House Prices: Advanced Regression Techniques

Hye Soo Cho

Contents

1. Objective

The goal is to predict house price given the features of the house based on analysis of 79 explanatory variables that describe every aspect of residential houses in Ames, Iowa, using machine learning techniques. This is competition from Kaggle.

From Kaggle: 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.

2. Data Description and Data Cleaning

2-1. Data loading

```
library(readr)
train <- read_csv("~/Desktop/Spring 2020/EB 445/final/house-prices/train.csv")
test <- read_csv("~/Desktop/Spring 2020/EB 445/final/house-prices/test.csv")
```

2-2. Sample size and number of variables

```
dim(train)
```

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

```
dim(test)
```

```
## [1] 1459   80
```

In train dataset, there are 1460 samples with 81 predictors (80 predictors and 1 target variable,SalePrice).

In test dataset, there are 1459 samples with the same 80 predictors except SalePrice.

2-3. Description of variables

- MSSubClass: The building class
- MSZoning: The general zoning classification
- LotFrontage: Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access
- Alley: Type of alley access
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to main road or railroad
- Condition2: Proximity to main road or railroad (if a second is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Overall material and finish quality
- OverallCond: Overall condition rating
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date

- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet
- ExterQual: Exterior material quality
- ExterCond: Present condition of the material on the exterior
- Foundation: Type of foundation
- BsmtQual: Height of the basement
- BsmtCond: General condition of the basement
- BsmtExposure: Walkout or garden level basement walls
- BsmtFinType1: Quality of basement finished area
- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Quality of second finished area (if present)
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second floor square feet
- LowQualFinSF: Low quality finished square feet (all floors)
- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms
- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- BedroomAbvGr: Number of bedrooms above basement level
- KitchenAbvGr: Number of kitchens above basement level
- KitchenQual: Kitchen quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality rating
- Fireplaces: Number of fireplaces
- FireplaceQu: Fireplace quality
- GarageType: Garage location
- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- 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
- PoolArea: Pool area in square feet
- PoolQC: Pool quality
- Fence: Fence quality
- MiscFeature: Miscellaneous feature not covered in other categories

- MiscVal: $Value of miscellaneous feature
- MoSold: Month Sold
- YrSold: Year Sold
- SaleType: Type of sale
- SaleCondition: Condition of sale
- SalePrice - the property's sale price in dollars

2-4. Merging train and test datasets for data cleaning

```
# Before combining, make test  have the same format as trainby creating NAs for SalePrice
test$SalePrice <- NA

# Combine train and test datasets into all
all <- rbind(train, test)
```

I combined train and test datasets for data cleaning.

2-5. Remaning variable

```
# Rename variable that started with numeric number
names(all)[names(all) == "1stFlrSF"] <- "FirstFlrSF"
names(all)[names(all) == "2ndFlrSF"] <- "SecondFlrSF"
names(all)[names(all) == "3SsnPorch"] <- "ThreeSsnPorch"
```

I renamed variables that started with numbers because it causes an error for making model matrix.

2-6. Converting variable into factor

```
library(plyr)
library(forcats)

# Extract colnames that were coded as numeric
num_var <- names(all[,-1])[which(sapply(all[,-1], is.numeric))]
num_var <- num_var[!num_var %in% c("MSSubClass", "OverallQual", "OverallCond")]

# Extract colnames that were coded as character
cat_var <- names(all)[which(sapply(all, is.character))]
cat_var <- c(cat_var, "MSSubClass", "OverallQual", "OverallCond")

# convert MSSubClass, OverallQual, and OverallCond variables into factor
all[,cat_var] <- lapply(all[,cat_var], factor)
```

MSSubClass, OverallQual, and OverallCond variables were coded as numeric, but they should be categorical variable. Thus, I converted them into a factor before plotting categorical variables.

2-7. Replacing NA with 'None'

```
all$Alley <- fct_explicit_na(all$Alley, na_level = "None")
all$BsmtQual <- fct_explicit_na(all$BsmtQual, na_level = "None")
all$BsmtCond <- fct_explicit_na(all$BsmtCond, na_level = "None")
all$BsmtExposure <- fct_explicit_na(all$BsmtExposure, na_level = "None")
all$BsmtFinType1 <- fct_explicit_na(all$BsmtFinType1, na_level = "None")
all$BsmtFinType2 <- fct_explicit_na(all$BsmtFinType2, na_level = "None")
```

```r
all$FireplaceQu <- fct_explicit_na(all$FireplaceQu, na_level = "None")
all$GarageType <- fct_explicit_na(all$GarageType, na_level = "None")
all$GarageFinish <- fct_explicit_na(all$GarageFinish, na_level = "None")
all$GarageQual <- fct_explicit_na(all$GarageQual, na_level = "None")
all$GarageCond <- fct_explicit_na(all$GarageCond, na_level = "None")
all$PoolQC <- fct_explicit_na(all$PoolQC, na_level = "None")
all$Fence <- fct_explicit_na(all$Fence, na_level = "None")
all$MiscFeature <- fct_explicit_na(all$MiscFeature, na_level = "None")
```

Based on the given data description file, I replaced NA with 'None' for Alley, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, FireplaceQu, GarageType, GarageFinish, GarageQual, GarageCond, PoolQC, Fence, MiscFeature because NA means that "there is no such thing".

2-8. Renaming and reordering factor levels

```r
# For easier interpretation, rename factor levels
category_level1 <- c("Po" = 1, "Fa" = 2, "TA" = 3,
                     "Gd" = 4, "Ex" = 5)
category_level2 <- c("None" = 0, "Po" = 1, "Fa" = 2,
                      "TA" = 3, "Gd" = 4, "Ex" = 5)

# Reorder factor levels ordinally
all$ExterQual <- factor(revalue(all$ExterQual, category_level1),
                        levels = c("1", "2", "3", "4", "5"))
all$ExterCond <- factor(revalue(all$ExterCond, category_level1),
                        levels = c("1", "2", "3", "4", "5"))
all$BsmtQual <- fct_explicit_na(all$BsmtQual, na_level = "None")
all$BsmtQual <- factor(revalue(all$BsmtQual, category_level2),
                       levels = c("0", "1", "2", "3", "4", "5"))
all$BsmtCond <- fct_explicit_na(all$BsmtCond, na_level = "None")
all$BsmtCond <- factor(revalue(all$BsmtCond, category_level2),
                       levels = c("0", "1", "2", "3", "4", "5"))
all$HeatingQC <- factor(revalue(all$HeatingQC, category_level1),
                        levels = c("1", "2", "3", "4", "5"))
all$KitchenQual <- factor(revalue(all$KitchenQual, category_level1),
                        levels = c("1", "2", "3", "4", "5"))
all$FireplaceQu <- factor(revalue(all$FireplaceQu, category_level2),
                       levels = c("0", "1", "2", "3", "4", "5"))
all$GarageQual <- factor(revalue(all$GarageQual, category_level2),
                       levels = c("0", "1", "2", "3", "4", "5"))
all$GarageCond <- factor(revalue(all$GarageCond, category_level2),
                       levels = c("0", "1", "2", "3", "4", "5"))
all$PoolQC <- factor(revalue(all$PoolQC, category_level2),
                       levels = c("0", "1", "2", "3", "4", "5"))
all$Fence <- factor(all$Fence, levels = c("None", "MnWw", "GdWo", "MnPrv", "GdPrv"))
```

For easier interpretation and better visualization, I renamed the factor levels and reordered factor levels ordinally

2-9. Resplitting combined data into train and test datasets

```r
# All observations that have non-NA for SalePrice are train dataset
train <- all[-which(is.na(all$SalePrice)),]
```

```
# All observations that have NA for SalePrice are test dataset
test <- all[which(is.na(all$SalePrice)),]
```

2-10. Missing Data in train dataset

```
library(dplyr)
library(tibble)
library(knitr)

# Variables that have missing data
train.missing <- data.frame(cbind(colSums(sapply(train[-81], is.na)), colSums(sapply(train[-81], is.na)),

# Keep row names
train.missing <- train.missing %>% rownames_to_column('new_column')
train.missing <- filter_at(train.missing, vars(-new_column), any_vars(. > 0))
train.missing <- train.missing %>% column_to_rownames('new_column')

colnames(train.missing)[1] <- "# of missing"
colnames(train.missing)[2] <- "% of missing"
train.missing[2] <- round(train.missing[2],3)
kable(train.missing)
```

	# of missing	% of missing
LotFrontage	259	0.177
MasVnrType	8	0.005
MasVnrArea	8	0.005
Electrical	1	0.001
GarageYrBlt	81	0.055

All other variables that are not included in the table above have 0 missing value. Since the percentage of missingness is not that large (Max: 17.7%), I decided to remove all observations from the data, which have a missing value in one or more variables (listwise deletion).

2-11. Listwise deletion

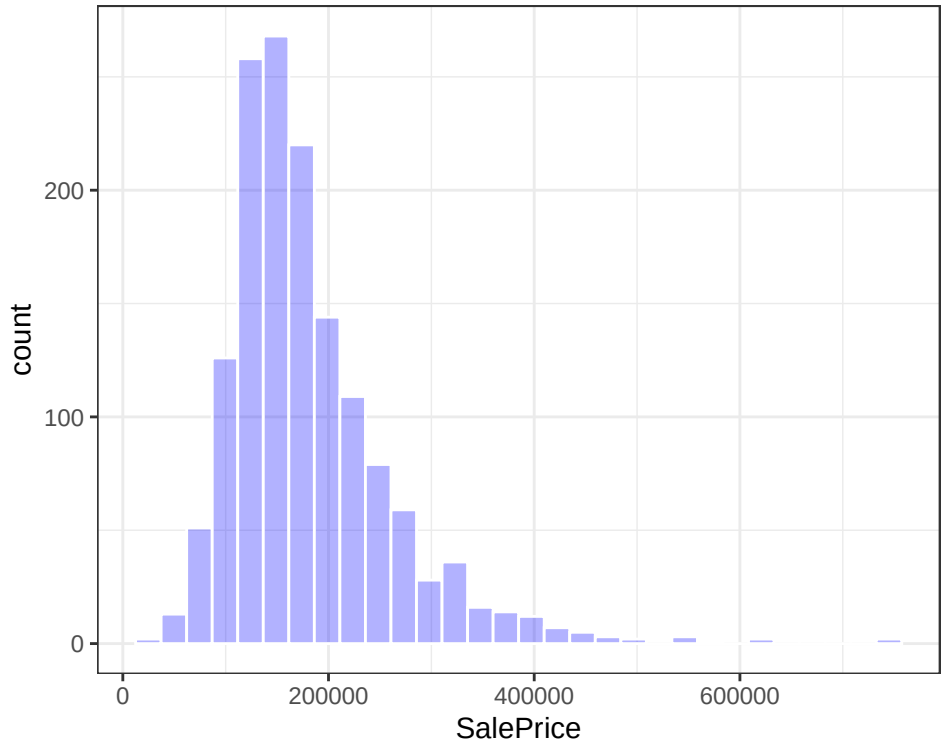
```
# Duplicate all datasets to keep original dataset
all.md <- all
train.md <- train
test.md <- test

train.md <- train.md[complete.cases(train.md), ]
test.md <- test.md[complete.cases(test.md[,-81]), ]
```

The sample size reduced from 1460 to 1120 and 1459 to 1139, respectively, after removing missing values.

2-12. Exploratory Data Analysis for target variable, SalePrice

```
options(scipen = 999)
library(ggplot2)

# Distribution of SalePrice
```

```
ggplot(train, aes(SalePrice)) + geom_histogram(color = 'white',
                                               fill = 'Blue', alpha = 0.3) + theme_bw()
```




```
# Basic numerical summaries (min, max, mean, median, 1st & 3rd quartiles)
summary(train$SalePrice)
```

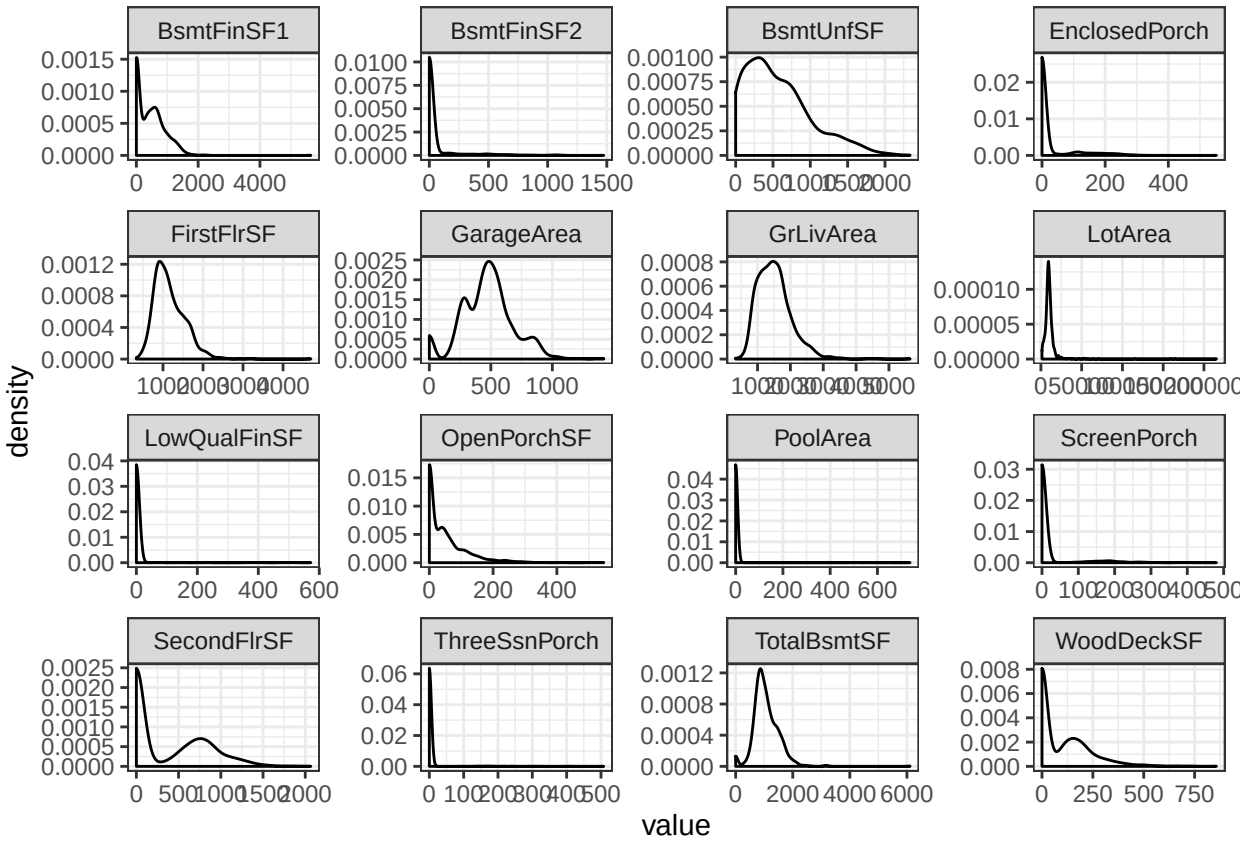
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   34900  129975  163000  180921  214000  755000
```

The Saleprice is higly right skewed. The average of sale price is $180,921, while the median of sale price is $163,000.

2-12. Exploratory Data Analysis for predictors

2-12-1. Numeric variables describing size of a house in square feet

```
library(purrr)
library(tidyr)
library(ggplot2)
library(knitr)
library(corrplot)

# Distribution of variables representing size of a house in square feet
train[,c("LotArea", "BsmtFinSF1", "BsmtFinSF2", "BsmtUnfSF", "TotalBsmtSF",
              "FirstFlrSF", "SecondFlrSF", "LowQualFinSF", "GrLivArea", "GarageArea",
              "WoodDeckSF", "OpenPorchSF", "EnclosedPorch", "ThreeSsnPorch",
              "ScreenPorch", "PoolArea")] %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
   facet_wrap(~ key, scales = "free") +
   geom_density() + theme_bw()
```



The density plots show that some of the variables have very similar distribution, which made me take a closer look because this can cause multicollinearity issues later.

```
# Combine the variables into one data frame to take a look closely
size.SF <- data.frame(cbind(train$LotArea, train$BsmtFinSF1, train$BsmtFinSF2,
                            train$BsmtUnfSF, train$TotalBsmtSF, train$FirstFlrSF,
                            train$`SecondFlrSF`, train$LowQualFinSF, train$GrLivArea,
```

```
                          train$GarageArea, train$WoodDeckSF, train$OpenPorchSF,
                          train$EnclosedPorch, train$`ThreeSsnPorch`, train$ScreenPorch,
                          train$PoolArea))

colnames(size.SF) <- c("LotArea", "BsmtFinSF1", "BsmtFinSF2",
                       "BsmtUnfSF", "TotalBsmtSF", "FirstFlrSF",
                       "SecondFlrSF", "LowQualFinSF", "GrLivArea",
                       "GarageArea", "WoodDeckSF", "OpenPorchSF",
                       "EnclosedPorch", "ThreeSsnPorch", "ScreenPorch","PoolArea")

# Sum BsmtFinSF1, BsmtFinSF2, BsmtUnfSF to compare to TotalBsmtSF
size.SF <- cbind(size.SF, rowSums(size.SF[,c(2,3,4)]))
colnames(size.SF)[17] <- "BsmtSF"
kable(head(size.SF[,c(2,3,4,17,5)]))
```

BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	BsmtSF	TotalBsmtSF
706	0	150	856	856
978	0	284	1262	1262
486	0	434	920	920
216	0	540	756	756
655	0	490	1145	1145
732	0	64	796	796

```
# Verify BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF = TotalBsmtSF
isFALSE(size.SF$BsmtSF == size.SF$TotalBsmtSF)
```

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

```
# Sum FirstFlrSF, SecondFlrSF, LowQualFinSF to compare to GrLivArea
size.SF <- cbind(size.SF, rowSums(size.SF[,c(6,7,8)]))
colnames(size.SF)[18] <- "GrLivSF"
kable(head(size.SF[,c(6,7,8,18,9)]))
```

FirstFlrSF	SecondFlrSF	LowQualFinSF	GrLivSF	GrLivArea
856	854	0	1710	1710
1262	0	0	1262	1262
920	866	0	1786	1786
961	756	0	1717	1717
1145	1053	0	2198	2198
796	566	0	1362	1362

```
# Verify FirstFlrSF + SecondFlrSF + LowQualFinSF = GrLivArea
isFALSE(size.SF$GrLivSF == size.SF$GrLivArea)
```

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

Based on the EDA, I realized the following:

- BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF = TotalBsmtSF
- FirstFlrSF + SecondFlrSF + LowQualFinSF = GrLivArea

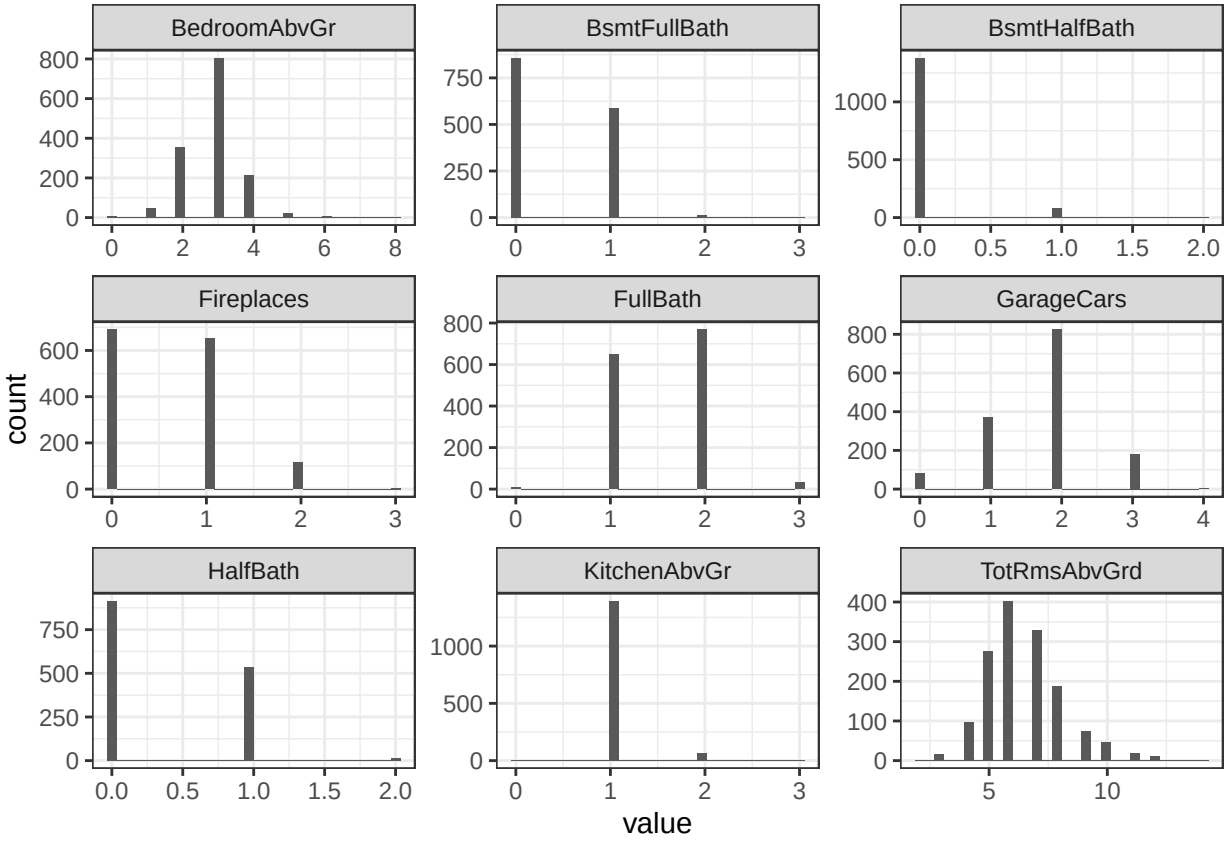
The tables (first observations) and isFalse function verified it as well. Thus, I removed the duplicated variables below.

2-12-2. Removing variables - SF variables

```
drops <- c("BsmtFinSF1", "BsmtFinSF2", "BsmtUnfSF",
                  "FirstFlrSF", "SecondFlrSF", "LowQualFinSF")
all.md <- all.md[ , !(names(all.md) %in% drops)]
train.md <- train.md[ , !(names(train.md) %in% drops)]
test.md <- test.md[ , !(names(test.md) %in% drops)]
```

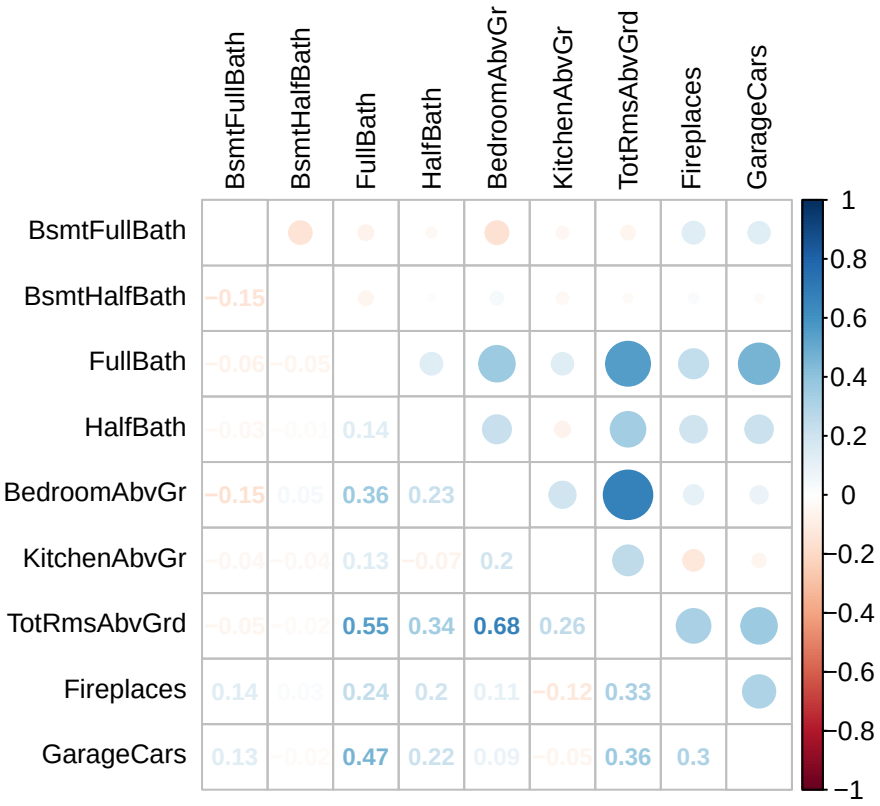
2-12-3. Numeric variables describing size of a house in integer

```
# Distribution of variables representing size of a house in integer
train[,c("BsmtFullBath", "BsmtHalfBath", "FullBath", "HalfBath",
         "BedroomAbvGr", "KitchenAbvGr", "TotRmsAbvGrd", "Fireplaces", "GarageCars")] %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
   facet_wrap(~ key, scales = "free") +
   geom_histogram() + theme_bw()
```




```
# Combine the variables into one data frame to take a look closely
size.intgr <- data.frame(cbind(train$BsmtFullBath, train$BsmtHalfBath, train$FullBath,
                               train$HalfBath, train$BedroomAbvGr, train$KitchenAbvGr,
                               train$TotRmsAbvGrd, train$Fireplaces, train$GarageCars))

colnames(size.intgr) <- c("BsmtFullBath", "BsmtHalfBath", "FullBath",
                          "HalfBath", "BedroomAbvGr", "KitchenAbvGr",
                          "TotRmsAbvGrd", "Fireplaces", "GarageCars")
```

```
cor_size.intgr <- cor(size.intgr, size.intgr)
corrplot.mixed(cor_size.intgr,
               tl.col="black",
               tl.pos = "lt",
               tl.cex = 0.8,
               number.cex = 0.75)
```



There are multiple variables that unnecessarily represent the number of bathrooms, so I decided to combine them into one variable.

TotRmsAbvGrd (Total rooms above grade (does not include bathrooms)) and Bedroom (Bedrooms above grade (does NOT include basement bedrooms)) have very similar distribution, and they are highly correlated. This seems logical because TotRmsAbvGrd includes Bedroom and Kitchen. Considering multicollinearity issues, TotRmsAbvGrd can be separated into 1) Kitchen, 2) bedroom, and 3) room for other purposes variables, and then TotRmsAbvGrd will be removed.

2-12-4. Combinding and categorizing variable - Bathroom / Room

```
# Combine bathroom variable into one variable
all.md$TotBath <- all.md$BsmtFullBath + all.md$BsmtHalfBath*0.5 +
  all.md$FullBath + all.md$HalfBath*0.5

train.md$TotBath <- train.md$BsmtFullBath + train.md$BsmtHalfBath*0.5 +
  train.md$FullBath + train.md$HalfBath*0.5

test.md$TotBath <- test.md$BsmtFullBath + test.md$BsmtHalfBath*0.5 +
  test.md$FullBath + test.md$HalfBath*0.5

# Separate TotRmsAbvGrd into 1) Kitchen, 2) bedroom, and 3) room for other purpose
```

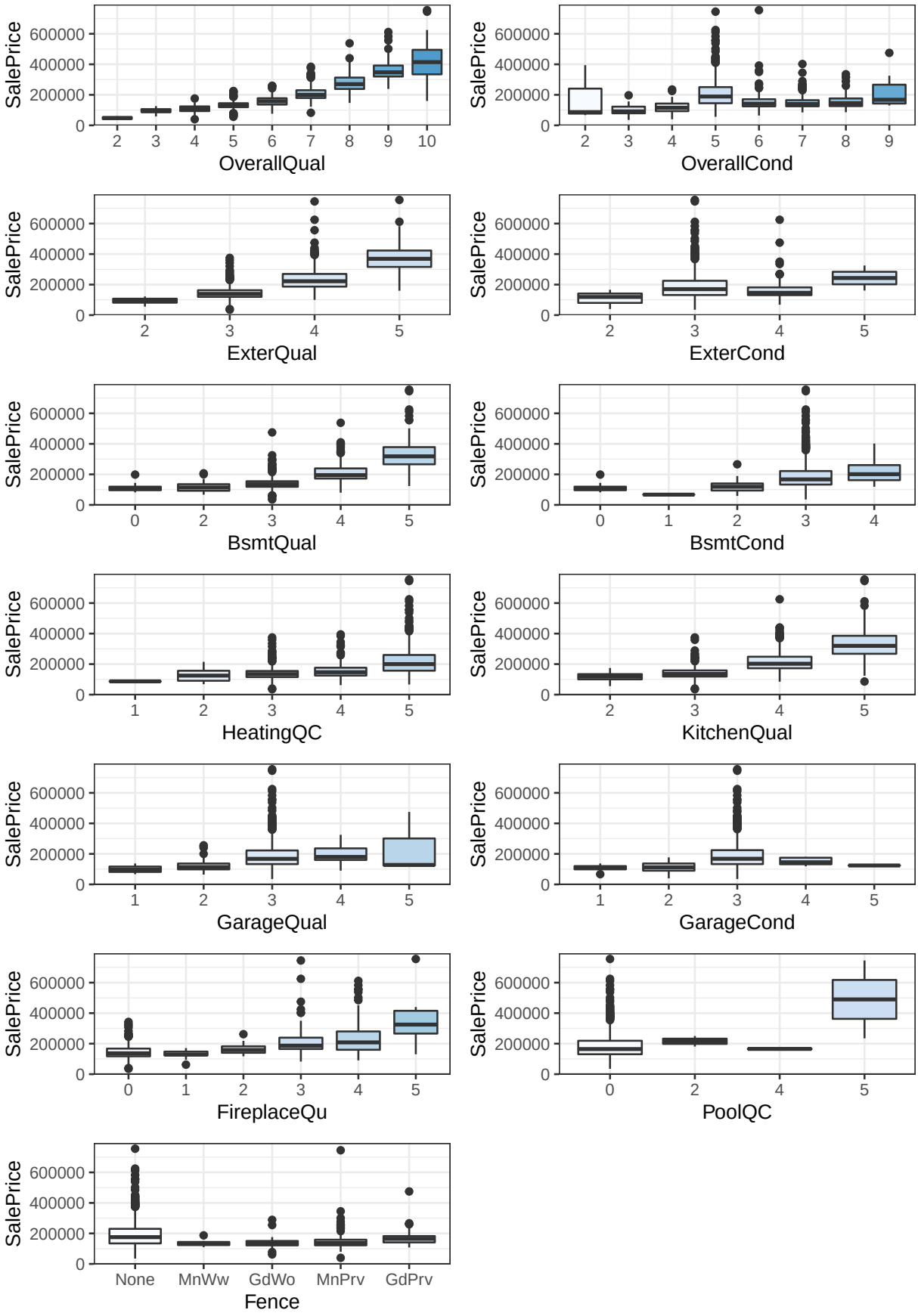
```
all.md$RoomOtherPurp <- all.md$TotRmsAbvGrd - all.md$KitchenAbvGr - all.md$BedroomAbvGr
train.md$RoomOtherPurp <- train.md$TotRmsAbvGrd - train.md$KitchenAbvGr - train.md$BedroomAbvGr
test.md$RoomOtherPurp <- test.md$TotRmsAbvGrd - test.md$KitchenAbvGr - test.md$BedroomAbvGr

drops <- c("BsmtFullBath", "BsmtHalfBath", "FullBath", "HalfBath", "TotRmsAbvGrd")
all.md <- all.md[ , !(names(all.md) %in% drops)]
train.md <- train.md[ , !(names(train.md) %in% drops)]
test.md <- test.md[ , !(names(test.md) %in% drops)]
```

For total number of bathroom, I calculated it by treating a half bathroom as 0.5 and a full bathroom as 1. I subtract number of kitchen and bedroom from total room (TotRmsAbvGrd) to calculate the number of room for other purpose.

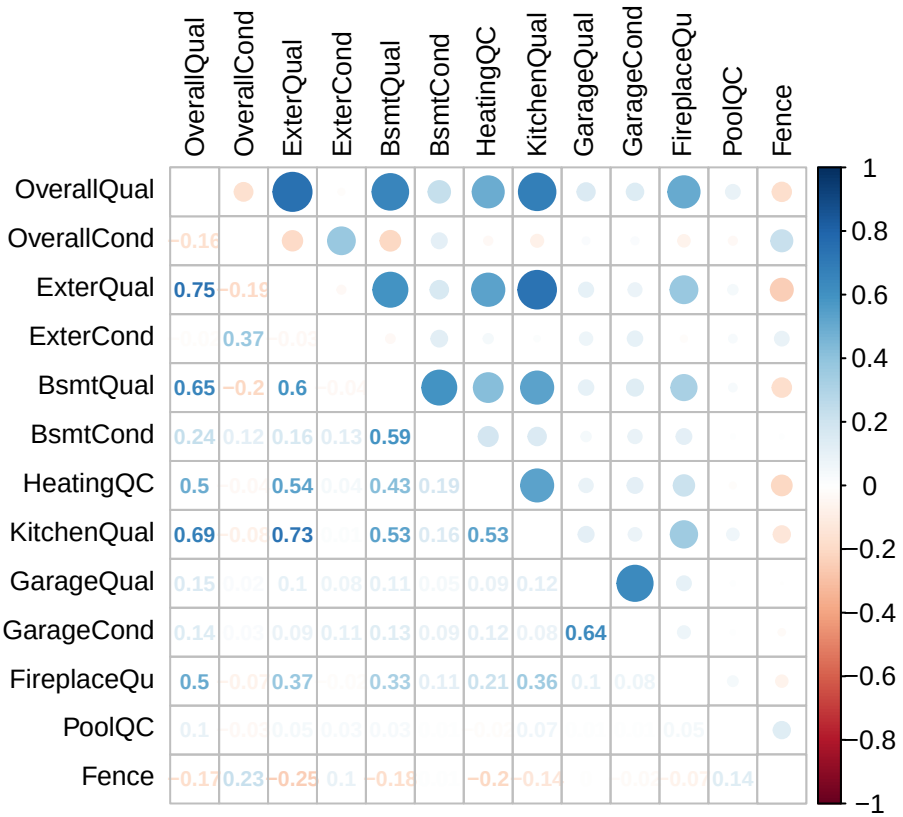
2-12-5. Categorical variables describing quality of a house

I did not include the code for ggplot because it's too long.

```
grid.arrange(OverallQual.plot, OverallCond.plot,
             ExterQual.plot, ExterCond.plot,
             BsmtQual.plot, BsmtCond.plot,
             HeatingQC.plot, KitchenQual.plot,
             GarageQual.plot, GarageCond.plot,
             FireplaceQu.plot, PoolQC.plot,
             Fence.plot,
             ncol = 2)
```


SalePrice
0
200000
400000
600000
OverallQual
2 3 4 5 6 7 8 9 10
OverallCond
2 3 4 5 6 7 8 9
ExterQual
2 3 4 5
ExterCond
2 3 4 5
BsmtQual
0 2 3 4 5
BsmtCond
0 1 2 3 4
HeatingQC
1 2 3 4 5
KitchenQual
2 3 4 5
GarageQual
1 2 3 4 5
GarageCond
1 2 3 4 5
FireplaceQu
0 1 2 3 4 5
PoolQC
0 2 4 5
Fence
None MnWw GdWo MnPrv GdPrv

```r
# convert factor to numeric to calculate correlation
quality <- as.data.frame(cbind(as.numeric(train.md$OverallQual),
                               as.numeric(train.md$OverallCond),
                               as.numeric(train.md$ExterQual),
                               as.numeric(train.md$ExterCond),
                               as.numeric(train.md$BsmtQual),
                               as.numeric(train.md$BsmtCond),
                               as.numeric(train.md$HeatingQC),
                               as.numeric(train.md$KitchenQual),
                               as.numeric(train.md$GarageQual),
                               as.numeric(train.md$GarageCond),
                               as.numeric(train.md$FireplaceQu),
                               as.numeric(train.md$PoolQC),
                               as.numeric(train.md$Fence)))
colnames(quality) <- c("OverallQual", "OverallCond", "ExterQual", "ExterCond", "BsmtQual",
                       "BsmtCond", "HeatingQC", "KitchenQual", "GarageQual", "GarageCond",
                       "FireplaceQu", "PoolQC", "Fence")

cor_quality <- cor(quality, quality)
corrplot.mixed(cor_quality, tl.col="black", tl.pos = "lt",
               tl.cex = 0.8, number.cex = 0.65)
```



OverallQual variable seems correlated with most of quality variables. Although it could be reasonable to remove the OverallQual variable, I decided to keep it because OverallQual was highly associated with SalePrice, as we can see from the boxplot above.

ExterCond and KitchenQual are also highly correlated to each other, but since these are important factor in determining house price and they evaluate two distinct quality, I decided to keep them.

Since (BsmtQual and BsmtCond) and (GarageQual and GarageCond) represent similar features, I decided to remove the BsmtCond and GarageCond variables.

2-12-6. Removing variables - BsmtCond / GarageCond

```
drops <- c("BsmtCond", "GarageCond")
all.md <- all.md[ , !(names(all.md) %in% drops)]
train.md <- train.md[ , !(names(train.md) %in% drops)]
test.md <- test.md[ , !(names(test.md) %in% drops)]
```

I removed BsmtCond and GarageCond variables.

2-12-7. Variables describing year of build

```
year <- data.frame(cbind(train$YearBuilt, train$YearRemodAdd, train$GarageYrBlt))
colnames(year) <- c("YearBuilt", "YearRemodAdd", "GarageYrBlt")
kable(cor(year, use="pairwise.complete.obs"))
```

	YearBuilt	YearRemodAdd	GarageYrBlt
YearBuilt	1.0000000	0.5928550	0.8256675
YearRemodAdd	0.5928550	1.0000000	0.6422768
GarageYrBlt	0.8256675	0.6422768	1.0000000

This implies that GarageYrBlt and YearBuilt are highly correlated. I decided to remove GarageYrBlt because the year house was built would be more important in determining house price than year garage was built.

2-12-8. Removing variables - GarageYrBlt

```
drops <- c("GarageYrBlt")
all.md <- all.md[ , !(names(all.md) %in% drops)]
train.md <- train.md[ , !(names(train.md) %in% drops)]
test.md <- test.md[ , !(names(test.md) %in% drops)]
```

I removed YearBuilt and GarageYrBlt variables.

2-12-9. Numeric variables describing garage

```
garage <- data.frame(cbind(train$GarageCars, train$GarageArea))
colnames(garage) <- c("GarageCars", "GarageArea")
kable(cor(garage, use="pairwise.complete.obs"))
```

	GarageCars	GarageArea
GarageCars	1.0000000	0.8824754
GarageArea	0.8824754	1.0000000

This implies that GarageCars and GarageArea are highly correlated. Since both represent size of garage, I decided to remove GarageArea.

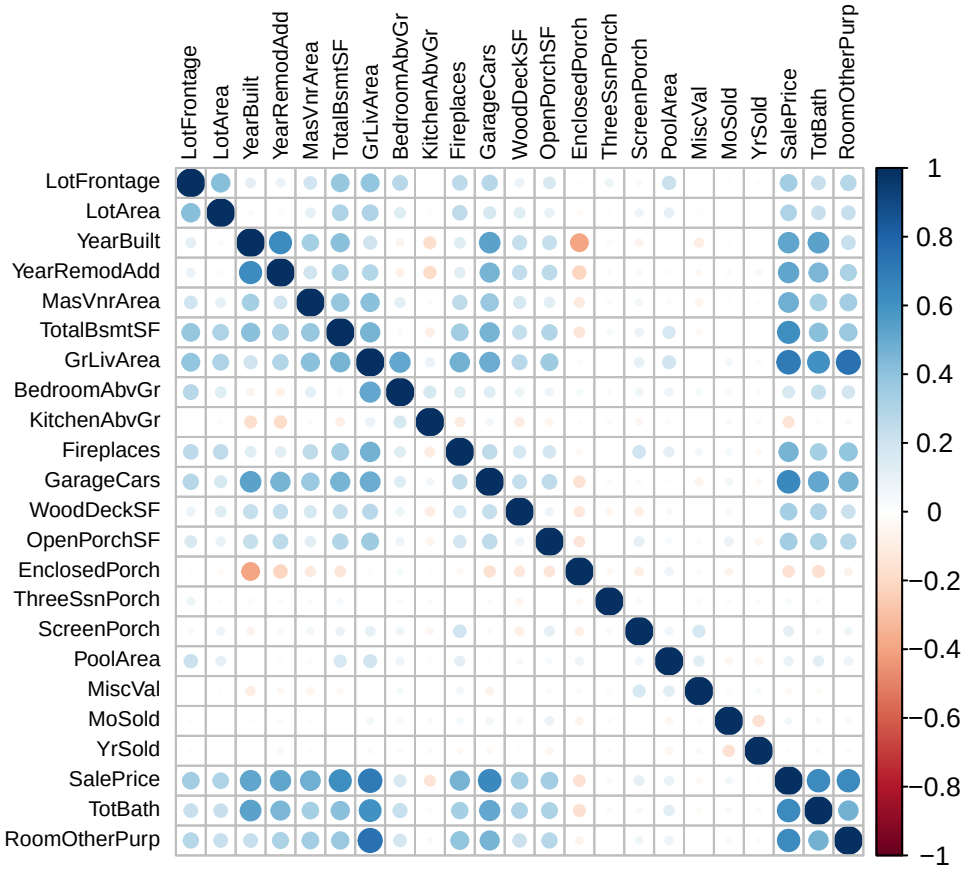
2-12-10. Removing variable - GarageArea

```
drops <- c("GarageArea")
all.md <- all.md[ , !(names(all.md) %in% drops)]
train.md <- train.md[ , !(names(train.md) %in% drops)]
test.md <- test.md[ , !(names(test.md) %in% drops)]
```

I removed GarageArea variable.

2-13. Checking correlation between numeric variables after data cleaning

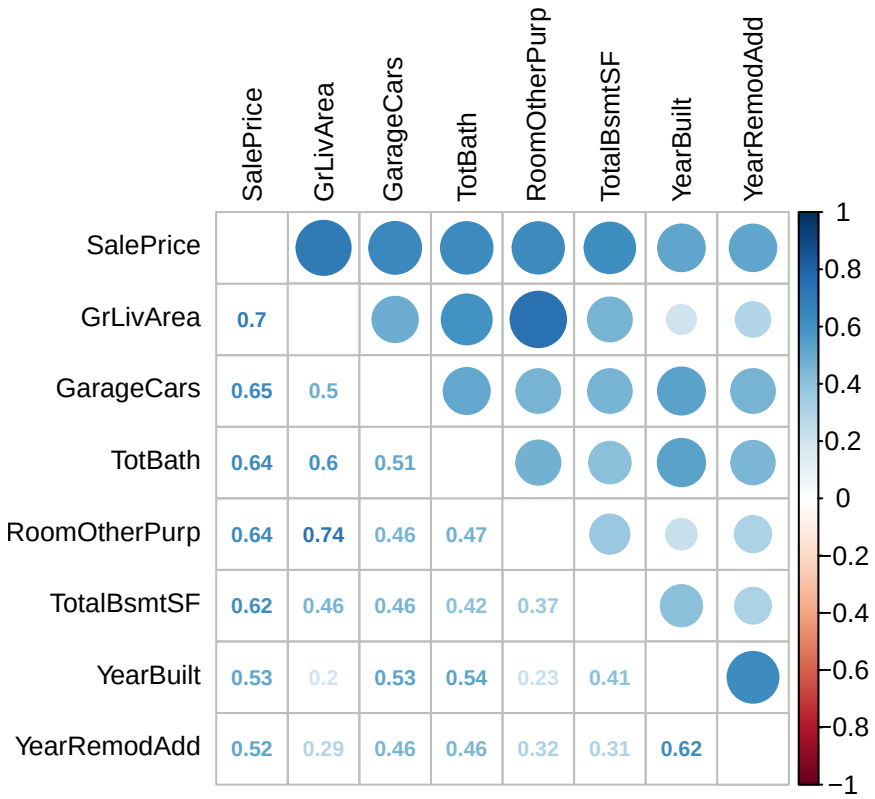
```
library(corrplot)

# Extract colnames for numeric variables
num_train <- train.md[,-1][which(sapply(train.md[,-1], is.numeric))]
# Listwise deletion (removes rows with missing observations) correlation
num_cor <- round(cor(num_train),3)
# Correlation plot for all numeric variables
corrplot(num_cor, tl.col = "black", tl.cex = 0.70)
```



2-14. Correlation with SalePrice

```
# Sort correlation with SalePrice from high to low
SalePrice_cor <- as.matrix(sort(num_cor[,"SalePrice"], decreasing = TRUE))
# Extract colnames for variables that have at least 0.5 correlation with the SalePrice
SalePrice_high_cor <- names(which(apply(SalePrice_cor, 1, function(x) abs(x)>0.5)))
```

```r
# Correlation matrix to plot
SalePrice_correlation <- num_cor[SalePrice_high_cor, SalePrice_high_cor]

# Correlation plot
corrplot.mixed(SalePrice_correlation, tl.col="black",
               tl.pos = "lt", tl.cex = 0.8, number.cex = 0.7)
```



Based on the analyis correlation associated with SalePrice, GrLivArea, GarageCars, TotBath, RoomOtherPurp, TotalBsmtSF, OverallQual, YearBuilt, YearRemodAdd should be considered to predict the target variable, SalePrice.

2-15. Boxplot for categorical variables

I plotted boxplots for all categorical variables, but it quite hard to tell the relationship between other variables and SalePrice by just looking at the plots. Thus, I did not include each plot here.

3. Data Split

3-1. Splitting training into 2 sets for validation

```
# Set seed for reproducibility
set.seed(31221)

## From cleaned dataset
# Generate a sample of 1/4 from the train dataset for test block
train.md.25 <- sample(1: nrow(train.md), nrow(train.md)/4)
# Rest of 3/4 for traing block
train.md.75 <- (-train.md.25)

train.md.25test <- train.md[train.md.25,]
train.md.75train <- train.md[train.md.75,]
```

In order to make sure the model is valid and generalizable to the new dataset, I created the training block and test block by randomly splitting the samples from the training dataset into two blocks. Within the same training dataset, I randomly chose a 75% to make the training block. The rest of 25% became the test block. Since the SalePrices in the test block are accurate, I calculated the test error by comparing the SalePrices with predicted SalePrices using the model, which I will discuss in the method section.

4. Methods

Random Forest and Lasso are performed in order to identify which method yields the smallest test error.

4-1. Random Forests

Random forests is a supervised learning algorithm, and it uses trees as building blocks to construct more powerful prediction model. The tree-Based Method has two steps:

1) Divide the predictor space into X distinct and non-overlapping regions.
2) For every observation that falls into each region, make the same prediction, which is simply the mean of the response values for the training observation in that region.

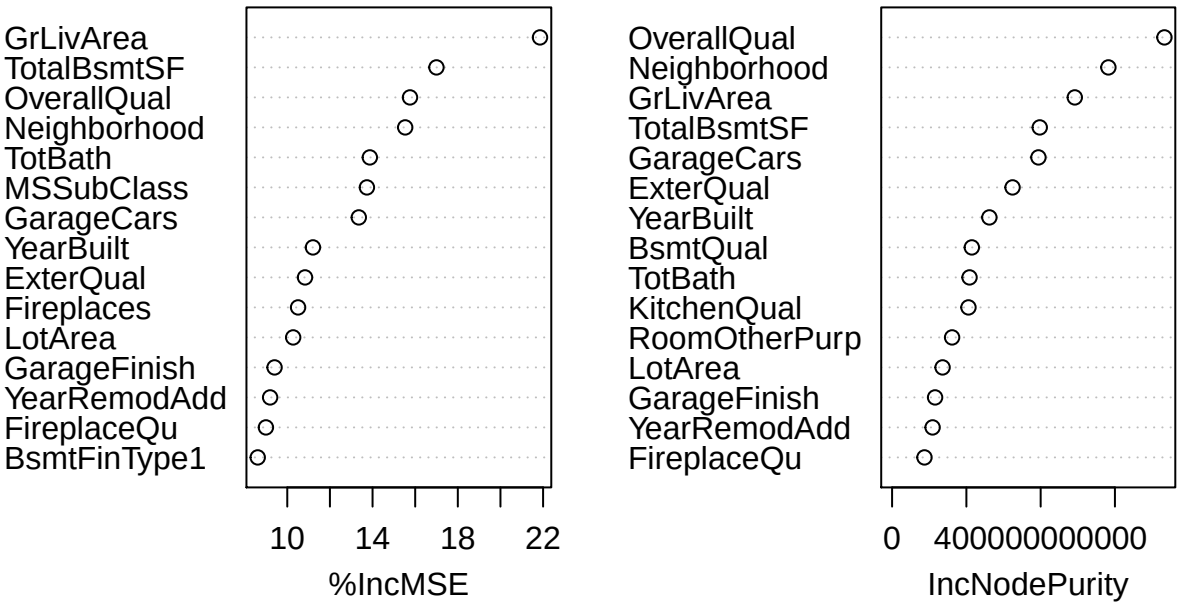
Random forests force each split to consider only a subset of the predictors, so it prevents building tree with highly correlated predictors, which makes the tree more reliable. Decision trees for regression is easier to interpret and has a nice graphical representation. However, it can be very non-robust, which means a small change in the data can cause a large change in the final estimated tree.

```
# Set seed for reproducibility
set.seed(31221)
library(randomForest)

# Implement random forest
rf.tree <- randomForest(SalePrice ~. -Id,
                        data = train.md,
                        subset = train.md.75,
                        mtry = 9,
                        importance = TRUE)
# Importance of each variable
importance(rf.tree)
```

```
##                  %IncMSE IncNodePurity
## MSSubClass    13.7346932   77630901490
## MSZoning       7.8989230   14937782014
## LotFrontage    3.1093302   77321101976
## LotArea       10.2759953  135909986785
## Street         0.7236914     415668659
## Alley          3.2708669    2794512730
## LotShape       0.1724454   15265017836
## LandContour    1.6019990   17881745093
## Utilities      0.0000000             0
## LotConfig      0.9264990   14025584057
## LandSlope     -0.3033776    6483122425
## Neighborhood  15.5309312  582384989119
## Condition1     2.9952373   10169575595
## Condition2    -0.8195744    1844009751
## BldgType       3.7713734    5888891440
## HouseStyle     5.9531619   19007994765
## OverallQual   15.7563975  733214316193
## OverallCond    8.5013674   23474822431
## YearBuilt     11.2041463  261899054032
## YearRemodAdd   9.1951392  109015512029
## RoofStyle      2.5461860   28414886383
## RoofMatl       1.2200038    2929184492
## Exterior1st    6.7022843   61811027889
## Exterior2nd    4.1451711   83408326838
```

```
## MasVnrType     4.2098574   17694076042
## MasVnrArea     4.9908770   84833404169
## ExterQual     10.8319652  324282989839
## ExterCond      2.5325898    6433942739
## Foundation     5.8982734   57357224626
## BsmtQual       7.9471391  214676817123
## BsmtExposure   1.9415624   41926169865
## BsmtFinType1   8.6142876   41633630128
## BsmtFinType2  -0.4496637    7317643117
## TotalBsmtSF   16.9975732  397552002403
## Heating        1.3410582    2017576167
## HeatingQC      4.4202594   28116732519
## CentralAir     6.9515385    5253530250
## Electrical     0.1154163    2490113463
## GrLivArea     21.8541653  491872256861
## BedroomAbvGr   6.7374930   33997937579
## KitchenAbvGr   2.5332578    1471888961
## KitchenQual    8.4735090  205649934996
## Functional    -0.4022521    4397533206
## Fireplaces    10.5067248   77349769380
## FireplaceQu    8.9979669   87115683305
## GarageType     6.7990803   43994762864
## GarageFinish   9.4032142  115701269968
## GarageCars    13.3551745  394024930102
## GarageQual     1.6681168    3122436998
## PavedDrive     1.2690323    2382186764
## WoodDeckSF     5.4199248   38350233496
## OpenPorchSF    7.4678330   65791956675
## EnclosedPorch  2.8920621    4659906663
## ThreeSsnPorch -0.6395569    1929410378
## ScreenPorch    0.9535523   17378021569
## PoolArea      -2.3768281   13318359556
## PoolQC         1.0010015   14699657919
## Fence          3.9952010   17587623915
## MiscFeature   -0.2797848     551007288
## MiscVal        0.9114930    1416266632
## MoSold         0.6239028   26212561916
## YrSold        -0.6199331   14695793776
## SaleType       2.0618417   14792233902
## SaleCondition  3.9815498   20429106762
## TotBath       13.8724086  208492498166
## RoomOtherPurp  8.4052250  161556577210
```

```r
# Plot of the importance measures
varImpPlot(rf.tree, n.var = 15, main = "Variable Importance")
```



To perform random forests, 9 out of 69 predictors are considered for each split of the tree (mtry = 9). This is because a small value of predictor subset size in building a random forest will typically be helpful when there is a large number of correlated predictors. Based on the 'An Introduction to Statistical Learning' textbook, sqrt(number of predictor) is recommended as a predictor subset size.

Random Forests also report importance measures. Based on the %IncMSE (Mean decrease of accuracy in predictions on the out of bag samples when a given variable is excluded from the model; the higher number, the more important), IncNodePurity (total decrease in node impurity that results from splits over that variable, averaged over all trees), and the plot, the important variables seems below:

GrLivArea, TotalBsmtSF, OverallQual, Neighborhood, TotBath, MSSubClass, GarageCars, YearBuilt, ExterQual, Fireplaces, LotArea, GarageFinish, YearRemodAdd, FireplaceQu, BsmtFinType1, YearBuilt, BsmtQual, KitchenQual, RoomOtherPurp.

```
# Predict SalePrice using the random forest model
rf.pred <- predict(rf.tree, newdata = train.md.25test)
# Test set MSE
rf.test.error <- mean((rf.pred - train.md.25test$SalePrice)^2)
```

The test set MSE is 959091256. Let's see whether random forests yield an improvement over Lasso.

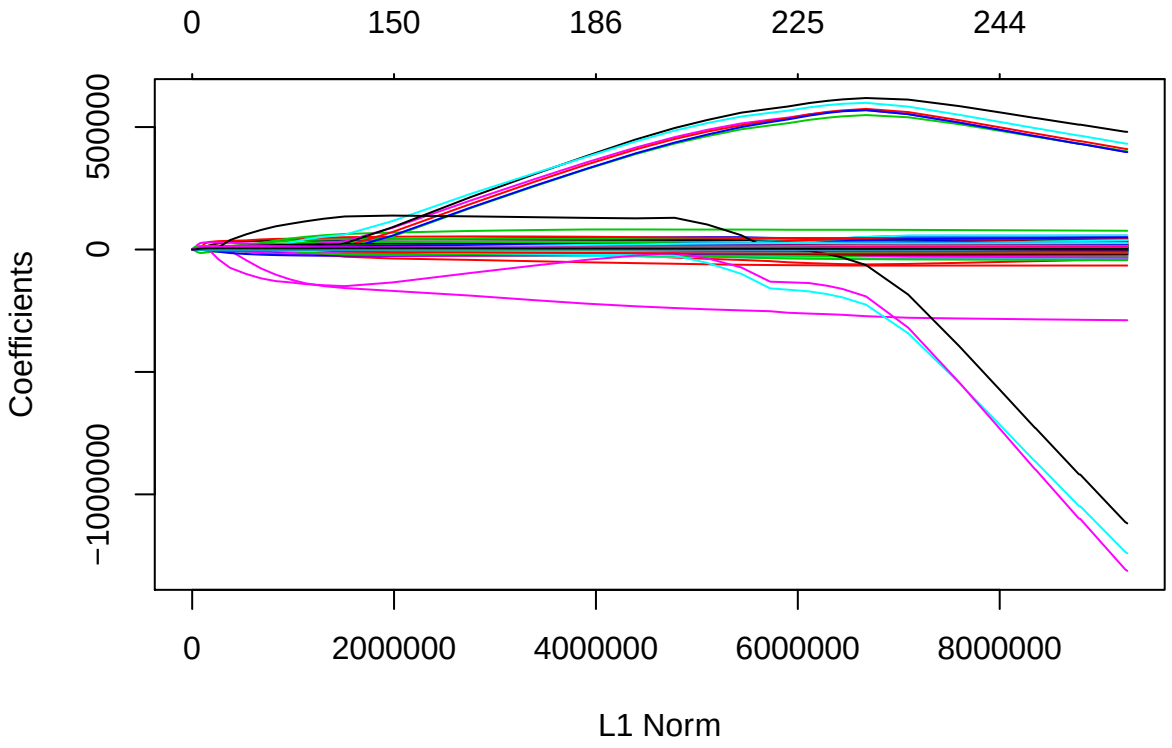
4-2. Lasso

Ridge and Lasso is the best-known techniques for shrinking the regression coefficients by regularizing the coefficient estimates towards zero. I decided to fit a model using Lasso because the Lasso can set any of them exactly to zero, which means the lasso performs variable selection and the lasso is easier to interpret.

The lasso leads to qualitatively similar behavior to ridge regression, in that as the tuning parameter increases, the variance decreases and the bias increases. In other words, neither ridge regression nor the lasso regression will universallyl dominate the other. In general, one might expect the lasso to perform better in a setting where a relatively small number of predictors have substantial coefficients, and the remaining predictors have

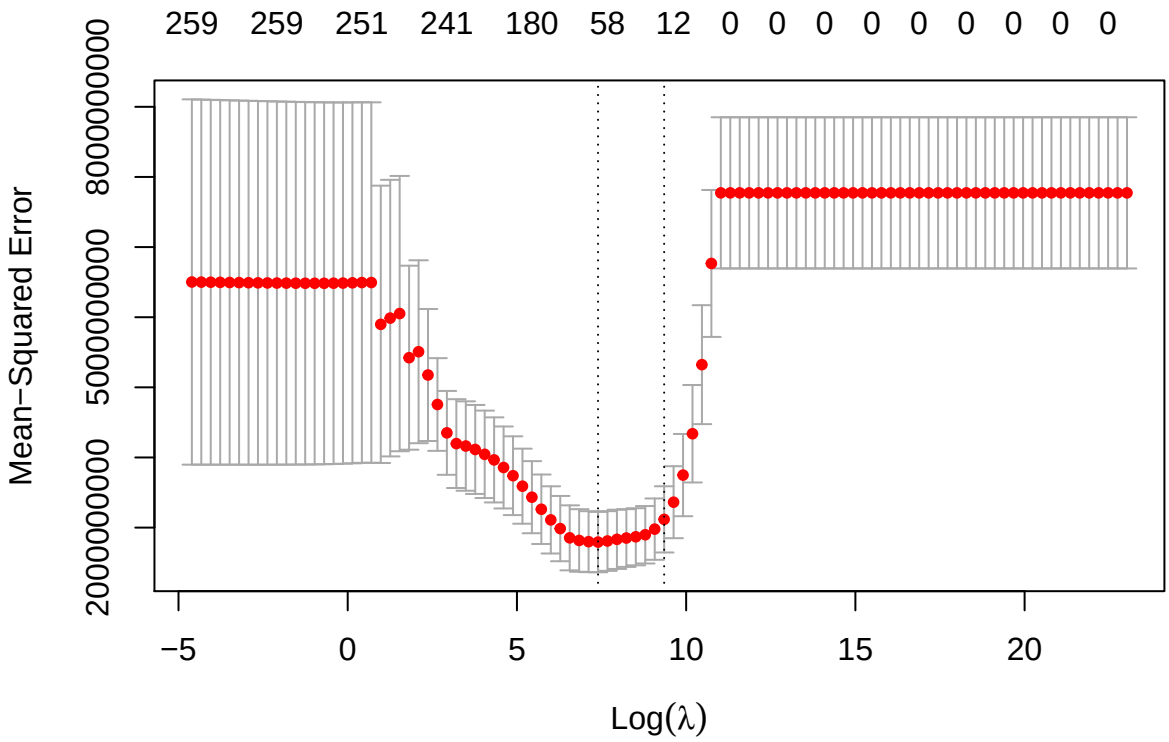
coefficients that are very small or that equal zero. Ridge regression will perform better when the response is a function of many predictors, all with coefficients of roughly equal size.

```
# Set seed for reproducibility
set.seed(31221)
library(glmnet)

# Response variable vector
y.train <- train.md.75train$SalePrice
# Produce a matrix corresponding to the predictors
x.train <- model.matrix( ~. -1 -Id -SalePrice, data = train.md.75train)
x.test <- model.matrix( ~. -1 -Id -SalePrice, data = train.md.25test)
# Covering the full range of scenarios from the null model
grid <- 10^seq(10, -2, length = 100)


# Fit a lasso regression model
# If alpha=0 then a ridge regression model is fit, and if alpha=1 then a lasso model is fit
lasso.mod <- glmnet(x.train, y.train, alpha = 1, lambda = grid)
plot(lasso.mod)
```



The coefficient plot shows that some of the coefficients are exactly equal to zero depending on the choice of tuning parameter. I performed cross-validation below to choose appropriate tuning parameter.

```
# cross-validation to choose tuning parameter
lasso.cv.mod <- cv.glmnet(x.train, y.train, alpha = 1, lambda = grid)
plot(lasso.cv.mod)
```



The plot includes the cross-validation curve (red dotted line), and upper and lower standard deviation curves along the tunning parameter sequence (error bars). It recommends that the number of variables seems appropriate between 12 to 58 (number on the top) including dummy variables.

```
# The smallest cross-validation error
bestlam <- lasso.cv.mod$lambda.min

# Predict SalePrice using the Lasso
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x.test)
# Test set MSE
lasso.test.error <- mean((lasso.pred - train.md.25test$SalePrice)^2) #635669314
```

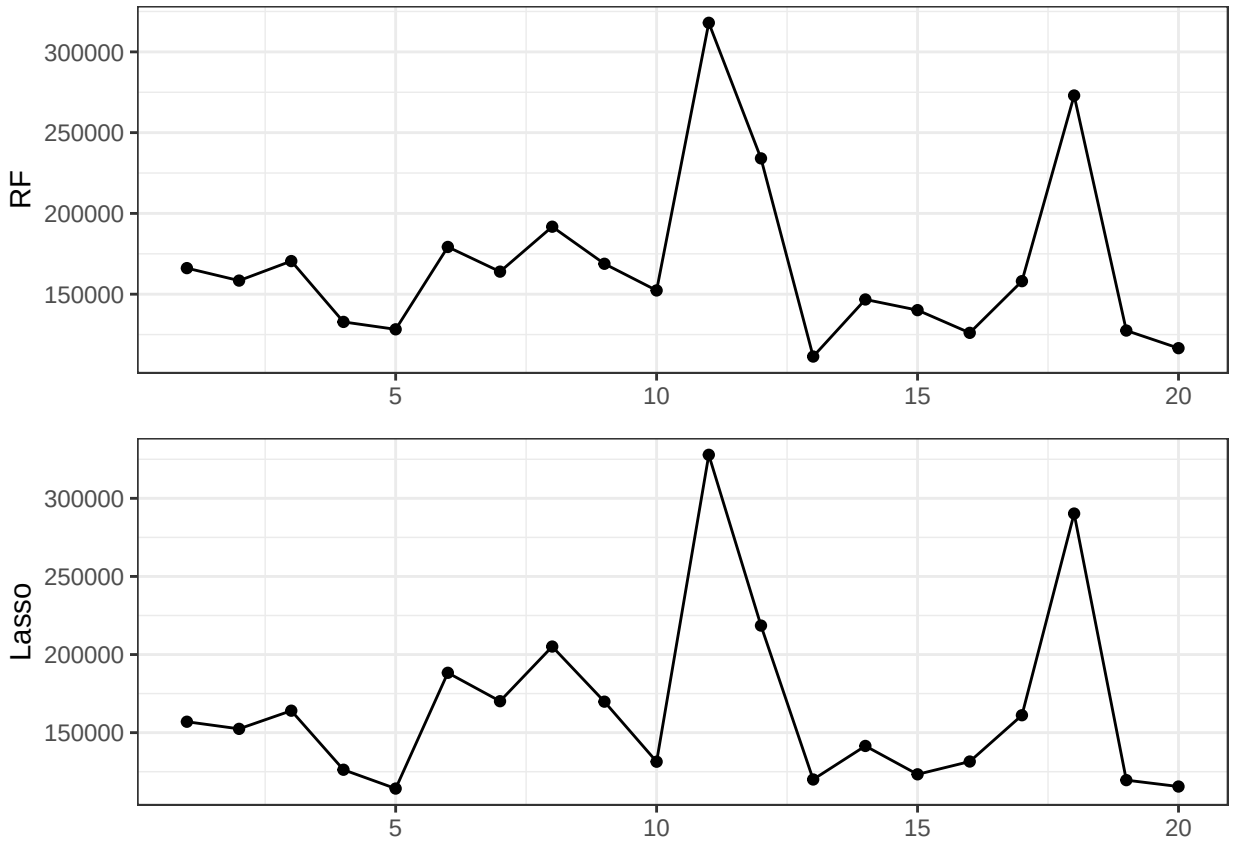
The smallest cross-validation error 1630, and the test MSE 775154586. The test MSE is higher than the test MSE obtained using random forests. Thus, Lasso did better than the random forests.

5. Results

5-1. Summary of Random Forests and Lasso

The test MSEs obtained using random forest and Lasso are 959091256 and 775154586, respectively. Thus, my final model is Lasso.

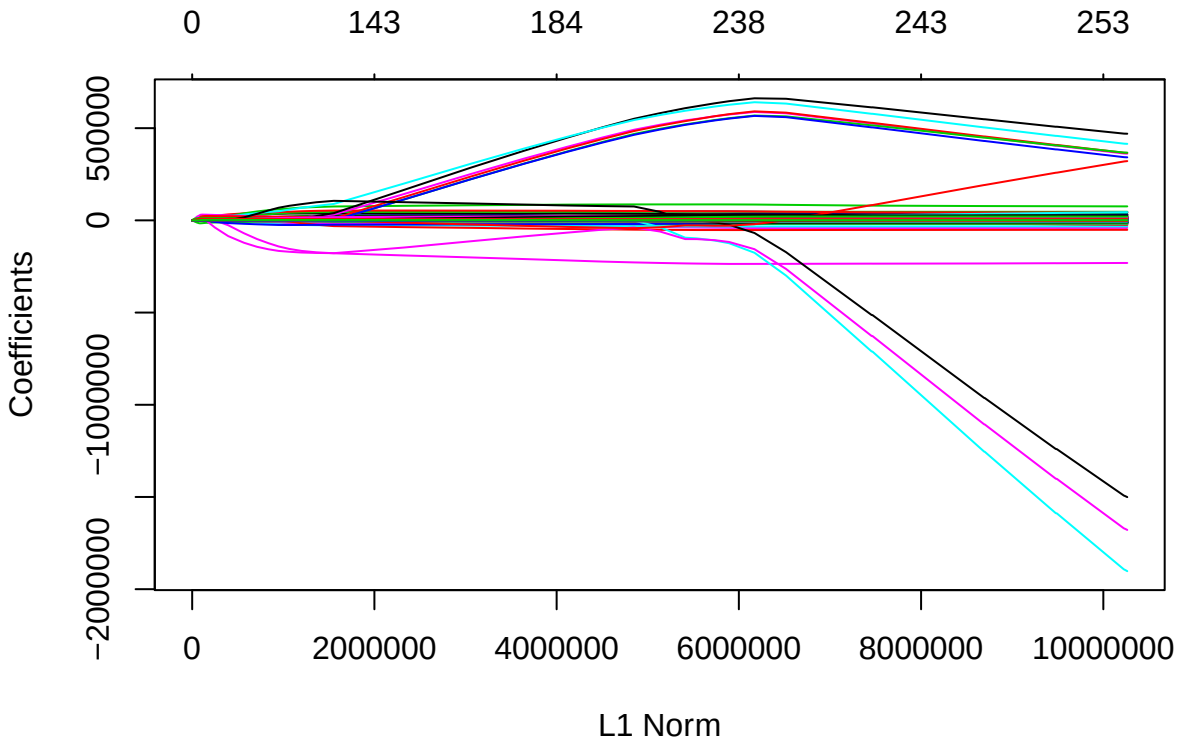
```
# First 20 predicted house prices
head.rf.pred <- cbind(1:20, data.frame(rf.pred[1:20]))
head.lasso.pred <- cbind(1:20, data.frame(lasso.pred[1:20]))

# Plot 20 predicted house prices using Random Forest and Lasso
plot.rf <- ggplot(head.rf.pred, aes(x=`1:20`, y = `rf.pred.1.20.`)) +
  geom_point() + geom_line() + ylab("RF") +
  theme_bw() + theme(axis.title.x = element_blank())
plot.lasso <- ggplot(head.lasso.pred, aes(x=`1:20`, y = `lasso.pred.1.20.`)) +
  geom_point() + geom_line() + ylab("Lasso") +
  theme_bw() + theme(axis.title.x = element_blank())

grid.arrange(plot.rf, plot.lasso, ncol = 1)
```



The trends of predicted house prices based on the two models seems very similar.

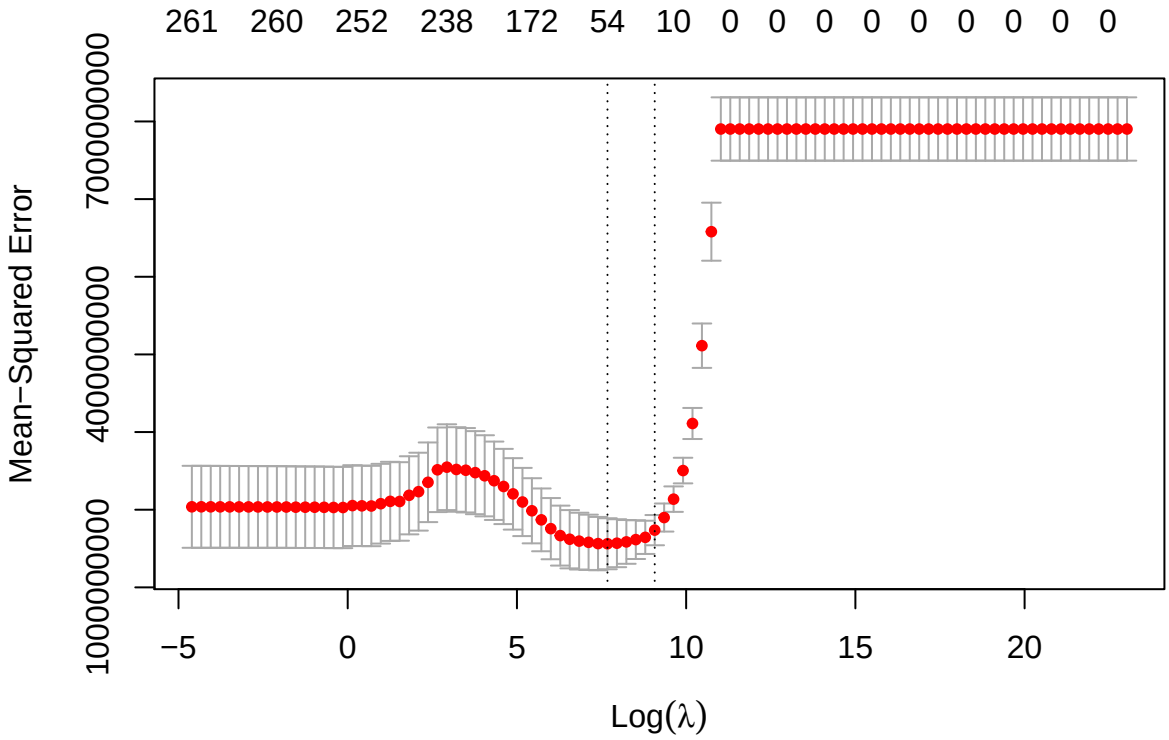
5-2. Fit Lasso model to the whole available data

```
# Set seed for reproducibility
set.seed(31221)
library(glmnet)

# Response variable vector
y.train <- train.md$SalePrice
# Produce a matrix corresponding to the predictors
x.train <- model.matrix( ~. -1 -Id -SalePrice, data = train.md)
x.test <- model.matrix( ~. -1 -Id, data = test.md[,-68])
# Covering the full range of scenarios from the null model
grid <- 10^seq(10, -2, length = 100)


# Fit a lasso regression model
# If alpha=0 then a ridge regression model is fit, and if alpha=1 then a lasso model is fit
lasso.mod <- glmnet(x.train, y.train, alpha = 1, lambda = grid)
plot(lasso.mod)
```



The coefficient plot shows that some of the coefficients are exactly equal to zero depending on the choice of tuning parameter. I performed cross-validation below to choose appropriate tuning parameter.

```
# cross-validation to choose tuning parameter
lasso.cv.mod <- cv.glmnet(x.train, y.train, alpha = 1, lambda = grid)
plot(lasso.cv.mod)
```



The plot includes the cross-validation curve (red dotted line), and upper and lower standard deviation curves along the tunning parameter sequence (error bars). It recommends that the number of variables seems appropriate between 10 to 54 (number on the top) including dummy variables.

```
# The smallest cross-validation error
bestlam <- lasso.cv.mod$lambda.min

# Predict SalePrice using the Lasso
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x.test)
```

I applied the lasso using the entire training dataset to predict the test dataset.

6. Conclusion

```
# variables that over 10000
mycoef <- coef(lasso.mod, s=bestlam);
myresult <- data.frame(cbind(mycoef@Dimnames[[1]][which(mycoef > 10000)], round(mycoef[which(mycoef > 1
colnames(myresult) <- c("Variable", "Coefficients")
kable(myresult)
```

Variable	Coefficients
LotConfigCulDSac	12098
NeighborhoodCrawfor	15084
NeighborhoodNoRidge	33961
NeighborhoodNridgHt	16088
NeighborhoodStoneBr	32759
OverallQual8	16563
OverallQual9	44172
OverallQual10	55266
RoofMatlWdShngl	45384
BsmtQual5	20445
BsmtExposureGd	13454
KitchenQual5	20214
GarageCars	10547
PoolQC5	53913
SaleTypeNew	11326

The variables that determine SalePrice significantly were slightly different from the variables from the Random Forest. Based on the Lasso, OverallQual, Neighboorhood, PoolQC, RoofMatl, BsmtQual, KitchenQual were important.

Specifically, these are the variables that increase sale price more than 10,000.

- LotConfig - Cul-de-sac
- Neighboorhood - Crawford, Northridge, Northridge Heights, Stone Brook
- Condition1 - Normal
- OverallQual - Very Good, Excellent, Very Excellent
- RoofMatl - Wood Shingles
- BsmtQual - Excellent (100+ inches)
- BsmtExposure - Good Exposure
- KitchenQual - Excellent
- GarageCars - Bigger size of garage is better
- PoolQC - Excellent
- SaleType - Home just constructed and sold