Practical Machine Learning Course Project

Gabriel Quintanar

19 de junio de 2018

Executive Summary

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: _http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har_ (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har_) (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv_)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv_)

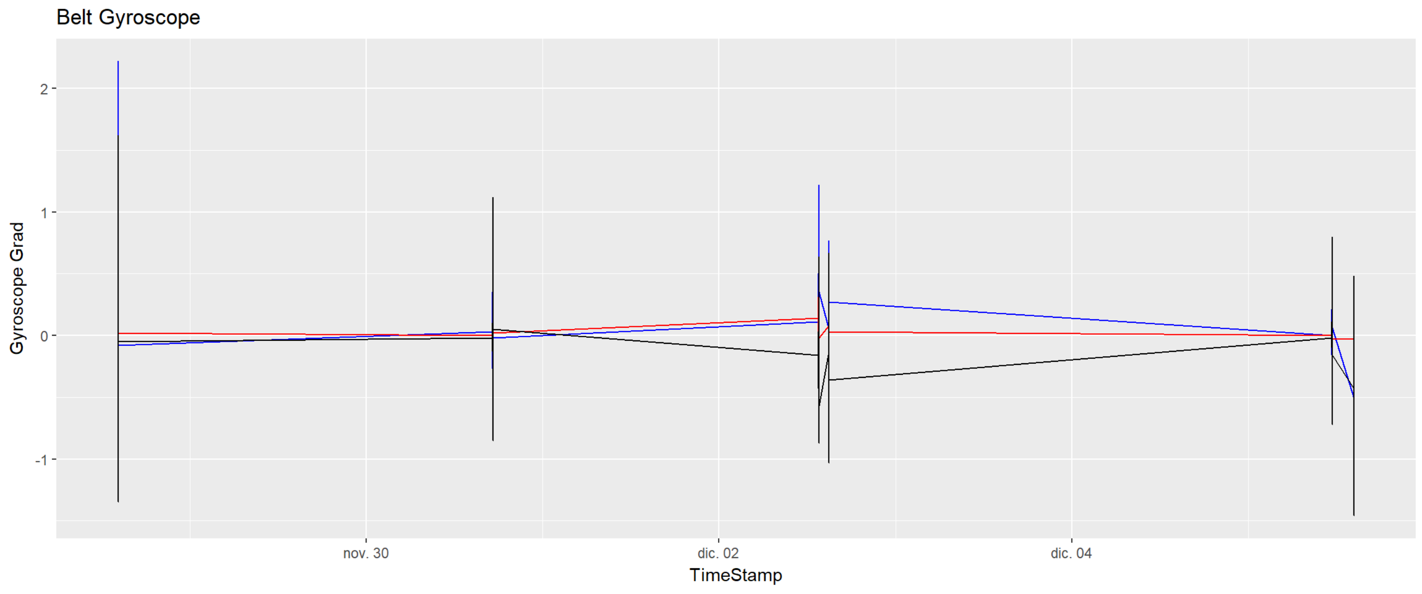
Exploratory Analysis

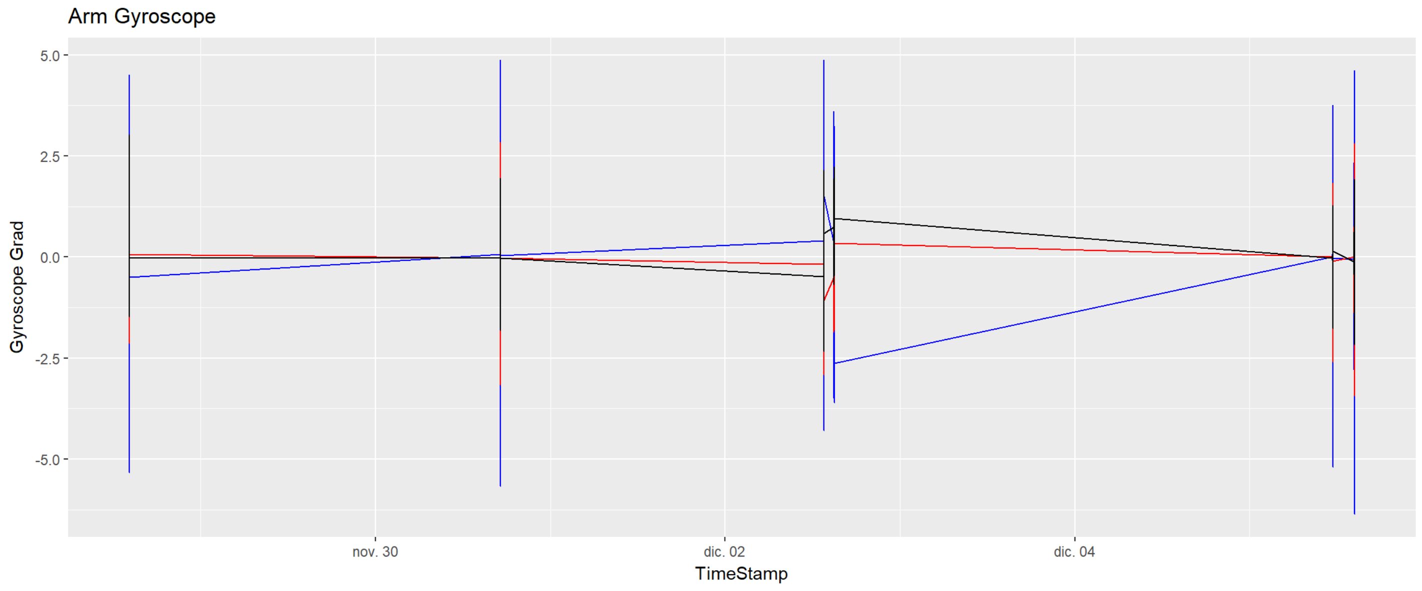
As suggested, looking at the variable ‘Classe’ there are 5 different levels which correspond to the 5 different activities that the subjects did during the test. To understand which activities are, a quick review of the gyroscope variables may be helpful.

```
training <- read.csv("pml-training.csv", header = T, na.strings = c("", "NA"))
testing <- read.csv("pml-testing.csv", header = T, c("", "NA"))

training$cvtd_timestamp <-parse_date_time(training$cvtd_timestamp, "dmYHM")
p1 <- ggplot() +
    geom_line(data = training, aes(x = training$cvtd_timestamp, y = gyros_belt_x), color = "blue") +
    geom_line(data = training, aes(x = training$cvtd_timestamp, y = gyros_belt_y), color = "red") +
    geom_line(data = training, aes(x = training$cvtd_timestamp, y = gyros_belt_z), color = "black") +
    labs(title = "Belt Gyroscope", x = "TimeStamp", y = "Gyroscope Grad") + scale_color_continuous(name
= "Gyroscope",
                                                                                                 label
s = c("X", "Y", "Z"))
p2 <- ggplot() +
    geom_line(data = training, aes(x = training$cvtd_timestamp, y = gyros_arm_x), color = "blue") +
    geom_line(data = training, aes(x = training$cvtd_timestamp, y = gyros_arm_y), color = "red") +
    geom_line(data = training, aes(x = training$cvtd_timestamp, y = gyros_arm_z), color = "black") +
    labs(title = "Arm Gyroscope", x = "TimeStamp", y = "Gyroscope Grad") + scale_color_continuous(name =
 "Gyroscope",
                                                                                                 label
s = c("X", "Y", "Z"))

grid.arrange(p1, p2, nrow = 2)
```

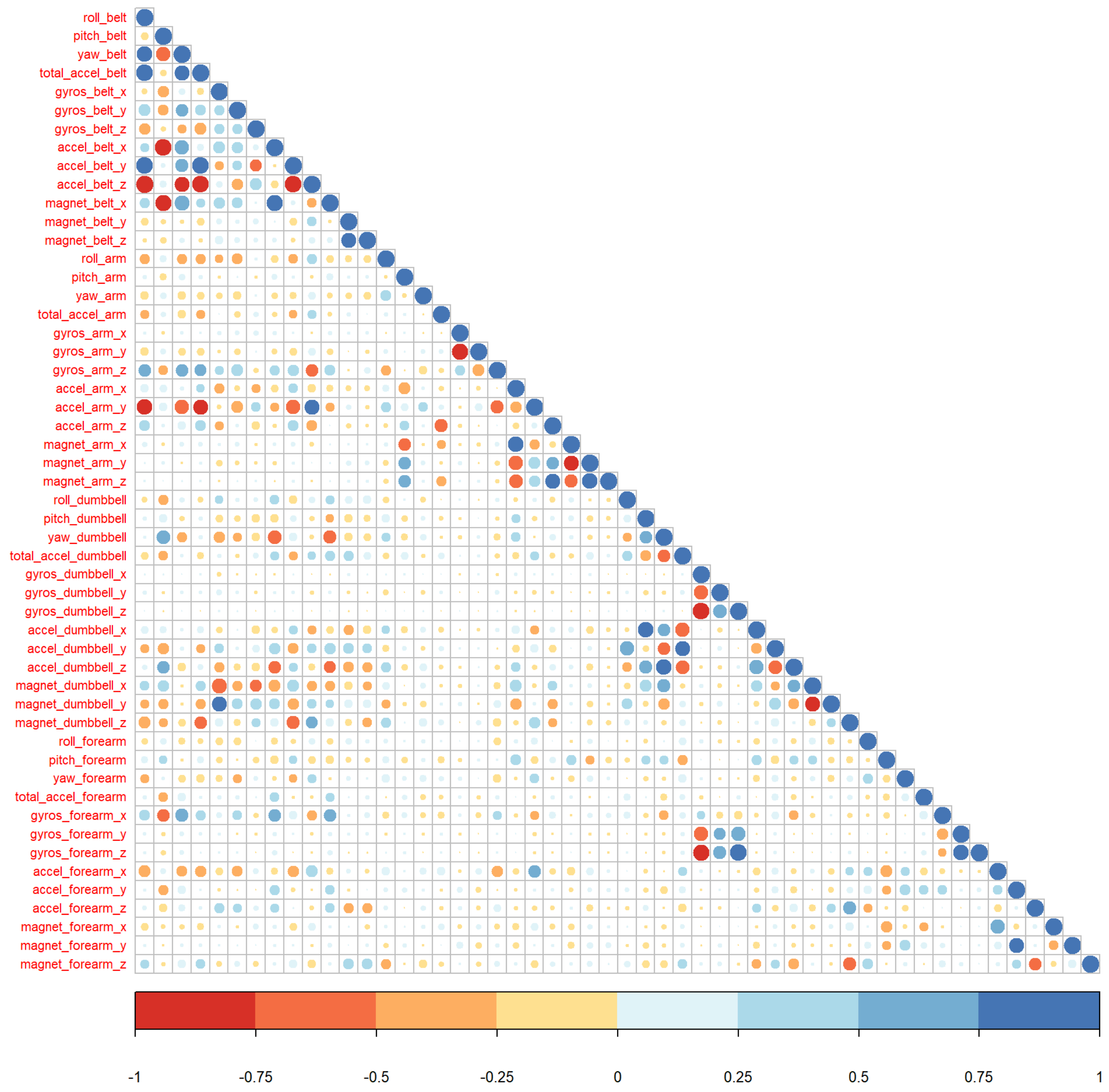




As it can be seen, there 5 particular moments that can be distinguished during the timeline. Our job will be train a model to predict which activity the subject is doing. For that, is obvious that there are many variables that we do not need. We need to get rid of them to make a correlation analysis and determine which variables use in our model.

Training and Testing Sets

```
trainingComplete <- training[,colSums(is.na(training)) == 0]
testingComplete <- testing[,colSums(is.na(testing)) == 0]
trainingComplete <- trainingComplete[, -c(1:7)]
testingComplete <- testingComplete[, -c(1:7)]
corTrain <- cor(trainingComplete[, -53])
corrplot::corrplot(corTrain, method = "circle", type = "lower", col = brewer.pal(n = 8, name = "RdYlBu")
, cl.pos = "b",
                   tl.pos = "l", tl.cex = 0.7)
```

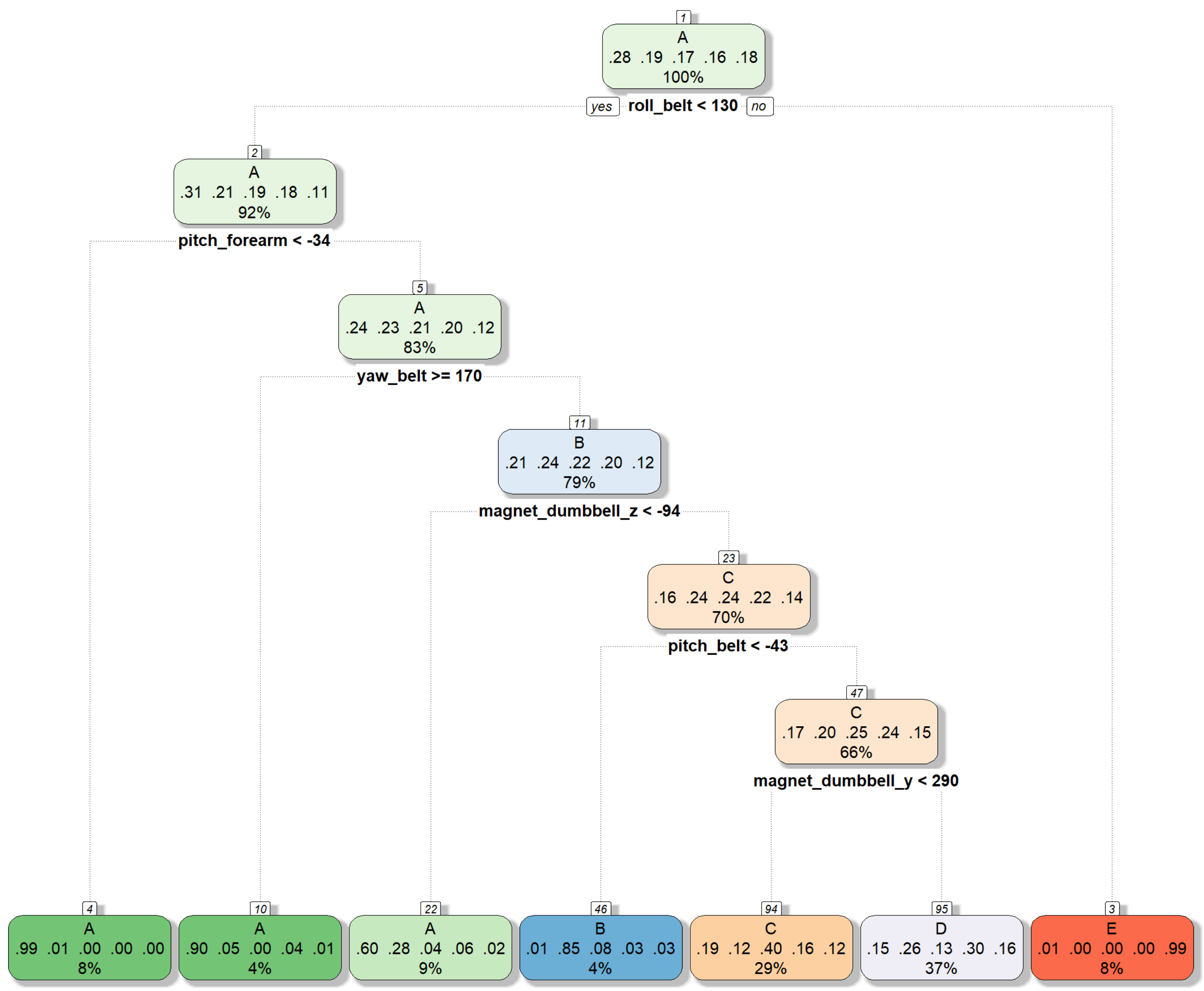


```
set.seed(123)
inTrain <- createDataPartition(trainingComplete$classe, p=0.6, list = F)
tTrain <- trainingComplete[inTrain, ]
tTest <- trainingComplete[-inTrain, ]
```

It makes sense that, to know a specified label, we used some classification techniques. In this case, a decision tree and a random forest will be used.

```
set.seed(220)
fitRF <- train(classe ~ ., data = tTrain, method = "rf")
set.seed(223)
fitDT <- train(classe ~ ., data = tTrain, method = "rpart", trControl =
                   trainControl(method = "repeatedcv", repeats = 5))

fancyRpartPlot(fitDT$finalModel)
```



Rattle 2018-jun.-21 12:36:48 gquin

```
predDT <- predict(fitDT, newdata = tTest)
predRF <- predict(fitRF, newdata = tTest)

cnfM1 <- confusionMatrix(predDT, tTest$classe)
cnfM2 <- confusionMatrix(predRF, tTest$classe)
cnfM1
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1356  239   42   68   20
##          B    5  264   34    8    6
##          C  441  263  907  397  269
##          D  422  752  385  813  510
##          E    8    0    0    0  637
## 
## Overall Statistics
## 
##                Accuracy : 0.5069
##                  95% CI : (0.4958, 0.518)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.3865
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6075  0.17391   0.6630   0.6322  0.44175
## Specificity            0.9343  0.99162   0.7885   0.6846  0.99875
## Pos Pred Value         0.7861  0.83281   0.3983   0.2821  0.98760
## Neg Pred Value         0.8569  0.83344   0.9172   0.9047  0.88821
## Prevalence             0.2845  0.19347   0.1744   0.1639  0.18379
## Detection Rate         0.1728  0.03365   0.1156   0.1036  0.08119
## Detection Prevalence   0.2199  0.04040   0.2902   0.3673  0.08221
## Balanced Accuracy      0.7709  0.58277   0.7258   0.6584  0.72025
```

```
cnfM2
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2227   11    0    0    0
##          B    5 1504    8    0    0
##          C    0    3 1347   17    5
##          D    0    0   13 1269    5
##          E    0    0    0    0 1432
## 
## Overall Statistics
## 
##                Accuracy : 0.9915
##                  95% CI : (0.9892, 0.9934)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9892
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9908   0.9846   0.9868   0.9931
## Specificity            0.9980   0.9979   0.9961   0.9973   1.0000
## Pos Pred Value         0.9951   0.9914   0.9818   0.9860   1.0000
## Neg Pred Value         0.9991   0.9978   0.9968   0.9974   0.9984
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2838   0.1917   0.1717   0.1617   0.1825
## Detection Prevalence   0.2852   0.1933   0.1749   0.1640   0.1825
## Balanced Accuracy      0.9979   0.9944   0.9904   0.9920   0.9965
```

As it can be seen, the random forest model have the highest accuracy. So using a cross-validation with the decision tree model wasn't helpful. Just as a last try, we are going to build a *General Boosted Model*

```
set.seed(400)
fitGBM <- train(classe ~ ., data = tTrain, method = "gbm", verbose = F)
predGBM <- predict(fitGBM, newdata = tTest)

cnfM3 <- confusionMatrix(predGBM, tTest$classe)
kAccuracy <- as.data.frame(bind_rows(cnfM1$overall, cnfM2$overall, cnfM3$overall))
row.names(kAccuracy) <- c("Decision Tree", "Random Forest", "GBM")
cnfM3
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2193   34    0    0    6
##          B   25 1437   56    8   13
##          C    8   41 1294   38   12
##          D    4    2   13 1233   17
##          E    2    4    5    7 1394
## 
## Overall Statistics
## 
##                Accuracy : 0.9624
##                  95% CI : (0.958, 0.9665)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9524
##  Mcnemar's Test P-Value : 1.872e-06
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9825   0.9466   0.9459   0.9588   0.9667
## Specificity            0.9929   0.9839   0.9847   0.9945   0.9972
## Pos Pred Value         0.9821   0.9337   0.9289   0.9716   0.9873
## Neg Pred Value         0.9931   0.9872   0.9885   0.9919   0.9925
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2795   0.1832   0.1649   0.1572   0.1777
## Detection Prevalence   0.2846   0.1962   0.1775   0.1617   0.1800
## Balanced Accuracy      0.9877   0.9653   0.9653   0.9766   0.9820
```

These are the accuracy rates from the three models.

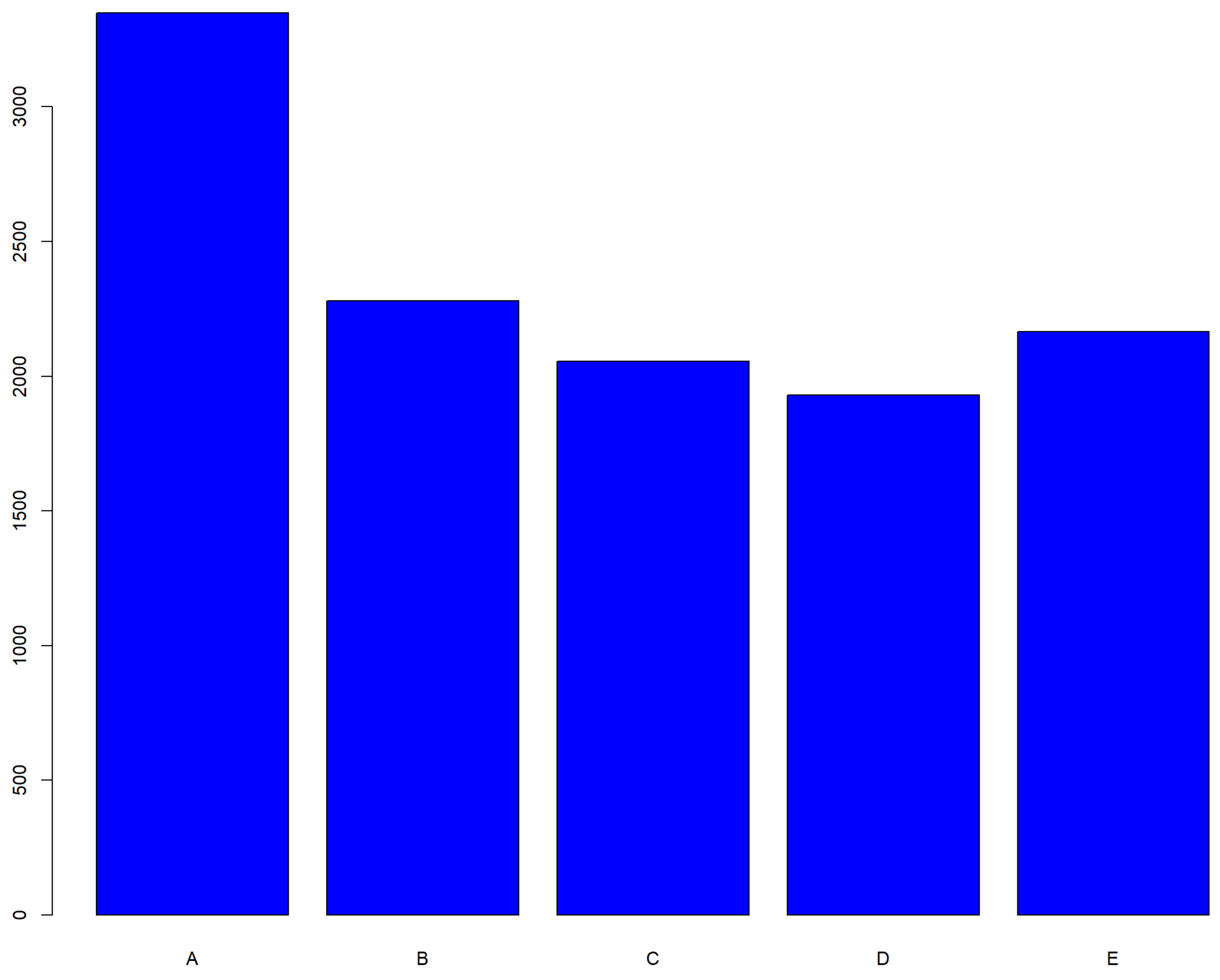
	Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
Decision Tree	0.5068825	0.3864593	0.4957554	0.5180044	0.2844762	0	0.0e+00
Random Forest	0.9914606	0.9891979	0.9891678	0.9933761	0.2844762	0	NaN
GBM	0.9624012	0.9524395	0.9579531	0.9665015	0.2844762	0	1.9e-06

Even though the GBM model has an accuracy greater than 95%, the Random Forest model still has the greatest accuracy rate.

Predicting Test Data

```
predFinal <- predict(fitRF, newdata = testingComplete$classe)

plot(predFinal, col = "blue")
```


3000
2500
2000
1500
1000
500
0
A
B
C
D
E