# Practical Machine Learning Project

*Robert Comyn*

*Saturday, August 22, 2015*

## Introduction

The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbbell of six participants to predict how well an activity was being performed by the wearer. Each participant was asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions (A through E).

The training data and test data sets were provided from the course project web page.

## Methods

Environment in which the analysis was run.

```
sessionInfo()
```

```
## R version 3.1.2 (2014-10-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] BradleyTerry2_1.0-6 brglm_0.5-9         car_2.0-25
##  [4] caret_6.0-52        codetools_0.2-9     colorspace_1.2-4
##  [7] digest_0.6.8        evaluate_0.5.5      foreach_1.4.2
## [10] formatR_1.0         ggplot2_1.0.0       grid_3.1.2
## [13] gtable_0.1.2        gtools_3.4.2        htmltools_0.2.6
## [16] iterators_1.0.7     kernlab_0.9-20      knitr_1.9
## [19] lattice_0.20-29     lme4_1.1-7          MASS_7.3-35
## [22] Matrix_1.1-4        mgcv_1.8-3          minqa_1.2.4
## [25] munsell_0.4.2       nlme_3.1-118        nloptr_1.0.4
## [28] nnet_7.3-8          parallel_3.1.2      pbkrtest_0.4-2
## [31] plyr_1.8.1          proto_0.3-10        quantreg_5.11
## [34] Rcpp_0.11.4         reshape2_1.4.1      rmarkdown_0.6.1
## [37] rpart_4.1-8         scales_0.2.4        SparseM_1.6
## [40] splines_3.1.2       stats4_3.1.2        stringr_0.6.2
## [43] tools_3.1.2         yaml_2.1.13
```

Load the required libraries.

```
library(downloader)
library(fields)
library(caret)
library(rpart)
library(plyr)
library(randomForest)
library(partykit)
library(C50)
library(kernlab)
library(dplyr)
library(ggplot2)
```

### Data Collection

A. Load the data

Download the training and testing data files if they do not exist and save the date downloaded.

```
trainingSaveFile <- "trainingData.rda"
if (! file.exists(trainingSaveFile)) {
    trainingDataFile <- "pml-training.csv"
    # Download training data.
    download("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
        destfile=trainingDataFile)
    trainingDateDownloaded <- date()
    # Read data file.
    trainingData <- read.csv(trainingDataFile, na.strings=c("NA","#DIV/0!",""))
    # Save as R-data file.
    save(trainingData, trainingDateDownloaded, file=trainingSaveFile)
    unlink(trainingDataFile)
    rm(trainingDataFile)
}
testingSaveFile <- "testingData.rda"
if (! file.exists(testingSaveFile)) {
    testingDataFile <- "pml-test.csv"
    # Download training data.
    download("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
        destfile=testingDataFile)
    testingDateDownloaded <- date()
    # Read data file.
    testingData <- read.csv(testingDataFile, na.strings=c("NA","#DIV/0!",""))
    # Save as R-data file.
    save(testingData, testingDateDownloaded, file=testingSaveFile)
    unlink(testingDataFile)
    rm(testingDataFile)
}
```

B. Load the training and testing data into memory if they have not been previously loaded.

```
if (! exists("trainingData")) {
    # Read R-data file.
    load(trainingSaveFile)
}
```

```
if (! exists("testingData")) {
    # Read R-data file.
    load(testingSaveFile)
}
```

Dates data was downloaded.

```
trainingDateDownloaded
```

```
## [1] "Sun Aug 16 19:07:57 2015"
```

```
testingDateDownloaded
```

```
## [1] "Sun Aug 16 19:08:02 2015"
```

### Reproducibility

This analysis is reproducible because seeds are set before any process that uses random sampling including the creation of the training/test sets and the training of each model.

### Partition Training Data

```
set.seed(123)
inTrain <- createDataPartition(y=trainingData$classe, p=0.6, list=FALSE)

training <- trainingData[inTrain,]
testing <- trainingData[-inTrain,]
```

### Exploratory Analysis

```
str(training)
```

```
## 'data.frame':    11776 obs. of  160 variables:
##  $ X                       : int  3 4 6 10 11 12 13 17 18 19 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  820366 120339 304277 484434 500302 528316 560359 692324 732306 740
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 12 12 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.42 1.48 1.45 1.45 1.45 1.43 1.42 1.51 1.55 1.57 ...
##  $ pitch_belt              : num  8.07 8.05 8.06 8.17 8.18 8.18 8.2 8.12 8.08 8.06 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
```

```
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0.02 0.03 0.03 0.02 0.02 0 0 0 ...
##  $ gyros_belt_y            : num  0 0 0 0 0 0 0 0 0.02 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.03 -0.02 0 -0.02 -0.02 0 -0.02 0 -0.02 ...
##  $ accel_belt_x            : int  -20 -22 -21 -21 -21 -22 -22 -21 -21 -20 ...
##  $ accel_belt_y            : int  5 3 4 4 2 2 4 4 5 5 ...
##  $ accel_belt_z            : int  23 21 21 22 23 23 21 22 21 21 ...
##  $ magnet_belt_x           : int  -2 -6 0 -3 -5 -2 -3 -6 1 -3 ...
##  $ magnet_belt_y           : int  600 604 603 609 596 602 606 598 600 603 ...
##  $ magnet_belt_z           : int  -305 -310 -312 -308 -317 -319 -309 -317 -316 -313 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -129 -129 -129 ...
##  $ pitch_arm               : num  22.5 22.1 22 21.6 21.5 21.5 21.4 21.3 21.2 21.2 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  -0.02 -0.03 -0.03 -0.03 -0.03 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ gyros_arm_z             : num  -0.02 0.02 0 -0.02 0 0 -0.02 -0.02 -0.03 -0.02 ...
##  $ accel_arm_x             : int  -289 -289 -289 -288 -290 -288 -287 -289 -288 -289 ...
##  $ accel_arm_y             : int  110 111 111 110 110 111 111 110 108 109 ...
##  $ accel_arm_z             : int  -126 -123 -122 -124 -123 -123 -124 -122 -124 -122 ...
##  $ magnet_arm_x            : int  -368 -372 -369 -376 -366 -363 -372 -371 -373 -369 ...
##  $ magnet_arm_y            : int  344 344 342 334 339 343 338 337 336 340 ...
##  $ magnet_arm_z            : int  513 512 513 516 509 520 509 512 510 509 ...
```

```
##  $ kurtosis_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  12.9 13.4 13.4 13.3 13.1 ...
##  $ pitch_dumbbell          : num  -70.3 -70.4 -70.8 -70.9 -70.6 ...
##  $ yaw_dumbbell            : num  -85.1 -84.9 -84.5 -84.4 -84.7 ...
##  $ kurtosis_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Almost all of the rows have columns with missing values.

```
sum(!complete.cases(trainingData))
```

```
## [1] 19622
```

### Preprocessing

The following function preprocesses a data set by removing columns that are not predictors, removing columns with near zero variance, and removing columns with NAs or more than 60% missing values.

```
preProcessData <- function(data) {
    # Remove columns that are not predictors
    data <- subset(data, select=-c(X, user_name,
                    raw_timestamp_part_1, raw_timestamp_part_2,
                    cvtd_timestamp, new_window, num_window))

    # Remove columns with near zero variance
    nzvCols <- nearZeroVar(data, saveMetrics = TRUE)
```

```
    data <- data[, !nzvCols$nzv]

    # Remove columns with NAs or more than 60% missing values
    tmp <- t(stats(data))
    naCols <- apply(tmp, 1, function(currow) {
            all(is.na(currow)) || (currow["missing values"] > nrow(data) * 0.6)
            #print(is.na(currow["mean"]))
            #print(currow)
         })
    naCols[length(naCols)] <- FALSE # save classe
    data <- data[!naCols]

    return(data)
}
```

Perform preprocessing on the training and testing portions of the Training Data.

```
training <- preProcessData(training)
testing <- preProcessData(testing)
```

Check for variables that have high correlation.

```
M <- abs(cor(subset(training, select=-classe)))
diag <- 0 # all variables are perfectly correlated with themselves
nrow(which(M > 0.8, arr.ind=T))
```

```
## [1] 90
```

Many rows have correlations $> 0.8$. We must use models that can handle data with high correlations between predictors (e.g, tree-based algorithms) or the data should be pre-processed with principal component analysis.

## Modeling and Testing

We chose to create and evaluate models with five different algoritms: RPART, C5.0 (a boosting algorithm in the same category as gbm), Random Forest (rf), Support Vector Machine (svmRadial), and Linear Discriminant Analysis (lda). These algorithms were evaluated with and without additional principal component analysis (pca) preprocessing using the caret package.

After modeling, the following function accumulates results from modeling, performs prediction on the testing data, and creates a results table for all of the models.

```
results <- NULL
testingPredictions <- function(modelName, fit, results, testing) {
    pred <- predict(fit, testing)
    table(pred, testing$classe)
    cm <- confusionMatrix(testing$classe, pred)
    newRow <- as.data.frame(cbind(fit$results[1,], cm$overall["Accuracy"]))
    if (modelName == "rpart" || modelName == "rpartPC") {
        newRow$cp <- NULL
    }
    if (modelName == "C5.0" || modelName == "C5.0PC") {
```

```
        newRow$model <- NULL
        newRow$winnow <- NULL
        newRow$trials <- NULL
    }
    if (modelName == "rf" || modelName == "rfPC") {
        newRow$mtry <- NULL
    }
    if (modelName == "svmRadial" || modelName == "svmRadialPC") {
        newRow[,1] <- NULL
        newRow[,1] <- NULL
    }
    if (modelName == "lda" || modelName == "ldaPC") {
        newRow[,1] <- NULL
    }
    names(newRow)[1:5] <- c("Accuracy", "Kappa", "AccuracySD", "KappaSD", "Test Accuracy")
    rownames(newRow) <- modelName
    results <- rbind(results, newRow)
    results
}
```

Enable parallel processing.

```
library(doParallel)
registerDoParallel(cores=4)
```

### RPART

```
fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 10)

set.seed(123)
rpartFit <- train(classe ~ ., data = training, method = "rpart",
    tuneLength = 30,
    trControl = fitControl)

set.seed(123)
rpartFitPC <- train(classe ~ ., data = training, preProcess="pca", method = "rpart",
    tuneLength = 30,
    trControl = fitControl)

results <- testingPredictions("rpart", rpartFit, results, testing)
results <- testingPredictions("rpartPC", rpartFitPC, results, testing)
```

### C5.0

```
fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

grid <- expand.grid(model = "tree", trials = c(1:100), winnow = FALSE)

set.seed(123)
```

```
C5.0Fit <- train(classe ~ ., data = training, method = "C5.0",
    tuneGrid = grid,
    trControl = fitControl)

set.seed(123)
C5.0FitPC <- train(classe ~ ., data = training, preProcess="pca", method = "C5.0",
    tuneGrid = grid,
    trControl = fitControl)

results <- testingPredictions("C5.0", C5.0Fit, results, testing)
results <- testingPredictions("C5.0PC", C5.0FitPC, results, testing)
```

### Random Forest

```
fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(123)
rfFit <- train(classe ~ ., data = training, method = "rf",
    trControl = fitControl)

set.seed(123)
system.time(rfFitPC <- train(classe ~ ., data = training, preProcess="pca", method = "rf",
    trControl = fitControl))

results <- testingPredictions("rf", rfFit, results, testing)
results <- testingPredictions("rfPC", rfFitPC, results, testing)
```

### Support Vector Machines

```
fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(123)
system.time(svmFit <- train(classe ~ ., data = training, method = "svmRadial",
    tuneLength = 9,
    trControl = fitControl))

set.seed(123)
system.time(svmFitPC <- train(classe ~ ., data = training, preProcess="pca", method = "svmRadial",
    tuneLength = 9,
    trControl = fitControl))

results <- testingPredictions("svmRadial", svmFit, results, testing)
results <- testingPredictions("svmRadialPC", svmFitPC, results, testing)
```

### Linear Discriminant Analysis

```
fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 10)

set.seed(123)
system.time(ldaFit <- train(classe ~ ., data = training, method = "lda",
    trControl = fitControl))

set.seed(123)
system.time(ldaFitPC <- train(classe ~ ., data = training, preProcess="pca", method = "lda",
    trControl = fitControl))

results <- testingPredictions("lda", ldaFit, results, testing)
results <- testingPredictions("ldaPC", ldaFitPC, results, testing)
```

## Results

The table below shows the results of the ten models created. The first four data columns are data from the final model chosen by "train" and the Test Accuracy is from predictions made and the test set.

```
results
```

```
##              Accuracy     Kappa  AccuracySD     KappaSD Test Accuracy
## rpart       0.8274115 0.7814212 0.017606996 0.022246460     0.8483304
## rpartPC     0.6061393 0.5012167 0.017266632 0.021924986     0.6149630
## C5.0        0.9515412 0.9387144 0.007752276 0.009794722     0.9931175
## C5.0PC      0.8420812 0.8002420 0.012808658 0.016172146     0.9645679
## rf          0.9906308 0.9881474 0.003392527 0.004292379     0.9910783
## rfPC        0.9692881 0.9611489 0.005378579 0.006805173     0.9704308
## svmRadial   0.8519309 0.8123256 0.012117371 0.015349921     0.9894214
## svmRadialPC 0.8266553 0.7804943 0.012022429 0.015234227     0.9816467
## lda         0.6984210 0.6182368 0.011893160 0.015058895     0.7032883
## ldaPC       0.5239899 0.3965619 0.011795999 0.014993871     0.5261280
```

The model with the best combination of training and testing accuracy is the random forest model without PCA adn this model was chosen for final testing on the withheld testingData set.

### Out of Sample Error

Out of sample error for random forest model without PCA is (1 - Testing Accuracy) or

```
osError <- sprintf("%1.3f", (1 - results[match("rf", rownames(results)),5]) * 100)
print(paste0("Out of Error Sample is: ", osError, "%"))
```

```
## [1] "Out of Error Sample is: 0.892%"
```

### Variable Importance

The twenty most important variables in the random forest without PCA model are:

```r
varImp(rfFit)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          63.34
## yaw_belt               58.19
## magnet_dumbbell_y      45.41
## magnet_dumbbell_z      44.56
## pitch_belt             44.47
## roll_forearm           41.93
## accel_dumbbell_y       21.76
## magnet_dumbbell_x      19.25
## roll_dumbbell          18.17
## accel_forearm_x        17.68
## magnet_belt_z          15.95
## accel_dumbbell_z       15.26
## accel_belt_z           14.76
## magnet_belt_y          14.64
## magnet_forearm_z       14.09
## total_accel_dumbbell   13.58
## yaw_arm                11.17
## magnet_belt_x          10.48
## gyros_belt_z           10.17
```

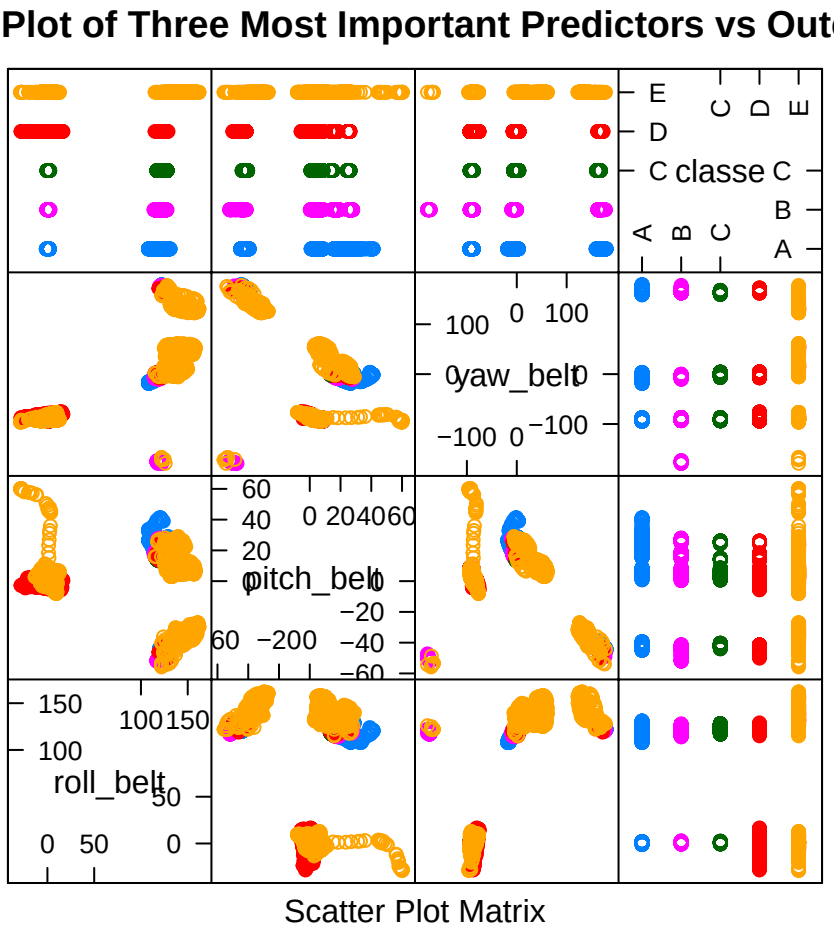


Figure 1: Pairs plot showing the strong correlation between each of the top three most important predictors and the outcome variable (classe).

### Final Testing on Withheld Testing Data

First prepare the data the same way the trainingData was prepared.

```
finalTesting <- preProcessData(testingData)
```

Make predictions.

```
answers <- as.character(predict(rfFit, newdata=finalTesting))
```

Generate files to submit with code provided in assignment.

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("answers/problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
```

```
}
pml_write_files(answers)
```

As expected with such a low out of sample error, all 20 predictions were correct.

## Conclusions

For this data set, the random forest algorithm without PCA provides a better than 99% prediction accuracy. C5.0 without PCA was a close second. The models created with principal component preprocessing (PCA) gave poorer results for each model than the models run without PCA.

## References


1. Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.
2. Coursera Practical Machine Learning Course Project. Training Data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv Test Data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv Accessesed: 8/15/2015
3. R Core. "The R Project for Statistical Computing" URL: http://www.R-project.org.
4. Adler, Joseph. R In A Nutshell. O'Reilly, 2010.
5. Teetor, Paul. R Cookbook. O'Reilly, 2011.
6. Chang, Winston. R Graphics Cookbook. O'Reilly, 2013.
7. R Markdown Page. URL: http://www.rstudio.com/ide/docs/authoring/using_markdown. Accessed 8/5/2015.