# Human Activity Recognition: Weight Lifting Exercises Dataset Analysis

```r
# load necessary libraries
suppressWarnings(suppressMessages(library(caret)))
suppressWarnings(suppressMessages(library(corrplot)))
suppressWarnings(suppressMessages(library(FactoMineR)))
suppressWarnings(suppressMessages(library(grDevices)))
suppressWarnings(suppressMessages(library(knitr)))
suppressWarnings(suppressMessages(library(randomForest)))
```

```r
# download files if they do not exist
if (!file.exists("pml-training.csv")) {
    download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
        destfile = "pml-training.csv")
}

if (!file.exists("pml-testing.csv")) {
    download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
        destfile = "pml-testing.csv")
}

# read training set and handle na strings
tmp.train <- read.csv("pml-training.csv", na.strings=c("NA","","#DIV/0!"," "), stringsAsFactors=T)
names(tmp.train)[1] <- "id"

# do not keep columns with mostly missing values or irrelevant fields.
keep <- c("id","roll_belt", "pitch_belt", "yaw_belt", "total_accel_belt",
    "gyros_belt_x", "gyros_belt_y", "gyros_belt_z", "accel_belt_x", "accel_belt_y",
    "accel_belt_z", "magnet_belt_x", "magnet_belt_y", "magnet_belt_z", "roll_arm",
    "pitch_arm", "yaw_arm", "total_accel_arm", "gyros_arm_x", "gyros_arm_y",
    "gyros_arm_z", "accel_arm_x", "accel_arm_y", "accel_arm_z", "magnet_arm_x",
    "magnet_arm_y", "magnet_arm_z", "roll_dumbbell", "pitch_dumbbell", "yaw_dumbbell",
    "total_accel_dumbbell", "gyros_dumbbell_x", "gyros_dumbbell_y", "gyros_dumbbell_z",
    "accel_dumbbell_x", "accel_dumbbell_y", "accel_dumbbell_z", "magnet_dumbbell_x",
    "magnet_dumbbell_y", "magnet_dumbbell_z", "roll_forearm", "pitch_forearm",
    "yaw_forearm", "total_accel_forearm", "gyros_forearm_x", "gyros_forearm_y",
    "gyros_forearm_z", "accel_forearm_x", "accel_forearm_y", "accel_forearm_z",
    "magnet_forearm_x", "magnet_forearm_y", "magnet_forearm_z","classe")

train <- tmp.train[,keep]

# load test set and conform it to match training set.
tmp.test <- read.csv("pml-testing.csv", na.strings=c("NA","","#DIV/0!"," "), stringsAsFactors=T)
names(tmp.test)[1] <- "id"
tmp.test$id <- tmp.test$problem_id # copy problem_id to row number field and use as id
tmp.test <- tmp.test[, -160]
tmp.test$classe <- ""
tmp.test <- tmp.test[,keep]
tmp.test$classe <- factor(tmp.test$classe, levels=c("A","B","C","D","E"))
test <- tmp.test

# dimensions of data sets
dim(train)
```

### Data Processing

```
## [1] 19622    54
```

```r
dim(test)
```

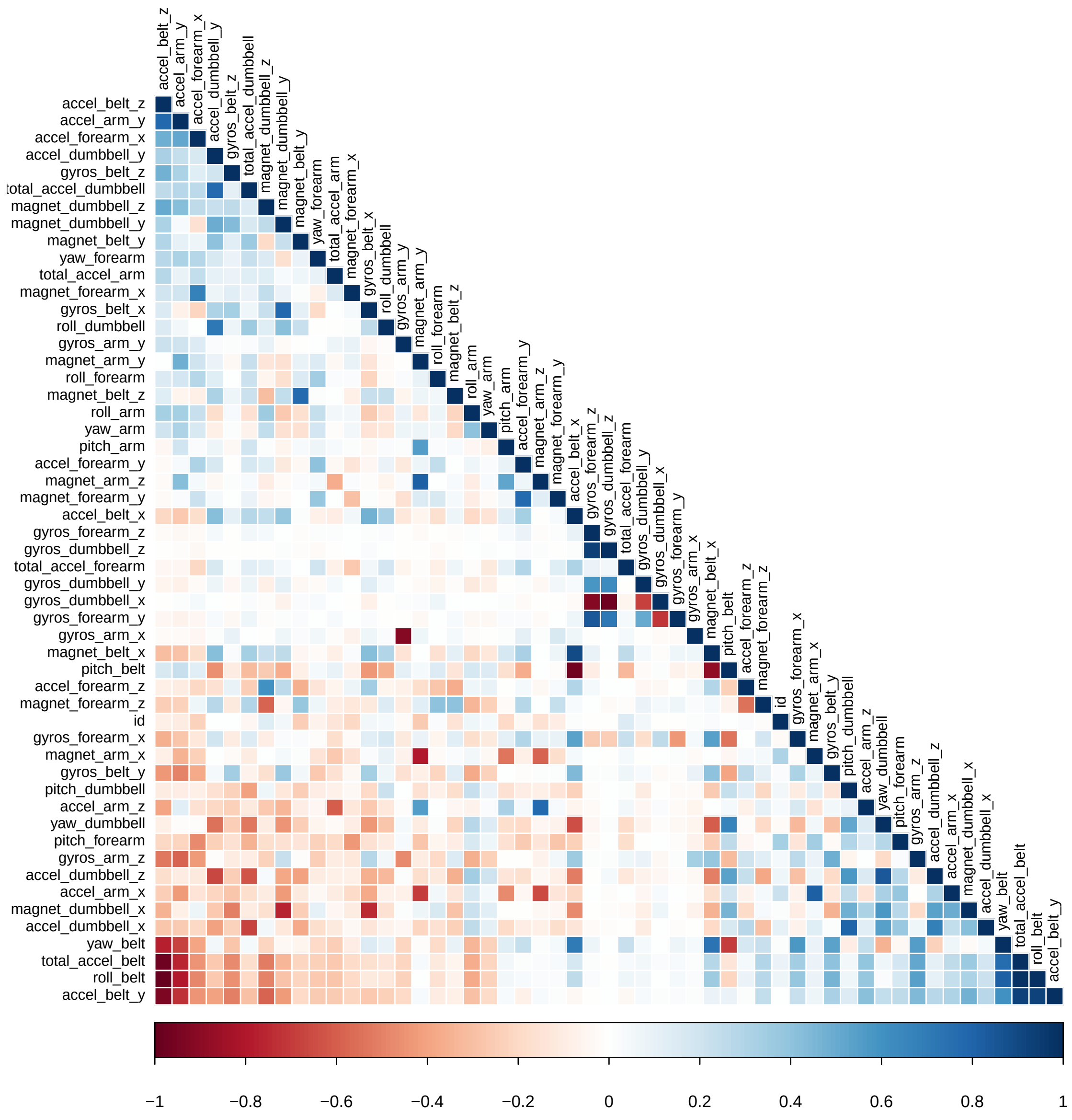
```
## [1] 20 54
```

```r
# remove unneeded variables.
rm(tmp.test,tmp.train)
```

**Feature Selection** The data set for analysis has 160 attributes, so exploratory analysis with scatter plot matrices is impractical. The first step of this analysis will be the attempt to reduce the number of attributes prior to modeling. Instead of allowing the model to select which attributes to use from the raw data set, I will filter the data to remove irrelevant, and highly correlated attributes prior to modeling.

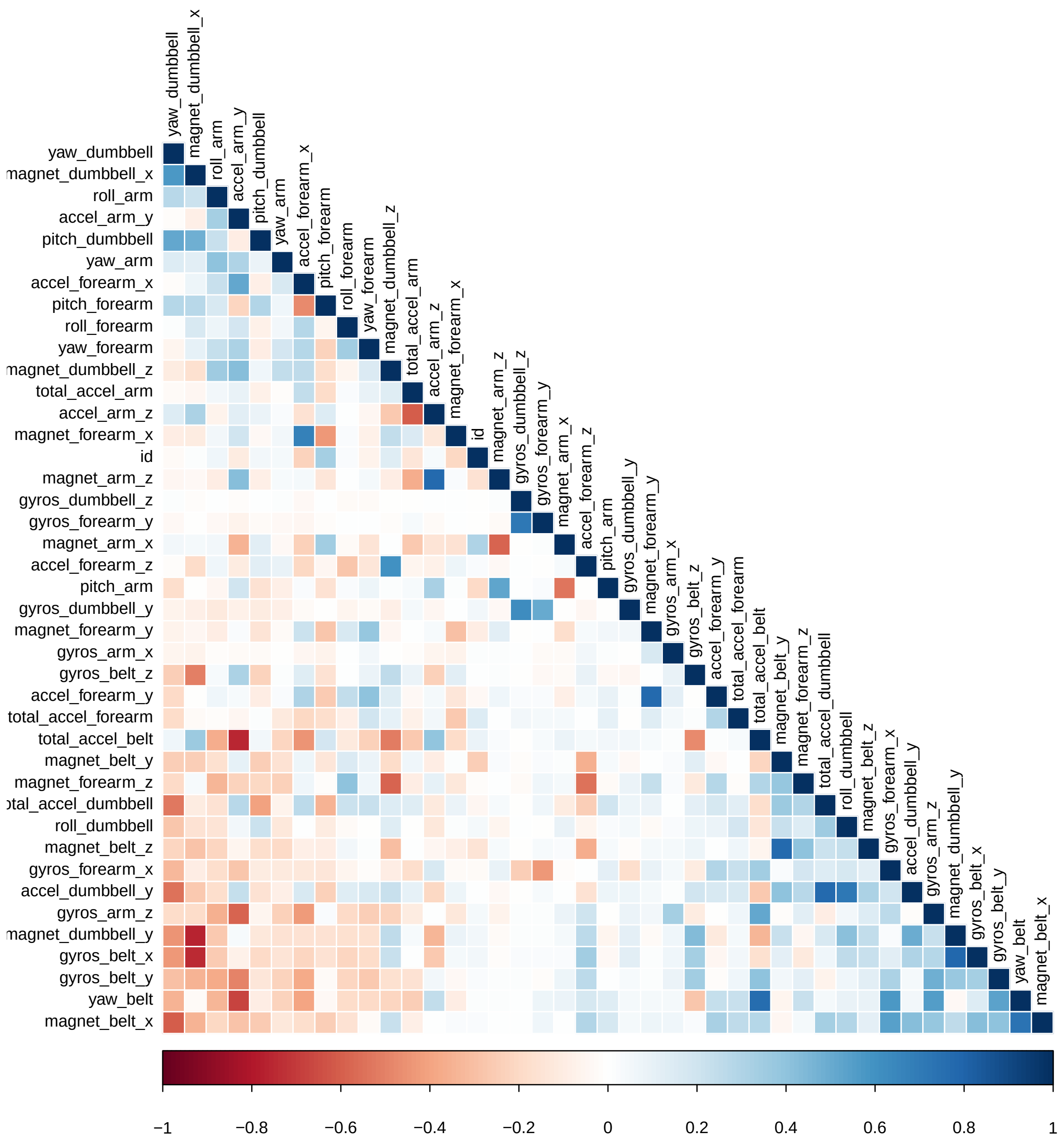
```r
vis.train <- train[,2:ncol(train)-1]

# scale and center training set without id or classe attributes.
vis.train.scale <- scale(vis.train, center=TRUE, scale=TRUE)
corMat <- cor(vis.train.scale) #compute the correlation matrix
```

Plot 1: Correlation plot of initial set of predictors. Extreme correlation values are positioned at the left and right edges.



```r
# remove highly correlated attributes and re-plot.
highlyCor <- findCorrelation(corMat, 0.80) # set highly correlated at 0.80
vis.Filtered.scale <- vis.train.scale[,-highlyCor]
corMat <- cor(vis.Filtered.scale)
```

Plot 2: Correlation plot of revised set of predictors.



**Modeling** Having removed problem attributes, and those that are highly correlated with each other, I will fit a clustering model to predict the outcomes of the testing set.

```
set.seed(123)
mod.train <- train[, 2:ncol(train)]
mod.test <- test[, 2:ncol(train)]
model <- randomForest(classe ~ ., data=mod.train, ntree=100)
model
```

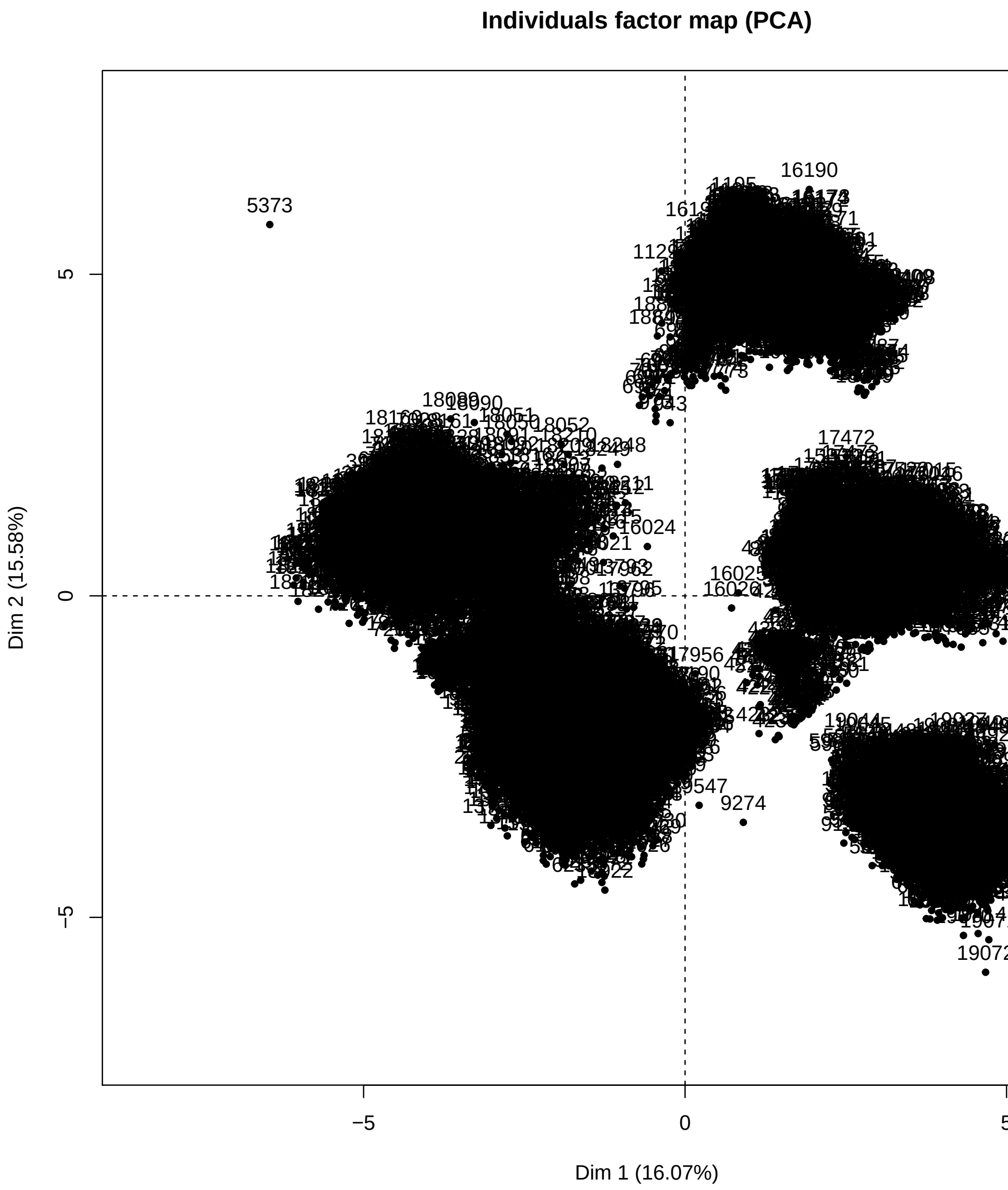
```
##
## Call:
##  randomForest(formula = classe ~ ., data = mod.train, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 0.48%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 5577    2    0    0    1   0.0005376
## B   12 3776    8    0    1   0.0055307
## C    0   22 3397    3    0   0.0073057
## D    0    0   32 3180    4   0.0111940
## E    0    0    3    7 3597   0.0027724
```

```
mod.test$classe <- predict(model, mod.test)
mod.test$classe
```

```
##  [1] 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
```

The predicted values proved to be 100% accurate. The predicted outcomes were nearly identical using 50 trees as they were using 2048 trees.

```


tmp <- read.csv("pml-training.csv", na.strings=c("NA","","#DIV/0!"," "), stringsAsFactors=T)
names(tmp)[1] <- "id" # row number is unique, so keep as id field
tmp.train <- tmp[,keep] # do not keep columns with mostly missing values or timestamps.

# PCA with function PCA
pca.train <- tmp.train
pca <- PCA(pca.train[,2:53], scale.unit=TRUE, ncp=5, graph=F) # scale all features
#plot.PCA(pca, axes=c(1, 2), choix="ind", habillage="ind")
plot.PCA(pca, axes=c(1, 2), choix="ind", habillage="none")
```



**Principal Component Analysis**

```r
#dimdesc(pca) # sort the variables most linked to each principal component

# remove unneeded variables.
rm(pca.train,tmp.train)
```

---

Data for this analysis courtesy of:

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.

References:

[1] http://www.r-bloggers.com/introduction-to-feature-selection-for-bioinformaticians-using-r-correlation-matrix-filters-pca-ba