Assignment Context:

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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Fetching the datasets for Analysis.

This section describes the R method for fetching the dataset and procedure for creating 20 files required to submit as a part of Course Project Submission.

```r
rm(list = ls())
if (!file.exists("pml-training.csv")) {
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "pml-t
}
if (!file.exists("pml-testing.csv")) {
  download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "pml-te
}
submit <- read.csv("pml-testing.csv", sep = ",", na.strings = c("", "NA"))
data <- read.csv("pml-training.csv", sep = ",", na.strings = c("", "NA"))
```

Cleansing the data

The columns full of NAs are excluded. The features containing NAs are the variance, mean and stddev within each window for each feature. Since the `submit` dataset has no time-dependence, these values are non-relevant and can be discarded. The first 7 features are removed since they are related to the time-series and non-numeric.

```r
# Remove columns full of NAs.
features <- names(submit[,colSums(is.na(submit)) == 0])[8:59]
# Only use features used in submit cases.
data <- data[,c(features,"classe")]
submit <- submit[,c(features,"problem_id")]
```

Bootstrap

withholding 25% of the dataset for testing after the final model is constructed.

```r
set.seed(916)
inTrain = createDataPartition(data$classe, p = 0.75, list = F)
training = data[inTrain,]
testing = data[-inTrain,]
```

Feature Selection

Some features may be highly correlated. The PCA method mixes the final features into components that are difficult to interpret; instead, dropping the features with high correlation (>90%).

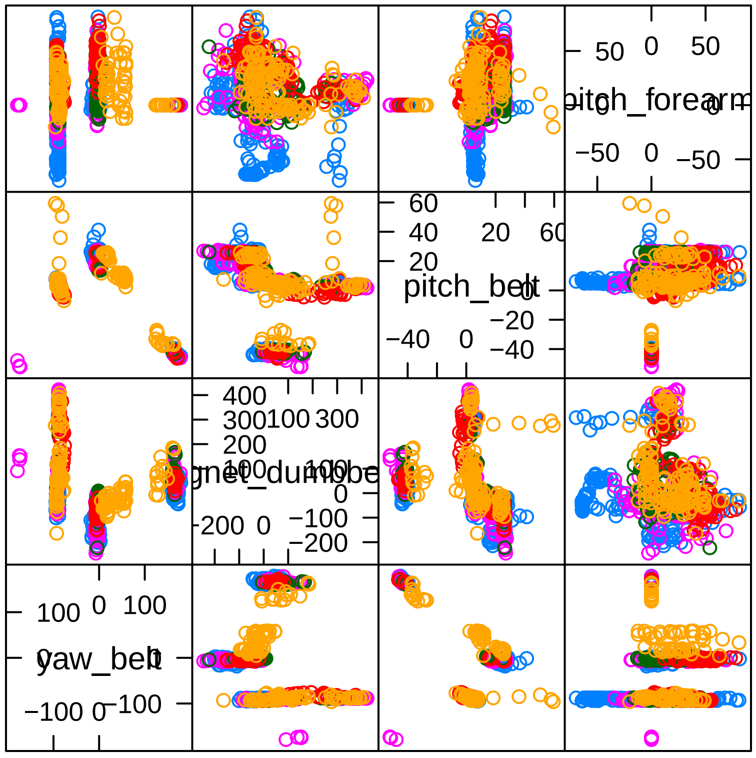
```r
outcome = which(names(training) == "classe")
highCorrCols = findCorrelation(abs(cor(training[,-outcome])),0.90)
highCorrFeatures = names(training)[highCorrCols]
training = training[,-highCorrCols]
outcome = which(names(training) == "classe")
```

The features with high correlation are accel_belt_z, roll_belt, accel_belt_y, accel_belt_x, gyros_arm_y, gyros_forearm_z, and gyros_dumbbell_x.

Feature Importance

The random forest method reduces overfitting and is good for nonlinear features. First, to see if the data is nonlinear, I use the random forest to discover the most important features. The feature plot for the 4 most important features is shown.

```r
fsRF = randomForest(training[,-outcome], training[,outcome], importance = T)
rfImp = data.frame(fsRF$importance)
impFeatures = order(-rfImp$MeanDecreaseGini)
inImp = createDataPartition(data$classe, p = 0.05, list = F)
featurePlot(training[inImp,impFeatures[1:4]],training$classe[inImp], plot = "pairs")
```



Scatter Plot Matrix

The most important features are:

- `pitch_belt`
- `yaw_belt`
- `total_accel_belt`
- `gyros_belt_x`

Training

Train using the random forest and k-nearest neighbors for comparison.

```r
ctrlKNN = trainControl(method = "adaptive_cv")
modelKNN = train(classe ~ ., training, method = "knn", trControl = ctrlKNN)
ctrlRF = trainControl(method = "oob")
modelRF = train(classe ~ ., training, method = "rf", ntree = 200, trControl = ctrlRF)
resultsKNN = data.frame(modelKNN$results)
resultsRF = data.frame(modelRF$results)
```

Testing Out-of-sample error

The random forest will give a larger accuracy compared to k-nearest neighbors. Here, I give the confusion matrix between the KNN and RF models to see how much they agree on the test set, then I compare each model using the test set outcomes.

```r
fitKNN = predict(modelKNN, testing)
fitRF = predict(modelRF, testing)
```

KNN vs. RF

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1360    8   12   16    7
##          B   37  834   27   24   25
##          C   14   31  780   22    9
##          D    9    3   56  721   14
##          E   14   33   22   36  790
##
## Overall Statistics
##
##                Accuracy : 0.915
##                  95% CI : (0.906, 0.922)
##     No Information Rate : 0.292
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.892
##  Mcnemar's Test P-Value : 3.1e-11
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.948    0.917    0.870    0.880    0.935
## Specificity             0.988    0.972    0.981    0.980    0.974
## Pos Pred Value          0.969    0.881    0.911    0.898    0.883
## Neg Pred Value          0.979    0.981    0.971    0.976    0.986
## Prevalence              0.292    0.185    0.183    0.167    0.172
## Detection Rate          0.277    0.170    0.159    0.147    0.161
## Detection Prevalence    0.286    0.193    0.175    0.164    0.183
## Balanced Accuracy       0.968    0.945    0.925    0.930    0.955
```

KNN vs. test set

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1354   44   13    9   14
##          B    6  835   32    3   33
##          C   12   25  779   59   22
##          D   16   24   20  723   36
##          E    7   21   11   10  796
##
## Overall Statistics
##
##                Accuracy : 0.915
##                  95% CI : (0.907, 0.923)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.892
##  Mcnemar's Test P-Value : 3.91e-15
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.971    0.880    0.911    0.899    0.883
## Specificity             0.977    0.981    0.971    0.977    0.988
## Pos Pred Value          0.944    0.919    0.868    0.883    0.942
## Neg Pred Value          0.988    0.971    0.981    0.980    0.974
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.276    0.170    0.159    0.147    0.162
## Detection Prevalence    0.292    0.185    0.183    0.167    0.172
## Balanced Accuracy       0.974    0.931    0.941    0.938    0.936
```

RF vs. test set

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    8    0    0    0
##          B    0  940    7    0    0
##          C    0    1  845   10    0
##          D    0    0    3  794    6
##          E    0    0    0    0  895
##
## Overall Statistics
##
##                Accuracy : 0.993
##                  95% CI : (0.99, 0.995)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.991    0.988    0.988    0.993
## Specificity             0.998    0.998    0.997    0.998    1.000
## Pos Pred Value          0.994    0.993    0.987    0.989    1.000
## Neg Pred Value          1.000    0.998    0.998    0.998    0.999
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.172    0.162    0.183
## Detection Prevalence    0.286    0.193    0.175    0.164    0.183
## Balanced Accuracy       0.999    0.994    0.993    0.993    0.997
```

The random forest fit is clearly more accurate than the k-nearest neighbors method with 99% accuracy.

Submit

Finally, using the random forest model to predict on the 20 cases required for Course Project: Submission

```
##  problem.id answers
##          1       B
##          2       A
##          3       B
##          4       A
##          5       A
##          6       E
##          7       D
##          8       B
##          9       A
##         10       A
##         11       B
##         12       C
##         13       B
##         14       A
##         15       E
##         16       E
##         17       A
##         18       B
##         19       B
##         20       B
```