Predicting "How well we do barbell lifts?" using data from Sports Devices

Sebastián Fuenzalida Garcés
August 2015

Executive Summary

Load Data

First we need to read the files (included in the repo). After an analysis of the data, to avoid problems with missing data I decided to consider as NA's: ["NA", "#DIV/0!", ""], considering them as moments when the user didn't do anything or problems with the devices gathering the data.

```
library(AppliedPredictiveModeling); library(caret); library(rattle); library(randomForest); library(doPa
registerDoParallel(cores=2)

train_data<-read.csv("pml-training.csv",header=TRUE,na.strings=c("NA","#DIV/0!",""))
test_data<-read.csv("pml-testing.csv",header=TRUE,na.strings=c("NA","#DIV/0!",""))
```

Relevant Data

Looking at the data we can classify the columns in 3 categories: -Useful Data: Variables that we will use in the model -Not Useful Data: Variables that don't have any relation with the classe -Data with too many NA's: variables where more than 80% of the rows are NA's so we are not going to include them in the model

Considering that we modify the train and test data to have only the useful columns:

```
#First 7 columns aren't useful for the model (name, time, window, etc...)
train_data<-train_data[,8:length(train_data)]
test_data<-test_data[,8:length(test_data)]

#NearZeroVar gives us a first approach of columns that we don't need
nsv<-nearZeroVar(train_data,saveMetrics=TRUE)

#We exclude the columns that NZV is TRUE (almost all of them are 0 or NA)
train_data<-train_data[,-which(names(train_data) %in% row.names(nsv[nsv$nzv==TRUE,]))]
test_data<-test_data[,-which(names(test_data) %in% row.names(nsv[nsv$nzv==TRUE,]))]

dim(train_data)
```

```
## [1] 19622   118
```

We see that we still have 118 columns, so we are going to do a deeper selection of variables. We are going to find the variables that have some correlation between them using what we learn in Lecture "Preprocessing with PCA", but using a low correlation (10%) to just find avoid the variables that have a lot of NA's and weren't find by the nearZeroVar function.

```
#We exclude the Classe column
M<-abs(cor(train_data[,-118])); diag(M)<-0
useful_var<-unique(row.names(which(M>0.1,arr.ind=T)))
useful_var<-c(useful_var,"classe")

#Now we subset the train and test data using this columns
vars<-names(train_data) %in% useful_var
train_data<-train_data[vars]
test_data<-test_data[vars]

#Just to be sure we check if both data sets have the same columns
all.equal(names(train_data),names(test_data))
```

```
## [1] "1 string mismatch"
```

We finally get 53 columns that are going to be part of our model, and the only different column is "classe" in train and "problem_id" in test.

Model Building

We will test 4 different models to compare their "out of sample error" and choose the better one, the models will be:

-Classifitacion Tree with "Cross Validation" and "PreProcessing" -Random Forest with "Cross Validation" -Random Forest with "PreProcessing" -Random Forest with "Cross Validation" and "Preprocessing"

First we need to split the train data in "training" and "testing" to check the accuracy of the models trained.

```
set.seed(110567)
inTrain<-createDataPartition(y=train_data$classe,p=0.70,list=FALSE)
training<-train_data[inTrain,]
testing<-train_data[-inTrain,]
```

Because of the time it takes to train each model, I decided to SAVE the resulting models in RDS files and load them to make the predictions, but the code used for each of them is ther but not running.

Model 1: Classification Tree with Cross Validation and PreProcessing

```
set.seed(110567)
##We train the First Model
modFit_tree <- train(classe ~ .,  preProcess=c("center", "scale"), trControl=trainControl(method = "cv"
```

```
#We load the model
modFit_tree<-readRDS("modFit_tree.RDS")
##We predict the values with the model
predic_tree<-predict(modFit_tree,newdata=testing)
#Now we compare the predictions with the real data
confusionMatrix(predic_tree,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1165  254   35  104   56
##          B  214  670   47  155  255
##          C  266  212  940  543  232
##          D    1    3    4  162   17
##          E   28    0    0    0  522
##
## Overall Statistics
##
##                Accuracy : 0.5878
##                  95% CI : (0.5751, 0.6004)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4785
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6959   0.5882   0.9162  0.16805  0.48244
## Specificity            0.8934   0.8586   0.7421  0.99492  0.99417
## Pos Pred Value         0.7218   0.4996   0.4286  0.86631  0.94909
## Neg Pred Value         0.8808   0.8968   0.9767  0.85925  0.89503
## Prevalence             0.2845   0.1935   0.1743  0.16381  0.18386
## Detection Rate         0.1980   0.1138   0.1597  0.02753  0.08870
## Detection Prevalence   0.2743   0.2279   0.3726  0.03178  0.09346
## Balanced Accuracy      0.7947   0.7234   0.8292  0.58148  0.73831
```

We can see that the model have a 58.78% of accuracy meaning a 0.4122 out of sample error, not a really good result. In the Appendix we can see the plot of the Tree (Figure 1)

Model 2: Random Forest with Cross Validation

```
set.seed(110567)
modFit_Fo_cv<-train(classe ~ ., method="rf", trControl=trainControl(method = "cv", number = 4,allowPara
```

```
modFit_Fo_cv<-readRDS("modFit_Fo_cv.RDS")
predic_Fo_cv<-predict(modFit_Fo_cv,newdata=testing)
confusionMatrix(predic_Fo_cv,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    6    0    0    0
##          B    1 1131    4    1    1
##          C    0    2 1015   13    0
```

```
##          D    0    0    7  950    5
##          E    0    0    0    0 1076
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9908, 0.9951)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9914
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9930   0.9893   0.9855   0.9945
## Specificity            0.9986   0.9985   0.9969   0.9976   1.0000
## Pos Pred Value         0.9964   0.9938   0.9854   0.9875   1.0000
## Neg Pred Value         0.9998   0.9983   0.9977   0.9972   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1922   0.1725   0.1614   0.1828
## Detection Prevalence   0.2853   0.1934   0.1750   0.1635   0.1828
## Balanced Accuracy      0.9990   0.9958   0.9931   0.9915   0.9972
```

Using Random Forest with Cross Validation we obtained a much better result with 99.32% accuracy (an interval of [99.08% - 99.51]), meaning a 0.0068 out of sample error.

Model 3: Random Forest with PreProcessing

```
set.seed(110567)
modFit_Fo_pp<-train(classe ~ ., method="rf", preProcess=c("center", "scale"), data=training)
```

```
modFit_Fo_pp<-readRDS("modFit_Fo_pp.RDS")
predic_Fo_pp<-predict(modFit_Fo_pp,newdata=testing)
confusionMatrix(predic_Fo_pp,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    6    0    0    0
##          B    2 1131    3    2    1
##          C    0    2 1017   12    0
##          D    0    0    6  950    5
##          E    0    0    0    0 1076
##
## Overall Statistics
##
##                Accuracy : 0.9934
##                  95% CI : (0.991, 0.9953)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9916
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9930   0.9912   0.9855   0.9945
## Specificity            0.9986   0.9983   0.9971   0.9978   1.0000
## Pos Pred Value         0.9964   0.9930   0.9864   0.9886   1.0000
## Neg Pred Value         0.9995   0.9983   0.9981   0.9972   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1922   0.1728   0.1614   0.1828
## Detection Prevalence   0.2851   0.1935   0.1752   0.1633   0.1828
## Balanced Accuracy      0.9987   0.9956   0.9942   0.9916   0.9972
```

In the case of using Rando Forest with PreProcessing but not Cross Validation we obtained 99.34% of accuracy, almost the same of the last one, and a 0.0066 out of sample error.

Model 4: Random Forest with Cross Validation and PreProcessing

```
set.seed(110567)
modFit_Fo_cvpp<-train(classe~.,data=training,preProcess=c("center","scale"),trControl=trainControl(metho
```

```
modFit_Fo_cvpp<-readRDS("modFit_Fo_cvpp.RDS")
predic_Fo_cvpp<-predict(modFit_Fo_cvpp,newdata=testing)
confusionMatrix(predic_Fo_cvpp,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    7    0    0    0
##          B    1 1129    3    1    1
##          C    0    3 1016   13    0
##          D    0    0    7  950    4
##          E    0    0    0    0 1077
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9908, 0.9951)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9914
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9912   0.9903   0.9855   0.9954
## Specificity            0.9983   0.9987   0.9967   0.9978   1.0000
## Pos Pred Value         0.9958   0.9947   0.9845   0.9886   1.0000
## Neg Pred Value         0.9998   0.9979   0.9979   0.9972   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1918   0.1726   0.1614   0.1830
## Detection Prevalence   0.2855   0.1929   0.1754   0.1633   0.1830
## Balanced Accuracy      0.9989   0.9950   0.9935   0.9916   0.9977
```

Using both, Cross Validation and PreProcessing gives us a accuracy of 99.32%, same as the one with only Cross Validation.

Model Comparison

Now we compare the results

-Clasification Tree with CV and PP: 58.78% Accuracy - 0.4122 Sample Error
-Random Forest with CV: 99.32% Accuracy - 0.0068 Sample Error
-Random Forest with PP: 99.34% Accuracy - 0.0066 Sample Error
-Random Forest with CV and PP: 99.32% Accuracy - 0.0068 Sample Error

We can see that clearly the Random Forest gives better results, as expected, and both Preprocessing and Cross Validation gives almost the same results.

In the Figure 2 of the Appendix we can see the plots of the error of the 3 models with Random Forest, showing similar results, and on each of them the comparison between the different classes.

Applied best Model to Test Cases

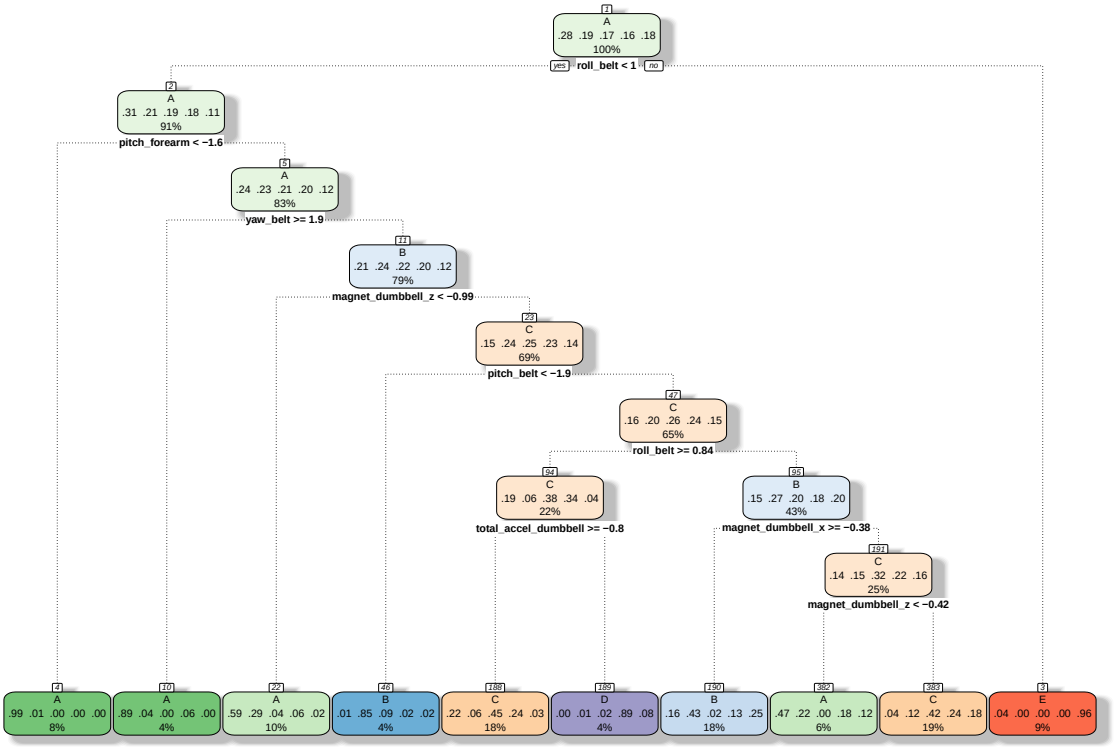
Considering the comparison we are going to use the Random Forest with Preprocessing to get the best prediction of the Test Cases.

```
predict(modFit_Fo_cvpp,newdata=test_data)
```

```
##  [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
```

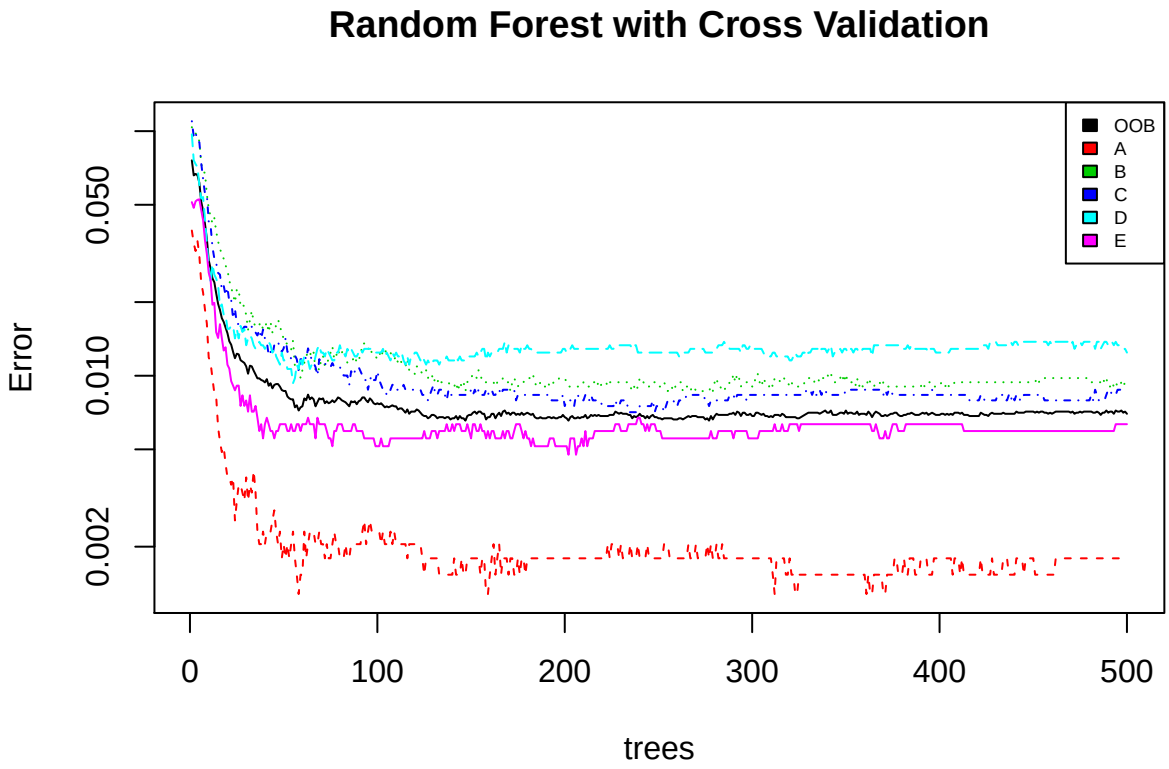
Appendix

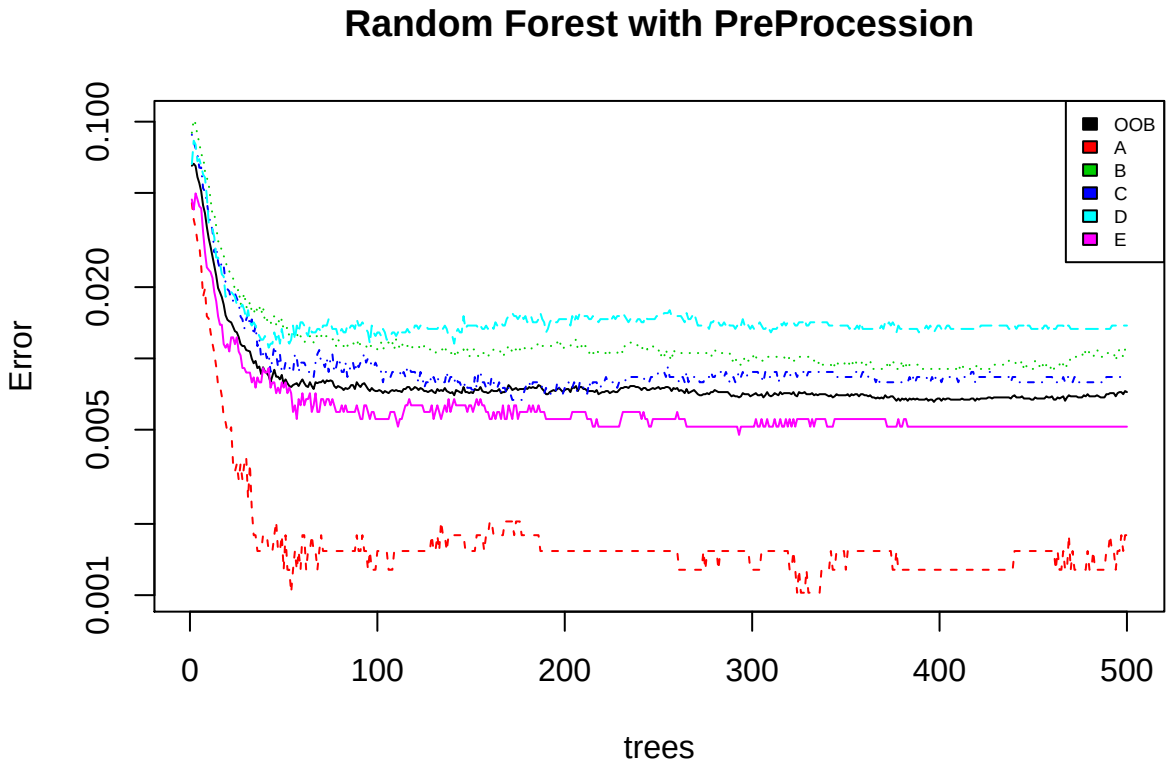
Figure 1

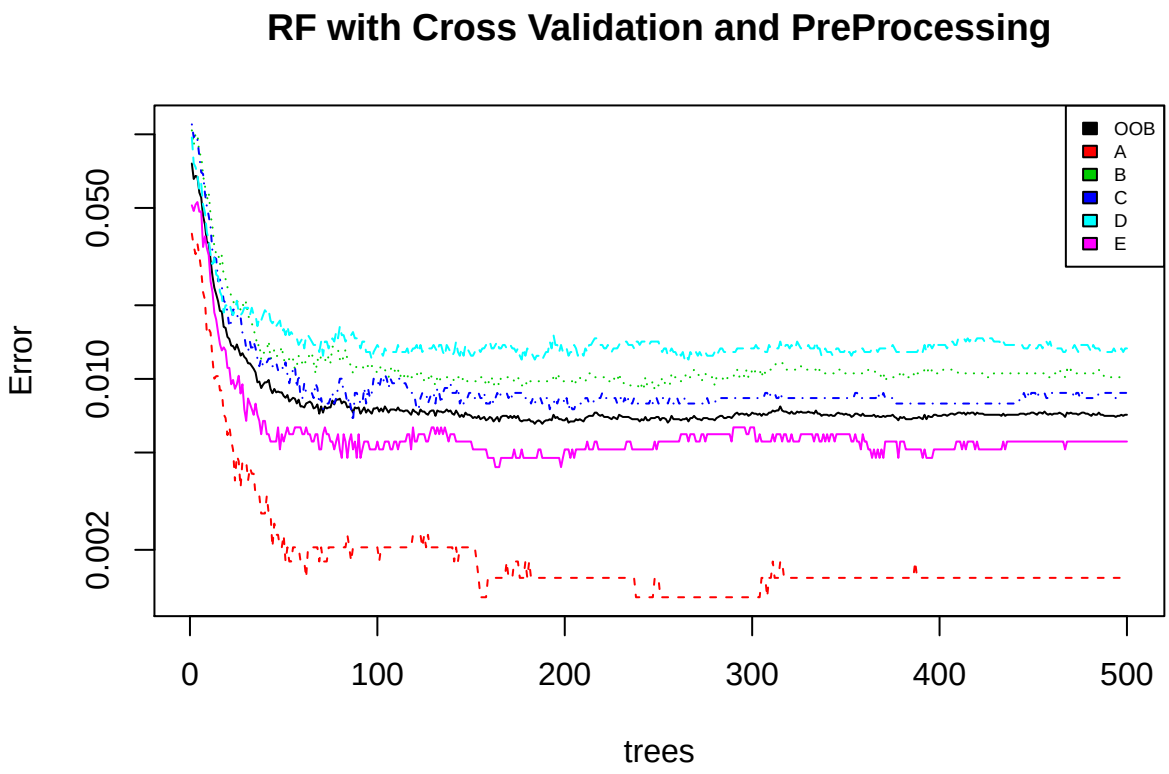


Rattle 2015−Ago.−23 18:26:39 Sebastián

Figure 2


Random Forest with Cross Validation
Error
0.002
0.010
0.050
0
100
200
300
400
500
trees
OOB
A
B
C
D
E


Random Forest with PreProcession
Error
0.001
0.005
0.020
0.100
0
100
200
300
400
500
trees
OOB
A
B
C
D
E


RF with Cross Validation and PreProcessing
OOB
A
B
C
D
E
Error
0.002
0.010
0.050
0
100
200
300
400
500
trees