# Final_project

CeciliaHermosilla

24/6/2021

## Download the data sets

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

#### Create R objects with datasets

```
training <- read.csv("training.csv")
testing <- read.csv("testing.csv")
```

## Explore the training data set

```
names(training) #Variables
str(training) #Summary of the object and variables
```

## Partition training data set to perform cross-validation

75% of the data will be kept on the training subset of the training set called "data_training" and the rest, in the testing subset called "test_training".

```
library("caret")
```

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

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

```
set.seed(1000)

intrain <- createDataPartition(y=training$classe,p=0.75,list=FALSE)
data_training <- training[intrain,]
test_training <- training[-intrain,]
```

## Transformation of variables

### Find variables to delete

```
#Delete variables with large missing data and number not-displayed (#DIV characters)
and delete them
miss <- data.frame(missing=colSums(is.na(data_training)))

#Check how many NA there are on every variable and delete them
rownames(miss) <- 1:nrow(miss)
miss_cols <- as.numeric(rownames(subset(miss,subset=missing!=0)))
data_training <- data_training[,-miss_cols]

#Find variables with #DIV characters due to errors in data
weird <- grep(pattern="#DIV.*",data_training)
```

A function was created to transform variables and delete columns that would not be helpful

```
#Transform outcome into factor variable
data_training$classe <- as.factor(data_training$classe)

#Function created
library("magrittr")
preproc_fx <- function(z) {
  z$cvtd_timestamp <- z$cvtd_timestamp %>%
    as.factor() %>%
    sapply(FUN = unclass)
  z$new_window <- z$new_window %>%
    as.factor() %>%
    sapply(FUN = unclass) %>%
    as.numeric()
  z$user_name <- z$user_name %>%
    as.factor() %>%
    sapply(FUN = unclass)

  z <- z[,-weird]

  z$X <- NULL
  new_data <<- z
}
```

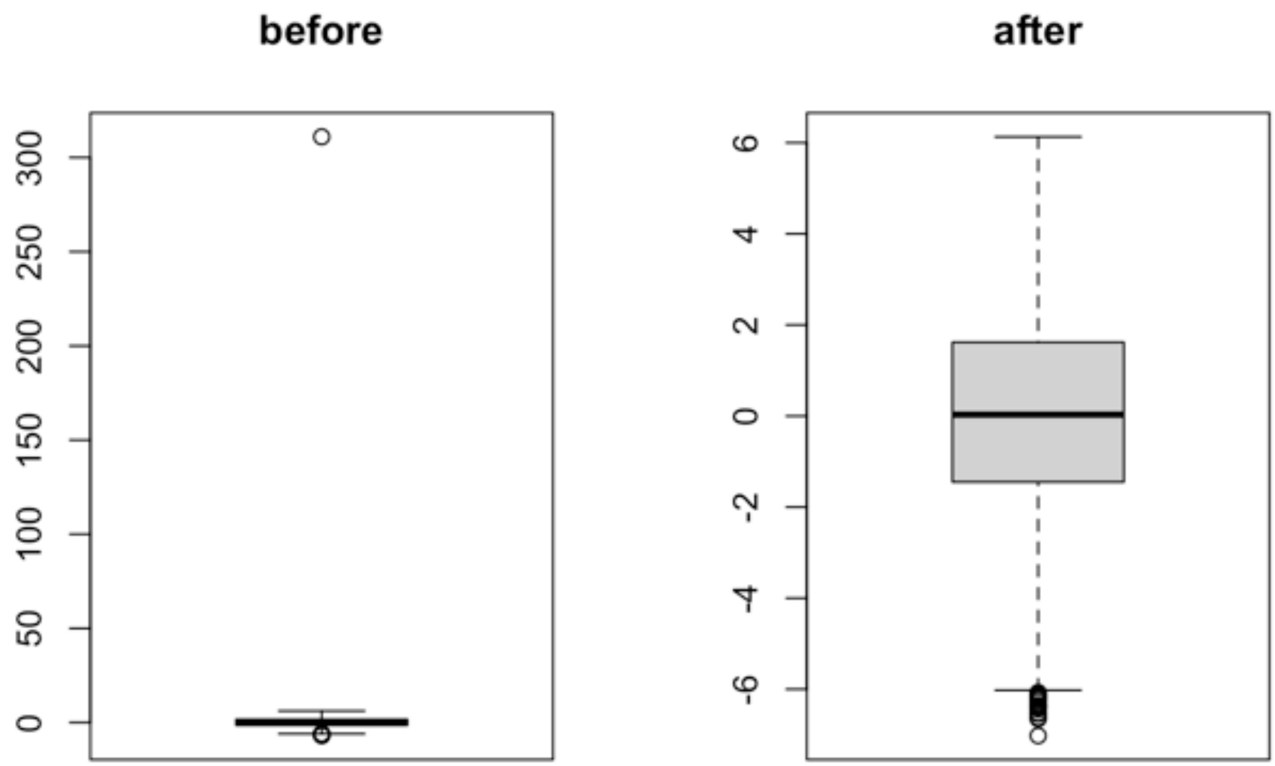
Apply function to the training subset of the training set to transform the data

```
preproc_fx(data_training)
data_training <- new_data
```

```
#Plot the data
par(mfrow=c(1,2))
boxplot(data_training$gyros_forearm_y,main="before")

#Delete row with outliers and apparently many errors
data_training <- data_training[-4031,]

#Plot the data again
boxplot(data_training$gyros_forearm_y, main="after")
```




## Establish correlation between variables

```
#Look for the correlation between all variables except the outcome
correlation <- abs(cor(data_training[,-59]))
diag(correlation) <- 0

#Establish variables >85% correlated
variables_correlated <- data.frame(which(correlation>0.85,arr.ind=T))
variables_correlated
```

```
##                     row col
## total_accel_belt    10   7
## accel_belt_y        15   7
## accel_belt_z        16   7
## accel_belt_x        14   8
## magnet_belt_x       17   8
## roll_belt            7  10
## accel_belt_y.1      15  10
## accel_belt_z.1      16  10
## pitch_belt           8  14
## magnet_belt_x.1     17  14
## roll_belt.1          7  15
## total_accel_belt.1  10  15
## accel_belt_z.2      16  15
## roll_belt.2          7  16
## total_accel_belt.2  10  16
## accel_belt_y.2      15  16
## pitch_belt.1         8  17
## accel_belt_x.1      14  17
## gyros_arm_y         25  24
## gyros_arm_x         24  25
```

Verification of the PCA for the seven variables >85% correlated between each other.

```
#Perform PCA in seven variables highly correlated to each other
featurePlot(x=data_training[,c(7:8,10,14:17)],y=data_training$classe,plot="pairs")
prePROC1 <- prcomp(data_training[,c(7:8,10,14:17)],center=TRUE,scale=TRUE)

summary(prePROC1)

#Assign color to values in outcome
data_training$color <- data_training$classe %>%
  gsub(pattern= "A", replacement="blue") %>%
  gsub(pattern="B", replacement="green") %>%
  gsub(pattern="C", replacement="orange") %>%
  gsub(pattern="D", replacement="magenta") %>%
  gsub(pattern="E", replacement="gray")

#Plot the PCA analysis
plot(prePROC1$x[,1],prePROC1$x[,2],
     col=alpha(data_training$color,0.2),
     xlab="PC1",ylab="PC2",pch=20)

#Delete the variable just created
data_training$color <- NULL
```

# Fit models

**Predict using PCA with the seven variables**

The seven variables will be used through PCA and caret package to check how prediction is done.

```
names(data_training[,c(7:8,10,14:17)])

#Fit a model preprocessing with PCA and using random forest
fit1 <- train(classe ~ roll_belt + pitch_belt + total_accel_belt + accel_belt_x
              + accel_belt_y + accel_belt_z + magnet_belt_x,
              preProcess="pca",method="rf",data=data_training)
```

```
#Transform data on the testing subset of the training set
test_training$classe <- as.factor(test_training$classe)
test_training <- test_training[,-miss_cols]
preproc_fx(test_training)
test_training <- new_data
```

```
#Predict using the first model fit
fit1_result <- confusionMatrix(test_training$classe, predict(fit1,test_training))
```

**The accuracy was of 0.47**, not too high even thought random forest was used.

# Another fit to compare building one tree and using different variables than before

Other variables were chosen including one of the past seven, "total_accel_belt" and others chosen from the pool of variables with little correlation between each other (less than 30%).

```
variables_little_correlated <- data.frame(which(correlation<0.3,arr.ind=T))
head(variables_little_correlated)
```

```
##                       row col
## user_name               1   1
## raw_timestamp_part_1    2   1
## raw_timestamp_part_2    3   1
## new_window              5   1
## num_window              6   1
## roll_belt               7   1
```

```
#Fit the model
fit2 <- train(classe ~ total_accel_belt + gyros_belt_z + roll_dumbbell + accel_arm_y
              + gyros_forearm_y + magnet_belt_z + pitch_forearm + magnet_forearm_x,
              method="rpart",data=data_training)

#Predict on testing subset of the training set
fit2_result <- confusionMatrix(test_training$classe, predict(fit2,test_training))
```

**The accuracy was of 0.49**, if this was with one tree, it will be larger for many trees, so the same variables will be used with random forest method.

# Using random forest on the same variables of second fit

```
#Fit the model
fit3 <- train(classe ~ total_accel_belt + gyros_belt_z + roll_dumbbell + accel_arm_y
+ gyros_forearm_y + magnet_belt_z + pitch_forearm + magnet_forearm_x,
              method="rf",data=data_training)

#Predict on testing subset of the training set
fit3_result <- confusionMatrix(test_training$classe, predict(fit3,test_training))
fit3_result
```

Confusion Matrix and Statistics

```
                  Reference
```

```
##   Prediction    A   B   C   D   E
## 1          A 1318  22  23  24   8
## 2          B   48 851  38   7   5
## 3          C   14  33 781  19   8
## 4          D   27  10  49 714   4
## 5          E    5   3   3   9 881
```

Overall Statistics

```
           Accuracy : 0.9268
             95% CI : (0.9191, 0.9339)
No Information Rate : 0.2879
P-Value [Acc > NIR] : < 2.2e-16

              Kappa : 0.9074
```

Mcnemar's Test P-Value : 0.0004798

Statistics by Class:

```
##                Measure Class_A Class_B Class_C Class_D Class_E
## 1          Sensitivity  0.9334  0.9260  0.8736  0.9237  0.9724
## 2          Specificity  0.9779  0.9754  0.9815  0.9782  0.9950
## 3       Pos Pred Value  0.9448  0.8967  0.9135  0.8881  0.9778
## 4       Neg Pred Value  0.9732  0.9828  0.9721  0.9856  0.9938
## 5           Prevalence  0.2879  0.1874  0.1823  0.1576  0.1847
## 6       Detection Rate  0.2688  0.1735  0.1593  0.1456  0.1796
## 7 Detection Prevalence  0.2845  0.1935  0.1743  0.1639  0.1837
## 8    Balanced Accuracy  0.9557  0.9507  0.9276  0.9509  0.9837
```

This model will be selected since it has **accuracy = 0.93**, 95% CI[0.92-0.93], Specificity ranging 0.97-0.99 and Sensitivity ranging 0.87-0.93 even though the **out of sample error** will be higher but since the testing set has no classe assigned it cannot be proven exactly how much.

# Predicting on the testing set with the model selected

```
#Transform the data as before
data_training <- data_training[,-miss_cols]
preproc_fx(testing)
testing <- new_data

#Predict using the model
prediction <- predict(fit3,testing)
prediction
```

[1] B A A A A E D B A A B C B A E E A B A B Levels: A B C D E