Predicting the House Sales in King County, USA

Lakshmimanaswitha Chimakurthi

1 Abstract

House Sales in King County, USA This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015. The idea of our approach is to predict house prices based relevant predictors. Also, since the number of observation is too high, we randomly choose three neighbourhoods from King County (as categorical variables) to reduce the row numbers in the dataset.

2 Introduction to the data set

2.1 Read data

```
# read the data
set.seed(678)
house_sales <- read.table("kc_house_data.csv", sep=",", header=TRUE)
```

This is a house sales data in Washington state, let's do some preliminary data exploration on this data set.

```
dim(house_sales)
```

```
## [1] 21613    21
```

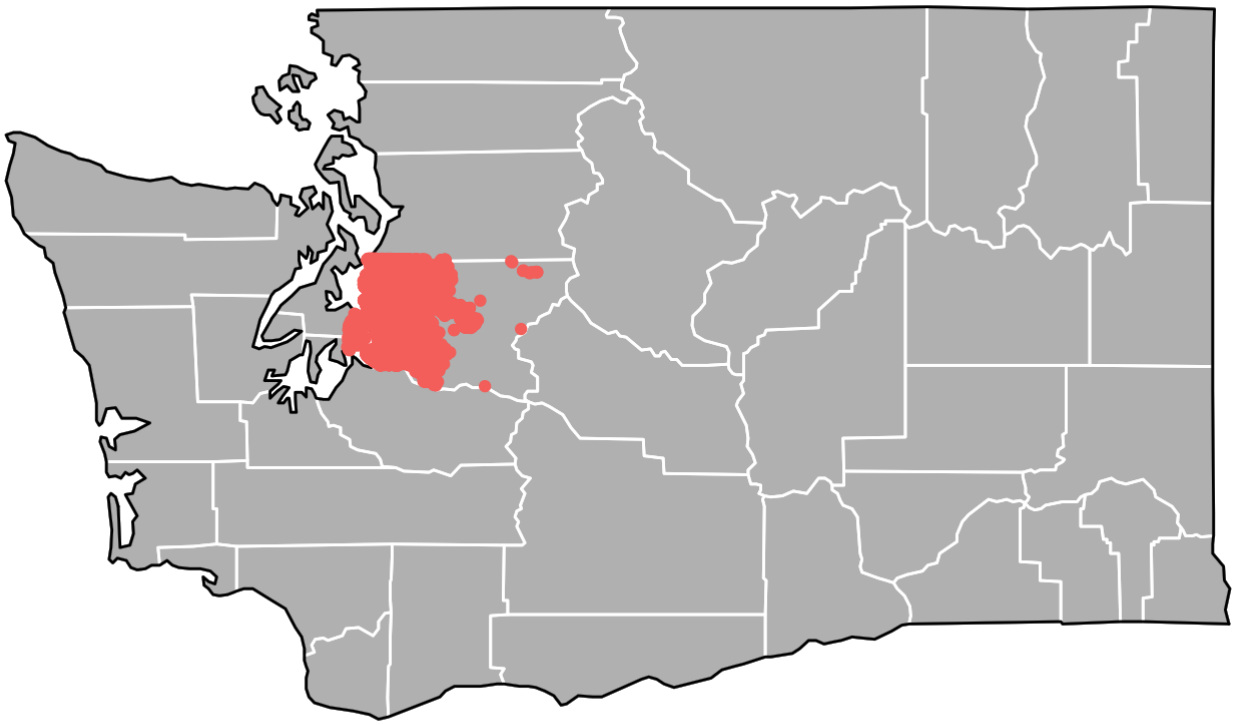
```
head(house_sales)
```

```
##           id            date   price bedrooms bathrooms sqft_living
## 1 7129300520 20141013T000000  221900        3      1.00        1180
## 2 6414100192 20141209T000000  538000        3      2.25        2570
## 3 5631500400 20150225T000000  180000        2      1.00         770
## 4 2487200875 20141209T000000  604000        4      3.00        1960
## 5 1954400510 20150218T000000  510000        3      2.00        1680
## 6 7237550310 20140512T000000 1225000        4      4.50        5420
##   sqft_lot floors waterfront view condition grade sqft_above sqft_basement
## 1     5650      1          0    0         3     7       1180             0
## 2     7242      2          0    0         3     7       2170           400
## 3    10000      1          0    0         3     6        770             0
## 4     5000      1          0    0         5     7       1050           910
## 5     8080      1          0    0         3     8       1680             0
## 6   101930      1          0    0         3    11       3890          1530
##   yr_built yr_renovated zipcode     lat     long sqft_living15 sqft_lot15
## 1     1955            0   98178 47.5112 -122.257          1340       5650
## 2     1951         1991   98125 47.7210 -122.319          1690       7639
## 3     1933            0   98028 47.7379 -122.233          2720       8062
## 4     1965            0   98136 47.5208 -122.393          1360       5000
## 5     1987            0   98074 47.6168 -122.045          1800       7503
## 6     2001            0   98053 47.6561 -122.005          4760     101930
```

2.2 Visualize data

This data set contains the prices of houses, the locations of houses and other parameters related with houses. First, let's visualize this data set on a map.

```
states <- map_data("state")
wa_df <- subset(states, region == "washington")
counties <- map_data("county")
wa_county <- subset(counties, region == "washington")
wa_base <- ggplot(data = wa_df, mapping = aes(x = long, y = lat, group = group)) +
  coord_fixed(1.3) +
  geom_polygon(color = "black", fill = "gray")

wa_city<-wa_base + theme_nothing() +
  geom_polygon(data = wa_county, fill = NA, color = "white") +
  geom_polygon(color = "black", fill = NA)  # get the state border back on top

 wa_city +
  geom_point(aes(x=long,y=lat,group=zipcode,color="red"),data=house_sales)
```



So

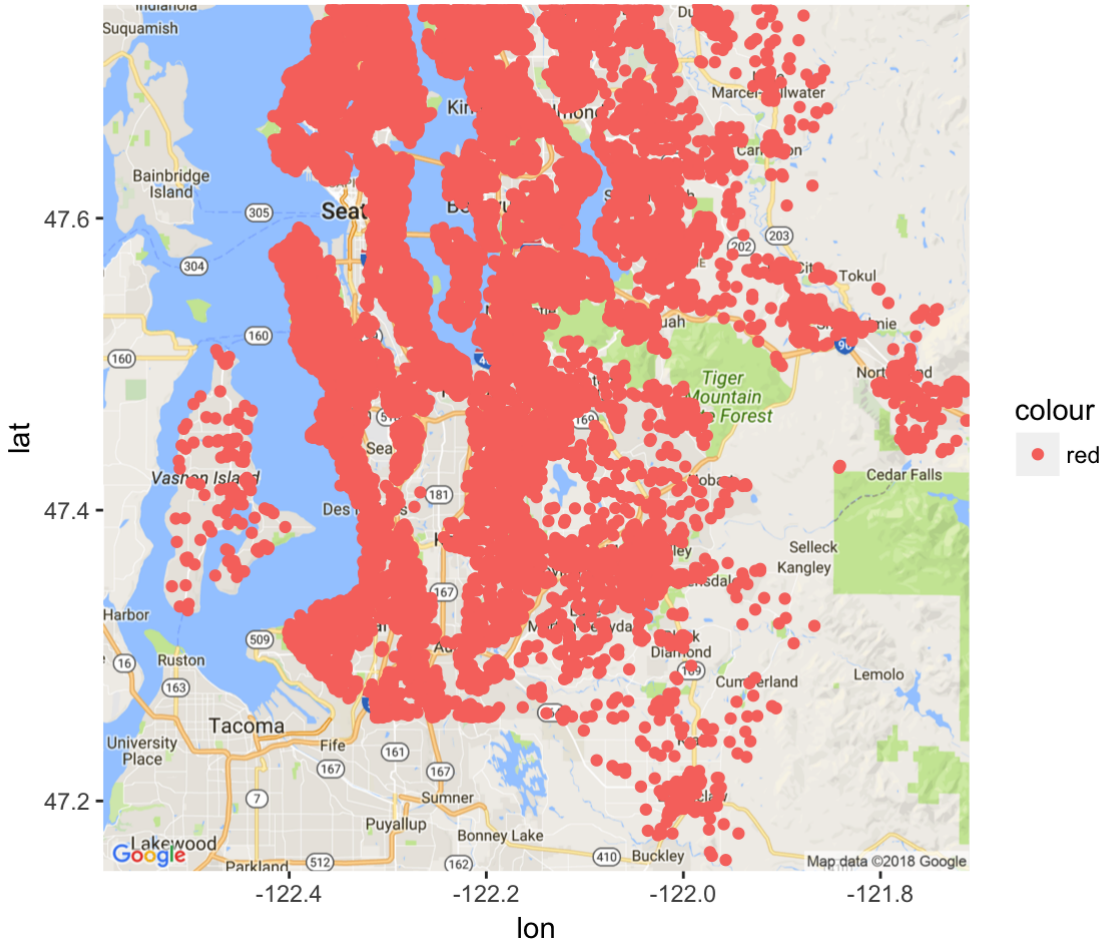
the sampling was not across the whole Washington state, while it focused on the King county around Seattle city.

```
map = get_map(location = c(-122.15, 47.45),
              zoom = 10, source = "google", maptype="roadmap")
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=47.45,-122.15&zoom=10&
size=640x640&scale=2&maptype=roadmap&language=en-EN
```

```
ggmap(map)+
  geom_point(aes(x=long,y=lat,group=zipcode,color="red"),data=house_sales)
```

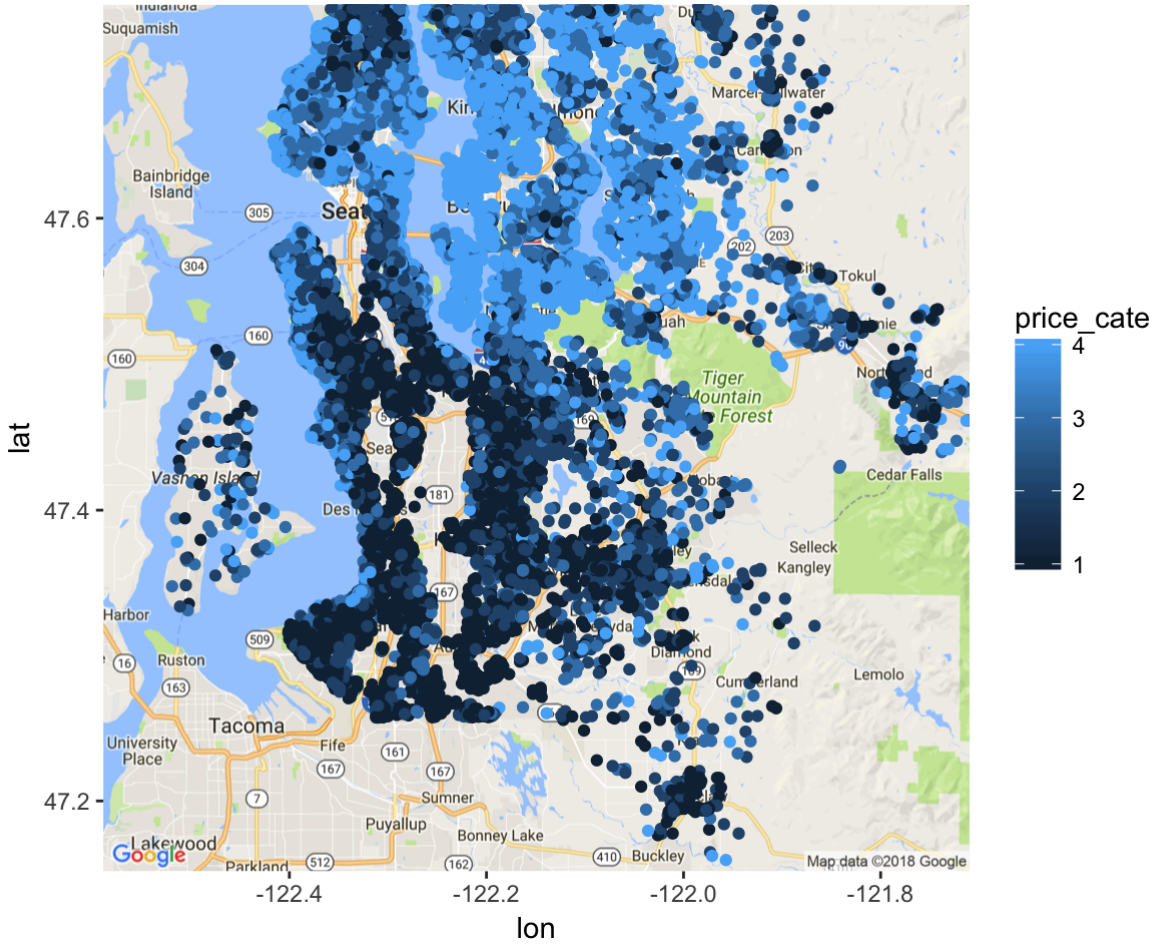
```
## Warning: Removed 1246 rows containing missing values (geom_point).
```



visualize the prices on the map.

```
for (i in 1:nrow(house_sales))
{
  if (house_sales$price[i]<321950)
  {house_sales$price_cate[i]=1}
  if (house_sales$price[i]>321950&house_sales$price[i]<450000)
  {house_sales$price_cate[i]=2}
  if (house_sales$price[i]>450000&house_sales$price[i]<645000)
  {house_sales$price_cate[i]=3}
  if (645000<house_sales$price[i])
  {house_sales$price_cate[i]=4}

}
ggmap(map)+
  geom_point(aes(x=long,y=lat,group=price_cate,color=price_cate),data=house_sales)
```

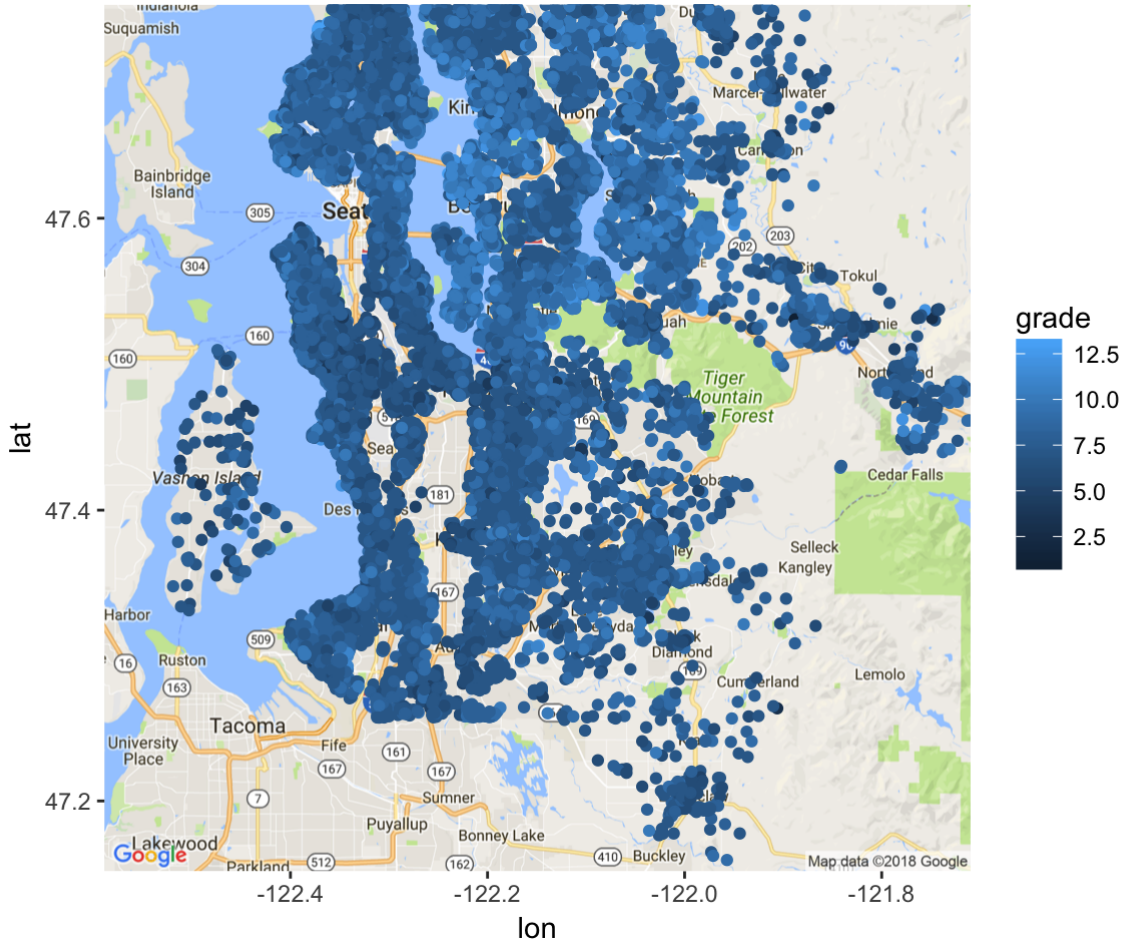
```
## Warning: Removed 1246 rows containing missing values (geom_point).
```



visualize the grade and condition:

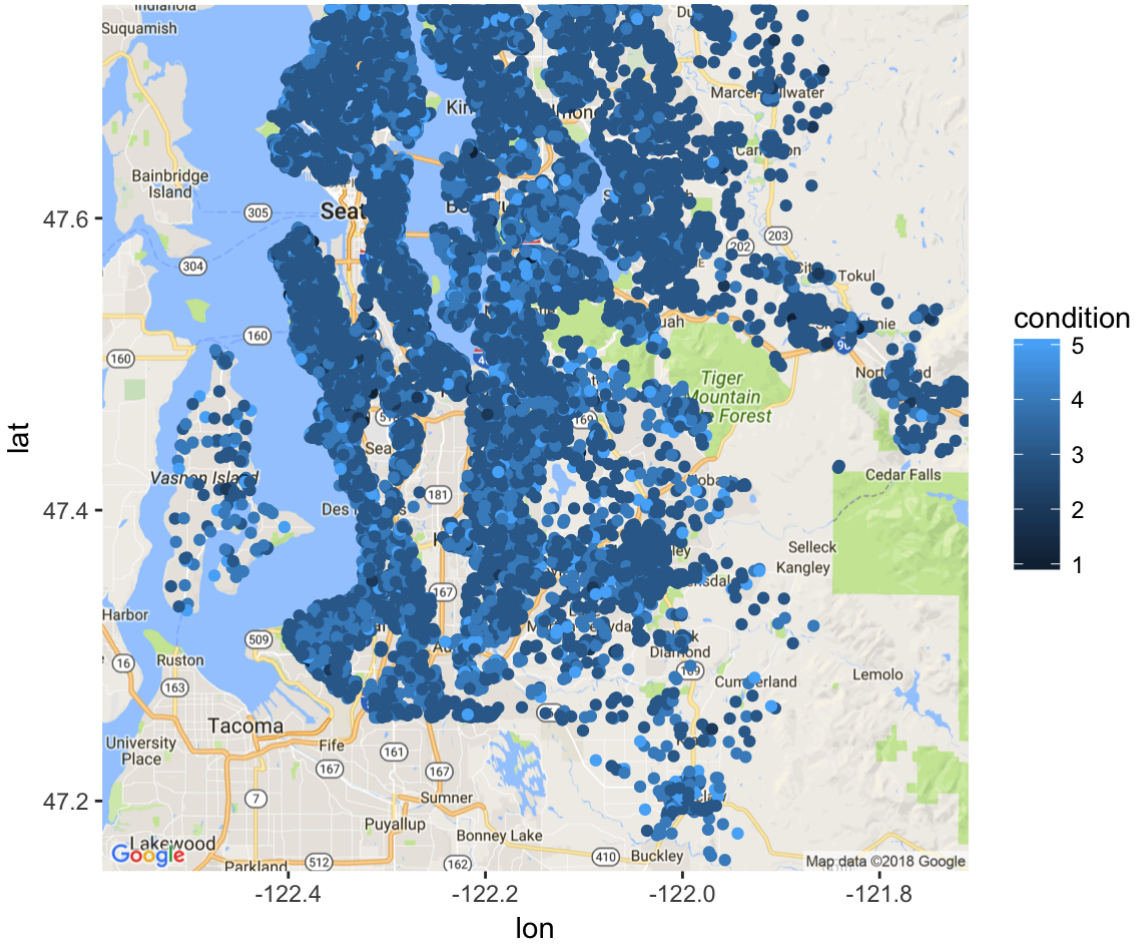
```
ggmap(map)+
  geom_point(aes(x=long,y=lat,group=grade,color=grade),data=house_sales)
```

```
## Warning: Removed 1246 rows containing missing values (geom_point).
```




```
ggmap(map)+
  geom_point(aes(x=long,y=lat,group=condition,color=condition),data=house_sales)
```

```
## Warning: Removed 1246 rows containing missing values (geom_point).
```




```
drops <- c("id","lat", "long", "sqft_living15",
           "sqft_lot15", "yr_renovated",
           "sqft_basement", "date", "waterfront",
           "sqft_above", "sqft_lot", "price_cate") # To visualize bettwe we omit some pr
edictors for the sake of simpilicity

house_sales <- house_sales[ , !(names(house_sales) %in% drops)]
house_sales <- house_sales[house_sales[, "zipcode"] == c(98001, 98039, 98199),] # To red
uce the volume
```

3 Preprocessing

3.1 Select the training set

```
#Training$waterfront <- as.numeric(Training$waterfront)
# get the training & validation set set
ratio = sample(1:nrow(house_sales), size = 0.7*nrow(house_sales)) # Ramdomly choose obse
rvation from the dataframe
Training = house_sales[ratio,] #Test dataset 30% of total
Validation = house_sales[-ratio,] #Train dataset 70% of total

head(Training)
```

```
##         price bedrooms bathrooms sqft_living floors view condition grade
## 4063   212500        3      1.00         920      1    0         4     7
## 14437  234000        4      2.50        1820      1    0         3     7
## 17565  600000        2      1.75        1560      1    0         5     7
## 10399  285000        3      2.50        1590      2    0         3     7
## 14457 1115000        4      2.50        3690      1    3         4    10
## 16009  215000        3      2.00        1760      1    0         5     7
##       yr_built zipcode
## 4063      1977   98001
## 14437     1987   98001
## 17565     1946   98199
## 10399     1985   98001
## 14457     1951   98199
## 16009     1947   98001
```

3.2 Preliminary data exploration

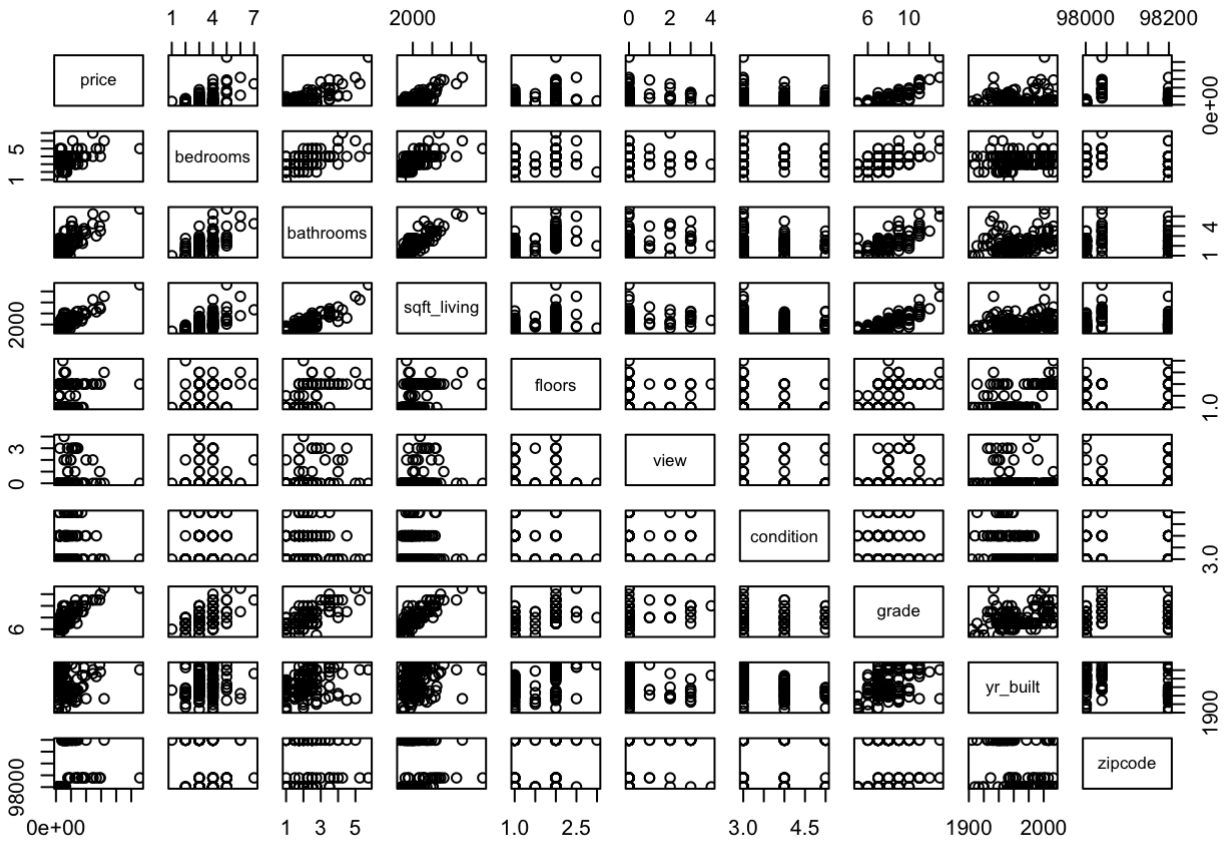
One variable summary statistics

```
summary(Training)
```

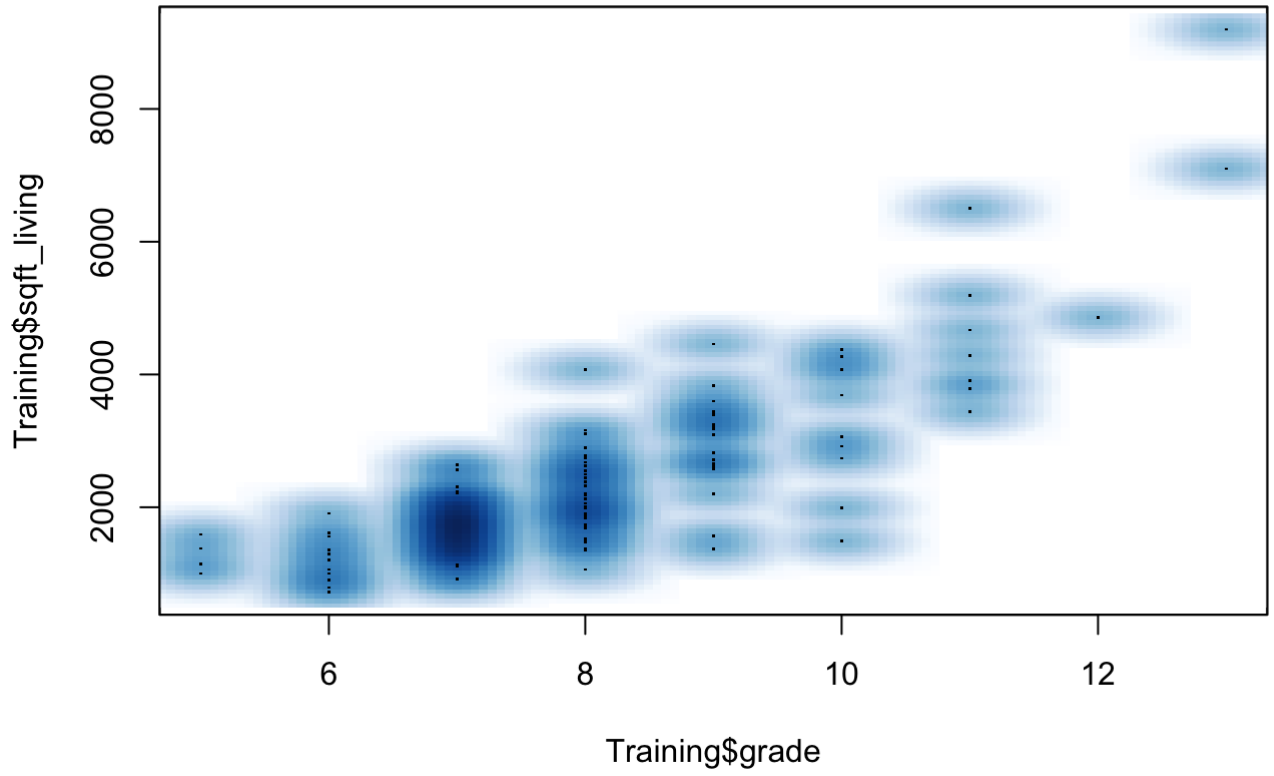
```
##      price            bedrooms       bathrooms      sqft_living
##  Min.   : 100000   Min.   :1.000   Min.   :1.000   Min.   : 720
##  1st Qu.: 250000   1st Qu.:3.000   1st Qu.:1.750   1st Qu.:1490
##  Median : 485000   Median :3.000   Median :2.000   Median :1925
##  Mean   : 683044   Mean   :3.436   Mean   :2.194   Mean   :2237
##  3rd Qu.: 780500   3rd Qu.:4.000   3rd Qu.:2.500   3rd Qu.:2631
##  Max.   :5570000   Max.   :7.000   Max.   :5.750   Max.   :9200
##      floors            view          condition         grade
##  Min.   :1.000   Min.   :0.0000   Min.   :3.000   Min.   : 5.000
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.: 7.000
##  Median :1.000   Median :0.0000   Median :3.000   Median : 7.000
##  Mean   :1.404   Mean   :0.2821   Mean   :3.442   Mean   : 7.827
##  3rd Qu.:2.000   3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.: 8.000
##  Max.   :3.000   Max.   :4.0000   Max.   :5.000   Max.   :13.000
##     yr_built       zipcode
##  Min.   :1904   Min.   :98001
##  1st Qu.:1948   1st Qu.:98001
##  Median :1964   Median :98039
##  Mean   :1968   Mean   :98083
##  3rd Qu.:1989   3rd Qu.:98199
##  Max.   :2014   Max.   :98199
```

Two variable summary statistics

```
pairs(Training) # scatter plot matrix
```




```
smoothScatter(Training$grade, Training$sqft_living)
```




```
smoothScatter(Training$sqft_living, Training$bathrooms)
```


```
round(cor(Training[,-1]), digits = 2)
```

```
##             bedrooms bathrooms sqft_living floors  view condition grade
## bedrooms        1.00      0.59        0.58   0.22  0.08      0.04  0.49
## bathrooms       0.59      1.00        0.84   0.58  0.21     -0.11  0.71
## sqft_living     0.58      0.84        1.00   0.45  0.24     -0.01  0.82
## floors          0.22      0.58        0.45   1.00  0.06     -0.30  0.55
## view            0.08      0.21        0.24   0.06  1.00      0.16  0.31
## condition       0.04     -0.11       -0.01  -0.30  0.16      1.00 -0.08
## grade           0.49      0.71        0.82   0.55  0.31     -0.08  1.00
## yr_built        0.16      0.35        0.23   0.57 -0.19     -0.44  0.33
## zipcode        -0.17      0.00        0.00  -0.02  0.23      0.17  0.19
##             yr_built zipcode
## bedrooms        0.16   -0.17
## bathrooms       0.35    0.00
## sqft_living     0.23    0.00
## floors          0.57   -0.02
## view           -0.19    0.23
## condition      -0.44    0.17
## grade           0.33    0.19
## yr_built        1.00   -0.42
## zipcode        -0.42    1.00
```

sqft_living is highly correlated with grade Later on we will see that sqft_living will be selected by best subset selection method as one of the predictors in linear regression. We can see that bathrooms has a high correlation with sqft_living, but later on, we will see that best subset selection never selects it, unless we want to include all predictors in our model.

3.3 Looking for missing values

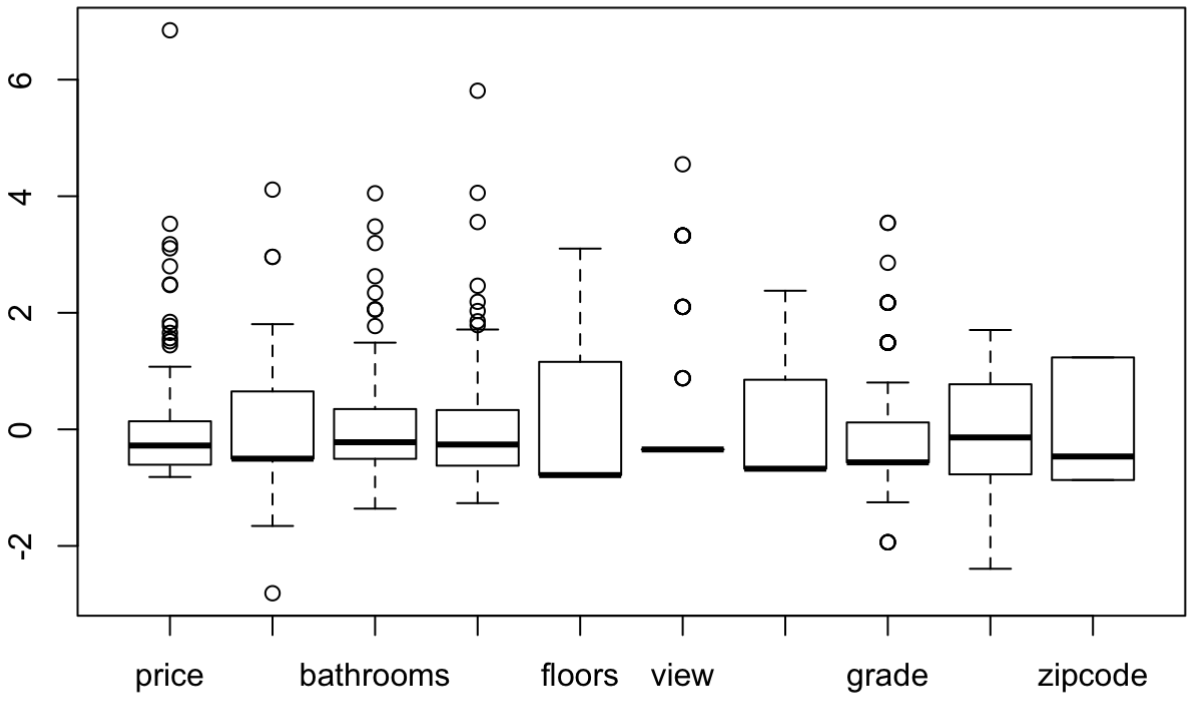
```
any(is.na(house_sales))
```

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

Yes! We don't have any NA values in our data set.

3.4 Looking for outliers

```
boxplot(scale(Training))
```



It is not obvious whether we can consider the points outside the boxplot area for each feature as outliers or not. Therefore, we do not remove any observations.

4 Full model fitting and assumption of normality

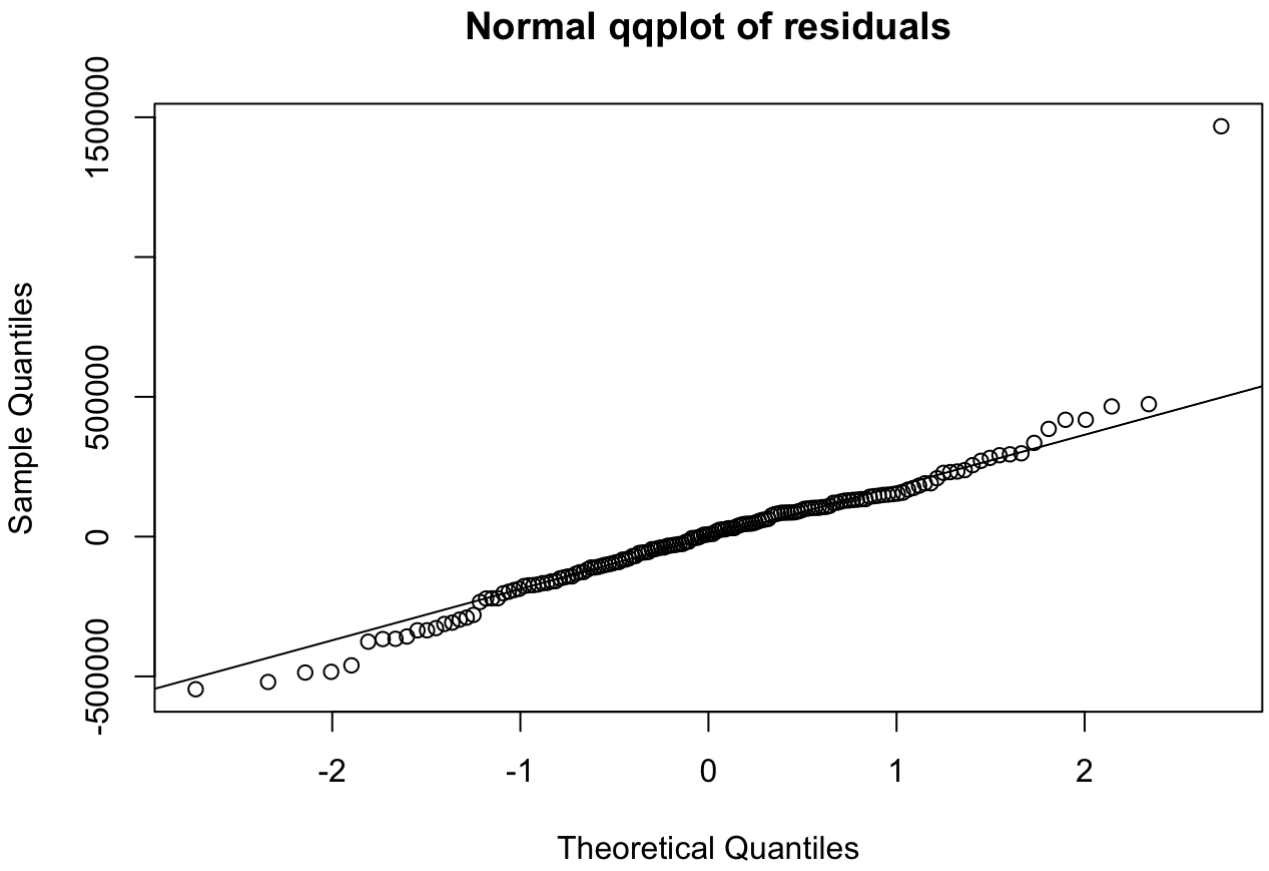
Fit linear regression with all predictors and make a qq-plot of the residuals.

```
#Fine-tuning the categorical variables
house_sales$zipcode <- factor(house_sales$zipcode)

Training = house_sales[ratio,] #Test dataset 30% of total
Validation = house_sales[-ratio,] #Train dataset 70% of total

lm_train <- lm(price~.,data = Training)
summary(lm_train)
```

```
##
## Call:
## lm(formula = price ~ ., data = Training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -545881 -127004    8155  121047 1467685
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4974831.41 2258563.25   2.203  0.02920 *
## bedrooms      -39722.10   29782.96  -1.334  0.18439
## bathrooms      -4736.47   47509.20  -0.100  0.92072
## sqft_living      356.13      41.71   8.538 1.69e-14 ***
## floors        -64465.89   56034.32  -1.150  0.25184
## view          -34576.99   27063.56  -1.278  0.20342
## condition     -27741.71   32875.32  -0.844  0.40015
## grade          89485.73   31827.44   2.812  0.00561 **
## yr_built       -2882.41    1175.46  -2.452  0.01539 *
## zipcode98039  893478.12   77911.23  11.468  < 2e-16 ***
## zipcode98199  317121.81   57677.35   5.498 1.68e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 235900 on 145 degrees of freedom
## Multiple R-squared:  0.8978, Adjusted R-squared:  0.8908
## F-statistic: 127.4 on 10 and 145 DF,  p-value: < 2.2e-16
```

Drawing the quantile quantile plot of the residuals. It indicates only minor departures from Normality.

```
#quantile-quantile plot
qqnorm(lm_train$residuals, main = "Normal qqplot of residuals")
qqline(lm_train$residuals)
```



We have long tails at both ends of the data distribution (thicker tails compared to Normal Distribution). Note that there's a residual far away from the q-q line. Let's check this outlier.

```
i=which(lm_train$residuals==max(lm_train$residuals));
Training[i,]
```

```
##        price bedrooms bathrooms sqft_living floors view condition grade
## 4412 5570000        5      5.75        9200      2    0         3    13
##      yr_built zipcode
## 4412     2001   98039
```

5 Model selection (variable selection)

```
library(leaps)
regfit.full <- regsubsets(price~., data = Training, really.big = T)
reg.summary <- summary(regfit.full)
reg.summary
```

```
## Subset selection object
## Call: regsubsets.formula(price ~ ., data = Training, really.big = T)
## 10 Variables  (and intercept)
##              Forced in Forced out
## bedrooms         FALSE      FALSE
## bathrooms        FALSE      FALSE
## sqft_living      FALSE      FALSE
## floors           FALSE      FALSE
## view             FALSE      FALSE
## condition        FALSE      FALSE
## grade            FALSE      FALSE
## yr_built         FALSE      FALSE
## zipcode98039     FALSE      FALSE
## zipcode98199     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          bedrooms bathrooms sqft_living floors view condition grade
## 1  ( 1 ) " "      " "       "*"         " "    " "  " "       " "
## 2  ( 1 ) " "      " "       "*"         " "    " "  " "       " "
## 3  ( 1 ) " "      " "       "*"         " "    " "  " "       " "
## 4  ( 1 ) " "      " "       "*"         " "    " "  " "       " "
## 5  ( 1 ) " "      " "       "*"         " "    " "  " "       "*"
## 6  ( 1 ) "*"      " "       "*"         " "    " "  " "       "*"
## 7  ( 1 ) "*"      " "       "*"         " "    "*"  " "       "*"
## 8  ( 1 ) "*"      " "       "*"         "*"    "*"  " "       "*"
##          yr_built zipcode98039 zipcode98199
## 1  ( 1 ) " "      " "           " "
## 2  ( 1 ) " "      "*"           " "
## 3  ( 1 ) " "      "*"           "*"
## 4  ( 1 ) "*"      "*"           "*"
## 5  ( 1 ) "*"      "*"           "*"
## 6  ( 1 ) "*"      "*"           "*"
## 7  ( 1 ) "*"      "*"           "*"
## 8  ( 1 ) "*"      "*"           "*"
```

In the output, an asterisk indicates that a given variable is included in the corresponding model. For instance, this output indicates that: the best one-variable model contains only sqft_living, the best two-variable model contains only sqft_living and zipcode and so on.

Now we have to find the best model among these 8 models. We know that the model containing all of the predictors will always have the smallest RSS and the largest R2, since these quantities are related to the training error.

Instead, we wish to choose a model with a low test error and the training error can be a poor estimate of the test error.

Therefore, note that RSS and R2 are not suitable for selecting the best model among a collection of models with different numbers of predictors.

In order to select the best model with respect to test error, we need to estimate this test error. There are two common approaches:

1. We can indirectly estimate test error by adding a penalty to the training error to account for the bias due to overfitting.

2. We can directly estimate the test error, using either a validation set approach or a cross-validation approach

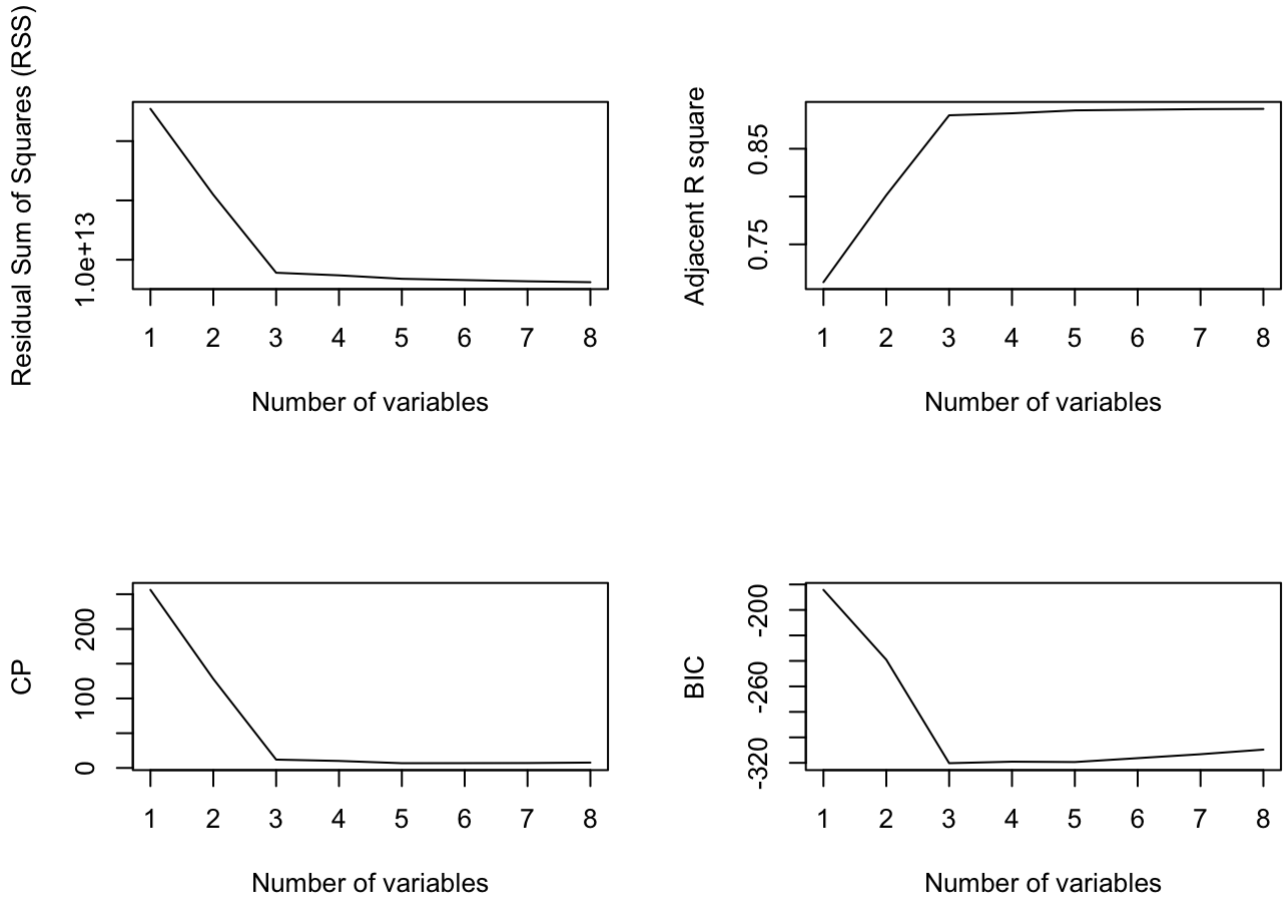
Here we will use the first approach.

A number of techniques for adjusting the training error for the model size are available. These approaches can be used to select among a set of models with different numbers of variables.

We now consider four such approaches: **Cp, Akaike information criterion (AIC), Bayesian information criterion (BIC), and adjusted R2.**

Let's plot them:

```
par(mfrow = c(2,2))
plot(reg.summary$rss, xlab = "Number of variables", ylab = "Residual Sum of Squares (RS
S)", type = "l")
plot(reg.summary$adjr2, xlab = "Number of variables", ylab = "Adjacent R square", type =
 "l")
plot(reg.summary$cp, xlab = "Number of variables", ylab = "CP", type = "l")
plot(reg.summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
```



By looking at the plots, it seems that choosing 3 predictors is fine, although we can see slight increase using more than 3 variables.

```
#We can also see it like this:
which.min(reg.summary$bic)
```

```
## [1] 3
```

These 3 predictors based on best subset selection that we did above are sqft_living and zipcodes separeted as two variables.

Note that if you look at the RSS plot you can see that as the number of predictors increases the RSS decreases. We will mention the reason and why we shouldn't rely on RSS in previous paragraph.

Now let's get the coefficients for the linear model based on 3 predictors.

```
# linear model based on 3 predictors
subset_select_3predictor <- lm(price ~ sqft_living + zipcode, data = Training)
# coefficients of the predictors
coef(regfit.full, 3)
```

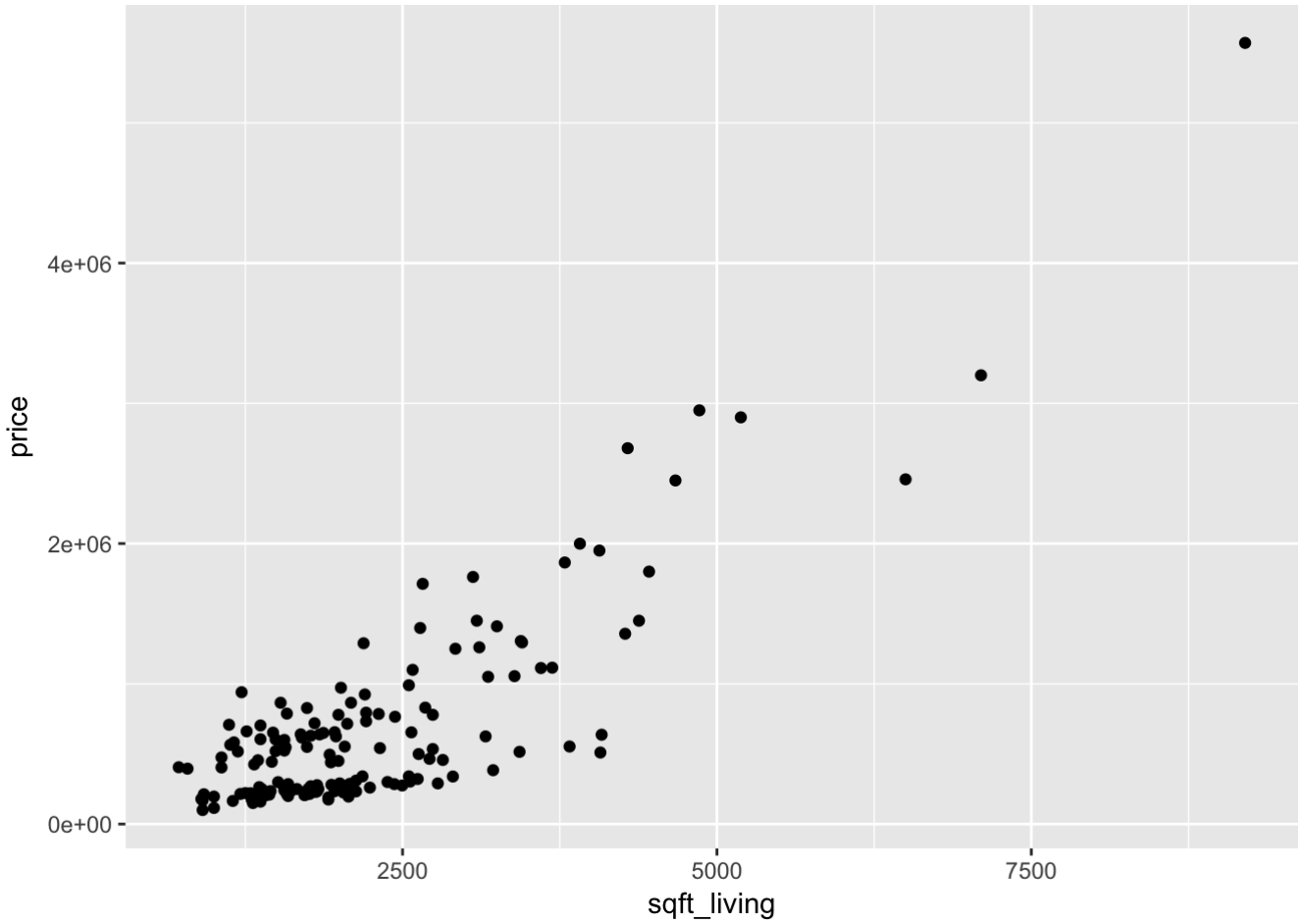
```
##  (Intercept)  sqft_living zipcode98039 zipcode98199
## -461912.8933     383.0053 1001784.6222  441607.0067
```

The linear model is : price = -461912.8933 + 383.0053×sqft_living + 1001784.6222xzipcode98039 + 441607.0067xzipcode98199 + error, error-iid->N(0, σ2)

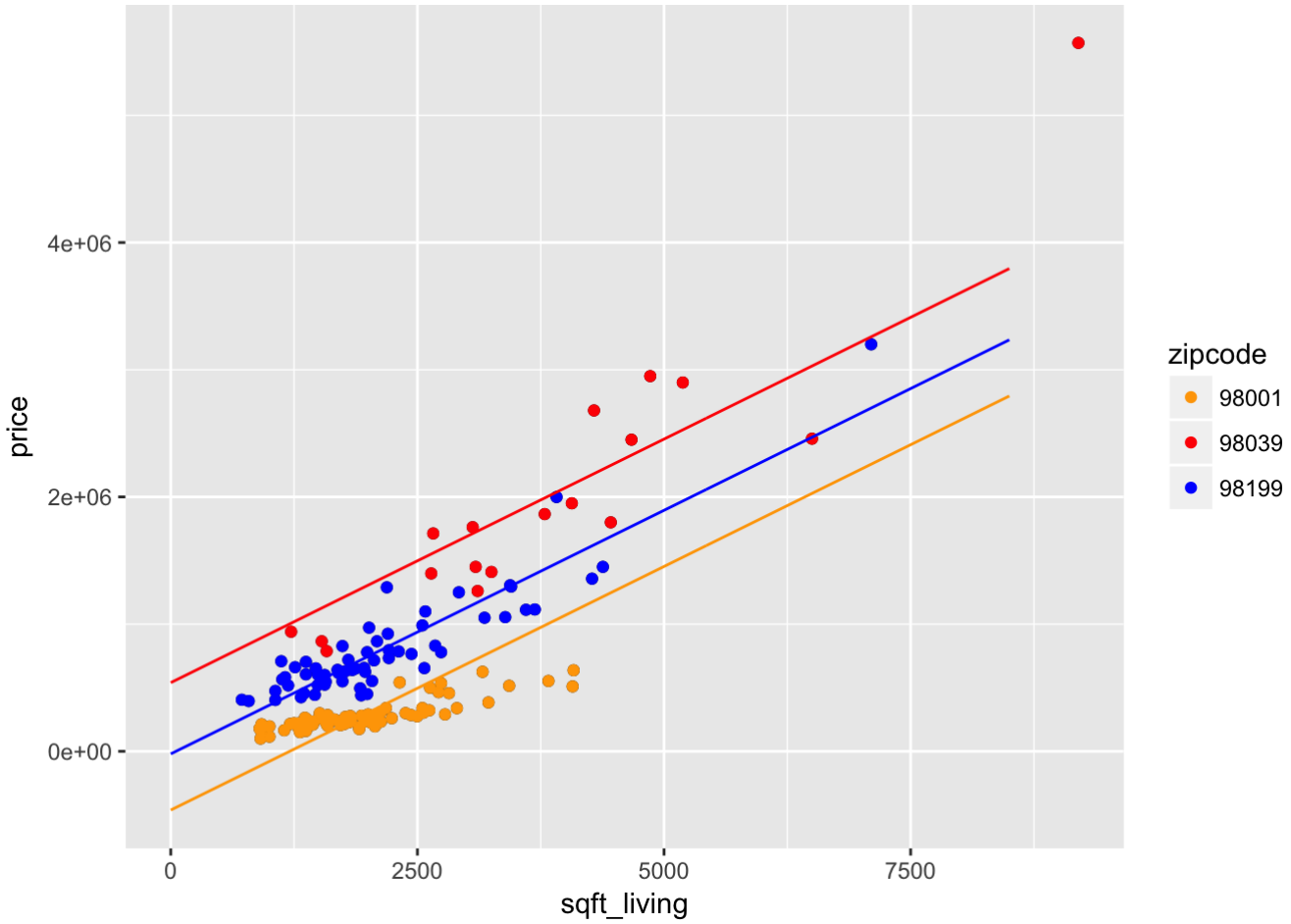
Moving on with **best subset selection**…

Our Tranining set:

```
g <- ggplot(data= Training, aes(y = price, x = sqft_living)) + geom_point(aes())
g
```




```
t = seq(0, 8500, 8500/155)

y1 = function(t) {
  coef(regfit.full, 3)[1] + coef(regfit.full, 3)[2] * t + coef(regfit.full, 3)[3]
  # -461912.8933 + 383.0053*t + 1001784.6222
}

y2 = function(t) {
  coef(regfit.full, 3)[1] + coef(regfit.full, 3)[2] * t + coef(regfit.full, 3)[4]
  # -461912.8933 + 383.0053*t +  441607.0067
}

y3 = function(t) {
  coef(regfit.full, 3)[1] + coef(regfit.full, 3)[2] * t
  # -461912.8933 + 383.0053*t
}

Palette <- c('orange','red','blue')

g +
  geom_point(aes(color=zipcode)) + scale_color_manual(values=Palette) +
  geom_line(aes(x=t, y=y1(t)), color='red') +
  geom_line(aes(x=t, y=y2(t)), color='blue') +
  geom_line(aes(x=t, y=y3(t)), color='orange')
```



If we add statistical interaction between zipcode and sqft_living

Interactions:

We have,

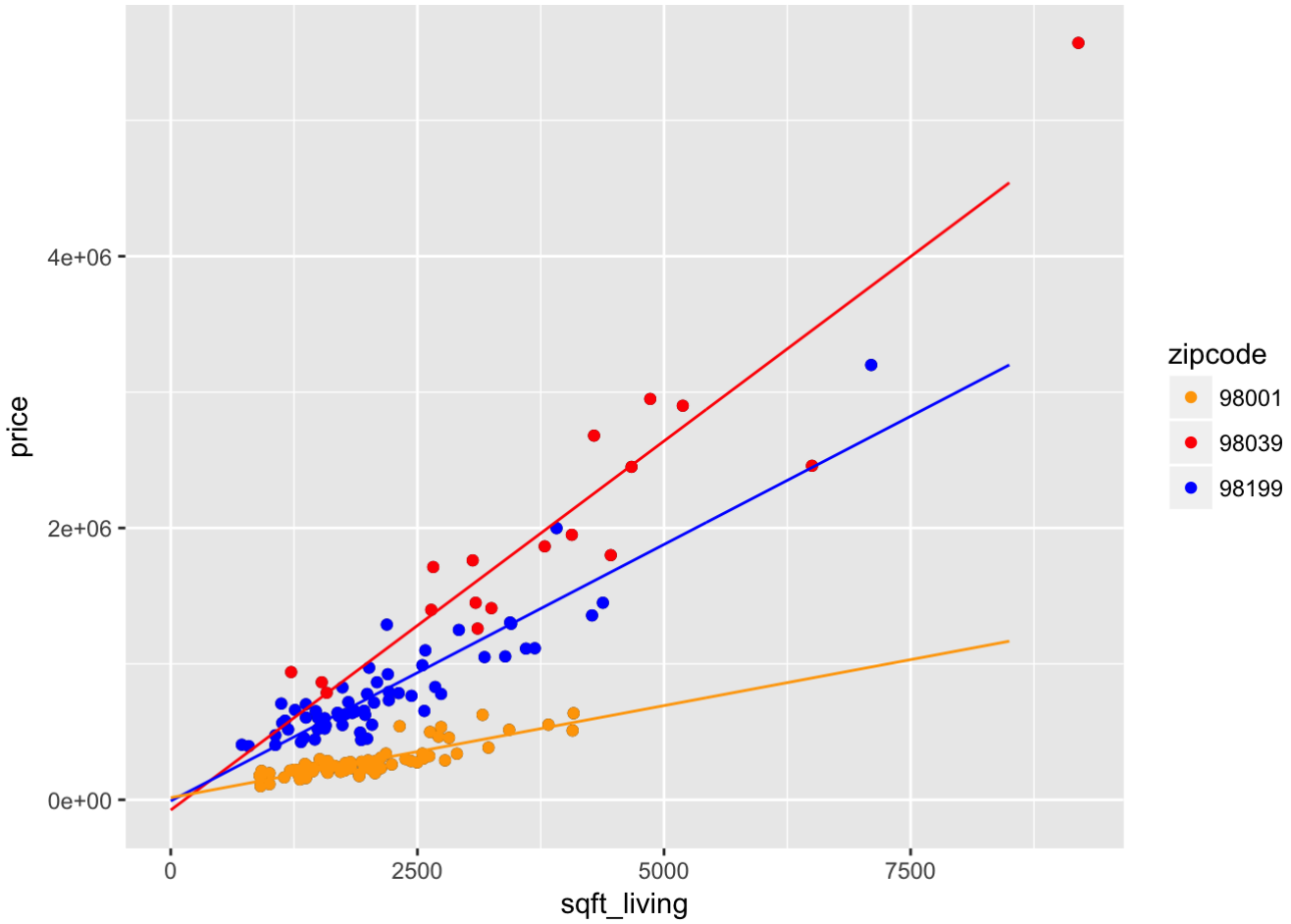
```
x <- model.matrix(price~.,Training)[,-1]
y <- Training$price

x_val <- model.matrix(price~.,Validation)[,-1]
y_val <- Validation$price
```

```
x_prime <- as.data.frame(x)
y_prime <- y
x_prime$sqft_zip98039 <- x_prime$sqft_living * x_prime$zipcode98039
x_prime$sqft_zip98199 <- x_prime$sqft_living * x_prime$zipcode98199
Training_prime <- cbind(y_prime, x_prime)

interaction_model <- lm(y_prime ~
                          sqft_living + zipcode98039 +
                          zipcode98199 + sqft_zip98039 +
                          sqft_zip98199, data = Training_prime)
interaction_model
```

```
##
## Call:
## lm(formula = y_prime ~ sqft_living + zipcode98039 + zipcode98199 +
##     sqft_zip98039 + sqft_zip98199, data = Training_prime)
##
## Coefficients:
##   (Intercept)    sqft_living   zipcode98039   zipcode98199  sqft_zip98039
##       15867.2          135.4       -91161.9       -24386.7          407.7
## sqft_zip98199
##         242.1
```

The linear model is : price = 15867.2 + 135.4×sqft_living + -91161.9xzipcode98039 + -24386.7xzipcode98199 + 407.7xsqft_zip98039 + 242.1xsqft_zip98199 + error, error-iid->N(0, σ2)

```
t = seq(0, 8500, 8500/155)

y1 = function(t) {
  coef(interaction_model, 5)[1] + coef(interaction_model, 5)[2]*t + coef(interaction_mod
el, 5)[3] + coef(interaction_model, 5)[5]*t
  # 15867.2 + 135.4*t + -91161.9 +  407.7*t
}

y2 = function(t) {
  coef(interaction_model, 5)[1] + coef(interaction_model, 5)[2]*t + coef(interaction_mod
el, 5)[4] + coef(interaction_model, 5)[6]*t
  # 15867.2 + 135.4*t + -24386.7 +  242.1*t
}

y3 = function(t) {
  coef(interaction_model, 5)[1] + coef(interaction_model, 5)[2]*t
  # 15867.2 + 135.4*t
}

ggplot(data= Training, aes(y = price, x = sqft_living)) + geom_point(aes()) +
  geom_point(aes(color=zipcode)) + scale_color_manual(values=Palette) +
  geom_line(aes(x=t, y=y1(t)), color='red') +
  geom_line(aes(x=t, y=y2(t)), color='blue') +
  geom_line(aes(x=t, y=y3(t)), color='orange')
```



5.1 Regularization

Ridge regression:

Plot the ridge regression coefficients for different values of λ :

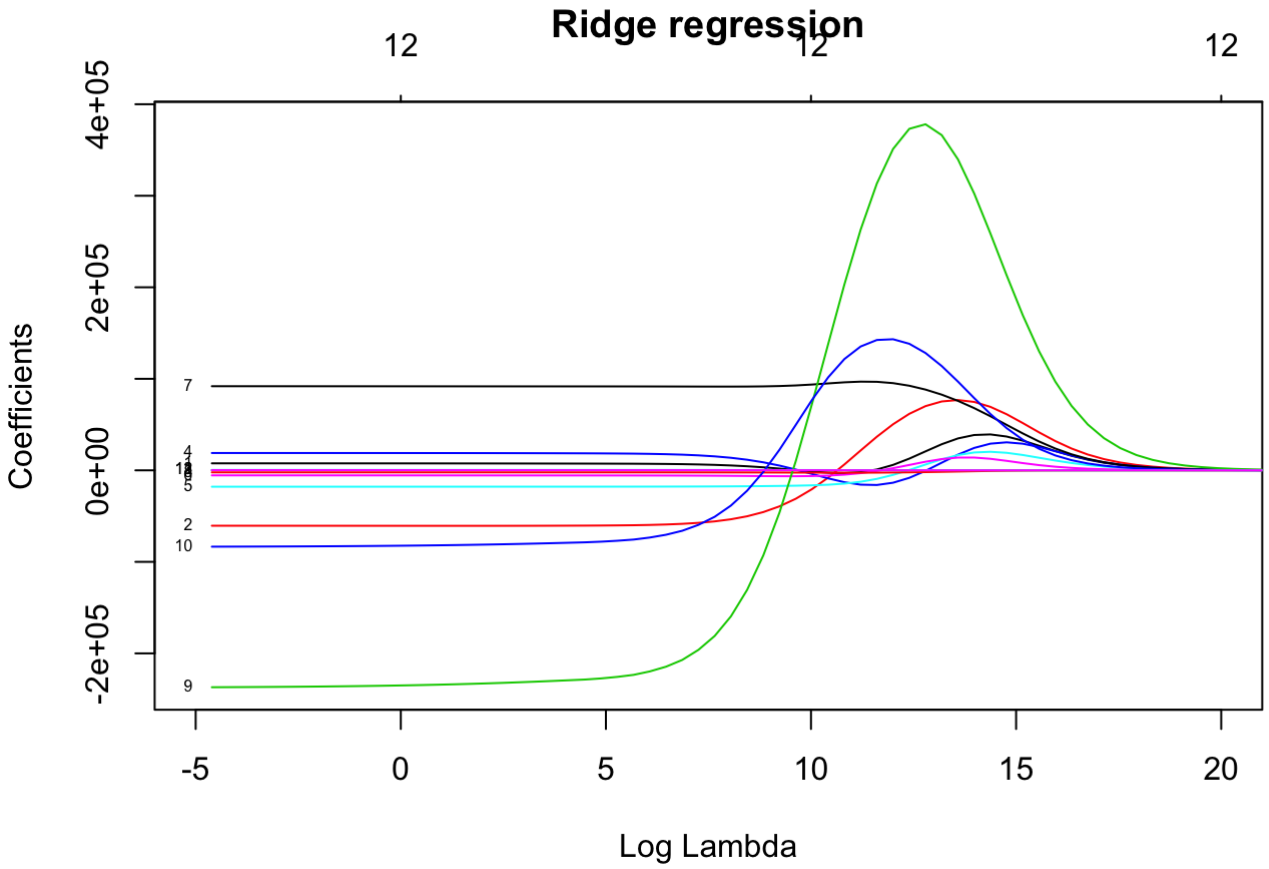
```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```
x <- model.matrix(y_prime~.,Training_prime)[,-1]
y <- Training_prime$y_prime
grid = 10^seq(15,-2, length = 100)
ridge.mode = glmnet(x, y, alpha=0, lambda = grid)
plot(ridge.mode, main = "Ridge regression",label = TRUE, xvar = "lambda", xlim = c(-5,20
))
```



By default the glmnet() function performs ridge regression for an automatically selected range of λ values.

However, here we have chosen to implement the function over a grid of values ranging from λ = 10^20 to λ = 10^-5 Essentially covering the full range of scenarios from the null model containing only the intercept, to the least squares fit.

Note that in order to evaluate the performance of a statistical learning method on a given data set, we need some way to measure how well its predictions actually match the observed data.

That is, we need to quantify the extent to which the predicted response value for a given observation is close to the true response value for that observation. In the regression setting, the most commonly-used measure is the mean squared error.

Basically, you will find a model based on the training set and you will evaluate your model on the validation set by computing MSE on the validation set.

The MSE will be small if the predicted responses are very close to the true responses, and will be large if for some of the observations, the predicted and true responses differ substantially.

6 Cross-Validation

In general, instead of arbitrarily choosing λ, it would be better to use cross-validation to choose the tuning parameter λ.

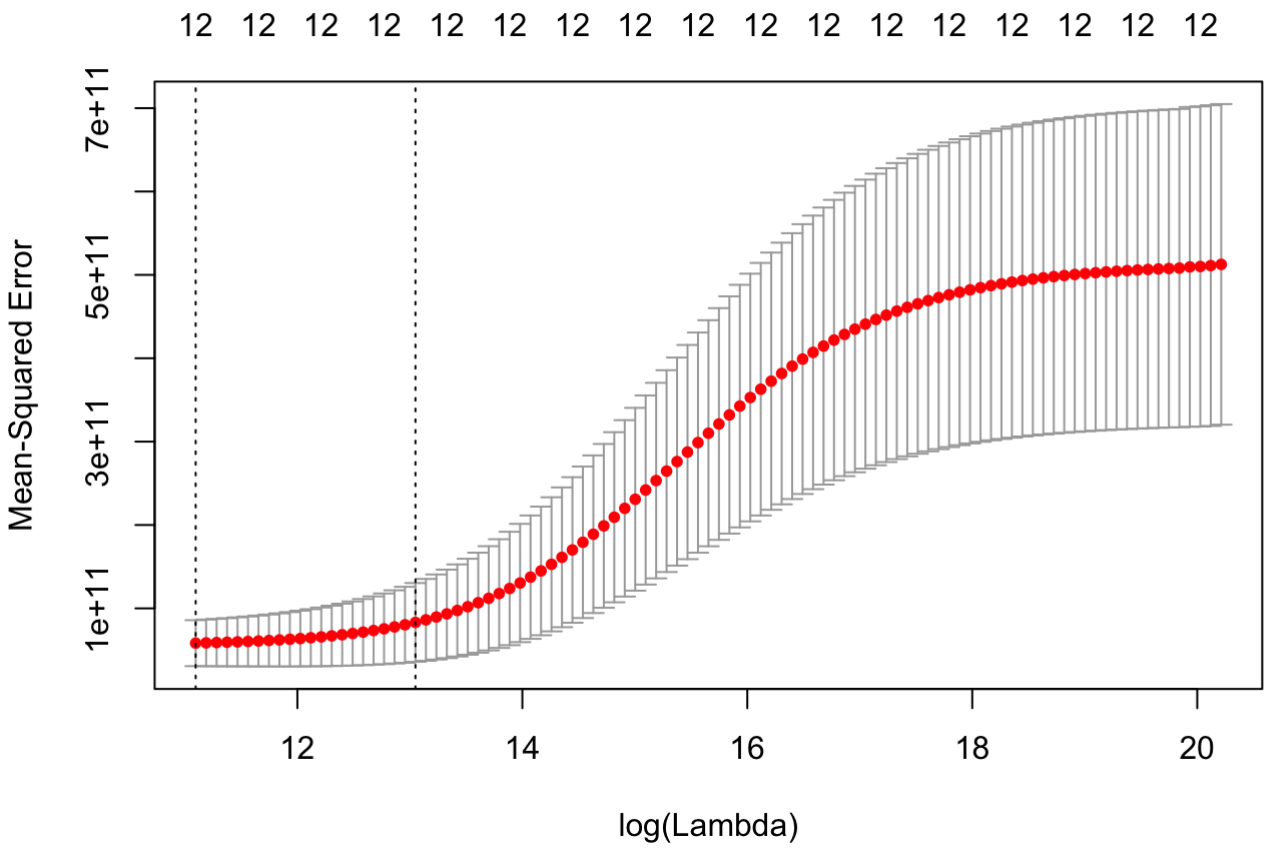
It involves randomly dividing the available set of observations (here our observations are `Training`) into two parts, a training set and a validation set or hold-out set.

The model is fit on the training set, and the fitted model is used to predict the response for the observations in the validation set.

The resulting validation set error rate typically used MSE in the case of a quantitative response to provide and estimate the test error rate.

We can do this using the built-in cross-validation function, cv.glmnet(). By default, the function performs ten-fold cross-validation, though this can be changed using the argument folds.

```
cv.out <- cv.glmnet(x,y, alpha = 0)
plot(cv.out)
```



If we look at the top of this plot, we can see that regardless of range of λ the model always selects 12 predictors.

This is because the ridge doesn't do variable selection. You can see that a range of log(λ) is specified between two vertical dashed lines which shows a range of log(λ) that the MSE is almost the same and it is significantly less than the MSE for other ranges of log(λ). The model chooses the λ with least MSE.

Here we find the best λ:

```
bestlam.ridge = cv.out$lambda.min
bestlam.ridge
```

```
## [1] 65898.37
```

```
log(bestlam.ridge)
```

```
## [1] 11.09587
```

Now we fit a ridge regression model on the training set.

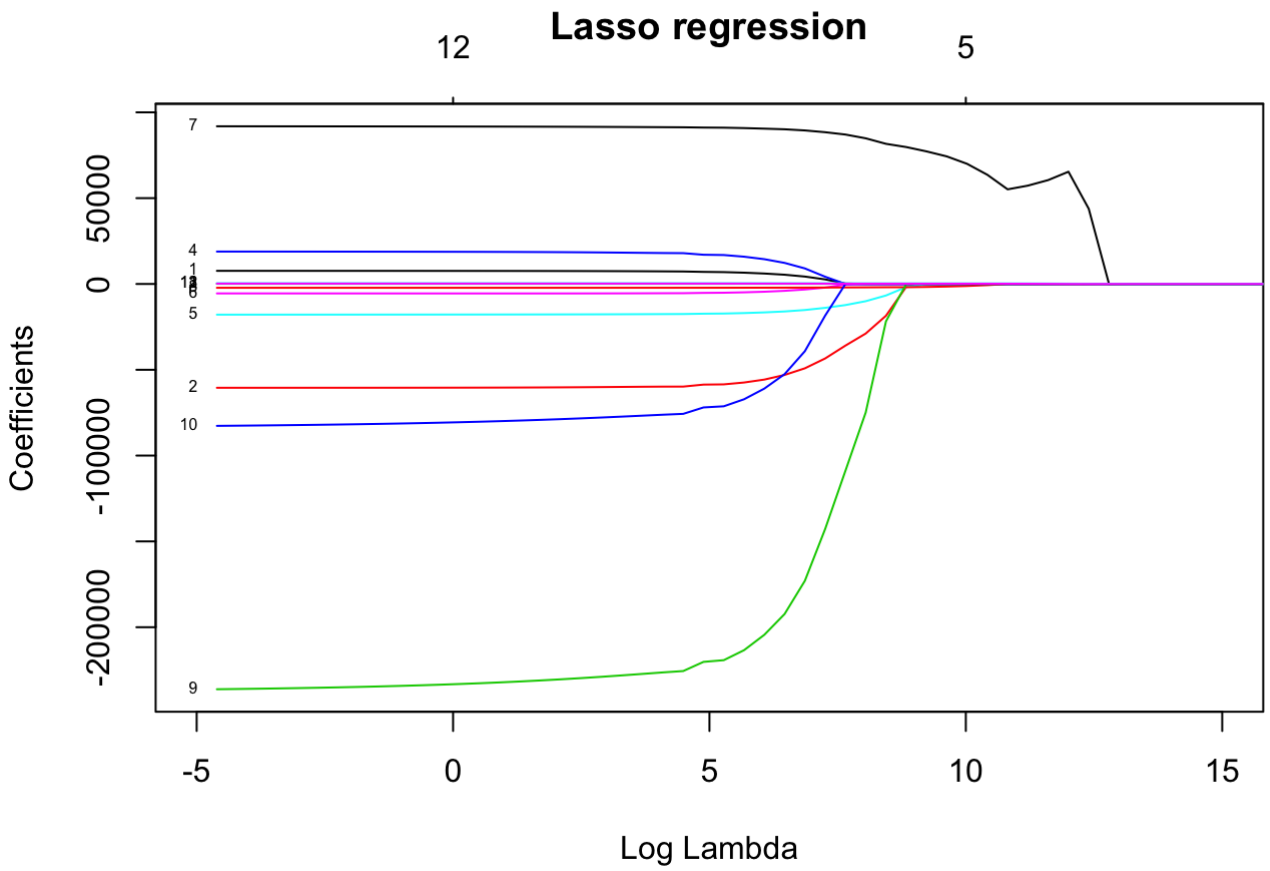
```
ridge.mode <- glmnet(x, y, alpha=0, lambda = bestlam.ridge)
predict(ridge.mode, s = bestlam.ridge, type = "coefficients")
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)   4410695.8350
## bedrooms        -2392.7268
## bathrooms       17173.7151
## sqft_living       164.3615
## floors         -14978.9992
## view           -13149.2518
## condition       -3639.7344
## grade           96722.0047
## yr_built        -2594.5672
## zipcode98039   245774.6393
## zipcode98199   131727.4689
## sqft_zip98039     233.8720
## sqft_zip98199      93.5022
```

We end up with a model with 12 predictors by ridge method.

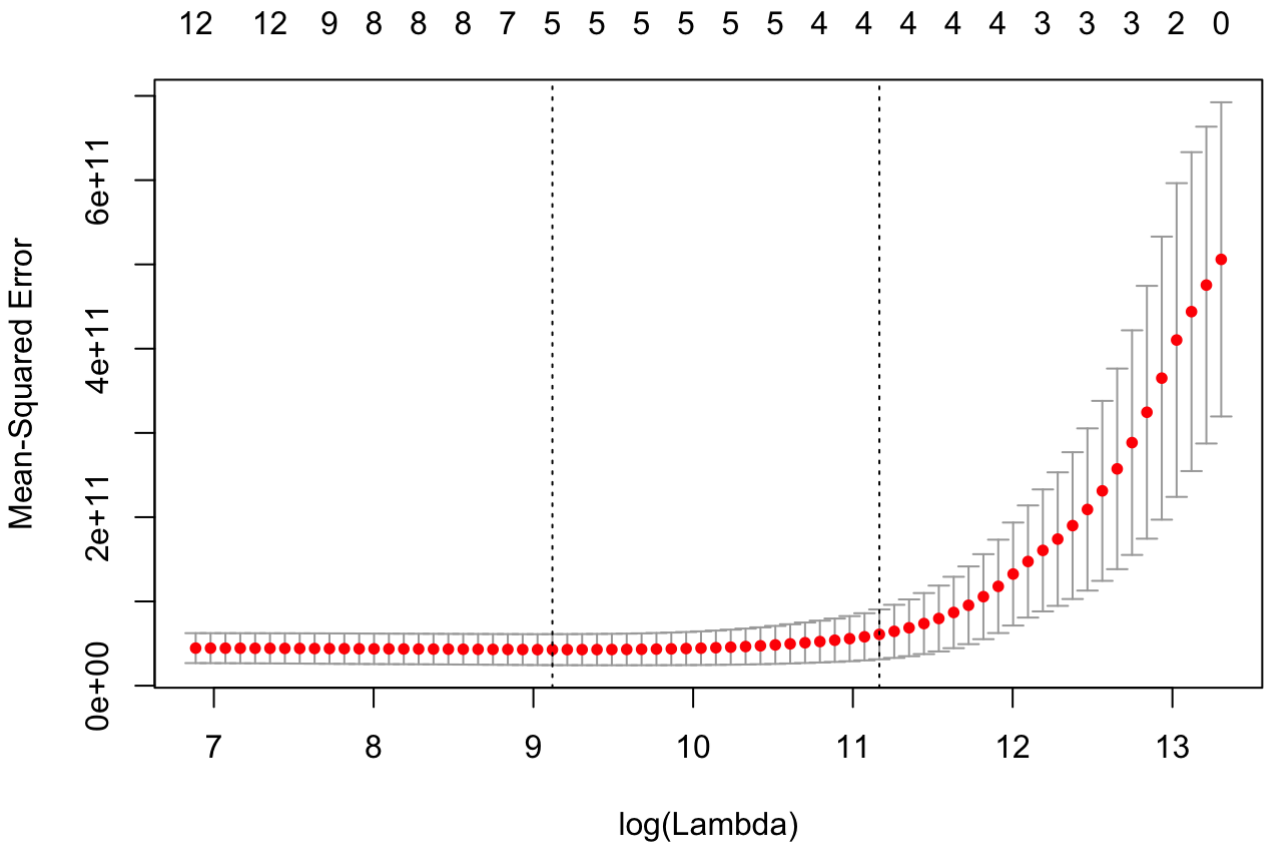
For the lasso regression, again, first We will plot the coefficients for different values of λ :

```
lasso.mod <- glmnet(x,y, alpha = 1, lambda = grid)
plot(lasso.mod, main = "Lasso regression", label = TRUE, xvar = "lambda", xlim = c(-5,15
))
```



Now we will perform cross-validation and compute the best λ :

```
cv.out <- cv.glmnet(x,y,alpha = 1)
plot(cv.out)
```




```
bestlam.lasso <- cv.out$lambda.min
bestlam.lasso
```

```
## [1] 9126.176
```

```
log(bestlam.lasso)
```

```
## [1] 9.118902
```

If you look at the top of this plot, you can see that for different values of λ the model selects different number of predictors. This is because the lasso does the variable selection (note that ridge does not). You can see that a range of log(λ) is specified between two vertical dashed lines which shows a range of log(λ) that the MSE is almost the same and it is significantly less than the MSE for other ranges of λ. For this range of log(λ) the number of predictors selected by lasso is one of 5 or 4. The model chooses the λ with least MSE which contains 5 predictors. Let's find the best λ:

```
lasso.mode <- glmnet(x, y, alpha=1, lambda = bestlam.lasso)
predict(lasso.mode, s = bestlam.lasso, type = "coefficients")[1:13,]
```

```
##   (Intercept)      bedrooms     bathrooms   sqft_living        floors
##  3055447.2960        0.0000        0.0000      110.7560        0.0000
##          view     condition         grade      yr_built  zipcode98039
##        0.0000        0.0000    78634.1747    -1797.1089        0.0000
##  zipcode98199 sqft_zip98039 sqft_zip98199
##        0.0000      346.7234      175.7655
```

You can see that the coefficients like `bathrooms` , `bedrooms` and etc. are zero, which means that we should not include it in our model.

We end up with a model with 5 predictors by lasso method.

There are various special algorithms for LASSO that you can read about them in the chapter 13 section 4 of KM textbook.

To evaluate the performance of the models that we found in previous sections, We will calculate the mean square error for all the models:

```
real_value_price <- Validation$price
Validation$price <- NULL
newx <- data.matrix(Validation)

x_prime_val <- as.data.frame(x_val)
y_prime_val <- y_val
x_prime_val$sqft_zip98039 <- x_prime_val$sqft_living * x_prime_val$zipcode98039
x_prime_val$sqft_zip98199 <- x_prime_val$sqft_living * x_prime_val$zipcode98199
Validation_prime <- cbind(y_prime_val, x_prime_val)

# predict the price based on the linear regression model on all predictors
all_variables_prediction <- predict(lm_train, newdata = Validation)
# predict price on 3 predictors selected in all subset selection
all_subset_selection_prediction <- predict(subset_select_3predictor, newdata = Validatio
n)
# predict price on 5 predictors selected in all subset selection with interactions
all_subset_selection_interaction_prediction <- predict(interaction_model, newdata = Vali
dation_prime)
```

```
Validation_reg = house_sales[-ratio,] #Validation dataset the same 30% of total

x <- model.matrix(price~.,Validation_reg)[,-1]
y <- Validation_reg$price

sqft_zip98039 <- x[,3] * x[,9]
sqft_zip98199 <- x[,3] * x[,10]

x <- cbind(x,sqft_zip98039,sqft_zip98199)

newx <- data.matrix(x)

# predict price based on lasso regression
lasso_prediction <- predict(lasso.mode, newx = newx)
# predict price based on ridge regression
ridge_prediction <- predict(ridge.mode, newx = newx)
```

Now that we have all the predicted values of price based on our 4 models, we can compare the real prices in validation set with the predicted values to find out which model performs best. We will choose to do it by computing the value of MSE for each model.

```
error_all_variables_prediction <- mean((all_variables_prediction-real_value_price)^2)
error_all_variables_prediction
```

```
## [1] 41084725488
```

```
cor(all_variables_prediction,real_value_price)
```

```
## [1] 0.9348527
```

```
error_all_subset_selection_prediction <- sum((all_subset_selection_prediction - real_val
ue_price)^2)/nrow(Validation)
error_all_subset_selection_prediction
```

```
## [1] 51380438439
```

```
cor(all_subset_selection_prediction,real_value_price)
```

```
## [1] 0.9172659
```

```
error_all_subset_selection_interaction_prediction <- sum((all_subset_selection_interacti
on_prediction - real_value_price)^2)/nrow(Validation)
error_all_subset_selection_interaction_prediction
```

```
## [1] 39276155374
```

```
cor(all_subset_selection_interaction_prediction,real_value_price)
```

```
## [1] 0.9324469
```

```
error_lasso_prediction <-
sum((lasso_prediction - real_value_price)^2)/nrow(Validation)
error_lasso_prediction
```

```
## [1] 29787281196
```

```
cor(lasso_prediction,real_value_price)
```

```
##         [,1]
## s0 0.9493449
```

```
error_ridge_prediction <-
sum((ridge_prediction - real_value_price)^2)/nrow(Validation)
error_ridge_prediction
```

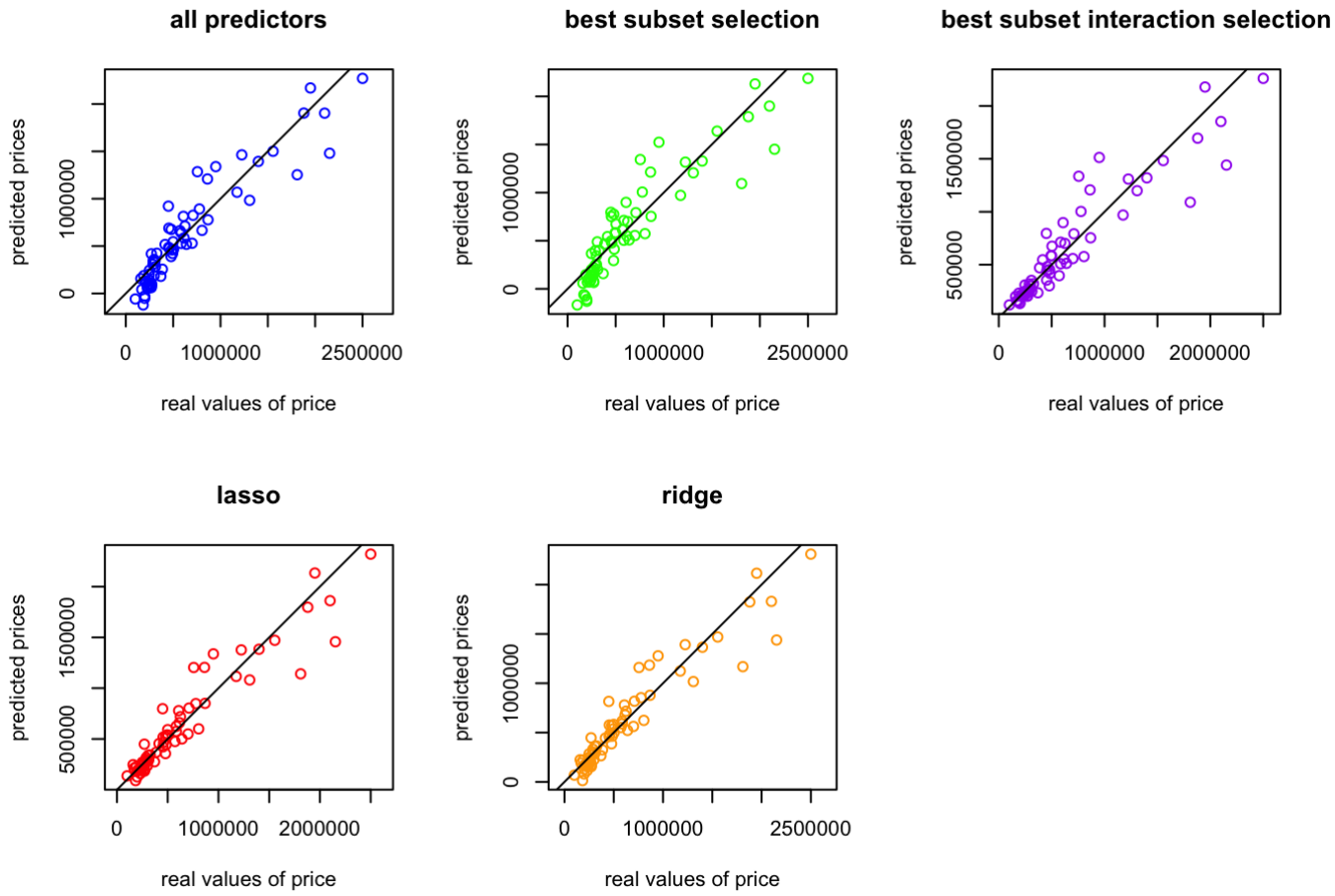
```
## [1] 30082320111
```

```
cor(ridge_prediction,real_value_price)
```

```
##         [,1]
## s0 0.9490769
```

Note line `y=x` is the best model

```
par(mfrow = c(2,3))
plot(x = real_value_price, y = all_variables_prediction, xlab = "real values of price",
ylab = "predicted prices", main = "all predictors", col = "blue", asp=1 )
abline(a = 0, b = 1)
plot(x = real_value_price, y = all_subset_selection_prediction, xlab = "real values of p
rice",
ylab = "predicted prices", main = "best subset selection", col = "green", asp=1)
abline(a = 0, b = 1)

plot(x = real_value_price, y = all_subset_selection_interaction_prediction, xlab = "real
 values of price",
ylab = "predicted prices", main = "best subset interaction selection", col = "purple", a
sp=1)
abline(a = 0, b = 1)

plot(x = real_value_price, y = lasso_prediction, xlab = "real values of price",
ylab = "predicted prices", main = "lasso", col = "red", asp=1)
abline(a = 0, b = 1)
plot(x = real_value_price, y = ridge_prediction, xlab = "real values of price",
ylab = "predicted prices", main = "ridge", col = "orange", asp=1)
abline(a = 0, b = 1)
```



7 Interpreting the Predictions

```
order(c(error_all_variables_prediction,
          error_all_subset_selection_prediction,
          error_all_subset_selection_interaction_prediction,
          error_lasso_prediction,
          error_ridge_prediction))
```

```
## [1] 4 5 3 1 2
```

It seems that the model chosen by lasso selection is the best model among other models (not much difference!). We know that lasso includes 5 predictors in the model, but with different coefficients assigned to them than what linear regression will assign to.

Just to dig more into variable selection and compare different methods that we have learned so far, we will want to compare all subset selection (and with interaction), lasso and linear regression as follows:

We know that lasso does the subset selection and if you look at the plot above you can see that the suggested 4 predictors and then applied that model on the validation set and calculated the MSE.

You can do this kind of comparison yourself. Pick a value of λ from the plot that results in 4 predictors. Fit a lasso regression again on the training set with that λ and find the coefficients for the model and find out which predictors get the zero coefficient and are not included in the model.

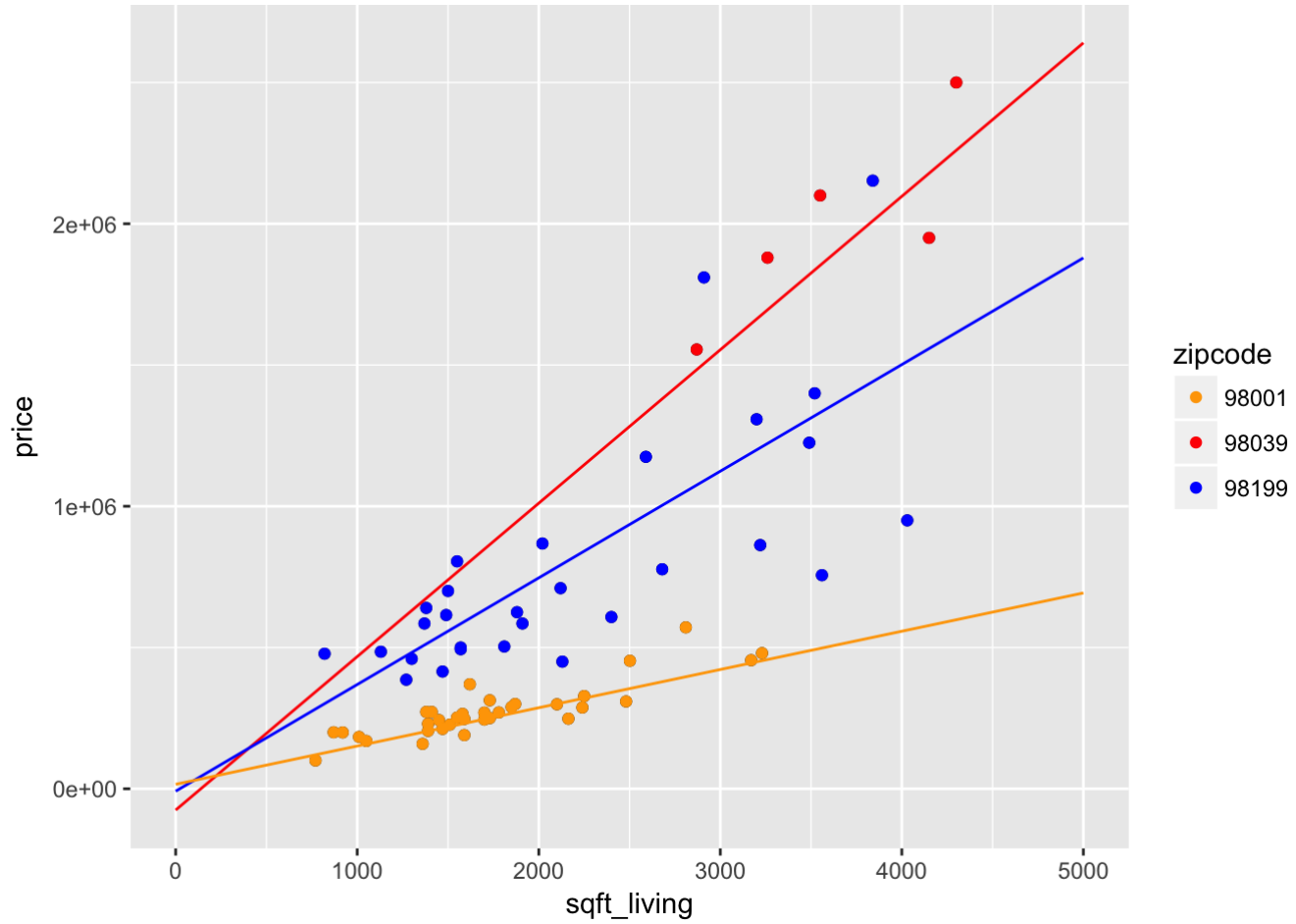
you should find predictors with zero coefficients! Fit your model on the validation set and find it's MSE.

Now move to simple linear regression and apply linear regression on those 4 predictors that lasso founds.

Then fit your linear model on the validation set and find it's MSE. Compare this two MSE with each other and see which one does a better job.

To see performance of `interaction_model` on validation set:

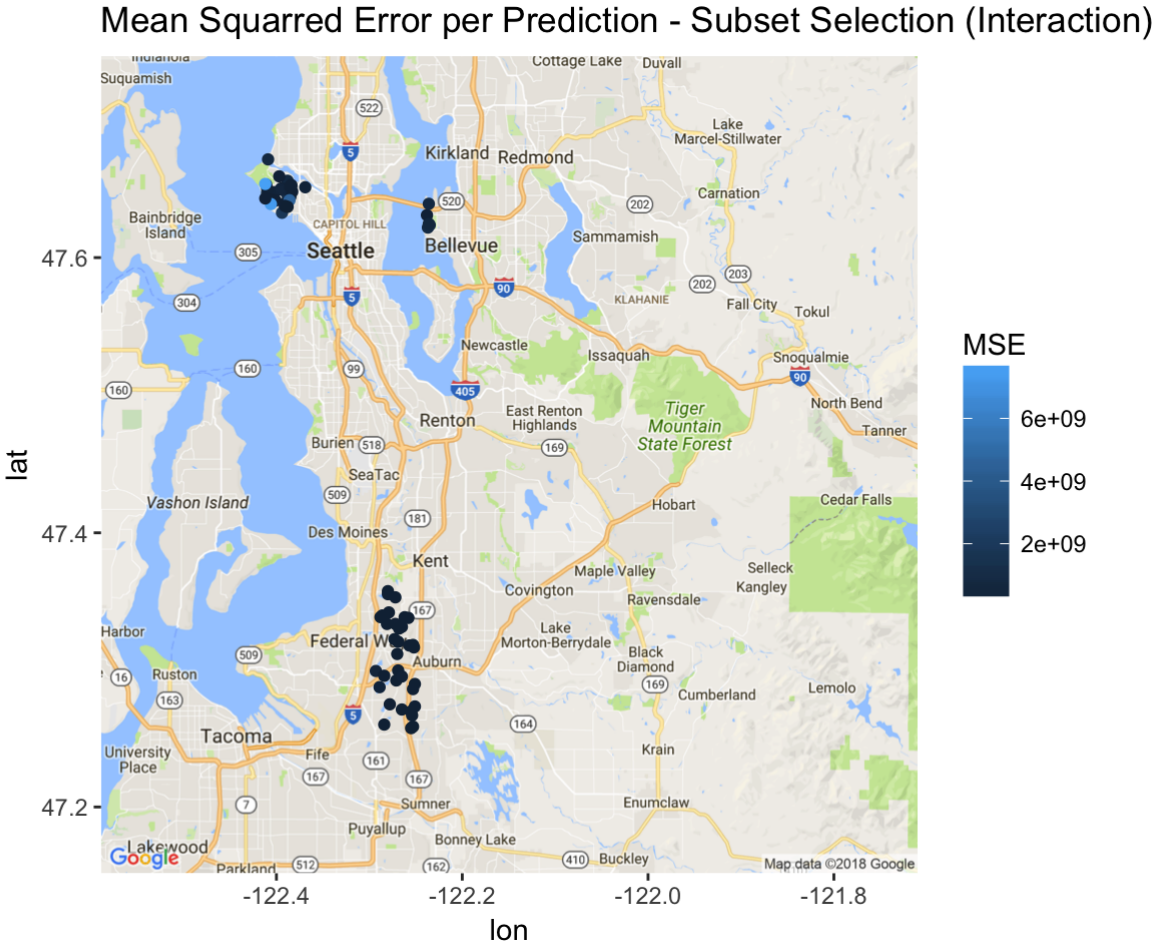
```
t = seq(0, 5000, 5000/67)



y1 = function(t) {
  coef(interaction_model, 5)[1] + coef(interaction_model, 5)[2]*t + coef(interaction_mod
el, 5)[3] + coef(interaction_model, 5)[5]*t
  # 15867.2 + 135.4*t + -91161.9 +  407.7*t
}

y2 = function(t) {
  coef(interaction_model, 5)[1] + coef(interaction_model, 5)[2]*t + coef(interaction_mod
el, 5)[4] + coef(interaction_model, 5)[6]*t
  # 15867.2 + 135.4*t + -24386.7 +  242.1*t
}

y3 = function(t) {
  coef(interaction_model, 5)[1] + coef(interaction_model, 5)[2]*t
  # 15867.2 + 135.4*t
}

ggplot(data= Validation_reg, aes(y = price, x = sqft_living)) + geom_point(aes()) +
  geom_point(aes(color=zipcode)) + scale_color_manual(values=Palette) +
  geom_line(aes(x=t, y=y1(t)), color='red') +
  geom_line(aes(x=t, y=y2(t)), color='blue') +
  geom_line(aes(x=t, y=y3(t)), color='orange')
```



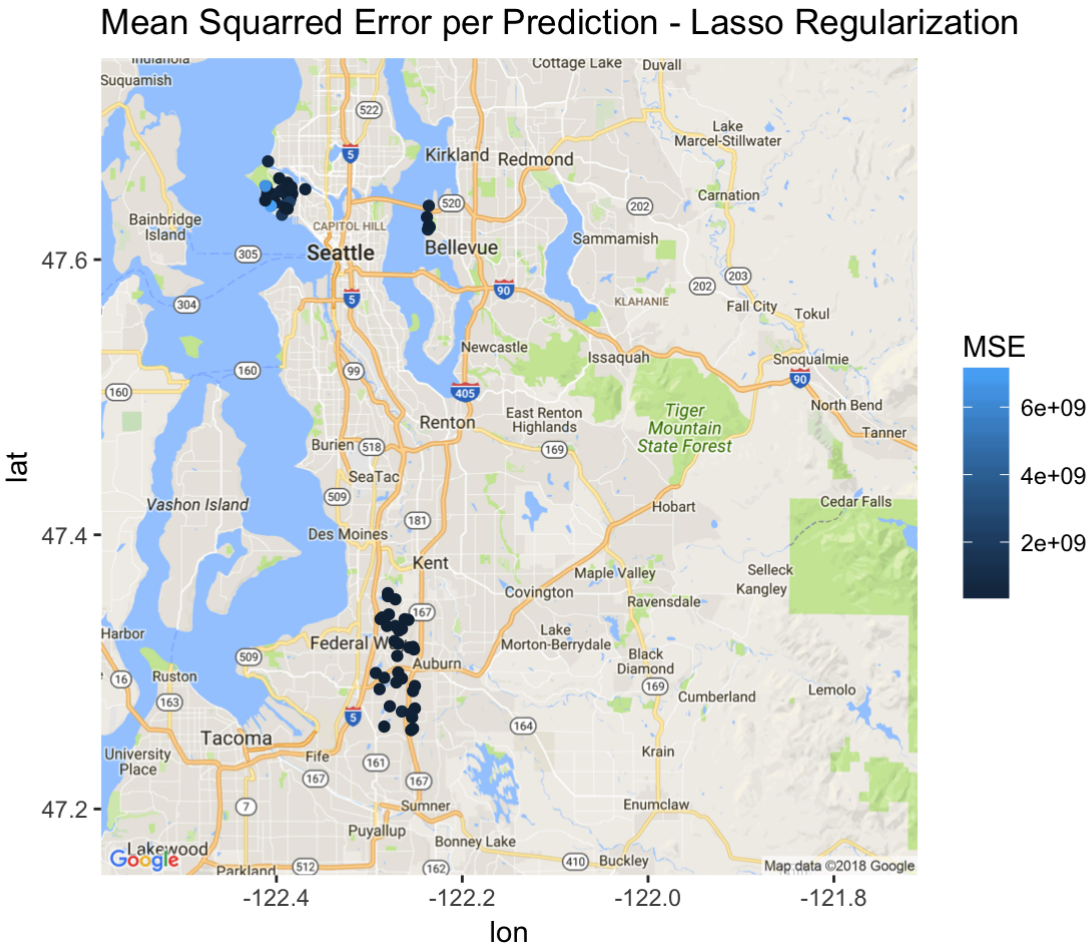
```
map = get_map(location = c(-122.15, 47.45),
              zoom = 10, source = "google", maptype="roadmap")
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=47.45,-122.15&zoom=10&
size=640x640&scale=2&maptype=roadmap&language=en-EN
```

```
comparison <- house_sales_all[-ratio,]
MSE <- (all_subset_selection_interaction_prediction - real_value_price)^2/nrow(Validatio
n)
ggmap(map)+
  geom_point(aes(x=long,y=lat,group=zipcode,color=MSE),data=comparison) +
  ggtitle("Mean Squarred Error per Prediction - Subset Selection (Interaction)")
```




```
MSE <- (lasso_prediction - real_value_price)^2/nrow(Validation)
ggmap(map)+
  geom_point(aes(x=long,y=lat,group=zipcode,color=MSE),data=comparison) +
  ggtitle("Mean Squarred Error per Prediction - Lasso Regularization")
```


Mean Squarred Error per Prediction - Lasso Regularization
lat
47.6
47.4
47.2
-122.4
-122.2
-122.0
-121.8
lon
MSE
6e+09
4e+09
2e+09
Seattle
Bellevue
Kirkland
Redmond
Renton
Kent
Auburn
Federal Way
Tacoma
Bainbridge Island
Vashon Island
Des Moines
SeaTac
Burien
Newcastle
Issaquah
Sammamish
Carnation
Duvall
Cottage Lake
Lake Marcel-Stillwater
Fall City
Tokul
Snoqualmie
North Bend
Tanner
Tiger Mountain State Forest
East Renton Highlands
Hobart
Cedar Falls
Maple Valley
Covington
Ravensdale
Selleck
Kangley
Lake Morton-Berrydale
Black Diamond
Cumberland
Lemolo
Krain
Enumclaw
Sumner
Puyallup
Bonney Lake
Buckley
Fife
University Place
Ruston
Harbor
Suquamish
Lakewood
Parkland
Map data ©2018 Google