Synopsis

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, the 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.

Data

Data is downloaded from used and read as csv files

```
set.seed(1)
library(lattice)
library(ggplot2)
library(caret)
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
training.url <-
"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv"
testing.url <-
"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training.file <- "pml-training.csv"
testing.file <- "pml-testing.csv"
# download.file(training.url, training.file)
download.file(testing.url,
# testing.file)
read.pml <- function(x) {
    read.csv(x, na.strings = c("", "NA", "#DIV/0!"))
}
training <- read.pml(training.file)
training.rows = nrow(training)
training.cols = ncol(training)
```

```
head(training)
```

Data has 19622 rows and 160 column, first 7 column present participant name, time window and time stamp data, they are remove as predictors. Sereval columns are mainly “NA” or “#DIV/0!” values, only columns with less than 10% na values are allowed as predictors.

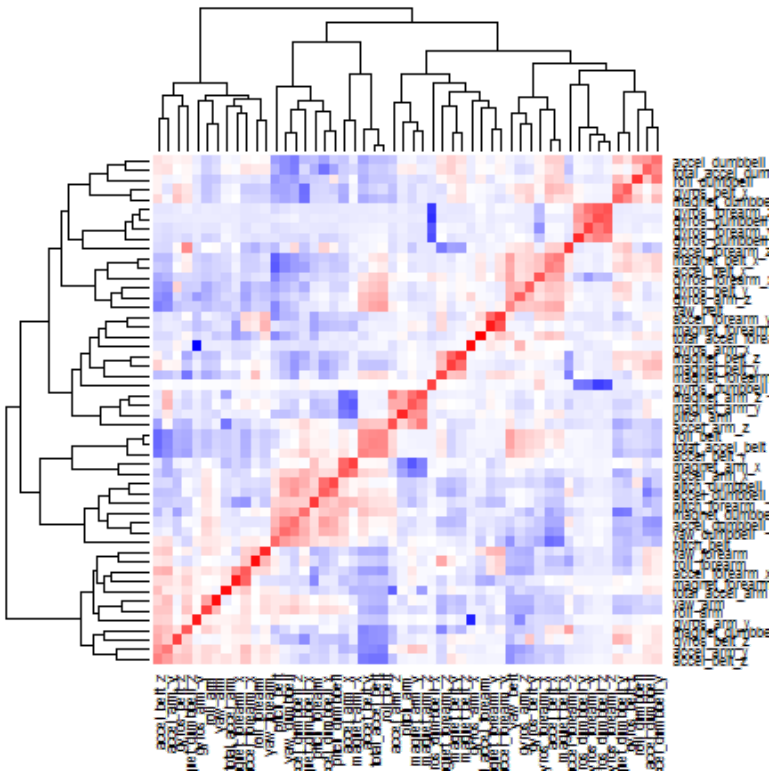
```
# drop columns 1:7
training[1:7] <- list(NULL)
# keep columns with less than % na values
training <- training[, colMeans(is.na(training)) < 0.1]
training.cols = ncol(training)
```

The size of columns are reduced to 53. Also near zero variance predictors are removed, but are is no pressence of near zero variance predictors.

```
nearZeroColumns <- nearZeroVar(training, saveMetrics = TRUE)
training <- training[, nearZeroColumns$nzv == FALSE]
training.cols = ncol(training)
```

Plot the correlation between predictors.

```
pred.corr <- cor(subset(training, select = -c(classe)))
heatmap(pred.corr, col = colorRampPalette(c("blue", "white", "red"))
(n = 199))
```



In the heat map of the correlation matrix, most of predictors do not exhibit high degree of correlation.

Data

Data is split in train and test sets ussing “classe” to perform cross validation.

```
training.train.index <- createDataPartition(training$classe, p = 0.7,
list = FALSE)
training.train <- training[training.train.index, ]
training.test <- training[-training.train.index, ]
```

Model

Random forest has been choosed as model algorithm. Positive factors on RF are the speed, the interpretability and dealing with overfitting. Therefore, it's always better to use more trees, memory and computational power allowing.

The algorithm allows for good in-training estimates of variable importance and generalization error, which largely eliminates the need for a separate validation stage, but cross validation has been used as a prudent mesure before submitting the testing results. The FR has been trained with 70% of the training data and 500 trees.

```
model.rf <- randomForest(classe ~ ., data = training.train, ntree =
500)
training.train.prediction <- predict(model.rf, newdata =
training.train)
confusionMatrix(training.train.prediction, training.train$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (1, 1)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    1.000    1.000    1.000    1.000
## Specificity             1.000    1.000    1.000    1.000    1.000
## Pos Pred Value          1.000    1.000    1.000    1.000    1.000
## Neg Pred Value          1.000    1.000    1.000    1.000    1.000
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.174    0.164    0.184
## Detection Prevalence    0.284    0.193    0.174    0.164    0.184
## Balanced Accuracy       1.000    1.000    1.000    1.000    1.000
```

We have an accuracy of 100%. We hope that there is not too much overfitting. We now test our model on our 30% test set from the training data.

```
training.test.prediction <- predict(model.rf, newdata =
training.test)
confusionMatrix(training.test.prediction, training.test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    2    0    0    0
##          B    1 1136    7    0    0
##          C    1    1 1018    5    0
##          D    0    0    1  957    2
##          E    2    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.994, 0.997)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.998    0.997    0.992    0.993    0.998
## Specificity             1.000    0.998    0.999    0.999    0.999
## Pos Pred Value          0.999    0.993    0.993    0.997    0.996
## Neg Pred Value          0.999    0.999    0.998    0.999    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.193    0.173    0.163    0.184
## Detection Prevalence    0.284    0.194    0.174    0.163    0.184
## Balanced Accuracy       0.999    0.998    0.995    0.996    0.999
```

We have an estimated out of sample error rate of 0.56% (1-Accuracy). This is a good result.

Prediction

We are ready to read the testing file and predict the values.

```
testing <- read.pml(testing.file)
testing.prediction<-predict(model.rf,newdata=testing)
testing.prediction
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")

write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)

  }
}
pml_write_files(testing.prediction)
```

Conclusion

Random forest looks like good choice to predict the data, it would be interesting to try different configuration params for training or using another algorithms.