ML-Project

Yiwen Shi

8/2/2020

Load Data

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
#training set
train <- read.csv("pml-training.csv", stringsAsFactors = F)
#test set
test <- read.csv("pml-testing.csv",stringsAsFactors = )
```

Pre-process

```
#remove columns with missing values > 10 %
count_na = nrow(train) * 0.1
cols_rm <- which(colSums(is.na(train) | train==""|train == "#DIV/0!") > count_na)
train = train[,-cols_rm]
test = test[,-cols_rm]
#remove timestamps
train <- train[,-(1:5) ]
test <- test[,-(1:5) ]
train_labels = as.factor(train$classe)
test_labels = as.factor(test$classe)
#near zero variance
nzvs <- nearZeroVar(train, saveMetrics = TRUE)
train <- train[, !nzvs$nzv]
test <- test[, !nzvs$nzv]
```

EDA

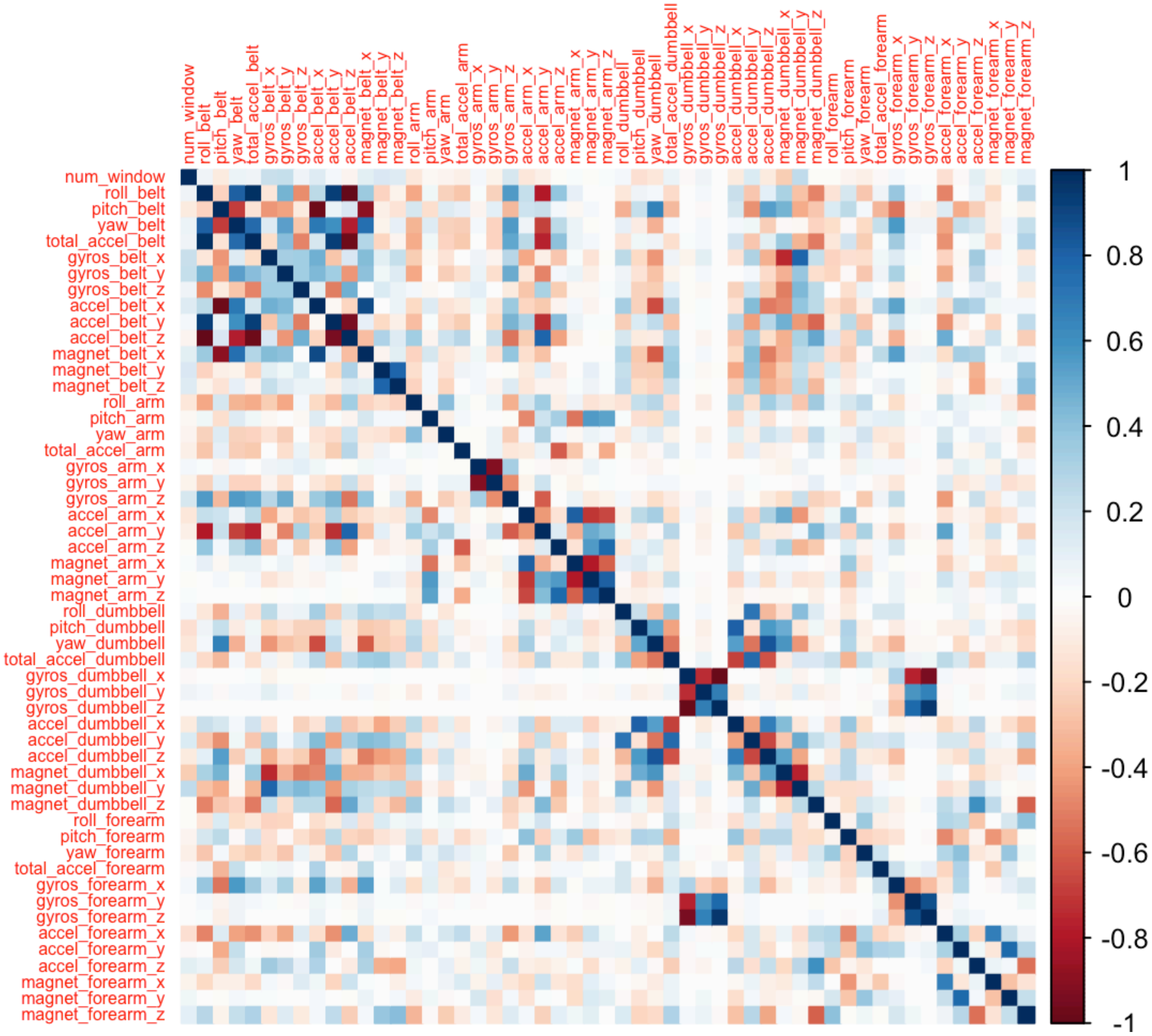
Creat data partitions first, then analyze training set

```
set.seed(123)
part_inds <- createDataPartition(y=train_labels, p=0.7, list=FALSE)
train_set <- train[part_inds, ]
test_set <- train[-part_inds, ]
train_set_labels = train_labels[part_inds]
test_set_labels = train_labels[-part_inds]
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
#visualize correlations among potential predictor variables
corrplot(cor(train_set[,-length(colnames(train_set))]), method = "color", tl.cex = 0.
5)
```



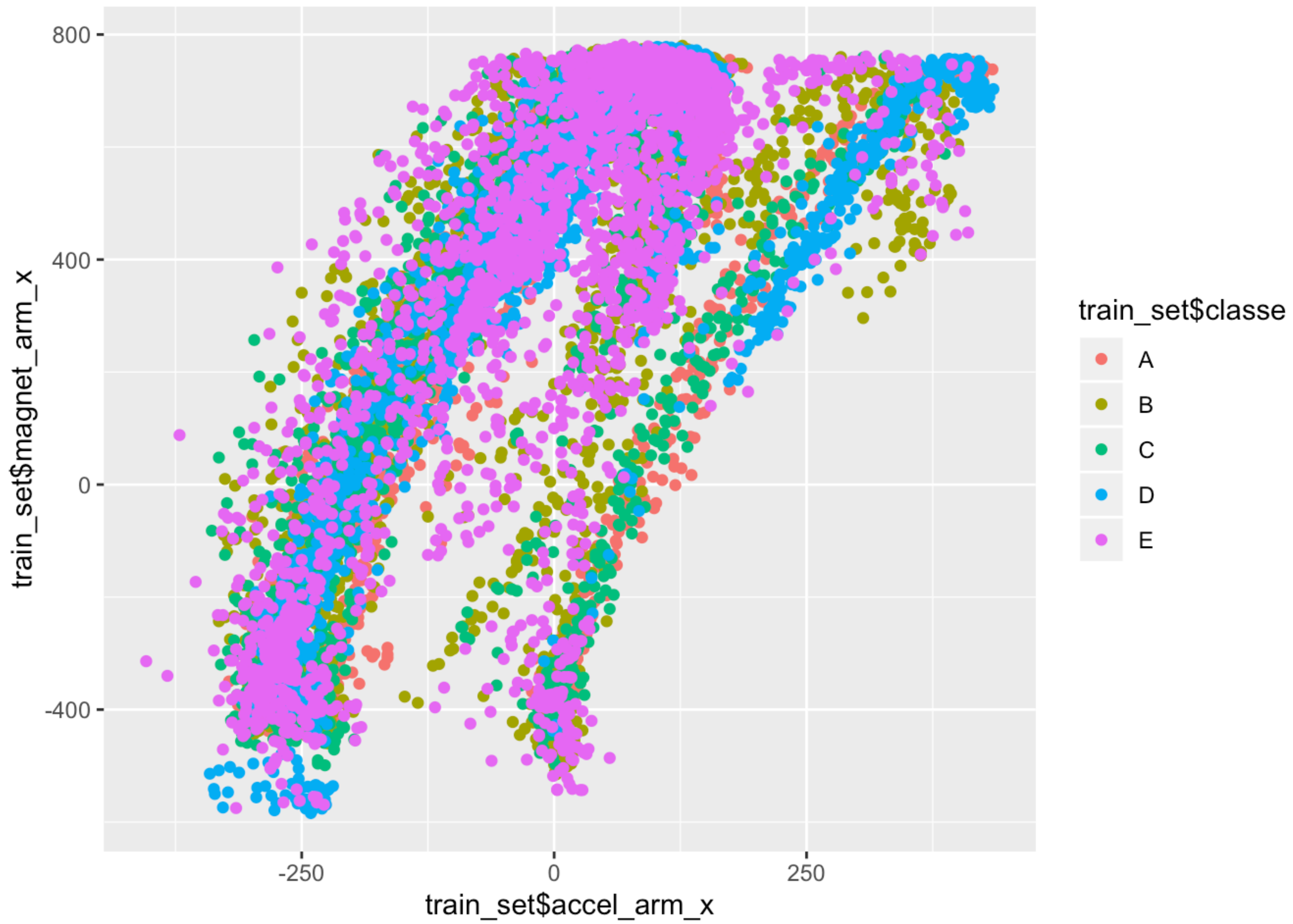

```
cor_df = as.data.frame(cor(train_set[,-length(colnames(train_set))]))
cor_label = as.data.frame(cor(train_set[,-length(colnames(train_set))], as.numeric(tr
ain_set_labels)))
```

Find some variables with higher correlations

```r
high_cor_labels = rownames(cor_label)[abs(cor_label) >= 0.2]
high_cor_values = cor_label[abs(cor_label) >= 0.2]
high_cor_matrix = cor_df[high_cor_labels,high_cor_labels]
#some of these variables are highly correlated with each other
high_cor_matrix
```

```
##                magnet_belt_y accel_arm_x magnet_arm_x magnet_arm_y
## magnet_belt_y    1.00000000  -0.1077090   0.01976657   0.09192562
## accel_arm_x     -0.10770904   1.0000000   0.81690635  -0.70116255
## magnet_arm_x     0.01976657   0.8169063   1.00000000  -0.79148314
## magnet_arm_y     0.09192562  -0.7011625  -0.79148314   1.00000000
## pitch_forearm   -0.13050697   0.3909852   0.35653526  -0.28606247
##                pitch_forearm
## magnet_belt_y    -0.1305070
## accel_arm_x       0.3909852
## magnet_arm_x      0.3565353
## magnet_arm_y     -0.2860625
## pitch_forearm     1.0000000
```

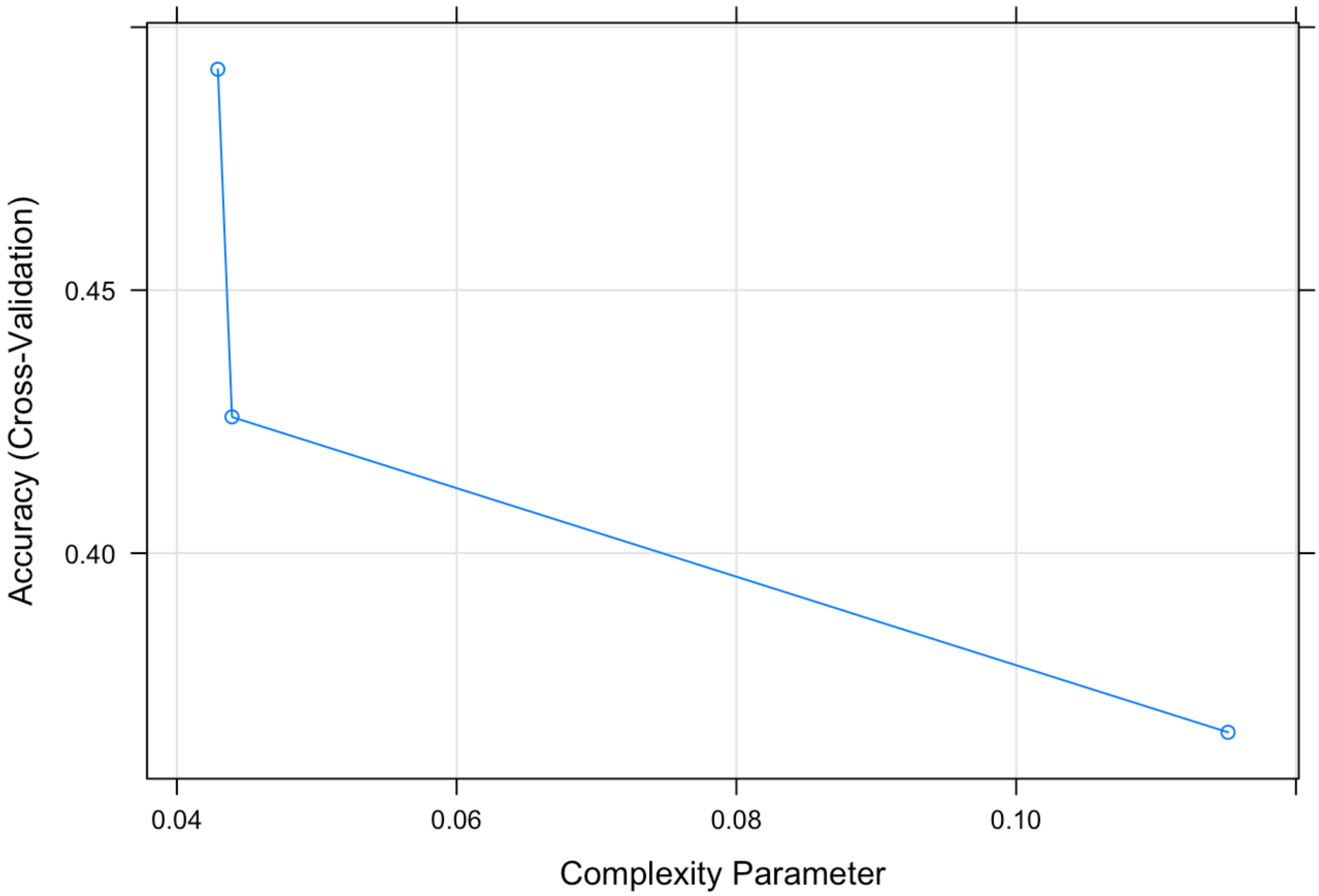
```r
library(ggplot2)
qplot(train_set$accel_arm_x, train_set$magnet_arm_x, colour=train_set$classe)
```



Modeling

Use all variables, fit a tree since multi-class, with cross validation

```
train_set$classe = as.factor(train_set$classe)
test_set$classe = as.factor(test_set$classe)
tree = train(classe ~ .,
                  data=train_set,
                  method="rpart",
                  trControl = trainControl(method = "cv"))
library(rpart)
plot(tree)
```




```
predictTree <- predict(tree, test_set,type = "raw")
confusionMatrix(test_set$classe, predictTree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1496   38  138    0    2
##          B  474  390  275    0    0
##          C  166   50  810    0    0
##          D  317  137  471    0   39
##          E   59  190  178    0  655
##
## Overall Statistics
##
##                Accuracy : 0.5694
##                  95% CI : (0.5566, 0.5821)
##     No Information Rate : 0.4268
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4443
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5955  0.48447   0.4327       NA   0.9411
## Specificity            0.9472  0.85256   0.9462   0.8362   0.9177
## Pos Pred Value         0.8937  0.34241   0.7895       NA   0.6054
## Neg Pred Value         0.7587  0.91256   0.7814       NA   0.9915
## Prevalence             0.4268  0.13679   0.3181   0.0000   0.1183
## Detection Rate         0.2542  0.06627   0.1376   0.0000   0.1113
## Detection Prevalence   0.2845  0.19354   0.1743   0.1638   0.1839
## Balanced Accuracy      0.7714  0.66852   0.6894       NA   0.9294
```

```
forest = train(classe ~ ., data = train_set, method = "rf", trControl = trainControl(
method = "cv", 5), ntree = 100)
forest
```

```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10990, 10988, 10990, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9919929  0.9898706
##   27    0.9970883  0.9963170
##   53    0.9949043  0.9935541
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
predictRF <- predict(forest, test_set,type = "raw")
confusionMatrix(test_set$classe, predictRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    1 1136    2    0    0
##          C    0    2 1024    0    0
##          D    0    0    1  963    0
##          E    0    0    1    4 1077
##
## Overall Statistics
##
##                Accuracy : 0.9981
##                  95% CI : (0.9967, 0.9991)
##     No Information Rate : 0.2846
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9976
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9982   0.9961   0.9959   1.0000
## Specificity            1.0000   0.9994   0.9996   0.9998   0.9990
## Pos Pred Value         1.0000   0.9974   0.9981   0.9990   0.9954
## Neg Pred Value         0.9998   0.9996   0.9992   0.9992   1.0000
## Prevalence             0.2846   0.1934   0.1747   0.1643   0.1830
## Detection Rate         0.2845   0.1930   0.1740   0.1636   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9997   0.9988   0.9978   0.9978   0.9995
```

Random Forest achieves very high accuracy

Prediction on the actual test set

```
predict(forest, test[, -length(colnames(test))])
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```