Analysis-Master

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).

Data Processing

1. Downloading, loading data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Download the file to Data directory

```
dir.create("data")
```

```
## Warning in dir.create("data"): 'data' already exists
```

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "./data

download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "./data/
```

Loading the data to R

```
rawData <- read.csv("./data/pml-training.csv", row.names = 1, na.strings = c("", "NA", "#DIV/0!"))
predicting <- read.csv("./data/pml-testing.csv", row.names = 1, na.strings = c("", "NA", "#DIV/0!"))
```

wrangling with the data

```
library(caret)
library(lubridate)
library(corrplot)
rawData$user_name <- as.factor(rawData$user_name)
rawData$cvtd_timestamp <- dmy_hm(rawData$cvtd_timestamp)
rawData$classe <- as.factor(rawData$classe)
predicting$user_name <- as.factor(predicting$user_name)
predicting$cvtd_timestamp <- dmy_hm(predicting$cvtd_timestamp)
predicting$problem_id <- as.factor(predicting$problem_id)
rawData <- rawData[,-c(2,3,5,6)] #removing timestamp and windows (overlapping info)
predicting<- predicting[,-c(2,3,5,6)]
dim(rawData)
```

```
## [1] 19622   155
```

```
NZV <- nearZeroVar(rawData)
rawData <- rawData[, -NZV] #removing columns with not much variance
predicting <- predicting[, -NZV]
dim(rawData)
```

```
## [1] 19622   120
```

```
AllNA    <- sapply(rawData, function(x) mean(is.na(x))) > 0.95
rawData <- rawData[, AllNA==FALSE] #removing columns with NA > 95%
predicting <- predicting[, AllNA==FALSE]
dim(rawData)
```

```
## [1] 19622    55
```

```
sum(is.na(rawData))
```

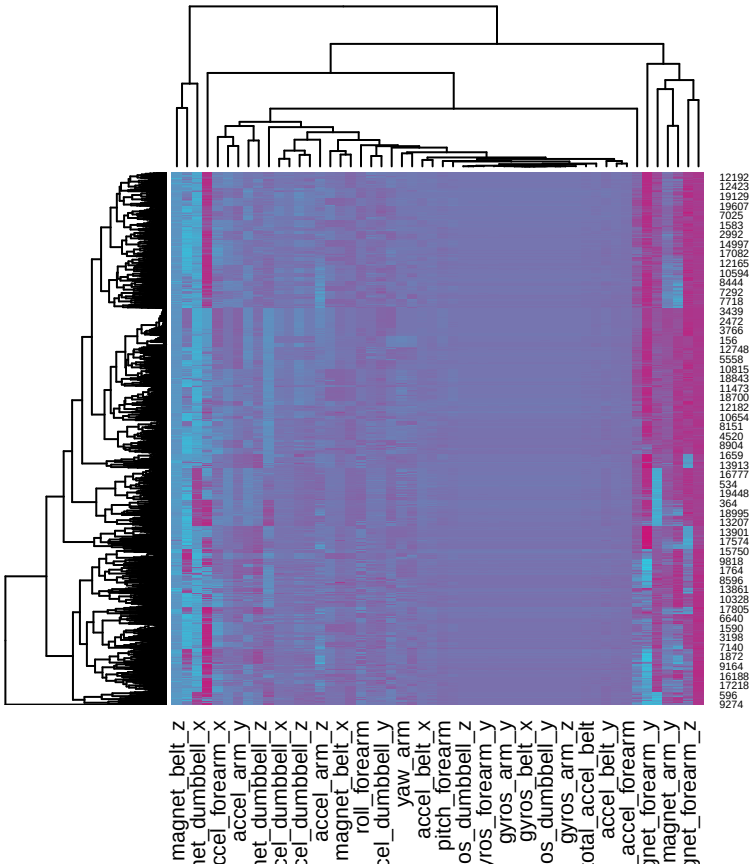
```
## [1] 0
```

```
set.seed(123456)
inTrain = createDataPartition(rawData$classe, p = 3/4)[[1]]
training <- rawData[inTrain, ]
testing <- rawData[-inTrain, ]
dim(training)
```

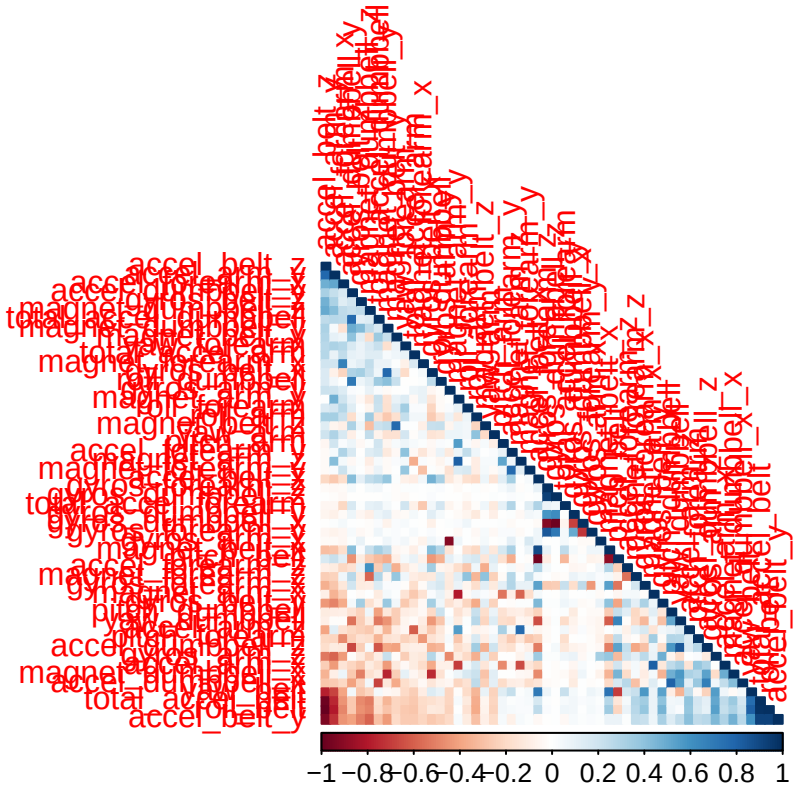
```
## [1] 14718    55
```

Exploratory Data analysis

```
library(ggplot2)
myColors <- colorRampPalette(c("cyan", "deeppink3"))
heatmap(as.matrix(rawData[3:54]), col=myColors(100))
```

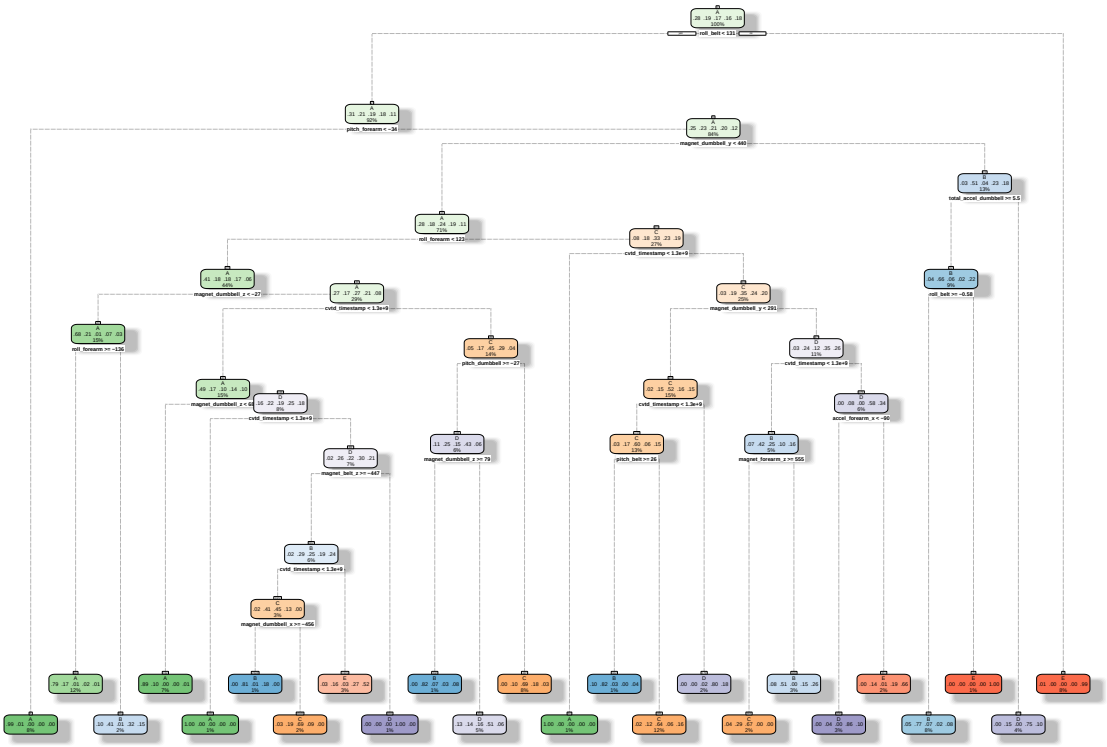



```
corMatrix <- cor(rawData[3:54])
corrplot(corMatrix, method='color', order='FPC', type='lower')
```



From the Heatmap we can see that Normalization are not needed for the data, and from the correlation we can see a cluster of positively correlated data on upper and bottom of the columns, thus we could try to use Decision Tree. ## Modelling data

```
library(rpart)
library(rattle)
decTreeFit <- rpart(classe~., data=training, method='class')
fancyRpartPlot(decTreeFit)
```




```
decTreeTrain <- predict(decTreeFit, newdata = training, type='class')
confusionMatrix(data=decTreeTrain, reference = training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3858  418   25   38   20
##          B  149 1648  103  254  285
##          C   56  452 2292  344  306
##          D   99  213  133 1596  200
##          E   23  117   14  180 1895
##
## Overall Statistics
##
##                Accuracy : 0.767
##                  95% CI : (0.7601, 0.7738)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7051
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9219   0.5787   0.8929   0.6617   0.7003
## Specificity            0.9524   0.9334   0.9047   0.9476   0.9722
## Pos Pred Value         0.8851   0.6757   0.6643   0.7122   0.8502
## Neg Pred Value         0.9684   0.9023   0.9756   0.9346   0.9351
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2621   0.1120   0.1557   0.1084   0.1288
## Detection Prevalence   0.2962   0.1657   0.2344   0.1523   0.1514
## Balanced Accuracy      0.9371   0.7560   0.8988   0.8046   0.8362
```

```
decTreePred <- predict(decTreeFit, newdata = testing, type='class')
confusionMatrix(data=decTreePred, reference = testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1268  158   10   15    7
##          B   55  530   39   97   96
##          C   22  147  745  144   97
##          D   44   70   60  502   78
##          E    6   44    1   46  623
##
## Overall Statistics
##
##                Accuracy : 0.748
##                  95% CI : (0.7356, 0.7601)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6809
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9090   0.5585   0.8713   0.6244   0.6915
## Specificity            0.9459   0.9274   0.8987   0.9385   0.9758
## Pos Pred Value         0.8697   0.6487   0.6450   0.6658   0.8653
## Neg Pred Value         0.9631   0.8975   0.9707   0.9272   0.9336
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2586   0.1081   0.1519   0.1024   0.1270
## Detection Prevalence   0.2973   0.1666   0.2355   0.1538   0.1468
## Balanced Accuracy      0.9274   0.7430   0.8850   0.7815   0.8336
```

We can see that Decision Tree giving an accuracy of 74% which is not enough, we could tuning the model predictor or we can change the algorithm with random forest which is an ensemble method of decision Tree sacrificing the explainability

```
library(randomForest)
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
randForestFit <- train(classe~., data=training, method='rf', trControl=controlRF)
randForestTrain <- predict(randForestFit, newdata = training)
confusionMatrix(data=randForestTrain, reference = training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
randForestPred <- predict(randForestFit, newdata = testing)
confusionMatrix(data=randForestPred, reference = testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394   12    0    0    0
##          B    1  934    4    0    0
##          C    0    3  851    2    0
##          D    0    0    0  801    3
##          E    0    0    0    1  898
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9922, 0.9965)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9842   0.9953   0.9963   0.9967
## Specificity            0.9966   0.9987   0.9988   0.9993   0.9998
## Pos Pred Value         0.9915   0.9947   0.9942   0.9963   0.9989
## Neg Pred Value         0.9997   0.9962   0.9990   0.9993   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1905   0.1735   0.1633   0.1831
## Detection Prevalence   0.2867   0.1915   0.1746   0.1639   0.1833
## Balanced Accuracy      0.9979   0.9915   0.9970   0.9978   0.9982
```

Random forest algorithm give an accuracy of 99% which is really good, thus we choose random forest algorithm as the clustering method. ## Predicting Data

```
finalModelResult <- predict(randForestFit, newdata=predicting)
finalModelResult
```

```
##  [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
```