Question 1

- lm:

```
lm(Y ~ .,data = train_df)
```

- Stepwise:

```
initial <- lm(formula = Y ~ 1,data = train\_df)
final <- lm(formula = Y ~ .,data = train\_df)
step(object=initial, scope=list(upper=final))
```

- Ridge:

```
lm.ridge(Y ~ ., lambda = seq(0,100,.05), data=train_df)
```

- Lasso:

```
cv.glmnet(x=train_matrix[,-1], y = train_df[,'Y'])
```

- PLS:

```
plsr(Y ~ ., data = train_df, validation = "CV")
```

- Random Forest:

```
randomForest(Y ~ .,data=train_df,mtry = 11,
             nodesize = 15,ntree = 2000)
```

- Boosting:

```
gbm(Y ~ ., data =train_df, distribution = "gaussian",
            n.trees = 10000, interaction.depth = 8,
            shrinkage = 0.01)
```

- Neural Network:

```
nnet(y = Y.train, x = X.train, linout = TRUE, size = 31,
        decay = 0.8, maxit = 500)
```

Question 2

We used 5 replicates of 10-folds CV on each model and computed the corresponding MSPEs. By doing a boxplot on the relative MSPE, we chose our final Boosting model because it outperformed other models almost all of the time.

Question 3

(a) Random forest, Boosting and Neutral Net all have tuning parameters. We used 5 replicates of 10-folds CV to tune each of them on a grid of the parameters. Then compare the models based on their computed RMSPEs.

(b)

- Random forest: we tuned on the combination of ntree = (500,1000,1500,2000), mtry = 3,4,5,...,12, and nodesize = 2,3,4,...,21
- Boosting: we tuned on the combination of shrinkage = (0.0001,0.001,0.01,0.1), interaction.depth=(3,4,5,6,7,8,9,10), and ntrees = (2000,5000,10000,150000)
- Neural Net: we tuned on the combinations of size = 1,2,3,...,200 and decay = 0,0.1,0.2,...,1

Question 4

We choose Boosting with tuned parameter as our prediction machine.

```
library(gbm)
# Read in test and train data
rm(list = ls())
train_df = read.csv("Data/Data2020.csv")
test_df = read.csv("Data/Data2020testX.csv")


set.seed(1)
gbm.model = gbm(Y ~ ., data =train_df, distribution = "gaussian",
                n.trees = 10000, interaction.depth = 8, shrinkage = 0.01)
n.trees.best = gbm.perf(gbm.model, plot.it = F) * 2 # Number of trees
prediction = predict(gbm.model, test_df, n.trees.best)
write.table(prediction, "Final/prediction.csv", sep = ",",
            row.names = F, col.names =F)
```

1 Question 5(Bonus)

We think X2,X4,X8,X12 are important