financial_eco_MHP

Marc-Henri Pélet

24/05/2020

Dataset : http://groupware.les.inf.puc-rio.br/har

Processing

```
trainer.raw <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"))
tester.raw <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"))
```

Data processing : the beginning

Look at the dimensions & head of the dataset to get an idea

```
# Res 1
dim(trainer.raw)
```

```
## [1] 19622   160
```

```
# Res 2 - excluded because over the required amount
# head(trainer.raw)
# Res 3 - excluded because over the required amount
#str(trainer.raw)
# Res 4 - excluded because over the required amount
#summary(trainer.raw)
```

We remove the empty data

```
maxNAPercentage = 20
maxNACount <- nrow(trainer.raw) / 100 * maxNAPercentage
removeColumns <- which(colSums(is.na(trainer.raw) | trainer.raw=="") > maxNACount)
trainer.cleaned01 <- trainer.raw[,-removeColumns]
tester.cleaned01 <- tester.raw[,-removeColumns]
```

As we do not need the related data, we take out these too

```
removeColumns <- grep("timestamp", names(trainer.cleaned01))
trainer.cleaned02 <- trainer.cleaned01[,-c(1, removeColumns )]
tester.cleaned02 <- tester.cleaned01[,-c(1, removeColumns )]
```

```
classeLevels <- levels(trainer.cleaned02$classe)
trainer.cleaned03 <- data.frame(data.matrix(trainer.cleaned02))
trainer.cleaned03$classe <- factor(trainer.cleaned03$classe, labels=classeLevels)
tester.cleaned03 <- data.frame(data.matrix(tester.cleaned02))
```

We have now the final datas

```
trainer.cleaned <- trainer.cleaned03
tester.cleaned <- tester.cleaned03
```

Analysis

```
set.seed(15691997)
library(caret)
clref <- which(names(trainer.cleaned) == "classe")
partition <- createDataPartition(y=trainer.cleaned$classe, p=0.7, list=FALSE)
trainer.subSetTrain <- trainer.cleaned[partition, ]
trainer.subSetTest <- trainer.cleaned[-partition, ]
```

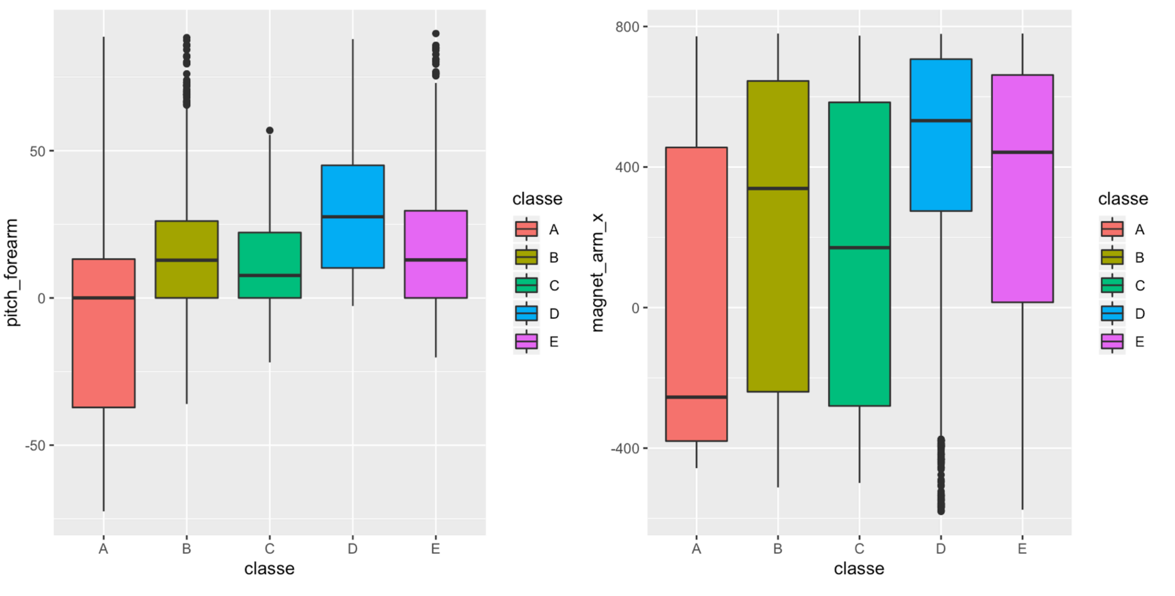
```
correlations <- cor(trainer.subSetTrain[, -clref], as.numeric(trainer.subSetTrain$classe))
optimalcorr <- subset(as.data.frame(as.table(correlations)), abs(Freq)>0.3)
optimalcorr
```

```
##              Var1 Var2    Freq
## 44 pitch_forearm    A 0.33046
```

Normally, the best correlations are not above 0.3

We now plot this in order to have a better visualisation

```
library(Rmisc)
library(ggplot2)
p1 <- ggplot(trainer.subSetTrain, aes(classe,pitch_forearm)) +
  geom_boxplot(aes(fill=classe))
p2 <- ggplot(trainer.subSetTrain, aes(classe, magnet_arm_x)) +
  geom_boxplot(aes(fill=classe))
multiplot(p1,p2,cols=2)
```

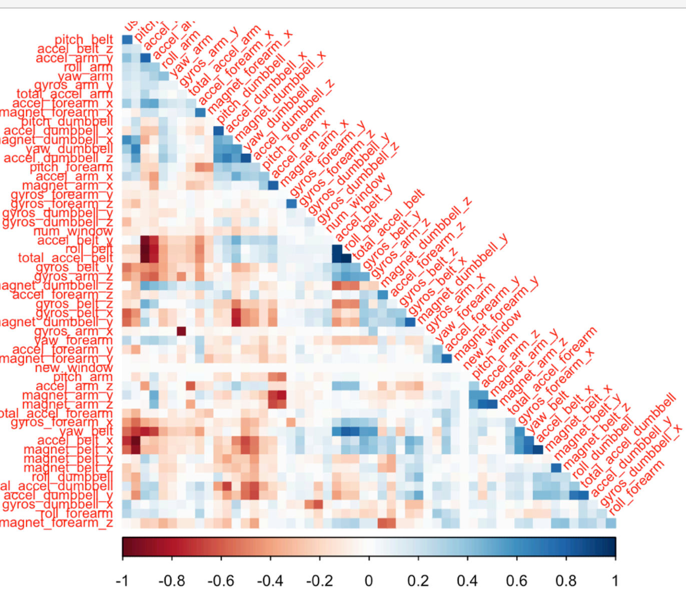


Models

Let's identify variables with high correlations amongst each other

We will then check if these modifications make the model more accurate

```
library(corrplot)
cormat <- cor(trainer.subSetTrain[, -clref])
highcor <- findCorrelation(cormat, cutoff=0.9, exact=TRUE)
excludeColumns <- c(highcor, clref)
corrplot(cormat, method="color", type="lower", order="hclust", tl.cex=0.70, tl.col="red", tl.srt = 45, diag = FAL
SE)
```



Therefore, some data seem corrolated with each other. We should then exclude them in order to have a better hindsight on the models.

```
pcaPreProcess.all <- preProcess(trainer.subSetTrain[, -clref], method = "pca", thresh = 0.99)
trainer.subSetTrain.pca.all <- predict(pcaPreProcess.all, trainer.subSetTrain[, -clref])
trainer.subSetTest.pca.all <- predict(pcaPreProcess.all, trainer.subSetTest[, -clref])
tester.pca.all <- predict(pcaPreProcess.all, tester.cleaned[, -clref])
pcaPreProcess.subset <- preProcess(trainer.subSetTrain[, -excludeColumns], method = "pca", thresh = 0.99)
trainer.subSetTrain.pca.subset <- predict(pcaPreProcess.subset, trainer.subSetTrain[, -excludeColumns])
trainer.subSetTest.pca.subset <- predict(pcaPreProcess.subset, trainer.subSetTest[, -excludeColumns])
tester.pca.subset <- predict(pcaPreProcess.subset, tester.cleaned[, -clref])
```

Now we use the Random Forest trainer model with 200 trees

```
library(randomForest)
ntree <- 150
start <- proc.time()
rfMod.cleaned <- randomForest(
  x=trainer.subSetTrain[, -clref],
  y=trainer.subSetTrain$classe,
  xtest=trainer.subSetTest[, -clref],
  ytest=trainer.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE)
proc.time() - start
```

```
##    user  system elapsed
##  79.297   2.508  82.692
```

```
start <- proc.time()
rfMod.exclude <- randomForest(
  x=trainer.subSetTrain[, -excludeColumns],
  y=trainer.subSetTrain$classe,
  xtest=trainer.subSetTest[, -excludeColumns],
  ytest=trainer.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE)
proc.time() - start
```

```
##    user  system elapsed
##  75.741   3.454  82.986
```

```
start <- proc.time()
rfMod.pca.all <- randomForest(
  x=trainer.subSetTrain.pca.all,
  y=trainer.subSetTrain$classe,
  xtest=trainer.subSetTest.pca.all,
  ytest=trainer.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE)
proc.time() - start
```

```
##    user  system elapsed
##  76.955   3.220  82.294
```

```
start <- proc.time()
rfMod.pca.subset <- randomForest(
  x=trainer.subSetTrain.pca.subset,
  y=trainer.subSetTrain$classe,
  xtest=trainer.subSetTest.pca.subset,
  ytest=trainer.subSetTest$classe,
  ntree=ntree,
  keep.forest=TRUE,
  proximity=TRUE)
proc.time() - start
```

```
##    user  system elapsed
##  74.704   2.970  78.497
```

#Model examination

We will check the accuracies of each of the 4 models

```
rfMod.cleaned
```

```
##
## Call:
##  randomForest(x = trainer.subSetTrain[, -clref], y = trainer.subSetTrain$classe,      xtest = trainer.subSetTe
st[, -clref], ytest = trainer.subSetTest$classe,      ntree = ntree, proximity = TRUE, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 150
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.32%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    1    0    0    0 0.0002560164
## B    6 2648    4    0    0 0.0037622272
## C    0    9 2386    1    0 0.0041736227
## D    0    0   16 2235    1 0.0075488455
## E    0    0    0    6 2519 0.0023762376
##                 Test set error rate: 0.27%
## Confusion matrix:
##      A    B    C   D    E  class.error
## A 1673    1    0   0    0 0.0005973716
## B    2 1136    1   0    0 0.0026338894
## C    0    3 1023   0    0 0.0029239766
## D    0    0    7 955    2 0.0093360996
## E    0    0    0   0 1082 0.0000000000
```

```
rfMod.cleaned.trainer.accuracy <- round(1-sum(rfMod.cleaned$confusion[, 'class.error']),3)
paste0("Accuracy trainer: ",rfMod.cleaned.trainer.accuracy)
```

```
## [1] "Accuracy trainer: 0.982"
```

```
rfMod.cleaned.tester.accuracy <- round(1-sum(rfMod.cleaned$test$confusion[, 'class.error']),3)
paste0("Accuracy tester: ",rfMod.cleaned.tester.accuracy)
```

```
## [1] "Accuracy tester: 0.985"
```

```
rfMod.exclude
```

```
##
## Call:
##  randomForest(x = trainer.subSetTrain[, -excludeColumns], y = trainer.subSetTrain$classe,      xtest = trainer
.subSetTest[, -excludeColumns], ytest = trainer.subSetTest$classe,      ntree = ntree, proximity = TRUE, keep.for
est = TRUE)
##                Type of random forest: classification
##                      Number of trees: 150
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.36%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    5 2649    4    0    0 0.003386005
## C    0   13 2382    1    0 0.005843072
## D    0    0   17 2233    2 0.008436945
## E    0    0    0    7 2518 0.002772277
##                 Test set error rate: 0.29%
## Confusion matrix:
##      A    B    C   D    E  class.error
## A 1673    0    0   0    1 0.0005973716
## B    0 1138    1   0    0 0.0008779631
## C    0    3 1022   1    0 0.0038986355
## D    0    0    9 953    2 0.0114107884
## E    0    0    0   0 1082 0.0000000000
```

```
rfMod.exclude.trainer.accuracy <- round(1-sum(rfMod.exclude$confusion[, 'class.error']),3)
paste0("Accuracy trainer: ",rfMod.exclude.trainer.accuracy)
```

```
## [1] "Accuracy trainer: 0.98"
```

```
rfMod.exclude.tester.accuracy <- round(1-sum(rfMod.exclude$test$confusion[, 'class.error']),3)
paste0("Accuracy tester: ",rfMod.exclude.tester.accuracy)
```

```
## [1] "Accuracy tester: 0.983"
```

```
rfMod.pca.all
```

```
##
## Call:
##  randomForest(x = trainer.subSetTrain.pca.all, y = trainer.subSetTrain$classe,      xtest = trainer.subSetTest
.pca.all, ytest = trainer.subSetTest$classe,      ntree = ntree, proximity = TRUE, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 150
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 2.43%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3881    9    6    8    2  0.00640041
## B   49 2583   24    2    0  0.02821670
## C    3   44 2331   16    2  0.02712855
## D    8    5  103 2129    7  0.05461812
## E    2    8   14   22 2479  0.01821782
##                 Test set error rate: 2.07%
## Confusion matrix:
##      A    B    C   D    E class.error
## A 1667    4    0   2    1 0.004181601
## B   24 1101   11   0    3 0.033362599
## C    2   13 1000   8    3 0.025341131
## D    1    0   30 929    4 0.036307054
## E    0    3    8   5 1066 0.014787431
```

```
rfMod.pca.all.trainer.accuracy <- round(1-sum(rfMod.pca.all$confusion[, 'class.error']),3)
paste0("Accuracy trainer: ",rfMod.pca.all.trainer.accuracy)
```

```
## [1] "Accuracy trainer: 0.865"
```

```
rfMod.pca.all.tester.accuracy <- round(1-sum(rfMod.pca.all$test$confusion[, 'class.error']),3)
paste0("Accuracy tester: ",rfMod.pca.all.tester.accuracy)
```

```
## [1] "Accuracy tester: 0.886"
```

```
rfMod.pca.subset
```

```
##
## Call:
##  randomForest(x = trainer.subSetTrain.pca.subset, y = trainer.subSetTrain$classe,      xtest = trainer.subSetT
est.pca.subset, ytest = trainer.subSetTest$classe,      ntree = ntree, proximity = TRUE, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 150
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 2.76%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3871   13    8   12    2 0.008960573
## B   56 2560   39    3    0 0.036869827
## C   11   37 2322   16   10 0.030884808
## D    8    3  101 2132    8 0.053285968
## E    3    8   23   18 2473 0.020594059
##                 Test set error rate: 2.04%
## Confusion matrix:
##      A    B   C   D    E class.error
## A 1663    2   5   3    1 0.006571087
## B   17 1111   8   1    2 0.024582968
## C    4   13 999   7    3 0.026315789
## D    4    0  36 922    2 0.043568465
## E    0    3   3   6 1070 0.011090573
```

```
rfMod.pca.subset.trainer.accuracy <- round(1-sum(rfMod.pca.subset$confusion[, 'class.error']),3)
paste0("Accuracy trainer: ",rfMod.pca.subset.trainer.accuracy)
```

```
## [1] "Accuracy trainer: 0.849"
```

```
rfMod.pca.subset.tester.accuracy <- round(1-sum(rfMod.pca.subset$test$confusion[, 'class.error']),3)
paste0("Accuracy tester: ",rfMod.pca.subset.tester.accuracy)
```

```
## [1] "Accuracy tester: 0.888"
```

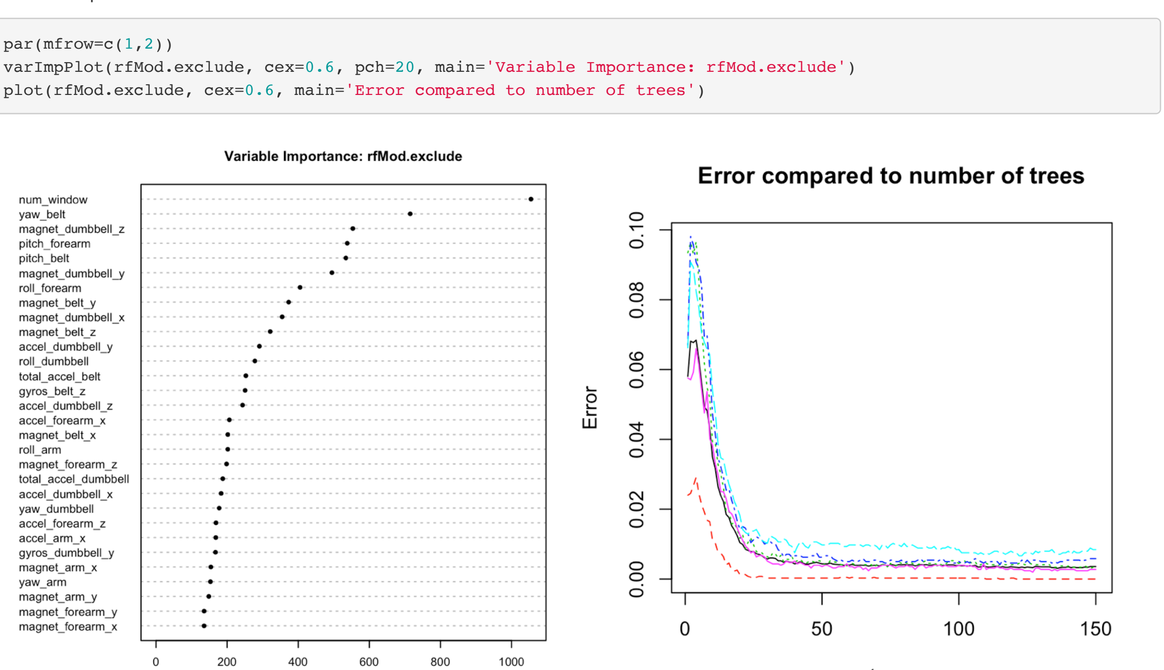
#Conclusion

The `rfMod.exclude` performs better then the 'rfMod.cleaned'

We will thereofre choose the `rfMod.exclude` model as the best model to use for predicting the test set as it has the higher accuracy and the lowest error rate

We will now plot this model

```
par(mfrow=c(1,2))
varImpPlot(rfMod.exclude, cex=0.6, pch=20, main='Variable Importance: rfMod.exclude')
plot(rfMod.exclude, cex=0.6, main='Error compared to number of trees')
```




```
par(mfrow=c(1,1))
```

#Results

We will run all four models for this final test

```
predictions <- t(cbind(
    exclude=as.data.frame(predict(rfMod.exclude, tester.cleaned[, -excludeColumns]), optional=TRUE),
    cleaned=as.data.frame(predict(rfMod.cleaned, tester.cleaned), optional=TRUE),
    pcaAll=as.data.frame(predict(rfMod.pca.all, tester.pca.all), optional=TRUE),
    pcaExclude=as.data.frame(predict(rfMod.pca.subset, tester.pca.subset), optional=TRUE)
))
predictions
```

```
##            1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20
## exclude    "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B" "B"
## cleaned    "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B" "B"
## pcaAll     "B" "A" "C" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B" "B"
## pcaExclude "B" "A" "C" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B" "B"
```

As we can see, there are not a lot of change between the results in these models. However, due to better accuracy, and mostly lower error rate, we will stick with the `rfMod.exclude` model