title: "Quantified Self Movement Prediction Assignment" author: "Raviraj Chittaranjan date: "July 1, 2016" output: word\_document

# Background

Using devices such as JawboneUp, NikeFuelBand, and Fitbitit 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 assignment, the goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and develop a machine learning algorithm. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website: <http://groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset).

# Data and R packages

## Load packages, set caching

## Loading required package: caret

## Warning: package 'caret' was built under R version 3.2.5

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

## Warning: package 'ggplot2' was built under R version 3.2.5

## Loading required package: corrplot

## Warning: package 'corrplot' was built under R version 3.2.5

## Loading required package: Rtsne

## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,  
## logical.return = TRUE, : there is no package called 'Rtsne'

## Loading required package: knitr

# Getting Data

Set the variables for the URL of training and testing data

train.link <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  
test.link <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

Set the variables for the file names

train.fname <- "pml-training.csv"  
test.fname <- "pml-testing.csv"  
  
# if files does not exist, download the files  
  
if (!file.exists(train.fname)) {  
 download.file(train.link, destfile=train.fname, method="curl")  
}  
if (!file.exists(test.fname)) {  
 download.file(test.link, destfile=test.fname, method="curl")  
}  
# load the CSV files as data.frame   
train.data = read.csv("pml-training.csv")  
test.data = read.csv("pml-testing.csv")  
names(train.data)

## [1] "X" "user\_name"   
## [3] "raw\_timestamp\_part\_1" "raw\_timestamp\_part\_2"   
## [5] "cvtd\_timestamp" "new\_window"   
## [7] "num\_window" "roll\_belt"   
## [9] "pitch\_belt" "yaw\_belt"   
## [11] "total\_accel\_belt" "kurtosis\_roll\_belt"   
## [13] "kurtosis\_picth\_belt" "kurtosis\_yaw\_belt"   
## [15] "skewness\_roll\_belt" "skewness\_roll\_belt.1"   
## [17] "skewness\_yaw\_belt" "max\_roll\_belt"   
## [19] "max\_picth\_belt" "max\_yaw\_belt"   
## [21] "min\_roll\_belt" "min\_pitch\_belt"   
## [23] "min\_yaw\_belt" "amplitude\_roll\_belt"   
## [25] "amplitude\_pitch\_belt" "amplitude\_yaw\_belt"   
## [27] "var\_total\_accel\_belt" "avg\_roll\_belt"   
## [29] "stddev\_roll\_belt" "var\_roll\_belt"   
## [31] "avg\_pitch\_belt" "stddev\_pitch\_belt"   
## [33] "var\_pitch\_belt" "avg\_yaw\_belt"   
## [35] "stddev\_yaw\_belt" "var\_yaw\_belt"   
## [37] "gyros\_belt\_x" "gyros\_belt\_y"   
## [39] "gyros\_belt\_z" "accel\_belt\_x"   
## [41] "accel\_belt\_y" "accel\_belt\_z"   
## [43] "magnet\_belt\_x" "magnet\_belt\_y"   
## [45] "magnet\_belt\_z" "roll\_arm"   
## [47] "pitch\_arm" "yaw\_arm"   
## [49] "total\_accel\_arm" "var\_accel\_arm"   
## [51] "avg\_roll\_arm" "stddev\_roll\_arm"   
## [53] "var\_roll\_arm" "avg\_pitch\_arm"   
## [55] "stddev\_pitch\_arm" "var\_pitch\_arm"   
## [57] "avg\_yaw\_arm" "stddev\_yaw\_arm"   
## [59] "var\_yaw\_arm" "gyros\_arm\_x"   
## [61] "gyros\_arm\_y" "gyros\_arm\_z"   
## [63] "accel\_arm\_x" "accel\_arm\_y"   
## [65] "accel\_arm\_z" "magnet\_arm\_x"   
## [67] "magnet\_arm\_y" "magnet\_arm\_z"   
## [69] "kurtosis\_roll\_arm" "kurtosis\_picth\_arm"   
## [71] "kurtosis\_yaw\_arm" "skewness\_roll\_arm"   
## [73] "skewness\_pitch\_arm" "skewness\_yaw\_arm"   
## [75] "max\_roll\_arm" "max\_picth\_arm"   
## [77] "max\_yaw\_arm" "min\_roll\_arm"   
## [79] "min\_pitch\_arm" "min\_yaw\_arm"   
## [81] "amplitude\_roll\_arm" "amplitude\_pitch\_arm"   
## [83] "amplitude\_yaw\_arm" "roll\_dumbbell"   
## [85] "pitch\_dumbbell" "yaw\_dumbbell"   
## [87] "kurtosis\_roll\_dumbbell" "kurtosis\_picth\_dumbbell"   
## [89] "kurtosis\_yaw\_dumbbell" "skewness\_roll\_dumbbell"   
## [91] "skewness\_pitch\_dumbbell" "skewness\_yaw\_dumbbell"   
## [93] "max\_roll\_dumbbell" "max\_picth\_dumbbell"   
## [95] "max\_yaw\_dumbbell" "min\_roll\_dumbbell"   
## [97] "min\_pitch\_dumbbell" "min\_yaw\_dumbbell"   
## [99] "amplitude\_roll\_dumbbell" "amplitude\_pitch\_dumbbell"  
## [101] "amplitude\_yaw\_dumbbell" "total\_accel\_dumbbell"   
## [103] "var\_accel\_dumbbell" "avg\_roll\_dumbbell"   
## [105] "stddev\_roll\_dumbbell" "var\_roll\_dumbbell"   
## [107] "avg\_pitch\_dumbbell" "stddev\_pitch\_dumbbell"   
## [109] "var\_pitch\_dumbbell" "avg\_yaw\_dumbbell"   
## [111] "stddev\_yaw\_dumbbell" "var\_yaw\_dumbbell"   
## [113] "gyros\_dumbbell\_x" "gyros\_dumbbell\_y"   
## [115] "gyros\_dumbbell\_z" "accel\_dumbbell\_x"   
## [117] "accel\_dumbbell\_y" "accel\_dumbbell\_z"   
## [119] "magnet\_dumbbell\_x" "magnet\_dumbbell\_y"   
## [121] "magnet\_dumbbell\_z" "roll\_forearm"   
## [123] "pitch\_forearm" "yaw\_forearm"   
## [125] "kurtosis\_roll\_forearm" "kurtosis\_picth\_forearm"   
## [127] "kurtosis\_yaw\_forearm" "skewness\_roll\_forearm"   
## [129] "skewness\_pitch\_forearm" "skewness\_yaw\_forearm"   
## [131] "max\_roll\_forearm" "max\_picth\_forearm"   
## [133] "max\_yaw\_forearm" "min\_roll\_forearm"   
## [135] "min\_pitch\_forearm" "min\_yaw\_forearm"   
## [137] "amplitude\_roll\_forearm" "amplitude\_pitch\_forearm"   
## [139] "amplitude\_yaw\_forearm" "total\_accel\_forearm"   
## [141] "var\_accel\_forearm" "avg\_roll\_forearm"   
## [143] "stddev\_roll\_forearm" "var\_roll\_forearm"   
## [145] "avg\_pitch\_forearm" "stddev\_pitch\_forearm"   
## [147] "var\_pitch\_forearm" "avg\_yaw\_forearm"   
## [149] "stddev\_yaw\_forearm" "var\_yaw\_forearm"   
## [151] "gyros\_forearm\_x" "gyros\_forearm\_y"   
## [153] "gyros\_forearm\_z" "accel\_forearm\_x"   
## [155] "accel\_forearm\_y" "accel\_forearm\_z"   
## [157] "magnet\_forearm\_x" "magnet\_forearm\_y"   
## [159] "magnet\_forearm\_z" "classe"

# Data Preparation

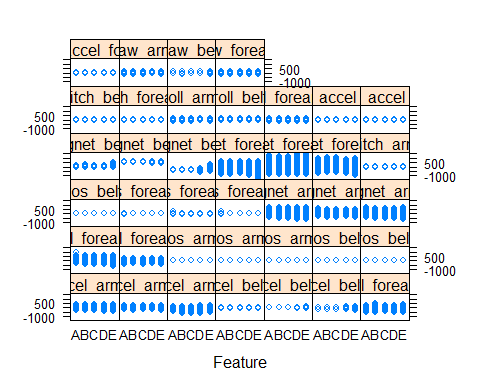
The assignment needs us to use data from accelerometers on the belt, forearm, arm, and dumbell, so the features are extracted based on these keywords along with the classe feature.

## [1] "A" "B" "C" "D" "E"

## [1] 1 1 1 1 1 1  
## Levels: 1 2 3 4 5

# Plot the relationship between features and outcome.

featurePlot(train, outcome.tmp, "strip")



From the above plot, we can see that each feature has relatively the same distribution among the 5 outcome levels (A, B, C, D, E).

## Check for features's variance

Based on the principal component analysis(PCA), it is necessary that features have maximum variance for maximum uniqueness, so that each feature is as distant as possible from other features.

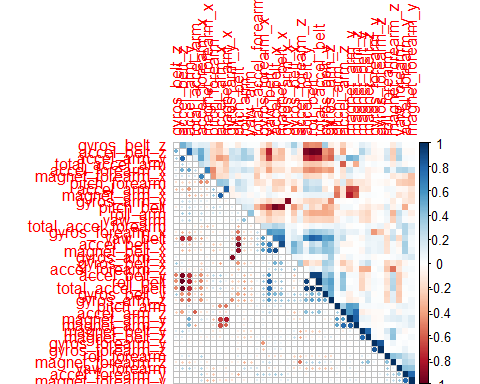
# check for zero variance  
zvar = nearZeroVar(train, saveMetrics=TRUE)  
zvar

## freqRatio percentUnique zeroVar nzv  
## roll\_belt 1.101904 6.7781062 FALSE FALSE  
## pitch\_belt 1.036082 9.3772296 FALSE FALSE  
## yaw\_belt 1.058480 9.9734991 FALSE FALSE  
## total\_accel\_belt 1.063160 0.1477933 FALSE FALSE  
## gyros\_belt\_x 1.058651 0.7134849 FALSE FALSE  
## gyros\_belt\_y 1.144000 0.3516461 FALSE FALSE  
## gyros\_belt\_z 1.066214 0.8612782 FALSE FALSE  
## accel\_belt\_x 1.055412 0.8357966 FALSE FALSE  
## accel\_belt\_y 1.113725 0.7287738 FALSE FALSE  
## accel\_belt\_z 1.078767 1.5237998 FALSE FALSE  
## magnet\_belt\_x 1.090141 1.6664968 FALSE FALSE  
## magnet\_belt\_y 1.099688 1.5187035 FALSE FALSE  
## magnet\_belt\_z 1.006369 2.3290184 FALSE FALSE  
## roll\_arm 52.338462 13.5256345 FALSE FALSE  
## pitch\_arm 87.256410 15.7323412 FALSE FALSE  
## yaw\_arm 33.029126 14.6570176 FALSE FALSE  
## total\_accel\_arm 1.024526 0.3363572 FALSE FALSE  
## gyros\_arm\_x 1.015504 3.2769341 FALSE FALSE  
## gyros\_arm\_y 1.454369 1.9162165 FALSE FALSE  
## gyros\_arm\_z 1.110687 1.2638875 FALSE FALSE  
## accel\_arm\_x 1.017341 3.9598410 FALSE FALSE  
## accel\_arm\_y 1.140187 2.7367241 FALSE FALSE  
## accel\_arm\_z 1.128000 4.0362858 FALSE FALSE  
## magnet\_arm\_x 1.000000 6.8239731 FALSE FALSE  
## magnet\_arm\_y 1.056818 4.4439914 FALSE FALSE  
## magnet\_arm\_z 1.036364 6.4468454 FALSE FALSE  
## roll\_forearm 11.589286 11.0895933 FALSE FALSE  
## pitch\_forearm 65.983051 14.8557741 FALSE FALSE  
## yaw\_forearm 15.322835 10.1467740 FALSE FALSE  
## total\_accel\_forearm 1.128928 0.3567424 FALSE FALSE  
## gyros\_forearm\_x 1.059273 1.5187035 FALSE FALSE  
## gyros\_forearm\_y 1.036554 3.7763735 FALSE FALSE  
## gyros\_forearm\_z 1.122917 1.5645704 FALSE FALSE  
## accel\_forearm\_x 1.126437 4.0464784 FALSE FALSE  
## accel\_forearm\_y 1.059406 5.1116094 FALSE FALSE  
## accel\_forearm\_z 1.006250 2.9558659 FALSE FALSE  
## magnet\_forearm\_x 1.012346 7.7667924 FALSE FALSE  
## magnet\_forearm\_y 1.246914 9.5403119 FALSE FALSE  
## magnet\_forearm\_z 1.000000 8.5771073 FALSE FALSE

It appears that there are no features without variability (all has enough variance). So there is no feature to be removed further.

# Let's plot a correlation matrix between features.

corrplot.mixed(cor(train), lower="circle", upper="color",   
 tl.pos="lt", diag="n", order="hclust", hclust.method="complete")



A good set of features are visibile when they are highly uncorrelated with each other. The plot above shows average correlation which is not too high, so no further PCA preprocessing is needed.

# Modeling

## Ran into trouble using Random Forest as my laptop couldn't complete the train function. Switched to XGBOOST instead. It just ran fine in few minutes

require(xgboost)

## Loading required package: xgboost

## Warning: package 'xgboost' was built under R version 3.2.5

train.matrix = as.matrix(train)  
mode(train.matrix) = "numeric"  
test.matrix = as.matrix(test)  
mode(test.matrix) = "numeric"  
# convert outcome from factor to numeric matrix   
# xgboost takes multi-labels in [0, numOfClass)  
y = as.matrix(as.integer(outcome)-1)  
  
param <- list("objective" = "multi:softprob", # multiclass classification   
 "num\_class" = num.class, # number of classes   
 "eval\_metric" = "merror", # evaluation metric   
 "nthread" = 8, # number of threads to be used   
 "max\_depth" = 16, # maximum depth of tree   
 "eta" = 0.3, # step size shrinkage   
 "gamma" = 0, # minimum loss reduction   
 "subsample" = 1, # part of data instances to grow tree   
 "colsample\_bytree" = 1, # subsample ratio of columns when constructing each tree   
 "min\_child\_weight" = 12 # minimum sum of instance weight needed in a child   
 )  
set.seed(1234)  
  
system.time( bst.cv <- xgb.cv(param=param, data=train.matrix, label=y,   
 nfold=4, nrounds=200, prediction=TRUE, verbose=FALSE) )

## user system elapsed   
## 489.08 23.67 171.89

pred.cv = matrix(bst.cv$pred, nrow=length(bst.cv$pred)/num.class, ncol=num.class)  
pred.cv = max.col(pred.cv, "last")

min.merror.idx = which.min(bst.cv$dt[, test.merror.mean])   
min.merror.idx

## [1] 187

# minimum merror  
bst.cv$dt[min.merror.idx,]

## train.merror.mean train.merror.std test.merror.mean test.merror.std  
## 1: 0 0 0.005402 0.000978

## Model training

Fit the XGBoost gradient boosting model on all of the training data.

system.time( bst <- xgboost(param=param, data=train.matrix, label=y,   
 nrounds=min.merror.idx, verbose=0) )

## user system elapsed   
## 146.59 6.58 43.53

## Predict test data using the trained model

pred <- predict(bst, test.matrix)   
head(pred, 10)

## [1] 2.722199e-04 9.982042e-01 1.132105e-03 1.515472e-04 2.398420e-04  
## [6] 9.991074e-01 6.353945e-04 2.402208e-04 3.893657e-06 1.299460e-05

## Decoding prediction

pred = matrix(pred, nrow=num.class, ncol=length(pred)/num.class)  
pred = t(pred)  
pred = max.col(pred, "last")  
pred.char = toupper(letters[pred])

## confusion matrix

confusionMatrix(factor(y+1), factor(pred.cv))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 1 2 3 4 5  
## 1 5566 10 2 2 0  
## 2 12 3772 12 0 1  
## 3 0 24 3384 14 0  
## 4 0 0 19 3194 3  
## 5 0 1 1 8 3597  
##   
## Overall Statistics  
##   
## Accuracy : 0.9944   
## 95% CI : (0.9933, 0.9954)  
## No Information Rate : 0.2843   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.993   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 1 Class: 2 Class: 3 Class: 4 Class: 5  
## Sensitivity 0.9978 0.9908 0.9901 0.9925 0.9989  
## Specificity 0.9990 0.9984 0.9977 0.9987 0.9994  
## Pos Pred Value 0.9975 0.9934 0.9889 0.9932 0.9972  
## Neg Pred Value 0.9991 0.9978 0.9979 0.9985 0.9998  
## Prevalence 0.2843 0.1940 0.1742 0.1640 0.1835  
## Detection Rate 0.2837 0.1922 0.1725 0.1628 0.1833  
## Detection Prevalence 0.2844 0.1935 0.1744 0.1639 0.1838  
## Balanced Accuracy 0.9984 0.9946 0.9939 0.9956 0.9991

You can see the confusion matrix shows concentration of correct predictions as expected. Hence the average accuracy is 99.44%.

## Estimation of the out-of-sample error rate

The testing subset data gives an unbiased estimate of the xgboost algorithm's prediction Accuracy (99.44% as calculated above). The out-of-sample error rate is derived by the formula 100% - Accuracy = 0.66%.

Hence the out-of-sample error rate is 0.66%.

# Creating submission files

path <-"" pml\_write\_files <- function(x) { n <- length(x) for(i in 1: n) { filename <- paste0("problem\_id\_", i, ".txt") write.table(x[i], file=file.path(path, filename), quote=FALSE, row.names=FALSE, col.names=FALSE) } } pml\_write\_files(pred.char)