Codes from Lab3

```
rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
get_complexity = function(model) {
  length(coef(model)) - 1
}
get_rmse = function(model, data, response) {
  rmse(actual = subset(data, select = response, drop = TRUE),
       predicted = predict(model, data))
}
```

Excercise 1

1

```
ames <- read.csv("ames.csv",header = TRUE)


ames <- ames[,c(1:17,20:81)] # Drop the variables OverallCond and OverallQual.
```

2.

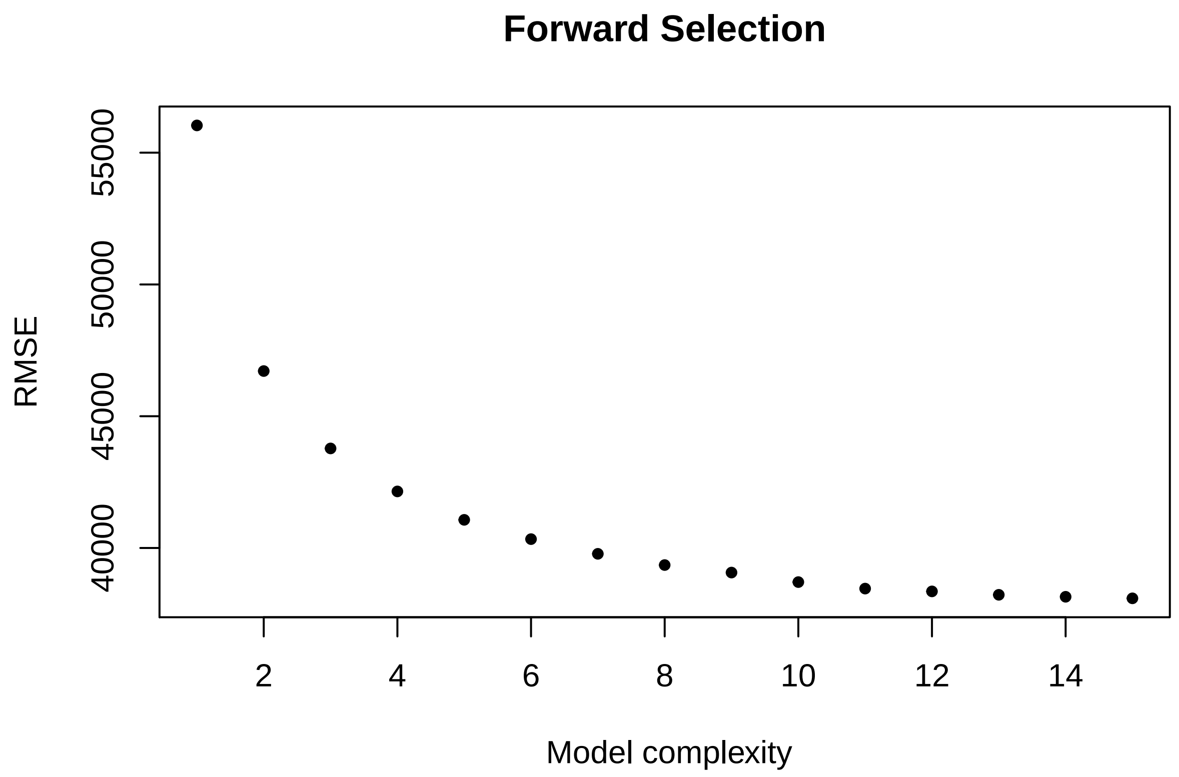
```
Forward_selcetion <- function(p,data=ames){
  # initial values
  RMSEs <- c()
  Predictors <- c()
  NAMES <- names(data)[2:78]
  # get the RMSE of null model
  RMSE <- rmse(actual = data$SalePrice,
               lm(SalePrice~1,data = data)$fitted.values)
  for (i in 1:p) {
    # forward selection
    for (v in NAMES ) {
      if (typeof(as.vector(ames[,v])) == "integer") {
        try(fit <- lm(SalePrice~.,
                      data = data[,c("SalePrice",Predictors,v)]),TRUE)
        RMSE1 <- rmse(actual = data$SalePrice,fit$fitted.values)
        if (RMSE1 < RMSE) {
          new_predictor <- v
          RMSE = RMSE1
        }
      }
    }
    NAMES <- NAMES[! NAMES %in% Predictors]
    RMSEs <- c(RMSEs,RMSE)
    Predictors <- c(Predictors,new_predictor) # add new predictor
  }
  Results <- list(Predictors,RMSEs)
  return(Results)
}
Forward_selcetion(15)
```

```
## [[1]]
##  [1] "GrLivArea"    "YearBuilt"    "TotalBsmtSF"  "GarageCars"  
##  [5] "KitchenAbvGr" "YearRemodAdd" "BsmtFinSF1"   "Fireplaces"  
##  [9] "BedroomAbvGr" "TotRmsAbvGrd" "MSSubClass"   "WoodDeckSF"  
## [13] "ScreenPorch"  "LotArea"      "BsmtFullBath"
## 
## [[2]]
##  [1] 56034.30 46714.14 43777.51 42146.77 41067.28 40335.73 39778.91
##  [8] 39352.56 39067.03 38706.32 38458.84 38353.09 38223.09 38148.11
## [15] 38090.09
```

```r
attach(ames)
fit1 <- lm(SalePrice~GrLivArea)
fit2 <- lm(SalePrice~GrLivArea+YearBuilt)
fit3 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF)
fit4 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars)
fit5 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
           +KitchenAbvGr)
fit6 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
           +KitchenAbvGr+YearRemodAdd)
fit7 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
           +KitchenAbvGr+YearRemodAdd+BsmtFinSF1)
fit8 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
           +KitchenAbvGr+YearRemodAdd+BsmtFinSF1+Fireplaces)
fit9 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
           +KitchenAbvGr+YearRemodAdd+BsmtFinSF1+Fireplaces+
             BedroomAbvGr)
fit10 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
            +KitchenAbvGr+YearRemodAdd+BsmtFinSF1+Fireplaces+
              BedroomAbvGr+TotRmsAbvGrd)
fit11 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
            +KitchenAbvGr+YearRemodAdd+BsmtFinSF1+Fireplaces+
              BedroomAbvGr+TotRmsAbvGrd+MSSubClass)
fit12 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
            +KitchenAbvGr+YearRemodAdd+BsmtFinSF1+Fireplaces+
              BedroomAbvGr+TotRmsAbvGrd+MSSubClass+WoodDeckSF)
fit13 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
            +KitchenAbvGr+YearRemodAdd+BsmtFinSF1+Fireplaces+
              BedroomAbvGr+TotRmsAbvGrd+MSSubClass+WoodDeckSF+ScreenPorch)
fit14 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
            +KitchenAbvGr+YearRemodAdd+BsmtFinSF1+Fireplaces+
              BedroomAbvGr+TotRmsAbvGrd+MSSubClass+WoodDeckSF+ScreenPorch+
              LotArea)
fit15 <- lm(SalePrice~GrLivArea+YearBuilt+TotalBsmtSF+GarageCars
            +KitchenAbvGr+YearRemodAdd+BsmtFinSF1+Fireplaces+
              BedroomAbvGr+TotRmsAbvGrd+MSSubClass+WoodDeckSF+ScreenPorch+
              LotArea+BsmtFullBath)

model_list <- list(fit1,fit2,fit3,fit4,fit5,
                   fit6,fit7,fit8,fit9,fit10,
                   fit11,fit12,fit13,fit14,fit15)
```

3.

```r
RMSEs = sapply(model_list, get_rmse,
               data = ames, response = "SalePrice")
model_complexity = sapply(model_list, get_complexity)
plot(model_complexity,RMSEs,xlab = " Model complexity",
     ylab = "RMSE",pch = 20,main = "Forward Selection")
```

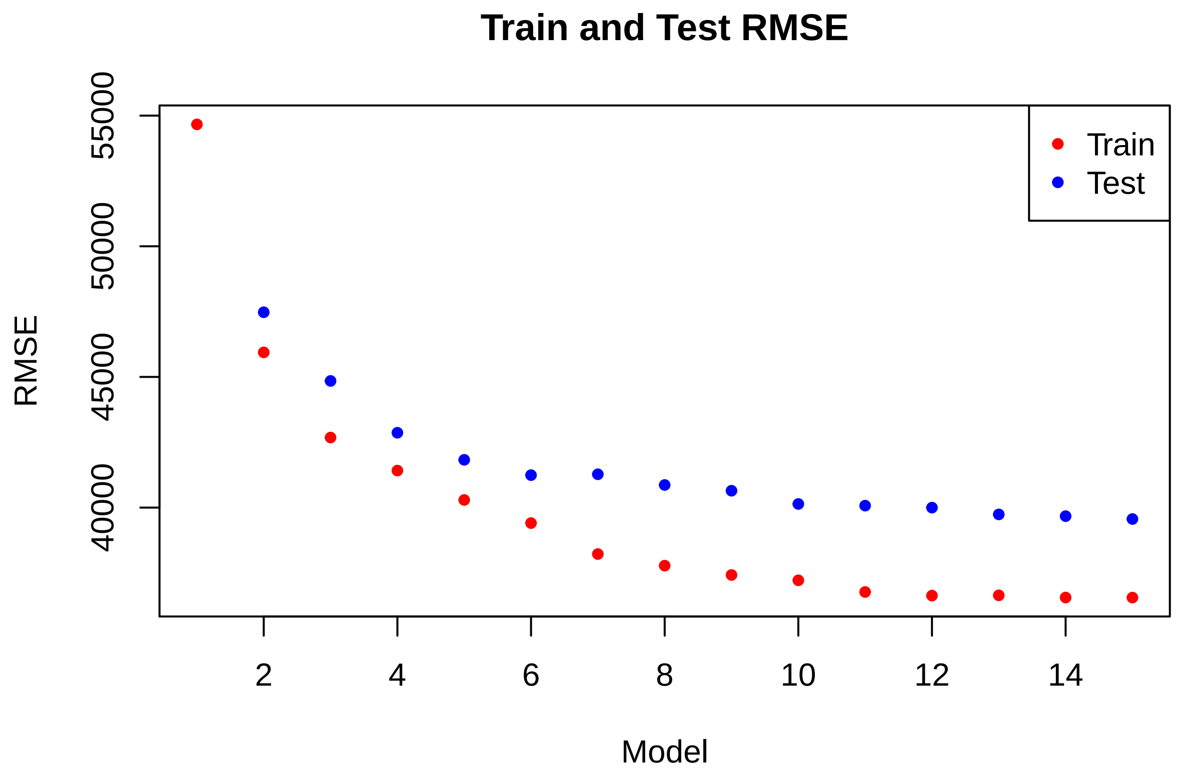


The RMSE of models decrease as the model complexity increase. We should not use the full-size model. There are two main reasons: Multicollinearity and Overfitting. First, Multicollinearity may cause the estimation parameters to be inconsistent with the practical significance Second, Overfitting is the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably

Excercise 2

1.

```r
set.seed(4)
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, ]

train_rmse = sapply(model_list, get_rmse,
                    data = train_data, response = "SalePrice")
test_rmse = sapply(model_list, get_rmse,
                   data = test_data, response = "SalePrice")
model_complexity = sapply(model_list, get_complexity)

plot(train_rmse,pch=20,col = "red",xlab = "Model",
     ylab = "RMSE",main = "Train and Test RMSE")
points(test_rmse,pch=20,col = "blue")
legend("topright",legend = c("Train","Test"),
       pch = 20,col = c("red","blue"))
```

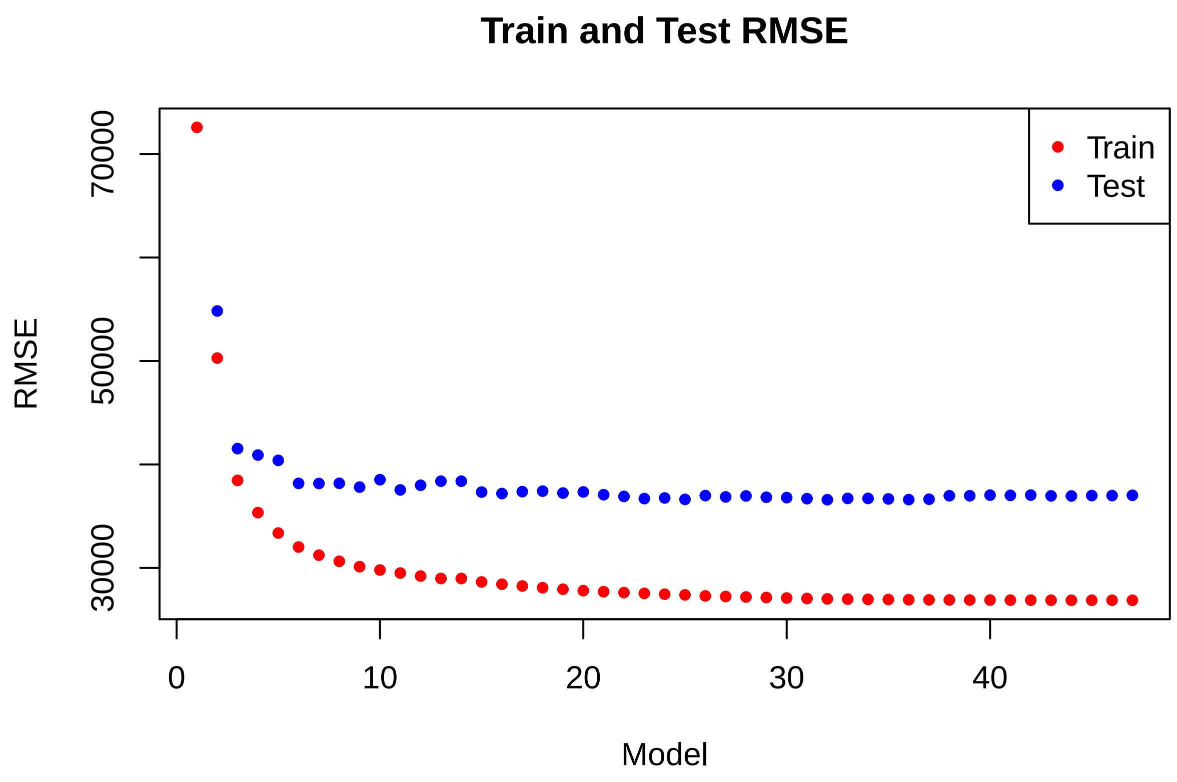


2-4

```
Train_model <- function(Predictors,v){
  fit_train <- lm(SalePrice~.,
                  data = train_data[,c("SalePrice",Predictors,v)])
  RMSE1 <- rmse(actual = train_data$SalePrice,
                fit_train$fitted.values)
  RMSE2 <- get_rmse(model = fit_train,
                    data=test_data[,c("SalePrice",Predictors,v)],
                    response = "SalePrice")
  structure(list(RMSE=c(RMSE1,RMSE2),V = c(v)))
}

Selcet_Model <- function(data){
  NAMES <- names(data)
  rms <- c()
  for (i in 1:79) {
    if(sum(is.na(data[,i]))>=1){
      rms <- c(rms,i)
    }
  }
  NAMES <- NAMES[-rms]
  data = data[, NAMES]

  num_obs = nrow(data)
  train_index = sample(num_obs, size = trunc(0.50 * num_obs))

  train_data = data[train_index, NAMES]
  test_data = data[-train_index, NAMES]
  # initial values
  Train_RMSEs <- c()
  Test_RMSEs <- c()
  Predictors <- c()

  # get the RMSE of null model
  RMSE <- rmse(actual = train_data$SalePrice,
               lm(SalePrice~1,data = train_data)$fitted.values)
  for (i in NAMES[2:(length(NAMES)-1)]) {
    # forward selection
    for (v in NAMES[2:(length(NAMES)-1)] ) {
      try(Results <- Train_model(Predictors,v),silent = TRUE)
      RMSE1 <- Results$RMSE[1]
      RMSE2 <- Results$RMSE[2]
      if (RMSE1 < RMSE) {
        new_predictor <- Results$V
        RMSE = RMSE1
      }
    }
    NAMES <- NAMES[! NAMES %in% Predictors]
    Train_RMSEs <- c(Train_RMSEs,RMSE1)
    Test_RMSEs <- c(Test_RMSEs,RMSE2)
    if(! new_predictor %in% Predictors){
      Predictors = c(Predictors,new_predictor) # add new predictor
    }else{break}
  }
  structure(list(Predictors=Predictors,Train_RMSEs=Train_RMSEs,
                 Test_RMSEs=Test_RMSEs))
}

PR <- list()
TrRMSEs <- list()
TeRMSEs <- list()
MODELS <- list()

for (i in 1:100) {
  Select_results <- Selcet_Model(ames)
  MODELS[[i]] <- Select_results
  PR[[i]] <- Select_results$Predictors
  TrRMSEs[[i]] <- Select_results$Train_RMSEs
  TeRMSEs[[i]]<- Select_results$Test_RMSEs
}
```

The plot shows below is the RSME for train set and test set from one of trainings.

```
plot(Select_results$Train_RMSEs,pch=20,col = "red",xlab = "Model",
     ylab = "RMSE",main = "Train and Test RMSE")
points(Select_results$Test_RMSEs,pch=20,col = "blue")
legend("topright",legend = c("Train","Test"),
       pch = 20,col = c("red","blue"))
```



The forward selection method is used to select the model, and the predictors of the model, the RMSE of the training set and the RMSE of the test set are recorded.

We use forward selection to filter out all the predictors that may make the RMSE of the test set the lowest. Each training model will change the data of training set and test set, so each training will generate a group of predictors. After 100 simulations, take the common predictors of all training model results, and then we test the robustness of the model through 1000 simulations, and get the train RMSE and test RMSE of each simulation as shown in the figure below.

```
plot_RMSE <- function(train,test){
  plot(train,pch=20,col = "red",xlab = "Model",
       ylab = "RMSE",main = "Train and Test RMSE",
       ylim = c(20000,80000))
  par(new=T)
  points(test,pch=20,col = "blue")
  legend("topright",legend = c("Train","Test"),
         pch = 20,col = c("red","blue"))
}
```

```
temp <- names(ames)
for (i in 1:100) {
  temp <- temp[temp %in% PR[[i]][1:which.min(TeRMSEs[[i]])]]
}
temp
```

```
##  [1] "MSZoning"     "LotArea"      "Street"       "LandContour"
##  [5] "LotConfig"    "LandSlope"    "Neighborhood" "BldgType"
##  [9] "HouseStyle"   "YearBuilt"    "YearRemodAdd" "RoofStyle"
## [13] "ExterQual"    "Foundation"   "BsmtFinSF1"   "TotalBsmtSF"
## [17] "X1stFlrSF"    "LowQualFinSF" "GrLivArea"    "BsmtFullBath"
## [21] "BedroomAbvGr" "KitchenAbvGr" "KitchenQual"  "TotRmsAbvGrd"
## [25] "Fireplaces"   "GarageCars"   "WoodDeckSF"   "EnclosedPorch"
## [29] "ScreenPorch"  "PoolArea"     "SaleType"     "SaleCondition"
```

```
RMSE_train <- rep(0,1000)
RMSE_test <- rep(0,1000)
for (i in 1:1000) {
  train_index = sample(num_obs, size = trunc(0.50 * num_obs))
  train_data = ames[train_index, ]
  test_data = ames[-train_index, ]
  try(fit1 <- lm(SalePrice~.,data = train_data[,c(temp,"SalePrice")]),silent = TRUE)
  RMSE_train[i] <- rmse(actual = train_data$SalePrice,
                   fit1$fitted.values)
  try(RMSE_test[i] <- get_rmse(fit1,test_data[,c(temp,"SalePrice")],response = "SalePrice"),silent = T)
}
get_complexity(fit1)
```

```
## [1] 95
```

This model does not limit the complexity of the model, so after 100 times of model training through forward selection method, the predictors of our final model are “MSZoning”, “LotArea”, “Street”, “LandContour”, “LotConfig”, “LandSlope”, “Neighborhood”, “BldgType”, “HouseStyle”, “YearBuilt”, “YearRemodAdd”, “RoofStyle”, “ExterQual”, “Foundation”, “BsmtFinSF1”, “TotalBsmtSF”, “X1stFlrSF”, “LowQualFinSF”, “GrLivArea”, “BsmtFullBath”, “BedroomAbvGr”, “KitchenAbvGr”, “KitchenQual”, “TotRmsAbvGrd”, “Fireplaces”, “GarageCars”, “WoodDeckSF”, “EnclosedPorch”, “ScreenPorch”, “PoolArea”, “SaleType” and “SaleCondition”, and the model complexity is 95.

```
par(mfrow=c(1,2))
plot(RMSE_train,ylim = c(min(RMSE_train)-0.5e4,max(RMSE_train)+0.5e4))
abline(h=mean(RMSE_train[RMSE_train<50000]),pch=20,col = "red",main = "Simmulation for 1000 times for t
mean(RMSE_train[RMSE_train<50000])
```

```
## [1] 27285.11
```

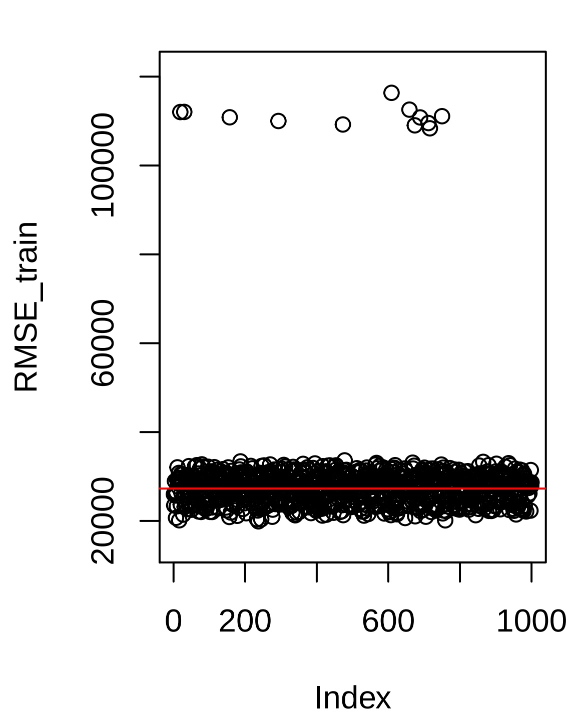
```
max(RMSE_train[RMSE_train<50000])
```

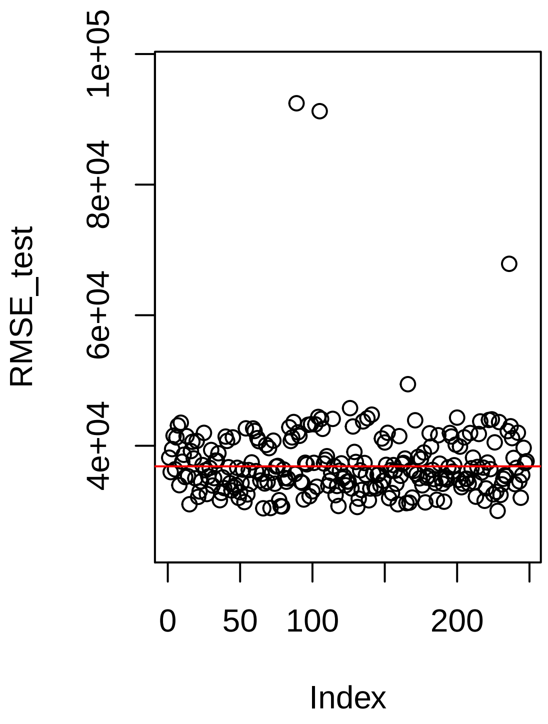
```
## [1] 33630.12
```

```
min(RMSE_train[RMSE_train<50000])
```

```
## [1] 19917.23
```

```
RMSE_test <- RMSE_test[RMSE_test!=0]
plot(RMSE_test,ylim = c(min(RMSE_test)-0.5e4,max(RMSE_test)+0.5e4))
abline(h=mean(RMSE_test[RMSE_test<50000]),pch=20,col = "red",main = "Simmulation for 1000 times for tes
```





```
mean(RMSE_test[RMSE_test<50000])
```

```
## [1] 36851.61
```

```
max(RMSE_test[RMSE_test<50000])
```

```
## [1] 49434.28
```

```
min(RMSE_test[RMSE_test<50000])
```

```
## [1] 30029.39
```

From the figure, we can see that the RMSE of train and test are distributed in a range with littel extreme values, which shows that the model is very stable for different training sets and test sets.

As we know the train set and test are generated randomly from the original data, so we simulate 1000 times to calculate the mean of Train RMSE (outliers are excluded) is 27285.11 and the mean of Test RMSE (outliers are excluded) is 36851.61, the maximum RMSE (outliers are excluded) for train set and test set are 33630.12, 49434.28 respectively and the minimum RMSE (outliers are excluded) for train set and test set are 19917.23, 30029.39 respectively.