Practical Machine Learning project

Loading required packages and Data

```
library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Loading required package: ggplot2

library(ggplot2)
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.5.3

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

test <- read.csv('pml-testing.csv')
train <- read.csv('pml-training.csv')
```

Cleaning & Exploration

```
head(names(train[,colSums(is.na(train))!=0])) #some names for the variables
that include NA values.

## [1] "max_roll_belt"        "max_picth_belt"       "min_roll_belt"
## [4] "min_pitch_belt"       "amplitude_roll_belt"  "amplitude_pitch_belt"
```

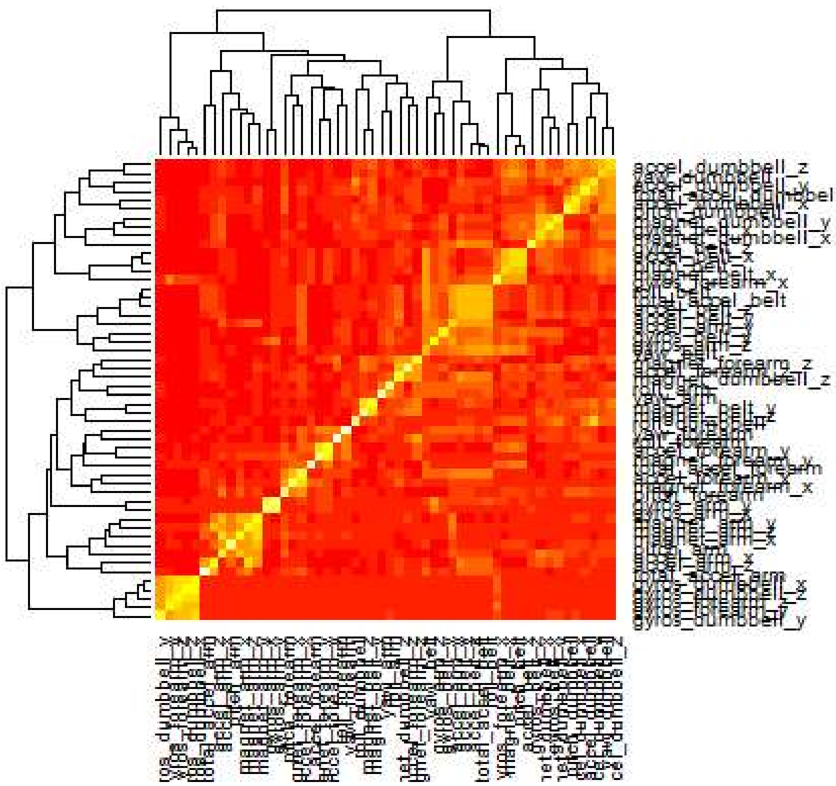
The dataset includes variables with a lot of missing values(NA) which seem to be inapropriate to use knn imputation as only some of the data in the variables is available for use. Also we can deduce that variables with user name, timestamp, index, window will not contribute much to the classification problem.

```
subsettrain<-train[,(colSums(is.na(train)) == 0)] #remove NA values for
training set
subsettest<-test[,(colSums(is.na(train)) == 0)] #remove NA values for test
set
subseting <-
!grepl('^X|user|window|kurtosis|skewness|timestamp|max_yaw|min_yaw|amplitude'
```

```
,names(subsettrain))
trainclean <- subsettrain[,subseting] #remove other missing/invalid variables (training)
testclean <- subsettest[,subseting] #remove other missing/invalid variables (test)
```

One other aspect to consider when building classifier is practicality of the classifier. We still have 53 variables and a lot of observation which could be troublesome as it takes too much time to train a classifier. What we can try is removing some variables that has high correlation which each other.

```
heatmap(abs(cor(trainclean[1:52])))
```



We can see from the heatmap of variables that some variables have big correlation, suggesting that some variables can be removed. we will remove some variables that has correlation over .7

```
training <- trainclean[,-findCorrelation(cor(trainclean[, 1:52]), cutoff = .7)]
testing <- testclean[,-findCorrelation(cor(trainclean[, 1:52]), cutoff = .7)]
```

Training classifier

As mentioned above, the data set has very big dimension, which can be very hard to work with. Thus for this classifier, Principle Component Analysis will be used to reduce dimension of the data, then classification method will be implemented.

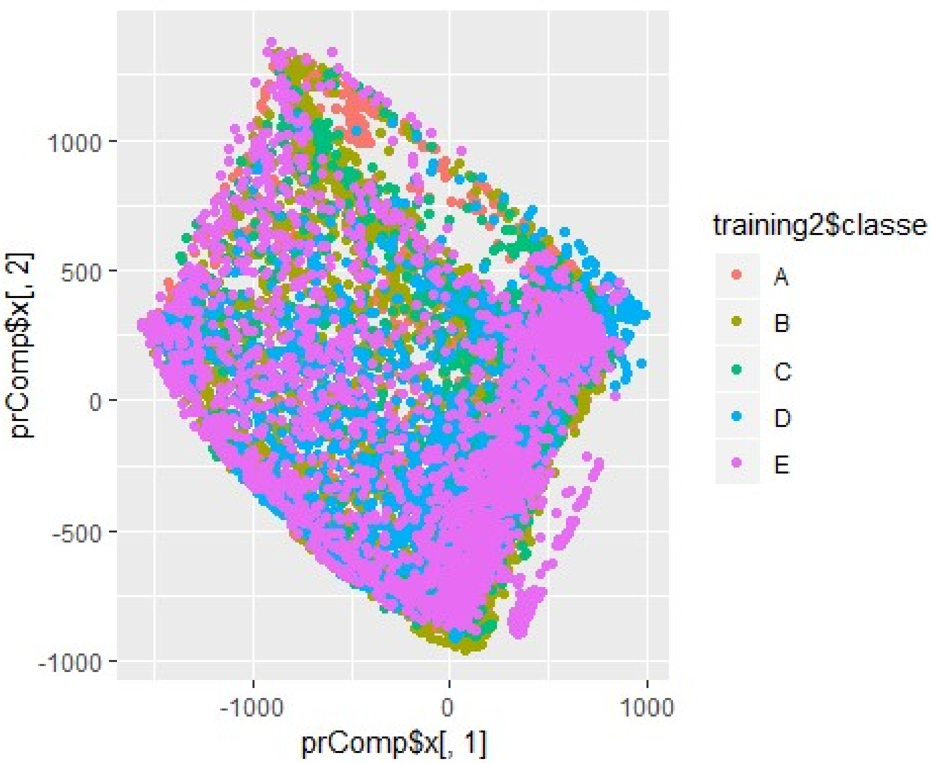
PCA

Before we implement any method, we want to divide the training data into another training set and validation set.

```
set.seed(123123)
inTrain <- createDataPartition(training$classe,p=0.7,list=FALSE)
training2 <- training[inTrain,]
validation <- training[-inTrain,]
```

PCA dimension reduction and visualization for first two components (just for the visualization).

```
prComp <- prcomp(training2[,1:30])
qplot(prComp$x[,1],prComp$x[,2],colour=training2$classe)
```




```
summary(prComp)

## Importance of components:
##                             PC1      PC2      PC3      PC4       PC5
## Standard deviation     559.2971 492.3318 367.4082 344.3714 231.58249
## Proportion of Variance   0.3327   0.2578   0.1436   0.1262   0.05705
## Cumulative Proportion    0.3327   0.5906   0.7342   0.8603   0.91739
##                              PC6      PC7      PC8      PC9      PC10
## Standard deviation     146.99278 129.7190 96.22364 84.86109 75.23063
## Proportion of Variance   0.02298   0.0179  0.00985  0.00766  0.00602
```

```
## Cumulative Proportion    0.94037   0.9583  0.96812  0.97578  0.98180
##                              PC11     PC12     PC13     PC14     PC15
## Standard deviation     66.36308 61.51035 51.90851 44.70150 41.04644
## Proportion of Variance  0.00468  0.00402  0.00287  0.00213  0.00179
## Cumulative Proportion   0.98649  0.99051  0.99338  0.99551  0.99730
##                              PC16     PC17     PC18     PC19    PC20    PC21
## Standard deviation     29.80230 24.20405 23.23210 19.13179 8.06356 7.03914
## Proportion of Variance  0.00094  0.00062  0.00057  0.00039 0.00007 0.00005
## Cumulative Proportion   0.99824  0.99887  0.99944  0.99983 0.99990 0.99995
##                             PC22  PC23   PC24   PC25   PC26   PC27   PC28
## Standard deviation     6.34857 2.136 0.8351 0.4655 0.3929 0.3572 0.2207
## Proportion of Variance 0.00004 0.000 0.0000 0.0000 0.0000 0.0000 0.0000
## Cumulative Proportion  0.99999 1.000 1.0000 1.0000 1.0000 1.0000 1.0000
##                          PC29    PC30
## Standard deviation     0.1382 0.04294
## Proportion of Variance 0.0000 0.00000
## Cumulative Proportion  1.0000 1.00000
```

We of course need more variable then two, 5 components seem to be enough to explain over 90% of variation

Random forest classifier

Random forest is strong classifier which trains a lot of trees that votes for a class for each observation. Random forest method will be used for this classification.

Our classifier will be controlled for gernalizability, as we optimize the classifier using cross-validation estimation for generalization error.

Train random forest classifier after preprocessing the principle component analysis.

```
set.seed(123123)
train_control <- trainControl(method="cv", number=10)
pca <- preProcess(training2[,1:30],method='pca',pcaComp=5)
trainpc <- predict(pca,training2[,1:30])
trainpc$classe <- training2$classe
modpca <- train(classe~.,data=trainpc,method='rf', trControl=train_control)
```

Valdiation set then will be used to find the accuracy and confusion matrix.

```
testpc <- predict(pca,validation[1:30])
testpc$classe <- validation$classe
table(validation$classe,predict(modpca,testpc)) #confusion matrix

##
##         A    B    C    D    E
##    A 1532   47   43   35   17
##    B   62  916   69   26   66
##    C   33   44  886   39   24
```

```
##    D    35    21    63   820    25
##    E    27    49    38    39   929
```

```r
mean(validation$classe==predict(modpca,testpc)) #accuracy
```

```
## [1] 0.8637213
```

Gradient boosting

We try with bgm method in r with processed data (PCA)

```r
set.seed(12345)
modgb <- train(classe~.,
data=trainpc,method='gbm',trControl=train_control,verbose=FALSE)
confusionMatrix(validation$classe,predict(modgb,testpc))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1293   89  144   85   63
##          B  127  624  174   75  139
##          C  110  118  637   95   66
##          D   71   93  184  544   72
##          E   86  138  134  109  615
## 
## Overall Statistics
## 
##                Accuracy : 0.6309
##                  95% CI : (0.6184, 0.6433)
##     No Information Rate : 0.2867
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.5332
## 
##  Mcnemar's Test P-Value : 1.468e-14
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7664   0.5876   0.5004  0.59912   0.6440
## Specificity            0.9092   0.8932   0.9157  0.91561   0.9053
## Pos Pred Value         0.7724   0.5478   0.6209  0.56432   0.5684
## Neg Pred Value         0.9064   0.9077   0.8691  0.92603   0.9292
## Prevalence             0.2867   0.1805   0.2163  0.15429   0.1623
## Detection Rate         0.2197   0.1060   0.1082  0.09244   0.1045
## Detection Prevalence   0.2845   0.1935   0.1743  0.16381   0.1839
## Balanced Accuracy      0.8378   0.7404   0.7080  0.75737   0.7746
```

Accuracy has been lowered (63 percent with validation set). We will keep our Random Forest classifier.