House prices: Lasso, XGBoost, and a detailed EDA

Erik Bruin

- 1 Executive Summary (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#executive-summary)
- 2 Introduction (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#introduction)
- 3 Loading and Exploring Data (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#loading-and-exploring-data)
  - 3.1 Loading libraries required and reading the data into R (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#loading-libraries-required-and-reading-the-data-into-r)
  - 3.2 Data size and structure (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#data-size-and-structure)
- 4 Exploring some of the most important variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#exploring-some-of-the-most-important-variables)
  - 4.1 The response variable; SalePrice (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#the-response-variable-saleprice)
  - 4.2 The most important numeric predictors (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#the-most-important-numeric-predictors)
    - 4.2.1 Correlations with SalePrice (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#correlations-with-saleprice)
    - 4.2.2 Overall Quality (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#overall-quality)
    - 4.2.3 Above Grade (Ground) Living Area (square feet) (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#above-grade-ground-living-area-square-feet)
- 5 Missing data, label encoding, and factorizing variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#missing-data-label-encoding-and-factorizing-variables)
  - 5.1 Completeness of the data (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#completeness-of-the-data)
  - 5.2 Imputing missing data (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#imputing-missing-data)
  - 5.3 Label encoding/factorizing the remaining character variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#label-encodingfactorizing-the-remaining-character-variables)
  - 5.4 Changing some numeric variables into factors (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#changing-some-numeric-variables-into-factors)
    - 5.4.1 Year and Month Sold (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#year-and-month-sold)
    - 5.4.2 MSSubClass (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#mssubclass)

- 6 Visualization of important variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#visualization-of-important-variables)
  - 6.1 Correlations again (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#correlations-again)
  - 6.2 Finding variable importance with a quick Random Forest (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#finding-variable-importance-with-a-quick-random-forest)
    - 6.2.1 Above Ground Living Area, and other surface related variables (in square feet) (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#above-ground-living-area-and-other-surface-related-variables-in-square-feet)
    - 6.2.2 The most important categorical variable; Neighborhood (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#the-most-important-categorical-variable-neighborhood)
    - 6.2.3 Overall Quality, and other Quality variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#overall-quality-and-other-quality-variables)
    - 6.2.4 The second most important categorical variable; MSSubClass (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#the-second-most-important-categorical-variable-mssubclass)
    - 6.2.5 Garage variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#garage-variables-1)
    - 6.2.6 Basement variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#basement-variables-1)
- 7 Feature engineering (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#feature-engineering)
  - 7.1 Total number of Bathrooms (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#total-number-of-bathrooms)
  - 7.2 Adding ‘House Age’, ‘Remodeled (Yes/No)’, and IsNew variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#adding-house-age-remodeled-yesno-and-isnew-variables)
  - 7.3 Binning Neighborhood (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#binning-neighborhood)
  - 7.4 Total Square Feet (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#total-square-feet)
  - 7.5 Consolidating Porch variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#consolidating-porch-variables)
- 8 Preparing data for modeling (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#preparing-data-for-modeling)
  - 8.1 Dropping highly correlated variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#dropping-highly-correlated-variables)
  - 8.2 Removing outliers (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#removing-outliers)
  - 8.3 PreProcessing predictor variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#preprocessing-predictor-variables)

        - 8.3.1 Skewness and normalizing of the numeric predictors (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#skewness-and-normalizing-of-the-numeric-predictors)
        - 8.3.2 One hot encoding the categorical variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#one-hot-encoding-the-categorical-variables)
        - 8.3.3 Removing levels with few or no observations in train or test (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#removing-levels-with-few-or-no-observations-in-train-or-test)
    - 8.4 Dealing with skewness of response variable (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#dealing-with-skewness-of-response-variable)
    - 8.5 Composing train and test sets (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#composing-train-and-test-sets)
- 9 Modeling (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#modeling)
    - 9.1 Lasso regression model (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#lasso-regression-model)
    - 9.2 XGBoost model (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#xgboost-model)
    - 9.3 Averaging predictions (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#averaging-predictions)

1 Executive Summary

I started this competition by just focusing on getting a good understanding of the dataset. The EDA is detailed and many visualizations are included. This version also includes modeling.

- Lasso regressions performs best with a cross validation RMSE-score of 0.1121. Given the fact that there is a lot of multicolinearity among the variables, this was expected. Lasso does not select a substantial number of the available variables in its model, as it is supposed to do.
- The XGBoost model also performs very well with a cross validation RMSE of 0.1162.
- As those two algorithms are very different, averaging predictions is likely to improve the predictions. As the Lasso cross validated RMSE is better than XGBoost's CV score, I decided to weight the Lasso results double.

2 Introduction

Kaggle describes this competition as follows (https://www.kaggle.com/c/house-prices-advanced-regression-techniques):

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home.

3 Loading and Exploring Data

3.1 Loading libraries required and reading the data into R

Loading R packages used besides base R.

```
library(knitr)
library(ggplot2)
library(plyr)
library(dplyr)
library(corrplot)
library(caret)
library(gridExtra)
library(scales)
library(Rmisc)
library(ggrepel)
library(randomForest)
library(psych)
library(xgboost)
```

Below, I am reading the csv's as dataframes into R.

```
train <- read.csv("../input/train.csv", stringsAsFactors = F)
test <- read.csv("../input/test.csv", stringsAsFactors = F)
```

3.2 Data size and structure

The train dataset consist of character and integer variables. Most of the character variables are actually (ordinal) factors, but I chose to read them into R as character strings as most of them require cleaning and/or feature engineering first. In total, there are 81 columns/variables, of which the last one is the response variable (SalePrice). Below, I am displaying only a glimpse of the variables. All of them are discussed in more detail throughout the document.

```
dim(train)
```

```
## [1] 1460   81
```

```
str(train[,c(1:10, 81)]) #display first 10 variables and the response variable
```

```
## 'data.frame':    1460 obs. of  11 variables:
##  $ Id         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning   : chr  "RL" "RL" "RL" "RL" ...
##  $ LotFrontage: int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea    : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420
...
##  $ Street     : chr  "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley      : chr  NA NA NA NA ...
##  $ LotShape   : chr  "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour: chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities  : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ SalePrice  : int  208500 181500 223500 140000 250000 143000 307000 20000
0 129900 118000 ...
```

```
#Getting rid of the IDs but keeping the test IDs in a vector. These are needed t
o compose the submission file
test_labels <- test$Id
test$Id <- NULL
train$Id <- NULL
```

```
test$SalePrice <- NA
all <- rbind(train, test)
dim(all)
```

```
## [1] 2919   80
```

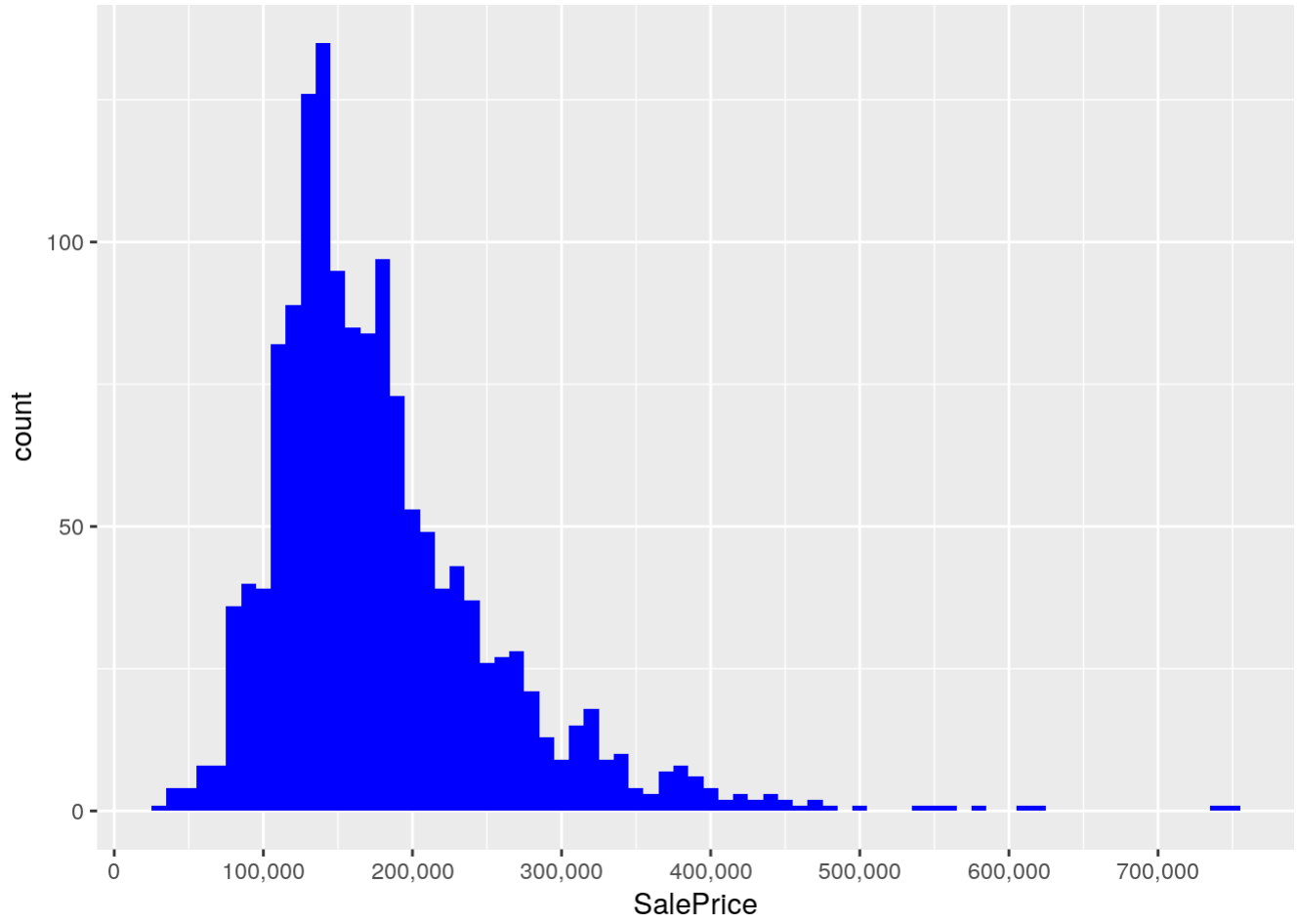
Without the Id's, the dataframe consists of 79 predictors and our response variable SalePrice.

4 Exploring some of the most important variables

4.1 The response variable; SalePrice

As you can see, the sale prices are right skewed. This was expected as few people can afford very expensive houses. I will keep this in mind, and take measures before modeling.

```
ggplot(data=all[!is.na(all$SalePrice),], aes(x=SalePrice)) +
        geom_histogram(fill="blue", binwidth = 10000) +
        scale_x_continuous(breaks= seq(0, 800000, by=100000), labels = comma)
```




```
summary(all$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   34900  129975  163000  180921  214000  755000    1459
```

4.2 The most important numeric predictors

The character variables need some work before I can use them. To get a feel for the dataset, I decided to first see which numeric variables have a high correlation with the SalePrice.

4.2.1 Correlations with SalePrice

Altogether, there are 10 numeric variables with a correlation of at least 0.5 with SalePrice. All those correlations are positive.

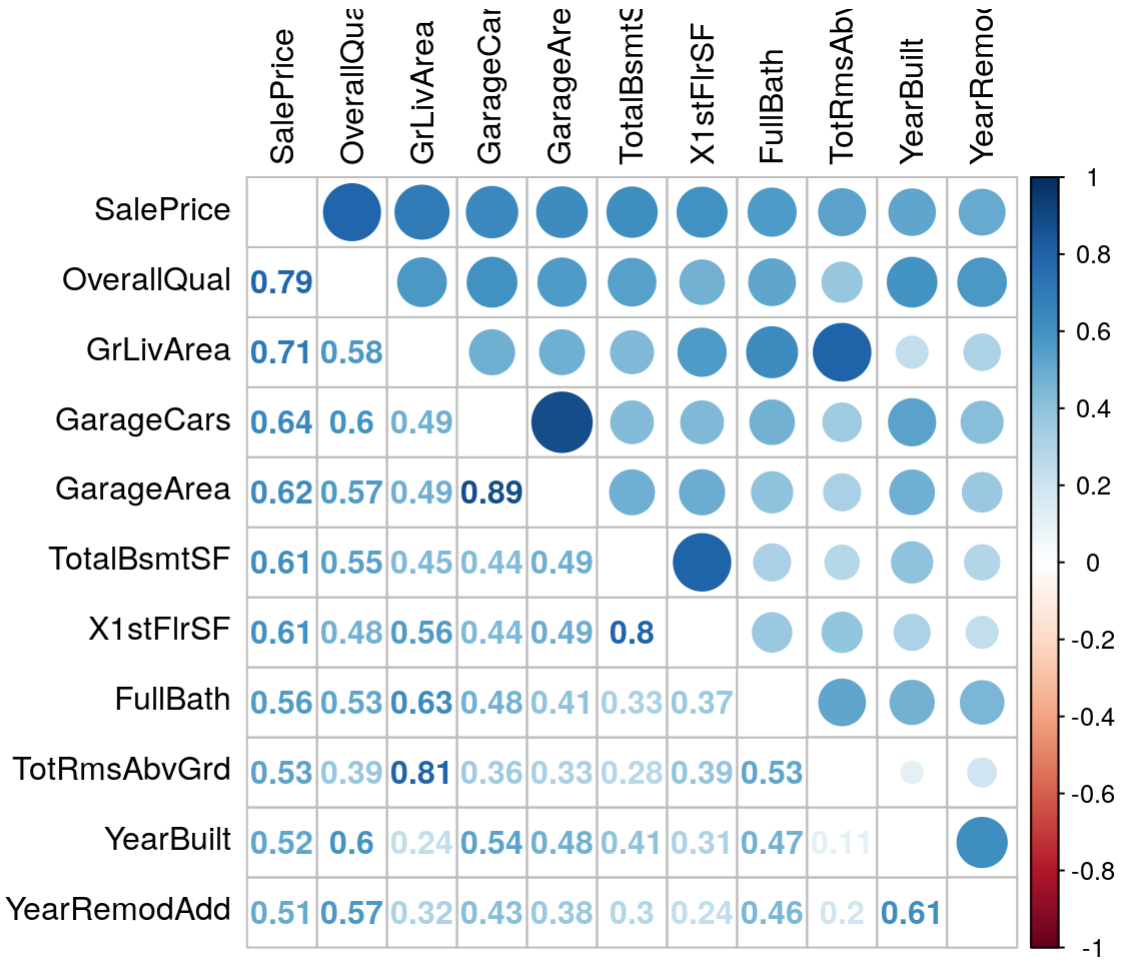
```
numericVars <- which(sapply(all, is.numeric)) #index vector numeric variables
numericVarNames <- names(numericVars) #saving names vector for use later on
cat('There are', length(numericVars), 'numeric variables')
```

```
## There are 37 numeric variables
```

```
all_numVar <- all[, numericVars]
cor_numVar <- cor(all_numVar, use="pairwise.complete.obs") #correlations of all numeric variables

#sort on decreasing correlations with SalePrice
cor_sorted <- as.matrix(sort(cor_numVar[,'SalePrice'], decreasing = TRUE))
 #select only high corelations
CorHigh <- names(which(apply(cor_sorted, 1, function(x) abs(x)>0.5)))
cor_numVar <- cor_numVar[CorHigh, CorHigh]

corrplot.mixed(cor_numVar, tl.col="black", tl.pos = "lt")
```



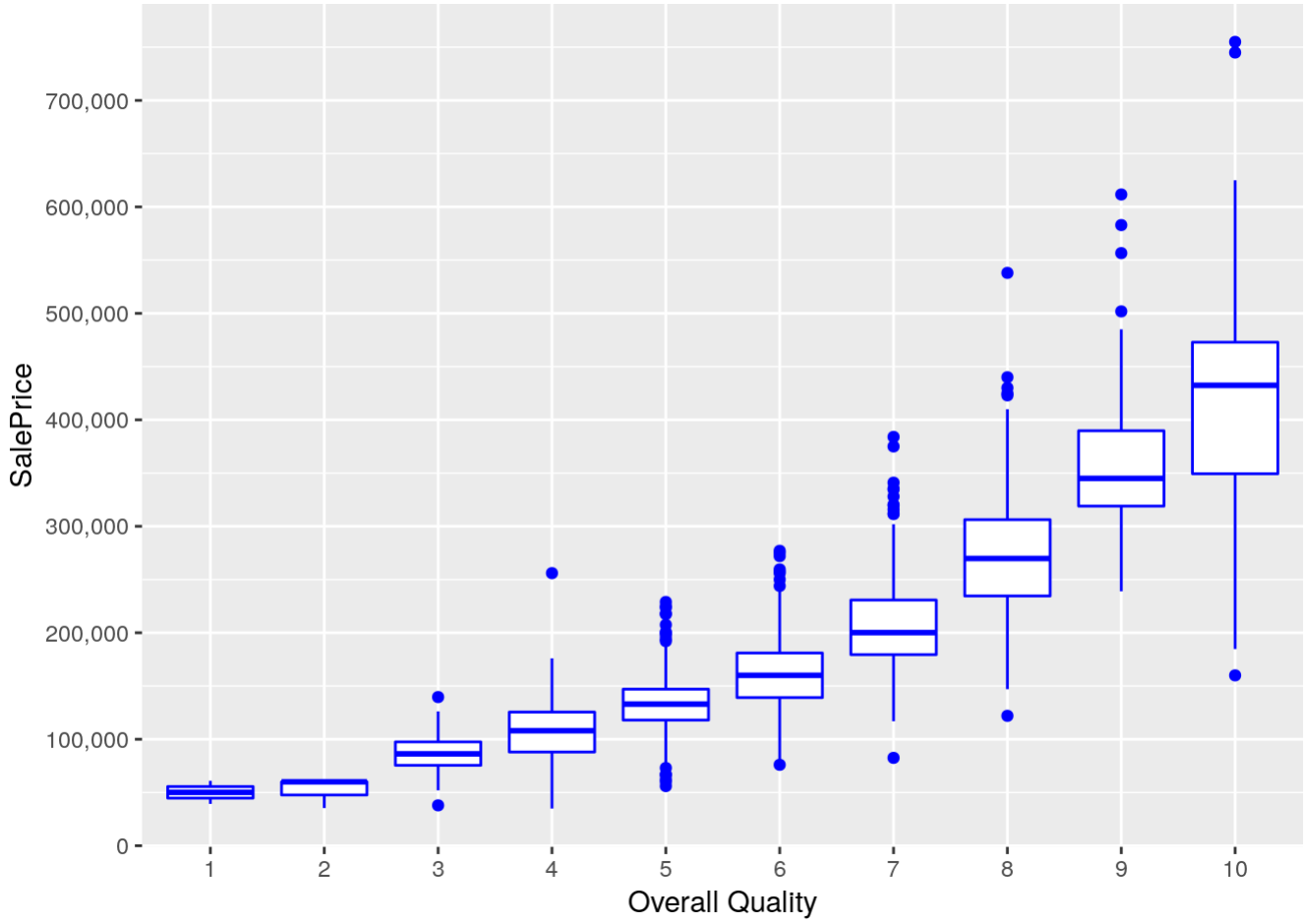
In the remainder of this section, I will visualize the relation between SalePrice and the two predictors with the highest correlation with SalePrice; Overall Quality and the 'Above Grade' Living Area (this is the proportion of the house that is not in a basement; link (http://www.gimme-shelter.com/above-grade-50066/)).

It also becomes clear the multicollinearity is an issue. For example: the correlation between GarageCars and GarageArea is very high (0.89), and both have similar (high) correlations with SalePrice. The other 6 six variables with a correlation higher than 0.5 with SalePrice are: -TotalBsmtSF: Total square feet of basement area -1stFlrSF: First Floor square feet -FullBath: Full bathrooms above grade -TotRmsAbvGrd: Total rooms above grade (does not include bathrooms) -YearBuilt: Original construction date -YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

4.2.2 Overall Quality

Overall Quality has the highest correlation with SalePrice among the numeric variables (0.79). It rates the overall material and finish of the house on a scale from 1 (very poor) to 10 (very excellent).

```
ggplot(data=all[!is.na(all$SalePrice),], aes(x=factor(OverallQual), y=SalePric
e))+
        geom_boxplot(col='blue') + labs(x='Overall Quality') +
        scale_y_continuous(breaks= seq(0, 800000, by=100000), labels = comma)
```



The positive correlation is certainly there indeed, and seems to be a slightly upward curve. Regarding outliers, I do not see any extreme values. If there is a candidate to take out as an outlier later on, it seems to be the expensive house with grade 4.

4.2.3 Above Grade (Ground) Living Area (square feet)

The numeric variable with the second highest correlation with SalesPrice is the Above Grade Living Area. This make a lot of sense; big houses are generally more expensive.

```
ggplot(data=all[!is.na(all$SalePrice),], aes(x=GrLivArea, y=SalePrice))+
        geom_point(col='blue') + geom_smooth(method = "lm", se=FALSE, color="b
lack", aes(group=1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=100000), labels = comma)
 +
        geom_text_repel(aes(label = ifelse(all$GrLivArea[!is.na(all$SalePric
e)]>4500, rownames(all), '')))
```

Especially the two houses with really big living areas and low SalePrices seem outliers (houses 524 and 1299, see labels in graph). I will not take them out yet, as taking outliers can be dangerous. For instance, a low score on the Overall Quality could explain a low price. However, as you can see below, these two houses actually also score maximum points on Overall Quality. Therefore, I will keep houses 1299 and 524 in mind as prime candidates to take out as outliers.

```
all[c(524, 1299), c('SalePrice', 'GrLivArea', 'OverallQual')]
```

```
##      SalePrice GrLivArea OverallQual
## 524     184750      4676          10
## 1299    160000      5642          10
```

5 Missing data, label encoding, and factorizing variables

5.1 Completeness of the data

First of all, I would like to see which variables contain missing values.

```
NAcol <- which(colSums(is.na(all)) > 0)
sort(colSums(sapply(all[NAcol], is.na)), decreasing = TRUE)
```

```
##       PoolQC  MiscFeature        Alley        Fence    SalePrice
##         2909         2814         2721         2348         1459
##  FireplaceQu  LotFrontage  GarageYrBlt GarageFinish   GarageQual
##         1420          486          159          159          159
##   GarageCond   GarageType     BsmtCond BsmtExposure     BsmtQual
##          159          157           82           82           81
## BsmtFinType2 BsmtFinType1   MasVnrType   MasVnrArea     MSZoning
##           80           79           24           23            4
##    Utilities BsmtFullBath BsmtHalfBath   Functional  Exterior1st
##            2            2            2            2            1
##  Exterior2nd   BsmtFinSF1   BsmtFinSF2    BsmtUnfSF  TotalBsmtSF
##            1            1            1            1            1
##   Electrical  KitchenQual   GarageCars   GarageArea     SaleType
##            1            1            1            1            1
```

```
cat('There are', length(NAcol), 'columns with missing values')
```

```
## There are 35 columns with missing values
```

Of course, the 1459 NAs in SalePrice match the size of the test set perfectly. This means that I have to fix NAs in 34 predictor variables.

5.2 Imputing missing data

In this section, I am going to fix the 34 predictors that contains missing values. I will go through them working my way down from most NAs until I have fixed them all. If I stumble upon a variable that actually forms a group with other variables, I will also deal with them as a group. For instance, there are multiple variables that relate to Pool, Garage, and Basement.

As I want to keep the document as readable as possible, I decided to use the “Tabs” option that knitr provides. You can find a short analysis for each (group of) variables under each Tab. You don’t have to go through all sections, and can also just have a look at a few tabs. If you do so, I think that especially the Garage and Basement sections are worthwhile, as I have been carefull in determing which houses really do not have a basement or garage.

Besides making sure that the NAs are taken care off, I have also converted character variables into ordinal integers if there is clear ordinality, or into factors if levels are categories without ordinality. I will convert these factors into numeric later on by using one-hot encoding (using the model.matrix function).

5.2.1 Pool variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#pool-variables)

5.2.2 Miscellaneous Feature (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#miscellaneous-feature)

5.2.3 Alley (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#alley)

5.2.4 Fence (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#fence)

5.2.5 Fireplace variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#fireplace-variables)

5.2.6 Lot variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#lot-variables)

5.2.7 Garage variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#garage-variables)

5.2.8 Basement Variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#basement-variables)

5.2.9 Masonry variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#masonry-variables)

5.2.10 MS Zoning (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#ms-zoning)

5.2.11 Kitchen variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#kitchen-variables)

5.2.12 Utilities (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#utilities)

5.2.13 Home functionality (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#home-functionality)

5.2.14 Exterior variables (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#exterior-variables)

5.2.15 Electrical system (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#electrical-system)

5.2.16 Sale Type and Condition (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#sale-type-and-condition)

Pool Quality and the PoolArea variable

The PoolQC is the variable with most NAs. The description is as follows:

PoolQC: Pool quality

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
NA   No Pool
```

So, it is obvious that I need to just assign ‘No Pool’ to the NAs. Also, the high number of NAs makes sense as normally only a small proportion of houses have a pool.

```
all$PoolQC[is.na(all$PoolQC)] <- 'None'
```

It is also clear that I can label encode this variable as the values are ordinal. As there a multiple variables that use the same quality levels, I am going to create a vector that I can reuse later on.

```
Qualities <- c('None' = 0, 'Po' = 1, 'Fa' = 2, 'TA' = 3, 'Gd' = 4, 'Ex' = 5)
```

Now, I can use the function ‘revalue’ to do the work for me.

```
all$PoolQC<-as.integer(revalue(all$PoolQC, Qualities))
table(all$PoolQC)
```

```
##
##    0    2    4    5
## 2909    2    4    4
```

However, there is a second variable that relates to Pools. This is the PoolArea variable (in square feet). As you can see below, there are 3 houses without PoolQC. First, I checked if there was a clear relation between the PoolArea and the PoolQC. As I did not see a clear relation (bigger of smaller pools with better PoolQC), I am going to impute PoolQC values based on the Overall Quality of the houses (which is not very high for those 3 houses).

```
all[all$PoolArea>0 & all$PoolQC==0, c('PoolArea', 'PoolQC', 'OverallQual')]
```

```
##      PoolArea PoolQC OverallQual
## 2421      368      0           4
## 2504      444      0           6
## 2600      561      0           3
```

```
all$PoolQC[2421] <- 2
all$PoolQC[2504] <- 3
all$PoolQC[2600] <- 2
```

Please return to the 5.2 Tabs menu to select other (groups of) variables

5.3 Label encoding/factorizing the remaining character variables

At this point, I have made sure that all variables with NAs are taken care of. However, I still need to also take care of the remaining character variables that without missing values. Similar to the previous section, I have created Tabs for groups of variables.

```
Charcol <- names(all[,sapply(all, is.character)])
Charcol
```

```
##  [1] "Street"       "LandContour"  "LandSlope"    "Neighborhood"
##  [5] "Condition1"   "Condition2"   "BldgType"     "HouseStyle"
##  [9] "RoofStyle"    "RoofMatl"     "Foundation"   "Heating"
## [13] "HeatingQC"    "CentralAir"   "PavedDrive"
```

```
cat('There are', length(Charcol), 'remaining columns with character values')
```

```
## There are 15 remaining columns with character values
```

5.3.1 Foundation (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#foundation)

5.3.2 Heating and airco (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#heating-and-airco)

5.3.3 Roof (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#roof)

5.3.4 Land (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#land)

5.3.5 Dwelling (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#dwelling)

5.3.6 Neighborhood and Conditions (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#neighborhood-and-conditions)

5.3.7 Pavement of Street & Driveway (https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda#pavement-of-street-driveway)

Foundation: Type of foundation

BrkTil	Brick & Tile
CBlock	Cinder Block
PConc	Poured Contrete
Slab	Slab
Stone	Stone
Wood	Wood

```
#No ordinality, so converting into factors
all$Foundation <- as.factor(all$Foundation)
table(all$Foundation)
```

```
##
## BrkTil CBlock  PConc   Slab  Stone   Wood
##    311   1235   1308     49     11      5
```

```
sum(table(all$Foundation))
```

```
## [1] 2919
```

Please return to the 5.3 Tabs menu to select other (groups of) variables

5.4 Changing some numeric variables into factors

At this point, all variables are complete (No NAs), and all character variables are converted into either numeric labels of into factors. However, there are 3 variables that are recorded numeric but should actually be categorical.

5.4.1 Year and Month Sold

While oridinality within YearBuilt (or remodeled) makes sense (old houses are worth less), we are talking about only 5 years of sales. These years also include an economic crisis. For instance: Sale Prices in 2009 (after the collapse) are very likely to be much lower than in 2007. I wil convert YrSold into a factor before modeling, but as I need the numeric version of YrSold to create an Age variable, I am not doing that yet.

Month Sold is also an Integer variable. However, December is not “better” than January. Therefore, I will convert MoSold values back into factors.

```
str(all$YrSold)
```

```
##  int [1:2919] 2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
```

```
str(all$MoSold)
```

```
##  int [1:2919] 2 5 9 2 12 10 8 11 4 1 ...
```

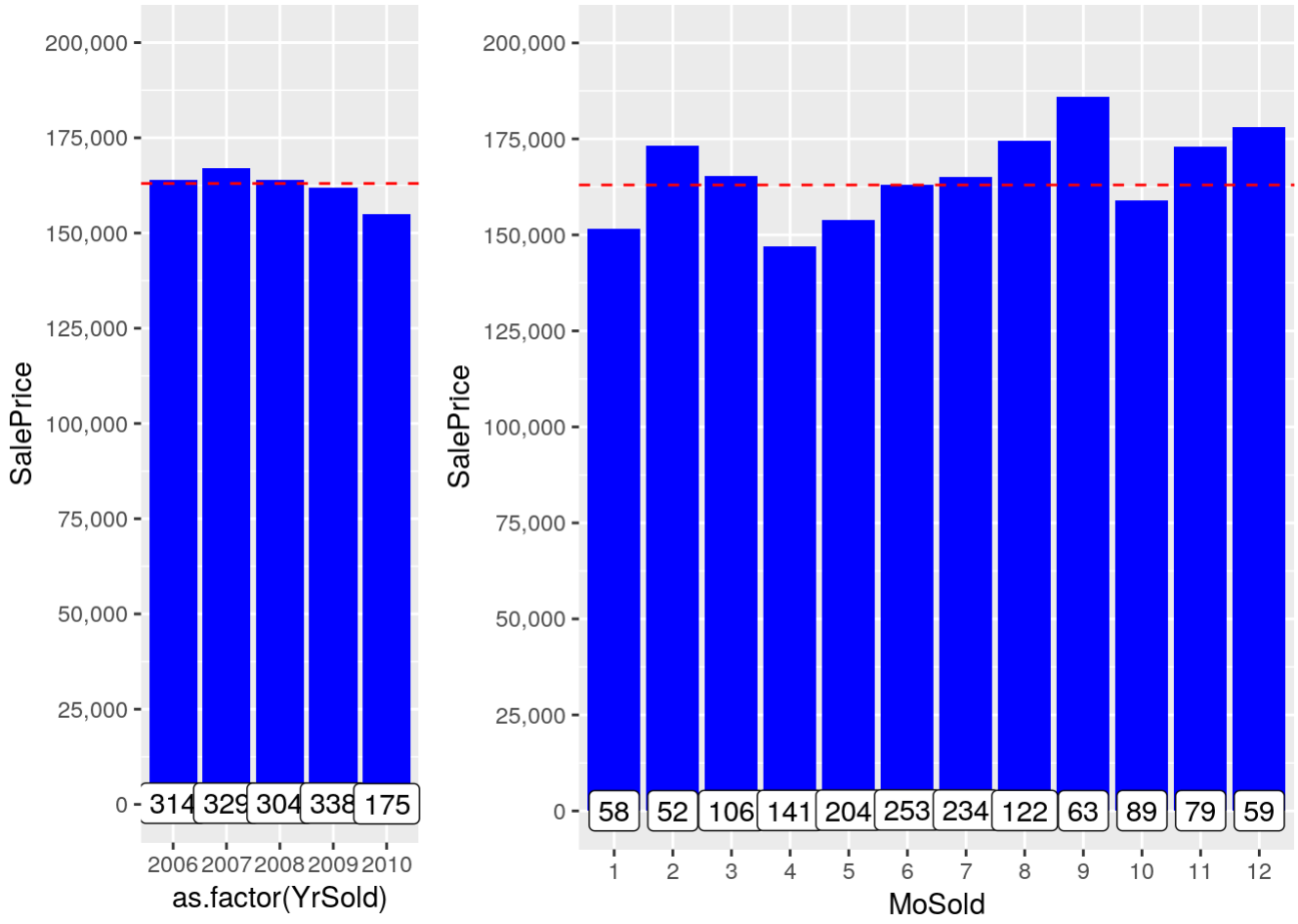
```
all$MoSold <- as.factor(all$MoSold)
```

Although possible a bit less steep than expected, the effects of the Banking crises that took place at the end of 2007 can be seen indeed. After the highest median prices in 2007, the prices gradually decreased. However, seasonality seems to play a bigger role, as you can see below.

```
ys <- ggplot(all[!is.na(all$SalePrice),], aes(x=as.factor(YrSold), y=SalePric
e)) +
        geom_bar(stat='summary', fun.y = "median", fill='blue')+
        scale_y_continuous(breaks= seq(0, 800000, by=25000), labels = comma) +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..)) +
        coord_cartesian(ylim = c(0, 200000)) +
        geom_hline(yintercept=163000, linetype="dashed", color = "red") #dashe
d line is median SalePrice

ms <- ggplot(all[!is.na(all$SalePrice),], aes(x=MoSold, y=SalePrice)) +
        geom_bar(stat='summary', fun.y = "median", fill='blue')+
        scale_y_continuous(breaks= seq(0, 800000, by=25000), labels = comma) +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..)) +
        coord_cartesian(ylim = c(0, 200000)) +
        geom_hline(yintercept=163000, linetype="dashed", color = "red") #dashe
d line is median SalePrice

grid.arrange(ys, ms, widths=c(1,2))
```



5.4.2 MSSubClass

MSSubClass: Identifies the type of dwelling involved in the sale.

```
    20  1-STORY 1946 & NEWER ALL STYLES
    30  1-STORY 1945 & OLDER
    40  1-STORY W/FINISHED ATTIC ALL AGES
    45  1-1/2 STORY - UNFINISHED ALL AGES
    50  1-1/2 STORY FINISHED ALL AGES
    60  2-STORY 1946 & NEWER
    70  2-STORY 1945 & OLDER
    75  2-1/2 STORY ALL AGES
    80  SPLIT OR MULTI-LEVEL
    85  SPLIT FOYER
    90  DUPLEX - ALL STYLES AND AGES
   120  1-STORY PUD (Planned Unit Development) - 1946 & NEWER
   150  1-1/2 STORY PUD - ALL AGES
   160  2-STORY PUD - 1946 & NEWER
   180  PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
   190  2 FAMILY CONVERSION - ALL STYLES AND AGES
```

These classes are coded as numbers, but really are categories.

```
str(all$MSSubClass)
```

```
##  int [1:2919] 60 20 60 70 60 50 20 60 50 190 ...
```

```
all$MSSubClass <- as.factor(all$MSSubClass)

#revalue for better readability
all$MSSubClass<-revalue(all$MSSubClass, c('20'='1 story 1946+', '30'='1 story
 1945-', '40'='1 story unf attic', '45'='1,5 story unf', '50'='1,5 story fin',
'60'='2 story 1946+', '70'='2 story 1945-', '75'='2,5 story all ages', '80'='s
plit/multi level', '85'='split foyer', '90'='duplex all style/age', '120'='1 s
tory PUD 1946+', '150'='1,5 story PUD all', '160'='2 story PUD 1946+', '180'=
'PUD multilevel', '190'='2 family conversion'))

str(all$MSSubClass)
```

```
##  Factor w/ 16 levels "1 story 1946+",..: 6 1 6 7 6 5 1 6 5 16 ...
```

6 Visualization of important variables

I have now finally reached the point where all character variables have been converted into categorical factors or have been label encoded into numbers. In addition, 3 numeric variables have been converted into factors, and I deleted one variable (Utilities). As you can see below, the number of numerical variables is now 56 (including the response variable), and the remaining 23 variables are categorical.

```
numericVars <- which(sapply(all, is.numeric)) #index vector numeric variables
factorVars <- which(sapply(all, is.factor)) #index vector factor variables
cat('There are', length(numericVars), 'numeric variables, and', length(factorV
ars), 'categoric variables')
```

```
## There are 56 numeric variables, and 23 categoric variables
```

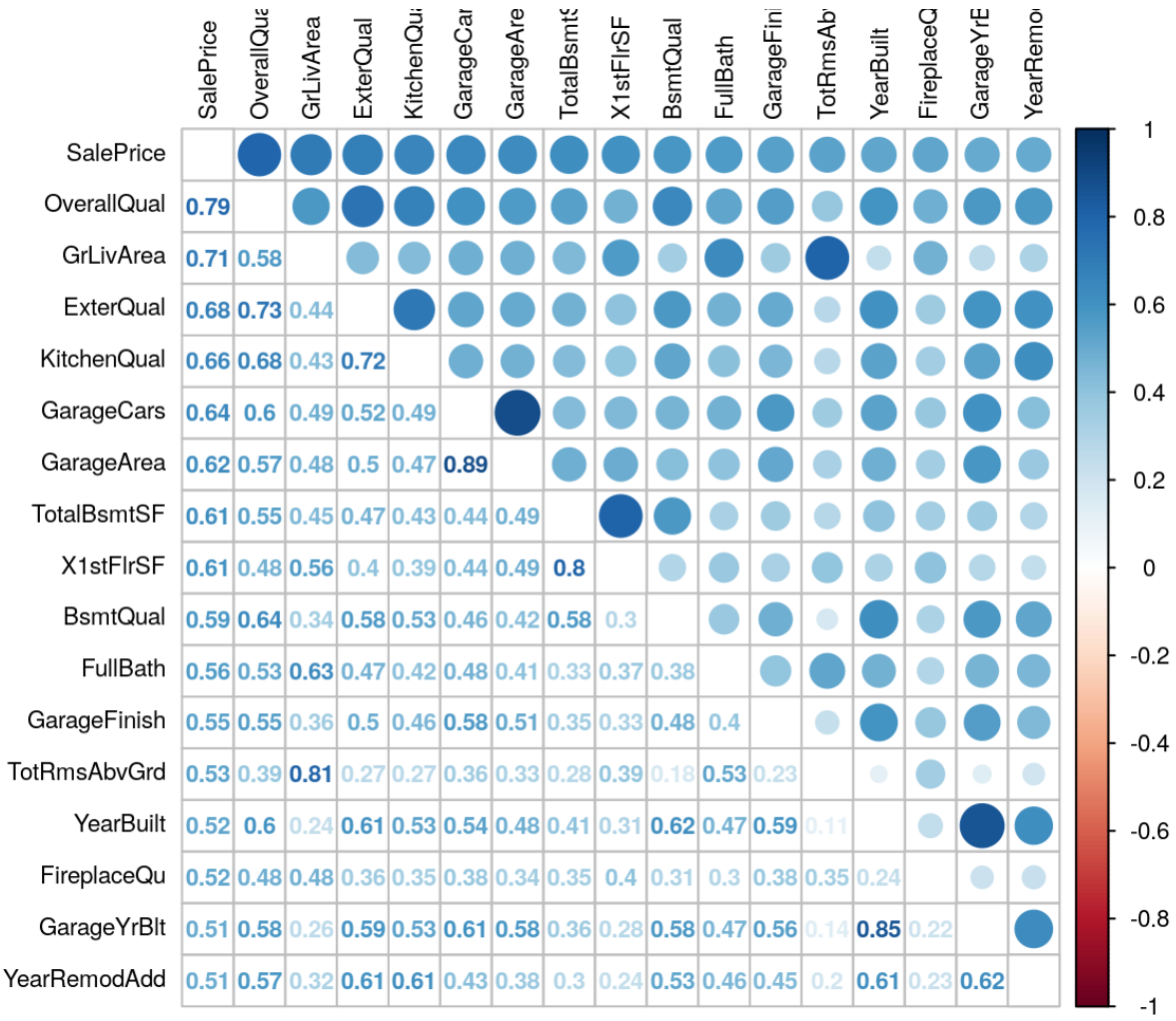
6.1 Correlations again

Below I am checking the correlations again. As you can see, the number of variables with a correlation of at least 0.5 with the SalePrice has increased from 10 (see section 4.2.1) to 16.

```
all_numVar <- all[, numericVars]
cor_numVar <- cor(all_numVar, use="pairwise.complete.obs") #correlations of all
numeric variables

#sort on decreasing correlations with SalePrice
cor_sorted <- as.matrix(sort(cor_numVar[,'SalePrice'], decreasing = TRUE))
 #select only high corelations
CorHigh <- names(which(apply(cor_sorted, 1, function(x) abs(x)>0.5)))
cor_numVar <- cor_numVar[CorHigh, CorHigh]

corrplot.mixed(cor_numVar, tl.col="black", tl.pos = "lt", tl.cex = 0.7,cl.cex
 = .7, number.cex=.7)
```



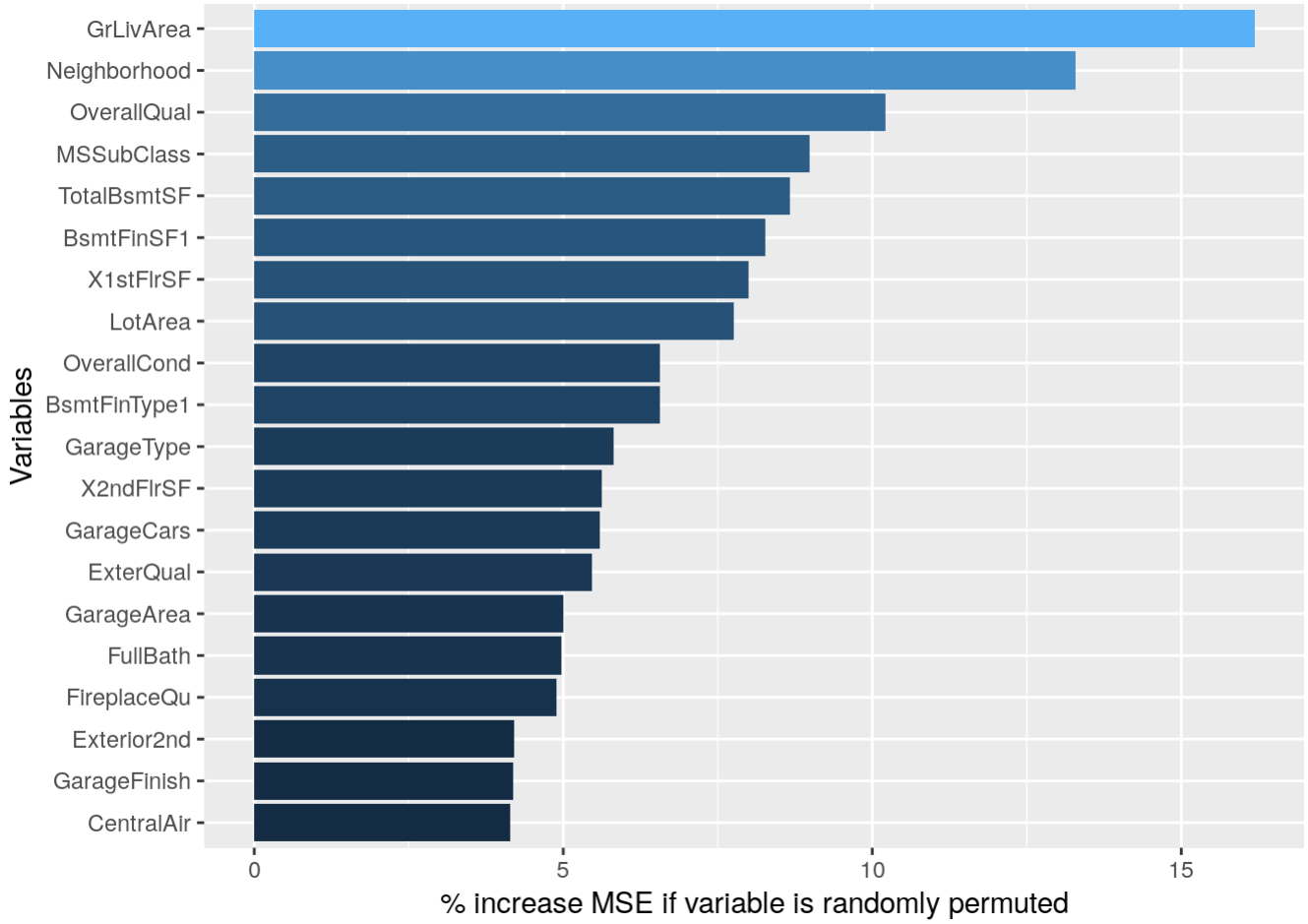
6.2 Finding variable importance with a quick Random Forest

Although the correlations are giving a good overview of the most important numeric variables and multicolinerity among those variables, I wanted to get an overview of the most important variables including the categorical variables before moving on to visualization.

I tried to get the relative importance of variables with a quick linear regression model with the calc.relimp function of package , and also tried the boruta function of package boruta which separates the variables into groups that are important or not. However, these method took a long time. As I only want to get an indication of the variable importance, I eventually decided to keep it simple and just use a quick and dirty Random Forest model with only 100 trees. This also does the job for me, and does not take very long as I can specify a (relatively) small number of trees.

```
set.seed(2018)
quick_RF <- randomForest(x=all[1:1460,-79], y=all$SalePrice[1:1460], ntree=100
,importance=TRUE)
imp_RF <- importance(quick_RF)
imp_DF <- data.frame(Variables = row.names(imp_RF), MSE = imp_RF[,1])
imp_DF <- imp_DF[order(imp_DF$MSE, decreasing = TRUE),]

ggplot(imp_DF[1:20,], aes(x=reorder(Variables, MSE), y=MSE, fill=MSE)) + geom_
bar(stat = 'identity') + labs(x = 'Variables', y= '% increase MSE if variable
 is randomly permuted') + coord_flip() + theme(legend.position="none")
```



Only 3 of those most important variables are categorical according to RF; Neighborhood, MSSubClass, and GarageType.

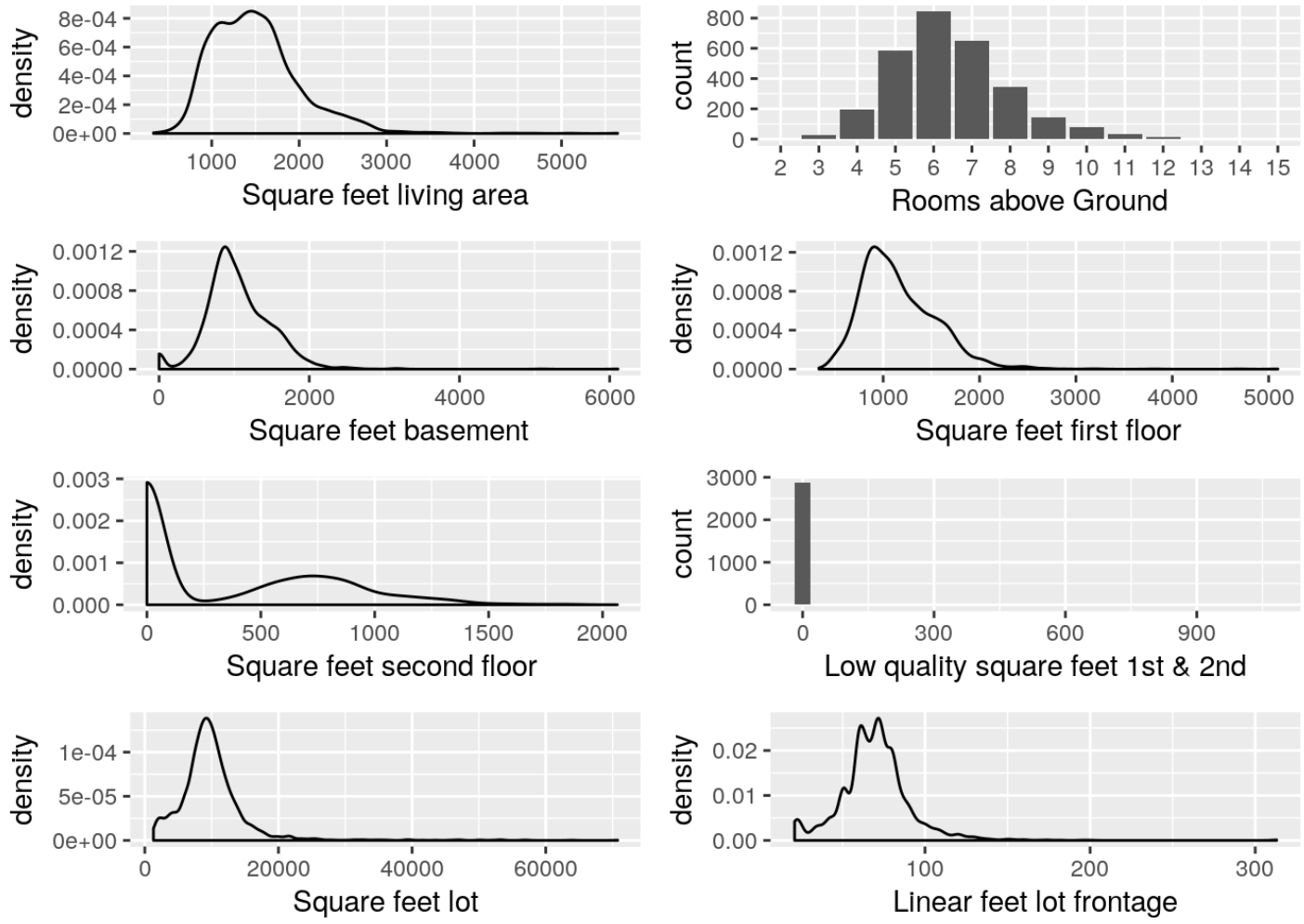
6.2.1 Above Ground Living Area, and other surface related variables (in square feet)

As I have already visualized the relation between the Above Ground Living Area and SalePrice in my initial explorations, I will now just display the distribution itself. As there are more 'square feet' surface measurements in the Top 20, I am taking the opportunity to bundle them in this section. Note: GarageArea is taken care of in the Garage variables section.

I am also adding 'Total Rooms Above Ground' (TotRmsAbvGrd) as this variable is highly correlated with the Above Ground Living Area(0.81).

```
s1 <- ggplot(data= all, aes(x=GrLivArea)) +
        geom_density() + labs(x='Square feet living area')
s2 <- ggplot(data=all, aes(x=as.factor(TotRmsAbvGrd))) +
        geom_histogram(stat='count') + labs(x='Rooms above Ground')
s3 <- ggplot(data= all, aes(x=X1stFlrSF)) +
        geom_density() + labs(x='Square feet first floor')
s4 <- ggplot(data= all, aes(x=X2ndFlrSF)) +
        geom_density() + labs(x='Square feet second floor')
s5 <- ggplot(data= all, aes(x=TotalBsmtSF)) +
        geom_density() + labs(x='Square feet basement')
s6 <- ggplot(data= all[all$LotArea<100000,], aes(x=LotArea)) +
        geom_density() + labs(x='Square feet lot')
s7 <- ggplot(data= all, aes(x=LotFrontage)) +
        geom_density() + labs(x='Linear feet lot frontage')
s8 <- ggplot(data= all, aes(x=LowQualFinSF)) +
        geom_histogram() + labs(x='Low quality square feet 1st & 2nd')

layout <- matrix(c(1,2,5,3,4,8,6,7),4,2,byrow=TRUE)
multiplot(s1, s2, s3, s4, s5, s6, s7, s8, layout=layout)
```



I will investigate several of these variables for outliers later on. For the lot visualization, I have already taken out the lots above 100,000 square feet (4 houses).

GrLivArea seemed to be just the total of square feet 1st and 2nd floor. However, in a later version, I discovered that there is also a variable called: LowQualFinSF: Low quality finished square feet (all floors). As you can see above (Low quality square feet 1st and 2nd) almost all houses have none of this (only 40 houses do have some). It turns out that these square feet are actually included in the GrLivArea. The correlation between those 3 variables and GrLivArea is exactely 1.

```
cor(all$GrLivArea, (all$X1stFlrSF + all$X2ndFlrSF + all$LowQualFinSF))
```

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

```
head(all[all$LowQualFinSF>0, c('GrLivArea', 'X1stFlrSF', 'X2ndFlrSF', 'LowQual
FinSF')])
```

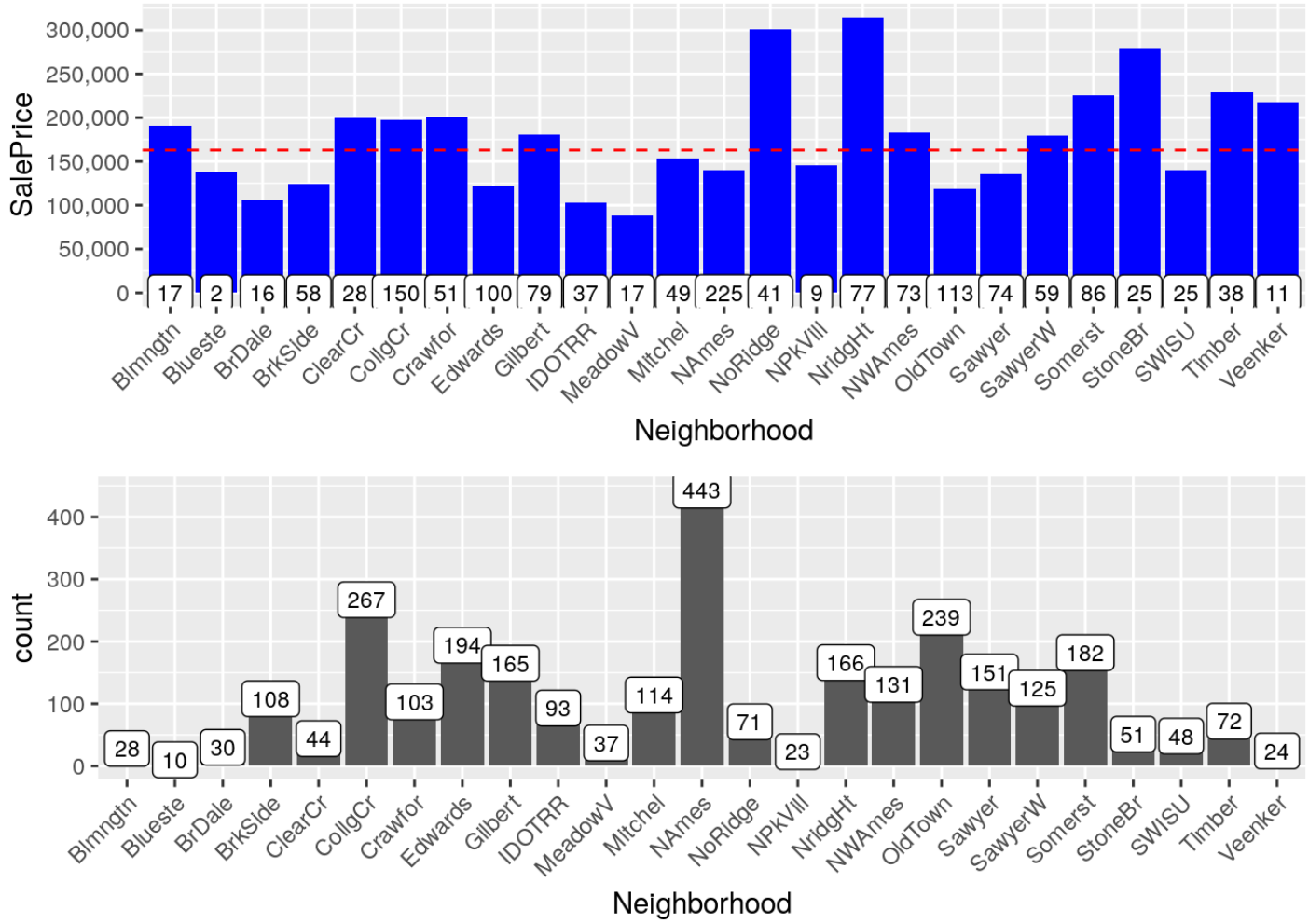
```
##     GrLivArea X1stFlrSF X2ndFlrSF LowQualFinSF
## 52       1176       816         0          360
## 89       1526      1013         0          513
## 126       754       520         0          234
## 171      1382       854         0          528
## 186      3608      1518      1518          572
## 188      1656       808       704          144
```

6.2.2 The most important categorical variable; Neighborhood

Th first graph shows the median SalePrice by Neighorhood. The frequency (number of houses) of each Neighborhood in the train set is shown in the labels.

The second graph below shows the frequencies across all data.

```
n1 <- ggplot(all[!is.na(all$SalePrice),], aes(x=Neighborhood, y=SalePrice)) +
        geom_bar(stat='summary', fun.y = "median", fill='blue') +
        theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=50000), labels = comma) +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size
=3) +
        geom_hline(yintercept=163000, linetype="dashed", color = "red") #dashe
d line is median SalePrice
n2 <- ggplot(data=all, aes(x=Neighborhood)) +
        geom_histogram(stat='count')+
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size
=3)+
        theme(axis.text.x = element_text(angle = 45, hjust = 1))
grid.arrange(n1, n2)
```

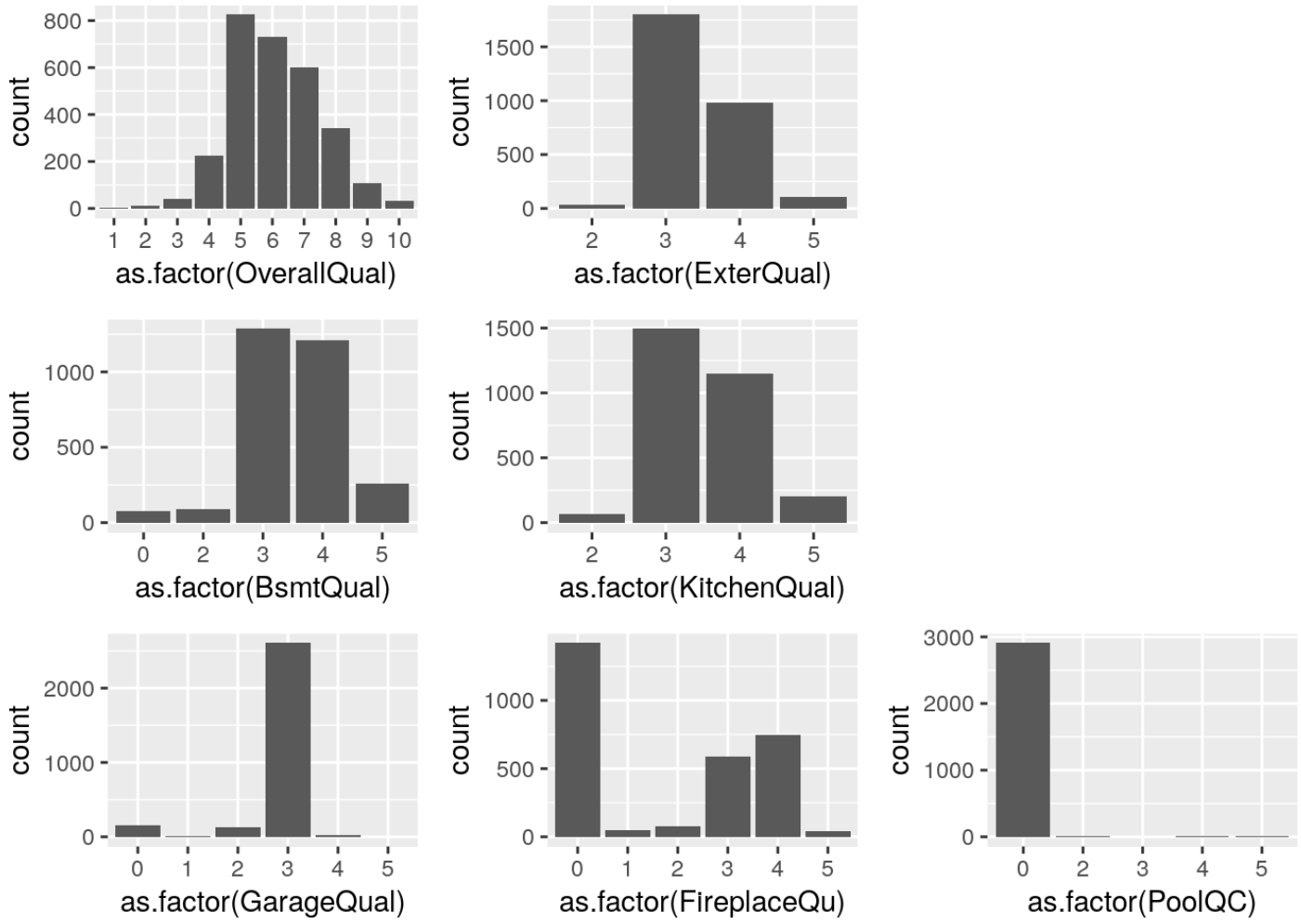


6.2.3 Overall Quality, and other Quality variables

I have already visualized the relation between Overall Quality and SalePrice in my initial explorations, but I want to visualize the frequency distribution as well. As there are more quality measurements, I am taking the opportunity to bundle them in this section.

```
q1 <- ggplot(data=all, aes(x=as.factor(OverallQual))) +
        geom_histogram(stat='count')
q2 <- ggplot(data=all, aes(x=as.factor(ExterQual))) +
        geom_histogram(stat='count')
q3 <- ggplot(data=all, aes(x=as.factor(BsmtQual))) +
        geom_histogram(stat='count')
q4 <- ggplot(data=all, aes(x=as.factor(KitchenQual))) +
        geom_histogram(stat='count')
q5 <- ggplot(data=all, aes(x=as.factor(GarageQual))) +
        geom_histogram(stat='count')
q6 <- ggplot(data=all, aes(x=as.factor(FireplaceQu))) +
        geom_histogram(stat='count')
q7 <- ggplot(data=all, aes(x=as.factor(PoolQC))) +
        geom_histogram(stat='count')

layout <- matrix(c(1,2,8,3,4,8,5,6,7),3,3,byrow=TRUE)
multiplot(q1, q2, q3, q4, q5, q6, q7, layout=layout)
```



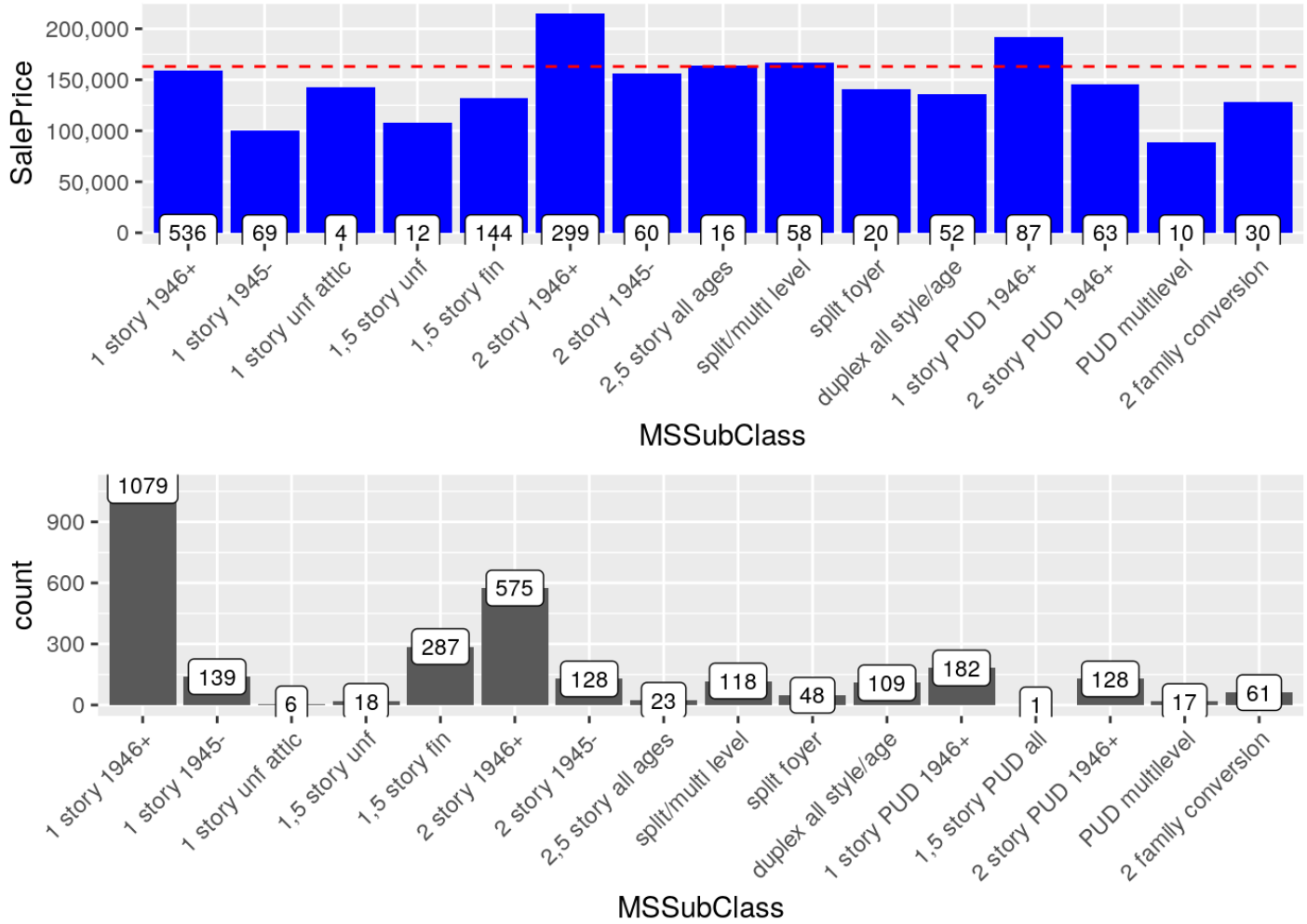
Overall Quality is very important, and also more granular than the other variables. External Quality is also improtant, but has a high correlation with Overall Quality (0.73). Kitchen Quality also seems one to keep, as all houses have a kitchen and there is a variance with some substance. Garage Quality does not seem to distinguish much, as the majority of garages have Q3. Fireplace Quality is in the list of high correlations, and in the important variables list. The PoolQC is just very sparse (the 13 pools cannot even be seen on this scale). I will look at creating a 'has pool' variable later on.

6.2.4 The second most important categorical variable; MSSubClass

The first visualization shows the median SalePrice by MSSubClass. The frequency (number of houses) of each MSSubClass in the train set is shown in the labels.

The histrogram shows the frequencies across all data. Most houses are relatively new, and have one or two stories.

```
ms1 <- ggplot(all[!is.na(all$SalePrice),], aes(x=MSSubClass, y=SalePrice)) +
        geom_bar(stat='summary', fun.y = "median", fill='blue') +
        theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=50000), labels = comma) +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size
=3) +
        geom_hline(yintercept=163000, linetype="dashed", color = "red") #dashe
d line is median SalePrice
ms2 <- ggplot(data=all, aes(x=MSSubClass)) +
        geom_histogram(stat='count')+
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size
=3) +
        theme(axis.text.x = element_text(angle = 45, hjust = 1))
grid.arrange(ms1, ms2)
```



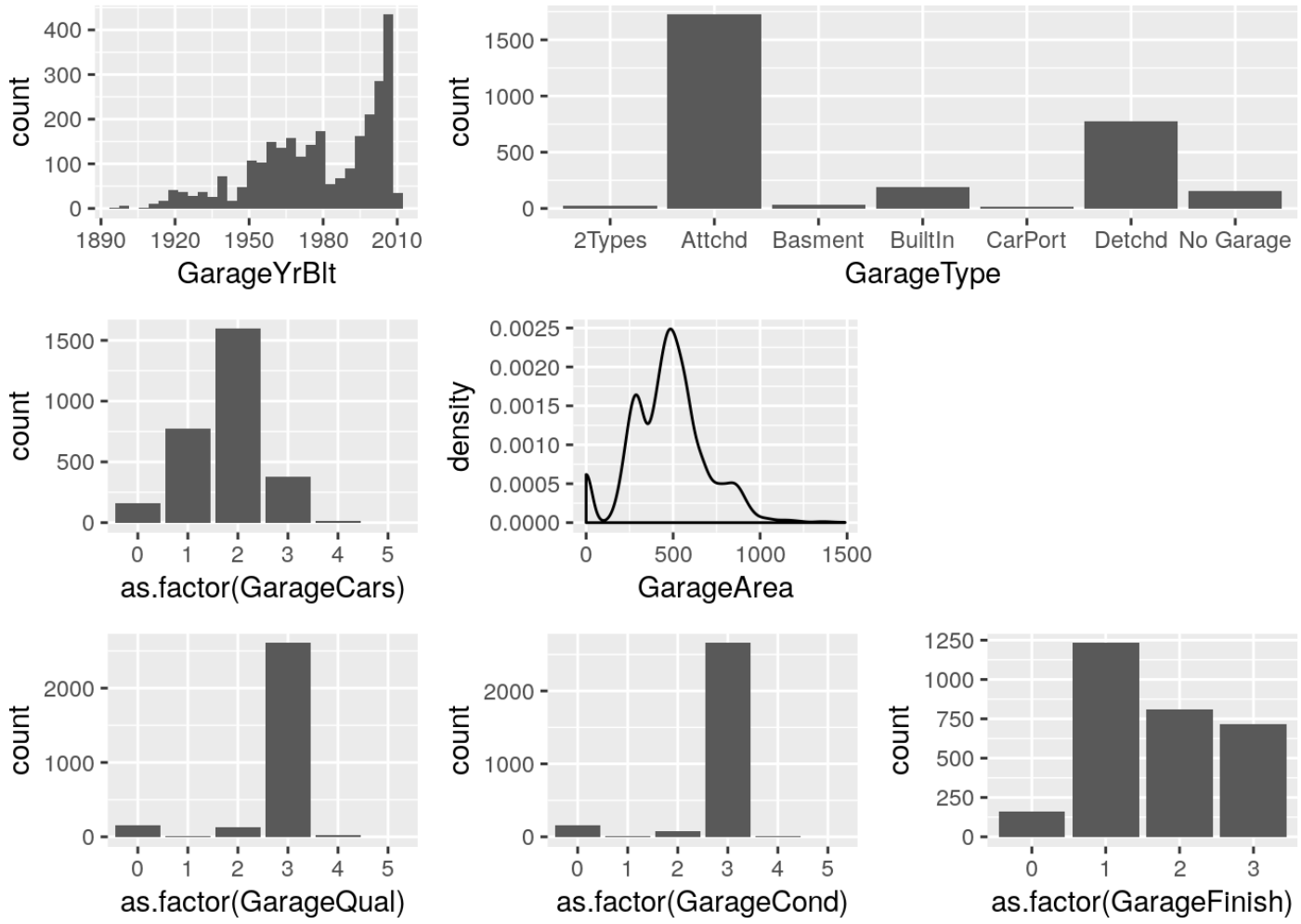
6.2.5 Garage variables

Several Garage variables have a high correlation with SalePrice, and are also in the top-20 list of the quick random forest. However, there is multicolinearity among them and I think that 7 garage variables is too many anyway. I feel that something like 3 variables should be sufficient (possibly GarageCars, GarageType, and a Quality measurement), but before I do any selection I am visualizing all of them in this section.

```
#correct error
all$GarageYrBlt[2593] <- 2007 #this must have been a typo. GarageYrBlt=2207, YearBuilt=2006, YearRemodAdd=2007.
```

```
g1 <- ggplot(data=all[all$GarageCars !=0,], aes(x=GarageYrBlt)) +
        geom_histogram()
g2 <- ggplot(data=all, aes(x=as.factor(GarageCars))) +
        geom_histogram(stat='count')
g3 <- ggplot(data= all, aes(x=GarageArea)) +
        geom_density()
g4 <- ggplot(data=all, aes(x=as.factor(GarageCond))) +
        geom_histogram(stat='count')
g5 <- ggplot(data=all, aes(x=GarageType)) +
        geom_histogram(stat='count')
g6 <- ggplot(data=all, aes(x=as.factor(GarageQual))) +
        geom_histogram(stat='count')
g7 <- ggplot(data=all, aes(x=as.factor(GarageFinish))) +
        geom_histogram(stat='count')

layout <- matrix(c(1,5,5,2,3,8,6,4,7),3,3,byrow=TRUE)
multiplot(g1, g2, g3, g4, g5, g6, g7, layout=layout)
```



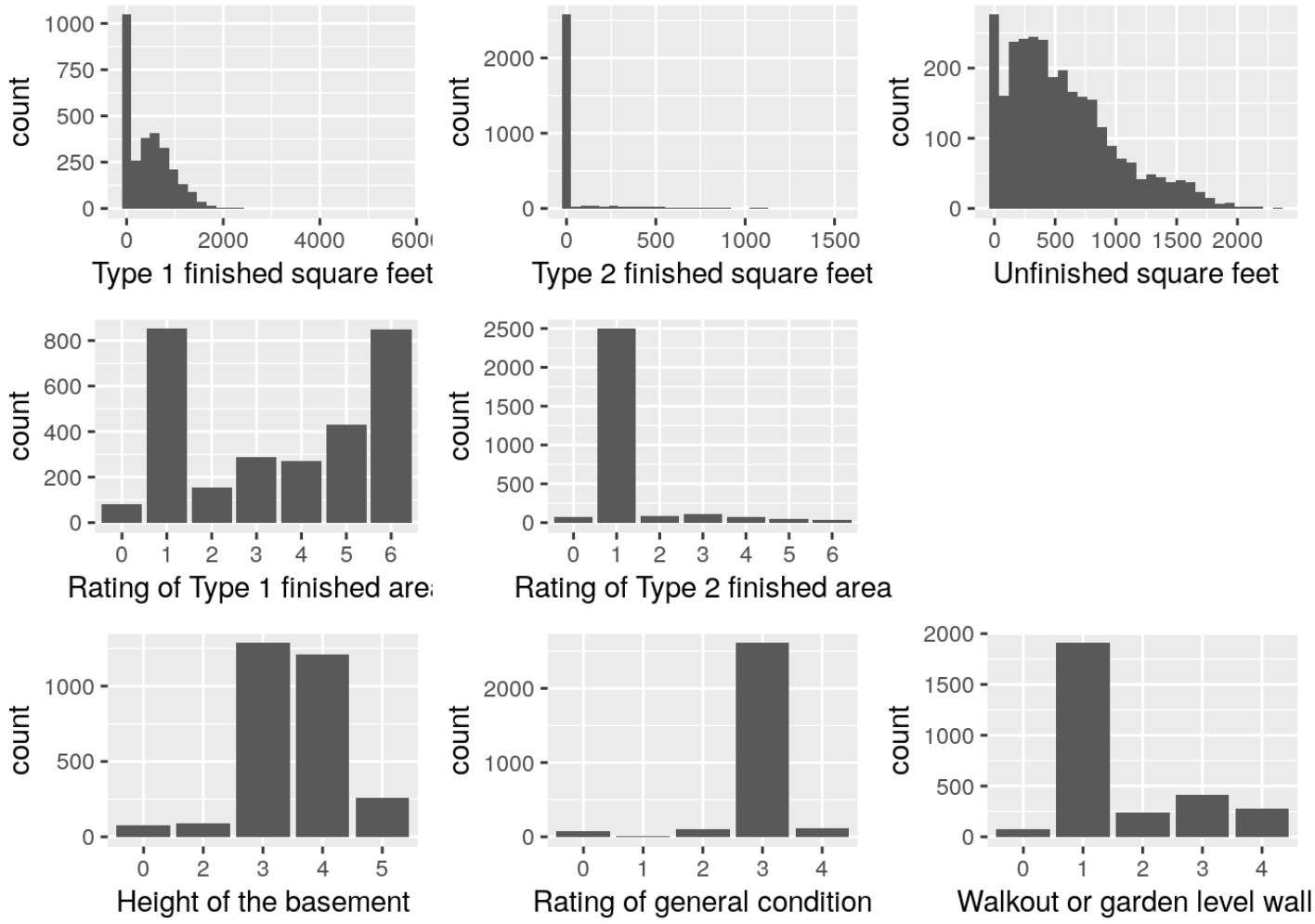
As already mentioned in section 4.2, GarageCars and GarageArea are highly correlated. Here, GarageQual and GarageCond also seem highly correlated, and both are dominated by level =3.

6.2.6 Basement variables

Similar the garage variables, multiple basement variables are important in the correlations matrix and the Top 20 RF predictors list. However, 11 basement variables seems an overkill. Before I decide what I am going to do with them, I am visualizing 8 of them below. The 2 “Bathroom” variables are dealt with in Feature Engineering (section 7.1), and the “Basement square feet” is already discussed in section 6.2.1.

```
b1 <- ggplot(data=all, aes(x=BsmtFinSF1)) +
        geom_histogram() + labs(x='Type 1 finished square feet')
b2 <- ggplot(data=all, aes(x=BsmtFinSF2)) +
        geom_histogram()+ labs(x='Type 2 finished square feet')
b3 <- ggplot(data=all, aes(x=BsmtUnfSF)) +
        geom_histogram()+ labs(x='Unfinished square feet')
b4 <- ggplot(data=all, aes(x=as.factor(BsmtFinType1))) +
        geom_histogram(stat='count')+ labs(x='Rating of Type 1 finished area')
b5 <- ggplot(data=all, aes(x=as.factor(BsmtFinType2))) +
        geom_histogram(stat='count')+ labs(x='Rating of Type 2 finished area')
b6 <- ggplot(data=all, aes(x=as.factor(BsmtQual))) +
        geom_histogram(stat='count')+ labs(x='Height of the basement')
b7 <- ggplot(data=all, aes(x=as.factor(BsmtCond))) +
        geom_histogram(stat='count')+ labs(x='Rating of general condition')
b8 <- ggplot(data=all, aes(x=as.factor(BsmtExposure))) +
        geom_histogram(stat='count')+ labs(x='Walkout or garden level walls')

layout <- matrix(c(1,2,3,4,5,9,6,7,8),3,3,byrow=TRUE)
multiplot(b1, b2, b3, b4, b5, b6, b7, b8, layout=layout)
```



So it seemed as if the Total Basement Surface in square feet (TotalBsmtSF) is further broken down into finished areas (2 if more than one type of finish), and unfinished area. I did a check between the correlation of total of those 3 variables, and TotalBsmtSF. The correlation is exactely 1, so that's a good thing (no errors or small discrepancies)!

Basement Quality is a confusing variable name, as it turns out that it specifically rates the Height of the basement.

7 Feature engineering

7.1 Total number of Bathrooms

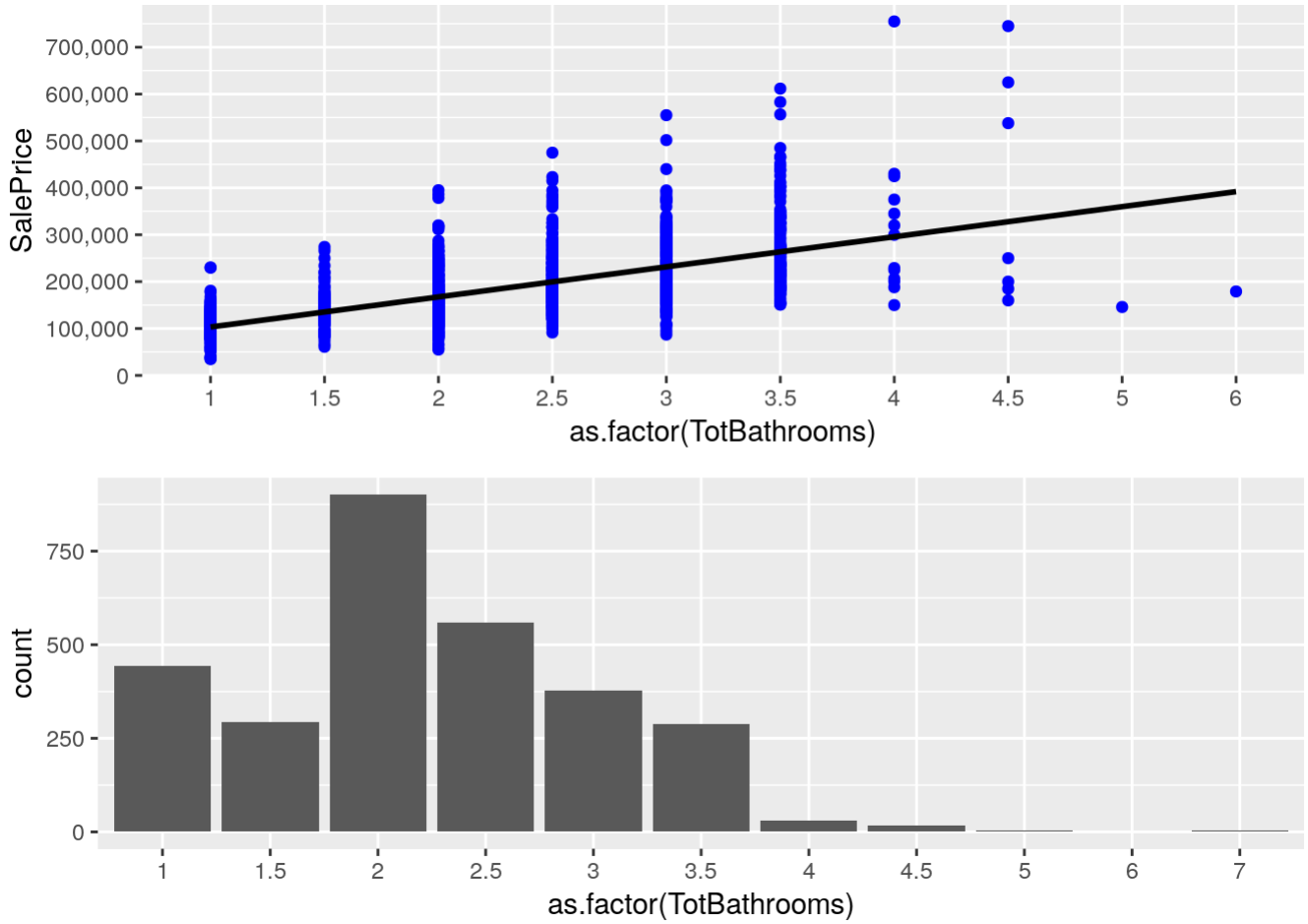
There are 4 bathroom variables. Individually, these variables are not very important. However, I assume that I if I add them up into one predictor, this predictor is likely to become a strong one.

"A half-bath, also known as a powder room or guest bath, has only two of the four main bathroom components- typically a toilet and sink." Consequently, I will also count the half bathrooms as half.

```
all$TotBathrooms <- all$FullBath + (all$HalfBath*0.5) + all$BsmtFullBath + (al
l$BsmtHalfBath*0.5)
```

As you can see in the first graph, there now seems to be a clear correlation (it's 0.63). The frequency distribution of Bathrooms in all data is shown in the second graph.

```
tb1 <- ggplot(data=all[!is.na(all$SalePrice),], aes(x=as.factor(TotBathrooms),
y=SalePrice))+
        geom_point(col='blue') + geom_smooth(method = "lm", se=FALSE, color="b
lack", aes(group=1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=100000), labels = comma)
tb2 <- ggplot(data=all, aes(x=as.factor(TotBathrooms))) +
        geom_histogram(stat='count')
grid.arrange(tb1, tb2)
```

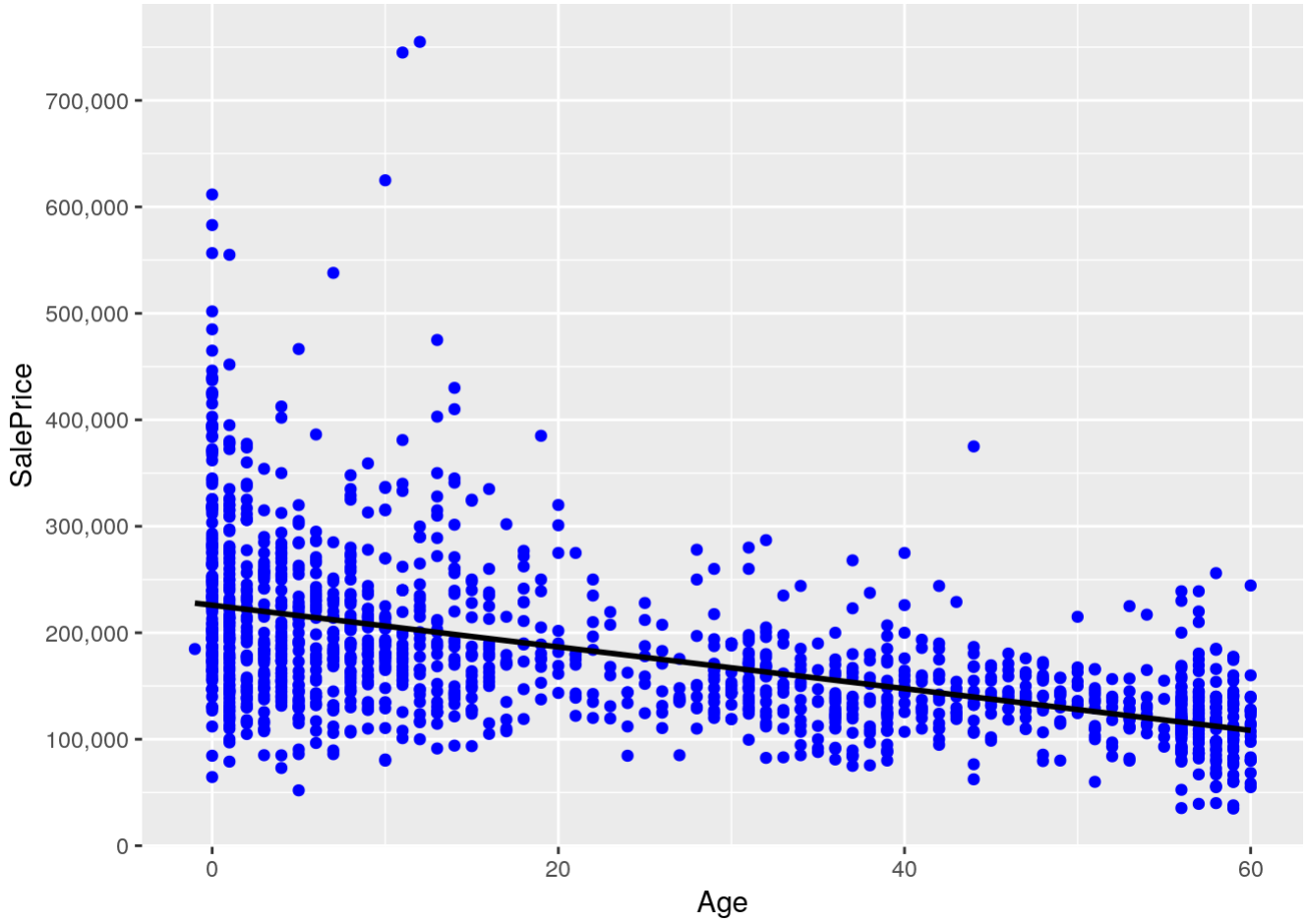


7.2 Adding 'House Age', 'Remodeled (Yes/No)', and IsNew variables

Altogether, there are 3 variables that are relevant with regards to the Age of a house; YearBlt, YearRemodAdd, and YearSold. YearRemodAdd defaults to YearBuilt if there has been no Remodeling/Addition. I will use YearRemodeled and YearSold to determine the Age. However, as parts of old constructions will always remain and only parts of the house might have been renovated, I will also introduce a Remodeled Yes/No variable. This should be seen as some sort of penalty parameter that indicates that if the Age is based on a remodeling date, it is probably worth less than houses that were built from scratch in that same year.

```
all$Remod <- ifelse(all$YearBuilt==all$YearRemodAdd, 0, 1) #0=No Remodeling, 1=Remodeling
all$Age <- as.numeric(all$YrSold)-all$YearRemodAdd
```

```
ggplot(data=all[!is.na(all$SalePrice),], aes(x=Age, y=SalePrice))+
        geom_point(col='blue') + geom_smooth(method = "lm", se=FALSE, color="black", aes(group=1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=100000), labels = comma)
```



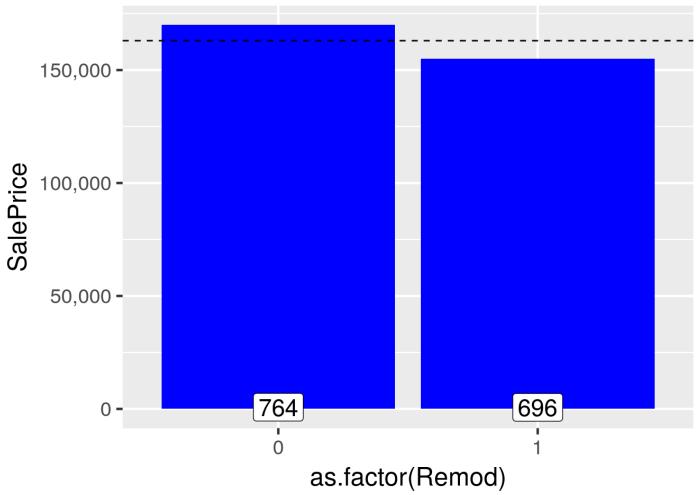
As expected, the graph shows a negative correlation with Age (old house are worth less).

```
cor(all$SalePrice[!is.na(all$SalePrice)], all$Age[!is.na(all$SalePrice)])
```

```
## [1] -0.5090787
```

As you can see below, houses that are remodeled are worth less indeed, as expected.

```
ggplot(all[!is.na(all$SalePrice),], aes(x=as.factor(Remod), y=SalePrice)) +
        geom_bar(stat='summary', fun.y = "median", fill='blue') +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size
=6) +
        scale_y_continuous(breaks= seq(0, 800000, by=50000), labels = comma) +
        theme_grey(base_size = 18) +
        geom_hline(yintercept=163000, linetype="dashed") #dashed line is median
SalePrice
```



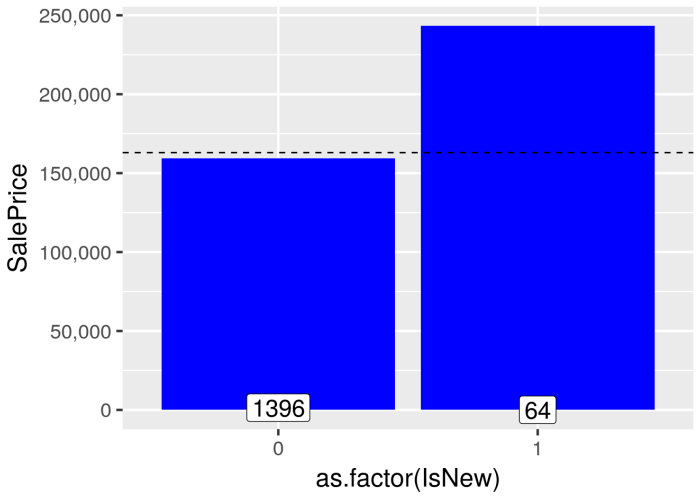
Finally, I am creating the IsNew variable below. Altogether, there are 116 new houses in the dataset.

```
all$IsNew <- ifelse(all$YrSold==all$YearBuilt, 1, 0)
table(all$IsNew)
```

```
##
##    0    1
## 2803  116
```

These 116 new houses are fairly evenly distributed among train and test set, and as you can see new houses are worth considerably more on average.

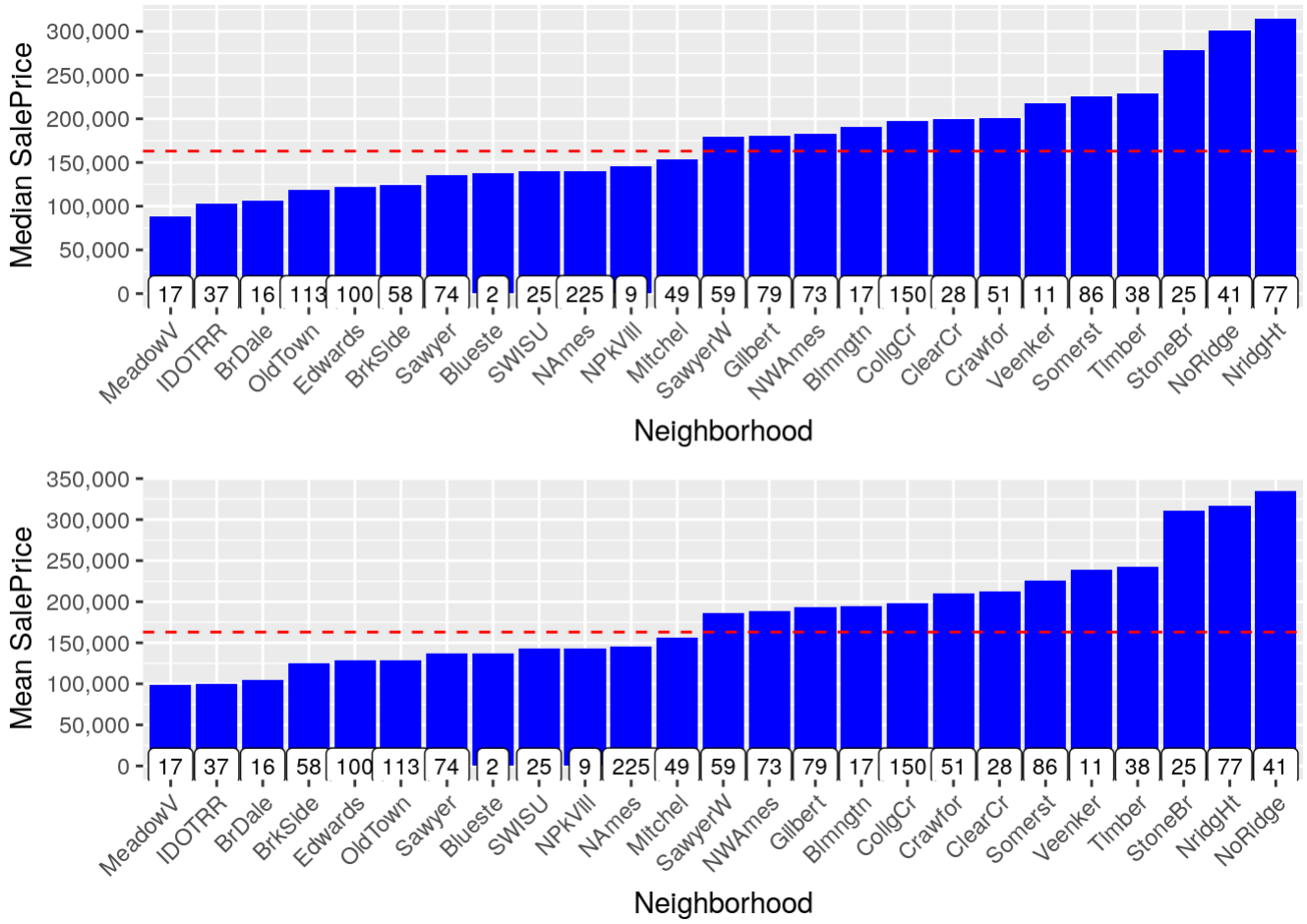
```
ggplot(all[!is.na(all$SalePrice),], aes(x=as.factor(IsNew), y=SalePrice)) +
        geom_bar(stat='summary', fun.y = "median", fill='blue') +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size
=6) +
        scale_y_continuous(breaks= seq(0, 800000, by=50000), labels = comma) +
        theme_grey(base_size = 18) +
        geom_hline(yintercept=163000, linetype="dashed") #dashed line is median
SalePrice
```




```
all$YrSold <- as.factor(all$YrSold) #the numeric version is now not needed anym
ore
```

7.3 Binning Neighborhood

```
nb1 <- ggplot(all[!is.na(all$SalePrice),], aes(x=reorder(Neighborhood, SalePri
ce, FUN=median), y=SalePrice)) +
        geom_bar(stat='summary', fun.y = "median", fill='blue') + labs(x='Neig
hborhood', y='Median SalePrice') +
        theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=50000), labels = comma) +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size
=3) +
        geom_hline(yintercept=163000, linetype="dashed", color = "red") #dashe
d line is median SalePrice
nb2 <- ggplot(all[!is.na(all$SalePrice),], aes(x=reorder(Neighborhood, SalePri
ce, FUN=mean), y=SalePrice)) +
        geom_bar(stat='summary', fun.y = "mean", fill='blue') + labs(x='Neighb
orhood', y="Mean SalePrice") +
        theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=50000), labels = comma) +
        geom_label(stat = "count", aes(label = ..count.., y = ..count..), size
=3) +
        geom_hline(yintercept=163000, linetype="dashed", color = "red") #dashe
d line is median SalePrice
grid.arrange(nb1, nb2)
```



Both the median and mean Saleprices agree on 3 neighborhoods with substantially higher saleprices. The separation of the 3 relatively poor neighborhoods is less clear, but at least both graphs agree on the same 3 poor neighborhoods. Since I do not want to 'overbin', I am only creating categories for those 'extremes'.

```
all$NeighRich[all$Neighborhood %in% c('StoneBr', 'NridgHt', 'NoRidge')] <- 2
all$NeighRich[!all$Neighborhood %in% c('MeadowV', 'IDOTRR', 'BrDale', 'StoneB
r', 'NridgHt', 'NoRidge')] <- 1
all$NeighRich[all$Neighborhood %in% c('MeadowV', 'IDOTRR', 'BrDale')] <- 0
```

```
table(all$NeighRich)
```

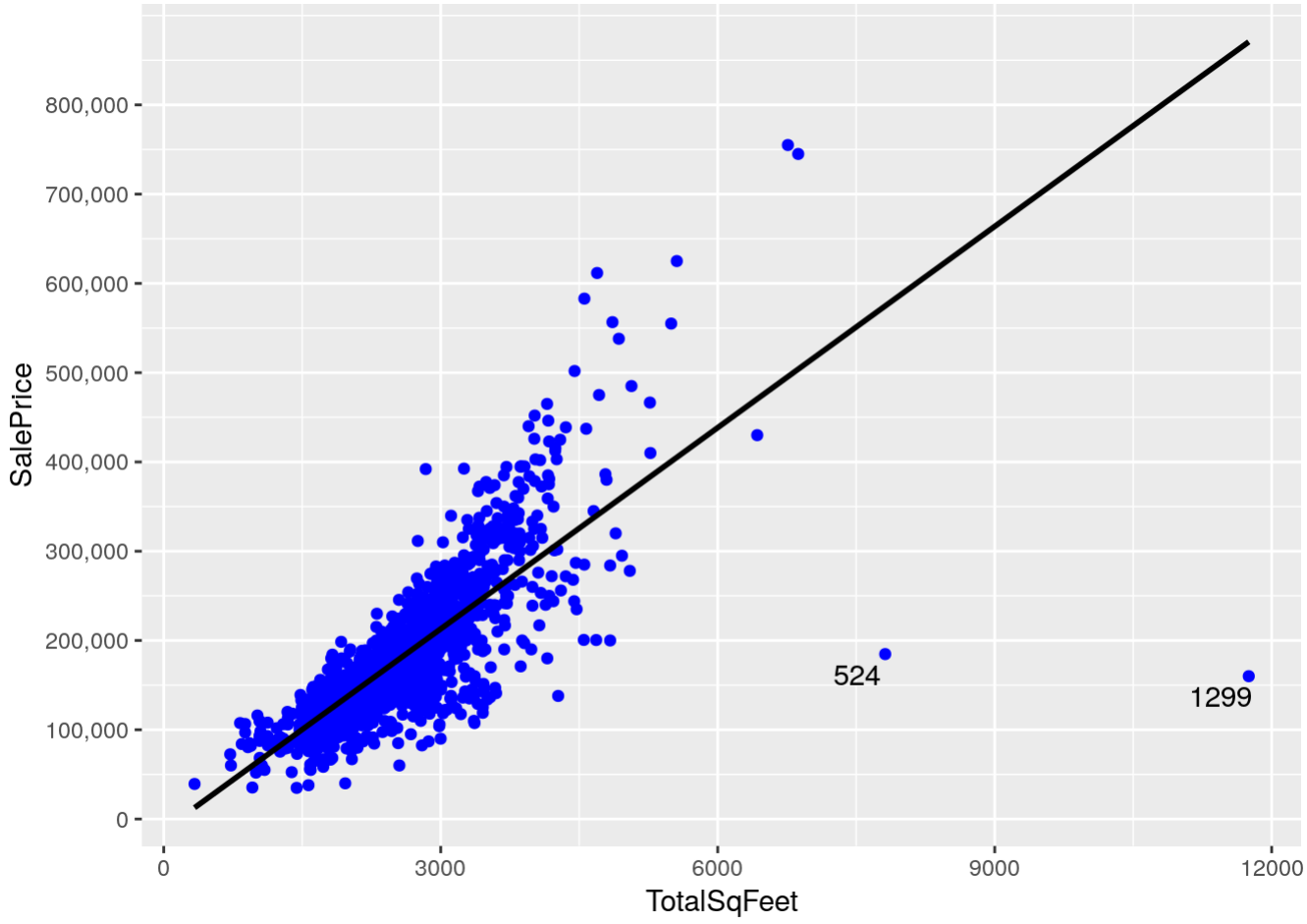
```
##
##    0    1    2
##  160 2471  288
```

7.4 Total Square Feet

As the total living space generally is very important when people buy houses, I am adding a predictors that adds up the living space above and below ground.

```
all$TotalSqFeet <- all$GrLivArea + all$TotalBsmtSF
```

```
ggplot(data=all[!is.na(all$SalePrice),], aes(x=TotalSqFeet, y=SalePrice))+
        geom_point(col='blue') + geom_smooth(method = "lm", se=FALSE, color="b
lack", aes(group=1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=100000), labels = comma)
 +
        geom_text_repel(aes(label = ifelse(all$GrLivArea[!is.na(all$SalePric
e)]>4500, rownames(all), '')))
```



As expected, the correlation with SalePrice is very strong indeed (0.78).

```
cor(all$SalePrice, all$TotalSqFeet, use= "pairwise.complete.obs")
```

```
## [1] 0.7789588
```

The two potential outliers seem to 'outlie' even more than before. By taking out these two outliers, the correlation increases by 5%.

```
cor(all$SalePrice[-c(524, 1299)], all$TotalSqFeet[-c(524, 1299)], use= "pairwi
se.complete.obs")
```

```
## [1] 0.829042
```

7.5 Consolidating Porch variables

Below, I listed the variables that seem related regarding porches.

- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet
- 3SsnPorch: Three season porch area in square feet
- ScreenPorch: Screen porch area in square feet

As far as I know, porches are sheltered areas outside of the house, and a wooden deck is unsheltered. Therefore, I am leaving WoodDeckSF alone, and are only consolidating the 4 porch variables.

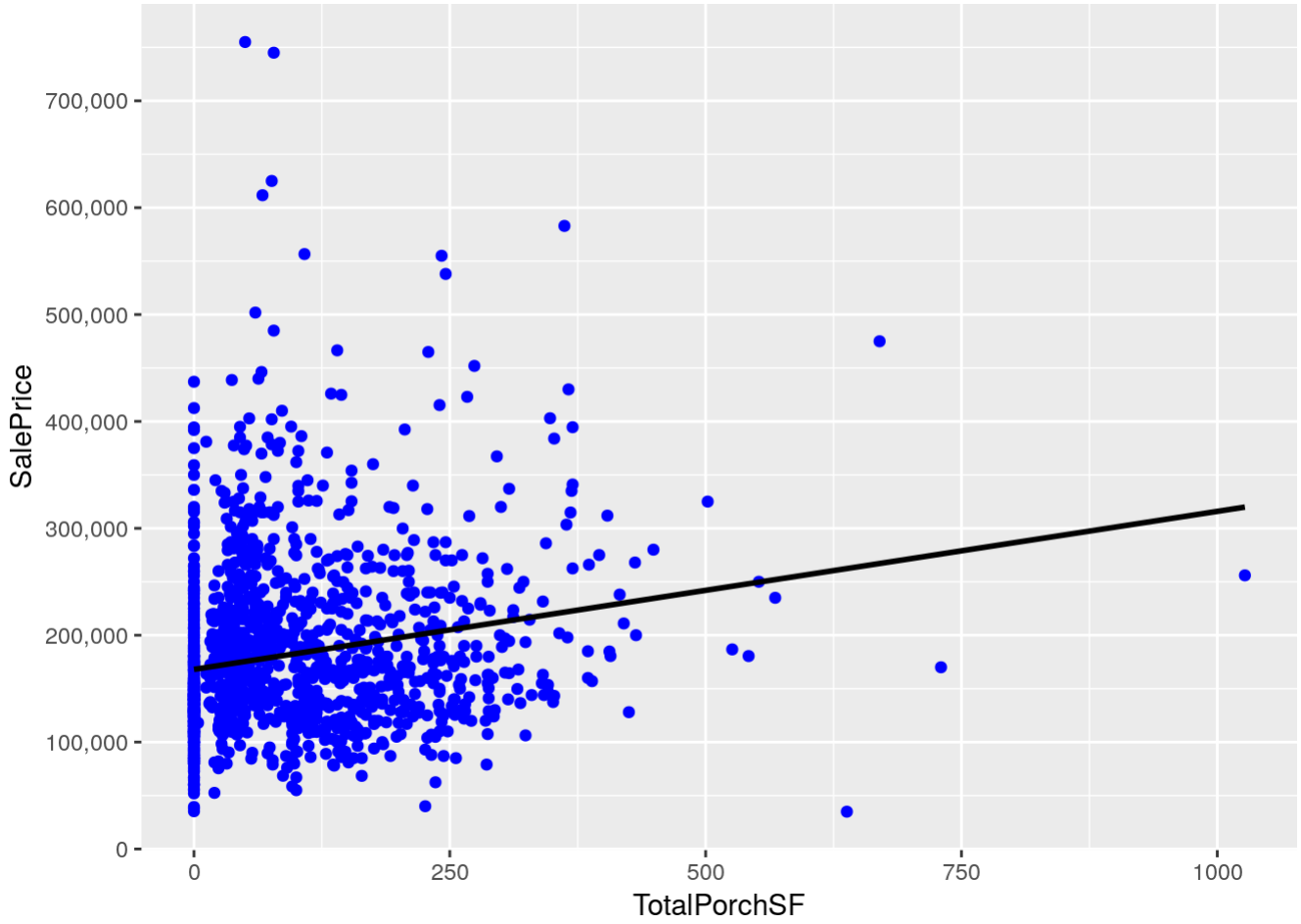
```
all$TotalPorchSF <- all$OpenPorchSF + all$EnclosedPorch + all$X3SsnPorch + all
$ScreenPorch
```

Although adding up these Porch areas makes sense (there should not be any overlap between areas), the correlation with SalePrice is not very strong.

```
cor(all$SalePrice, all$TotalPorchSF, use= "pairwise.complete.obs")
```

```
## [1] 0.1957389
```

```
ggplot(data=all[!is.na(all$SalePrice),], aes(x=TotalPorchSF, y=SalePrice))+
        geom_point(col='blue') + geom_smooth(method = "lm", se=FALSE, color="b
lack", aes(group=1)) +
        scale_y_continuous(breaks= seq(0, 800000, by=100000), labels = comma)
```



8 Preparing data for modeling

8.1 Dropping highly correlated variables

First of all, I am dropping a variable if two variables are highly correlated. To find these correlated pairs, I have used the correlations matrix again (see section 6.1). For instance: GarageCars and GarageArea have a correlation of 0.89. Of those two, I am dropping the variable with the lowest correlation with SalePrice (which is GarageArea with a SalePrice correlation of 0.62. GarageCars has a SalePrice correlation of 0.64).

```
dropVars <- c('YearRemodAdd', 'GarageYrBlt', 'GarageArea', 'GarageCond', 'Tota
lBsmtSF', 'TotalRmsAbvGrd', 'BsmtFinSF1')

all <- all[,!(names(all) %in% dropVars)]
```

8.2 Removing outliers

For the time being, I am keeping it simple and just remove the two really big houses with low SalePrice manually. However, I intend to investigate this more thorough in a later stage (possibly using the 'outliers' package).

```
all <- all[-c(524, 1299),]
```

8.3 PreProcessing predictor variables

Before modeling I need to center and scale the 'true numeric' predictors (so not variables that have been label encoded), and create dummy variables for the categorical predictors. Below, I am splitting the dataframe into one with all (true) numeric variables, and another dataframe holding the (ordinal) factors.

```
numericVarNames <- numericVarNames[!(numericVarNames %in% c('MSSubClass', 'MoS
old', 'YrSold', 'SalePrice', 'OverallQual', 'OverallCond'))] #numericVarNames w
as created before having done anything
numericVarNames <- append(numericVarNames, c('Age', 'TotalPorchSF', 'TotBathro
oms', 'TotalSqFeet'))

DFnumeric <- all[, names(all) %in% numericVarNames]

DFfactors <- all[, !(names(all) %in% numericVarNames)]
DFfactors <- DFfactors[, names(DFfactors) != 'SalePrice']

cat('There are', length(DFnumeric), 'numeric variables, and', length(DFfactor
s), 'factor variables')
```

```
## There are 30 numeric variables, and 49 factor variables
```

8.3.1 Skewness and normalizing of the numeric predictors

Skewness Skewness is a measure of the symmetry in a distribution. A symmetrical dataset will have a skewness equal to 0. So, a normal distribution will have a skewness of 0. Skewness essentially measures the relative size of the two tails. As a rule of thumb, skewness should be between -1 and 1. In this range, data are considered fairly symmetrical. In order to fix the skewness, I am taking the log for all numeric predictors with an absolute skew greater than 0.8 (actually: log+1, to avoid division by zero issues).

```
for(i in 1:ncol(DFnumeric)){
        if (abs(skew(DFnumeric[,i]))>0.8){
                DFnumeric[,i] <- log(DFnumeric[,i] +1)
        }
}
```

Normalizing the data

```
PreNum <- preProcess(DFnumeric, method=c("center", "scale"))
print(PreNum)
```

```
## Created from 2917 samples and 30 variables
##
## Pre-processing:
##   - centered (30)
##   - ignored (0)
##   - scaled (30)
```

```
DFnorm <- predict(PreNum, DFnumeric)
dim(DFnorm)
```

```
## [1] 2917   30
```

8.3.2 One hot encoding the categorical variables

The last step needed to ensure that all predictors are converted into numeric columns (which is required by most Machine Learning algorithms) is to ‘one-hot encode’ the categorical variables. This basically means that all (not ordinal) factor values are getting a seperate colums with 1s and 0s (1 basically means Yes/Present). To do this one-hot encoding, I am using the `model.matrix()` function.

```
DFdummies <- as.data.frame(model.matrix(~.-1, DFfactors))
dim(DFdummies)
```

```
## [1] 2917  201
```

8.3.3 Removing levels with few or no observations in train or test

In previous versions, I worked with Caret's `Near Zero Variance` function. Although this works, it also is a quick fix and too much information got lost. For instance, by using the defaults, all Neighborhoods with less than 146 houses are omitted as (one-hot encoded) variables (frequency ratio higher than 95/5). Therefore, I have taken amore carefull manual approach in this version.

```
#check if some values are absent in the test set
ZerocolTest <- which(colSums(DFdummies[(nrow(all[!is.na(all$SalePrice),])+1):n
row(all),])==0)
colnames(DFdummies[ZerocolTest])
```

```
##  [1] "Condition2RRAe"     "Condition2RRAn"     "Condition2RRNn"
##  [4] "HouseStyle2.5Fin"   "RoofMatlMembran"    "RoofMatlMetal"
##  [7] "RoofMatlRoll"       "Exterior1stImStucc" "Exterior1stStone"
## [10] "Exterior2ndOther"   "HeatingOthW"        "ElectricalMix"
## [13] "MiscFeatureTenC"
```

```
DFdummies <- DFdummies[,-ZerocolTest] #removing predictors
```

```
#check if some values are absent in the train set
ZerocolTrain <- which(colSums(DFdummies[1:nrow(all[!is.na(all$SalePrice),]),])
==0)
colnames(DFdummies[ZerocolTrain])
```

```
## [1] "MSSubClass1,5 story PUD all"
```

```
DFdummies <- DFdummies[,-ZerocolTrain] #removing predictor
```

Also taking out variables with less than 10 'ones' in the train set.

```
fewOnes <- which(colSums(DFdummies[1:nrow(all[!is.na(all$SalePrice),]),])<10)
colnames(DFdummies[fewOnes])
```

```
##  [1] "MSSubClass1 story unf attic" "LotConfigFR3"
##  [3] "NeighborhoodBlueste"         "NeighborhoodNPkVill"
##  [5] "Condition1PosA"              "Condition1RRNe"
##  [7] "Condition1RRNn"              "Condition2Feedr"
##  [9] "Condition2PosA"              "Condition2PosN"
## [11] "RoofStyleMansard"            "RoofStyleShed"
## [13] "RoofMatlWdShake"             "RoofMatlWdShngl"
## [15] "Exterior1stAsphShn"          "Exterior1stBrkComm"
## [17] "Exterior1stCBlock"           "Exterior2ndAsphShn"
## [19] "Exterior2ndBrk Cmn"          "Exterior2ndCBlock"
## [21] "Exterior2ndStone"            "FoundationStone"
## [23] "FoundationWood"              "HeatingGrav"
## [25] "HeatingWall"                 "ElectricalFuseP"
## [27] "GarageTypeCarPort"           "MiscFeatureOthr"
## [29] "SaleTypeCon"                 "SaleTypeConLD"
## [31] "SaleTypeConLI"               "SaleTypeConLw"
## [33] "SaleTypeCWD"                 "SaleTypeOth"
## [35] "SaleConditionAdjLand"
```

```
DFdummies <- DFdummies[,-fewOnes] #removing predictors
dim(DFdummies)
```

```
## [1] 2917  152
```

Altogether, I have removed 49 one-hot encoded predictors with little or no variance. Altough this may seem a significant number, it is actually much less than the number of predictors that were taken out by using caret's `near zero variance` function (using its default thresholds).

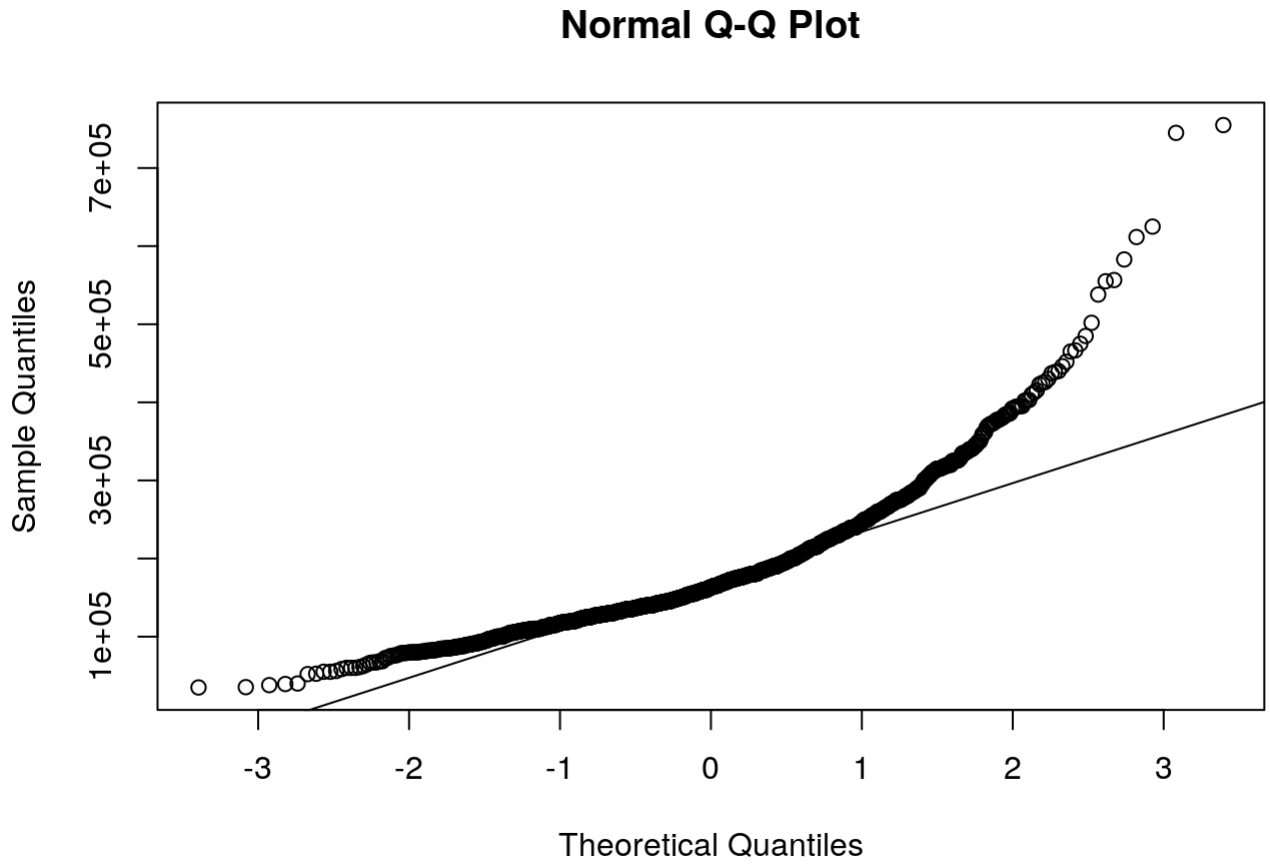
```
combined <- cbind(DFnorm, DFdummies) #combining all (now numeric) predictors in to one dataframe
```

8.4 Dealing with skewness of response variable

```
skew(all$SalePrice)
```

```
## [1] 1.877427
```

```
qqnorm(all$SalePrice)
qqline(all$SalePrice)
```



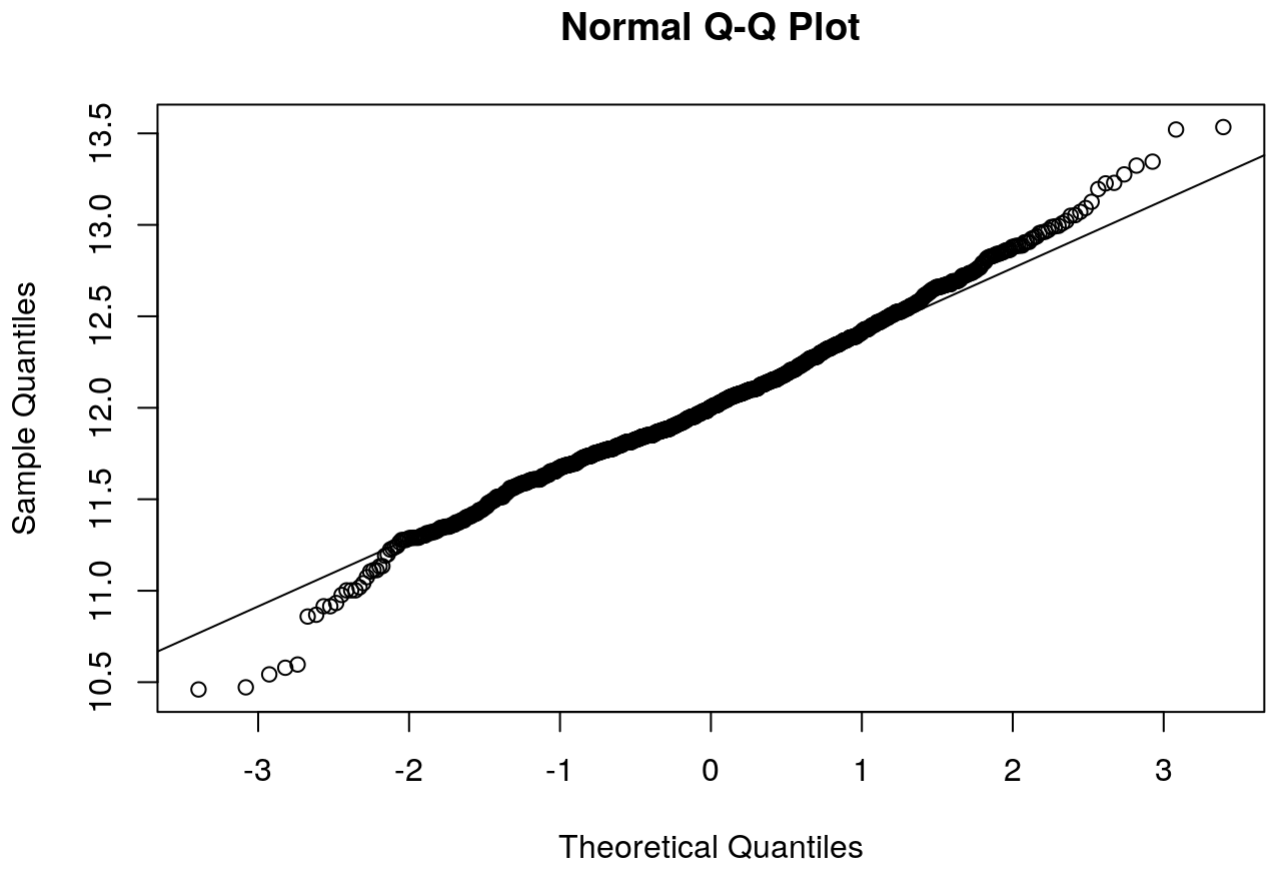
The skew of 1.87 indicates a right skew that is too high, and the Q-Q plot shows that sale prices are also not normally distributed. To fix this I am taking the log of SalePrice.

```
all$SalePrice <- log(all$SalePrice) #default is the natural logarithm, "+1" is
 not necessary as there are no 0's
skew(all$SalePrice)
```

```
## [1] 0.1213182
```

As you can see,the skew is now quite low and the Q-Q plot is also looking much better.

```
qqnorm(all$SalePrice)
qqline(all$SalePrice)
```



8.5 Composing train and test sets

```
train1 <- combined[!is.na(all$SalePrice),]
test1 <- combined[is.na(all$SalePrice),]
```

9 Modeling

9.1 Lasso regression model

I have also tried Ridge and Elastic Net models, but since lasso gives the best results of those 3 models I am only keeping the lasso model in the document.

The elastic-net penalty is controlled by alpha, and bridges the gap between lasso (alpha=1) and ridge (alpha=0). The tuning parameter lambda controls the overall strength of the penalty. It is known that the ridge penalty shrinks the coefficients of correlated predictors towards each other while the lasso tends to pick one of them and discard the others.

Below, I am using caret cross validation to find the best value for lambda, which is the only hyperparameter that needs to be tuned for the lasso model.

```
set.seed(27042018)
my_control <-trainControl(method="cv", number=5)
lassoGrid <- expand.grid(alpha = 1, lambda = seq(0.001,0.1,by = 0.0005))

lasso_mod <- train(x=train1, y=all$SalePrice[!is.na(all$SalePrice)], method='g
lmnet', trControl= my_control, tuneGrid=lassoGrid)
lasso_mod$bestTune
```

```
##   alpha lambda
## 4     1 0.0025
```

```
min(lasso_mod$results$RMSE)
```

```
## [1] 0.1121579
```

The documentation of the caret `varImp' function says: for glmboost and glmnet the absolute value of the coefficients corresponding to the tuned model are used.

Although this means that a real ranking of the most important variables is not stored, it gives me the opportunity to find out how many of the variables are not used in the model (and hence have coefficient 0).

```
lassoVarImp <- varImp(lasso_mod,scale=F)
lassoImportance <- lassoVarImp$importance

varsSelected <- length(which(lassoImportance$Overall!=0))
varsNotSelected <- length(which(lassoImportance$Overall==0))

cat('Lasso uses', varsSelected, 'variables in its model, and did not select',
 varsNotSelected, 'variables.')
```

```
## Lasso uses 100 variables in its model, and did not select 82 variables.
```

So lasso did what it is supposed to do: it seems to have dealt with multicolinearity well by not using about 45% of the available variables in the model.

```
LassoPred <- predict(lasso_mod, test1)
predictions_lasso <- exp(LassoPred) #need to reverse the log to the real values
head(predictions_lasso)
```

```
##     1461     1462     1463     1464     1465     1466
## 114351.8 162204.8 179455.3 197564.7 205952.8 169839.8
```

9.2 XGBoost model

Initially, I just worked with the XGBoost package directly. The main reason for this was that the package uses its own efficient datastructure (xgb.DMatrix). The package also provides a cross validation function. However, this CV function only determines the optimal number of rounds, and does not support a full grid search of hyperparameters.

Although caret does not seem to use the (fast) datastructure of the xgb package, I eventually decided to do hyperparameter tuning with it anyway, as it at least supports a full grid search. As far as I understand it, the main parameters to tune to avoid overfitting are max_depth, and min_child_weight (see XGBoost documentation (http://xgboost.readthedocs.io/en/latest/how_to/param_tuning.html)). Below I am setting up a grid that tunes both these parameters, and also the eta (learning rate).

```
xgb_grid = expand.grid(
nrounds = 1000,
eta = c(0.1, 0.05, 0.01),
max_depth = c(2, 3, 4, 5, 6),
gamma = 0,
colsample_bytree=1,
min_child_weight=c(1, 2, 3, 4 ,5),
subsample=1
)
```

The next step is to let caret find the best hyperparameter values (using 5 fold cross validation).

```
#xgb_caret <- train(x=train1, y=all$SalePrice[!is.na(all$SalePrice)], method='xg
bTree', trControl= my_control, tuneGrid=xgb_grid)
#xgb_caret$bestTune
```

As expected, this took quite a bit of time (locally). As I want to limit the running time on Kaggle, I disabled the code, and am just continuing with the results. According to caret, the 'bestTune' parameters are:

- Max_depth=3
- eta=0.05
- Min_child_weight=4

In the remainder of this section, I will continue to work with the xgboost package directly. Below, I am starting with the preparation of the data in the recommended format.

```
label_train <- all$SalePrice[!is.na(all$SalePrice)]

# put our testing & training data into two seperates Dmatrixs objects
dtrain <- xgb.DMatrix(data = as.matrix(train1), label= label_train)
dtest <- xgb.DMatrix(data = as.matrix(test1))
```

In addition, I am taking over the best tuned values from the caret cross validation.

```
default_param<-list(
        objective = "reg:linear",
        booster = "gbtree",
        eta=0.05, #default = 0.3
        gamma=0,
        max_depth=3, #default=6
        min_child_weight=4, #default=1
        subsample=1,
        colsample_bytree=1
)
```

The next step is to do cross validation to determine the best number of rounds (for the given set of parameters).

```
xgbcv <- xgb.cv( params = default_param, data = dtrain, nrounds = 500, nfold =
5, showsd = T, stratified = T, print_every_n = 40, early_stopping_rounds = 10,
maximize = F)
```

```
## [1]  train-rmse:10.955588+0.004477   test-rmse:10.955537+0.019106
## Multiple eval metrics are present. Will use test_rmse for early stopping.
## Will train until test_rmse hasn't improved in 10 rounds.
##
## [41] train-rmse:1.428274+0.000561    test-rmse:1.428515+0.011791
## [81] train-rmse:0.219833+0.000801    test-rmse:0.230612+0.009490
## [121]    train-rmse:0.102497+0.001285    test-rmse:0.128856+0.008654
## [161]    train-rmse:0.090461+0.001218    test-rmse:0.122128+0.007505
## [201]    train-rmse:0.084142+0.001214    test-rmse:0.119557+0.007344
## [241]    train-rmse:0.079398+0.001195    test-rmse:0.118374+0.007088
## [281]    train-rmse:0.075716+0.001302    test-rmse:0.117645+0.006772
## [321]    train-rmse:0.072567+0.001139    test-rmse:0.117136+0.006720
## [361]    train-rmse:0.069770+0.001156    test-rmse:0.116745+0.006603
## [401]    train-rmse:0.067201+0.001059    test-rmse:0.116505+0.006574
## [441]    train-rmse:0.064814+0.001145    test-rmse:0.116386+0.006366
## Stopping. Best iteration:
## [454]    train-rmse:0.063958+0.001067    test-rmse:0.116289+0.006326
```

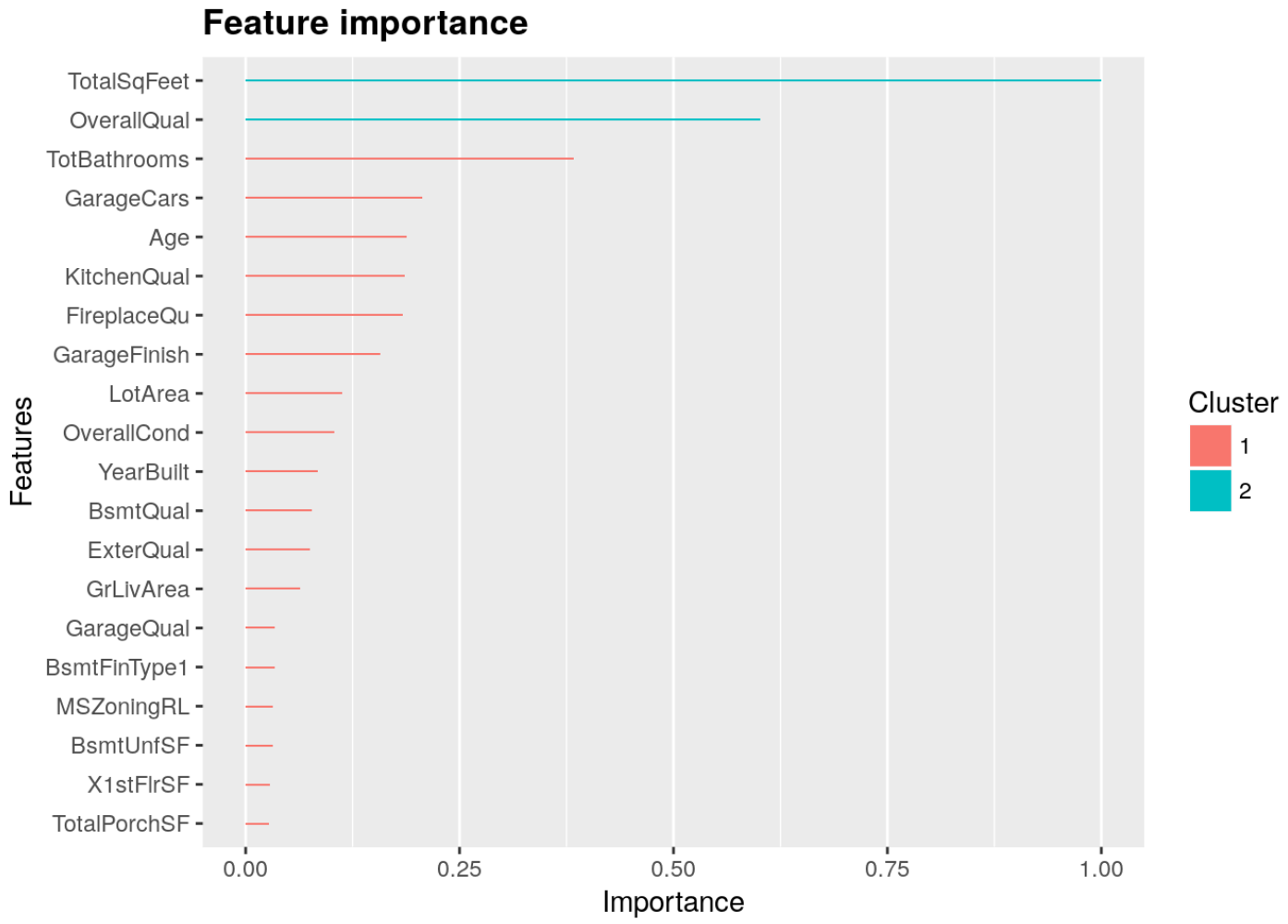
Although it was a bit of work, the hyperparameter tuning definitly paid of, as the cross validated RMSE inproved considerably (from 0.1225 without the caret tuning, to 0.1162 in this version)!

```
#train the model using the best iteration found by cross validation
xgb_mod <- xgb.train(data = dtrain, params=default_param, nrounds = 454)
```

```
XGBpred <- predict(xgb_mod, dtest)
predictions_XGB <- exp(XGBpred) #need to reverse the log to the real values
head(predictions_XGB)
```

```
## [1] 116386.8 162307.3 186494.0 187440.4 187258.3 166241.4
```

```
#view variable importance plot
library(Ckmeans.1d.dp) #required for ggplot clustering
mat <- xgb.importance (feature_names = colnames(train1),model = xgb_mod)
xgb.ggplot.importance(importance_matrix = mat[1:20], rel_to_first = TRUE)
```



9.3 Averaging predictions

Since the lasso and XGBoost algorithms are very different, averaging predictions likely improves the scores. As the lasso model does better regarding the cross validated RMSE score (0.1121 versus 0.1162), I am weigting the lasso model double.

```
sub_avg <- data.frame(Id = test_labels, SalePrice = (predictions_XGB+2*predict
ions_lasso)/3)
head(sub_avg)
```

```
##        Id SalePrice
## 1461 1461  115030.1
## 1462 1462  162238.9
## 1463 1463  181801.5
## 1464 1464  194189.9
## 1465 1465  199721.3
## 1466 1466  168640.3
```

```
write.csv(sub_avg, file = 'average.csv', row.names = F)
```