Dylan Patel

Ave'r McKay

Jianqi Liu

Lab 3

Exercise 1:

Before the exercise, we modified the codes for the functions *rmse* because sometimes the predicted values contain NAs which will then return an NA.

```
#function to get rmse
rmse = function(actual, predicted) {
  i.good = !is.na(predicted)
  return(sqrt(mean((actual[i.good] - predicted[i.good]) ^ 2)))
}
```

For function *get_complexity*, we think it is inaccurate for categorical variables, because it will sum the number of levels (minus 1) for each categorical variable. For example, the following model has only one predictor, and the complexity should be 1, but when we used the function it returned 4. So, we will not use this function.

```
> m = lm(SalePrice ~ MSZoning, Ames)
> get_complexity(m)
[1] 4
```

1. We load the dataset and drop the variables *OverallCond* and *OverallQual* by the following code:

```
Ames = read.csv("ames.csv")
Ames = subset(Ames, select=-c(OverallCond, OverallQual))
```

Besides, based on the summary information, several variables contain large proportion of NAs (among 1460 observations, *Alley* has 1369 NAs, *PoolQC* has 1453 NAs, *Fence* has 1179 NAs, and *MiscFeature* has 1406 NAs), which should also be removed:

```
summary(Ames)
Ames = subset(Ames, select=-c(Alley, PoolQC, Fence, MiscFeature))
```

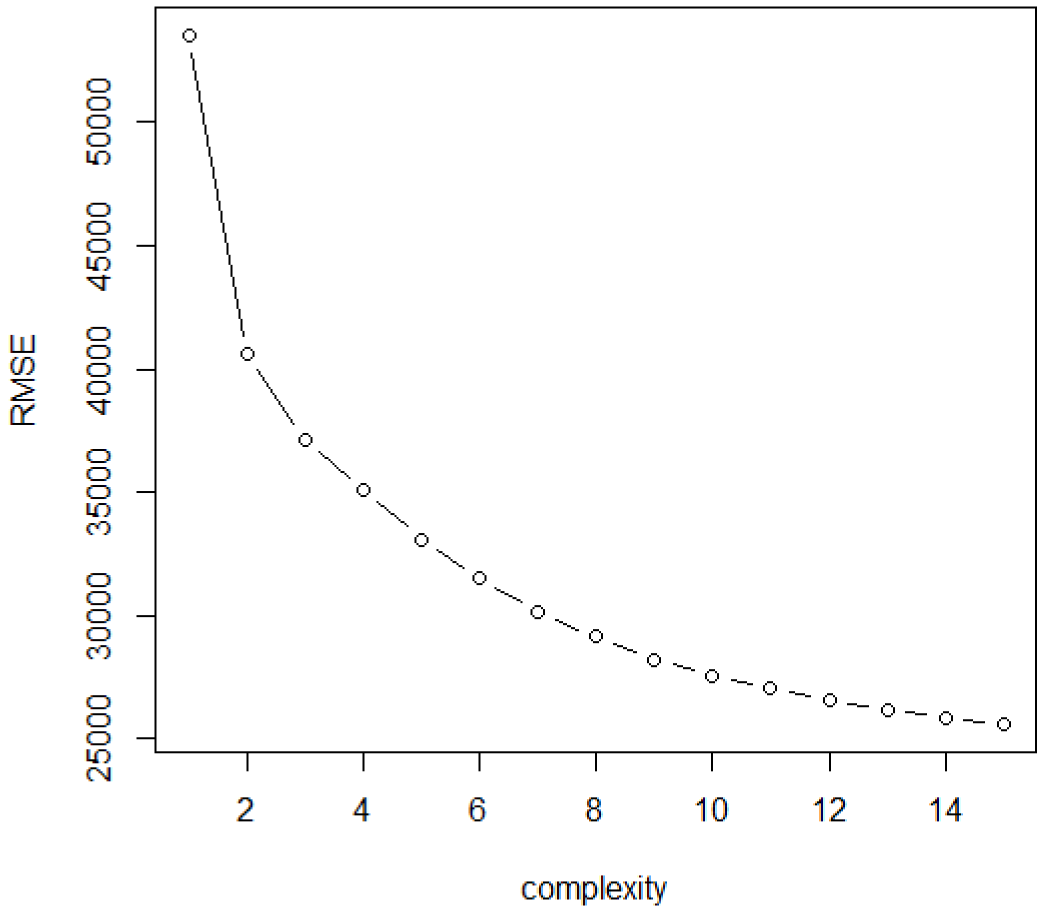
2. R code:

```
name.predictor = names(Ames)[-ncol(Ames)]
Model = list()    #to store the 15 formula
RMSE = rep(0, 15) #vector to store rmse
for(i in 1:15){           #number of predictors in the model
   rmse.each = rep(0, length(name.predictor)) #to store rmse with each addition
   for(j in 1:length(name.predictor)){    #number of reminding variables
      if(i==1){           #set up the first variable
         formula0 = name.predictor[j]
         formula1 = paste0("SalePrice~", formula0)
      }else{formula1 = paste0("SalePrice~", formula0, "+", name.predictor[j])}

      m0 = lm(as.formula(formula1), data=Ames) #linear model
      pred = predict(m0)                #prediction

      if(length(pred) == 1460){
         rmse.each[j] = rmse(Ames$SalePrice, pred)
      }else{rmse.each[j] = rmse(Ames$SalePrice[-m0$na.action], pred)}
   }
   i.best = which.min(rmse.each)    #the index of the best variable
   if(i==1){
      formula0 = name.predictor[i.best]
   }else{formula0 = paste0(formula0, "+", name.predictor[i.best])}
   name.predictor = name.predictor[-i.best]    #update variable names

   model = lm(as.formula(paste0("SalePrice~", formula0)), data=Ames)
   pred = predict(model)
   if(length(pred) == 1460){
      RMSE[i] = rmse(Ames$SalePrice, pred)
   }else{RMSE[i] = rmse(Ames$SalePrice[-model$na.action], pred)}
```

```
   Model[[i]] = model
}
```

3. R code:

```
complexity = 1:15
plot(RMSE ~ complexity, type="b")
```



From the chart, RMSE decreases nonlinearly with complexity. The speed of the decrease is low when complexity is high. However the once the complexity becomes too high the drop off becomes very minimal so we would not want a too of high complexity.

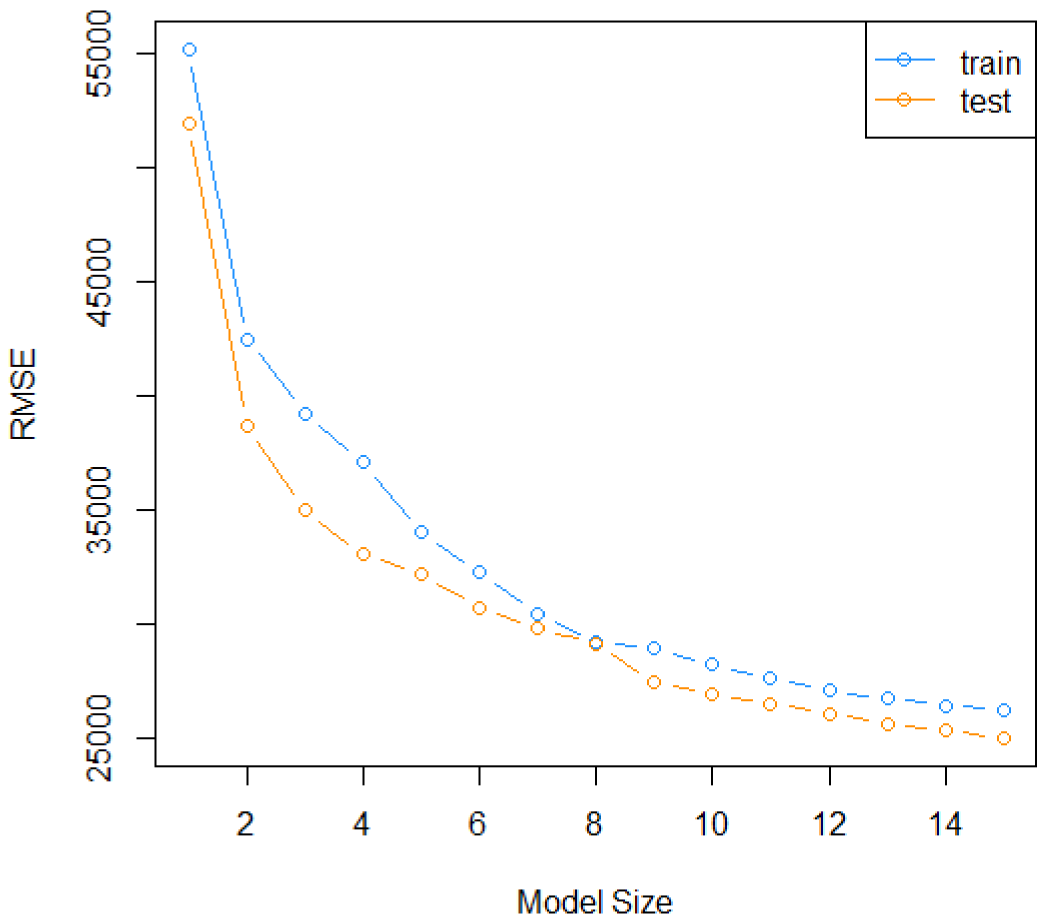
Exercise 2:

Before the exercise, we copy the codes for the function *get_rmse*:

```
get_rmse = function(model, data, response) {
  rmse(actual = subset(data, select = response, drop = TRUE),
       predicted = predict(model, data))
}
```

1. Here we set the initial seed as 9, and split the dataset into 50% as train set and the reminding 50% as test set. Notice that one row (Exterior1st == AsphShn) was deleted in the test set because this level is not contained in the model (maybe because this row contains NA). With this row in the set, it will throw out an error.

```
set.seed(9)
num_obs = nrow(Ames)

train_index = sample(num_obs, size = trunc(0.50 * num_obs))
train_data = Ames[train_index, ]
test_data = Ames[-train_index, ]
test_data = test_data[test_data$Exterior1st!="AsphShn",] #it seems AsphShn does
not appear in the model

train_rmse = sapply(Model, get_rmse, data = train_data, response = "SalePrice")
test_rmse = sapply(Model, get_rmse, data = test_data, response = "SalePrice")

plot(1:15, train_rmse, type = "b",
     ylim = c(min(c(train_rmse, test_rmse)) - 0.02,
              max(c(train_rmse, test_rmse)) + 0.02),
     col = "dodgerblue",
     xlab = "Model Size",
     ylab = "RMSE")
lines(1:15, test_rmse, type = "b", col = "darkorange")
legend("topright", c("train", "test"), pch=1, lty="solid", col=c("dodgerblue",
"darkorange"))
```

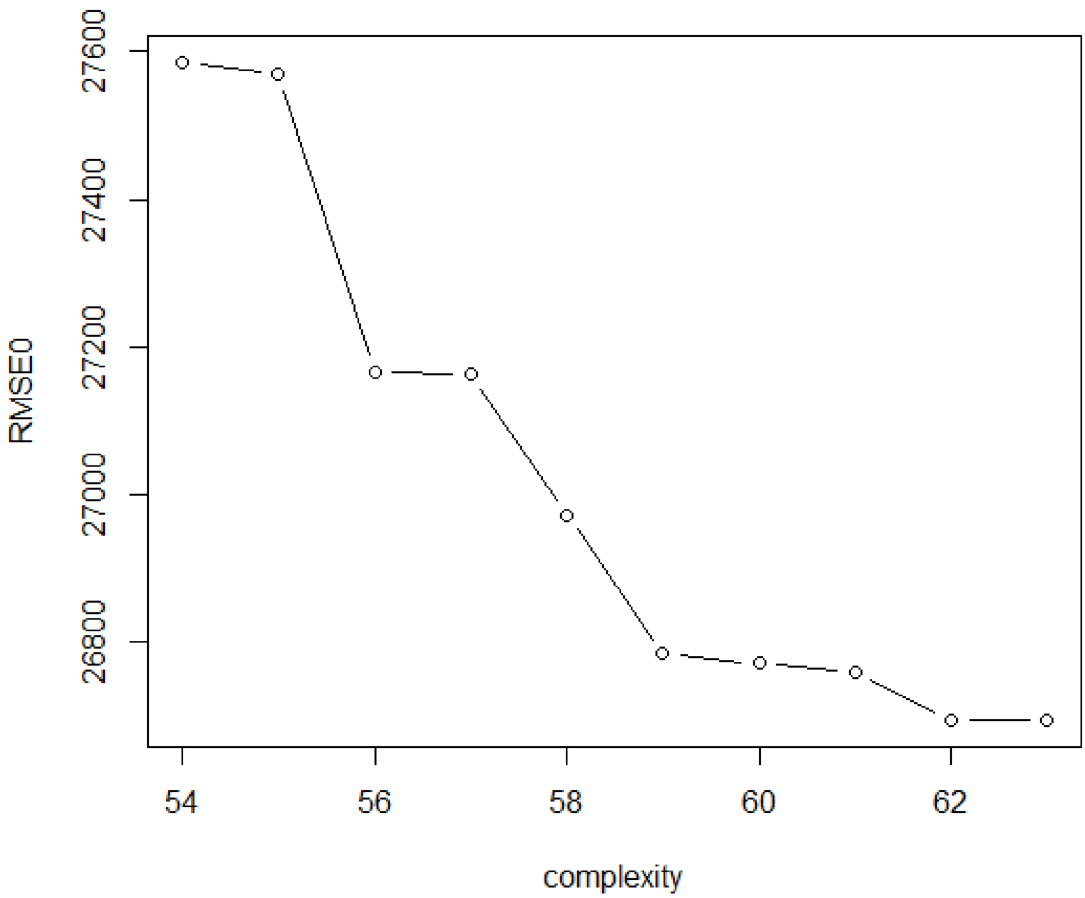


2. Firstly, we try to clean the data by removing predictors which contains many NAs: *LotFrontage* contains 259 NAs and *FireplaceQu* contains 690 NAs (notice that in Exercise 1 we have removed *Alley*, *PoolQC*, *Fence*, and *MiscFeature*, which also contains many NAs). Besides, *Id* should not be used as a predictor, which will also be removed. Then we remove 5 categorical variables which have an extremely uneven distributed (one of the factors has over 1400 observations from all the 1460 observations): *Street, Utilities, Condition2, RoofMatl, Heating*. Next, we removed 3 continuous variables which contain over 1400 zeros: *X3SsnPorch, PoolArea, MiscVal*. Lastly, we remove 122 rows containing NAs. The final clean dataset contains 1338 rows and 64 variables.

```
#remove variables which have many NAs
Ames0 = subset(Ames, select = -c(Id, LotFrontage, FireplaceQu))

#remove categorical variables which is extremely uneven distributed
Ames0 = subset(Ames0, select = -c(Street, Utilities,   Condition2,
                                               RoofMatl, Heating))

#remove continous variables which contain many 0
n.zeros =   rep(0, 67)
names(n.zeros) = names(Ames0)
for(i in 1:67){n.zeros[i] = sum(Ames0[,i]==0, na.rm=T)}
n.zeros
Ames0 = subset(Ames0, select = -c(X3SsnPorch, PoolArea, MiscVal))

#remove NAs
Ames0 = Ames0[complete.cases(Ames0), ]
```

Then we use backward stepwise method to get model with lowest RMSE.

```
name.predictor = names(Ames0)[-ncol(Ames0)]
Model0 = list()   #to store the 15 models
RMSE0 = rep(0, 10) #vector to store rmse
for(i in 1:10){          #number of predictors in the model
  formula1 = paste0("SalePrice~", paste(name.predictor, collapse="+"))
  m0 = lm(as.formula(formula1), data=Ames0) #linear model
  z = anova(m0)
  i.largest = which.max(z$`Pr(>F)`)
  name.predictor = name.predictor[-i.largest]   #update variable names

  model = lm(as.formula(paste0("SalePrice~", paste(name.predictor,
collapse="+"))), data=Ames)
  pred = predict(model)
  RMSE0[i] = rmse(Ames0$SalePrice, pred)

  Model0[[i]] = model
}

plot(63:54, RMSE0, type="b", xlab="complexity")
```



Based on the figure, the model with a complexity of 62 should be good enough. Lastly we calculate the train and test RMSE.

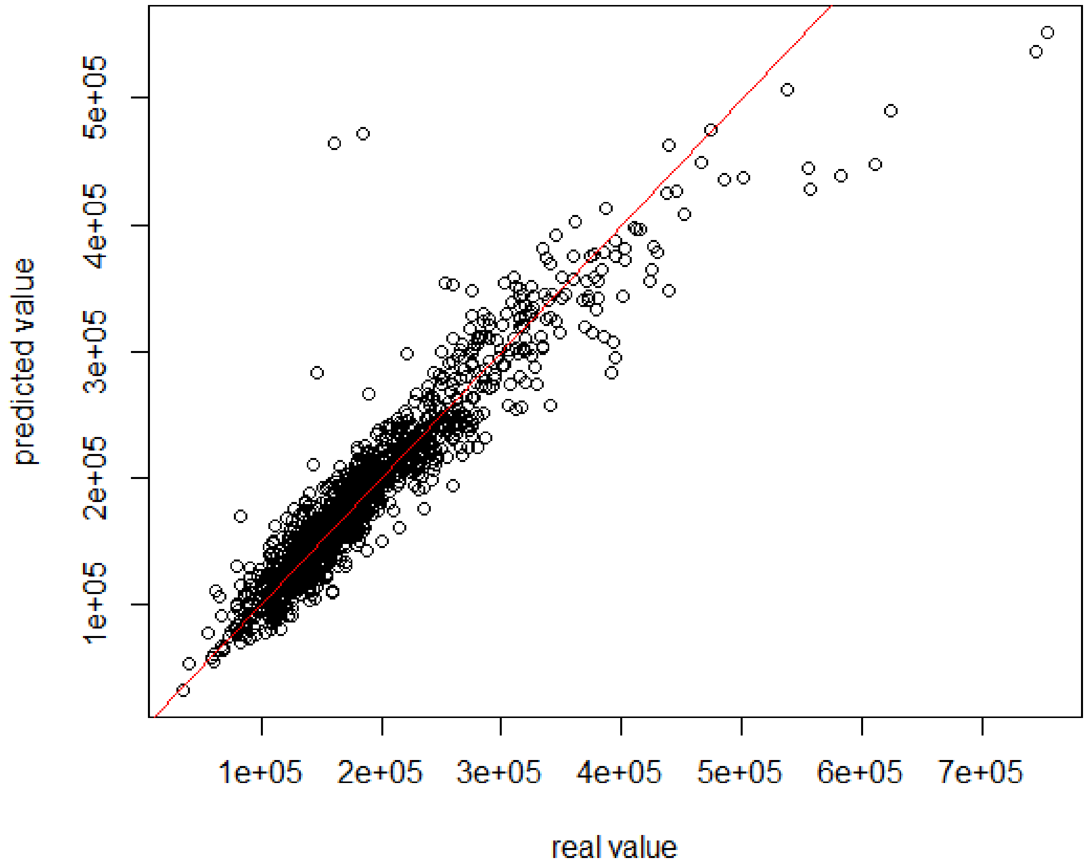
```
set.seed(9)
num_obs = nrow(Ames0)
train_index = sample(num_obs, size = trunc(0.50 * num_obs))
train_data0 = Ames0[train_index, ]
test_data0 = Ames0[-train_index, ]

model.final = Model0[[2]]
rmse.train = get_rmse(model.final, data = train_data0, response = "SalePrice")
rmse.test = get_rmse(model.final, data = test_data0, response = "SalePrice")
```

The train RMSE is 26444 and the test RMSE is 26944.

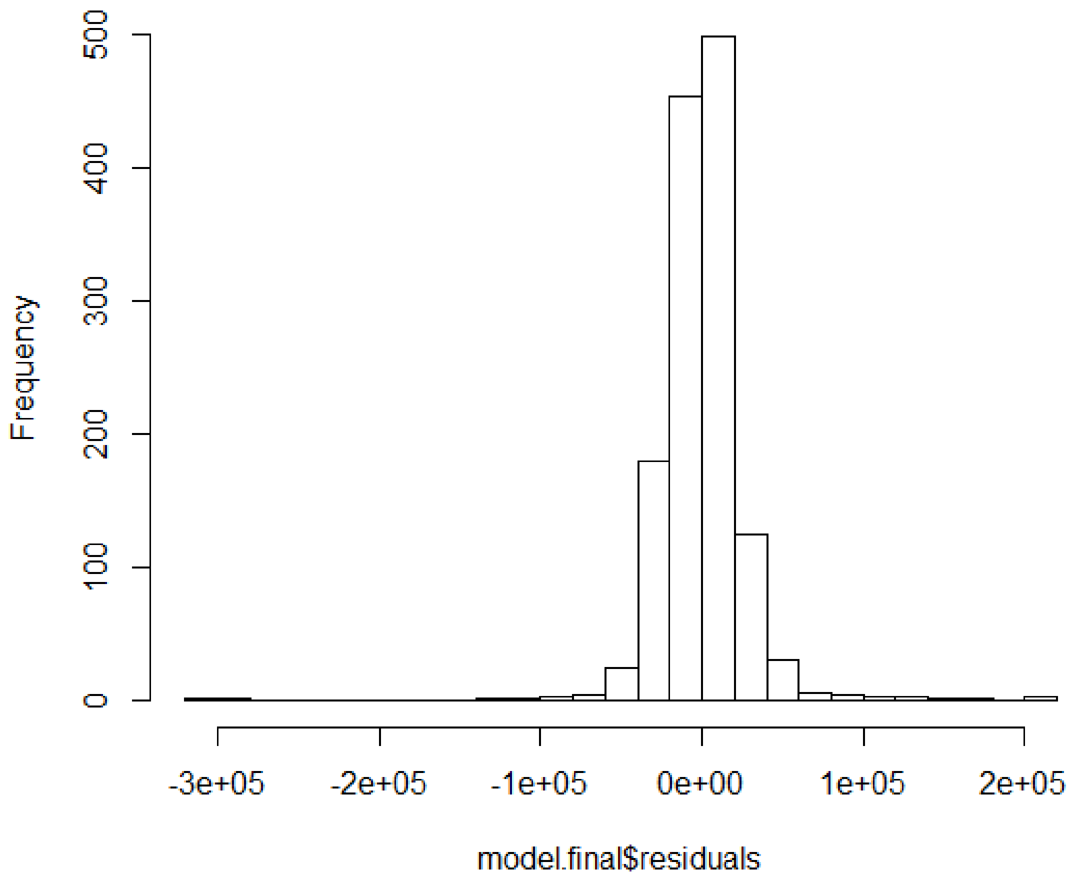
3. The final model contains 62 predictors: *MSSubClass, MSZoning, LotArea, LotShape, LandContour, LotConfig, LandSlope, Neighborhood, Condition1,*

BldgType, HouseStyle, YearBuilt, YearRemodAdd, RoofStyle, Exterior1st, Exterior2nd, MasVnrType, MasVnrArea, ExterQual, ExterCond, Foundation, BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinSF1, BsmtFinType2, BsmtFinSF2, BsmtUnfSF, HeatingQC, CentralAir, Electrical, X1stFlrSF, X2ndFlrSF, LowQualFinSF, BsmtFullBath, BsmtHalfBath, FullBath, HalfBath, BedroomAbvGr, KitchenAbvGr, KitchenQual, TotRmsAbvGrd, Functional, Fireplaces, GarageType, GarageYrBlt, GarageFinish, GarageCars, GarageArea, GarageQual, GarageCond, PavedDrive, OpenPorchSF, EnclosedPorch, ScreenPorch, MoSold, YrSold, SaleType, and *SaleCondition*. We arrived at this model from the full model and removed the variable which has the largest p-value. We did not consider the interactions because there are C(64,2) = 2016 combinations which are very hard to choose. Because of this reason, we may not get the best model with the lowest RMSE.

4. We showed the relationship between the real and the predicted values of *SalePrice*, with a one-one line. It can be seen that there are some outliers which may contribute a lot to the RMSE. Besides, when the values are high, the predictions generally are lower than the real data. From the histogram of the residuals, it is generally symmetric, but there are some extreme low and high values which reflects the outliers.



predicted value
1e+05
2e+05
3e+05
4e+05
5e+05
1e+05
2e+05
3e+05
4e+05
5e+05
6e+05
7e+05
real value



Frequency
0
100
200
300
400
500
-3e+05
-2e+05
-1e+05
0e+00
1e+05
2e+05
model.final$residuals