# Practical Machine Learning Project

*Lisa Mudgett*

*December 21, 2017*

# Introduction

The goal of the project is to use fitness data to predict the manner in which someone did an exercise.

This analysis uses data from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). The data come from devices such as Jawbone Up, Nike FuelBand, and Fitbit, and for this project, we look at data collected from accelerometer on the belt, forearm, arm, and dumbell of six participants.

The data were provided, split into testing and training data for this assignment, and can be found here:

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

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

# Analysis

Let's load the data and packages we'll need for analysis.

```
setwd('C:/Users/r624461/Desktop/Data Science/Practical Machine Learning')
training <- read.csv('pml-training.csv')
testing <- read.csv('pml-testing.csv')

library(caret)
```

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

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

```
## Warning: Installed Rcpp (0.12.10) different from Rcpp used to build dplyr (0.12.1
1).
## Please reinstall dplyr to avoid random crashes or undefined behavior.
```

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)
```

Doing some exploration on the data shows there are columns with near zero variance, columns that are almost always NA, and some informational columns that we won't need for predicting.

```
training <- training[, colSums(is.na(training)) ==0]

nzv <- nearZeroVar(training)
training <- training[, -nzv]

training <- training[, -(1:5)]
```

For the out of sample error assessment and cross validation parts of the assignment, I want to break the training set into another training and testing set. Since this is a medium size data set, I'll use the 60/40 split.

```
inTrain <- createDataPartition(y=training$classe, p=.6, list=FALSE)
newtrain <- training[inTrain,]
newtest <- training[-inTrain,]
```

Let's build some models!

# Random Forest

```
set.seed(4309)
fit_rf <- randomForest(classe~., data=newtrain, prox=TRUE)
predict_rf <- predict(fit_rf, newtest)
confusionMatrix(newtest$class, predict_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1517    1    0    0
##          C    0   10 1358    0    0
##          D    0    0   15 1271    0
##          E    0    0    0    9 1433
##
## Overall Statistics
##
##                Accuracy : 0.9955
##                  95% CI : (0.9938, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9935   0.9884   0.9930   1.0000
## Specificity            1.0000   0.9998   0.9985   0.9977   0.9986
## Pos Pred Value         1.0000   0.9993   0.9927   0.9883   0.9938
## Neg Pred Value         1.0000   0.9984   0.9975   0.9986   1.0000
## Prevalence             0.2845   0.1946   0.1751   0.1631   0.1826
## Detection Rate         0.2845   0.1933   0.1731   0.1620   0.1826
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   0.9966   0.9934   0.9953   0.9993
```

Hey, pretty good results! Our accuracy is 99.48%, so the out of sample error is about half a percent. I like those odds.

# Boosting

```
set.seed(11712)
fit_gbm <- train(classe~., method="gbm", data=newtrain, verbose=FALSE)
predict_gbm <- predict(fit_gbm, newtest)
confusionMatrix(newtest$class, predict_gbm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2228    4    0    0    0
##          B   16 1482   18    1    1
##          C    0   10 1357    1    0
##          D    0    1   28 1256    1
##          E    0    6    2   12 1422
##
## Overall Statistics
##
##                Accuracy : 0.9871
##                  95% CI : (0.9844, 0.9895)
##     No Information Rate : 0.286
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9837
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9929   0.9860   0.9658   0.9890   0.9986
## Specificity            0.9993   0.9943   0.9983   0.9954   0.9969
## Pos Pred Value         0.9982   0.9763   0.9920   0.9767   0.9861
## Neg Pred Value         0.9971   0.9967   0.9926   0.9979   0.9997
## Prevalence             0.2860   0.1916   0.1791   0.1619   0.1815
## Detection Rate         0.2840   0.1889   0.1730   0.1601   0.1812
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9961   0.9902   0.9821   0.9922   0.9977
```

Boosting is pretty good too: 98.41% accuracy on the test set, so the out of sample error is about 1.5%. The Random Forest method was a bit better still.

# Linear Discriminant Analysis

```
set.seed(22115)
fit_lda <- train(classe~., method="lda", data=newtrain)
predict_lda <- predict(fit_lda, newtest)
confusionMatrix(newtest$class, predict_lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1862   63  138  162    7
##          B  186 1020  174   64   74
##          C  145  125  892  145   61
##          D   80   51  149  955   51
##          E   57  215  113  146  911
##
## Overall Statistics
##
##                Accuracy : 0.7188
##                  95% CI : (0.7087, 0.7288)
##     No Information Rate : 0.297
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6441
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7991   0.6920   0.6085   0.6488   0.8252
## Specificity            0.9329   0.9218   0.9254   0.9481   0.9212
## Pos Pred Value         0.8342   0.6719   0.6520   0.7426   0.6318
## Neg Pred Value         0.9166   0.9283   0.9114   0.9212   0.9699
## Prevalence             0.2970   0.1879   0.1868   0.1876   0.1407
## Detection Rate         0.2373   0.1300   0.1137   0.1217   0.1161
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.8660   0.8069   0.7669   0.7984   0.8732
```

Yikes, we're moving in the wrong direction! The accuracy of this prediction method is only 71.41%, so nearly 30% out of sample error.

# Model Selection

Based on the three prediction models I tested above, the Random Forest method was the best, with a 99.49% accuracy on my cross-validation test sample.

Applying the Random Forest model on the original testing set, I scored 100% on the prediction quiz!

```
predict_final <- predict(fit_rf, newdata=testing)
predict_final
```

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