Statistic (t)  Pr(>|t|)
## LotArea          0.50589             -4.0795 4.759e-05 ***
## YearBuilt       69.59853             14.9243 < 2.2e-16 ***
## MasVnrArea       0.99009             -0.1764      0.86
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  5
##
```

```
## Score test for null hypothesis that all lambdas = 1:
## F = 79.82, df = 3 and 1450, Pr(>F) = < 2.2e-16

# We should apply sqrt(LotArea). YearBuilt's lambda is too large, so it
 would be difficult to interpret the model using it. MasVnrArea has a t
oo large p-value, so we cannot reject the null hypothesis that its lamb
da = 1.
boxTidwell(log(SalePrice)~LotFrontage, data = df_num2)

## Warning in boxTidwell.default(y, X1, X2, max.iter = max.iter, tol =
tol, :
## maximum iterations exceeded

##  MLE of lambda Score Statistic (t)  Pr(>|t|)
##        -2.8984              11.274 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  26

# Too small lambda
boxTidwell(log(SalePrice)~BedroomAbvGr, data = df_num2)

## Warning in boxTidwell.default(y, X1, X2, max.iter = max.iter, tol =
tol, :
## maximum iterations exceeded

##  MLE of lambda Score Statistic (t) Pr(>|t|)
##          1.528              0.7048    0.481
##
## iterations =  26

# Too large p-value
boxTidwell(log(SalePrice)~Fireplaces, data =df_num2)

##  MLE of lambda Score Statistic (t)  Pr(>|t|)
##        0.24931             -7.6471 3.717e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  2

# We apply log() to Fireplaces
boxTidwell(log(SalePrice)~WoodDeckSF, data = df_num2)

##  MLE of lambda Score Statistic (t)  Pr(>|t|)
##        0.52909             -4.9103 1.012e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  7
```

```
# We apply sqrt() to WoodDeckSF
boxTidwell(log(SalePrice)~OpenPorchSF, data = df_num2)

## Warning in boxTidwell.default(y, X1, X2, max.iter = max.iter, tol =
tol, :
## maximum iterations exceeded

##  MLE of lambda Score Statistic (t)  Pr(>|t|)
##        -8.7203                 -12 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## iterations =  26

# Too small lambda
```

Using the boxTidwell method, the transformation below can be applied to m4.

```
m4 = lm(log(SalePrice) ~LotFrontage+sqrt(LotArea)+YearBuilt+MasVnrArea+
BedroomAbvGr+log(Fireplaces)+sqrt(WoodDeckSF)+OpenPorchSF,data=df_num2)
summary(m4)

##
## Call:
## lm(formula = log(SalePrice) ~ LotFrontage + sqrt(LotArea) +YearBuilt
 + MasVnrArea + BedroomAbvGr + log(Fireplaces) + sqrt(WoodDeckSF) + Ope
nPorchSF, data = df_num2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10570 -0.14470 -0.00927  0.13002  0.87600
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.955e-01  4.629e-01   1.286 0.198506
## LotFrontage      7.427e-04  1.950e-04   3.809 0.000145 ***
## sqrt(LotArea)    4.345e-03  3.689e-04  11.777  < 2e-16 ***
## YearBuilt        5.484e-03  2.337e-04  23.471  < 2e-16 ***
## MasVnrArea       3.457e-04  4.712e-05   7.335 3.68e-13 ***
## BedroomAbvGr     4.847e-02  8.193e-03   5.916 4.10e-09 ***
## log(Fireplaces)  1.480e-02  9.829e-04  15.058  < 2e-16 ***
## sqrt(WoodDeckSF) 6.073e-03  9.215e-04   6.591 6.11e-11 ***
## OpenPorchSF      1.004e-03  1.178e-04   8.525  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2361 on 1448 degrees of freedom
## Multiple R-squared:  0.6446, Adjusted R-squared:  0.6426
## F-statistic: 328.3 on 8 and 1448 DF,  p-value: < 2.2e-16
```

Adjusted R-squared has increased slightly. Since we cannot find a significant improvement, we will compare m3 and m4 with a more advanced tool, the BIC.

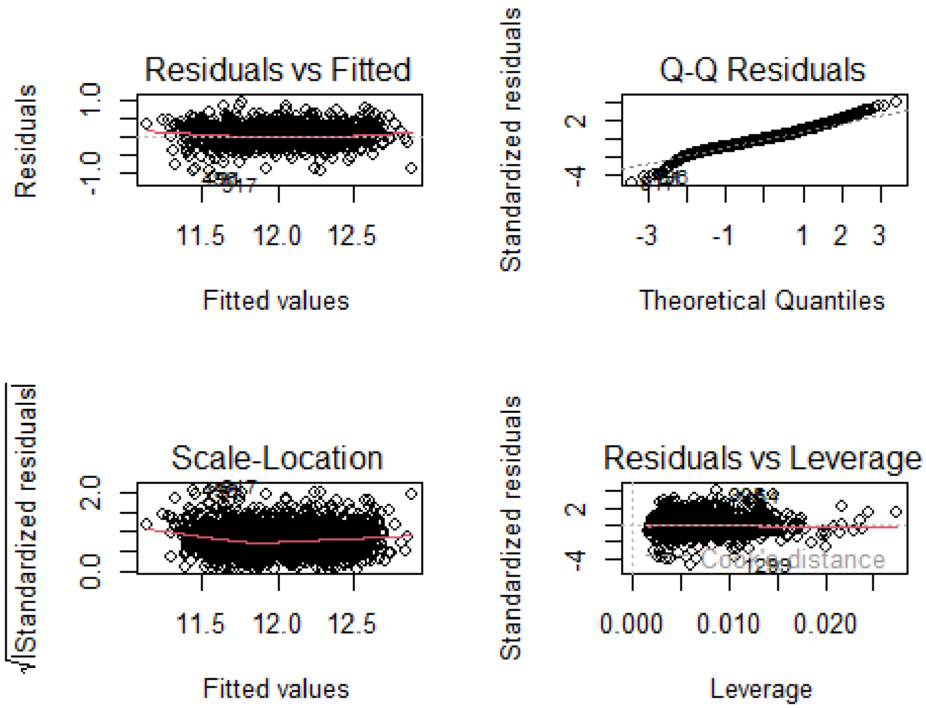
```
BIC(m3, m4)

##    df        BIC
## m3 10 -16.599825
## m4 10  -7.344876
```

The overall improvement of applying all transformations simultaneously is small, so we decided to check different combinations to find a better result.

```
m5 = lm(log(SalePrice) ~ LotFrontage+LotArea+YearBuilt+MasVnrArea+ Bedr
oomAbvGr+log(Fireplaces)+sqrt(WoodDeckSF)+OpenPorchSF,data=df_num2)
m6 = lm(log(SalePrice) ~LotFrontage+sqrt(LotArea)+YearBuilt+MasVnrArea+
BedroomAbvGr+Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF,data=df_num2)
m7 = lm(log(SalePrice) ~ LotFrontage+sqrt(LotArea)+YearBuilt+MasVnrArea
+BedroomAbvGr+log(Fireplaces)+WoodDeckSF+OpenPorchSF,data=df_num2)
m8 = lm(log(SalePrice)~LotFrontage+sqrt(LotArea)+YearBuilt+MasVnrArea+
          BedroomAbvGr+Fireplaces+WoodDeckSF+OpenPorchSF, data=df_num2)
m9 = lm(log(SalePrice)~LotFrontage+LotArea+YearBuilt+MasVnrArea+Bedroom
AbvGr+log(Fireplaces)+WoodDeckSF+OpenPorchSF, data=df_num2)
m10 = lm(log(SalePrice)~LotFrontage+LotArea+YearBuilt+MasVnrArea+Bedroo
mAbvGr+Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF, data=df_num2)
BIC(m4,m5,m6,m7,m8,m9,m10)

##     df        BIC
## m4  10  -7.344876
## m5  10   1.508573
## m6  10 -26.417016
## m7  10  -6.294640
## m8  10 -24.705618
## m9  10   2.411503
## m10 10 -18.163916
```

The best model is m6, that only applies sqrt() to LotArea and WoodDeckSF. For this model we have compared the distribution of residuals and realized that it is very similar to the original model.

```
par(mfrow=c(2,2))
m11 = lm(log(SalePrice) ~LotFrontage+sqrt(LotArea)+YearBuilt+MasVnrArea
+BedroomAbvGr+Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF,data=df_num)
BIC(m3,m11)

##     df       BIC
## m3  10 -16.59982
## m11 10 -26.41669

plot(m11)
```



10. Adding Factors to the numerical model

We followed a heuristic approach when we added factors to the model. As there was an important amount of numeric variables, we tried to add factor variables one by one. We started with the predictor most correlated with the target and continued in decreasing order. To test the improvement of the model's forecasting capability we analysed its BIC and R^2. Moreover, Anova() and step() methods suggest whether some predictors should be removed.

The results of the code of this section are very long and repetitive, so we hide them in the report.

```
m12 = lm(log(SalePrice)~LotFrontage+sqrt(LotArea)+YearBuilt+MasVnrArea
+BedroomAbvGr+Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF+OverallQual, data
=df)
BIC(m11,m12)
Anova(m12)
step(m12, k = log(nrow(df)))
```

Comparing m11 and m12, there was a huge improvement in terms of BIC and Adjusted R-squared, as we expected.

The Anova test indicates that LotFrontage loses its significance once we add OverallQual, and the step method suggests to remove it.

```
m12.1 = lm(log(SalePrice)~sqrt(LotArea)+YearBuilt+MasVnrArea+BedroomAbv
Gr +Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF+OverallQual, data=df)
summary(m12.1)
BIC(m10,m12,m12.1)
```

After removing LotFrontage, although R^2 didn't change, BIC increased because we used less variables and avoided overfitting.

Next, in m13, we have added ExterQual.

```
m13 = lm(log(SalePrice)~sqrt(LotArea)+YearBuilt+MasVnrArea+BedroomAbvGr
+ Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF+OverallQual+ExterQual, data=d
f)
summary(m13)
BIC(m13,m12.1)
Anova(m13)
step(m13, k = log(nrow(df)))
```

All parameters show that it is correct to add ExterQual, so we continue by adding BsmtQual to the model.

```
m14 = lm(log(SalePrice)~sqrt(LotArea)+YearBuilt+MasVnrArea+BedroomAbvGr
+ Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF+OverallQual+ExterQual+BsmtQua
l, data=df)
summary(m14)
BIC(m14,m13)
Anova(m14)
step(m14, k = log(nrow(df)))
```

After this, we add KitcheQual.

```
m15 = lm(log(SalePrice)~sqrt(LotArea)+YearBuilt+MasVnrArea+BedroomAbvGr
+Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF+OverallQual+ExterQual+BsmtQual
+KitchenQual, data=df); summary(m15)
BIC(m15,m14)
Anova(m15)
step(m15, k = log(nrow(df)))
```

The step method shows that ExterQual, after adding the KitchenQual, has lost significance and suggests to remove it. Indeed, BIC improves afterwards.

```
m15.1 = lm(log(SalePrice)~sqrt(LotArea)+YearBuilt+MasVnrArea+BedroomAbv
Gr+Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF+OverallQual+BsmtQual+Kitchen
Qual, data=df); summary(m15.1)
BIC(m15.1,m15)
Anova(m15.1)
step(m15.1, k = log(nrow(df)))
```

Adding Neighborhood to the model.

```
m16.1 = lm(log(SalePrice)~sqrt(LotArea)+YearBuilt+MasVnrArea+BedroomAbv
Gr+Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF+OverallQual+BsmtQual+Kitchen
```

```
Qual+Neighborhood, data=df); summary(m16.1)
BIC(m16.1,m15.1)
Anova(m16.1)
step(m16.1, k = log(nrow(df)))
```

Adding GarageFinish.

```
m16.2 = lm(log(SalePrice)~sqrt(LotArea)+YearBuilt+MasVnrArea+BedroomAbv
Gr+Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF+OverallQual+BsmtQual+Kitchen
Qual+Neighborhood+GarageFinish, data=df); summary(m16.2)
BIC(m16.2,m16.1)
Anova(m16.2)
step(m16.2, k = log(nrow(df)))
```

Adding FireplaceQu.

```
m16.3 = lm(log(SalePrice)~sqrt(LotArea)+YearBuilt+MasVnrArea+BedroomAbv
Gr+Fireplaces+sqrt(WoodDeckSF)+OpenPorchSF+OverallQual+BsmtQual+Kitchen
Qual+Neighborhood+GarageFinish+FireplaceQu, data=df)
summary(m16.3)
BIC(m16.3,m16.2,m16.1)
Anova(m16.3)
step(m16.3, k = log(nrow(df)))
```

In m16.3, FireplaceQu's coefficient has a p-value larger than 0.05 and, indeed, step() suggests to remove it from the model. Hence, we stop adding new categorical variables.

11. Checking possible Interactions

YearBuilt and OverallQual intuitively should interact because of inflation. Indeed, all variables could interact with YearBuilt, but OverallQual summarizes them.

We will also hide the output of this section's chunks to shorten the report.

```
m17 = lm(log(SalePrice)~sqrt(LotArea)+MasVnrArea+BedroomAbvGr+Fireplace
s+sqrt(WoodDeckSF)+OpenPorchSF+YearBuilt*OverallQual+BsmtQual+KitchenQu
al+Neighborhood+GarageFinish, data=df)
summary(m17)
BIC(m17,m16.2)
Anova(m17)
step(m17, k = log(nrow(df)))
```

2. LotArea and YearBuilt should interact as well because of inflation.

```
m18 = lm(log(SalePrice)~MasVnrArea+BedroomAbvGr+Fireplaces+sqrt(WoodDec
kSF)+OpenPorchSF+YearBuilt*OverallQual+sqrt(LotArea)*YearBuilt+OverallQ
ual+BsmtQual+KitchenQual+Neighborhood+GarageFinish, data=df); summary(m
18)
BIC(m18,m17,m16.2)
```

```r
Anova(m18)
step(m18, k = log(nrow(df)))
```

Any of these interactions have improved much the model, so we won't keep them. No other interaction would make sense, so we will not try anymore.

Our final model is m16.2. That is, log(SalePrice) ~ sqrt(LotArea) + YearBuilt + MasVnrArea + BedroomAbvGr + Fireplaces + sqrt(WoodDeckSF) + OpenPorchSF + OverallQual + BsmtQual + KitchenQual + Neighborhood + GarageFinish. Its adjusted R^2 is **0.8195** and its BIC is about **-972**.

```r
summary(m16.2)

## 
## Call:
## lm(formula = log(SalePrice) ~ sqrt(LotArea) + YearBuilt + MasVnrArea
##  + BedroomAbvGr + Fireplaces + sqrt(WoodDeckSF) + OpenPorchSF + Overall
Qual + BsmtQual + KitchenQual + Neighborhood + GarageFinish, data = df)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.94527 -0.08542  0.00503  0.09201  0.55128 
## 
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)    
## (Intercept)          9.065e+00  5.168e-01  17.539  < 2e-16 ***
## sqrt(LotArea)        3.614e-03  2.581e-04  14.002  < 2e-16 ***
## YearBuilt            1.239e-03  2.558e-04   4.841 1.43e-06 ***
## MasVnrArea           1.485e-04  3.440e-05   4.316 1.70e-05 ***
## BedroomAbvGr         5.447e-02  5.992e-03   9.090  < 2e-16 ***
## Fireplaces           8.099e-02  7.800e-03  10.384  < 2e-16 ***
## sqrt(WoodDeckSF)     2.996e-03  6.588e-04   4.548 5.87e-06 ***
## OpenPorchSF          3.190e-04  8.559e-05   3.728 0.000201 ***
## OverallQualGood      2.749e-01  2.140e-02  12.846  < 2e-16 ***
## OverallQualModerate  1.360e-01  1.689e-02   8.056 1.65e-15 ***
## OverallQualVBad     -4.854e-01  7.886e-02  -6.155 9.70e-10 ***
## OverallQualVGood     4.193e-01  3.798e-02  11.039  < 2e-16 ***
## BsmtQualFa          -1.584e-01  3.962e-02  -3.999 6.68e-05 ***
## BsmtQualGd          -8.603e-02  2.166e-02  -3.971 7.51e-05 ***
## BsmtQualNBsmt       -2.649e-01  3.759e-02  -7.049 2.79e-12 ***
## BsmtQualTA          -1.341e-01  2.568e-02  -5.221 2.05e-07 ***
## KitchenQualFa       -2.170e-01  3.791e-02  -5.724 1.26e-08 ***
## KitchenQualGd       -7.900e-02  2.315e-02  -3.412 0.000663 ***
## KitchenQualTA       -1.746e-01  2.485e-02  -7.027 3.24e-12 ***
## NeighborhoodPoor    -3.994e-02  1.289e-02  -3.099 0.001981 ** 
## NeighborhoodRich     1.354e-01  1.331e-02  10.168  < 2e-16 ***
## GarageFinishNGar    -1.900e-01  2.437e-02  -7.797 1.21e-14 ***
## GarageFinishRFn     -1.761e-02  1.257e-02  -1.401 0.161522    
## GarageFinishUnf     -6.589e-02  1.439e-02  -4.578 5.09e-06 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1657 on 1433 degrees of freedom
## Multiple R-squared:  0.8268, Adjusted R-squared:  0.824
## F-statistic: 297.4 on 23 and 1433 DF,  p-value: < 2.2e-16
```

```
BIC(m16.2, m11, m1)
```

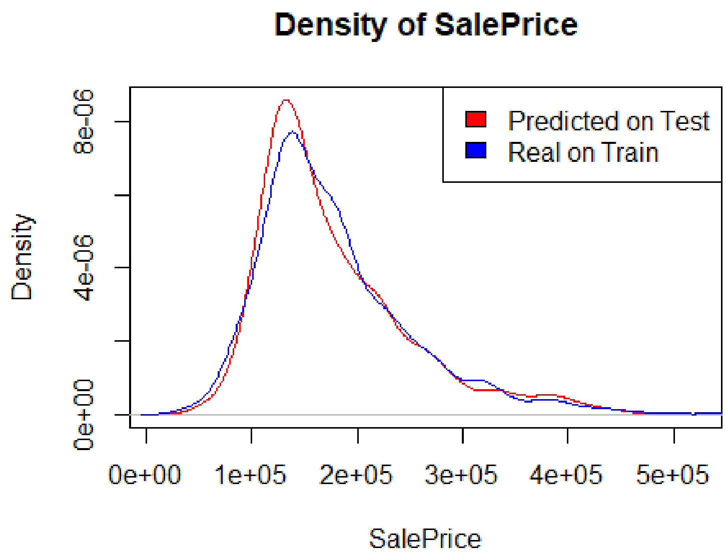
```
##        df         BIC
## m16.2 25  -945.28326
## m11   10   -26.41669
## m1    11 35747.73550
```

12. Model validation

We predict the SalePrice on the test dataset and compare its distribution with the original one in train.

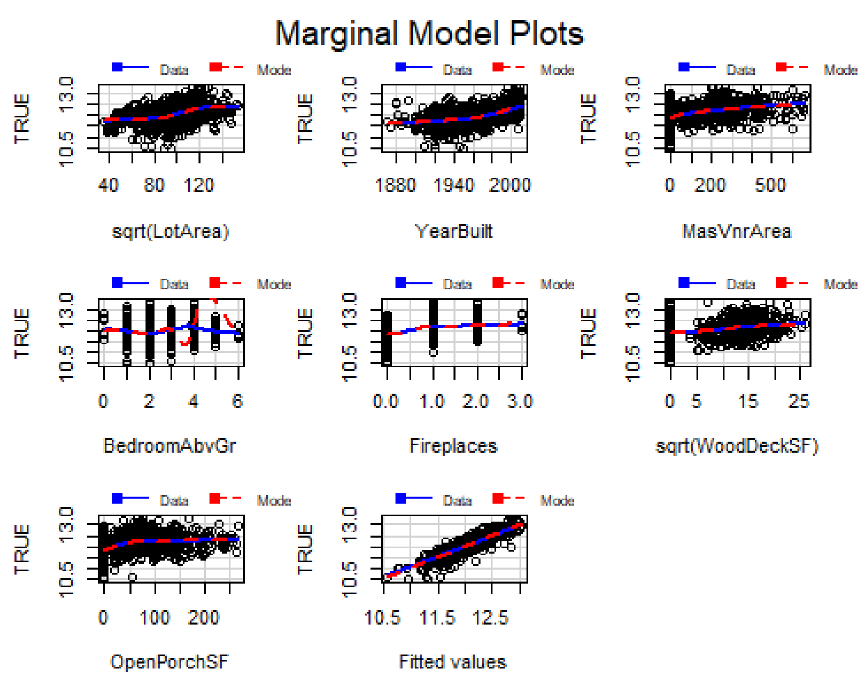
```
predicted_values = predict.lm(m16.2, df_test, se.fit=TRUE,
                              interval="prediction", level=0.95)
test_price = exp(predicted_values$fit)

par(mfrow=c(1,2))
hist(test_price[,1], main = "Predicted Sale Price on Test",
     xlab =  "Predicted test$SalePrice")
hist(df$SalePrice, main = "Sale Price on Train",
     xlab = "Real train$SalePrice")
```

```r
par(mfrow=c(1,1))
plot(density(test_price[,1]), col="red", main = "Density of SalePrice",
     xlab = "SalePrice")
lines(density(df$SalePrice), col="blue")
legend("topright",fill = c("red", "blue"), c("Predicted on Test","Real
on Train"))
```

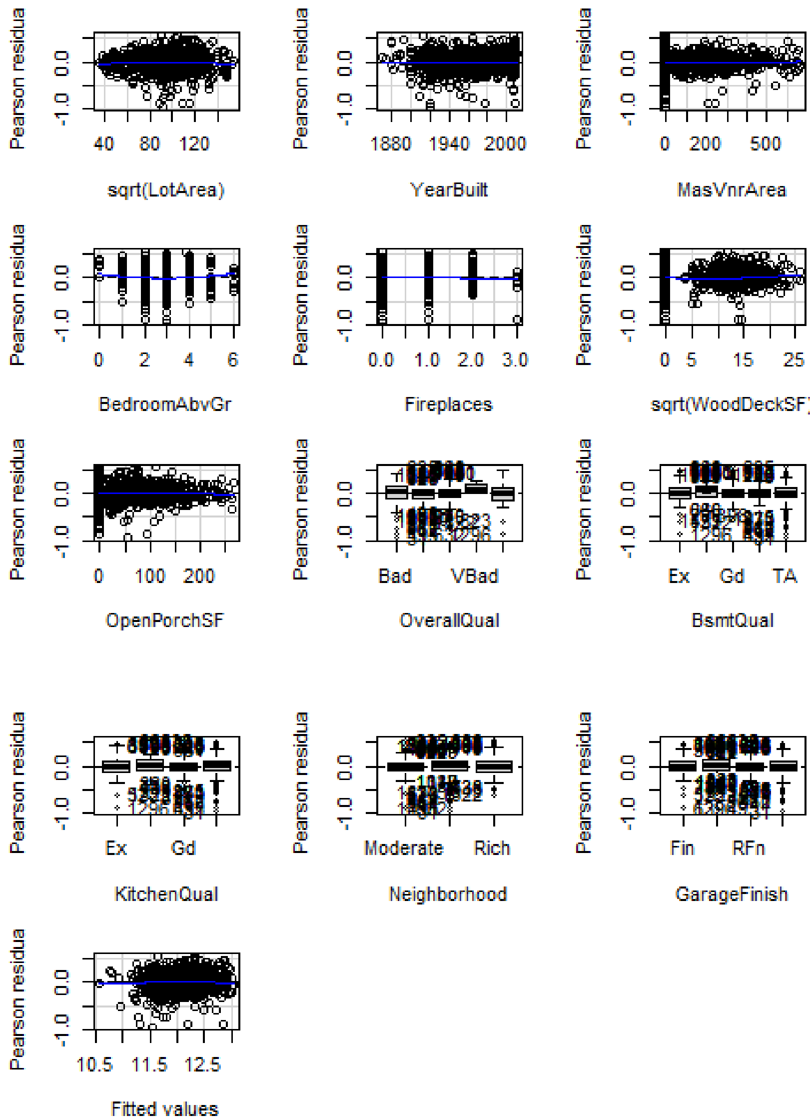


As can be seen in the previous plots, the real and the predicted distributions of SalePrice are similar, but not identical. This was exactly our goal, since both test and train come from the same population and we wanted to avoid overfitting.

```r
marginalModelPlots(m16.2, id=list(n=0))
```



```r
residualPlots( m16.2, id=list(n=0))
```




```
##                    Test stat Pr(>|Test stat|)
## sqrt(LotArea)        -2.2806         0.022718 *
## YearBuilt             0.3192         0.749596
## MasVnrArea            0.3619         0.717489
## BedroomAbvGr          2.8178         0.004902 **
## Fireplaces           -1.0510         0.293417
## sqrt(WoodDeckSF)      2.2279         0.026043 *
## OpenPorchSF          -1.0755         0.282343
## OverallQual
## BsmtQual
## KitchenQual
## Neighborhood
## GarageFinish
```

```
## Tukey test          -1.0993         0.271638
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

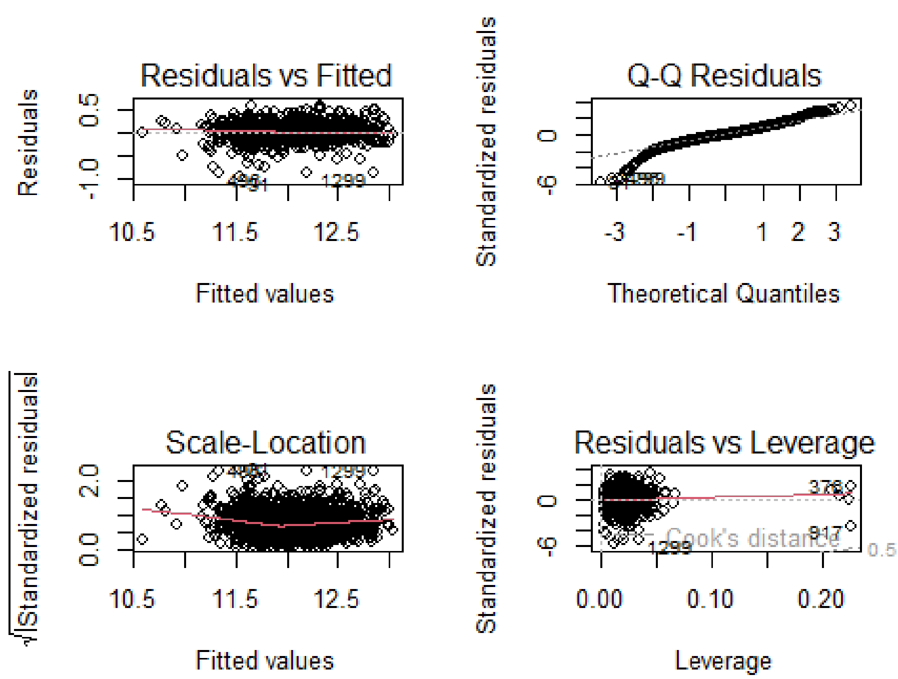
In general, using the marginal model plots, we can see that the residuals distribution for most variables are close to 0. However, sqrt(LotArea) seems to have bad residuals in marginalModelPlots(), but not in residualPlots(). This could simply mean the first method doesn't properly represent the residuals of this variable. As for categorical variables, all errors are close to 0, except for the level "VBad" of OverallQual, which is due to the fact that it contains few individuals.

```r
ks_test_result <- ks.test(test_price[,1], df$SalePrice)

ks_test_result

##
##  Asymptotic two-sample Kolmogorov-Smirnov test
##
## data:  test_price[, 1] and df$SalePrice
## D = 0.045956, p-value = 0.09198
## alternative hypothesis: two-sided
```

The Kolmogorov-Smirnov test shows that predicted and real distributions of SalePrice should be assumed to be different.

Finally, we will check the normality of the residuals.

```r
par(mfrow=c(2,2))
plot(m16.2)
```



```r
shapiro.test(m16.2$residuals)

##
##  Shapiro-Wilk normality test
##
## data:  m16.2$residuals
## W = 0.95837, p-value < 2.2e-16
```

Residuals don't follow a normal distribution, so the model won't give very accurate results. Nevertheless, we are happy with our results, so we will not apply any more changes.

13. Model interpretation

First, let us remember the model we have obtained:

$$\boldsymbol{log}(\boldsymbol{SalePrice}) \sim sqrt(LotArea) + YearBuilt + MasVnrArea + BedroomAbvGr + Fireplaces + sqrt(WoodDeckSF) + OpenPorchSF + OverallQual + BsmtQual + KitchenQual + Neighborhood + GarageFinish$$

With adjusted R^2 is **0.8195** and its BIC is about **-972**.

We are modeling the logarithm of SalePrice. That is, an increase of one unit in any of the predictors (except for LotArea and WoodDeckSF) causes the price of the sale to be multiplied by the number e. All the predictors we are using make sense intuitively: the area of the lot, the masonry veneer, the wood deck and the open porch, the amount of bedrooms above ground and fireplaces, the overall quality but also that of the basement and the kitchen, the interior finish of the garage, the dwelling neighborhood's wealth and the year it was built. The area of the lot and the wood deck appear with an exponent of 1/2 in the model, which means that the slope of their contribution to log(SalePrice) is lower than that of the other terms for values larger than 1/4.

In total, our model predictors are composed of 7 numerical features and 5 categorical variables, with three transformations and no interactions.

After the modelling of the model, we have applied it in among the test data sets to predict the sale price based on the known feature. As the SalePrice is missing, we're not able to validate the correctness of our model, but the alternative is to compare the distribution of two Sale price from the train and test data sets.

As they are coming from the same population, the density plot should be similar. As we can observe in the "Density of SalePrice" graph.