Linear Regression Analysis: Regression Case Study

Neerja Doshi, Sri Santhosh Hari, Ker-Yu Ong, Nicha Ruchirawat

Part I: Explanatory Modelling

Task 0: Exploratory Data Analysis and Data Cleaning

```
# rawDF <-
# read.csv('/Users/booranium/usf/601_regression/project/housing.txt',
# stringsAsFactors = T)
rawDF <- read.csv("/Users/santhoshhari/Documents/Coursework/LinearRegression/IowaHousing/Data/housing.t
    stringsAsFactors = T)
# rawDF <- read.csv('housing.txt', stringsAsFactors = T)
```

The Iowa housing dataset contains 1460 rows and 81 variables, a glimpse of which is as follows:

```
str(rawDF)
```

```
## 'data.frame':    1460 obs. of  81 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     : Factor w/ 5 levels "C (all)","FV",..: 4 4 4 4 4 4 4 4 5 4 ...
##  $ 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       : Factor w/ 2 levels "Grvl","Pave": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alley        : Factor w/ 2 levels "Grvl","Pave": NA NA NA NA NA NA NA NA NA NA ...
##  $ LotShape     : Factor w/ 4 levels "IR1","IR2","IR3",..: 4 4 1 1 1 1 4 1 4 4 ...
##  $ LandContour  : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Utilities    : Factor w/ 2 levels "AllPub","NoSeWa": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LotConfig    : Factor w/ 5 levels "Corner","CulDSac",..: 5 3 5 1 3 5 5 1 5 1 ...
##  $ LandSlope    : Factor w/ 3 levels "Gtl","Mod","Sev": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood : Factor w/ 25 levels "Blmngtn","Blueste",..: 6 25 6 7 14 12 21 17 18 4 ...
##  $ Condition1   : Factor w/ 9 levels "Artery","Feedr",..: 3 2 3 3 3 3 3 5 1 1 ...
##  $ Condition2   : Factor w/ 8 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ BldgType     : Factor w/ 5 levels "1Fam","2fmCon",..: 1 1 1 1 1 1 1 1 1 2 ...
##  $ HouseStyle   : Factor w/ 8 levels "1.5Fin","1.5Unf",..: 6 3 6 6 6 1 3 6 1 2 ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
##  $ RoofStyle    : Factor w/ 6 levels "Flat","Gable",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ RoofMatl     : Factor w/ 8 levels "ClyTile","CompShg",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Exterior1st  : Factor w/ 15 levels "AsbShng","AsphShn",..: 13 9 13 14 13 13 13 7 4 9 ...
##  $ Exterior2nd  : Factor w/ 16 levels "AsbShng","AsphShn",..: 14 9 14 16 14 14 14 7 16 9 ...
##  $ MasVnrType   : Factor w/ 4 levels "BrkCmn","BrkFace",..: 2 3 2 3 2 3 4 4 3 3 ...
##  $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 4 3 4 3 4 4 4 ...
##  $ ExterCond    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ Foundation   : Factor w/ 6 levels "BrkTil","CBlock",..: 3 2 3 1 3 6 3 2 1 1 ...
##  $ BsmtQual     : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 3 3 4 3 3 1 3 4 4 ...
##  $ BsmtCond     : Factor w/ 4 levels "Fa","Gd","Po",..: 4 4 4 2 4 4 4 4 4 4 ...
```

```
##  $ BsmtExposure : Factor w/ 4 levels "Av","Gd","Mn",..: 4 2 3 4 1 4 1 3 4 4 ...
##  $ BsmtFinType1 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 3 1 3 1 3 3 3 1 6 3 ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 6 6 6 6 6 6 6 2 6 6 ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : Factor w/ 6 levels "Floor","GasA",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ HeatingQC    : Factor w/ 5 levels "Ex","Fa","Gd",..: 1 1 1 3 1 1 1 1 3 1 ...
##  $ CentralAir   : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical   : Factor w/ 5 levels "FuseA","FuseF",..: 5 5 5 5 5 5 5 5 2 5 ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 3 3 4 3 4 4 4 ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : Factor w/ 7 levels "Maj1","Maj2",..: 7 7 7 7 7 7 7 7 3 7 ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : Factor w/ 5 levels "Ex","Fa","Gd",..: NA 5 5 3 5 NA 3 5 5 5 ...
##  $ GarageType   : Factor w/ 6 levels "2Types","Attchd",..: 2 2 2 6 2 2 2 2 6 2 ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : Factor w/ 3 levels "Fin","RFn","Unf": 2 2 2 3 2 3 2 2 3 2 ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 2 3 ...
##  $ GarageCond   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ PavedDrive   : Factor w/ 3 levels "N","P","Y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : Factor w/ 3 levels "Ex","Fa","Gd": NA NA NA NA NA NA NA NA NA NA ...
##  $ Fence        : Factor w/ 4 levels "GdPrv","GdWo",..: NA NA NA NA NA 3 NA NA NA NA ...
##  $ MiscFeature  : Factor w/ 4 levels "Gar2","Othr",..: NA NA NA NA NA 3 NA 3 NA NA ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : Factor w/ 9 levels "COD","Con","ConLD",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 1 5 5 5 5 1 5 ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```

At first glance, we see that most of the variables are categorical - both numeric and character types - and only a handful are continuous. The response variable for our analysis is `SalePrice`, and the remaining 79 variables (excluding the record `ID` column) are considered potential predictor variables. Checking the data dictionary, we found the following distribution for the predictor variables:

- 49 categorical
- 19 are continuous, e.g. area, price
- 11 are discrete, e.g. count, year

There are 0 duplicate rows in the dataset.

Handling NA Values

Below, we compute that number and percentage of `NA`s per variable in the dataset having at least 1 `NA`.

```
NA_columns <- colnames(rawDF)[unique(which(is.na(rawDF), arr.ind = T)[,
    2])]

NA_count <- rawDF %>% dplyr::select(NA_columns) %>% summarise_all(funs(sum(is.na(.)))) %>%
    gather(key = "Variable", value = "num_na", everything()) %>% arrange(desc(num_na))

NA_count %<>% mutate(perc_na = paste(round(num_na/nrow(rawDF), 4) *
    100, "%"))
colnames(NA_count) <- c("**Variable**", "**Number of NA**", "**Percentage of NA**")
row.names(NA_count) <- NULL
knitr::kable(NA_count, caption = "\\label{tab:NACount} Variable NA Count and Percentage",
    format.args = list(big.mark = ","))
```

Table 1: Variable NA Count and Percentage

Variable	**Number of NA**	**Percentage of NA**
PoolQC	1,453	99.52 %
MiscFeature	1,406	96.3 %
Alley	1,369	93.77 %
Fence	1,179	80.75 %
FireplaceQu	690	47.26 %
LotFrontage	259	17.74 %
GarageType	81	5.55 %
GarageYrBlt	81	5.55 %
GarageFinish	81	5.55 %
GarageQual	81	5.55 %
GarageCond	81	5.55 %
BsmtExposure	38	2.6 %
BsmtFinType2	38	2.6 %
BsmtQual	37	2.53 %
BsmtCond	37	2.53 %
BsmtFinType1	37	2.53 %
MasVnrType	8	0.55 %
MasVnrArea	8	0.55 %
Electrical	1	0.07 %

The data dictionary tells us that for most of the fields in Table 1, `NA` is actually meaningful, indicating non-applicability or a lack of the feature rather than missing data. After checking the data dictionary for the meaning of each field, we imputed - for every categorical variable for which `NA` was meaningful - `NA`s with `0`s.

```
# Create a copy of rawDF to be our working data frame
housingDF <- rawDF

# Update NAs with 0s for applicable fields
```

```r
levels(housingDF$PoolQC) <- c("0", levels(housingDF$PoolQC))
housingDF$PoolQC[is.na(housingDF$PoolQC)] <- "0"
levels(housingDF$MiscFeature) <- c("0", levels(housingDF$MiscFeature))
housingDF$MiscFeature[is.na(housingDF$MiscFeature)] <- "0"
levels(housingDF$Alley) <- c("0", levels(housingDF$Alley))
housingDF$Alley[is.na(housingDF$Alley)] <- "0"
levels(housingDF$Fence) <- c("0", levels(housingDF$Fence))
housingDF$Fence[is.na(housingDF$Fence)] <- "0"
levels(housingDF$FireplaceQu) <- c("0", levels(housingDF$FireplaceQu))
housingDF$FireplaceQu[is.na(housingDF$FireplaceQu)] <- "0"
levels(housingDF$GarageType) <- c("0", levels(housingDF$GarageType))
housingDF$GarageType[is.na(housingDF$GarageType)] <- "0"
levels(housingDF$GarageFinish) <- c("0", levels(housingDF$GarageFinish))
housingDF$GarageFinish[is.na(housingDF$GarageFinish)] <- "0"
levels(housingDF$GarageQual) <- c("0", levels(housingDF$GarageQual))
housingDF$GarageQual[is.na(housingDF$GarageQual)] <- "0"
levels(housingDF$GarageCond) <- c("0", levels(housingDF$GarageCond))
housingDF$GarageCond[is.na(housingDF$GarageCond)] <- "0"
levels(housingDF$BsmtExposure) <- c("0", levels(housingDF$BsmtExposure))
housingDF$BsmtExposure[is.na(housingDF$BsmtExposure)] <- "0"
levels(housingDF$BsmtFinType2) <- c("0", levels(housingDF$BsmtFinType2))
housingDF$BsmtFinType2[is.na(housingDF$BsmtFinType2)] <- "0"
levels(housingDF$BsmtQual) <- c("0", levels(housingDF$BsmtQual))
housingDF$BsmtQual[is.na(housingDF$BsmtQual)] <- "0"
levels(housingDF$BsmtCond) <- c("0", levels(housingDF$BsmtCond))
housingDF$BsmtCond[is.na(housingDF$BsmtCond)] <- "0"
levels(housingDF$BsmtFinType1) <- c("0", levels(housingDF$BsmtFinType1))
housingDF$BsmtFinType1[is.na(housingDF$BsmtFinType1)] <- "0"
```

We then re-check the count and percentage of `NA`s per variable left in the dataset.

```r
NA_columns <- colnames(housingDF)[unique(which(is.na(housingDF), arr.ind = T)[,
    2])]

NA_count <- housingDF %>% dplyr::select(NA_columns) %>% summarise_all(funs(sum(is.na(.)))) %>%
    gather(key = "Variable", value = "num_na", everything()) %>% arrange(desc(num_na))

NA_count %<>% mutate(perc_na = paste(round(num_na/nrow(housingDF),
    4) * 100, "%"))
colnames(NA_count) <- c("**Variable**", "**Number of NA**", "**Percentage of NA**")
row.names(NA_count) <- NULL
knitr::kable(NA_count, caption = "\\label{tab:NACount1} Variable NA Count and Percentage(after replacin
    format.args = list(big.mark = ","))
```

Table 2: Variable NA Count and Percentage(after replacing NAs with 0s, where appropriate)

Variable	**Number of NA**	**Percentage of NA**
LotFrontage	259	17.74 %
GarageYrBlt	81	5.55 %
MasVnrType	8	0.55 %
MasVnrArea	8	0.55 %
Electrical	1	0.07 %

```r
colnames(housingDF)
```

```
##  [1] "Id"            "MSSubClass"    "MSZoning"      "LotFrontage"
##  [5] "LotArea"       "Street"        "Alley"         "LotShape"
##  [9] "LandContour"   "Utilities"     "LotConfig"     "LandSlope"
## [13] "Neighborhood"  "Condition1"    "Condition2"    "BldgType"
## [17] "HouseStyle"    "OverallQual"   "OverallCond"   "YearBuilt"
## [21] "YearRemodAdd"  "RoofStyle"     "RoofMatl"      "Exterior1st"
## [25] "Exterior2nd"   "MasVnrType"    "MasVnrArea"    "ExterQual"
## [29] "ExterCond"     "Foundation"    "BsmtQual"      "BsmtCond"
## [33] "BsmtExposure"  "BsmtFinType1"  "BsmtFinSF1"    "BsmtFinType2"
## [37] "BsmtFinSF2"    "BsmtUnfSF"     "TotalBsmtSF"   "Heating"
## [41] "HeatingQC"     "CentralAir"    "Electrical"    "X1stFlrSF"
## [45] "X2ndFlrSF"     "LowQualFinSF"  "GrLivArea"     "BsmtFullBath"
## [49] "BsmtHalfBath"  "FullBath"      "HalfBath"      "BedroomAbvGr"
## [53] "KitchenAbvGr"  "KitchenQual"   "TotRmsAbvGrd"  "Functional"
## [57] "Fireplaces"    "FireplaceQu"   "GarageType"    "GarageYrBlt"
## [61] "GarageFinish"  "GarageCars"    "GarageArea"    "GarageQual"
## [65] "GarageCond"    "PavedDrive"    "WoodDeckSF"    "OpenPorchSF"
## [69] "EnclosedPorch" "X3SsnPorch"    "ScreenPorch"   "PoolArea"
## [73] "PoolQC"        "Fence"         "MiscFeature"   "MiscVal"
## [77] "MoSold"        "YrSold"        "SaleType"      "SaleCondition"
## [81] "SalePrice"
```

Table 2 shows the list of remaining variables where `NA` indicates missing data. We impute `NA`s in these variables with

- mean of the data, for continuous variables (LotFrontage)
- median of the data, for discrete variables (GarageYrBlt)
- mode of the data, for categorical variables (MasVnrType, Electrical)

```r
# Function to get mode of data
getmode <- function(v) {
    uniqv <- unique(v)
    uniqv[which.max(tabulate(match(v, uniqv)))]
}

# Impute NAs
housingDF$LotFrontage[is.na(housingDF$LotFrontage)] <- mean(housingDF$LotFrontage,
    na.rm = T)
housingDF$GarageYrBlt[is.na(housingDF$GarageYrBlt)] <- median(housingDF$GarageYrBlt,
    na.rm = T)
housingDF$MasVnrType[is.na(housingDF$MasVnrType)] <- getmode(housingDF$MasVnrType)
housingDF$MasVnrArea[is.na(housingDF$MasVnrArea)] <- 0
housingDF$Electrical[is.na(housingDF$Electrical)] <- getmode(housingDF$Electrical)

# Convert MSSubClass to factor
housingDF$MSSubClass <- factor(housingDF$MSSubClass)
housingDF$MoSold <- factor(housingDF$MoSold)
```

Since Masonry veneer area (`MasVnrArea`) is directly related to `MasVnrType`, we impute for area based on the mode of `MasVnrType`, which is None. Our cleaned dataset is named `housingDF`.

Exploratory Data Visualization

With our clean dataset, we perform exploratory data visualization of the distribution of key measures such as volume and sale price of houses by what we hypothesize to be key predictor variables.

To begin with, we check the distribution of sale prices using a histogram and box-plot.

```
# hist(housingDF$SalePrice, main = 'Histogram of Sale Price')
# boxplot(housingDF$SalePricem, main = 'Boxplot of Sale Price')
summary(housingDF$SalePrice)
```

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

Intuition suggests the neighborhood is a key determining factor in a house's sale price, hence below, we plot the distribution of sale price by neighborhood.

```
housingDF %>% dplyr::select(Neighborhood, SalePrice) %>% ggplot(aes(factor(Neighborhood),
    SalePrice)) + geom_boxplot() + theme(axis.text.x = element_text(angle = 90,
    hjust = 1)) + xlab("Neighborhoods")
```

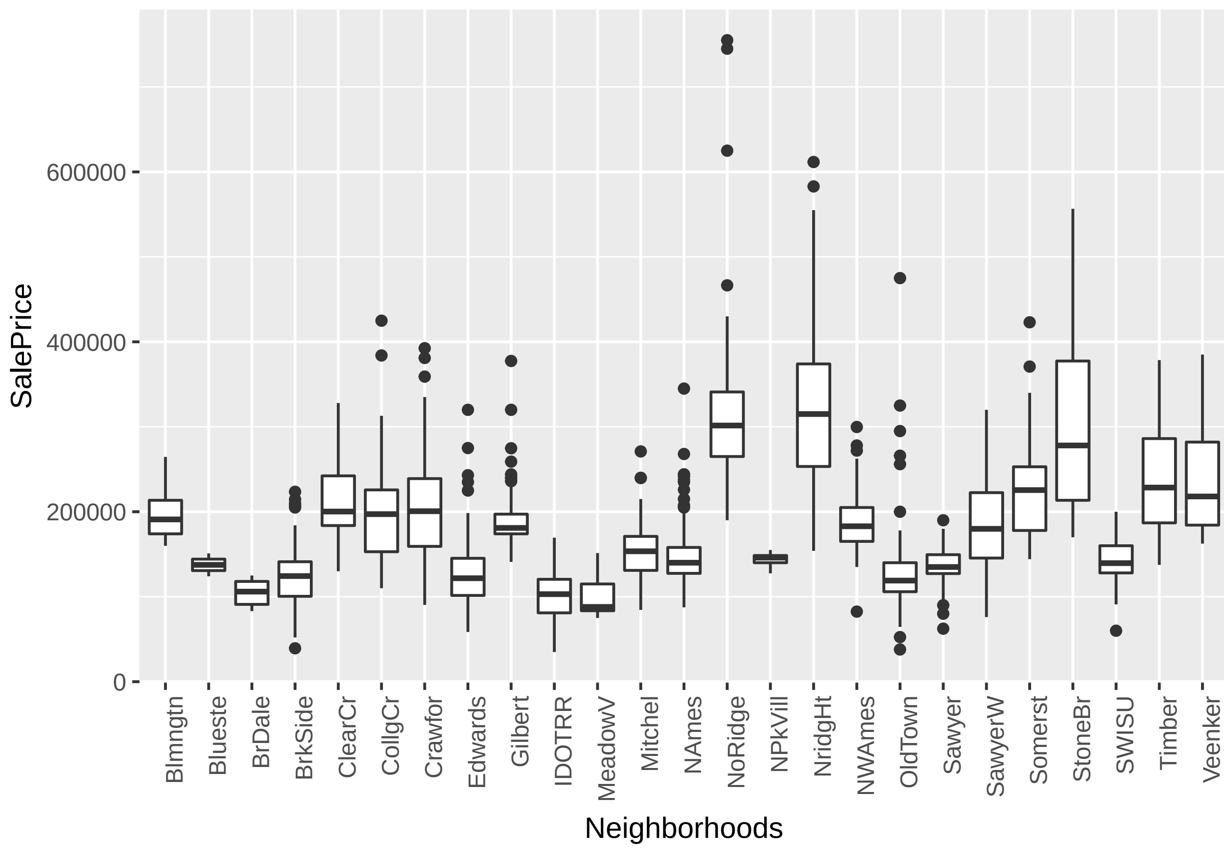


Figure 1: SalePrice distribution per neighborhood

From Figure 1, we can observe that Brookside and Meadow Vista have the lowest median house price while Northridge and Northridge Height have the heighest median house price as well as several outliers.

We then distribution of houses by a number of key features we hypothesize to be important in determining housing price: the property's zoning class (`MSZoning`), type of road access to the property (`Street`), type of alley access to the property (`Alley`), and type of utiles available (`Utilities`)

```
plotHist <- function(data_in, i) {
    data <- data.frame(x = data_in[[i]])
```

```
    p <- ggplot(data = data, aes(x = factor(x))) + stat_count() +
        xlab(colnames(data_in)[i]) + theme_light() + theme(axis.text.x = element_text(angle = 90,
        hjust = 1))
    return(p)
}

doPlots <- function(data_in, fun, ii, ncol = 3) {
    pp <- list()
    for (i in ii) {
        p <- fun(data_in = data_in, i = i)
        pp <- c(pp, list(p))
    }
    do.call("grid.arrange", c(pp, ncol = ncol))
}

plotDen <- function(data_in, i) {
    data <- data.frame(x = data_in[[i]], SalePrice = data_in$SalePrice)
    p <- ggplot(data = data) + geom_line(aes(x = x), stat = "density",
        size = 1, alpha = 1) + xlab(paste0((colnames(data_in)[i]),
        "\n", "Skewness: ", round(skewness(data_in[[i]], na.rm = TRUE),
            2))) + theme_light()
    return(p)
}

plotCorr <- function(data_in, i) {
    data <- data.frame(x = data_in[[i]], SalePrice = data_in$SalePrice)
    p <- ggplot(data, aes(x = x, y = SalePrice)) + geom_point(na.rm = TRUE) +
        geom_smooth(method = lm) + xlab(paste0(colnames(data_in)[i],
        "\n", "R-Squared: ", round(cor(data_in[[i]], data$SalePrice,
            use = "complete.obs"), 2))) + theme_light()
    return(suppressWarnings(p))
}
```

```
doPlots(housingDF, fun = plotHist, ii = c(3, 6, 7, 10), ncol = 2)
```

We also plot the distribution of houses against a number of features related to the physical geography of the property:

```
doPlots(housingDF, fun = plotHist, ii = c(8, 9, 11, 12), ncol = 2)
```

Figure 2 suggests that most of the houses are located in `Medium/Low Density` residential areas. We can also observe that most of the houses have paved road access, do not have alleys and have all public utilities(E,G,W,& S). From Figure **??**{fig:hist2}, we can notice that most of the properties are regular or slightly irregular in share, built on level surfaces with gentle slope.

```
housingDF %>% dplyr::select(LandSlope, Neighborhood) %>% arrange(Neighborhood) %>%
    group_by(Neighborhood, LandSlope) %>% summarize(Count = n()) %>%
    ggplot(aes(Neighborhood, Count)) + geom_bar(aes(fill = LandSlope),
    position = "dodge", stat = "identity") + theme(axis.text.x = element_text(angle = 90,
    hjust = 1))
```

From Figure **??**{fig:hist3}, we can see that houses with severe slope are located only in Clear Creek and Timberland while more than 10 neighborhoods have properties with moderate slope.

```
num_var <- names(housingDF)[which(sapply(housingDF, is.numeric))]
housing_numeric <- housingDF[num_var]
```

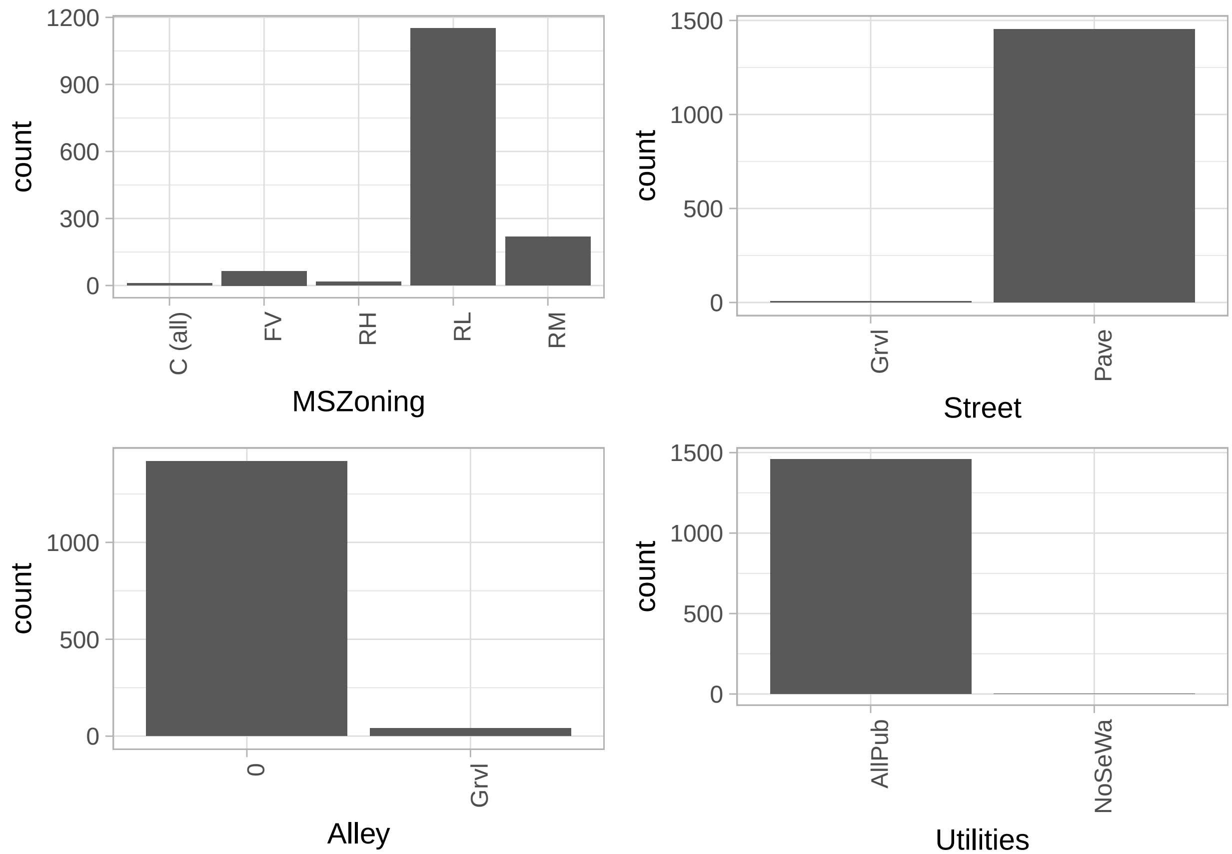


Figure 2: Locality, access, utility features distribution

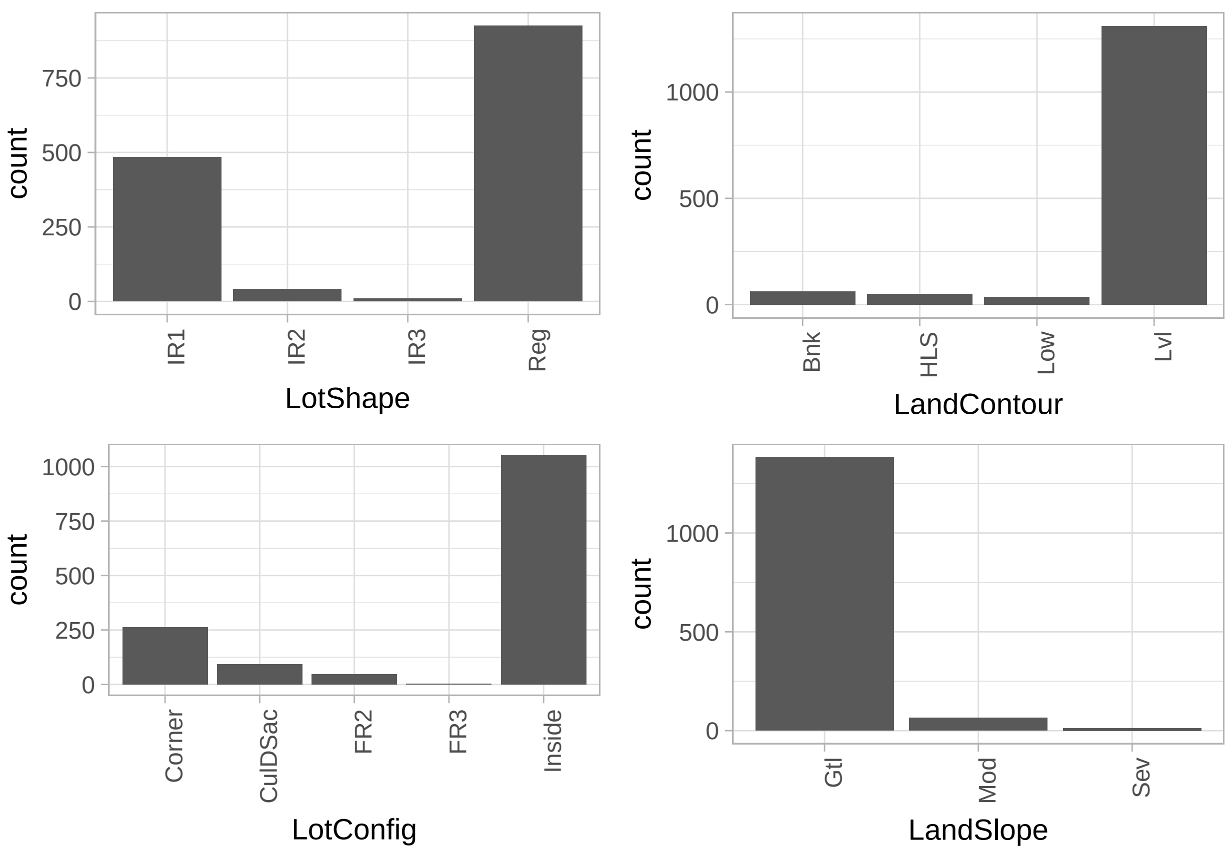


Figure 3: Lot/Land feature distribution

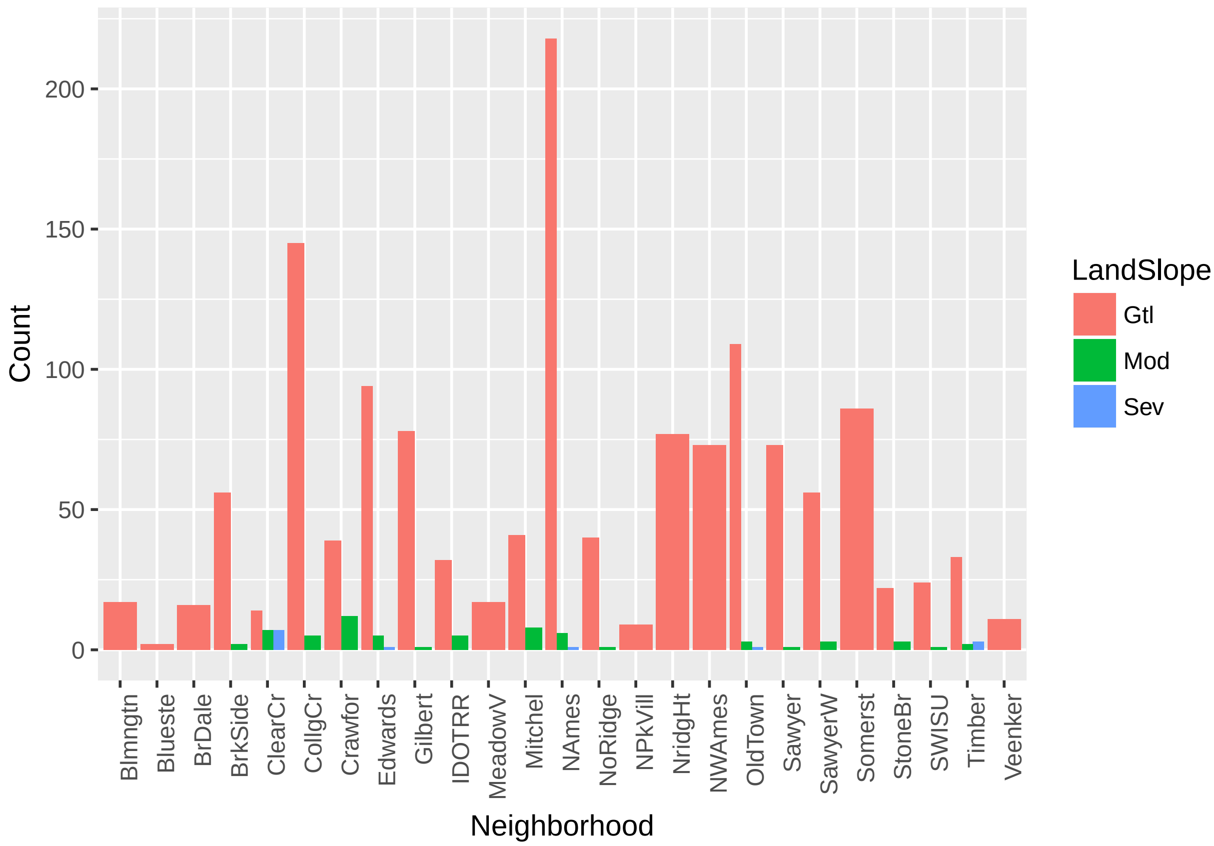


Figure 4: Neighborhood level slope distribution

```r
correlations <- cor(housing_numeric[, -1])
highcorr <- c(names(correlations[, "SalePrice"])[which(correlations[,
    "SalePrice"] > 0.5)], names(correlations[, "SalePrice"])[which(correlations[,
    "SalePrice"] < -0.2)])
data_corr <- housingDF[highcorr]
img <- doPlots(data_corr, fun = plotCorr, ii = 1:10)
```

```r
png(filename = "faithful.png")
plot(img)
```

Task 1. Building the Explanatory Model

Testing for Influential Points

Having dealt with the NAs in our dataset, we use the `model.matrix()` function from the `glmnet` package to convert each categorical variable into an appropriate set of binary indicators: for a categorical variable that takes k levels, `model.matrix()` produces k-1 binary indicators. We then reappend our response vector `SalePrice` to the resulting wide design matrix `designDF` to create `workingDF`, which includes both the converted predictors and response variables.

```r
designDF <- model.matrix(SalePrice ~ ., data = housingDF)[, -1]
designDF <- as.data.frame(designDF)
workingDF <- cbind(designDF, SalePrice = housingDF$SalePrice)
```

In looking for influential points, we leverage the `OLSRR` package to test observations for influence according to the DFFITS diagnostic. We do this by first fitting a saturated model on `workingDF` and then calling `ols_dffits_plot()` on it.

```r
ols_model <- lm(SalePrice ~ ., data = workingDF)
ols_dffits_plot(ols_model)
```

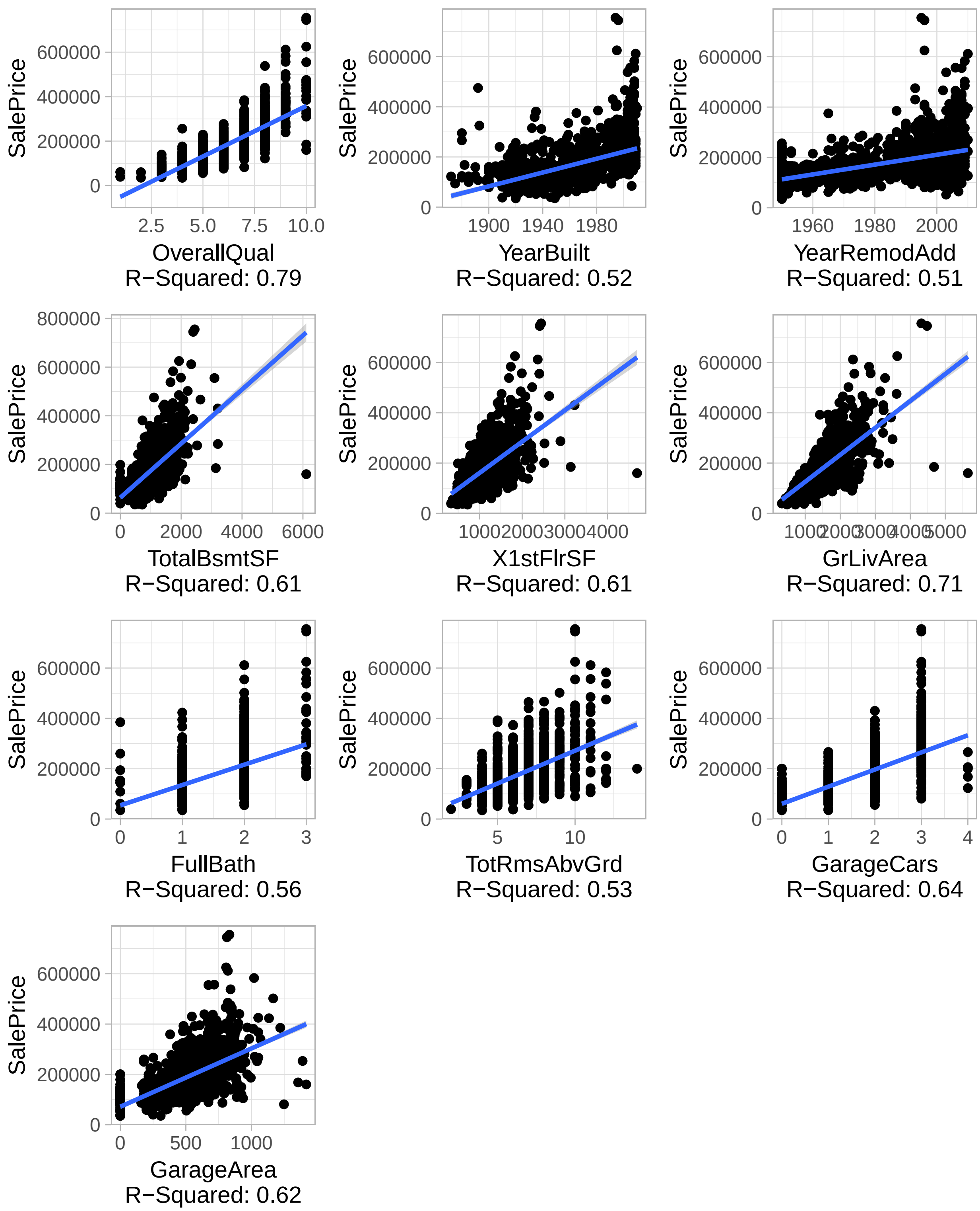
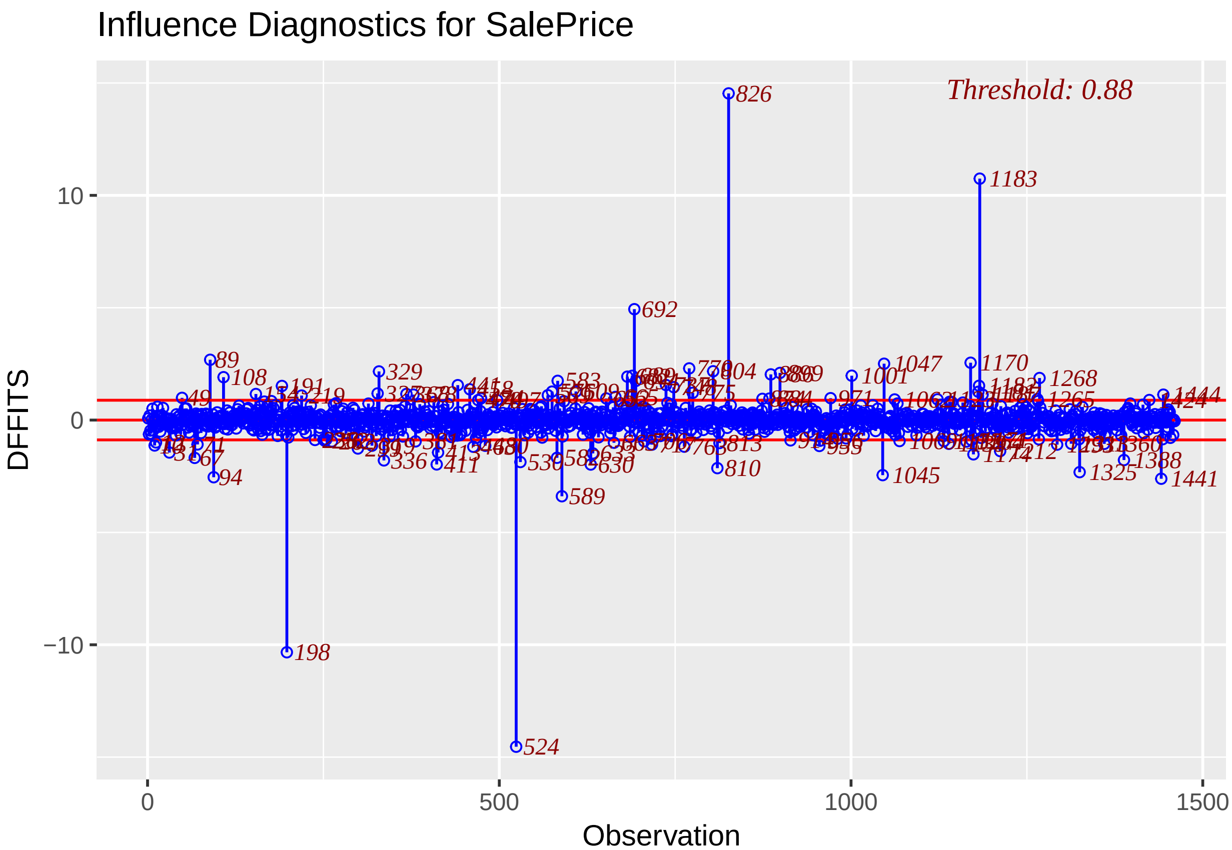


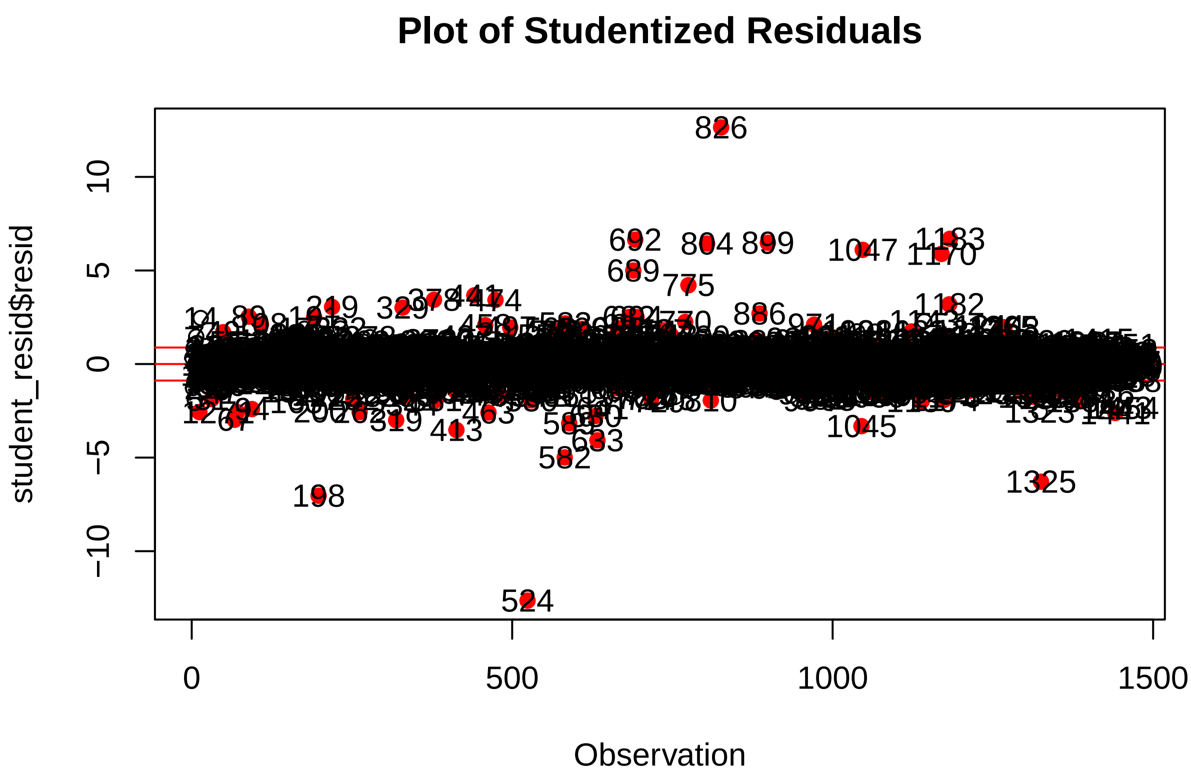
Figure 5: Scatter plot of variables showing high positive linear relationship with SalePrice




```
# identify threshold t for points of influence
n = nrow(workingDF)
p = ncol(workingDF - 1)  # remove response var
t = 2 * sqrt(p/n)
df <- dffits(ols_model)
influential_points <- which(abs(df) > t)
```

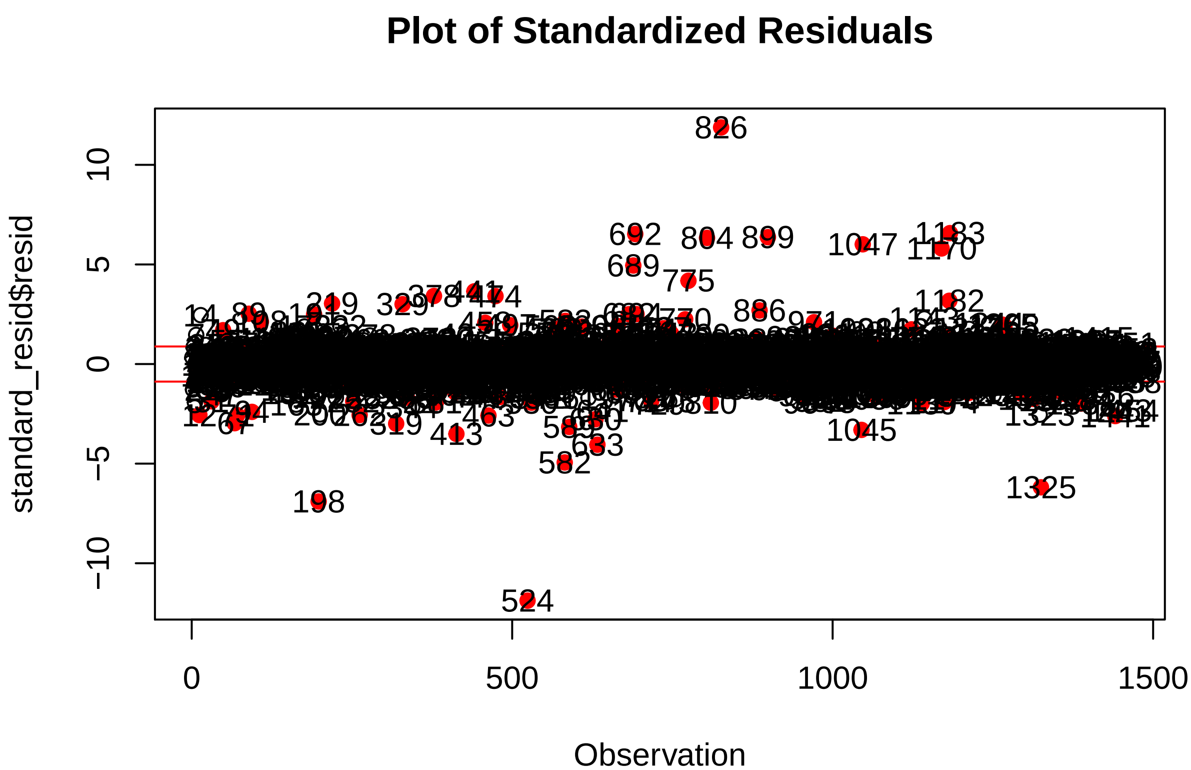
Note that according to the criterion of threshold t = 2*sqrt(n/p) = 0.88, the DFFITS plot shows a large number of influential observations. Below we plot the standardized and studentized residuals to check these observations for being outliers and/or points of leverage respectively.

```
# Creating from scratch because OLSRR ols_srsd_plot() function is
# not working.

# Create df of studentized residuals
student_resid <- as.data.frame(rstudent(ols_model))
student_resid <- setDT(student_resid, keep.rownames = TRUE)[]
colnames(student_resid) <- c("ix", "resid")

# Plot
plot(student_resid$ix, student_resid$resid, col = ifelse(workingDF$Id %in%
    influential_points, "red", "black"), pch = ifelse(workingDF$Id %in%
    influential_points, 19, 1), main = "Plot of Studentized Residuals",
    xlab = "Observation")
abline(h = t, col = "red")
abline(h = 0, col = "red")
abline(h = -t, col = "red")
text(student_resid$ix, student_resid$resid, labels = student_resid$ix)
```



Our plot of studentized resdiuals indicates that observations all 5 points are leverage points.

```r
# Create df of standardized residuals
standard_resid <- as.data.frame(rstandard(ols_model))
standard_resid <- setDT(standard_resid, keep.rownames = TRUE)[]
colnames(standard_resid) <- c("ix", "resid")

# Plot
plot(standard_resid$ix, standard_resid$resid, col = ifelse(workingDF$Id %in%
    influential_points, "red", "black"), pch = ifelse(workingDF$Id %in%
    influential_points, 19, 1), main = "Plot of Standardized Residuals",
    xlab = "Observation")
abline(h = t, col = "red")
abline(h = -t, col = "red")
text(standard_resid$ix, standard_resid$resid, labels = standard_resid$ix)
```



Our plots of studentized and standardized resdiuals indicate that all 5 observations are both points of leverage and outliers. We remove them from our dataset and recreate the saturated OLS model below:

```
# remove influential points
workingDF <- filter(workingDF, !Id %in% influential_points)
# nrow(workingDF) #1455

# recreate model
ols_model <- lm(SalePrice ~ ., data = workingDF)
```

For the purposes of variable selection, we refer to the saturated OLS model created above and perform stepwise model selection according to both AIC and BIC criterions.

```
# Code chunk not run; load saved model for expediency model_aic <-
# step(ols_model, direction = 'backward', trace = F)
model_bic <- step(ols_model, k = log(nrow(workingDF)), direction = "backward",
    trace = F)

## save the models save(model_aic, file =
## '/Users/santhoshhari/Documents/Coursework/LinearRegression/IowaHousing/AIC_model.rda')
save(model_bic, file = "/Users/santhoshhari/Documents/Coursework/LinearRegression/IowaHousing/BIC_model
```

Per the BIC criterion, the following are the predictor variables signficant at the alpha = 0.05 level.

```
# load('/Users/booranium/usf/601_regression/project/IowaHousing/BIC_model.rda')
# # model loaded as 'model_bic'
# load('/Users/santhoshhari/Documents/Coursework/LinearRegression/IowaHousing/BIC_model.rda')
# # model loaded as 'model_bic'
load("BIC_model.rda")
# find coefficients significant at the alpha = 0.01 level
bool_bic <- summary(model_bic)$coeff[-1, 4] < 0.01
sig_var_bic <- names(bool_bic)[bool_bic == TRUE]
```

We can now perform OLS regression with our subset of 80 significant variables. The model summary is as follows:

```
model_sig_formula <- as.formula(paste("SalePrice ~ ", paste(sig_var_bic,
    collapse = "+")))
model_sig <- lm(formula = model_sig_formula, data = workingDF)

# create a model with scaled Xs and Y for multicollinearity
# detection
model_sig_scaled <- lm(formula = model_sig_formula, data = as.data.frame(scale(workingDF)))
```

We check for multicollinearity in our model by checking for Variance Inflation Factors:

```
vif(model_sig_scaled)[(sqrt(vif(model_sig_scaled)) > 10) == TRUE]
```

```
## RoofStyleGable   RoofStyleHip   GarageQualPo   GarageCondPo
##       408.8299       386.1746       201.3496       183.9733
```

We see that there are two variables with VIF values > threshold = 10. This tells us there is multicollinearity present in the dataset. We verify this by checking the Singular Value Criteria for multicollinearity:

```
coll_out = colldiag(model_sig, scale = TRUE, center = FALSE, add.intercept = TRUE)
coll_out$condindx[(coll_out$condindx > 30) == TRUE]
```

```
##  [1]   35.12950   37.25354   41.00134   44.70393   47.03426   50.09702
##  [7]   86.71754  174.90633  323.29430  339.42217 1188.08143
```

We see that there are several entries > threshold = 30, and hence we conclude that multicollinearity exists. In order to identify which variables are multicollinear:

```
coll_out = colldiag(model_sig, scale = TRUE, center = FALSE, add.intercept = TRUE)
# unlist(coll_out[1], use.names = F) >30
# colnames(coll_out$pi)[unlist(coll_out[1], use.names = F) >30]
# dim(as.matrix(coll_out$pi[unlist(coll_out[1], use.names = F)
# >30, ]))
```

Based on results, we drop the Garage Condition variables since they are collinear with the Garage Quality variables.

```
# Drop GarageCondition variables since they are correlated with
# Garage Quality variables
drop = c("GarageCondEx", "GarageCondFa", "GarageCondGd", "GarageCondPo")
new_sig_var_bic = sig_var_bic[!sig_var_bic %in% drop]
```

We then rerun the model and recheck for multicollinearity using VIFS:

```
new_model_formula <- as.formula(paste("SalePrice ~ ", paste(new_sig_var_bic,
    collapse = "+")))
new_model <- lm(formula = new_model_formula, data = workingDF)

# create a model with scaled Xs and Y for multicollinearity
# detection
new_model_scaled <- lm(formula = new_model_formula, data = as.data.frame(scale(workingDF)))

# check for MC
vif(new_model_scaled)[(sqrt(vif(new_model_scaled)) > 10) == TRUE]
```

```
## RoofStyleGable   RoofStyleHip
```

```
##       407.4860       384.8541
```

Using our rule of thumb, we conclude that there is no more multicollinearity in our model since there are no VIF values > 10. We print the summary of our model below:

```r
summary(new_model)
```

```
##
## Call:
## lm(formula = new_model_formula, data = workingDF)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -50967  -9114      0   9374  78453
##
## Coefficients:
##                           Estimate    Std. Error t value
## (Intercept)          -769803.22374   69781.54526 -11.032
## MSSubClass90          -22689.26055    2758.21438  -8.226
## MSZoningFV             15381.27867    2658.36595   5.786
## MSZoningRL              4396.49518    1656.92896   2.653
## LotArea                    0.57966       0.06814   8.506
## StreetPave             32019.70536   11328.45116   2.826
## LotConfigCulDSac        5908.51136    1840.73593   3.210
## LandSlopeSev          -25580.10603    7657.75678  -3.340
## NeighborhoodCrawfor    21045.29180    2765.10971   7.611
## NeighborhoodEdwards   -10110.29347    1891.31812  -5.346
## NeighborhoodMitchel   -13544.46199    2506.21139  -5.404
## NeighborhoodNAmes      -8796.73927    1508.69058  -5.831
## NeighborhoodNoRidge    27759.64167    3033.04050   9.152
## NeighborhoodNridgHt    15615.47433    2547.23037   6.130
## NeighborhoodNWAmes     -8411.14189    2142.32902  -3.926
## NeighborhoodOldTown    -6974.48433    2214.06568  -3.150
## NeighborhoodStoneBr    29791.88536    3679.67175   8.096
## Condition1Norm          6663.77087    1349.20317   4.939
## Condition1RRAe        -16515.17983    4906.26379  -3.366
## Condition2PosA         47202.57547   16744.21592   2.819
## Condition2RRAe        -73213.65564   24829.41270  -2.949
## BldgType2fmCon        -12550.17442    3345.99326  -3.751
## BldgTypeTwnhs         -22181.44300    2819.31338  -7.868
## BldgTypeTwnhsE        -16634.03401    1978.01885  -8.409
## OverallQual             7032.60050     610.66124  11.516
## OverallCond             6162.75755     464.20112  13.276
## YearBuilt                378.63849      34.76266  10.892
## RoofStyleGable       -605093.86455   20604.27439 -29.367
## RoofStyleGambrel     -598056.03302   21290.78233 -28.090
## RoofStyleHip         -605102.33452   20678.97378 -29.262
## RoofStyleMansard     -595582.60167   21932.88817 -27.155
## RoofStyleShed        -560862.89835   26950.28127 -20.811
## RoofMatlCompShg       604404.77851   18829.27005  32.099
## RoofMatlMembran        53181.36870   17541.47082   3.032
## RoofMatlRoll          596448.58633   24424.41285  24.420
## RoofMatlWdShake       594623.91323   20585.68659  28.885
## RoofMatlWdShngl       677209.32964   20608.09059  32.861
## Exterior1stBrkFace     13471.08442    2710.33052   4.970
```

```
## Exterior1stHdBoard     -4479.94318    1413.55072  -3.169
## Exterior1stVinylSd    -14615.25487    4716.53131  -3.099
## `Exterior1stWd Sdng`   -6855.75853    2669.47924  -2.568
## Exterior2ndVinylSd     15538.49393    4702.31706   3.304
## `Exterior2ndWd Sdng`    7434.92237    2618.53639   2.839
## MasVnrTypeStone         7622.31009    1777.09860   4.289
## MasVnrArea                14.91012       3.10336   4.805
## ExterQualGd           -16592.28757    2870.20045  -5.781
## ExterQualTA           -17804.42088    3008.10305  -5.919
## BsmtQualEx            -10152.23753    3652.37134  -2.780
## BsmtQualFa            -19073.67397    1948.06502  -9.791
## BsmtQualGd            -15996.55862    2206.32545  -7.250
## BsmtExposureAv         15064.22546    1761.73428   8.551
## BsmtFinType1BLQ         4421.05427    1343.29747   3.291
## BsmtFinSF1                33.06943       2.51301  13.159
## BsmtFinSF2                26.23439       3.49838   7.499
## BsmtUnfSF                 20.90566       2.28915   9.133
## X1stFlrSF                 54.74564       2.67723  20.449
## X2ndFlrSF                 57.20752       1.80482  31.697
## BsmtFullBath            3903.84720    1187.14682   3.288
## BedroomAbvGr           -3663.13465     785.59144  -4.663
## KitchenQualFa         -25976.65017    3799.26907  -6.837
## KitchenQualGd         -26010.76619    2272.93843 -11.444
## KitchenQualTA         -26007.85333    2534.24297 -10.263
## FunctionalSev         -61018.92495   17443.07821  -3.498
## FunctionalTyp          11992.47408    1938.30569   6.187
## GarageTypeBasment       5878.26738    2055.69272   2.860
## GarageCars              4995.81503    1432.12340   3.488
## GarageArea                19.70027       4.77627   4.125
## GarageQualEx          -15057.49664    3368.52679  -4.470
## GarageQualFa           -7439.13822    5779.82461  -1.287
## GarageQualGd          -20855.42475    9393.23306  -2.220
## GarageQualPo          -11751.55107    2474.78897  -4.749
## WoodDeckSF                14.43592       3.79301   3.806
## ScreenPorch               31.07006       7.81124   3.978
## MoSold5                 3130.26063    1213.39222   2.580
## SaleConditionFamily    14177.10074    4114.95393   3.445
## SaleConditionNormal    11019.55137    1674.71876   6.580
## SaleConditionPartial   25470.25314    2445.28056  10.416
##                                   Pr(>|t|)
## (Intercept)          < 0.0000000000000002 ***
## MSSubClass90          0.000000000000000469 ***
## MSZoningFV            0.000000009042801631 ***
## MSZoningRL                        0.008067 **
## LotArea              < 0.0000000000000002 ***
## StreetPave                        0.004779 **
## LotConfigCulDSac                  0.001361 **
## LandSlopeSev                      0.000860 ***
## NeighborhoodCrawfor   0.000000000000052341 ***
## NeighborhoodEdwards   0.000000106492527113 ***
## NeighborhoodMitchel   0.000000077426485898 ***
## NeighborhoodNAmes     0.000000006970896714 ***
## NeighborhoodNoRidge  < 0.0000000000000002 ***
## NeighborhoodNridgHt   0.000000001163926728 ***
```

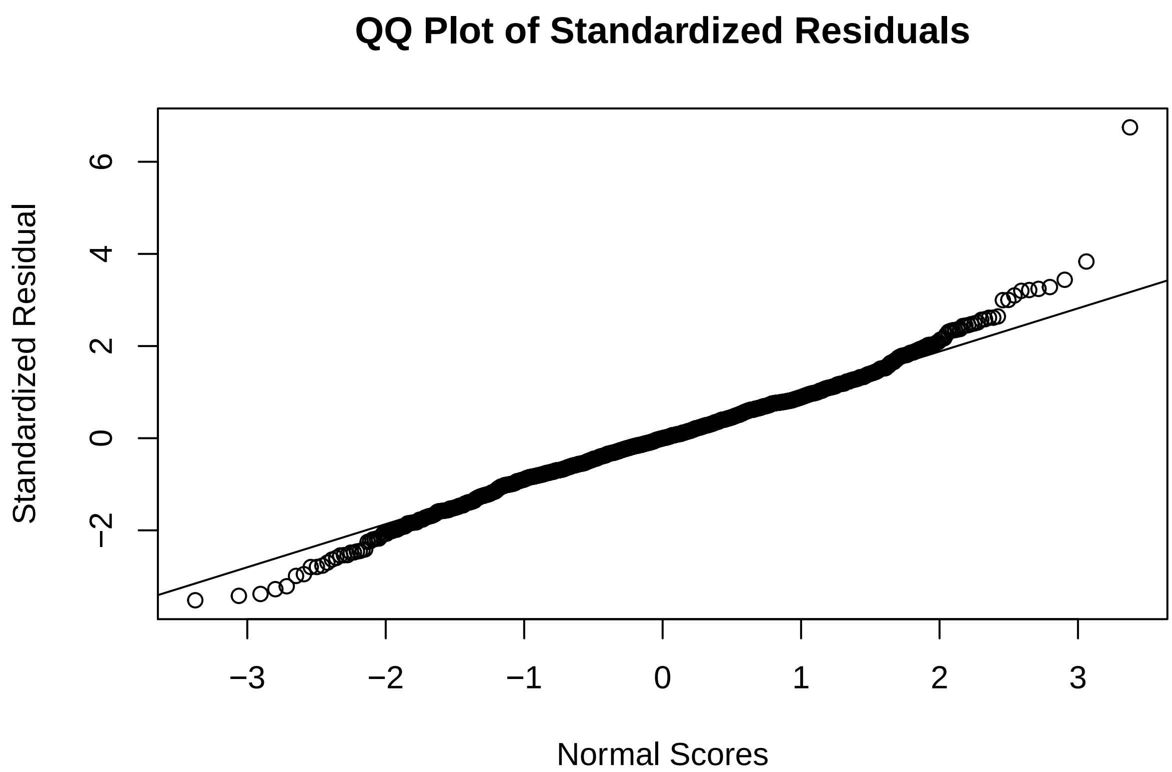
```
## NeighborhoodNWAmes   0.000090882357182539 ***
## NeighborhoodOldTown              0.001670 **
## NeighborhoodStoneBr  0.000000000000001301 ***
## Condition1Norm       0.000000888179203778 ***
## Condition1RRAe                   0.000785 ***
## Condition2PosA                   0.004891 **
## Condition2RRAe                   0.003249 **
## BldgType2fmCon                   0.000184 ***
## BldgTypeTwnhs        0.000000000000007607 ***
## BldgTypeTwnhsE       < 0.0000000000000002 ***
## OverallQual          < 0.0000000000000002 ***
## OverallCond          < 0.0000000000000002 ***
## YearBuilt            < 0.0000000000000002 ***
## RoofStyleGable       < 0.0000000000000002 ***
## RoofStyleGambrel     < 0.0000000000000002 ***
## RoofStyleHip         < 0.0000000000000002 ***
## RoofStyleMansard     < 0.0000000000000002 ***
## RoofStyleShed        < 0.0000000000000002 ***
## RoofMatlCompShg      < 0.0000000000000002 ***
## RoofMatlMembran                  0.002480 **
## RoofMatlRoll         < 0.0000000000000002 ***
## RoofMatlWdShake      < 0.0000000000000002 ***
## RoofMatlWdShngl      < 0.0000000000000002 ***
## Exterior1stBrkFace   0.000000758674350388 ***
## Exterior1stHdBoard               0.001564 **
## Exterior1stVinylSd               0.001986 **
## `Exterior1stWd Sdng`             0.010335 *
## Exterior2ndVinylSd               0.000978 ***
## `Exterior2ndWd Sdng`             0.004592 **
## MasVnrTypeStone      0.000019273208690787 ***
## MasVnrArea           0.000001733532723539 ***
## ExterQualGd          0.000000009314589434 ***
## ExterQualTA          0.000000004152878727 ***
## BsmtQualEx                       0.005521 **
## BsmtQualFa           < 0.0000000000000002 ***
## BsmtQualGd           0.000000000000715252 ***
## BsmtExposureAv       < 0.0000000000000002 ***
## BsmtFinType1BLQ                  0.001025 **
## BsmtFinSF1           < 0.0000000000000002 ***
## BsmtFinSF2           0.000000000000119315 ***
## BsmtUnfSF            < 0.0000000000000002 ***
## X1stFlrSF            < 0.0000000000000002 ***
## X2ndFlrSF            < 0.0000000000000002 ***
## BsmtFullBath                     0.001035 **
## BedroomAbvGr         0.000003442450258690 ***
## KitchenQualFa        0.000000000012434815 ***
## KitchenQualGd        < 0.0000000000000002 ***
## KitchenQualTA        < 0.0000000000000002 ***
## FunctionalSev                    0.000484 ***
## FunctionalTyp        0.000000000821993857 ***
## GarageTypeBasment                0.004311 **
## GarageCars                       0.000502 ***
## GarageArea           0.000039517637115641 ***
## GarageQualEx         0.000008508026657887 ***
```

```
## GarageQualFa                       0.198295
## GarageQualGd                       0.026575 *
## GarageQualPo         0.000002278770849592 ***
## WoodDeckSF                         0.000148 ***
## ScreenPorch          0.000073489409243423 ***
## MoSold5                            0.009997 **
## SaleConditionFamily                0.000589 ***
## SaleConditionNormal  0.000000000068308430 ***
## SaleConditionPartial < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15250 on 1287 degrees of freedom
## Multiple R-squared:  0.9551, Adjusted R-squared:  0.9524
## F-statistic: 360.1 on 76 and 1287 DF,  p-value: < 0.00000000000000022
```

At first glance, the multiple R-squared value of 0.9551 indicates that 95.51% of the variability in SalePrice around its mean is explained by the mode, i.e. by the predictor variables that have been included. This suggests a high-performing explanatory model.

Before welcoming this conclusion, we validate the linearity and normality assumptions of our model by checking our residuals as follows:

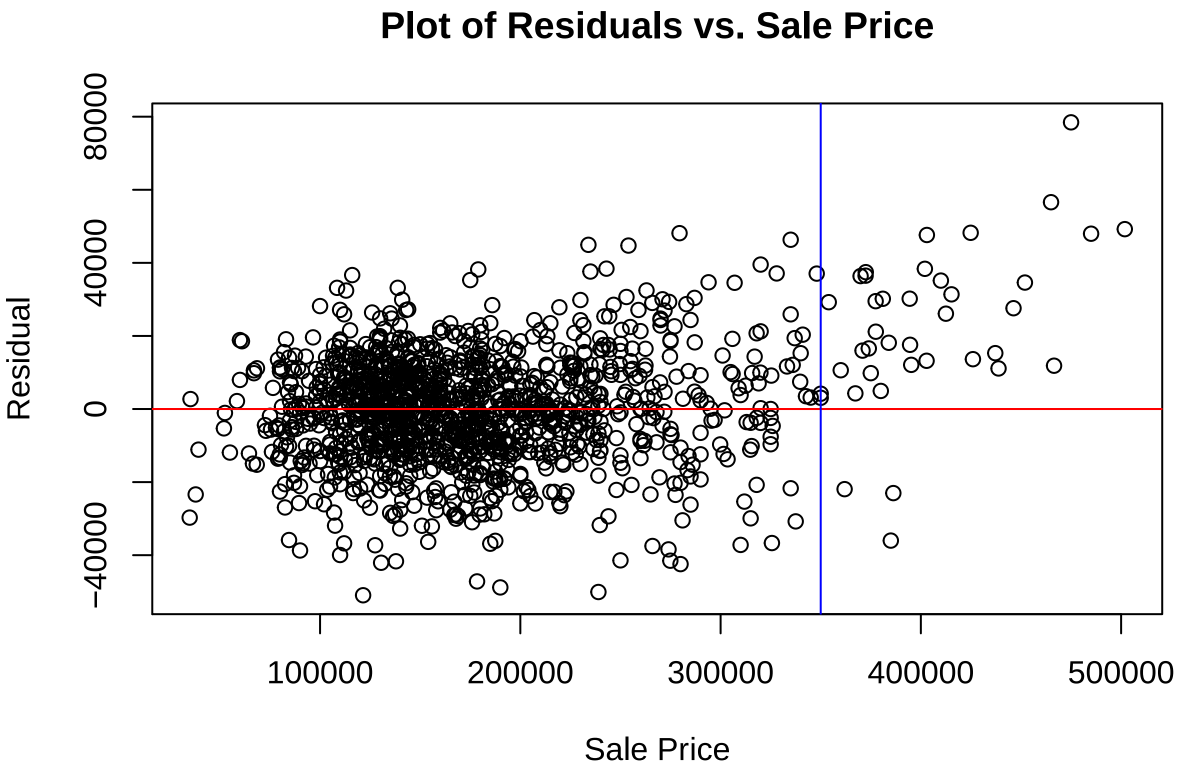
```r
# Residual plots

res = resid(new_model)  # residuals
stdres = rstandard(new_model)  # standardized residuals

# QQ Plot of Residuals - for normality
qqnorm(stdres, main = "QQ Plot of Standardized Residuals", xlab = "Normal Scores",
    ylab = "Standardized Residual")
qqline(stdres)
```




```
# Test for normality
ks.test(scale(res), rnorm(length(workingDF)))
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  scale(res) and rnorm(length(workingDF))
## D = 0.056359, p-value = 0.4442
## alternative hypothesis: two-sided
```

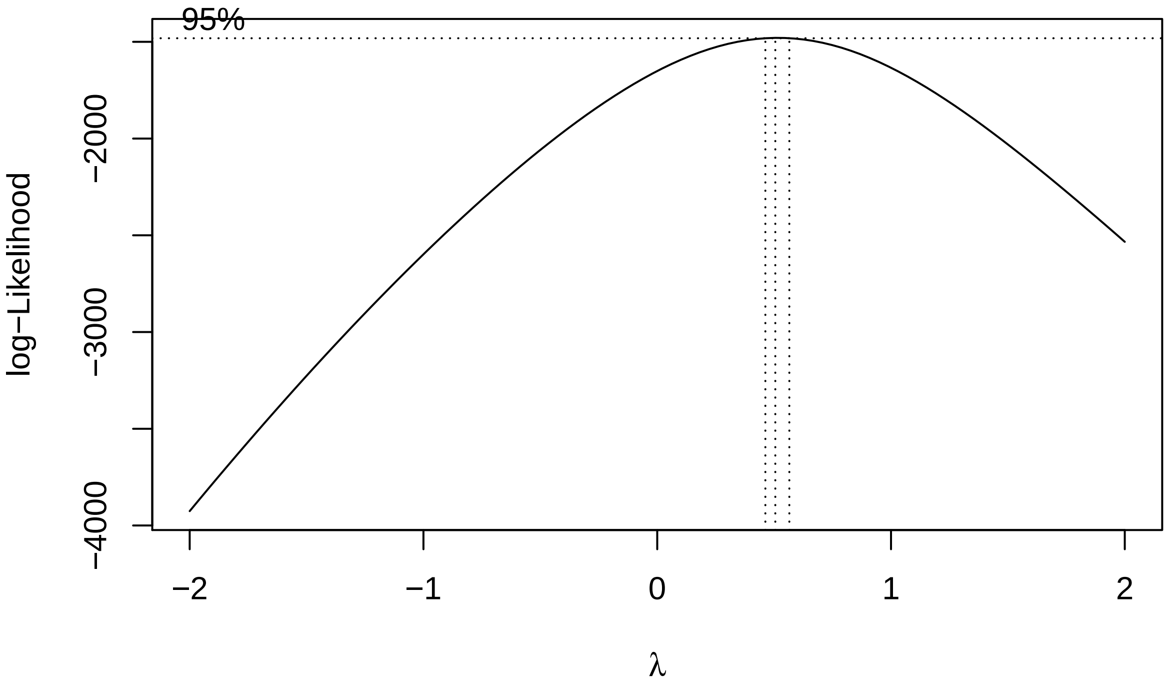
```
# Plot of residuals vs. fitted values - for linearity
plot(workingDF$SalePrice, res, main = "Plot of Residuals vs. Sale Price",
    xlab = "Sale Price", ylab = "Residual")
abline(h = c(0, 0), col = "red")
abline(v = 350000, col = "blue")
```



The plot of residuals against fitted values shows that for the most part, the residuals are evenly distributed across the y = 0 line. However, we see that as Sale Price increases, the residuals start to deviate homoscedasticity. More specifically, we see this deviation happen at approximately Sale Price = $350K, which our earlier summary showed to be between the variable's 3rd quartile and maximum. This suggets that for the last quartile of high-priced houses, the fitted regression model is not as adequate as it is for the reset of the population. The normal probability (QQ) plot corroborates this finding: it shows deviance from linearity at both tail ends of the residual rang, which suggests a heavy tailed distribution. This occurs at both ends of the distribution, i.e. both extremely low-priced houses and extremely high-priced houses are pulling the distribution away from normality.

As a remedial measure, we consider performing a transformation on Sale Price. We use the `boxcox()` function to determine the transformation under which the maximum likelihood is attained.

```
boxcox(new_model)
```

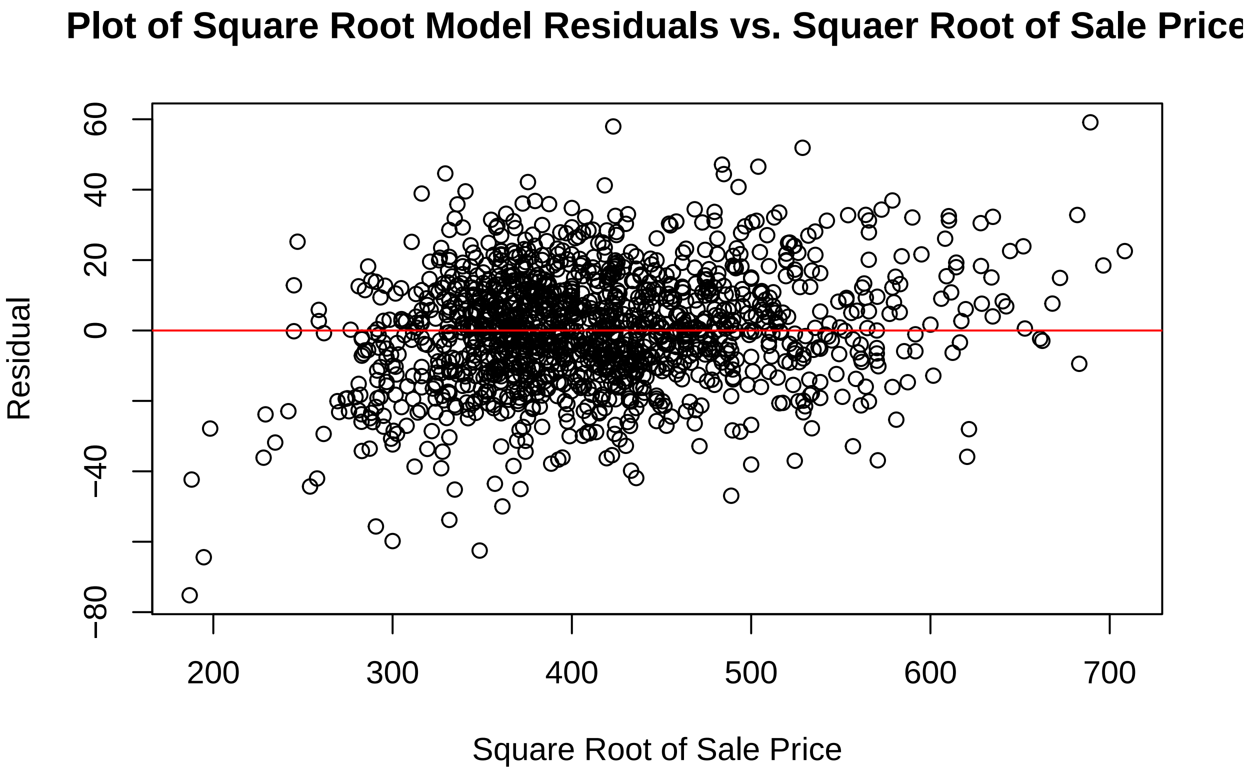


We see that lambda = 1 is not captured in the 95% CI of lambdas, indicated by the three vertical dashed lines. This means a transformation is necessary. We choose lambda = ~ 0.5 since this is an interpretable transformation value. Below, we apply a square root transformation to Sale Price, refit the model, and revalidate our model assumptions with residual plots as above.

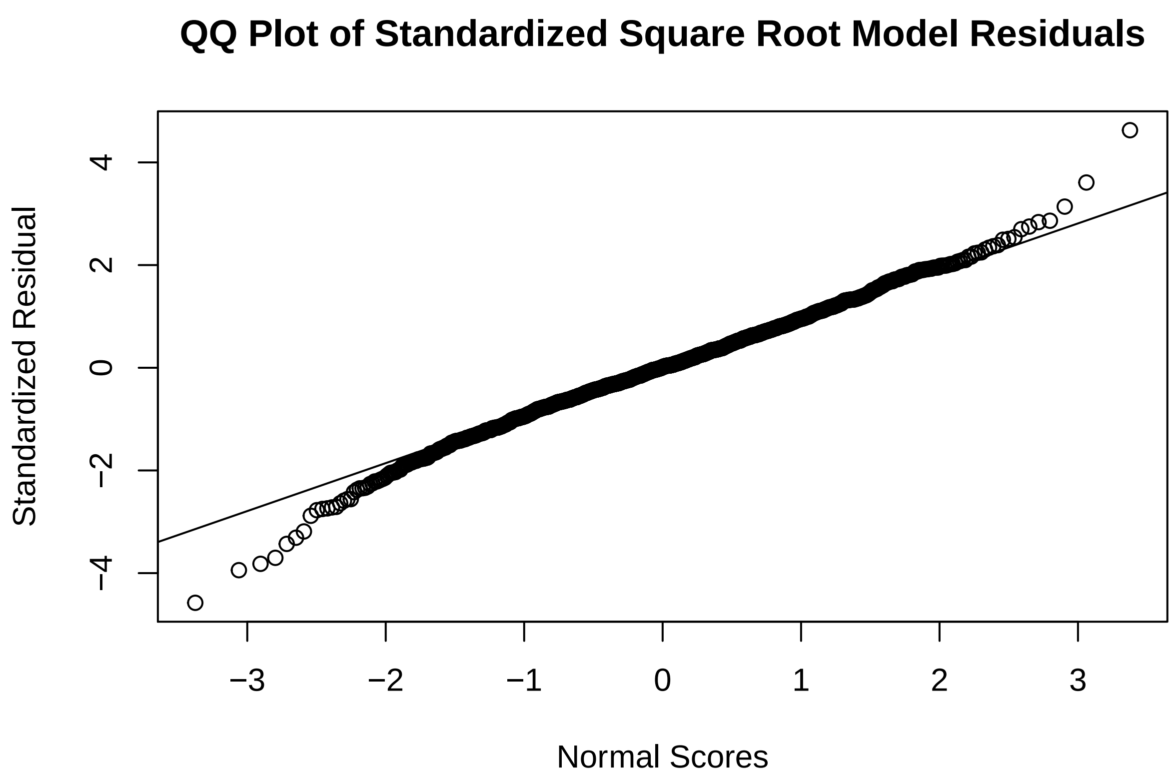
```r
# sqrt Transformation
sqrt_model_formula <- as.formula(paste("sqrt(SalePrice) ~ ", paste(new_sig_var_bic,
    collapse = "+")))
sqrt_model <- lm(formula = sqrt_model_formula, data = workingDF)

res_sqrt = resid(sqrt_model)  # residuals
stdres_sqrt = rstandard(sqrt_model)  # standardized residuals

ks.test(scale(res_sqrt), rnorm(length(workingDF)))
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  scale(res_sqrt) and rnorm(length(workingDF))
## D = 0.083216, p-value = 0.07712
## alternative hypothesis: two-sided
```

```r
# Plot of Residuals from Log Model vs. Fitted Values
plot(sqrt(workingDF$SalePrice), res_sqrt, main = "Plot of Square Root Model Residuals vs. Squaer Root of
    xlab = "Square Root of Sale Price", ylab = "Residual")
abline(h = c(0, 0), col = "red")
abline(v = 350000, col = "blue")
```




```
# Normal Probability Plot of Residuals from Log Model
qqnorm(stdres_sqrt, main = "QQ Plot of Standardized Square Root Model Residuals",
    xlab = "Normal Scores", ylab = "Standardized Residual")
qqline(stdres_sqrt)
```



Our new model produces an R-squared value of 0.9573, indicating that 95.73% of the variance in Sale Price is captured by the model. Our residuals vs. fitted values plot shows a better pattern of homoscedasticity, which suggests that our linear regression model adequately captures the trend in the log-transformed data. The normal probability plot of the standardized residuals also shows better adherence to a linear pattern, which suggests that our assumption of normality is better.

Accepting our new model, we provide its summary as follows:

```r
summary(sqrt_model)
```

```
##
## Call:
## lm(formula = sqrt_model_formula, data = workingDF)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -75.202  -9.998   0.111  10.508  59.122
##
## Coefficients:
##                          Estimate    Std. Error t value
## (Intercept)          -890.28254116   76.69053930 -11.609
## MSSubClass90          -23.15882369    3.03130215  -7.640
## MSZoningFV             22.15426203    2.92156784   7.583
## MSZoningRL              9.67178844    1.82097968   5.311
## LotArea                 0.00057479    0.00007489   7.675
## StreetPave             22.16246322   12.45006865   1.780
## LotConfigCulDSac        6.13108230    2.02298517   3.031
## LandSlopeSev          -23.65915508    8.41594286  -2.811
## NeighborhoodCrawfor    25.24236666    3.03888018   8.306
## NeighborhoodEdwards   -11.94390815    2.07857545  -5.746
## NeighborhoodMitchel   -12.49230836    2.75434862  -4.535
## NeighborhoodNAmes      -8.80702331    1.65806437  -5.312
## NeighborhoodNoRidge    18.51954122    3.33333850   5.556
## NeighborhoodNridgHt    13.88472594    2.79942885   4.960
## NeighborhoodNWAmes     -7.84634507    2.35443866  -3.333
## NeighborhoodOldTown    -6.37394080    2.43327789  -2.619
## NeighborhoodStoneBr    25.12627583    4.04399200   6.213
## Condition1Norm          7.49007874    1.48278630   5.051
## Condition1RRAe        -19.51489711    5.39202757  -3.619
## Condition2PosA         49.77247369   18.40204232   2.705
## Condition2RRAe        -79.99579747   27.28774554  -2.932
## BldgType2fmCon         -9.05502288    3.67727637  -2.462
## BldgTypeTwnhs         -26.12426275    3.09845049  -8.431
## BldgTypeTwnhsE        -14.76703735    2.17386032  -6.793
## OverallQual             9.55545336    0.67112214  14.238
## OverallCond             8.57981690    0.51016116  16.818
## YearBuilt               0.52341684    0.03820447  13.700
## RoofStyleGable       -649.30821616   22.64428093 -28.674
## RoofStyleGambrel     -641.37385650   23.39875927 -27.411
## RoofStyleHip         -649.56789142   22.72637623 -28.582
## RoofStyleMansard     -637.32586325   24.10443930 -26.440
## RoofStyleShed        -595.88088406   29.61859898 -20.118
## RoofMatlCompShg       646.46439673   20.69353537  31.240
## RoofMatlMembran        57.40737796   19.27823255   2.978
## RoofMatlRoll          640.03737672   26.84264710  23.844
```

```
## RoofMatlWdShake       631.04286310   22.62385278  27.893
## RoofMatlWdShngl       705.99736303   22.64847497  31.172
## Exterior1stBrkFace     16.04936281    2.97867736   5.388
## Exterior1stHdBoard     -4.88734062    1.55350482  -3.146
## Exterior1stVinylSd    -12.36421667    5.18350988  -2.385
## `Exterior1stWd Sdng`   -6.20610191    2.93378144  -2.115
## Exterior2ndVinylSd     13.56179268    5.16788830   2.624
## `Exterior2ndWd Sdng`    6.73927198    2.87779480   2.342
## MasVnrTypeStone         7.51361770    1.95304718   3.847
## MasVnrArea              0.01077825    0.00341062   3.160
## ExterQualGd            -7.07349777    3.15437584  -2.242
## ExterQualTA            -8.23659526    3.30593203  -2.491
## BsmtQualEx             -4.98117594    4.01398861  -1.241
## BsmtQualFa            -13.60195837    2.14094079  -6.353
## BsmtQualGd            -11.64935230    2.42477130  -4.804
## BsmtExposureAv         13.89051056    1.93616165   7.174
## BsmtFinType1BLQ         3.97099614    1.47629588   2.690
## BsmtFinSF1              0.03634633    0.00276182  13.160
## BsmtFinSF2              0.02820104    0.00384475   7.335
## BsmtUnfSF               0.02318960    0.00251579   9.218
## X1stFlrSF               0.06408485    0.00294230  21.781
## X2ndFlrSF               0.06487064    0.00198351  32.705
## BsmtFullBath            5.30056192    1.30468492   4.063
## BedroomAbvGr           -2.25272998    0.86337198  -2.609
## KitchenQualFa         -24.31436828    4.17543052  -5.823
## KitchenQualGd         -22.45943281    2.49797957  -8.991
## KitchenQualTA         -23.02728484    2.78515558  -8.268
## FunctionalSev         -72.48691931   19.17009819  -3.781
## FunctionalTyp          12.41703740    2.13021520   5.829
## GarageTypeBasment       3.09072499    2.25922460   1.368
## GarageCars              5.82143117    1.57391637   3.699
## GarageArea              0.01801140    0.00524916   3.431
## GarageQualEx           -8.76961540    3.70204093  -2.369
## GarageQualFa            2.70398412    6.35207868   0.426
## GarageQualGd          -21.05107960   10.32324673  -2.039
## GarageQualPo           -2.90558874    2.71981511  -1.068
## WoodDeckSF              0.01710165    0.00416856   4.103
## ScreenPorch             0.04309816    0.00858462   5.020
## MoSold5                 3.21682667    1.33352885   2.412
## SaleConditionFamily    16.82125582    4.52237099   3.720
## SaleConditionNormal    14.71028952    1.84053082   7.992
## SaleConditionPartial   27.35381620    2.68738510  10.179
##                                   Pr(>|t|)    
## (Intercept)          < 0.0000000000000002 ***
## MSSubClass90          0.000000000000042250 ***
## MSZoningFV            0.000000000000064384 ***
## MSZoningRL            0.000000128115158746 ***
## LotArea               0.000000000000032508 ***
## StreetPave                        0.075294 .  
## LotConfigCulDSac                  0.002488 ** 
## LandSlopeSev                      0.005010 ** 
## NeighborhoodCrawfor   0.000000000000000247 ***
## NeighborhoodEdwards   0.000000011381546539 ***
## NeighborhoodMitchel   0.000006282817422515 ***
```

```
## NeighborhoodNAmes    0.000000127895980016 ***
## NeighborhoodNoRidge  0.000000033536088995 ***
## NeighborhoodNridgHt  0.000000799757976834 ***
## NeighborhoodNWAmes               0.000885 ***
## NeighborhoodOldTown              0.008910 **
## NeighborhoodStoneBr  0.000000000699547941 ***
## Condition1Norm       0.000000501840987753 ***
## Condition1RRAe                   0.000307 ***
## Condition2PosA                   0.006926 **
## Condition2RRAe                   0.003432 **
## BldgType2fmCon                   0.013930 *
## BldgTypeTwnhs        < 0.0000000000000002 ***
## BldgTypeTwnhsE       0.000000000016738966 ***
## OverallQual          < 0.0000000000000002 ***
## OverallCond          < 0.0000000000000002 ***
## YearBuilt            < 0.0000000000000002 ***
## RoofStyleGable       < 0.0000000000000002 ***
## RoofStyleGambrel     < 0.0000000000000002 ***
## RoofStyleHip         < 0.0000000000000002 ***
## RoofStyleMansard     < 0.0000000000000002 ***
## RoofStyleShed        < 0.0000000000000002 ***
## RoofMatlCompShg      < 0.0000000000000002 ***
## RoofMatlMembran                  0.002957 **
## RoofMatlRoll         < 0.0000000000000002 ***
## RoofMatlWdShake      < 0.0000000000000002 ***
## RoofMatlWdShngl      < 0.0000000000000002 ***
## Exterior1stBrkFace   0.000000084603300879 ***
## Exterior1stHdBoard               0.001693 **
## Exterior1stVinylSd               0.017209 *
## `Exterior1stWd Sdng`             0.034588 *
## Exterior2ndVinylSd               0.008787 **
## `Exterior2ndWd Sdng`             0.019342 *
## MasVnrTypeStone                  0.000125 ***
## MasVnrArea                       0.001613 **
## ExterQualGd                      0.025103 *
## ExterQualTA                      0.012847 *
## BsmtQualEx                       0.214849
## BsmtQualFa           0.000000000291833967 ***
## BsmtQualGd           0.000001735178809924 ***
## BsmtExposureAv       0.000000000001223774 ***
## BsmtFinType1BLQ                  0.007241 **
## BsmtFinSF1           < 0.0000000000000002 ***
## BsmtFinSF2           0.000000000000391208 ***
## BsmtUnfSF            < 0.0000000000000002 ***
## X1stFlrSF            < 0.0000000000000002 ***
## X2ndFlrSF            < 0.0000000000000002 ***
## BsmtFullBath         0.000051438664627682 ***
## BedroomAbvGr                     0.009180 **
## KitchenQualFa        0.000000007283232196 ***
## KitchenQualGd        < 0.0000000000000002 ***
## KitchenQualTA        0.000000000000000336 ***
## FunctionalSev                    0.000163 ***
## FunctionalTyp        0.000000007040642111 ***
## GarageTypeBasment                0.171536
```

```
## GarageCars                          0.000226 ***
## GarageArea                          0.000620 ***
## GarageQualEx                        0.017990 *
## GarageQualFa                        0.670409
## GarageQualGd                        0.041635 *
## GarageQualPo                        0.285584
## WoodDeckSF             0.000043431150624157 ***
## ScreenPorch            0.000000588049883416 ***
## MoSold5                             0.015993 *
## SaleConditionFamily                 0.000208 ***
## SaleConditionNormal    0.000000000000002920 ***
## SaleConditionPartial < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.76 on 1287 degrees of freedom
## Multiple R-squared:  0.9573, Adjusted R-squared:  0.9548
## F-statistic:   380 on 76 and 1287 DF,  p-value: < 0.00000000000000022
```

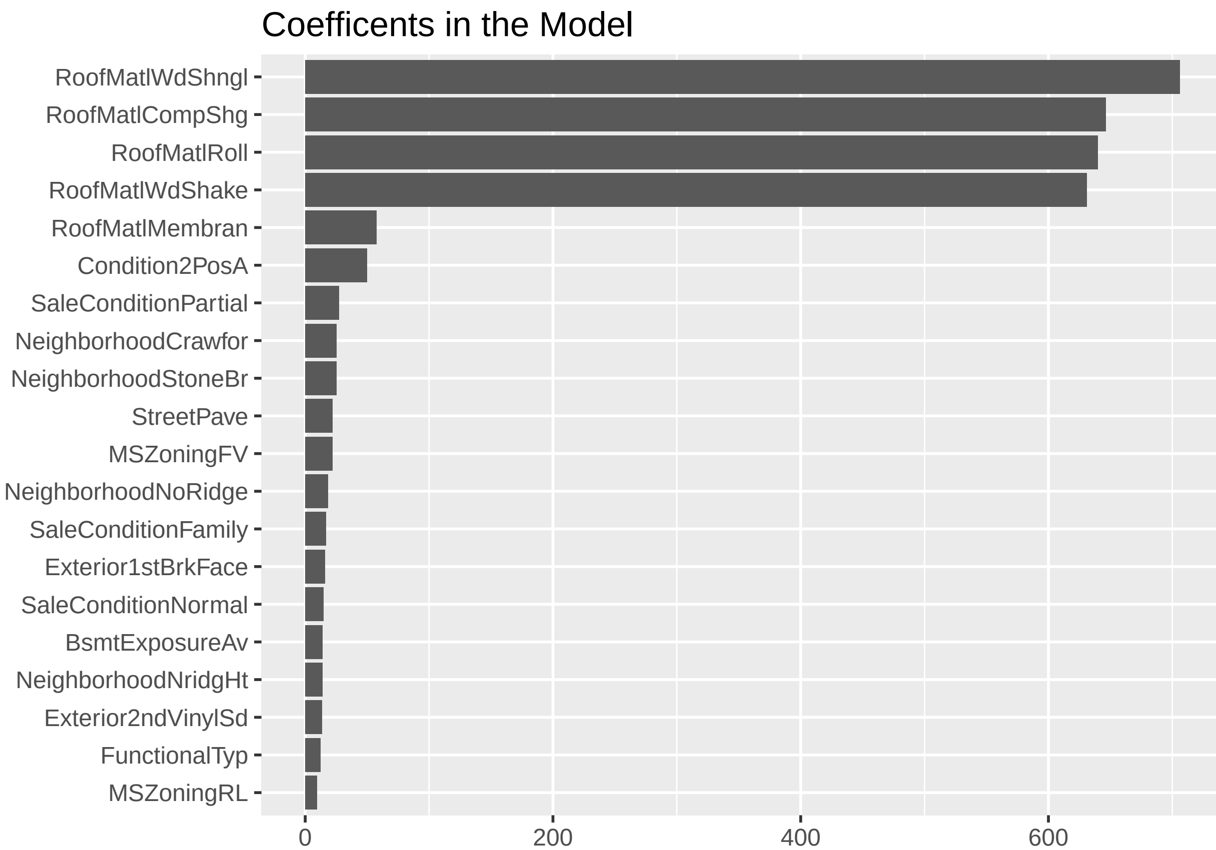
We conclude that the variables included above are most relevant in determining a house's sale price. In particular, those variables that are significant at the 0.01 level, i.e., are most significant.

Task 2: Making recommendations

```r
# morty <-
# read.csv('/Users/booranium/usf/601_regression/project/Morty.txt',
# stringsAsFactors = T)
morty <- read.csv("/Users/santhoshhari/Documents/Coursework/LinearRegression/IowaHousing/Data/Morty.txt
    stringsAsFactors = T)
morty <- morty[, -1]
new_data <- rbind(housingDF, morty)
new_data <- model.matrix(SalePrice ~ ., data = new_data)[, -1]
x_morty <- as.data.frame(tail(new_data, 1))
new_sig_var_bic[which(new_sig_var_bic == "`RoofMatlTar&Grv`")] <- "RoofMatlTar&Grv"
new_sig_var_bic[which(new_sig_var_bic == "`Exterior1stWd Sdng`")] <- "Exterior1stWd Sdng"
new_sig_var_bic[which(new_sig_var_bic == "`Exterior2ndWd Sdng`")] <- "Exterior2ndWd Sdng"
sig_var_morty <- x_morty[, new_sig_var_bic]
```

To make a recommendation to Morty regarding the selling price of his house, we leverage the model built above. We see that the 95% CI for the predicted Sale Price of Morty's house, based on selected attributes, is \$151502, As such, we recommend that Morty sell his house at a maximum of . . . , which is more/less than the other firm's recommendation of \$143K.

```r
coef <- data.frame(coef.name = names(coef(sqrt_model)), coef.value = matrix(coef(sqrt_model)))

# exclude the (Intercept) term
coef <- coef[-1, ]
coef <- arrange(coef, -coef.value)
imp_coef <- head(coef, 20)
ggplot(imp_coef) + geom_bar(aes(x = reorder(coef.name, coef.value),
    y = coef.value), stat = "identity") + coord_flip() + ggtitle("Coefficents in the Model") +
    theme(axis.title = element_blank())
```



Part II: Predictive Modelling

For comparing the prediction accuracy, we use 4 models - 1. Ridge Regression 2. Lasso Regression 3. Elastic Net

Transformations done on the data before building models - 1. Imputing NAs 2. Creating dummy variables 3. Removing influential points

Train and test sets: 75% of the data forms the train set 25% of the data forms the test set

```
x <- workingDF[, -1]
x$SalePrice <- log(workingDF[, ncol(workingDF)])
# y_log = log(y) train/test
set.seed(121)
train <- sample(1:nrow(x), 3 * nrow(x)/4)
test <- (-train)
```

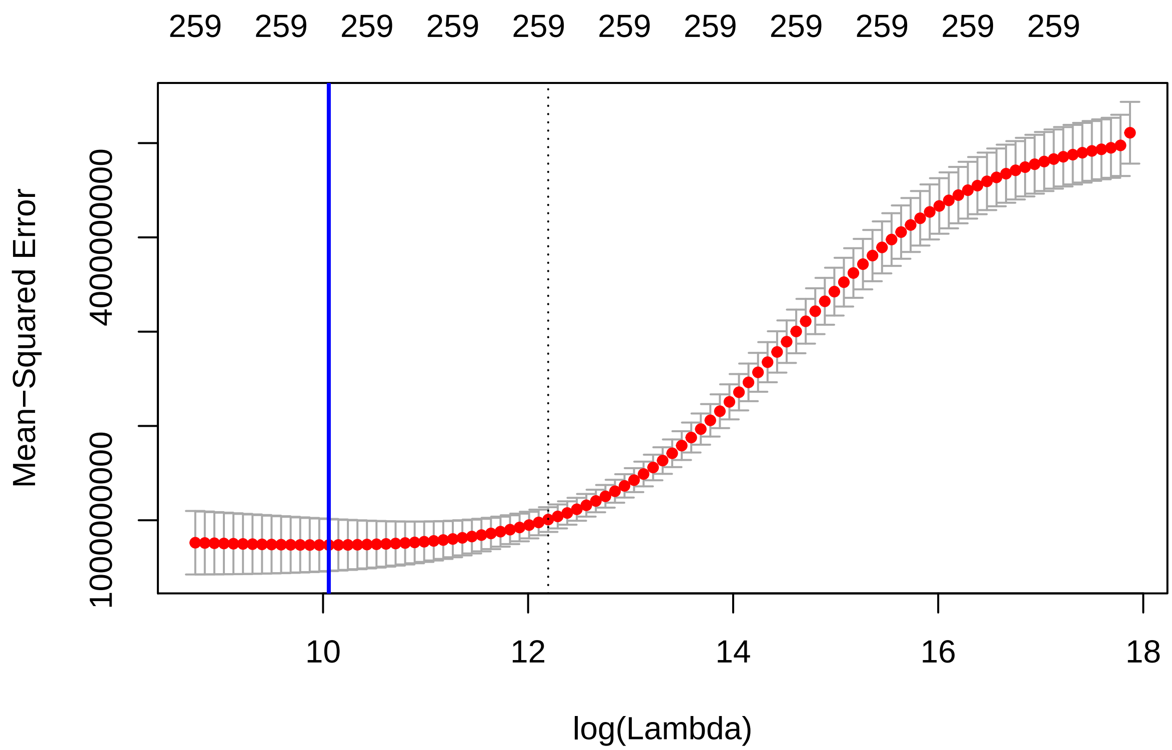
Ridge:

```
set.seed(121)

x <- workingDF[, 2:ncol(workingDF) - 1]
y <- workingDF$SalePrice

# train/test
y.train <- y[train]
y.test <- y[test]
```

```r
ridge <- cv.glmnet(as.matrix(x[train, ]), y.train, alpha = 0)
plot(ridge)
best.lambda <- ridge$lambda.min
best.lambda
```

```
## [1] 23298.94
```

```r
abline(v = log(best.lambda), col = "blue", lwd = 2)
```




```r
ridge.model.train <- glmnet(as.matrix(x[train, ]), y.train, alpha = 0,
    lambda = best.lambda)


ridge.pred <- predict(ridge.model.train, s = best.lambda, newx = as.matrix(x[test,
    ]))
mspe.ridge <- mean((ridge.pred - y.test)^2)

coef_ridge <- coef(ridge.model.train)
length(coef_ridge[coef_ridge != 0])
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
## [1] 260
```

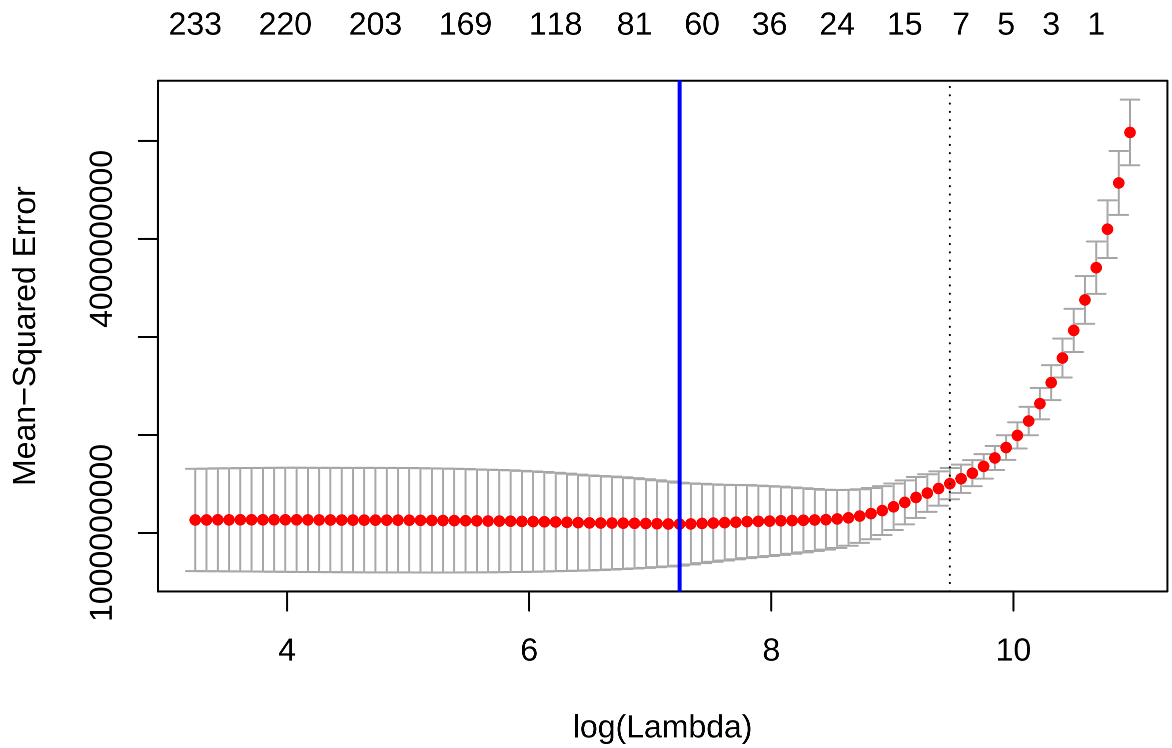
Lasso:

```r
set.seed(121)

x <- workingDF[, 2:ncol(workingDF) - 1]
y <- workingDF$SalePrice

lasso <- cv.glmnet(as.matrix(x[train, ]), y.train, alpha = 1)
```

```r
plot(lasso)
best.lambda <- lasso$lambda.min
best.lambda
```

```
## [1] 1396.734
```

```r
abline(v = log(best.lambda), col = "blue", lwd = 2)
```




```r
lasso.model.train <- glmnet(as.matrix(x[train, ]), y.train, alpha = 1,
    lambda = best.lambda)


lasso.pred <- predict(lasso.model.train, s = best.lambda, newx = as.matrix(x[test,
    ]))
mspe.lasso <- mean((lasso.pred - y.test)^2)
mspe.lasso
```

```
## [1] 317582059
```

```r
# coef(lasso)
coef_lasso <- coef(lasso.model.train)
length(coef_lasso[coef_lasso != 0])
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
## [1] 66
```

Elastic Net:

```r
set.seed(121)
x <- workingDF[, 2:ncol(workingDF) - 1]
y <- workingDF$SalePrice
# find best alpha
```

```
alphalist <- seq(0, 1, by = 0.1)
elasticnet <- lapply(alphalist, function(a) {
    cv.glmnet(as.matrix(x[train, ]), y.train, alpha = a)
})
mse <- list()
for (i in 1:11) {
    mse <- c(mse, min(elasticnet[[i]]$cvm))
    print(min(elasticnet[[i]]$cvm))
    print(i)
}
```

```
## [1] 737133836
## [1] 1
## [1] 764698552
## [1] 2
## [1] 762534661
## [1] 3
## [1] 763107197
## [1] 4
## [1] 786831497
## [1] 5
## [1] 792881423
## [1] 6
## [1] 803704732
## [1] 7
## [1] 808429440
## [1] 8
## [1] 801245048
## [1] 9
## [1] 842510672
## [1] 10
## [1] 1073135639
## [1] 11
```

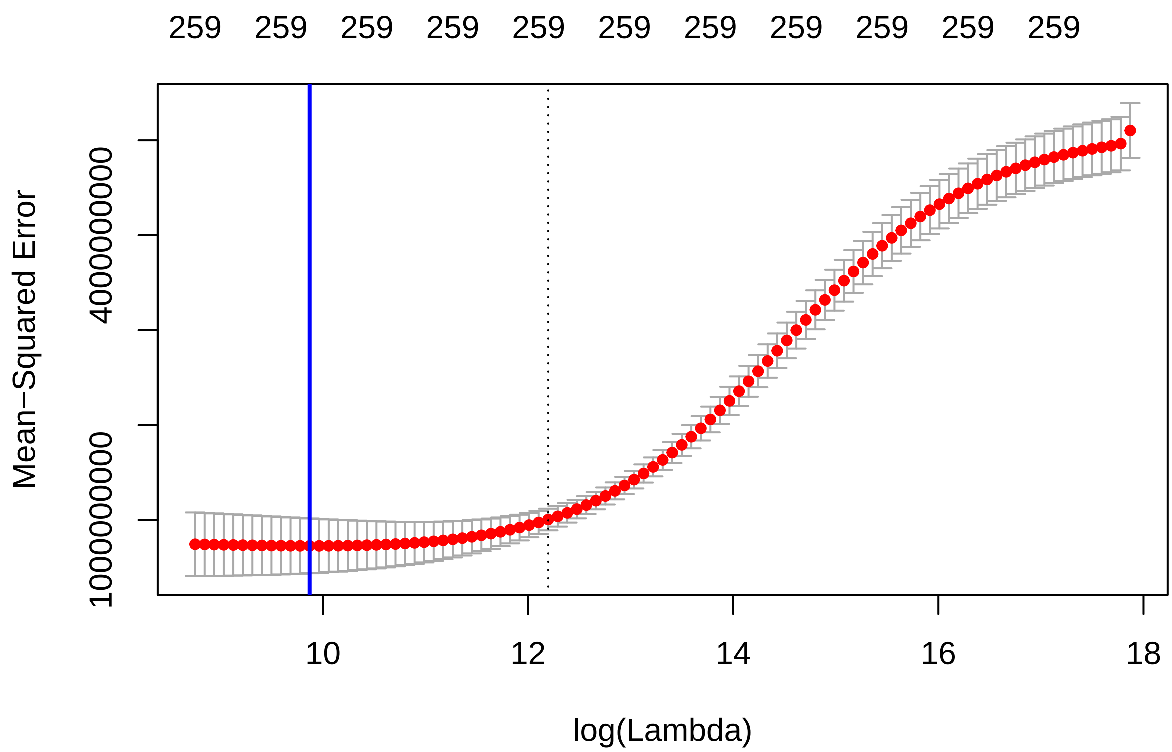
```
which.min(unlist(mse))
```

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

```
alpha_min = alphalist[which.min(unlist(mse))]

# find best lambda for best alpha
cv.out <- cv.glmnet(as.matrix(x[train, ]), y.train, alpha = alpha_min)

plot(cv.out)
best.lambda <- cv.out$lambda.min
best.lambda
```

```
## [1] 19343.19
```

```
abline(v = log(best.lambda), col = "blue", lwd = 2)
```




```r
# train EN
EN.model.train <- glmnet(as.matrix(x[train, ]), y.train, alpha = alpha_min,
    lambda = best.lambda)

EN.pred <- predict(EN.model.train, newx = as.matrix(x[test, ]))
mspe.EN <- mean((EN.pred - y.test)^2)
coef_elastic <- coef(EN.model.train)
length(coef_elastic[coef_elastic != 0])
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
## [1] 260
```

```r
MSPE <- data.frame(Ridge = mspe.ridge, Lasso = mspe.lasso, EN = mspe.EN)
```