Homework 5

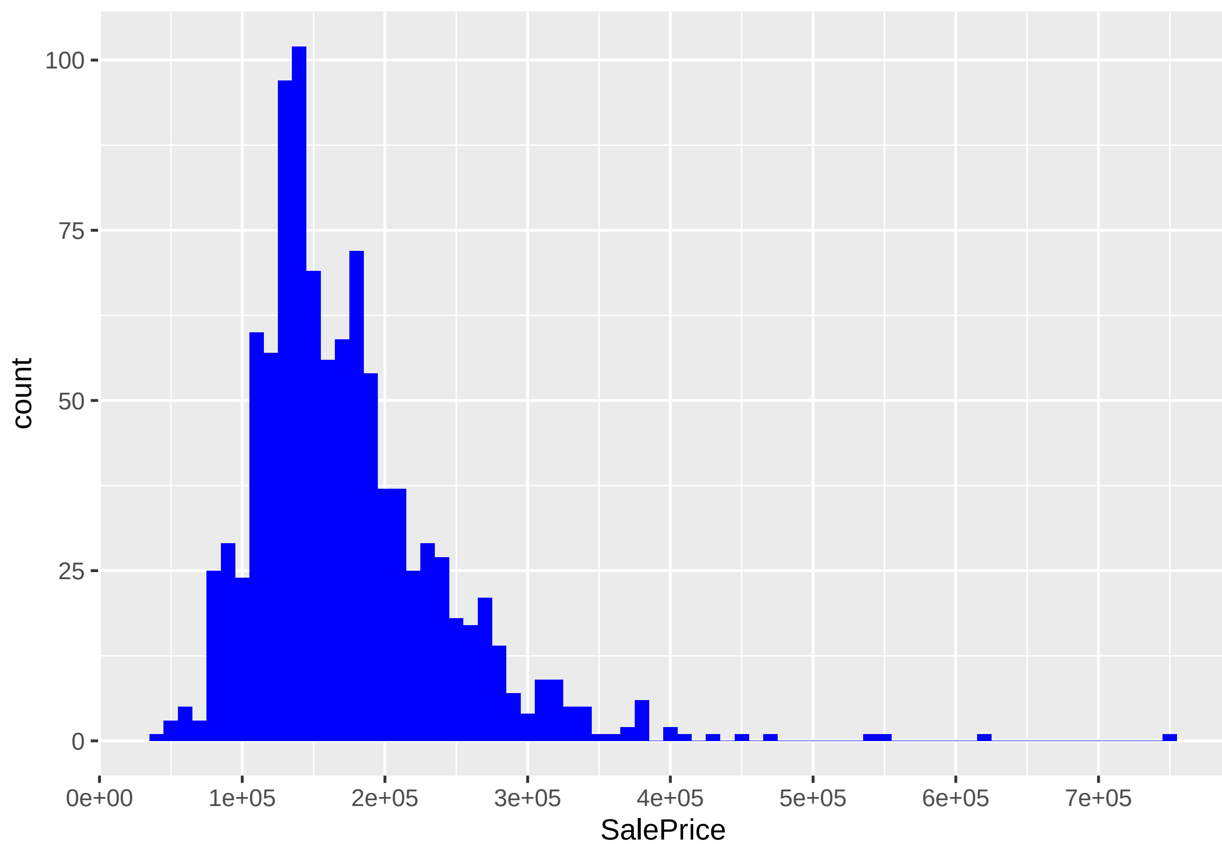
Students Names Redacted

2023-10-09

For the first step, it is better to evaluate the d ataset. by Using summary function, we obtain statistical summary of numeric variables and get an overview of the structure of the dataset.

As you can see, the sale prices are right skewed. which means a few people can afford v ery e xpensive houses

```
ggplot(data = housingData,aes(x=SalePrice)) +
        geom_histogram(fill="blue", binwidth = 10000) +
        scale_x_continuous(breaks= seq(0, 800000, by=100000))
```



```r
summary(housingData$SalePrice)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   39300  130000  160000  174561  205000  755000
```

processioning part: first we implement factor level and we consider 5 level and add new leveled variable to our dataset. Transforming all text data into numeric data

```r
# Calculate number of unique levels for each factor column
housingData_pre <- housingData
factor_levels <- sapply(housingData_pre, function(col) if(is.factor(col) |
                              is.character(col)) length(unique(col)) else NA)


# Filter out non-factor columns
factor_levels <- factor_levels[!is.na(factor_levels)]

# Sort the factors by number of levels
factor_levels_sorted <- sort(factor_levels, decreasing = TRUE)

# Display factors with their levels
#factor_levels_sorted

# Identify factor columns with 10 or more levels
factors_to_display <- names(factor_levels_sorted)[factor_levels_sorted >= 5]

# Loop through the factor columns with 10 or more levels
for(factor_name in factors_to_display) {

  # Remove the column if there are any NA values
if(anyNA(housingData[[factor_name]])) {
  housingData[[factor_name]] <- NULL
  next
}

  # Display the most frequent levels of the factor column
  factor_freq <- housingData_pre %>%
    filter(!is.na(.data[[factor_name]])) %>%
    count(.data[[factor_name]], sort = TRUE)

  # Extract the levels of the factor column and store them
  most_frequent_levels <- factor_freq[[factor_name]]

  # Using the first four most frequent levels from the list
  top_5_levels <- most_frequent_levels[1:5]

  # Convert top_5_levels to character vector
  top_5_levels_char <- as.character(top_5_levels)
```

```r
  # Apply the fct_other function
  collapsed_column_name <- paste(factor_name, "collapsed", sep = "_")
  str(collapsed_column_name)

  housingData_pre[[collapsed_column_name]] <-
            fct_other(housingData_pre[[factor_name]], keep = top_5_levels_char)

  # Convert the factor levels to integers
  housingData_pre[[collapsed_column_name]] <-
                as.integer(as.factor(housingData_pre[[collapsed_column_name]]))
  summary(housingData_pre)
}
```

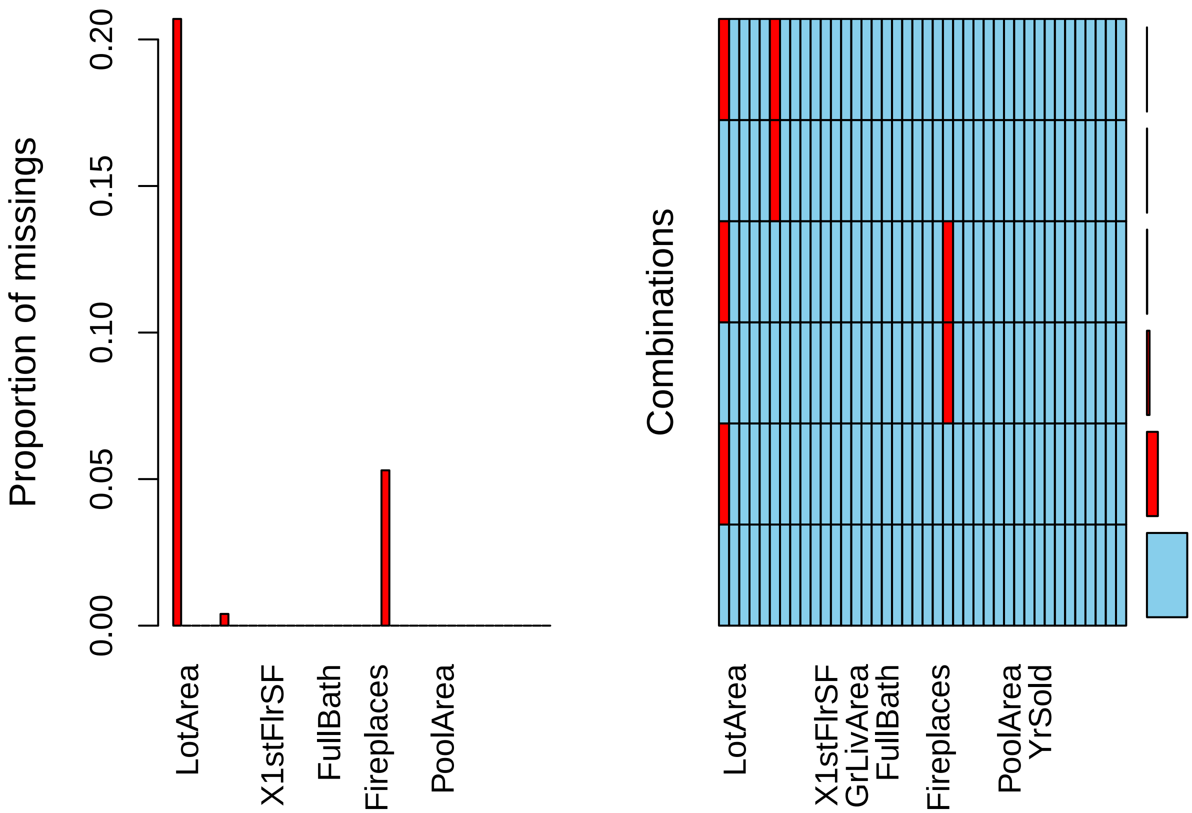
```
##  chr "Neighborhood_collapsed"
##  chr "Exterior2nd_collapsed"
##  chr "HouseStyle_collapsed"
##  chr "Exterior1st_collapsed"
##  chr "Condition1_collapsed"
##  chr "Functional_collapsed"
##  chr "BldgType_collapsed"
```

/ Then we handle missing values. As we can see, lotfrontage, MasVnArea, and GarageYrBlt has missing values. We check in the heatmap to find out which variable is more compatible with each of them then use Imputation regression with error. And finally mutate salePrice with Log_SalePrice /

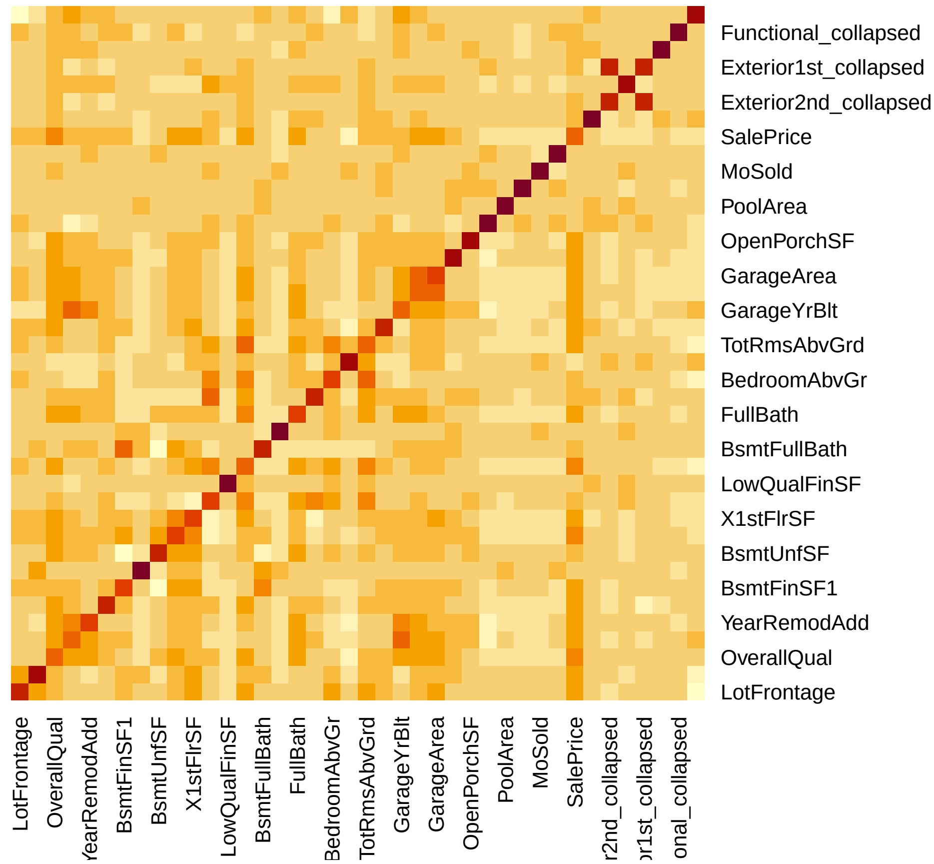
```r
#housingNumeric <- housingData_pre %>%
#select(where(is.numeric))
housingNumeric <- housingData_pre[sapply(housingData_pre, is.numeric)]
#remove cardinal data
housingNumeric <- subset(housingNumeric, select =
                                                -c(Id, MSSubClass, OverallCond))
#keep the list of missing values
missing_lot <- is.na(housingNumeric$LotFrontage)
missing_Mas <- is.na(housingNumeric$MasVnrArea)
missing_Gar <- is.na(housingNumeric$GarageYrBlt)
#remove NA values
housingNumeric_NNA <- na.omit(housingNumeric)
#keep columns name with missing values
name = colnames(housingNumeric)
columns_with_na <- colnames(housingNumeric)[colSums(is.na(housingNumeric)) > 0]
print(columns_with_na)
```

```
## [1] "LotFrontage" "MasVnrArea"  "GarageYrBlt"
```

```r
a <- aggr(housingNumeric)
```




```r
#summary(a)
housing_numeric_NNA <- na.omit(housingNumeric)
cor_matrix <- cor(housing_numeric_NNA)
par(mar=c(0.1, 0.1, 0.5, 0.1))
heatmap(cor_matrix, Colv = NA, Rowv = NA)
```



as we can see, lotFrontage has highest correlation with lotArea, MasVnrArea with BsmtFinSF1 and GarageYrBlt with YearBuilt fit a linear model to the data

```
fit_lot <- lm(housingNumeric$LotFrontage~housingNumeric$LotArea)
fit_Mas <- lm(housingNumeric$MasVnrArea~housingNumeric$BsmtFinSF1)
fit_Gar <- lm(housingNumeric$GarageYrBlt~housingNumeric$BsmtFinSF1)
f_lot<-summary(fit_lot)
f_Mas<-summary(fit_Mas)
f_Gar<-summary(fit_Gar)
#print (f)

#str(f_lot)

# extract the coefficients
c_lot<-f_lot[[4]]
# extract the model standard error
se_lot<-f_lot[[6]]
# extract the coefficients
c_Mas<-f_Mas[[4]]
# extract the model standard error
se_Mas<-f_Mas[[6]]
# extract the coefficients
c_Gar<-f_Gar[[4]]
# extract the model standard error
se_Gar<-f_Gar[[6]]
#imputation by regression with error (remember that se = standard error of model)
```

```r
housing_reg_imp <- housingNumeric
#imputataion with regression
housing_reg_imp[missing_lot,"LotFrontage"] <-
  (c_lot[1] + c_lot[2]*housing_reg_imp[missing_lot,"LotArea"])

housing_reg_imp[missing_Mas,"MasVnrArea"] <-
  (c_Mas[1] + c_Mas[2]*housing_reg_imp[missing_Mas,"BsmtFinSF1"])

housing_reg_imp[missing_Gar,"GarageYrBlt"] <-
  (c_Gar[1] + c_Gar[2]*housing_reg_imp[missing_Gar,"YearBuilt"])

housing_reg_err_imp <- housing_reg_imp

#imputation by regression with error (remember that se = standard error of model)
housing_reg_imp[missing_lot,"LotFrontage"] <-
  housing_reg_err_imp[missing_lot,"LotFrontage"] +
  rnorm(sum(missing_lot),0,se_lot**2)

housing_reg_imp[missing_Mas,"MasVnrArea"] <-
  housing_reg_err_imp[missing_Mas,"MasVnrArea"] +
  rnorm(sum(missing_Mas),0,se_Mas**2)

housing_reg_imp[missing_Gar,"GarageYrBlt"] <-
  housing_reg_err_imp[missing_Gar,"GarageYrBlt"] +
  rnorm(sum(missing_Gar),0,se_Gar**2)

# Transform the dependent variable
housing_reg_err_imp <- housing_reg_err_imp %>%
  mutate(log_SalePrice = log(SalePrice))
```

(a) OLS Model

i. Create a hold-out validation set using the first 100 observations in the data. report the variables, the coefficient estimates, p-values, adjusted R2, AIC,BIC, VIF, and RMSE.

```r
#removing SalePrice because we already have log(SalePrice)
housing_reg_err_imp[["SalePrice"]] <- NULL
#Implementing hold-out
validation_set <- housing_reg_err_imp [1:100, ]
trainig_set <- housing_reg_err_imp[101:1000, ]
#making model 1
model_1 <- lm(log_SalePrice~ LotArea + OverallQual + GrLivArea + GarageArea ,
                                                              data = trainig_set)
#summary(model_1)
lm_predict <- predict(model_1, validation_set)
head(lm_predict)
```

```
##        1        2        3        4        5        6
## 11.79848 11.98201 11.71829 11.89948 11.88612 12.20646
```

```r
lm_val1 <- data.frame(obs = validation_set$log_SalePrice, pred = lm_predict)
defaultSummary(lm_val1)
```

```
##      RMSE  Rsquared       MAE
## 0.1634543 0.8056837 0.1180667
```

```r
AIC(model_1)
```

```
## [1] -775.6474
```

```r
BIC(model_1)
```

```
## [1] -746.8331
```

```r
vif(model_1)
```

```
##     LotArea OverallQual   GrLivArea  GarageArea
##    1.064193    1.718440    1.586775    1.456099
```

```r
#plot(model_1$fitted.values, model_1$residuals, col = "black", pch = 21, bg = "red")
#abline(h=0)


model_2 <- lm(log_SalePrice~ LotArea + OverallQual + YearBuilt + YearRemodAdd +
    BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + X1stFlrSF + X2ndFlrSF +
    KitchenAbvGr + TotRmsAbvGrd + Fireplaces + GarageYrBlt +
    GarageCars + GarageArea + EncPorchSF + Exterior2nd_collapsed +
   Condition1_collapsed + Functional_collapsed + BldgType_collapsed,
                                                  data = trainig_set)
#summary(model_2)
lm_predict <- predict(model_2, validation_set)
head(lm_predict)
```

```
##        1        2        3        4        5        6
## 11.90351 12.06602 11.81343 11.91937 11.91204 12.16372
```

```r
lm_val2 <- data.frame(obs = validation_set$log_SalePrice, pred = lm_predict)
defaultSummary(lm_val2)
```

```
##       RMSE   Rsquared        MAE
## 0.10952648 0.91606724 0.08042787
```

```r
b <- defaultSummary(lm_val2)
AIC(model_2)
```

```
## [1] -1325.659
```

```r
BIC(model_2)
```

```
## [1] -1220.006
```

```r
vif(model_2)
```

```
##               LotArea           OverallQual             YearBuilt
##              1.196225              2.916422              3.483212
##          YearRemodAdd            BsmtFinSF1            BsmtFinSF2
##              1.768668              3.453650              1.294131
##             BsmtUnfSF              X1stFlrSF             X2ndFlrSF
##              3.298541              4.809941              3.733360
##          KitchenAbvGr          TotRmsAbvGrd            Fireplaces
##              1.507374              3.882997              1.552375
##           GarageYrBlt            GarageCars            GarageArea
##              2.698912              5.167495              4.809089
##            EncPorchSF Exterior2nd_collapsed  Condition1_collapsed
##              1.165449              1.071154              1.067313
##  Functional_collapsed    BldgType_collapsed
##              1.176578              1.361586
```

```r
#plot(model_2$fitted.values, model_2$residuals, col = "black", pch = 21, bg = "red")
#abline(h=0)

# Add results to results_df
results_df <- data.frame(
  Model_Name = "OLS",
  Model_Notes = "Ordinary Least Squares Regression",
  Model_Hyper = "N/A",
  Model_RMSE = b[1],
  Model_R2 = b[2]
)
```

interaction part

```r
model_3 <- lm(log_SalePrice~ LotArea * OverallQual * GrLivArea * GarageArea ,
                                                               data = trainig_set)
#summary(model_3)
lm_predict <- predict(model_3, validation_set)
head(lm_predict)
```

```
##        1        2        3        4        5        6
## 11.81822 12.13027 11.73693 11.89729 11.88417 12.21327
```

```r
lm_val3 <- data.frame(obs = validation_set$log_SalePrice, pred = lm_predict)
defaultSummary(lm_val3)
```

```
##      RMSE  Rsquared       MAE
## 0.1544971 0.8271491 0.1138388
```

```r
AIC(model_3)
```

```
## [1] -813.6172
```

```r
BIC(model_3)
```

```
## [1] -731.9765
```

```r
vif(model_3)
```

```
## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif
```

```
##                                  LotArea
##                                4911.3807
##                              OverallQual
##                                 188.7867
##                                GrLivArea
##                                 354.4490
##                               GarageArea
##                                 346.4301
##                      LotArea:OverallQual
##                                7877.0357
##                        LotArea:GrLivArea
##                                7916.7336
##                    OverallQual:GrLivArea
##                                 956.3831
##                       LotArea:GarageArea
##                                3306.1910
##                   OverallQual:GarageArea
##                                 691.1869
##                     GrLivArea:GarageArea
##                                 964.8681
##            LotArea:OverallQual:GrLivArea
##                               12426.7759
##           LotArea:OverallQual:GarageArea
##                                4873.1759
##             LotArea:GrLivArea:GarageArea
##                                5648.5800
##         OverallQual:GrLivArea:GarageArea
##                                1541.1597
## LotArea:OverallQual:GrLivArea:GarageArea
##                                8098.0100
```

```r
#plot(model_3$fitted.values, model_3$residuals, col = "black", pch = 21, bg = "red")
#abline(h=0)

model_4 <- lm(log_SalePrice~ LotArea * OverallQual * GrLivArea + GarageArea ,
                                                          data = trainig_set)
#summary(model_4)
lm_predict <- predict(model_4, validation_set)
head(lm_predict)
```

```
##        1        2        3        4        5        6
## 11.81399 12.07570 11.73001 11.90136 11.88866 12.21055
```

```r
lm_val4 <- data.frame(obs = validation_set$log_SalePrice, pred = lm_predict)
defaultSummary(lm_val4)
```

```
##      RMSE  Rsquared       MAE
## 0.1578111 0.8194585 0.1134000
```

```r
b <- defaultSummary(lm_val4)
AIC(model_4)
```

```
## [1] -796.1786
```

```r
BIC(model_4)
```

```
## [1] -748.1547
```

```r
vif(model_4)
```

```
## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif
```

```
##                        LotArea                    OverallQual
##                     392.787984                      25.517019
##                      GrLivArea                     GarageArea
##                      54.509470                       1.517609
##            LotArea:OverallQual              LotArea:GrLivArea
##                     395.023826                     536.047337
##          OverallQual:GrLivArea LotArea:OverallQual:GrLivArea
##                     113.509378                     516.847145
```

```r
# Add results to results_df
results_df <- rbind(results_df, data.frame(
  Model_Name = "OLS",
  Model_Notes = "lm + 2 way interactions",
  Model_Hyper = "N/A",
  Model_RMSE = b[1],
  Model_R2 = b[2]
))
```

As we can see the best result is related to model_2

```r
summary(model_2)
```

```
##
## Call:
## lm(formula = log_SalePrice ~ LotArea + OverallQual + YearBuilt +
```

```
##     YearRemodAdd + BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF + X1stFlrSF +
##     X2ndFlrSF + KitchenAbvGr + TotRmsAbvGrd + Fireplaces + GarageYrBlt +
##     GarageCars + GarageArea + EncPorchSF + Exterior2nd_collapsed +
##     Condition1_collapsed + Functional_collapsed + BldgType_collapsed,
##     data = trainig_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71478 -0.05571  0.00554  0.06447  0.36170
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.289e+00  5.349e-01   6.150 1.18e-09 ***
## LotArea                2.603e-06  4.057e-07   6.415 2.30e-10 ***
## OverallQual            7.189e-02  4.927e-03  14.590  < 2e-16 ***
## YearBuilt              2.142e-03  2.458e-04   8.713  < 2e-16 ***
## YearRemodAdd           2.302e-03  2.521e-04   9.131  < 2e-16 ***
## BsmtFinSF1             2.046e-04  1.755e-05  11.662  < 2e-16 ***
## BsmtFinSF2             1.715e-04  2.882e-05   5.951 3.85e-09 ***
## BsmtUnfSF              8.848e-05  1.679e-05   5.269 1.73e-07 ***
## X1stFlrSF              2.704e-04  2.376e-05  11.377  < 2e-16 ***
## X2ndFlrSF              2.660e-04  1.724e-05  15.425  < 2e-16 ***
## KitchenAbvGr          -8.178e-02  2.267e-02  -3.608 0.000326 ***
## TotRmsAbvGrd           3.520e-03  4.744e-03   0.742 0.458198
## Fireplaces             3.820e-02  7.405e-03   5.159 3.07e-07 ***
## GarageYrBlt           -6.662e-04  2.707e-04  -2.461 0.014036 *
## GarageCars             2.706e-02  1.239e-02   2.184 0.029216 *
## GarageArea             1.322e-04  4.288e-05   3.083 0.002117 **
## EncPorchSF             1.294e-04  5.138e-05   2.518 0.011978 *
## Exterior2nd_collapsed  1.859e-03  2.452e-03   0.758 0.448703
## Condition1_collapsed   1.288e-02  6.722e-03   1.917 0.055593 .
## Functional_collapsed   3.152e-02  5.910e-03   5.334 1.22e-07 ***
## BldgType_collapsed    -1.687e-02  3.704e-03  -4.554 6.01e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1144 on 879 degrees of freedom
## Multiple R-squared:  0.9027, Adjusted R-squared:  0.9005
## F-statistic: 407.7 on 20 and 879 DF,  p-value: < 2.2e-16
```

```r
defaultSummary(lm_val2)
```

```
##       RMSE   Rsquared        MAE
## 0.10952648 0.91606724 0.08042787
```

```r
AIC(model_2)
```

```
## [1] -1325.659
```

```r
BIC(model_2)
```

```
## [1] -1220.006
```

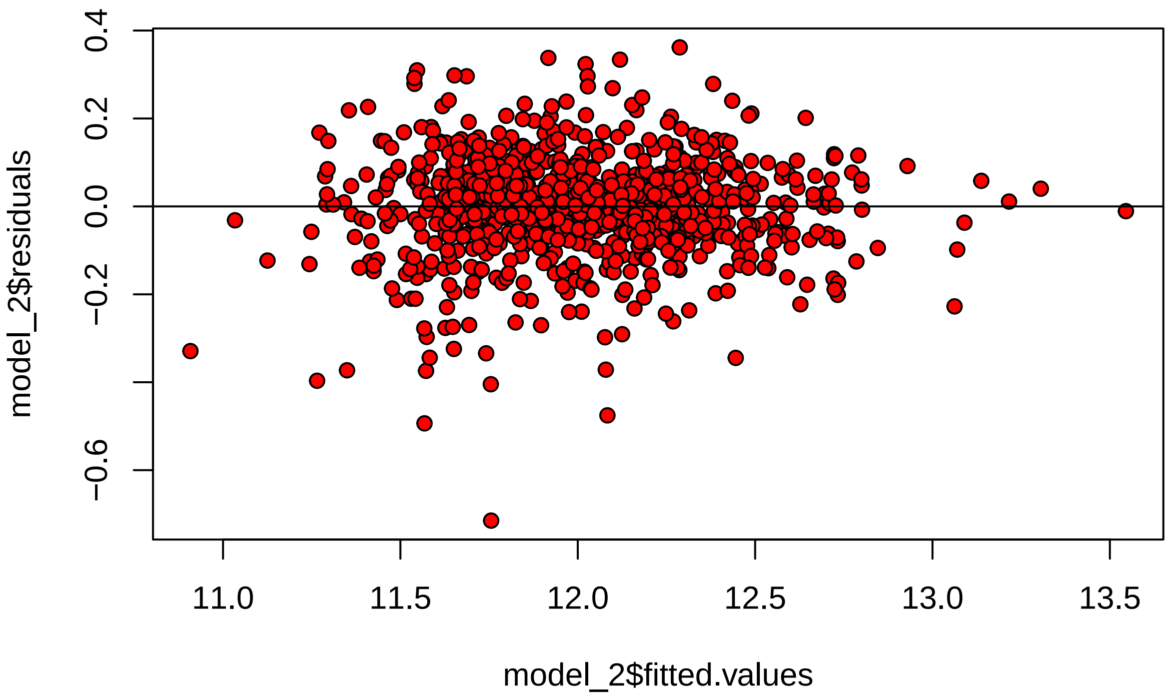
```r
vif(model_2)
```

```
##                LotArea            OverallQual              YearBuilt 
##               1.196225               2.916422               3.483212 
##           YearRemodAdd             BsmtFinSF1             BsmtFinSF2 
##               1.768668               3.453650               1.294131 
##              BsmtUnfSF              X1stFlrSF              X2ndFlrSF 
##               3.298541               4.809941               3.733360 
##           KitchenAbvGr           TotRmsAbvGrd             Fireplaces 
##               1.507374               3.882997               1.552375 
##            GarageYrBlt             GarageCars             GarageArea 
##               2.698912               5.167495               4.809089 
##             EncPorchSF Exterior2nd_collapsed  Condition1_collapsed 
##               1.165449               1.071154               1.067313 
##   Functional_collapsed      BldgType_collapsed 
##               1.176578               1.361586
```

The varibles are LotArea, OverallQual, YearBuilt , YearRemodAdd , BsmtFinSF1 , BsmtFinSF2 , BsmtUnfSF , X1stFlrSF , X2ndFlrSF , KitchenAbvGr , TotRmsAbvGrd , Fireplaces , GarageYrBlt , GarageCars , GarageArea , EncPorchSF , Exterior2nd_collapsed , Condition1_collapsed , Functional_collapsed , BldgType_collapsed and the coefficient estimates can be seen in the upper table. The p-value is so low so not to reject null.In the context of regression analysis, p-values are associated with individual predictor variables. They indicate whether each predictor variable is statistically significant in explaining the variation in the dependent variable. Lower p-values (typically less than a chosen significance level, e.g., 0.05) suggest that a variable is more likely to be relevant.

ii. Provide a complete analysis of the residuals.

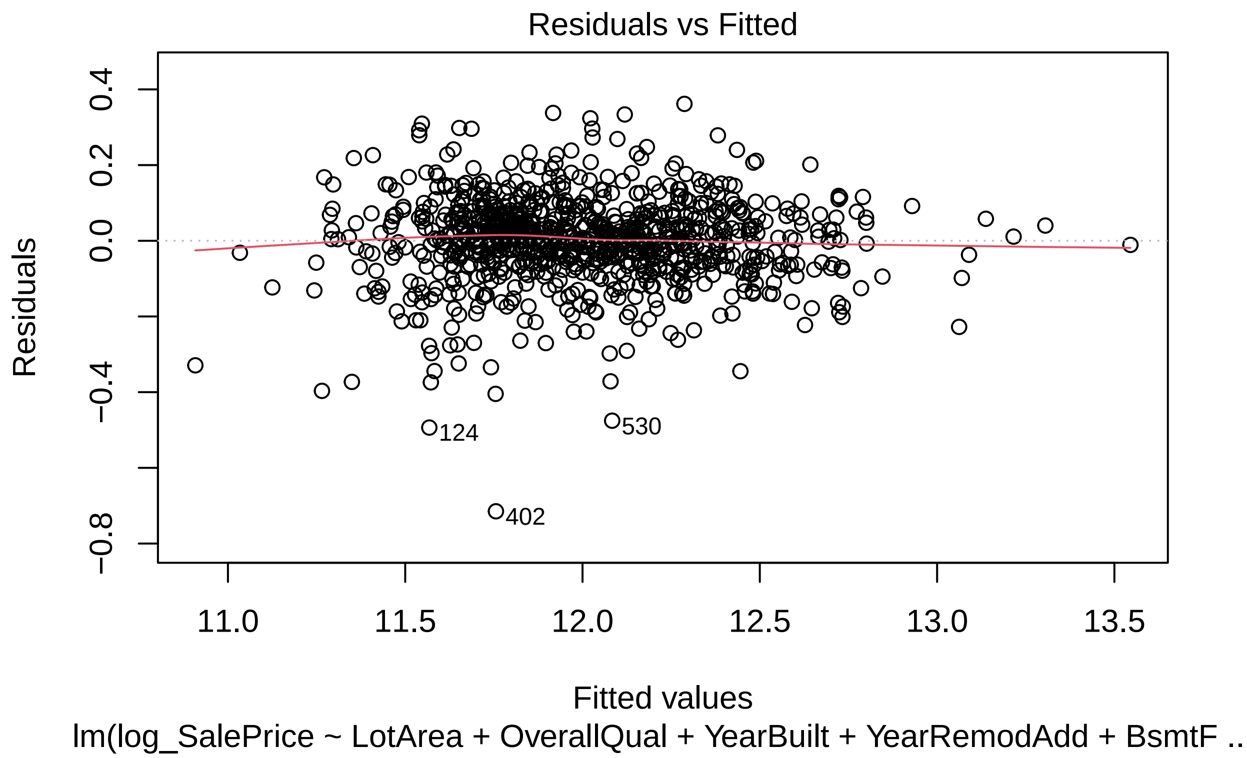
```r
plot(model_2$fitted.values, model_2$residuals, col = "black", pch = 21, bg = "red")
abline(h=0)
```

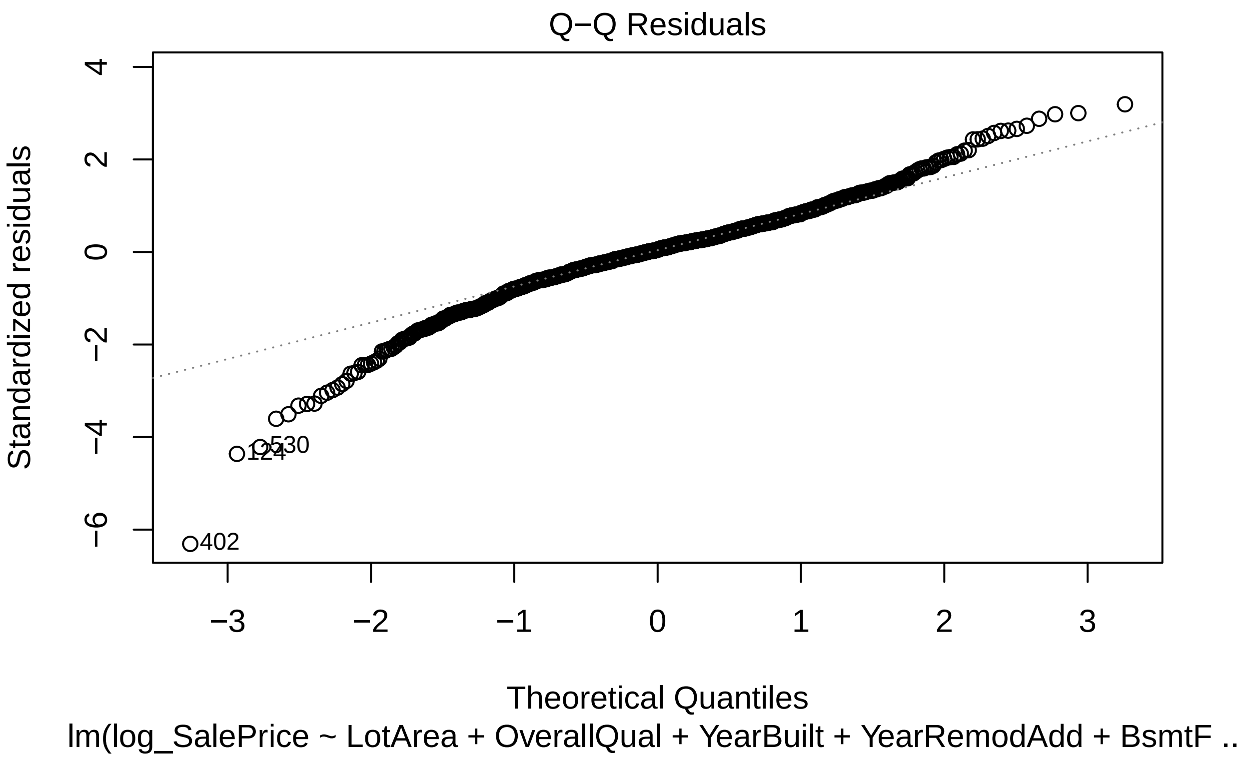



```r
#testing for non-constant variance, or heteroscedasticity with respect to 2.2e-16 p value
ncvTest(model_2)
```

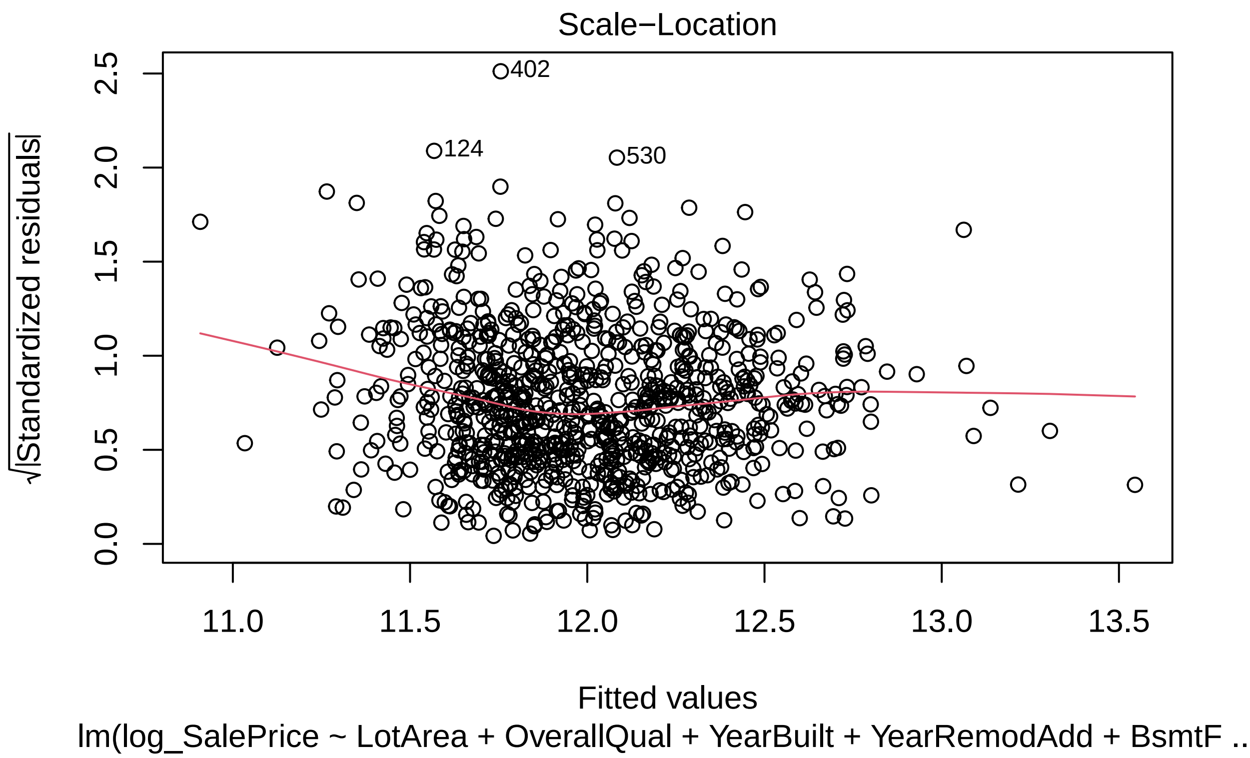
```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 22.96214, Df = 1, p = 1.6522e-06
```

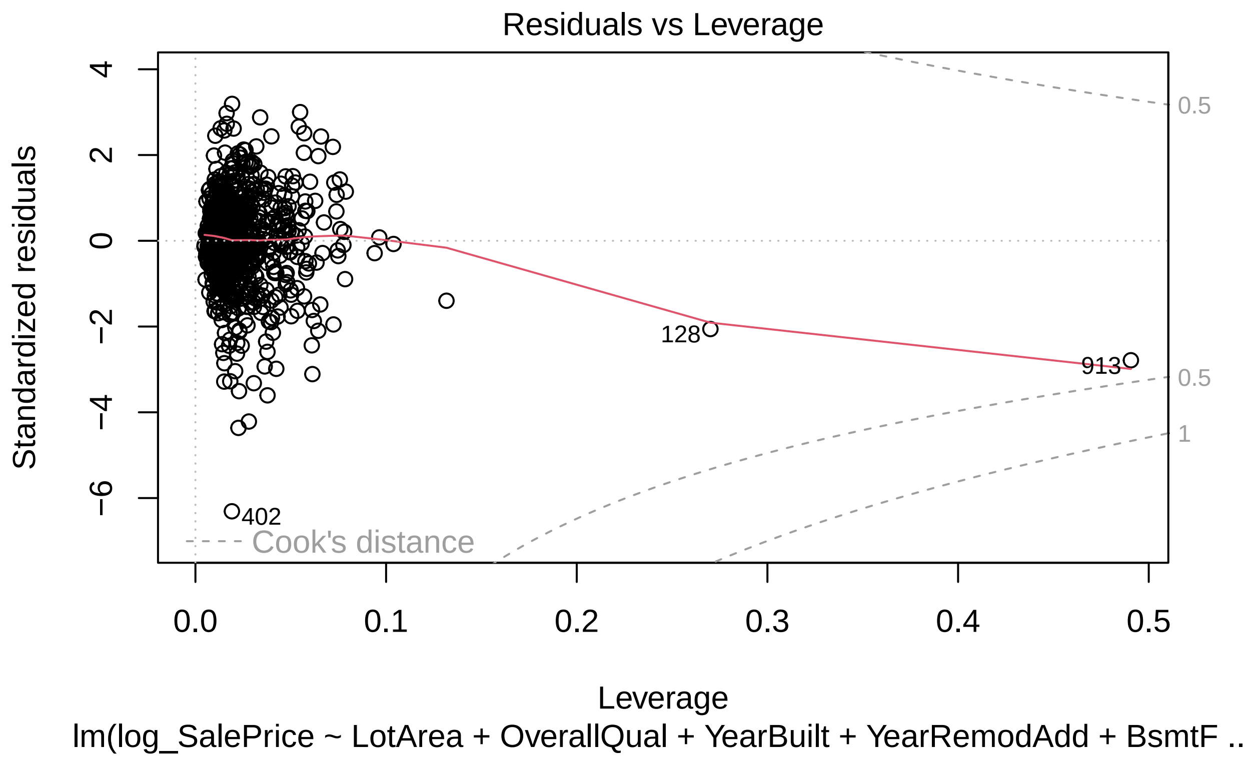
```r
## P value is less than 0.05 so we can say that we dont have enough evidence to
#reject the null hypothesis (heteroscedasticity)

#automatic plots from R for linear models
plot(model_2)
```


Residuals vs Fitted
Residuals
0.4
0.2
0.0
−0.4
−0.8
124
530
402
11.0
11.5
12.0
12.5
13.0
13.5
Fitted values
lm(log_SalePrice ~ LotArea + OverallQual + YearBuilt + YearRemodAdd + BsmtF ...



Q−Q Residuals
Standardized residuals
Theoretical Quantiles
lm(log_SalePrice ~ LotArea + OverallQual + YearBuilt + YearRemodAdd + BsmtF ...
4
2
0
−2
−4
−6
−3
−2
−1
0
1
2
3
530
124
402

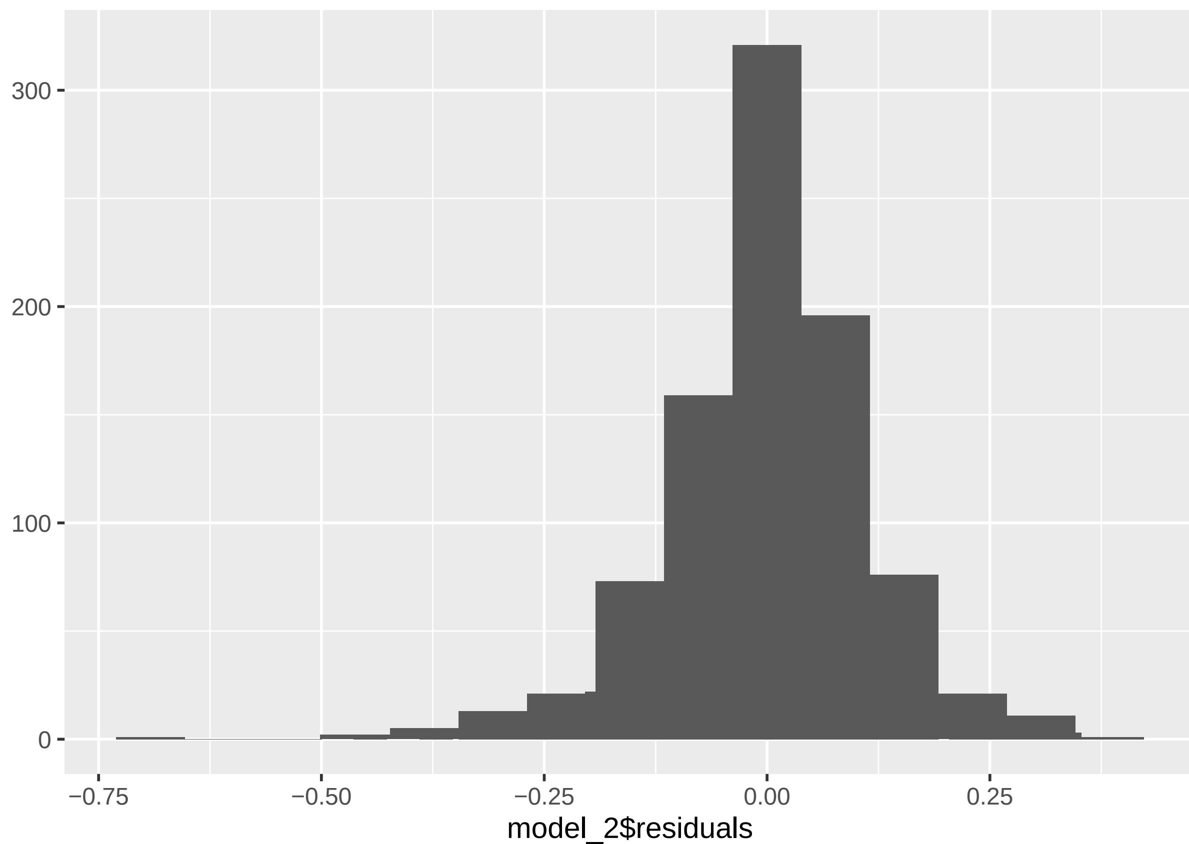

Scale−Location
402
124
530
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
11.0
11.5
12.0
12.5
13.0
13.5
Fitted values
lm(log_SalePrice ~ LotArea + OverallQual + YearBuilt + YearRemodAdd + BsmtF ...



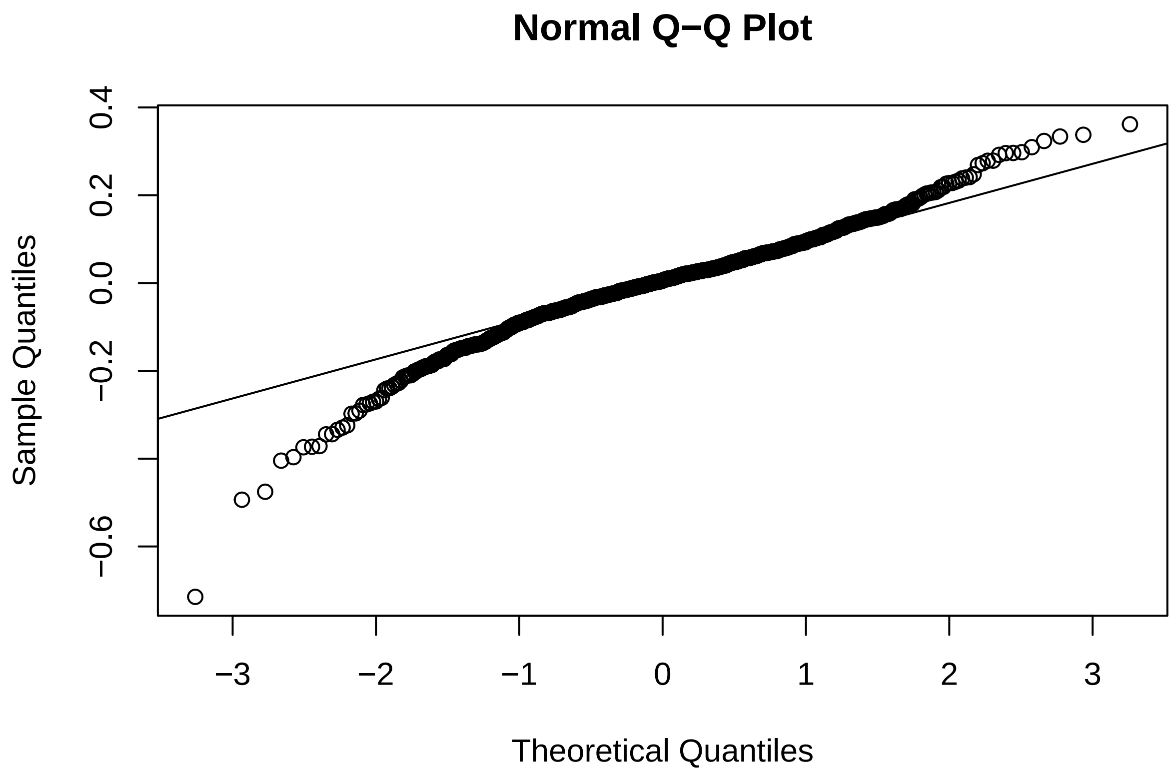

```r
#histogram of residuals
qplot(model_2$residuals) + geom_histogram(bins=15)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

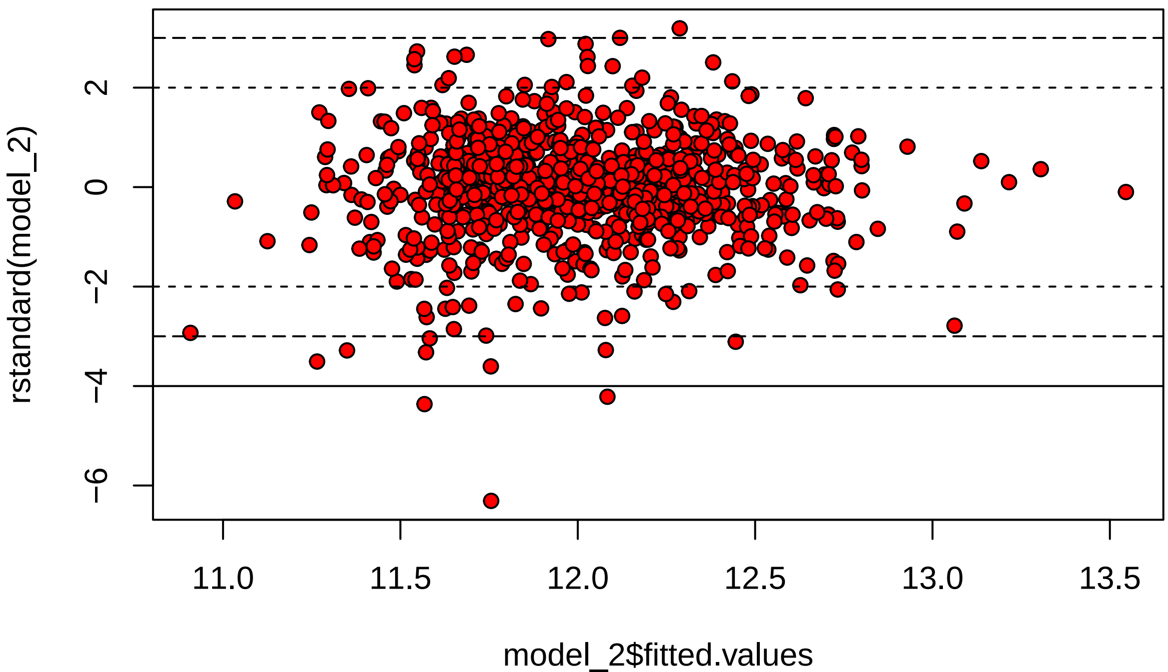
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



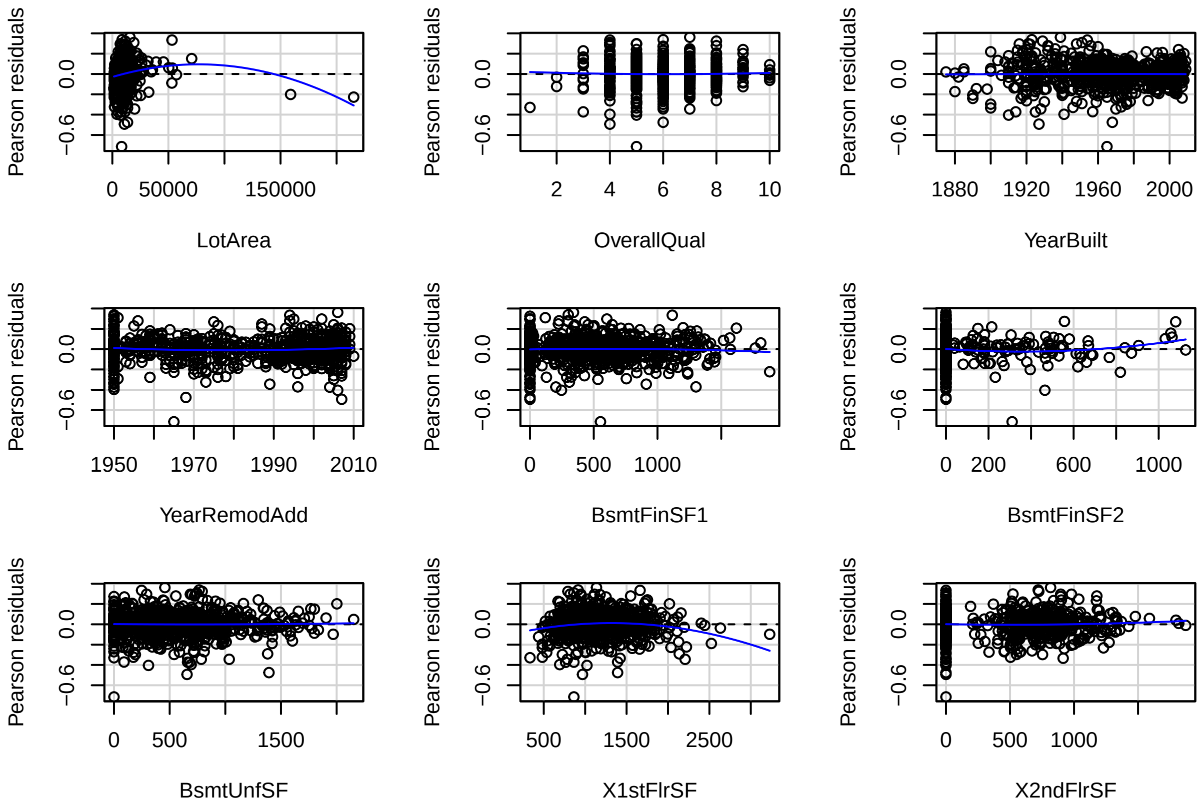

```r
#We can say that data shows somehow normal distribution
#qq plot of residuals
qqnorm(model_2$residuals)
qqline(model_2$residuals)
```

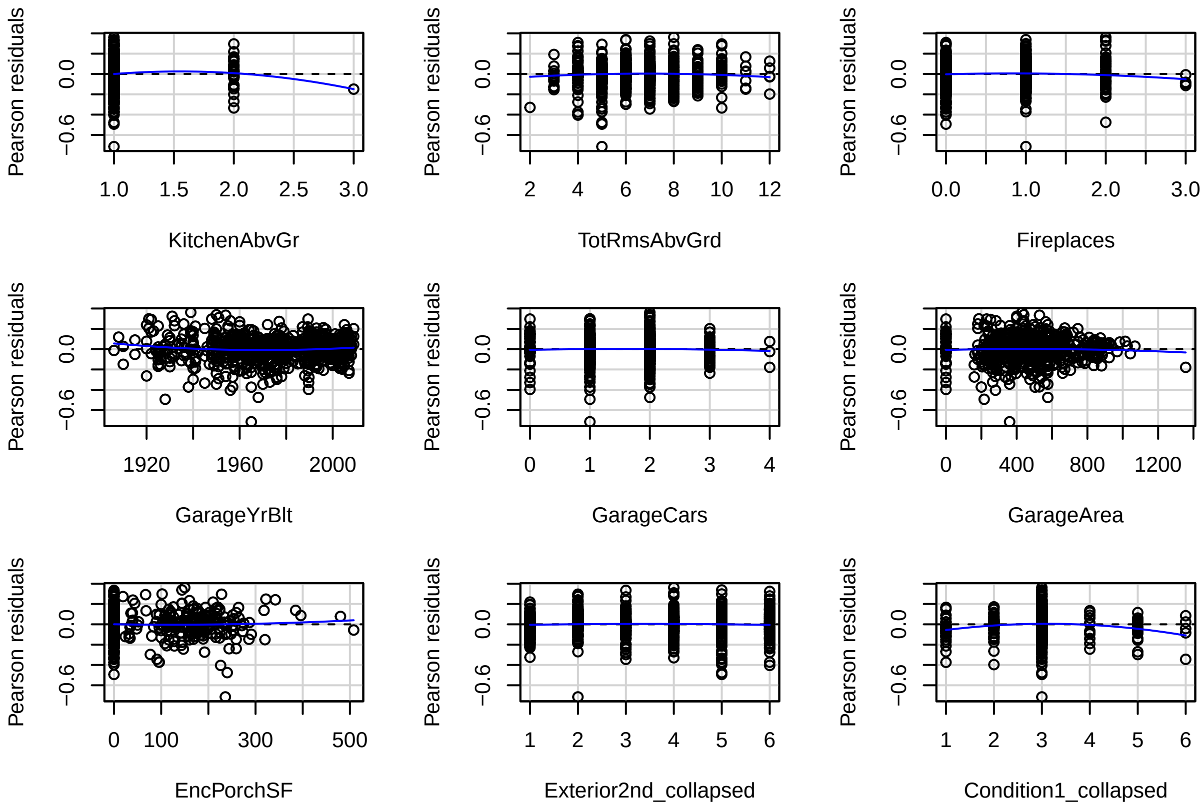


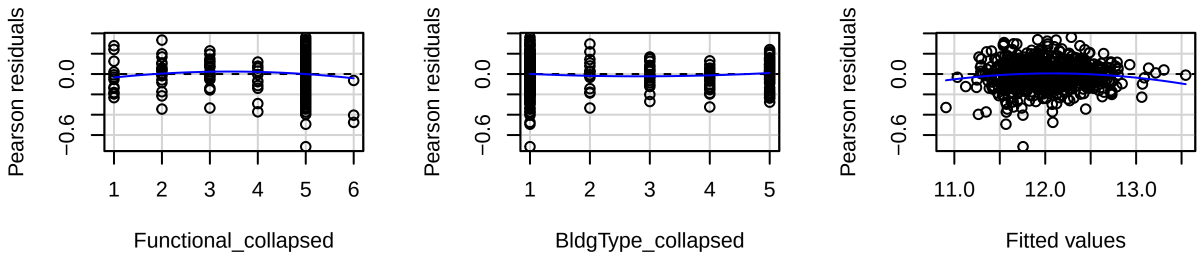

```r
#STANDARDIZED residuals (index plot)
#+-2(something unusual) +-3(really strange) +-4(outerspace weird)

plot(model_2$fitted.values, rstandard(model_2),col = "black", pch = 21, bg = "red")
abline(h=c(-2,2), lty = 2)
abline(h=c(-3,3), lty = 5)
abline(h=c(-4, 4), lty = 1)
```




```r
# residual plots from car package
residualPlots(model_2)
```

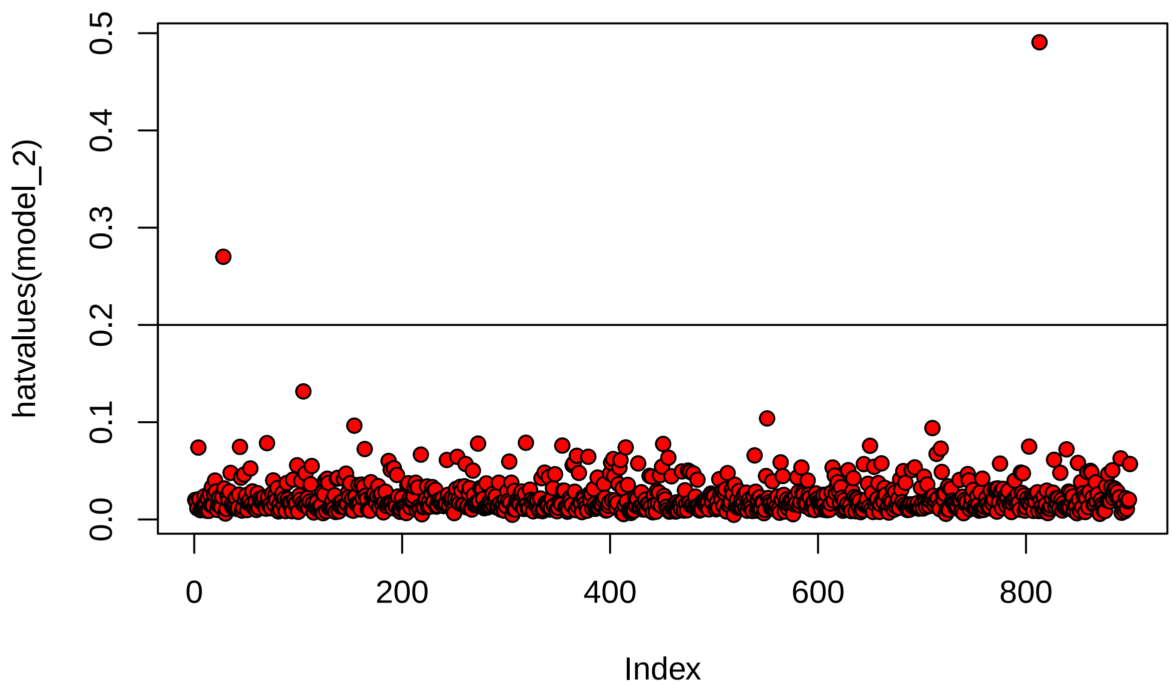

Pearson residuals
0.0
−0.6
0
50000
150000
LotArea
2
4
6
8
10
OverallQual
1880
1920
1960
2000
YearBuilt
1950
1970
1990
2010
YearRemodAdd
500
1000
BsmtFinSF1
200
600
BsmtFinSF2
1500
BsmtUnfSF
2500
X1stFlrSF
X2ndFlrSF


Pearson residuals
0.0
−0.6
1.0
1.5
2.0
2.5
3.0
KitchenAbvGr
2
4
6
8
10
12
TotRmsAbvGrd
0.0
1.0
2.0
3.0
Fireplaces
1920
1960
2000
GarageYrBlt
0
1
2
3
4
GarageCars
0
400
800
1200
GarageArea
0
100
300
500
EncPorchSF
1
2
3
4
5
6
Exterior2nd_collapsed
1
2
3
4
5
6
Condition1_collapsed




```
##                         Test stat Pr(>|Test stat|)
## LotArea                   -5.1813        2.735e-07 ***
## OverallQual                0.5565         0.578018
## YearBuilt                 -0.2545         0.799183
## YearRemodAdd               2.2250         0.026336 *
## BsmtFinSF1                -0.9859         0.324431
## BsmtFinSF2                 2.3287         0.020102 *
## BsmtUnfSF                  0.3738         0.708647
## X1stFlrSF                 -4.2367        2.508e-05 ***
## X2ndFlrSF                  0.9385         0.348238
## KitchenAbvGr              -1.3988         0.162220
## TotRmsAbvGrd              -1.3957         0.163172
## Fireplaces                -1.6445         0.100436
## GarageYrBlt                3.1349         0.001776 **
## GarageCars                -0.7283         0.466594
## GarageArea                -0.8211         0.411832
## EncPorchSF                 0.6672         0.504834
## Exterior2nd_collapsed     -1.0374         0.299825
## Condition1_collapsed      -4.2279        2.606e-05 ***
## Functional_collapsed      -1.7117         0.087299 .
## BldgType_collapsed         2.1093         0.035199 *
## Tukey test                -2.6033         0.009232 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

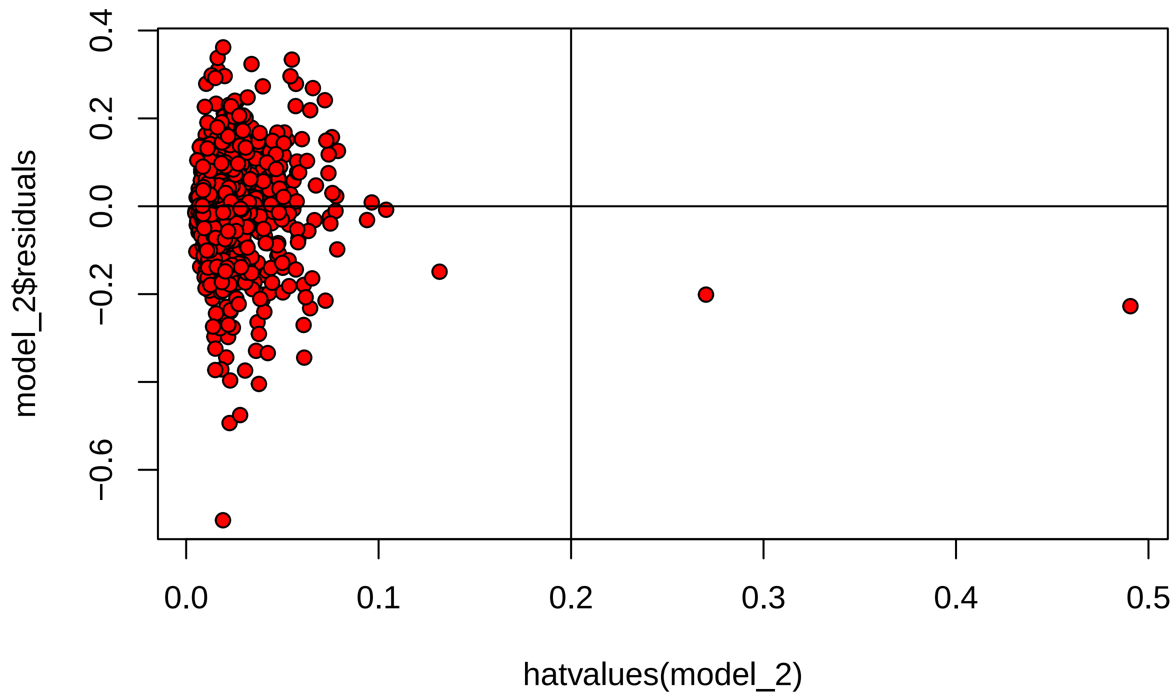
```r
#Now move on to the leverage analysis. So we need the hat matrix
# The leverage of an observation measures its ability to move the regression model all by itself by
#simply moving in the y direction

plot(hatvalues(model_2) ,col = "black", pch = 21, bg = "red")
abline(abline(h = 2*5/50))
```




```r
#which obs exceed the rule-of-thumb values?
hatvalues(model_2)[hatvalues(model_2)>0.2]
```

```
##       128       913
## 0.2701038 0.4906397
```

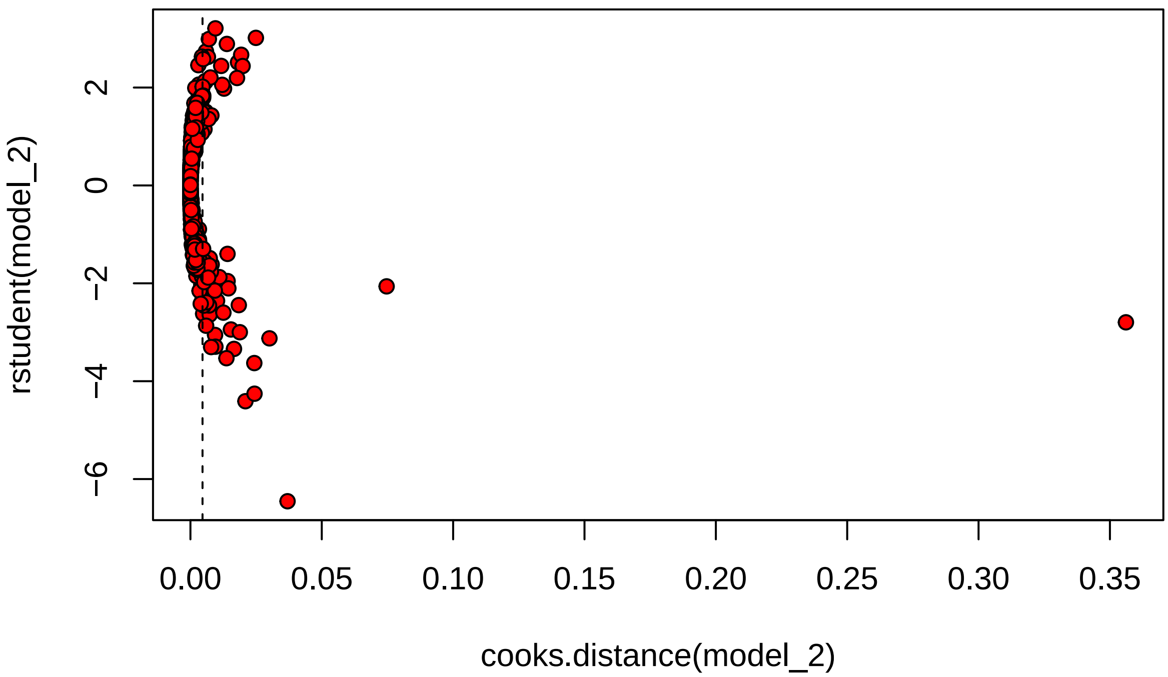
```r
#plot residuals vs. hatvalues
plot(hatvalues(model_2),model_2$residuals,col = "black", pch = 21, bg = "red")
abline(h=0,v=2*5/50)
```



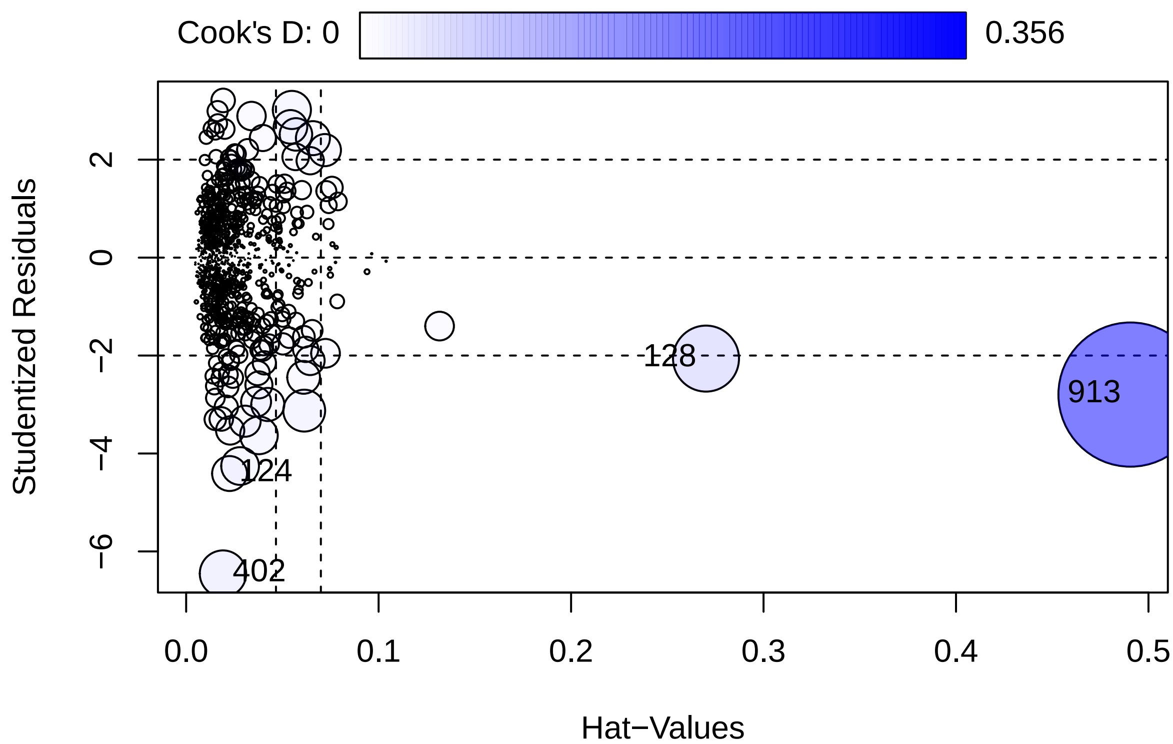

```r
#outlier test
outlierTest(model_2)
```

```
##      rstudent unadjusted p-value Bonferroni p
## 402 -6.452896         1.8120e-10   1.6308e-07
## 124 -4.409013         1.1671e-05   1.0504e-02
## 530 -4.255891         2.3061e-05   2.0755e-02
```

```r
#Results indicated that there are no statistically significant outliers in our model based on p 0.05

# Now lets look at the Cook's distance

#A value greater 1 is usually considered influential.

plot(cooks.distance(model_2), rstudent(model_2), col = "black", pch = 21, bg = 'red')
abline(v=(4/(900-30-1)), lty=2)
```




```r
#Now we can look at the influence plot


#influence measures
#influence.measures(model_2)

#influence plot
influencePlot(model_2)
```




```
##        StudRes        Hat      CookD
## 124 -4.409013 0.02256133 0.02092774
## 128 -2.062451 0.27010381 0.07468146
## 402 -6.452896 0.01912990 0.03696258
## 913 -2.797088 0.49063974 0.35610014
```

```r
vif(model_2)
```

```
##              LotArea           OverallQual             YearBuilt
##             1.196225              2.916422              3.483212
##         YearRemodAdd            BsmtFinSF1            BsmtFinSF2
##             1.768668              3.453650              1.294131
##            BsmtUnfSF             X1stFlrSF             X2ndFlrSF
##             3.298541              4.809941              3.733360
##         KitchenAbvGr          TotRmsAbvGrd            Fireplaces
##             1.507374              3.882997              1.552375
##          GarageYrBlt            GarageCars            GarageArea
##             2.698912              5.167495              4.809089
##          EncPorchSF Exterior2nd_collapsed  Condition1_collapsed
##             1.165449              1.071154              1.067313
##   Functional_collapsed     BldgType_collapsed
##             1.176578              1.361586
```

In our analysis of residuals, several key observations can be made: Linearity Assessment: When we examine the proximity of the red regression line to the dashed line, it is apparent that linearity appears to be maintained. This suggests that the relationship between the predictors and the response variable is roughly linear.

Outliers Identification: There are a couple of data points with notably high residual values, specifically points 128 and 913. These points could be considered outliers due to their substantial deviation from the overall trend.

QQ Plot Evaluation: While the points on the QQ plot do not perfectly align along a straight line, the deviations from linearity are not significant enough to raise concerns. This indicates that the assumption of normality in the residuals is reasonably met.

Influence of Fitted Values: The fitted values from the model do not appear to have a substantial impact on the average magnitude of the standardized residuals. This suggests that the model's predictions are not exerting undue influence on the residuals.

Leverage Points: It's worth noting that certain data points exhibit high leverage. These points could have a significant impact on the model if removed. This is an important consideration because removing influential points could substantially alter the model's results.

Cook's Distance: In the context of Cook's distance, represented by the grey dotted line in the image, it measures the effect of eliminating individual data points. Points located outside the dotted line have a substantial impact on the model. In this specific scenario, no data points fall outside the dotted line.

Residual Distribution: The distribution of the residuals tends to follow a normal distribution, as indicated by both the residual plots and the histogram. However, there are some extreme outliers within the residual data that warrant attention and potential treatment before proceeding with further modeling efforts.**

Part B: PLS model

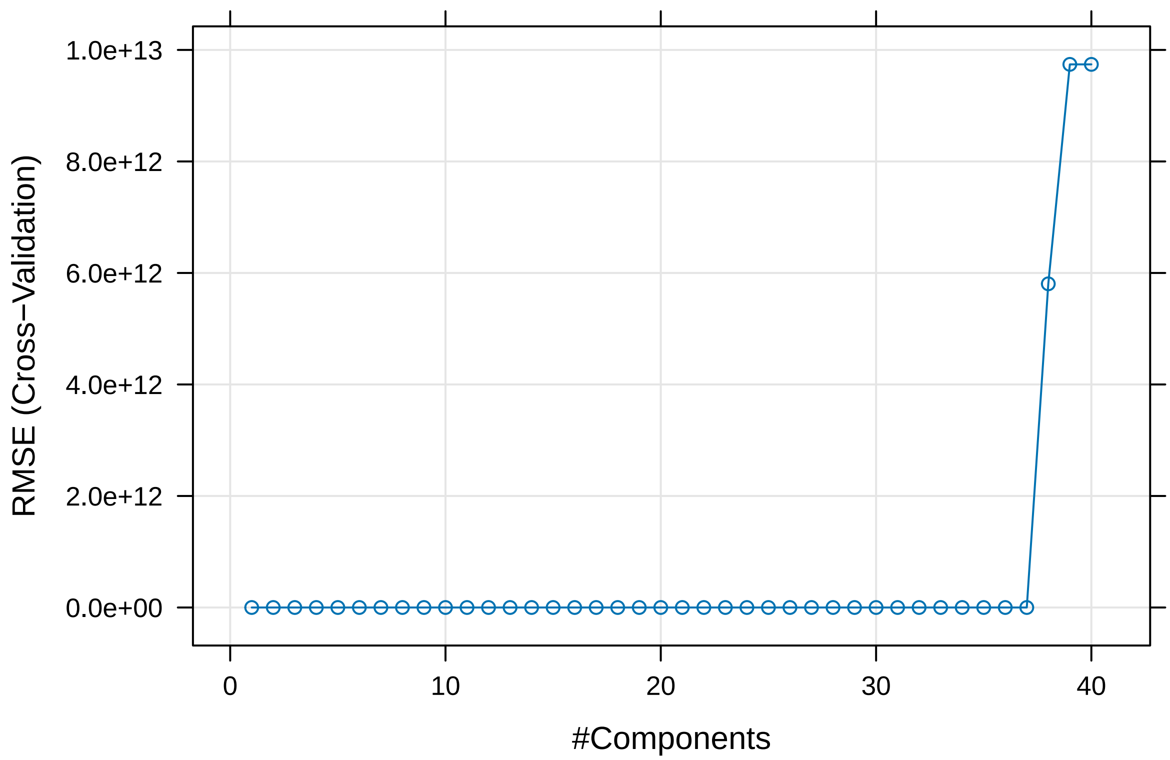
```
predictors <- subset(housing_reg_err_imp, select = -log_SalePrice)
target <- housing_reg_err_imp$log_SalePrice

# Create a grid of components to tune
tune_grid <- expand.grid(ncomp = 1:40)

# Create a PLS model with cross-validation
cv_results <- train(
  x = predictors,
  y = target,
  method = "pls",
  tuneGrid = tune_grid,
  trControl = trainControl(method = "cv", number =40, verboseIter = TRUE),
  metric = "RMSE"
)
```

```
## + Fold01: ncomp=40
## - Fold01: ncomp=40
## + Fold02: ncomp=40
## - Fold02: ncomp=40
## + Fold03: ncomp=40
## - Fold03: ncomp=40
## + Fold04: ncomp=40
## - Fold04: ncomp=40
## + Fold05: ncomp=40
## - Fold05: ncomp=40
```

```
## + Fold06: ncomp=40
## - Fold06: ncomp=40
## + Fold07: ncomp=40
## - Fold07: ncomp=40
## + Fold08: ncomp=40
## - Fold08: ncomp=40
## + Fold09: ncomp=40
## - Fold09: ncomp=40
## + Fold10: ncomp=40
## - Fold10: ncomp=40
## + Fold11: ncomp=40
## - Fold11: ncomp=40
## + Fold12: ncomp=40
## - Fold12: ncomp=40
## + Fold13: ncomp=40
## - Fold13: ncomp=40
## + Fold14: ncomp=40
## - Fold14: ncomp=40
## + Fold15: ncomp=40
## - Fold15: ncomp=40
## + Fold16: ncomp=40
## - Fold16: ncomp=40
## + Fold17: ncomp=40
## - Fold17: ncomp=40
## + Fold18: ncomp=40
## - Fold18: ncomp=40
## + Fold19: ncomp=40
## - Fold19: ncomp=40
## + Fold20: ncomp=40
## - Fold20: ncomp=40
## + Fold21: ncomp=40
## - Fold21: ncomp=40
## + Fold22: ncomp=40
## - Fold22: ncomp=40
## + Fold23: ncomp=40
## - Fold23: ncomp=40
## + Fold24: ncomp=40
## - Fold24: ncomp=40
## + Fold25: ncomp=40
## - Fold25: ncomp=40
## + Fold26: ncomp=40
## - Fold26: ncomp=40
## + Fold27: ncomp=40
## - Fold27: ncomp=40
## + Fold28: ncomp=40
## - Fold28: ncomp=40
## + Fold29: ncomp=40
## - Fold29: ncomp=40
## + Fold30: ncomp=40
## - Fold30: ncomp=40
## + Fold31: ncomp=40
## - Fold31: ncomp=40
## + Fold32: ncomp=40
## - Fold32: ncomp=40
```

```
## + Fold33: ncomp=40
## - Fold33: ncomp=40
## + Fold34: ncomp=40
## - Fold34: ncomp=40
## + Fold35: ncomp=40
## - Fold35: ncomp=40
## + Fold36: ncomp=40
## - Fold36: ncomp=40
## + Fold37: ncomp=40
## - Fold37: ncomp=40
## + Fold38: ncomp=40
## - Fold38: ncomp=40
## + Fold39: ncomp=40
## - Fold39: ncomp=40
## + Fold40: ncomp=40
## - Fold40: ncomp=40
## Aggregating results
## Selecting tuning parameters
## Fitting ncomp = 30 on full training set
```

```
# Plot RMSE vs. Number of Components
plot(cv_results)
```




```
# Select the optimal number of components based on the lowest RMSE
optimal_ncomp <- cv_results$bestTune$ncomp
print(optimal_ncomp)
```

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

```r
final_pls_model <- plsr(log_SalePrice ~ ., data = housing_reg_err_imp,
                                                      ncomp = optimal_ncomp)
summary(final_pls_model)
```

```
## Data:    X dimension: 1000 39
##  Y dimension: 1000 1
## Fit method: kernelpls
## Number of components considered: 30
## TRAINING: % variance explained
##                1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X               98.817    99.24    99.49    99.58    99.83    99.88    99.89
## log_SalePrice    8.966    72.12    74.43    77.37    77.94    78.66    81.13
##                8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X                99.91    99.93     99.96     99.97     99.99     99.99
## log_SalePrice    83.04    83.71     84.20     84.85     84.98     85.25
##                14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## X                100.00    100.00    100.00    100.00    100.00    100.00
## log_SalePrice     85.64     86.01     86.09     86.13     86.37     88.45
##                20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## X                100.00     100.0    100.00    100.00    100.00     100.0
## log_SalePrice     89.94      90.4     90.47     90.53     90.57      90.6
##                26 comps  27 comps  28 comps  29 comps  30 comps
## X                100.00    100.00    100.00    100.00    100.00
## log_SalePrice     90.62     90.66     90.67     90.68     90.68
```

```r
cat("Optimal Number of Components:", optimal_ncomp, "\n")
```

```
## Optimal Number of Components: 30
```

```r
cat("Cross-Validated RMSE Estimate:", min(cv_results$results$RMSE), "\n")
```

```
## Cross-Validated RMSE Estimate: 0.1142139
```

```r
# Add results to results_df
results_df <- rbind(results_df, data.frame(
  Model_Name = "PLS",
  Model_Notes = "pls",
  Model_Hyper = optimal_ncomp,
  Model_RMSE = min(cv_results$results$RMSE),
  Model_R2 = max(cv_results$results$Rsquared)
))
```

```r
pls_model_1 <- plsr(log_SalePrice ~ ., data = housing_reg_err_imp, ncomp = 26)
pls_sum <- summary(pls_model_1)
```

```
## Data:    X dimension: 1000 39
##  Y dimension: 1000 1
## Fit method: kernelpls
## Number of components considered: 26
```

```
## TRAINING: % variance explained
##                 1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X                98.817    99.24    99.49    99.58    99.83    99.88    99.89
## log_SalePrice     8.966    72.12    74.43    77.37    77.94    78.66    81.13
##                 8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X                 99.91    99.93     99.96     99.97     99.99     99.99
## log_SalePrice     83.04    83.71     84.20     84.85     84.98     85.25
##                 14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## X                 100.00    100.00    100.00    100.00    100.00    100.00
## log_SalePrice      85.64     86.01     86.09     86.13     86.37     88.45
##                 20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
## X                 100.00     100.0    100.00    100.00    100.00     100.0
## log_SalePrice      89.94      90.4     90.47     90.53     90.57      90.6
##                 26 comps
## X                 100.00
## log_SalePrice      90.62
```

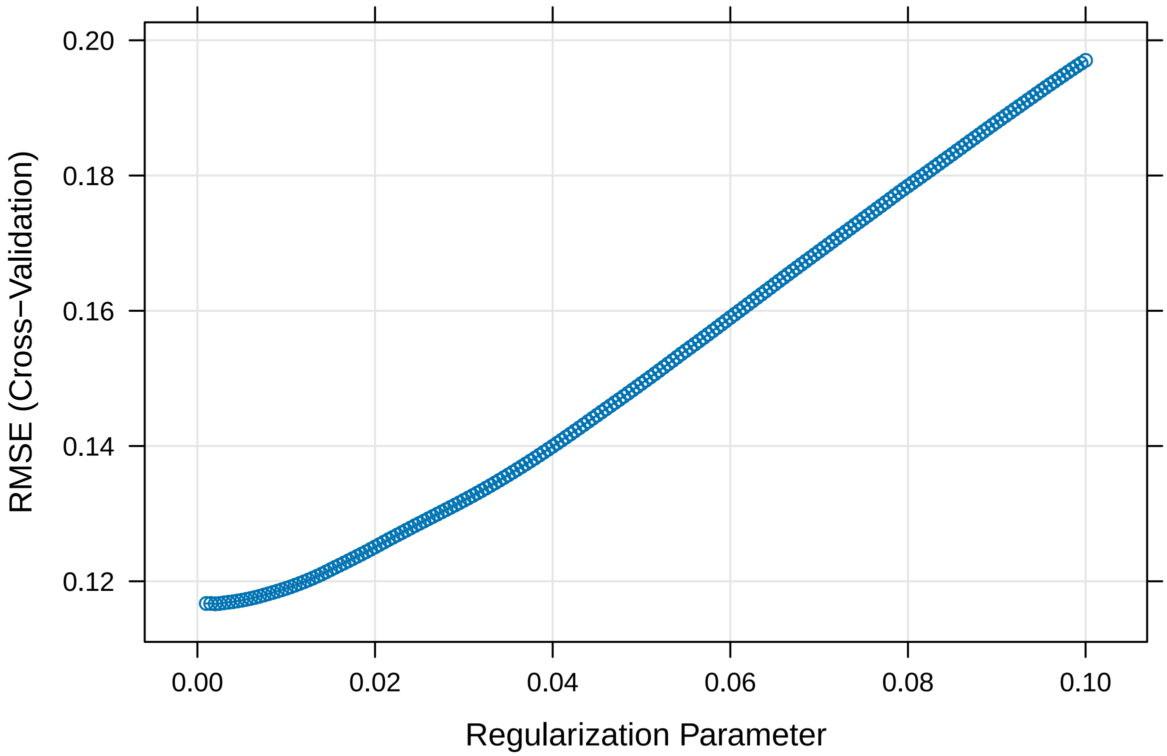
```
rmse <- sqrt(mean(pls_model_1$residuals^2))
print(rmse)
```

```
## [1] 0.1545739
```

```
plot(pls_model_1)
```

Part C:LASSO Model

```r
set.seed(27042018)
X <- subset(housing_reg_err_imp, select = -log_SalePrice)
X <- as.matrix(X)
Y <- housing_reg_err_imp$log_SalePrice
my_control <-trainControl(method="cv", number=5)
lassoGrid <- expand.grid(alpha = 1, lambda = seq(0.001,0.1,by = 0.0005))

lasso_mod <- train(x=X, y=Y, method='glmnet', trControl= my_control, tuneGrid=lassoGrid)
plot(lasso_mod)
```




```r
bestLambda <- lasso_mod$bestTune$lambda
bestAlpha <- lasso_mod$bestTune$alpha
final_lasso <- glmnet(X,Y, alpha = bestAlpha, lambda = bestLambda)
coef_lasso <- coef(final_lasso)
non_zero_coef <- coef_lasso[coef_lasso[,1] != 0, , drop = FALSE]
print(non_zero_coef)
```

```
## 34 x 1 sparse Matrix of class "dgCMatrix"
##                           s0
## (Intercept)     5.592673e+00
## LotFrontage     2.654112e-04
## LotArea         2.412079e-06
## OverallQual     7.337206e-02
## YearBuilt       2.002724e-03
## YearRemodAdd    2.132625e-03
```

```
## BsmtFinSF1              9.867145e-05
## BsmtFinSF2              4.733191e-05
## TotalBsmtSF             9.695586e-05
## X1stFlrSF               5.193190e-07
## LowQualFinSF           -8.207237e-05
## GrLivArea               2.600751e-04
## BsmtFullBath            1.302086e-02
## BsmtHalfBath            1.513665e-02
## HalfBath                2.047322e-03
## BedroomAbvGr           -3.203566e-03
## KitchenAbvGr           -7.248573e-02
## TotRmsAbvGrd            1.294717e-03
## Fireplaces              3.810588e-02
## GarageYrBlt            -4.246761e-04
## GarageCars              2.601591e-02
## GarageArea              1.279964e-04
## WoodDeckSF              7.224984e-05
## OpenPorchSF             6.918683e-05
## EncPorchSF              1.264242e-04
## PoolArea                3.301159e-05
## MiscVal                -5.676666e-06
## YrSold                 -1.069492e-03
## Neighborhood_collapsed  1.697937e-03
## HouseStyle_collapsed   -4.069389e-04
## Exterior1st_collapsed   1.937216e-03
## Condition1_collapsed    7.043012e-03
## Functional_collapsed    2.566113e-02
## BldgType_collapsed     -1.474100e-02
```

```
print(lasso_mod$results[lasso_mod$results$lambda == bestLambda,])
```

```
##   alpha lambda      RMSE  Rsquared        MAE      RMSESD RsquaredSD
## 3     1  0.002 0.1166649 0.8962221 0.08444249 0.007477959 0.01624865
##         MAESD
## 3 0.003525047
```

```
results_df <- rbind(results_df, data.frame(
  Model_Name = "LASSO",
  Model_Notes = "caret and elasticnet",
  Model_Hyper = paste("Lambda:", bestLambda, ", Fraction:", bestAlpha),
  Model_RMSE = min(lasso_mod$results$RMSE),
  Model_R2 = max(lasso_mod$results$Rsquared)
))
```

Part D Regression to predict the final log sale price

```
#reset training and test for mars
# Selecting the specified columns
selected_set <- housing_reg_err_imp[, c("log_SalePrice", "LotArea",
                                        "OverallQual", "YearBuilt",
                                        "YearRemodAdd",
```

```
                                        "BsmtFinSF1", "BsmtFinSF2", "BsmtUnfSF",
                                        "X1stFlrSF", "X2ndFlrSF",
                                        "KitchenAbvGr", "TotRmsAbvGrd",
                                        "Fireplaces", "GarageYrBlt",
                                        "GarageCars", "GarageArea", "EncPorchSF"
                                        , "Exterior2nd_collapsed",
                                        "Condition1_collapsed",
                                        "Functional_collapsed",
                                        "BldgType_collapsed")]


validation_set <- selected_set [1:100, ]
training_set <- selected_set[101:1000, ]

X_train <- subset(training_set, select = -log_SalePrice)
X_train <- as.matrix(X_train)
Y_train <- training_set$log_SalePrice
X_test <- subset(validation_set, select = -log_SalePrice)
X_test <- as.matrix(X_test)
Y_test <- validation_set$log_SalePrice
```

Robust

```
#robust
library(MASS)
robust_model <- rlm(log_SalePrice ~ ., data = training_set)
summary(robust_model)
```

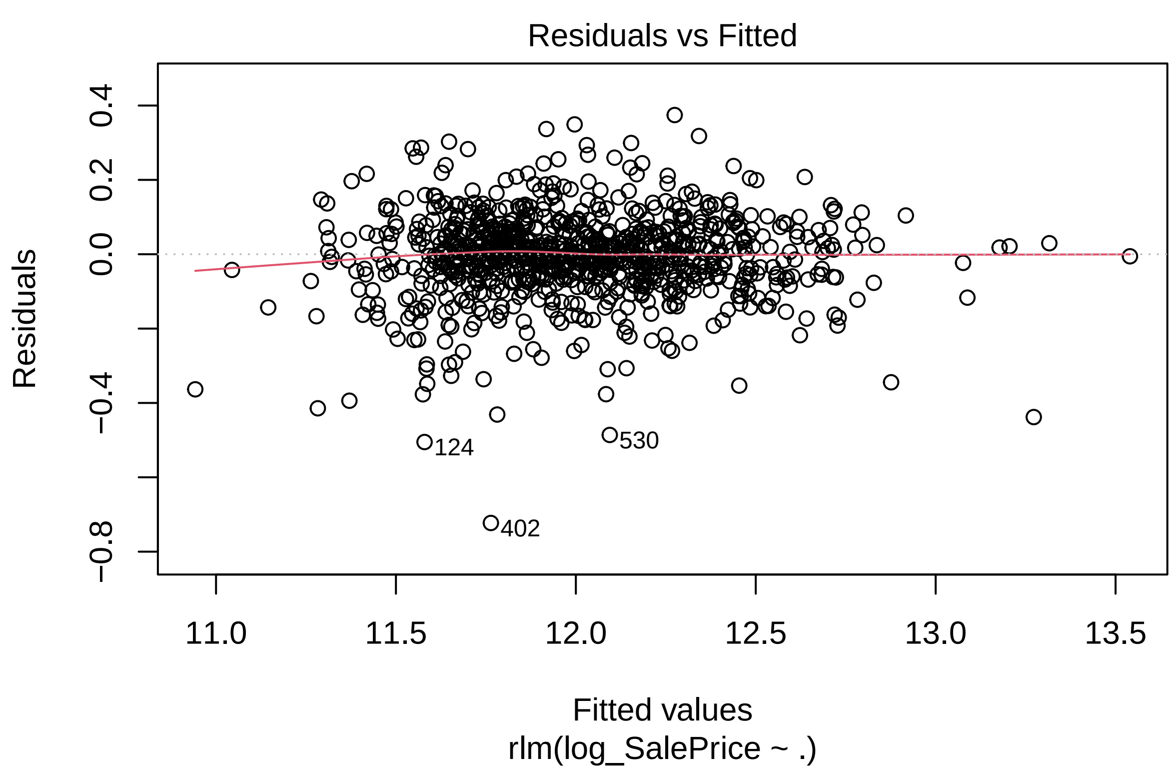
```
##
## Call: rlm(formula = log_SalePrice ~ ., data = training_set)
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.722840 -0.058176  0.001527  0.057250  0.374420
##
## Coefficients:
##                       Value   Std. Error t value
## (Intercept)            3.0803  0.4631     6.6517
## LotArea                0.0000  0.0000    10.4041
## OverallQual            0.0670  0.0043    15.7131
## YearBuilt              0.0021  0.0002     9.6677
## YearRemodAdd           0.0023  0.0002    10.3897
## BsmtFinSF1             0.0002  0.0000    13.4572
## BsmtFinSF2             0.0002  0.0000     7.0516
## BsmtUnfSF              0.0001  0.0000     6.1431
## X1stFlrSF              0.0003  0.0000    13.6436
## X2ndFlrSF              0.0003  0.0000    18.0614
## KitchenAbvGr          -0.0912  0.0196    -4.6469
## TotRmsAbvGrd           0.0017  0.0041     0.4047
## Fireplaces             0.0301  0.0064     4.6920
## GarageYrBlt           -0.0004  0.0002    -1.8775
## GarageCars             0.0219  0.0107     2.0380
## GarageArea             0.0001  0.0000     3.9380
## EncPorchSF             0.0002  0.0000     3.5148
```

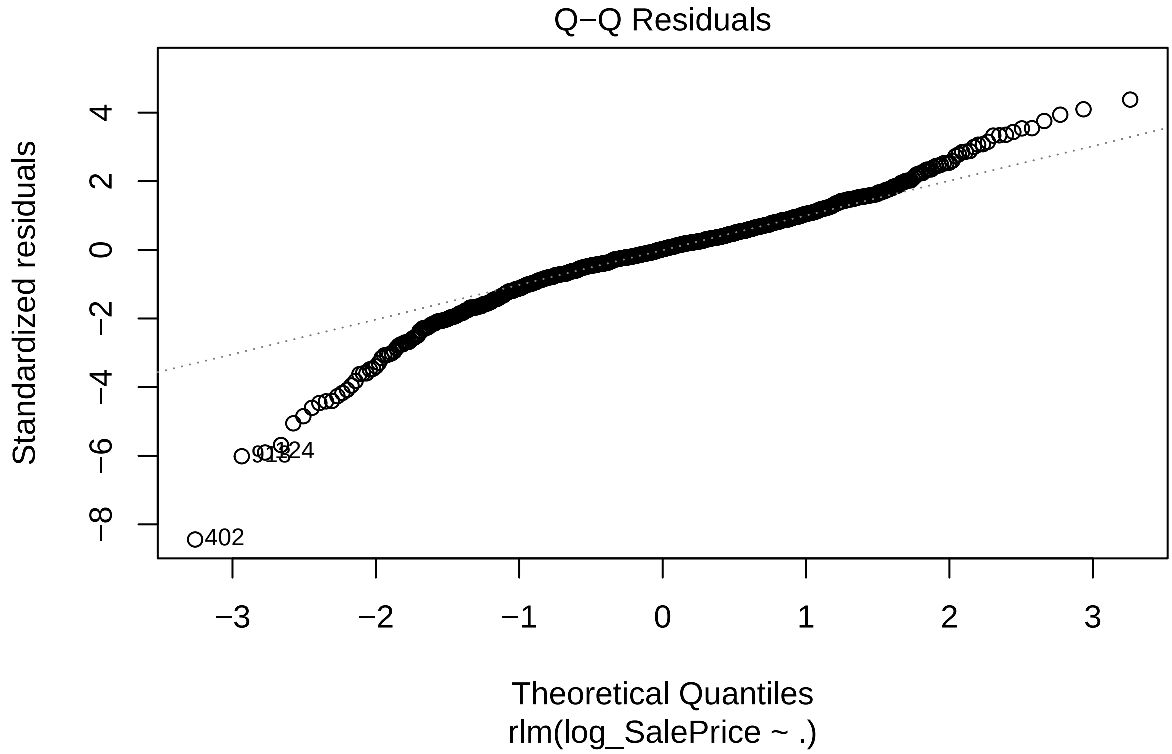
```
## Exterior2nd_collapsed  0.0027  0.0021     1.2702
## Condition1_collapsed   0.0156  0.0058     2.6843
## Functional_collapsed   0.0354  0.0051     6.9250
## BldgType_collapsed    -0.0138  0.0032    -4.3089
##
## Residual standard error: 0.08578 on 879 degrees of freedom
```

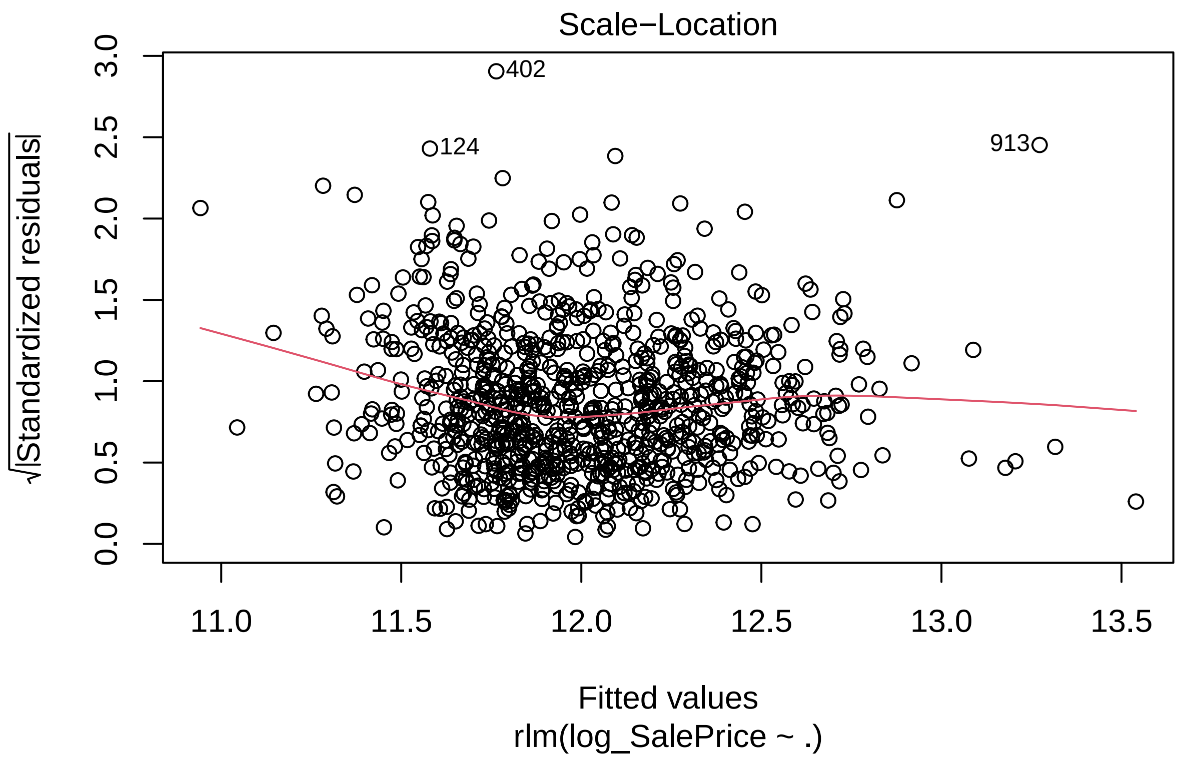
```r
coef(robust_model, s = 0.01)
```

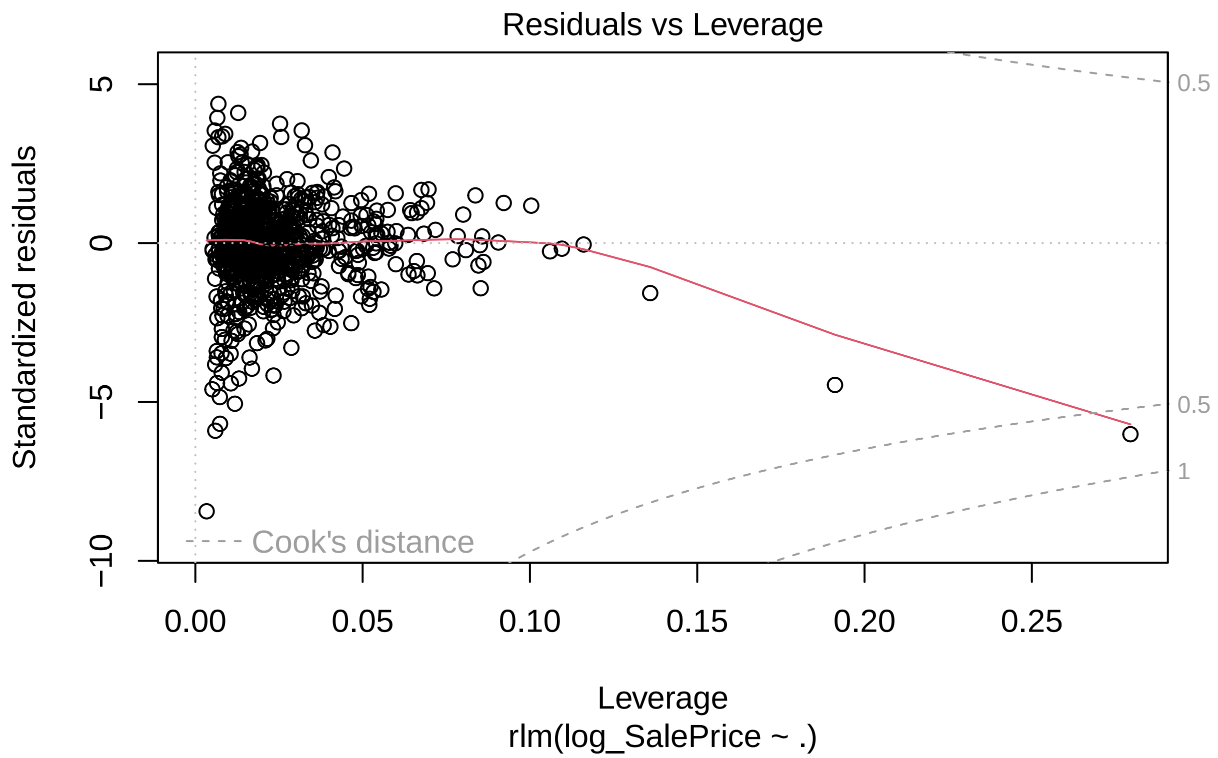
```
##           (Intercept)               LotArea           OverallQual
##          3.080300e+00          3.654149e-06          6.702495e-02
##             YearBuilt          YearRemodAdd            BsmtFinSF1
##          2.056997e-03          2.267813e-03          2.044079e-04
##            BsmtFinSF2             BsmtUnfSF             X1stFlrSF
##          1.759484e-04          8.930910e-05          2.806848e-04
##             X2ndFlrSF          KitchenAbvGr          TotRmsAbvGrd
##          2.695902e-04         -9.118808e-02          1.661767e-03
##            Fireplaces           GarageYrBlt            GarageCars
##          3.007857e-02         -4.399392e-04          2.186152e-02
##            GarageArea           EncPorchSF Exterior2nd_collapsed
##          1.461836e-04          1.563271e-04          2.696762e-03
##  Condition1_collapsed  Functional_collapsed    BldgType_collapsed
##          1.562072e-02          3.542900e-02         -1.381630e-02
```

```r
plot(robust_model)
```




Q−Q Residuals
Standardized residuals
4
2
0
−2
−4
−6
−8
913
1324
402
−3
−2
−1
0
1
2
3
Theoretical Quantiles
rlm(log_SalePrice ~ .)


Scale−Location
√|Standardized residuals|
3.0
2.5
2.0
1.5
1.0
0.5
0.0
402
124
913
11.0
11.5
12.0
12.5
13.0
13.5
Fitted values
rlm(log_SalePrice ~ .)




```r
train_control <- trainControl(method = "cv", number = 10)
robust_model <- train(log_SalePrice ~ .,
                      data = training_set,
                      method = "rlm",
                      trControl = train_control,
                      preProcess = c("center", "scale"))
```

```
## Warning in rlm.default(x, y, weights, method = method, wt.method = wt.method, :
## 'rlm' failed to converge in 20 steps
```

```r
print(robust_model)
```

```
## Robust Linear Model
##
## 900 samples
##  20 predictor
##
## Pre-processing: centered (20), scaled (20)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 811, 810, 809, 810, 810, 811, ...
## Resampling results across tuning parameters:
##
##   intercept  psi           RMSE        Rsquared   MAE
##   FALSE      psi.huber     12.0011350  0.8989691  12.00057027
```

```
##   FALSE      psi.hampel    12.0011350  0.8989691  12.00057027
##   FALSE      psi.bisquare  12.0011360  0.8989739  12.00057132
##    TRUE      psi.huber      0.1187101  0.8950750   0.08426773
##    TRUE      psi.hampel     0.1224423  0.8891507   0.08549019
##    TRUE      psi.bisquare   0.1228409  0.8883420   0.08501922
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were intercept = TRUE and psi = psi.huber.
```

```r
print(robust_model$bestTune)
```

```
##   intercept       psi
## 4      TRUE psi.huber
```

```r
robust_model$bestTune
```

```
##   intercept       psi
## 4      TRUE psi.huber
```

```r
r2 <- max(robust_model$results$Rsquared)
print(r2)
```

```
## [1] 0.8989739
```

```r
rmse <- min(robust_model$results$RMSE)
print(rmse)
```

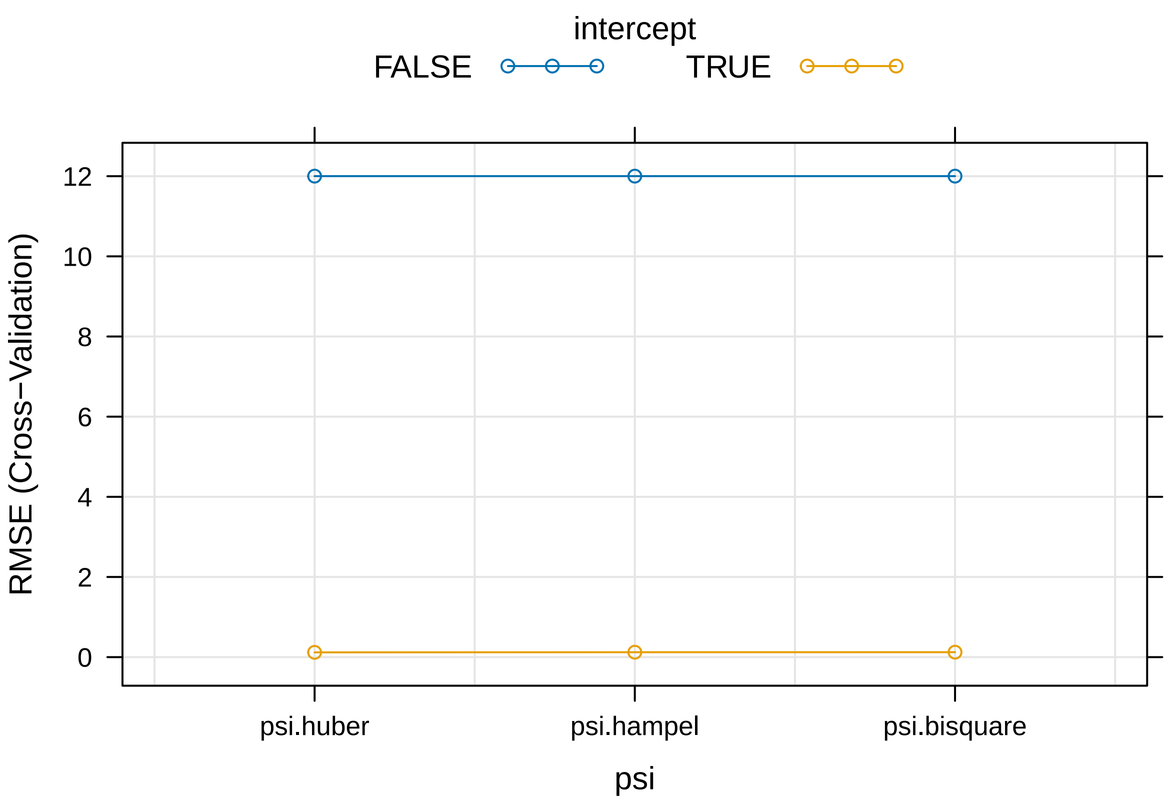
```
## [1] 0.1187101
```

```r
summary(robust_model)
```

```
##
## Call: rlm(formula = .outcome ~ ., data = dat, psi = psi)
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.722840 -0.058176  0.001527  0.057250  0.374420
##
## Coefficients:
##                        Value     Std. Error t value
## (Intercept)              12.0057    0.0033  3636.6946
## LotArea                   0.0376    0.0036    10.4041
## OverallQual               0.0886    0.0056    15.7131
## YearBuilt                 0.0596    0.0062     9.6677
## YearRemodAdd              0.0456    0.0044    10.3897
## BsmtFinSF1                0.0826    0.0061    13.4572
## BsmtFinSF2                0.0265    0.0038     7.0516
## BsmtUnfSF                 0.0369    0.0060     6.1431
## X1stFlrSF                 0.0988    0.0072    13.6436
## X2ndFlrSF                 0.1153    0.0064    18.0614
## KitchenAbvGr             -0.0188    0.0041    -4.6469
```

```
## TotRmsAbvGrd              0.0026    0.0065     0.4047
## Fireplaces                0.0193    0.0041     4.6920
## GarageYrBlt              -0.0102    0.0054    -1.8775
## GarageCars                0.0153    0.0075     2.0380
## GarageArea                0.0285    0.0072     3.9380
## EncPorchSF                0.0125    0.0036     3.5148
## Exterior2nd_collapsed     0.0043    0.0034     1.2702
## Condition1_collapsed      0.0092    0.0034     2.6843
## Functional_collapsed      0.0248    0.0036     6.9250
## BldgType_collapsed       -0.0166    0.0039    -4.3089
##
## Residual standard error: 0.08578 on 879 degrees of freedom
```

```r
plot(robust_model)
```




```r
results_df <- rbind(results_df, data.frame(
  Model_Name = "Robust",
  Model_Notes = "",
  Model_Hyper = "",
  Model_RMSE = rmse,
  Model_R2 = r2
))
```

```r
#fit OLS model on some of the features
olsFit<-lm(log_SalePrice~. ,data = training_set)  #OLS including the interaction term
summary(olsFit)
```

```
##
## Call:
## lm(formula = log_SalePrice ~ ., data = training_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.71478 -0.05571  0.00554  0.06447  0.36170
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            3.289e+00  5.349e-01   6.150 1.18e-09 ***
## LotArea                2.603e-06  4.057e-07   6.415 2.30e-10 ***
## OverallQual            7.189e-02  4.927e-03  14.590  < 2e-16 ***
## YearBuilt              2.142e-03  2.458e-04   8.713  < 2e-16 ***
## YearRemodAdd           2.302e-03  2.521e-04   9.131  < 2e-16 ***
## BsmtFinSF1             2.046e-04  1.755e-05  11.662  < 2e-16 ***
## BsmtFinSF2             1.715e-04  2.882e-05   5.951 3.85e-09 ***
## BsmtUnfSF              8.848e-05  1.679e-05   5.269 1.73e-07 ***
## X1stFlrSF              2.704e-04  2.376e-05  11.377  < 2e-16 ***
## X2ndFlrSF              2.660e-04  1.724e-05  15.425  < 2e-16 ***
## KitchenAbvGr          -8.178e-02  2.267e-02  -3.608 0.000326 ***
## TotRmsAbvGrd           3.520e-03  4.744e-03   0.742 0.458198
## Fireplaces             3.820e-02  7.405e-03   5.159 3.07e-07 ***
## GarageYrBlt           -6.662e-04  2.707e-04  -2.461 0.014036 *
## GarageCars             2.706e-02  1.239e-02   2.184 0.029216 *
## GarageArea             1.322e-04  4.288e-05   3.083 0.002117 **
## EncPorchSF             1.294e-04  5.138e-05   2.518 0.011978 *
## Exterior2nd_collapsed  1.859e-03  2.452e-03   0.758 0.448703
## Condition1_collapsed   1.288e-02  6.722e-03   1.917 0.055593 .
## Functional_collapsed   3.152e-02  5.910e-03   5.334 1.22e-07 ***
## BldgType_collapsed    -1.687e-02  3.704e-03  -4.554 6.01e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1144 on 879 degrees of freedom
## Multiple R-squared:  0.9027, Adjusted R-squared:  0.9005
## F-statistic: 407.7 on 20 and 879 DF,  p-value: < 2.2e-16
```

```r
# For OLS model
ols_pred <- predict(olsFit, validation_set)
summary(ols_pred)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.18   11.81   12.03   12.03   12.23   12.93
```

```r
ols_rmse <- sqrt(mean(olsFit$residuals^2))
ols_r2 <- summary(olsFit)$adj.r.squared
```

```r
# Add results to results_df
results_df <- rbind(results_df, data.frame(
  Model_Name = "OLS",
  Model_Notes = "Ordinary Least Squares Regression",
  Model_Hyper = "N/A",
  Model_RMSE = ols_rmse,
  Model_R2 = ols_r2
))

print(results_df)
```

```
##       Model_Name                       Model_Notes                 Model_Hyper
## RMSE         OLS Ordinary Least Squares Regression                         N/A
## RMSE1        OLS          lm + 2 way interactions                          N/A
## 1            PLS                               pls                          30
## 11         LASSO              caret and elasticnet Lambda: 0.002 , Fraction: 1
## 12        Robust
## 13           OLS Ordinary Least Squares Regression                         N/A
##       Model_RMSE  Model_R2
## RMSE   0.1095265 0.9160672
## RMSE1  0.1578111 0.8194585
## 1      0.1142139 0.9028384
## 11     0.1166649 0.8962221
## 12     0.1187101 0.8989739
## 13     0.1130573 0.9004656
```

MARS Model

```r
# Fit the MARS model
mars_model <- train(
  log_SalePrice~. ,data = training_set,
  method = "earth",
  metric = "RMSE",
  trControl = trainControl(method = "cv", number = 10)
)
```

```
## Loading required package: earth

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```r
# Print summary
summary(mars_model)
```

```
## Call: earth(x=matrix[900,20], y=c(11.58,12.03,1...), keepxy=TRUE, degree=1,
##             nprune=17)
##
##                             coefficients
## (Intercept)                   11.8661912
## h(15426-LotArea)              -0.0000124
## h(LotArea-15426)               0.0000022
## h(6-OverallQual)              -0.0801978
## h(OverallQual-6)               0.0940266
## h(1990-YearBuilt)             -0.0017938
## h(YearRemodAdd-1965)           0.0028428
## h(1410-BsmtFinSF1)            -0.0001227
## h(754-X1stFlrSF)              -0.0009392
## h(X1stFlrSF-754)               0.0003293
## h(224-X2ndFlrSF)              -0.0001921
## h(X2ndFlrSF-224)               0.0002978
## h(2-KitchenAbvGr)              0.1383056
## h(308-GarageArea)             -0.0003159
## h(GarageArea-308)              0.0001045
## h(5-Functional_collapsed)     -0.0390813
## h(Functional_collapsed-5)     -0.2649059
##
## Selected 17 of 21 terms, and 10 of 20 predictors (nprune=17)
## Termination condition: RSq changed by less than 0.001 at 21 terms
## Importance: OverallQual, X1stFlrSF, X2ndFlrSF, BsmtFinSF1, LotArea, ...
## Number of terms at each degree of interaction: 1 16 (additive model)
## GCV 0.01384401    RSS 11.56265    GRSq 0.8948277    RSq 0.9021817
```

```r
#fit MARS model on same features + 3 degrees of interactions + 5-fold CV
marsFit <- earth(log_SalePrice~. ,data = training_set,
                degree=3,nk=50,pmethod="cv",nfold=5,ncross=5)
summary(marsFit)
```

```
## Call: earth(formula=log_SalePrice~., data=training_set, pmethod="cv", degree=3,
##             nfold=5, ncross=5, nk=50)
##
##                                                                coefficients
## (Intercept)                                                      12.0540166
## h(15426-LotArea)                                                 -0.0000191
## h(LotArea-15426)                                                  0.0000019
## h(6-OverallQual)                                                 -0.0675164
## h(OverallQual-6)                                                  0.0783283
## h(YearRemodAdd-1965)                                              0.0034243
## h(1410-BsmtFinSF1)                                               -0.0001363
## h(670-BsmtFinSF2)                                                -0.0001466
## h(754-X1stFlrSF)                                                 -0.0008290
## h(X2ndFlrSF-224)                                                  0.0003850
## h(3-Condition1_collapsed)                                        -0.0474838
## h(15426-LotArea) * h(GarageArea-180)                              0.0000000
## h(15426-LotArea) * h(180-GarageArea)                             -0.0000001
## h(OverallQual-6) * h(864-X2ndFlrSF)                               0.0000557
## h(1915-YearBuilt) * h(X2ndFlrSF-224)                             -0.0000138
## h(1961-YearBuilt) * h(224-X2ndFlrSF)                             -0.0000129
## h(1410-BsmtFinSF1) * h(Functional_collapsed-5)                   -0.0001987
```

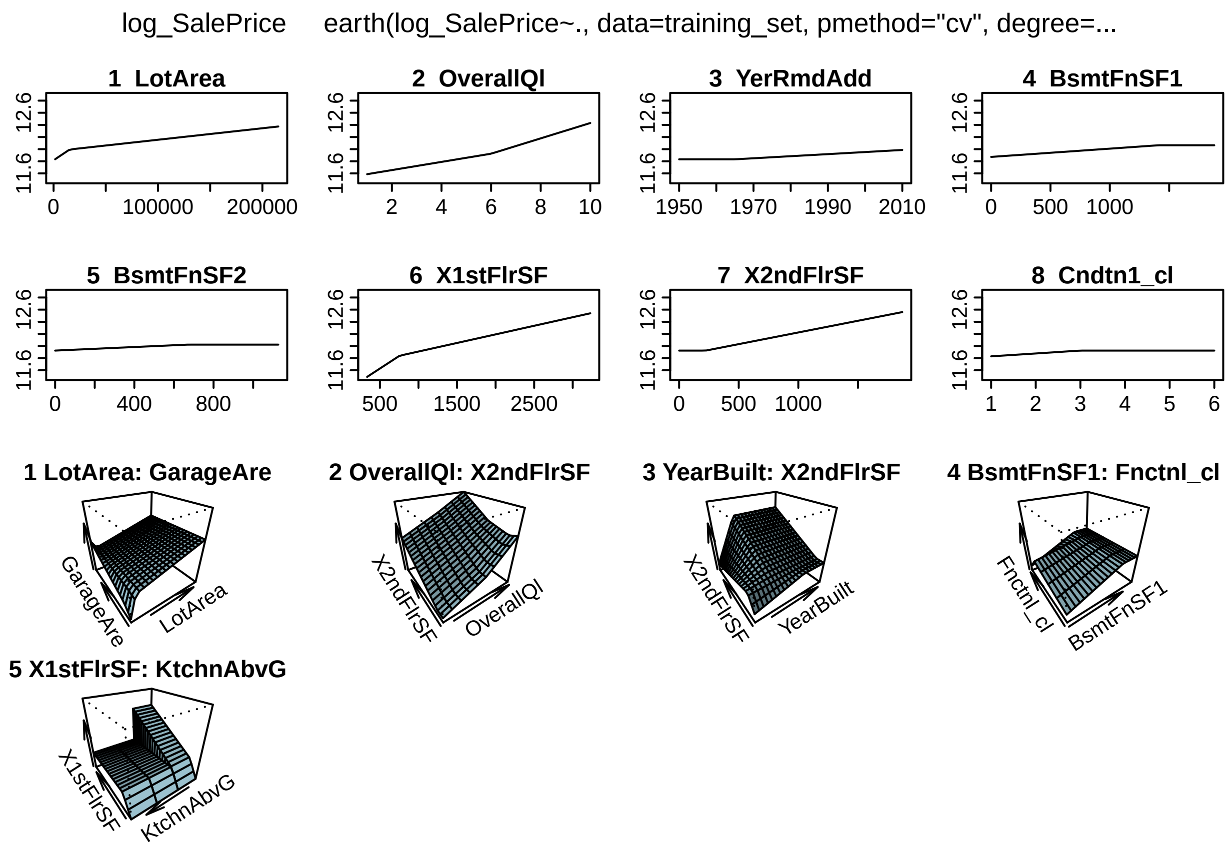
```
## h(1410-BsmtFinSF1) * h(5-Functional_collapsed)                -0.0000371
## h(X1stFlrSF-754) * h(2-KitchenAbvGr)                            0.0002830
## h(1990-YearBuilt) * h(290-BsmtFinSF1) * h(BsmtUnfSF-664)        0.0000000
## h(1990-YearBuilt) * h(1-Fireplaces) * h(GarageYrBlt-1989.45)   -0.0001696
## h(1990-YearBuilt) * h(1-Fireplaces) * h(1989.45-GarageYrBlt)   -0.0000316
##
## Selected 22 of 47 terms, and 15 of 20 predictors (pmethod="cv")
## Termination condition: RSq changed by less than 0.001 at 47 terms
## Importance: OverallQual, X1stFlrSF, X2ndFlrSF, KitchenAbvGr, BsmtFinSF1, ...
## Number of terms at each degree of interaction: 1 10 8 3
## GRSq 0.9094274  RSq 0.9196971  mean.oof.RSq 0.8794326 (sd 0.0207)
##
## pmethod="backward" would have selected:
##     31 terms 16 preds,  GRSq 0.915087  RSq 0.9286639  mean.oof.RSq 0.8188289
```

```r
summary(marsFit)
```

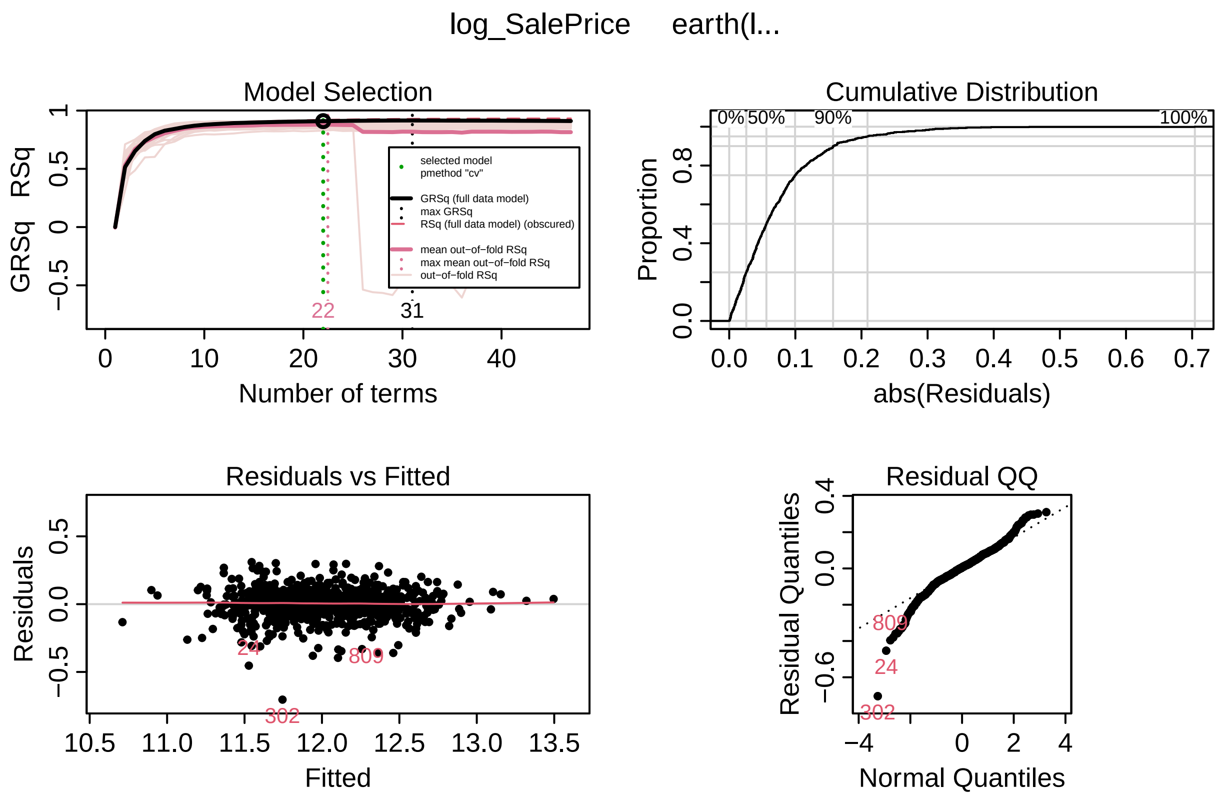
```
## Call: earth(formula=log_SalePrice~., data=training_set, pmethod="cv", degree=3,
##             nfold=5, ncross=5, nk=50)
##
##                                                               coefficients
## (Intercept)                                                     12.0540166
## h(15426-LotArea)                                                -0.0000191
## h(LotArea-15426)                                                 0.0000019
## h(6-OverallQual)                                                -0.0675164
## h(OverallQual-6)                                                 0.0783283
## h(YearRemodAdd-1965)                                             0.0034243
## h(1410-BsmtFinSF1)                                              -0.0001363
## h(670-BsmtFinSF2)                                               -0.0001466
## h(754-X1stFlrSF)                                                -0.0008290
## h(X2ndFlrSF-224)                                                 0.0003850
## h(3-Condition1_collapsed)                                       -0.0474838
## h(15426-LotArea) * h(GarageArea-180)                             0.0000000
## h(15426-LotArea) * h(180-GarageArea)                            -0.0000001
## h(OverallQual-6) * h(864-X2ndFlrSF)                              0.0000557
## h(1915-YearBuilt) * h(X2ndFlrSF-224)                            -0.0000138
## h(1961-YearBuilt) * h(224-X2ndFlrSF)                            -0.0000129
## h(1410-BsmtFinSF1) * h(Functional_collapsed-5)                  -0.0001987
## h(1410-BsmtFinSF1) * h(5-Functional_collapsed)                  -0.0000371
## h(X1stFlrSF-754) * h(2-KitchenAbvGr)                             0.0002830
## h(1990-YearBuilt) * h(290-BsmtFinSF1) * h(BsmtUnfSF-664)         0.0000000
## h(1990-YearBuilt) * h(1-Fireplaces) * h(GarageYrBlt-1989.45)    -0.0001696
## h(1990-YearBuilt) * h(1-Fireplaces) * h(1989.45-GarageYrBlt)    -0.0000316
##
## Selected 22 of 47 terms, and 15 of 20 predictors (pmethod="cv")
## Termination condition: RSq changed by less than 0.001 at 47 terms
## Importance: OverallQual, X1stFlrSF, X2ndFlrSF, KitchenAbvGr, BsmtFinSF1, ...
## Number of terms at each degree of interaction: 1 10 8 3
## GRSq 0.9094274  RSq 0.9196971  mean.oof.RSq 0.8794326 (sd 0.0207)
##
## pmethod="backward" would have selected:
##     31 terms 16 preds,  GRSq 0.915087  RSq 0.9286639  mean.oof.RSq 0.8188289
```

```r
plotmo(marsFit)
```

```
## plotmo grid:    LotArea OverallQual YearBuilt YearRemodAdd BsmtFinSF1
##                   9402           6      1971         1992        388
## BsmtFinSF2 BsmtUnfSF X1stFlrSF X2ndFlrSF KitchenAbvGr TotRmsAbvGrd Fireplaces
##          0       448      1056         0            1            6          1
## GarageYrBlt GarageCars GarageArea EncPorchSF Exterior2nd_collapsed
##        1979          2        470          0                     4
## Condition1_collapsed Functional_collapsed BldgType_collapsed
##                    3                    5                  1
```




```r
plot(marsFit)
```




```r
# For MARS model
mars_pred <- predict(marsFit, validation_set)
summary(mars_pred)
```

```
##  log_SalePrice
##  Min.   :10.97
##  1st Qu.:11.81
##  Median :12.01
##  Mean   :12.01
##  3rd Qu.:12.21
##  Max.   :12.87
```

```r
mars_rmse <-  sqrt(mean(marsFit$residuals^2))
mars_r2 <- summary(marsFit)$rsq


mars_rmse
```

```
## [1] 0.1026983
```

```r
mars_r2
```

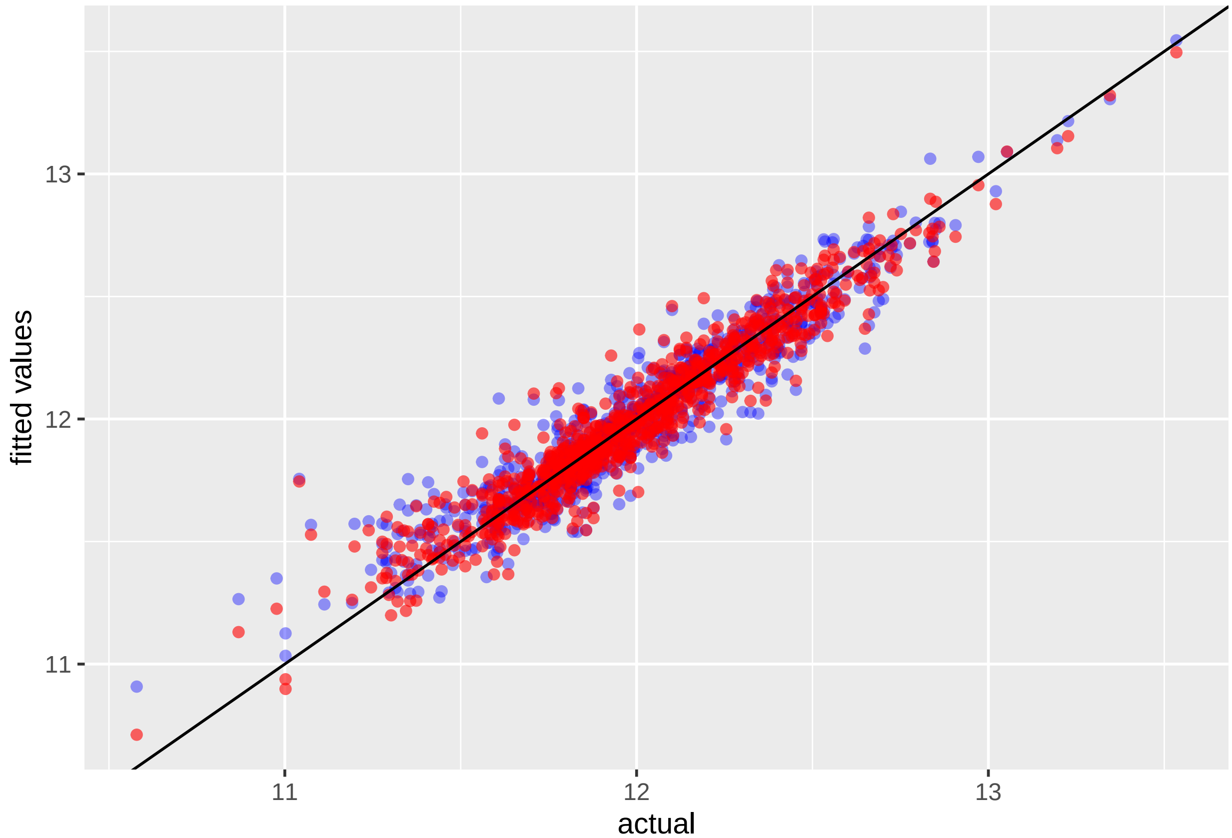
```
## [1] 0.9196971
```

```r
# Add results to results_df
results_df <- rbind(results_df, data.frame(
  Model_Name = "MARS",
  Model_Notes = "Multivariate Adaptive Regression Splines",
  Model_Hyper = "degree=3,nk=50",
  Model_RMSE = mars_rmse,
  Model_R2 = mars_r2
))


print (results_df)
```

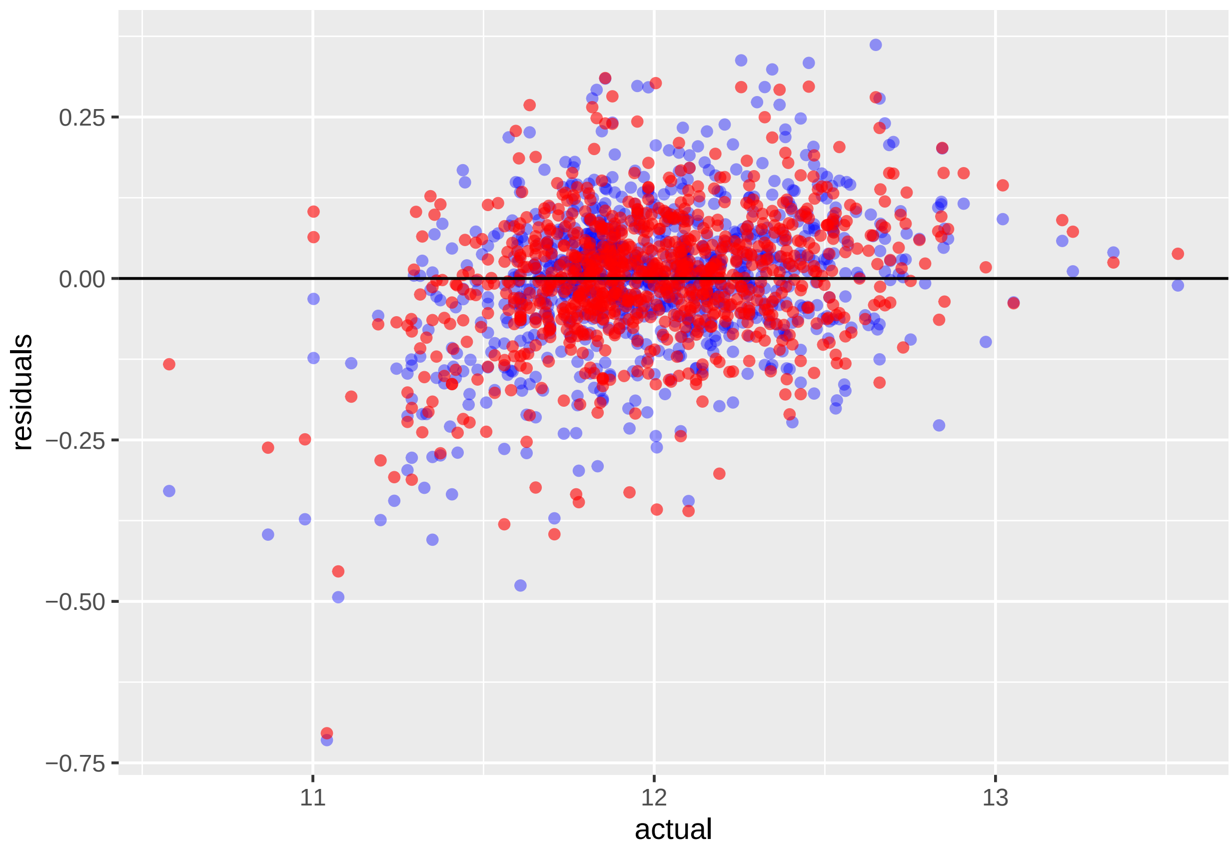
```
##       Model_Name                              Model_Notes
## RMSE         OLS         Ordinary Least Squares Regression
## RMSE1        OLS                   lm + 2 way interactions
## 1            PLS                                       pls
## 11         LASSO                      caret and elasticnet
## 12        Robust
## 13           OLS         Ordinary Least Squares Regression
## 14          MARS  Multivariate Adaptive Regression Splines
##                        Model_Hyper Model_RMSE  Model_R2
## RMSE                           N/A  0.1095265 0.9160672
## RMSE1                          N/A  0.1578111 0.8194585
## 1                               30  0.1142139 0.9028384
## 11    Lambda: 0.002 , Fraction: 1   0.1166649 0.8962221
## 12                                  0.1187101 0.8989739
## 13                             N/A  0.1130573 0.9004656
## 14                  degree=3,nk=50  0.1026983 0.9196971
```

Additional plots as per class sample compare MARS with OLS

```r
#prepare some data for plotting
plotDf <- data.frame(training_set$log_SalePrice, olsFit$fitted.values, marsFit$fitted.values,
                     olsFit$residuals, marsFit$residuals) %>%
          rename(marsPred=log_SalePrice, actual=training_set.log_SalePrice, olsPred=olsFit.fitted.values
                olsResiduals=olsFit.residuals, marsResiduals=log_SalePrice.1)

#and plot results
ggplot(data=plotDf, aes(x=actual, y=olsPred)) +geom_point(alpha=0.4,color="blue")+
  geom_point(aes(y=marsPred),color="red", alpha=.6)+
  ylab("fitted values")+geom_abline(intercept = 0, slope = 1)
```



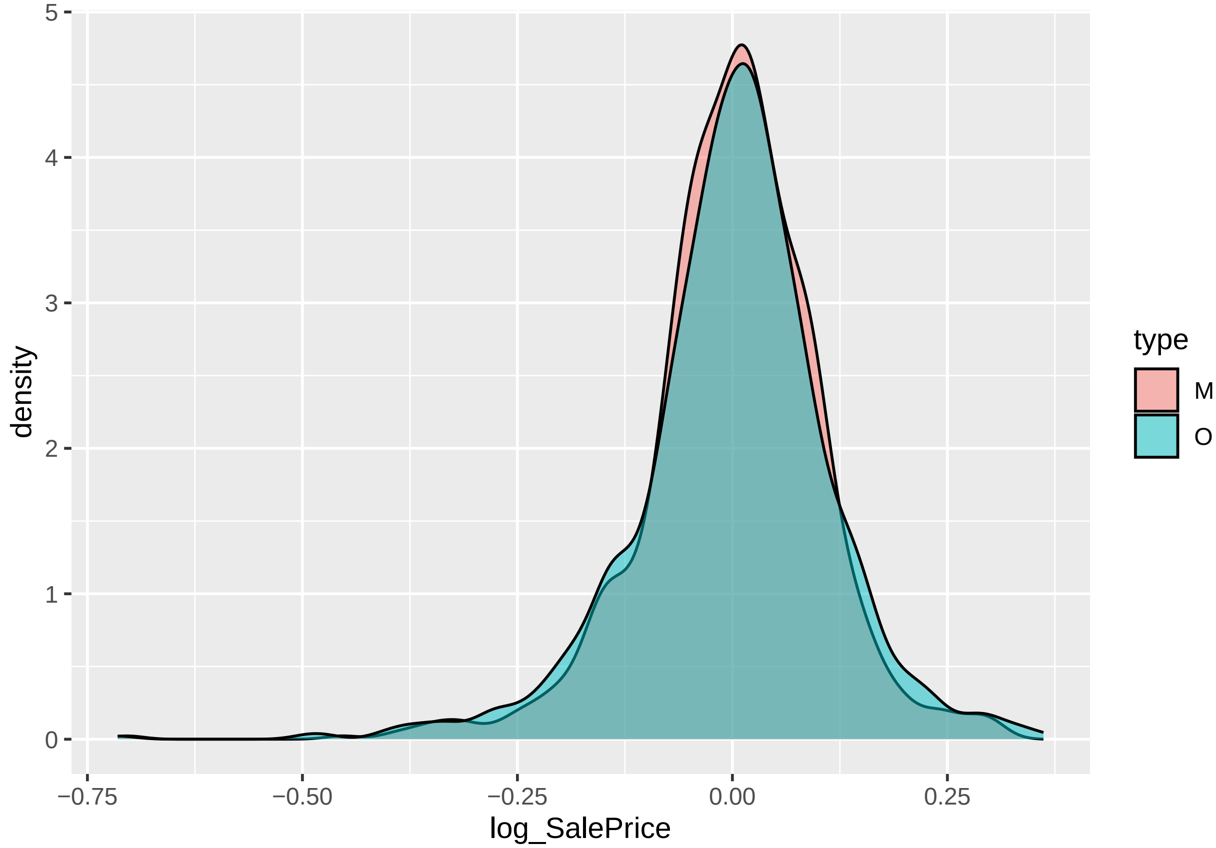

```
ggplot(data=plotDf, aes(x=actual, y=olsResiduals)) +geom_point(alpha=0.4,color="blue")+
  geom_point(aes(y=marsResiduals),color="red", alpha=.6)+
  ylab("residuals")+geom_hline(yintercept = 0)
```




```
#look at overlay of density of residuals for OLS and MARS
dat1<-data.frame(y=marsFit$residuals,type=rep("M",length(marsFit$residuals)))

str(marsFit$residuals)
```

```
##  num [1:900, 1] -0.00638 0.0384 0.01706 0.10749 -0.00894 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "log_SalePrice"
```

```
dat <- rbind(data.frame(log_SalePrice=marsFit$residuals,type=rep("M",length(marsFit$residuals))),
             data.frame(log_SalePrice=olsFit$residuals,type=rep("O",length(olsFit$residuals))))

ggplot(dat, aes(x = log_SalePrice, fill = type)) + geom_density(alpha = 0.5)
```




```r
#some models have multiple parameters to train
#e.g., elasticnet: method = "enet", parameters: fraction, lambda

fit1 = glmnet(X_train, Y_train, relax = TRUE)

?glmnet
```

```
## starting httpd help server ... done
```

```r
print(fit1)
```

```
##
## Call:  glmnet(x = X_train, y = Y_train, relax = TRUE)
## Relaxed
##
##    Df  %Dev %Dev R   Lambda
## 1   0  0.00   0.00 0.292800
## 2   1 11.08  65.26 0.266700
## 3   1 20.28  65.26 0.243100
## 4   1 27.91  65.26 0.221500
## 5   1 34.25  65.26 0.201800
## 6   1 39.52  65.26 0.183900
## 7   1 43.89  65.26 0.167500
## 8   2 48.48  70.98 0.152600
## 9   3 53.13  76.33 0.139100
```

```
## 10  4 57.09  76.60 0.126700
## 11  4 60.40  76.60 0.115500
## 12  5 63.53  78.85 0.105200
## 13  5 66.13  78.85 0.095860
## 14  7 68.36  82.78 0.087350
## 15  7 70.81  82.78 0.079590
## 16  9 73.13  85.17 0.072520
## 17  9 75.17  85.17 0.066080
## 18  9 76.87  85.17 0.060210
## 19 10 78.60  87.66 0.054860
## 20 11 80.25  88.42 0.049980
## 21 11 81.63  88.42 0.045540
## 22 11 82.78  88.42 0.041500
## 23 11 83.74  88.42 0.037810
## 24 11 84.54  88.42 0.034450
## 25 11 85.19  88.42 0.031390
## 26 11 85.74  88.42 0.028600
## 27 12 86.29  89.01 0.026060
## 28 12 86.75  89.01 0.023750
## 29 12 87.14  89.01 0.021640
## 30 13 87.47  89.16 0.019710
## 31 14 87.83  89.57 0.017960
## 32 14 88.12  89.57 0.016370
## 33 14 88.37  89.57 0.014910
## 34 15 88.59  89.77 0.013590
## 35 15 88.79  89.77 0.012380
## 36 15 88.96  89.77 0.011280
## 37 16 89.10  89.81 0.010280
## 38 18 89.25  90.20 0.009366
## 39 18 89.41  90.20 0.008534
## 40 18 89.54  90.20 0.007776
## 41 18 89.66  90.20 0.007085
## 42 18 89.75  90.20 0.006456
## 43 18 89.82  90.20 0.005882
## 44 18 89.89  90.20 0.005360
## 45 18 89.94  90.20 0.004883
## 46 18 89.98  90.20 0.004450
## 47 18 90.02  90.20 0.004054
## 48 18 90.05  90.20 0.003694
## 49 18 90.07  90.20 0.003366
## 50 19 90.10  90.26 0.003067
## 51 19 90.13  90.26 0.002794
## 52 19 90.15  90.26 0.002546
## 53 19 90.17  90.26 0.002320
## 54 19 90.19  90.26 0.002114
## 55 20 90.20  90.27 0.001926
## 56 20 90.21  90.27 0.001755
## 57 20 90.22  90.27 0.001599
## 58 20 90.23  90.27 0.001457
## 59 20 90.24  90.27 0.001328
## 60 20 90.24  90.27 0.001210
## 61 20 90.25  90.27 0.001102
## 62 20 90.25  90.27 0.001004
## 63 20 90.25  90.27 0.000915
```

```
## 64 20 90.25  90.27 0.000834
## 65 20 90.26  90.27 0.000760
## 66 20 90.26  90.27 0.000692
## 67 20 90.26  90.27 0.000631
## 68 20 90.26  90.27 0.000575
## 69 20 90.26  90.27 0.000524
## 70 20 90.26  90.27 0.000477
## 71 20 90.26  90.27 0.000435
```

```r
coef(fit1, s = 0.01)
```

```
## 21 x 1 sparse Matrix of class "dgCMatrix"
##                                 s1
## (Intercept)           4.148043e+00
## LotArea               2.317231e-06
## OverallQual           8.830360e-02
## YearBuilt             1.534693e-03
## YearRemodAdd          1.839638e-03
## BsmtFinSF1            1.209058e-04
## BsmtFinSF2            3.497211e-05
## BsmtUnfSF             9.291547e-07
## X1stFlrSF             2.889327e-04
## X2ndFlrSF             2.149490e-04
## KitchenAbvGr         -5.258271e-02
## TotRmsAbvGrd          4.675495e-03
## Fireplaces            3.852429e-02
## GarageYrBlt           .
## GarageCars            2.863413e-02
## GarageArea            1.225853e-04
## EncPorchSF            2.646649e-06
## Exterior2nd_collapsed .
## Condition1_collapsed  4.866596e-04
## Functional_collapsed  1.782380e-02
## BldgType_collapsed   -1.261427e-02
```

```r
predict(fit1, X_test , s = c(0.01, 0.005))
```

```
##          s1       s2
## 1   11.88817 11.89593
## 2   12.02054 12.04071
## 3   11.81979 11.81163
## 4   11.89660 11.90470
## 5   11.92904 11.92131
## 6   12.15399 12.15843
## 7   11.77280 11.77849
## 8   12.35236 12.39360
## 9   11.98825 12.00242
## 10  12.12140 12.13717
## 11  12.02606 12.05311
## 12  12.14956 12.14198
## 13  11.95497 11.95346
## 14  12.05606 12.04942
## 15  11.74589 11.74724
```

```
## 16  11.64625 11.63463
## 17  12.01130 12.02648
## 18  11.72556 11.74232
## 19  11.73392 11.74529
## 20  12.12952 12.12434
## 21  12.80731 12.83773
## 22  12.20103 12.23184
## 23  11.66772 11.67200
## 24  12.13566 12.12727
## 25  12.29308 12.29227
## 26  12.14028 12.13225
## 27  11.49723 11.49324
## 28  12.15714 12.16144
## 29  11.56566 11.57503
## 30  12.59401 12.60281
## 31  12.28449 12.29484
## 32  12.33695 12.36966
## 33  12.55387 12.56828
## 34  11.80122 11.79435
## 35  12.12080 12.12103
## 36  11.71850 11.70540
## 37  11.98667 11.97695
## 38  12.39119 12.37912
## 39  11.88475 11.87253
## 40  11.67816 11.67052
## 41  12.22841 12.22356
## 42  11.36480 11.36182
## 43  12.22501 12.24319
## 44  12.05529 12.06614
## 45  12.09066 12.09569
## 46  12.12983 12.11982
## 47  11.65648 11.66760
## 48  11.39945 11.37794
## 49  12.06697 12.06106
## 50  11.96793 11.97745
## 51  12.84343 12.88830
## 52  12.18973 12.21622
## 53  12.00298 12.00180
## 54  12.27968 12.28921
## 55  12.38780 12.39828
## 56  11.30082 11.25265
## 57  11.91897 11.92638
## 58  12.01545 12.01395
## 59  12.28007 12.28980
## 60  11.54477 11.54246
## 61  11.68291 11.67439
## 62  12.46130 12.48900
## 63  12.41898 12.44813
## 64  12.71006 12.71240
## 65  11.94854 11.95450
## 66  11.66167 11.60933
## 67  12.41893 12.42264
## 68  12.00377 12.01111
## 69  11.79170 11.79072
```

```
## 70  12.47997 12.52513
## 71  12.02667 12.02432
## 72  12.25525 12.26874
## 73  11.91702 11.90931
## 74  11.70764 11.70802
## 75  12.17413 12.19699
## 76  12.19501 12.17545
## 77  12.31256 12.30924
## 78  11.94394 11.93837
## 79  11.97917 11.95407
## 80  11.49973 11.48910
## 81  11.85181 11.84507
## 82  11.35442 11.34646
## 83  11.92753 11.93421
## 84  12.11964 12.13575
## 85  11.91177 11.89780
## 86  12.36463 12.38064
## 87  12.21054 12.20371
## 88  11.91594 11.90193
## 89  11.94763 11.93202
## 90  12.44869 12.46572
## 91  12.06187 12.04829
## 92  11.72902 11.72553
## 93  12.13072 12.11288
## 94  12.20167 12.21262
## 95  11.74393 11.73885
## 96  11.84781 11.83140
## 97  12.07828 12.06578
## 98  12.29712 12.29307
## 99  12.02203 12.02620
## 100 11.70380 11.72219
```

```r
plot(fit1, plotType="level")
```

```
## Warning in plot.window(...): "plotType" is not a graphical parameter
```

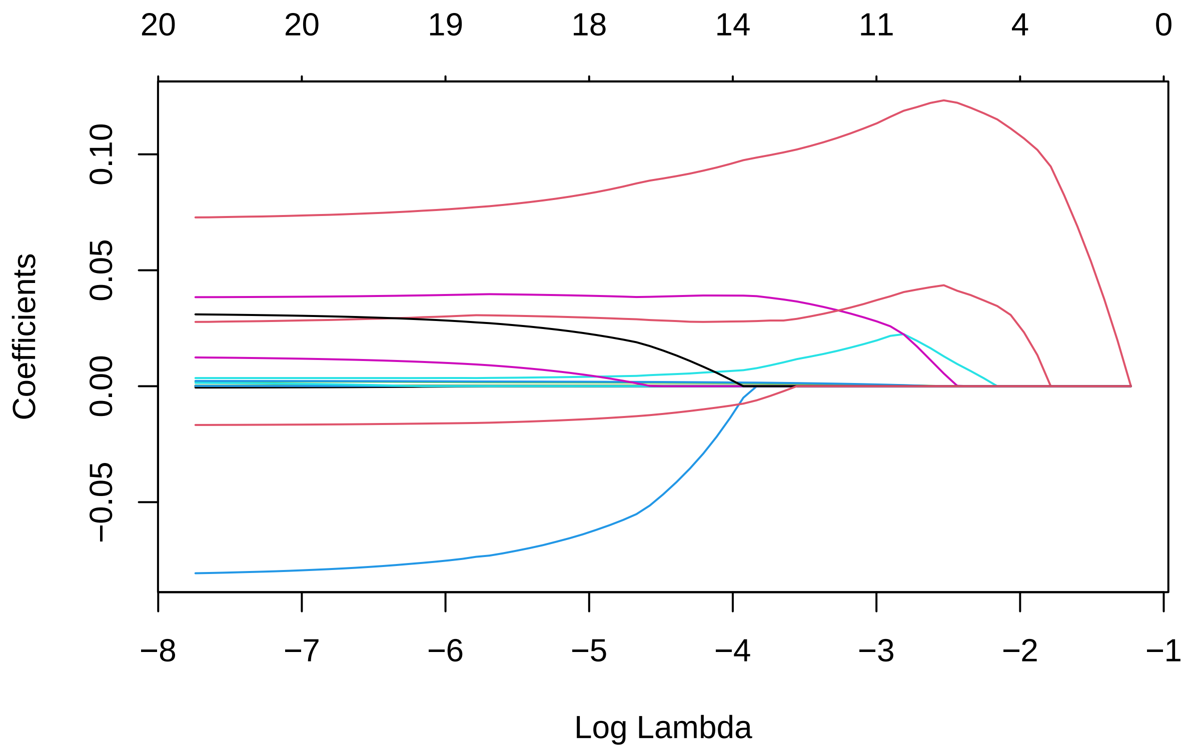
```
## Warning in plot.xy(xy, type, ...): "plotType" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "plotType" is not
## a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "plotType" is not
## a graphical parameter
```

```
## Warning in box(...): "plotType" is not a graphical parameter
```

```
## Warning in title(...): "plotType" is not a graphical parameter
```



Elasticnet

```r
enetGrid <- expand.grid(lambda=seq(0.0004,0.3,length=5),
                        fraction=seq(0.1,1,length=5))
fitControl <- trainControl(method="cv", number=10)
enet_model <- train(log_SalePrice~. ,data = training_set,
             method="enet",
             trControl=fitControl,
             tuneGrid=enetGrid)

# Extracting best hyperparameters
best_lambda <- enet_model$bestTune$lambda
best_fraction <- enet_model$bestTune$fraction

# Showing Results
print(paste("Best Lambda: ", best_lambda))
```

```
## [1] "Best Lambda:  4e-04"
```

```r
print(paste("Best Fraction: ", best_fraction))
```

```
## [1] "Best Fraction:  1"
```

```r
r2 <- max(enet_model$results$Rsquared)
print(paste("R-squared: ", r2))
```

```
## [1] "R-squared:  0.897060862402794"
```

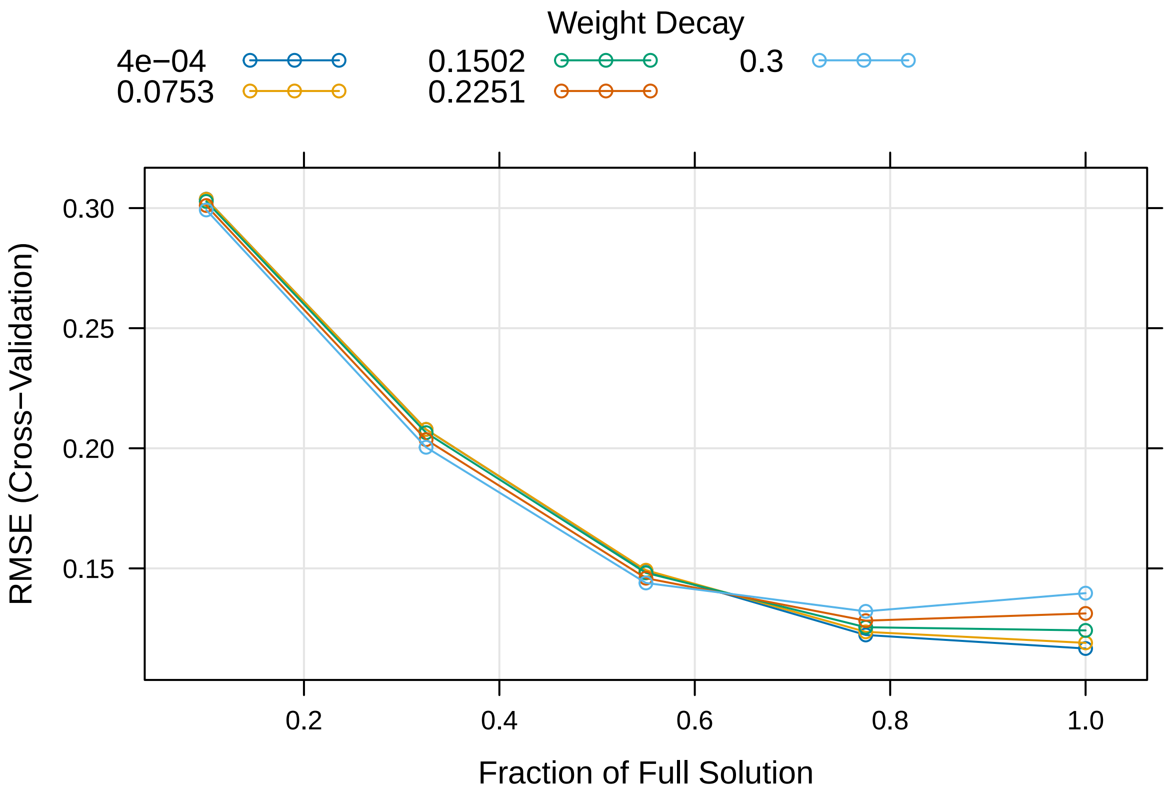
```r
rmse <- min(enet_model$results$RMSE)
print(paste("RMSE: ", rmse))
```

```
## [1] "RMSE:  0.116647900823067"
```

```r
summary(enet_model)
```

```
##             Length Class      Mode
## call          4    -none-     call
## actions      21    -none-     list
## allset       20    -none-     numeric
## beta.pure   420    -none-     numeric
## vn           20    -none-     character
## mu            1    -none-     numeric
## normx        20    -none-     numeric
## meanx        20    -none-     numeric
## lambda        1    -none-     numeric
## L1norm       21    -none-     numeric
## penalty      21    -none-     numeric
## df           21    -none-     numeric
## Cp           21    -none-     numeric
## sigma2        1    -none-     numeric
## xNames       20    -none-     character
## problemType   1    -none-     character
## tuneValue     2    data.frame list
## obsLevels     1    -none-     logical
## param         0    -none-     list
```

```r
plot(enet_model)
```




```
# Add results to the dataframe
results_df <- rbind(results_df, data.frame(
  Model_Name = "Elastic Net",
  Model_Notes = "Elastic Net Regression",
  Model_Hyper = paste("Lambda:", best_lambda, ", Fraction:", best_fraction),
  Model_RMSE = rmse,
  Model_R2 = r2
))

kable(results_df)
```

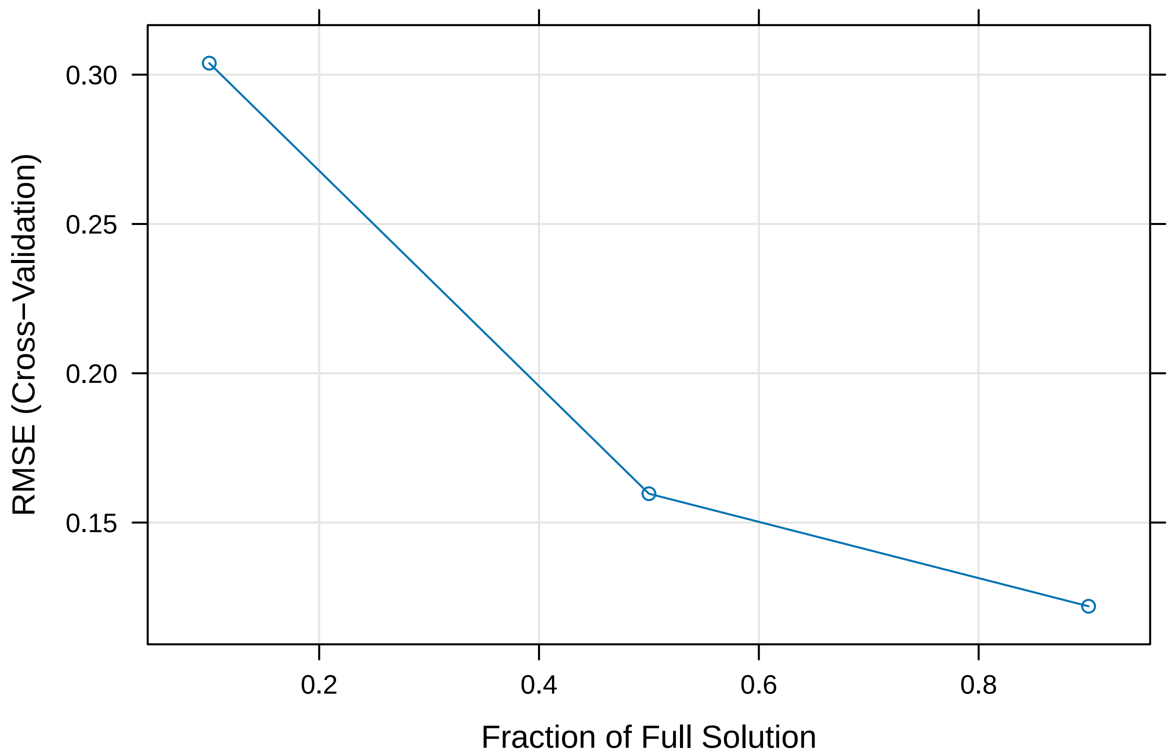
	Model_Name	Model_Notes	Model_Hyper	Model_RMSE	Model_R2
RMSE	OLS	Ordinary Least Squares Regression	N/A	0.1095265	0.9160672
RMSE1	OLS	lm + 2 way interactions	N/A	0.1578111	0.8194585
1	PLS	pls	30	0.1142139	0.9028384
11	LASSO	caret and elasticnet	Lambda: 0.002 , Fraction: 1	0.1166649	0.8962221
12	Robust			0.1187101	0.8989739
13	OLS	Ordinary Least Squares Regression	N/A	0.1130573	0.9004656
14	MARS	Multivariate Adaptive Regression Splines	degree=3,nk=50	0.1026983	0.9196971
15	Elastic Net	Elastic Net Regression	Lambda: 4e-04 , Fraction: 1	0.1166479	0.8970609

LASSO for this training set

```r
set.seed(12345)
fit_control <-trainControl(method="cv", number=5)

lasso_mod1 <- train(x=X_train, y=Y_train,
                    method='lasso',
                    trControl= fit_control
                    )

lasso_mod1$bestTune
```

```
##   fraction
## 3      0.9
```

```r
plot (lasso_mod1)
```




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

```
## [1] 0.1166649
```

```r
#lasso_mod$results$RMSE

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

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

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

```
## Lasso uses 33 variables in its model, and did not select 6 variables.
```

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

```
##        1        2        3        4        5        6
## 143747.6 177230.5 134219.0 147134.5 152778.6 190728.7
```

Plot different alpha and lambda values

```r
plot(lasso_mod)
```