Background

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).

Getting, partioning and cleaning the data

The training and testing data sets can be found on the following URLs:

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

Load data to memory

```
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Partioning Training data set into two data sets, 60% for myTraining, 40% for myTesting:

```
library(caret)
```

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

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]; myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

The following transformations were used to clean the data:

Transformation 1: Cleaning NearZeroVariance Variables Run this code to view possible NZV Variables:

```
myDataNZV <- nearZeroVar(myTraining, saveMetrics=TRUE)
```

Run this code to create another subset without NZV variables:

```
myNZVvars <- names(myTraining) %in% c("new_window",
"kurtosis_roll_belt", "kurtosis_picth_belt",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1",
"skewness_yaw_belt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm",
"stddev_roll_arm",
"var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm",
"avg_yaw_arm",
"stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm",
"kurtosis_picth_arm",
"kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm",
"skewness_yaw_arm",
"max_roll_arm", "min_roll_arm", "min_pitch_arm",
"amplitude_roll_arm", "amplitude_pitch_arm",
"kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell",
"kurtosis_yaw_dumbbell", "skewness_roll_dumbbell",
"skewness_pitch_dumbbell", "skewness_yaw_dumbbell",
"max_yaw_dumbbell", "min_yaw_dumbbell",
"amplitude_yaw_dumbbell", "kurtosis_roll_forearm",
"kurtosis_picth_forearm", "kurtosis_yaw_forearm",
"skewness_roll_forearm", "skewness_pitch_forearm",
"skewness_yaw_forearm", "max_roll_forearm",
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm",
"amplitude_roll_forearm",
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm",
"var_roll_forearm",
"avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm",
"avg_yaw_forearm",
"stddev_yaw_forearm", "var_yaw_forearm")
myTraining <- myTraining[!myNZVvars]
#To check the new N?? of observations
dim(myTraining)
```

```
## [1] 11776   100
```

Transformation 2: Killing first column of Dataset - ID Removing first ID variable so that it does not interfer with ML Algorithms:

```
myTraining <- myTraining[c(-1)]
```

Transformation 3: Cleaning Variables with too many NAs. For Variables that have more than a 60% threshold of NA's I'm going to leave them out:

```
trainingV3 <- myTraining #creating another subset to iterate in loop
for(i in 1:length(myTraining)) { #for every column in the training
dataset
        if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .6 )
{ #if n?? NAs > 60% of total observations
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)
[j]) ) ==1)  { #if the columns are the same:
                trainingV3 <- trainingV3[ , -j] #Remove that column
            }
        }
    }
}
#To check the new N?? of observations
dim(trainingV3)
```

```
## [1] 11776    58
```

```
#Seting back to our set:
myTraining <- trainingV3
rm(trainingV3)
```

Now let us do the exact same 3 transformations but for our myTesting and testing data sets.

```
clean1 <- colnames(myTraining)
clean2 <- colnames(myTraining[, -58]) #already with classe column
removed
myTesting <- myTesting[clean1]
testing <- testing[clean2]

#To check the new N?? of observations
dim(myTesting)
```

```
## [1] 7846   58
```

```
#To check the new N?? of observations
dim(testing)
```

```
## [1] 20 57
```

```
#Note: The last column - problem_id - which is not equal to training
sets, was also "automagically" removed
#No need for this code:
#testing <- testing[-length(testing)]
```

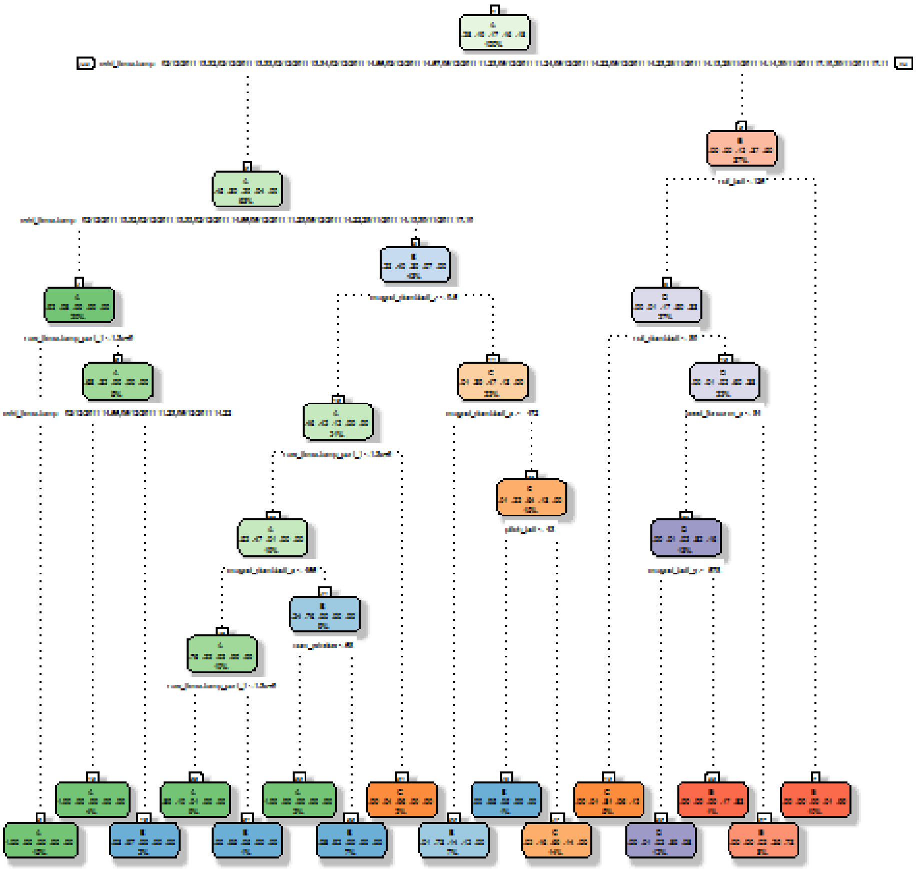
In order to ensure proper functioning of Decision Trees and especially RandomForest Algorithm with the Test data set (data set provided), we need to coerce the data into the same type.

```
for (i in 1:length(testing) ) {
        for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) )
==1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}
#And to make sure Coertion really worked, simple smart ass technique:
testing <- rbind(myTraining[2, -58] , testing) #note row 2 does not
mean anything, this will be removed right.. now:
testing <- testing[-1,]
```

Using ML algorithms for prediction: Decision Tree

```
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")

fancyRpartPlot(modFitA1)
```



Rattle 2015-фев-22 01:26:46 Мария

Predicting:

```
predictionsA1 <- predict(modFitA1, myTesting, type = "class")
```

Using confusion Matrix to test results:

```
confusionMatrix(predictionsA1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2145   55    7    3    0
##          B   65 1261   78   55    0
##          C   22  191 1239  207   47
##          D    0   11   25  805   81
##          E    0    0   19  216 1314
## 
## Overall Statistics
## 
##                Accuracy : 0.8621
##                  95% CI : (0.8543, 0.8697)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.8255
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9610   0.8307   0.9057   0.6260   0.9112
## Specificity            0.9884   0.9687   0.9279   0.9822   0.9633
## Pos Pred Value         0.9706   0.8643   0.7263   0.8731   0.8483
## Neg Pred Value         0.9846   0.9598   0.9790   0.9305   0.9797
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2734   0.1607   0.1579   0.1026   0.1675
## Detection Prevalence   0.2817   0.1860   0.2174   0.1175   0.1974
## Balanced Accuracy      0.9747   0.8997   0.9168   0.8041   0.9373
```

```
#Overall Statistics

#               Accuracy : 0.8683
#                 95% CI : (0.8607, 0.8757)
#    No Information Rate : 0.2845
#    P-Value [Acc > NIR] : < 2.2e-16

#                  Kappa : 0.8335
```

Using ML algorithms for prediction: Random Forests

```
modFitB1 <- randomForest(classe ~. , data=myTraining)
```

Predicting:

```
predictionsB1 <- predict(modFitB1, myTesting, type = "class")
```

Using confusion Matrix to test results:

```
confusionMatrix(predictionsB1, myTesting$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1516    2    0    0
##          C    0    0 1361    8    0
##          D    0    0    5 1278    1
##          E    0    0    0    0 1441
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9977          
##                  95% CI : (0.9964, 0.9986)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9971          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   0.9949   0.9938   0.9993
## Specificity            0.9996   0.9997   0.9988   0.9991   1.0000
## Pos Pred Value         0.9991   0.9987   0.9942   0.9953   1.0000
## Neg Pred Value         1.0000   0.9997   0.9989   0.9988   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1735   0.1629   0.1837
## Detection Prevalence   0.2847   0.1935   0.1745   0.1637   0.1837
## Balanced Accuracy      0.9998   0.9992   0.9968   0.9964   0.9997
```

```
#Overall Statistics

 #              Accuracy : 0.999           
 #                95% CI : (0.998, 0.9996)  
 #   No Information Rate : 0.2845          
 #   P-Value [Acc > NIR] : < 2.2e-16       

 #                 Kappa : 0.9987          
 #Mcnemar's Test P-Value : NA
```

Random Forests yielded better Results, as expected!

Generating Files to submit as answers for the Assignment:

Finally, using the provided Test Set out-of-sample error:

For Random Forests is, which yielded a much better prediction:

```
predictionsB2 <- predict(modFitB1, testing, type = "class")
```

Function to generate files with predictions to submit for assignment

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")

write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)

  }
}

pml_write_files(predictionsB2)
```