# Machine Learning Course Project

*TZiegler*

*27 März 2016*

# Introduction

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.

This was done in an eximination by Velloso et al. They used data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

The goal of this project is to predict the manner in which the participants did the exercise. This is the “classe” variable in the training set.

The raw data is provided by Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H.. More information is available from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

# Data Preprocessing

## Data Loading

Load the training and test data.

```
if (!file.exists("data/pml-training.csv")) {
    download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.
csv",
        destfile = "pml-training.csv")
}
if (!file.exists("data/pml-testing.csv")) {
    download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.c
sv",
        destfile = "pml-testing.csv")
}
```

Removing empty fields, miscellaneous NA and #DIV/0! as “NA”.

```
rawDataTrain <- read.csv("data/pml-training.csv", sep = ",", na.strings = c("",
    "NA", "#DIV/0!"))
str(rawDataTrain)
dim(rawDataTrain)

rawDataTest <- read.csv("data/pml-testing.csv", sep = ",", na.strings = c("",
    "NA", "#DIV/0!"))
str(rawDataTest)
dim(rawDataTest)
```

Check the rows of training and test data which has complete cases.

```
sum(complete.cases(rawDataTrain))
```

```
## [1] 0
```

```
sum(complete.cases(rawDataTest))
```

```
## [1] 0
```

There are no complete cases neither in the training nor the testing data set. Therefore, we have to clean the data.

# Data Cleaning and Preperation

First, we remove variables with no variability at all. These variables are not useful when we want to construct a prediction model.

```
trainVar <- rawDataTrain[, -nearZeroVar(rawDataTrain)]
testVar <- rawDataTest[, -nearZeroVar(rawDataTest)]
```

Next, we remove the columns that do not contribute to the results like variables with user information, time and undefined:

```
varToRm <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
    "cvtd_timestamp", "num_window")

trainNew <- subset(trainVar, select = setdiff(names(trainVar), varToRm))
dataTest <- subset(testVar, select = setdiff(names(testVar), varToRm))

dim(trainNew)
```

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

```
dim(dataTest)
```

```
## [1] 20 53
```

Remove columns that contain majority of NAs.

```
index = vector()
for (i in 1:ncol(trainNew)) {
    if (sum(is.na(trainNew[, i]))/nrow(trainNew) > 0.6) {
        index = c(index, i)
    }
}
dataTrain <- trainNew[, -index]
```

## Harmonise variables in training amd test sets

```
clsTrain <- dataTrain[, "classe"]
dataTrain <- dataTrain[, colnames(dataTrain) %in% colnames(dataTest)]  # classe wil
l be removed because not in test set
dataTrain <- cbind(clsTrain, dataTrain)  # add classe clumn again
names(dataTrain)[1] <- "classe"  # and rename back again
# Check if classe variable is a factor class(dataTrain$classe) => is true
```

With the cleaning process the number of variables for the analysis has been reduced to 53 only.

According to the correlation analysis (see Appendix, Figure 1 ) we could make an evem more compact analysis by performing a PCA (Principal Components Analysis) as a pre-processing step to the datasets. But as the there are a great numvber of correlations, this step will not be applied for this assignment.

# Model Building

## Partitioning the original training data for Cross Validation

The cleaned data set has about 20K rows, this is a moderate data set and I am splitting it 70:30.

```
set.seed(1234)
inTrain <- createDataPartition(y = dataTrain$classe, p = 0.7, list = FALSE)
training <- dataTrain[inTrain, ]
testing <- dataTrain[-inTrain, ]
```

```
dim(training)
```

```
## [1] 13737    53
```

```
dim(testing)
```

```
## [1] 5885   53
```

# Data Modelling

We fit different classification model types using 5-fold cross validation. To speed up the process we use parallel processing.

```
## Initialize parallel processing
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

tcontrol <- trainControl(method = "cv", number = 10, verboseIter = FALSE)
rp.fit <- train(classe ~ ., data = training, method = "rpart", trControl = tcontrol
,
    tuneLength = 10)
lda.fit <- train(classe ~ ., data = training, method = "lda", trControl = tcontrol)
gbm.fit <- train(classe ~ ., data = training, method = "gbm", trControl = tcontrol,
    verbose = FALSE)
knn.fit <- train(classe ~ ., data = training, method = "knn", trControl = tcontrol)
rf.fit <- train(classe ~ ., data = training, method = "rf", trControl = tcontrol,
    ntree = 200)

## Stop parallel processing
stopCluster(cluster)
```

Use the models to predict the results on the testing set.

```
rp.pred.test <- predict(rp.fit, testing)
lda.pred.test <- predict(lda.fit, testing)
gbm.pred.test <- predict(gbm.fit, testing)
knn.pred.test <- predict(knn.fit, testing)
rf.pred.test <- predict(rf.fit, testing)
```

Create a confusion matrix for each model:

```
cmRP <- confusionMatrix(testing$classe, rp.pred.test)
cmLDA <- confusionMatrix(testing$classe, lda.pred.test)
cmGBM <- confusionMatrix(testing$classe, gbm.pred.test)
cmKNN <- confusionMatrix(testing$classe, knn.pred.test)
cmRF <- confusionMatrix(testing$classe, rf.pred.test)
```

The results of the models are shown in the following table: TrainAaccuracy (performance in building the model), ValidationAccuracy (model performance against a separate data set than we used to train the model), ValidationKappa (measuring validation agreement between actual and predicted values) and the OutOfSampleErr (one minus validation accuracy).

```
require(knitr)
ModelType <- c("Rpart tree", "Linear discriminant", "Gradient boosting machine",
    "K nearest neighbor", "Random forest")

max(rf.fit$results$Accuracy)
```

```
## [1] 0.9928656
```

```
TrainAccuracy <- c(max(rp.fit$results$Accuracy), max(lda.fit$results$Accuracy),
    max(gbm.fit$results$Accuracy), max(knn.fit$results$Accuracy), max(rf.fit$result
s$Accuracy))

ValidationAccuracy <- c(cmRP$overall[1], cmLDA$overall[1], cmGBM$overall[1],
    cmKNN$overall[1], cmRF$overall[1])

ValidationKappa <- c(cmRP$overall[2], cmLDA$overall[2], cmGBM$overall[2], cmKNN$ove
rall[2],
    cmRP$overall[2])

OutOfSampleErr <- 1 - ValidationAccuracy

metrics <- data.frame(ModelType, TrainAccuracy, ValidationAccuracy, ValidationKappa
,
    OutOfSampleErr)
kable(metrics, digits = 5)
```

| ModelType | TrainAccuracy | ValidationAccuracy | ValidationKappa | OutOfSampleErr |
|---|---|---|---|---|
| Rpart tree | 0.68786 | 0.67239 | 0.58603 | 0.32761 |
| Linear discriminant | 0.70226 | 0.70484 | 0.62645 | 0.29516 |
| Gradient boosting machine | 0.96149 | 0.96483 | 0.95548 | 0.03517 |
| K nearest neighbor | 0.89648 | 0.91487 | 0.89222 | 0.08513 |
| Random forest | 0.99287 | 0.99473 | 0.58603 | 0.00527 |

The **decision tree model (Rpart)** is 67.24% accurate on the testing data partitioned from the training data (Figure 3). The expected out of sample error is roughly 0.33%. The tree is shown in Figure 2.

The best model is **Random forest**. It has the lowest out of sample error 0.0052676 and an validation accurency of 99.47 (Figure 4). Some models have a better validation accuracy than training accuracy. This behavior often points to some degree of overfitting because normally validation accuracy tends to be lower than training accuracy.

The print of the confusion matrix shows that the sensitivity and specificity of the random forest model both seem to be quite good. The detection rate (the rate of truepositives) closely matches the prevalence (the estimated population prevalence) of the classes.

```
print(rf.fit)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12363, 12362, 12365, 12363, 12363, 12364, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9928656  0.9909744  0.002025067  0.002562715
##   27    0.9926464  0.9906975  0.003063022  0.003875775
##   52    0.9903896  0.9878421  0.002403073  0.003041491
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
print(cmRF)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    7 1130    2    0    0
##          C    0    8 1016    2    0
##          D    0    0   10  954    0
##          E    0    0    1    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9925, 0.9964)
##     No Information Rate : 0.2856
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9958   0.9930   0.9874   0.9969   1.0000
## Specificity            1.0000   0.9981   0.9979   0.9980   0.9996
## Pos Pred Value         1.0000   0.9921   0.9903   0.9896   0.9982
## Neg Pred Value         0.9983   0.9983   0.9973   0.9994   1.0000
## Prevalence             0.2856   0.1934   0.1749   0.1626   0.1835
## Detection Rate         0.2845   0.1920   0.1726   0.1621   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9979   0.9955   0.9927   0.9974   0.9998
```

The random forest model shows a rapid decline in predictor importance. This supports the result of the correlation analysis.

# Applying Selected Model to Test Set

We will use the Random Forest model to make the predictions on the test data to predict the way 20 participates performed the exercise.

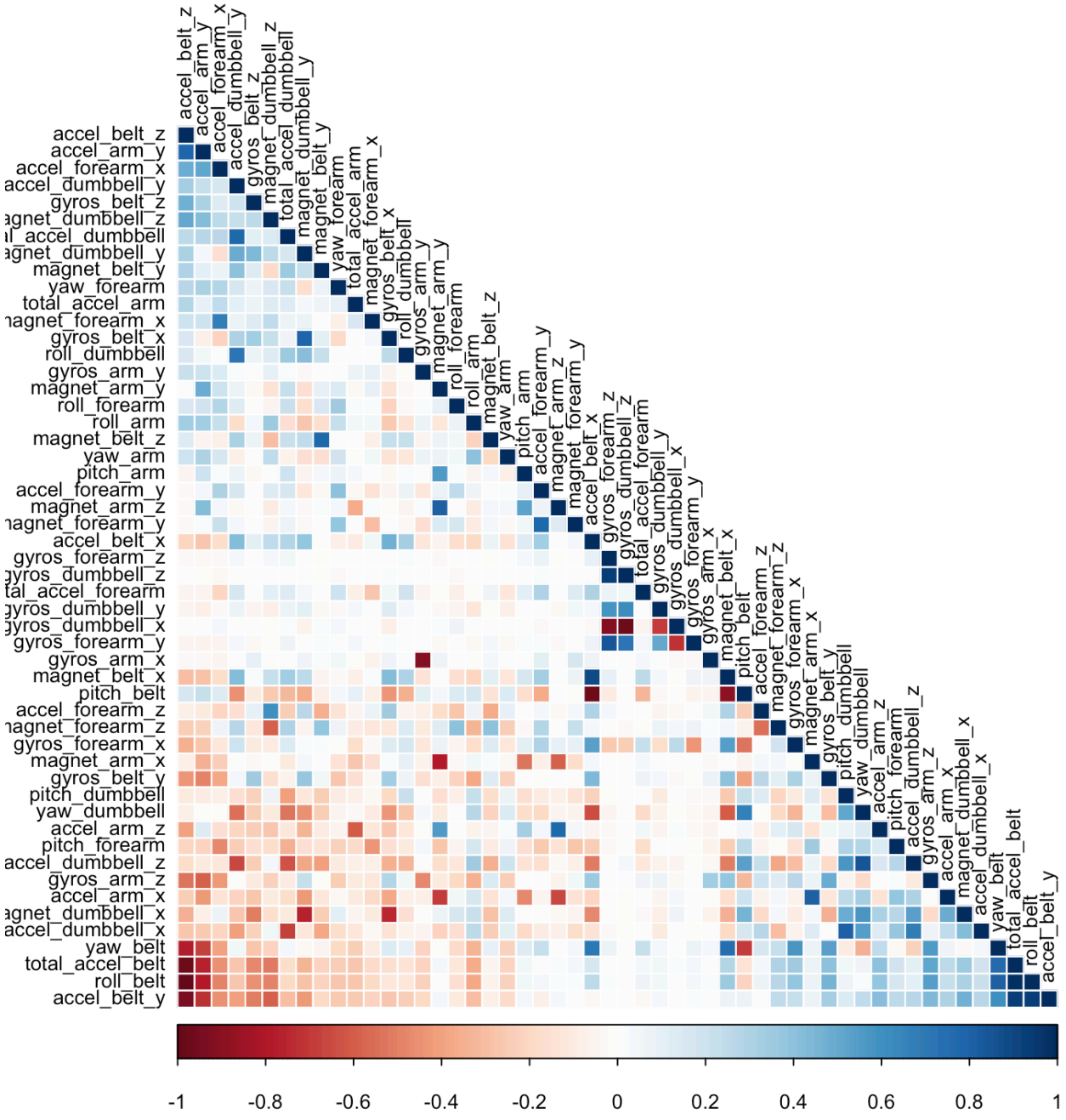
```
parTesting <- predict(rf.fit, dataTest)
parTesting
```

```
##  [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: Correlation Plot

```
suppressMessages(library(corrplot))  # nice correlation plots
corrMatrix <- cor(dataTrain[, -1])
corrplot(corrMatrix, order = "FPC", method = "color", type = "lower", tl.cex = 0.8,
    tl.col = rgb(0, 0, 0))
```




Highly correlated variables are shown in dark colors.

Figure 2: Plot of the decision tree model (Rpart)

```
fancyRpartPlot(rp.fit$finalModel)
```

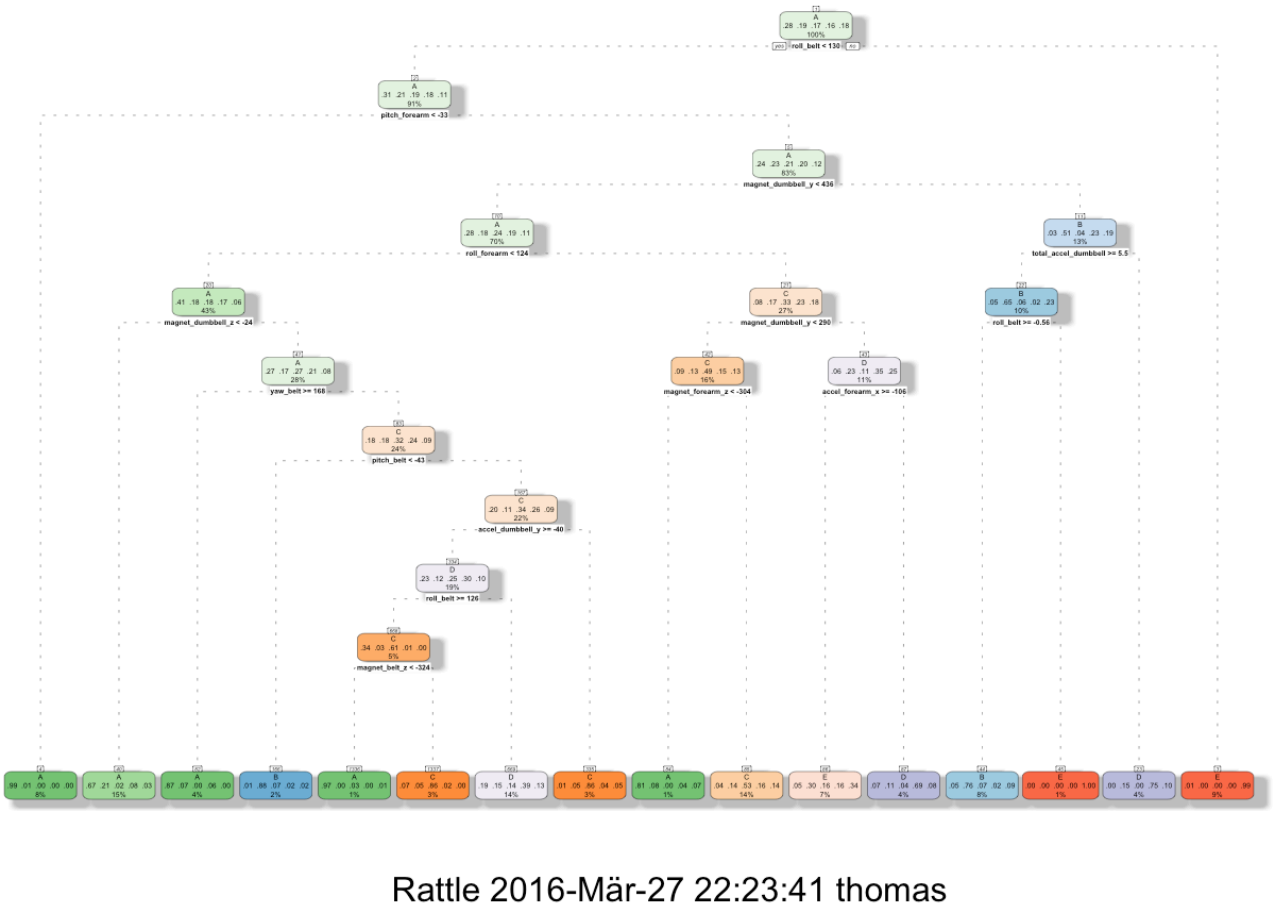



Figure 3: Decision tree accuracy

```
plot(cmRP$table, col = cmRP$byClass, main = paste("Decision Tree - Accuracy =",
    round(cmRP$overall["Accuracy"], 4)))
```

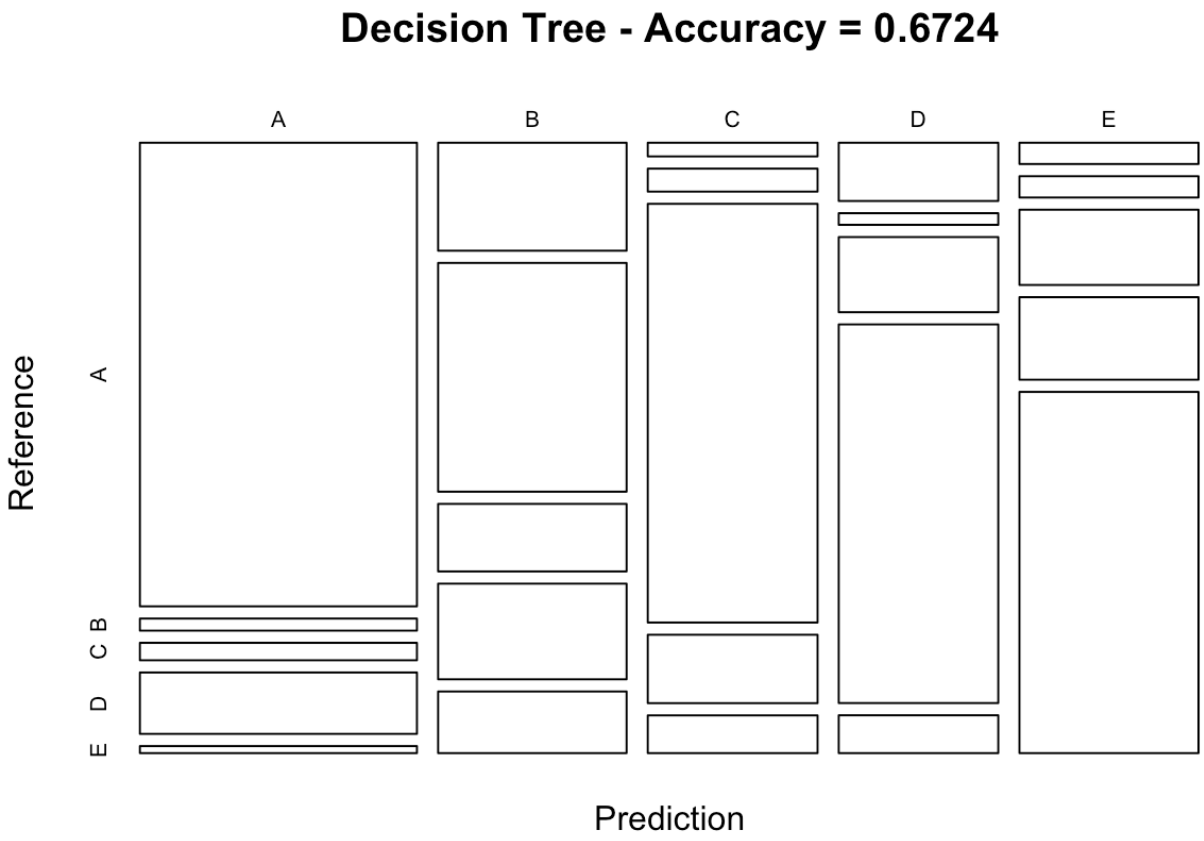



Figure 4: Random forest accuracy

```
plot(cmRF$table, col = cmRF$byClass, main = paste("Random Forest - Accuracy =",
    round(cmRF$overall["Accuracy"], 4)))
```

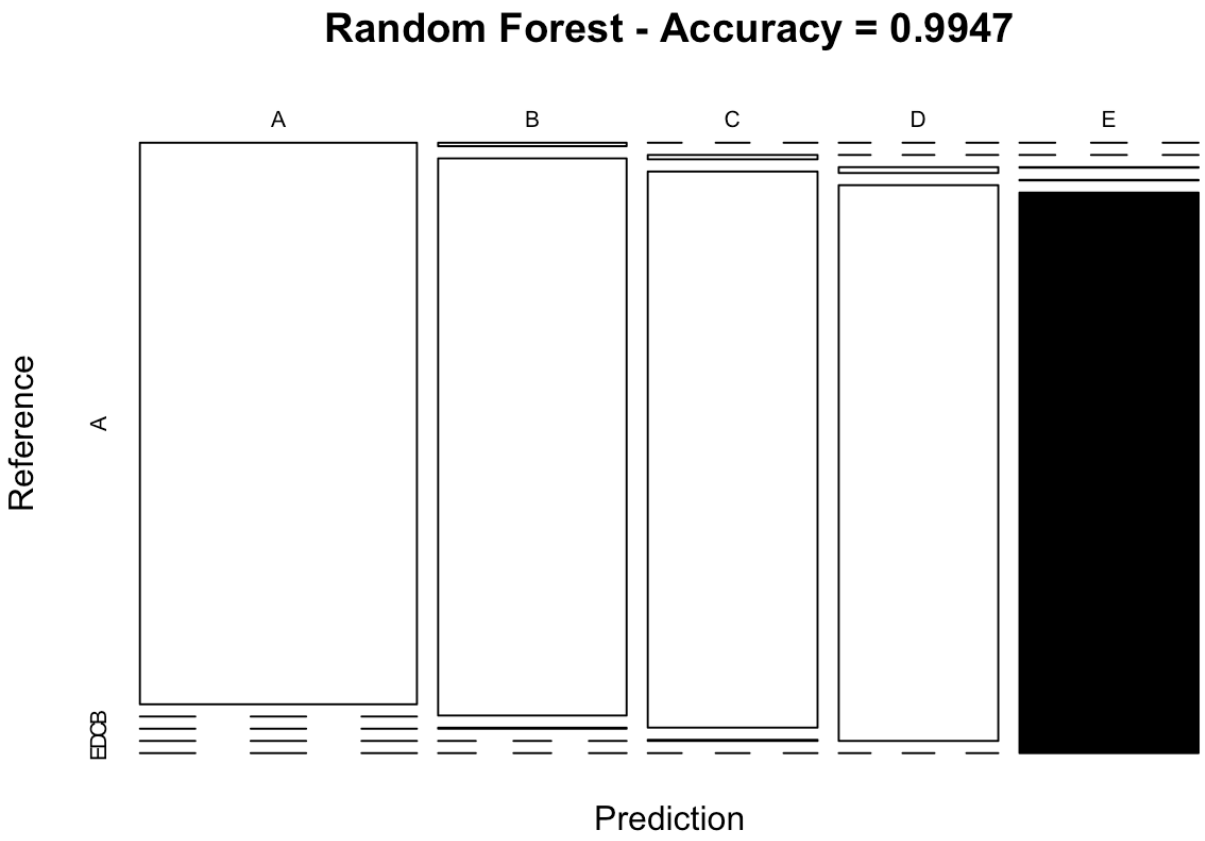
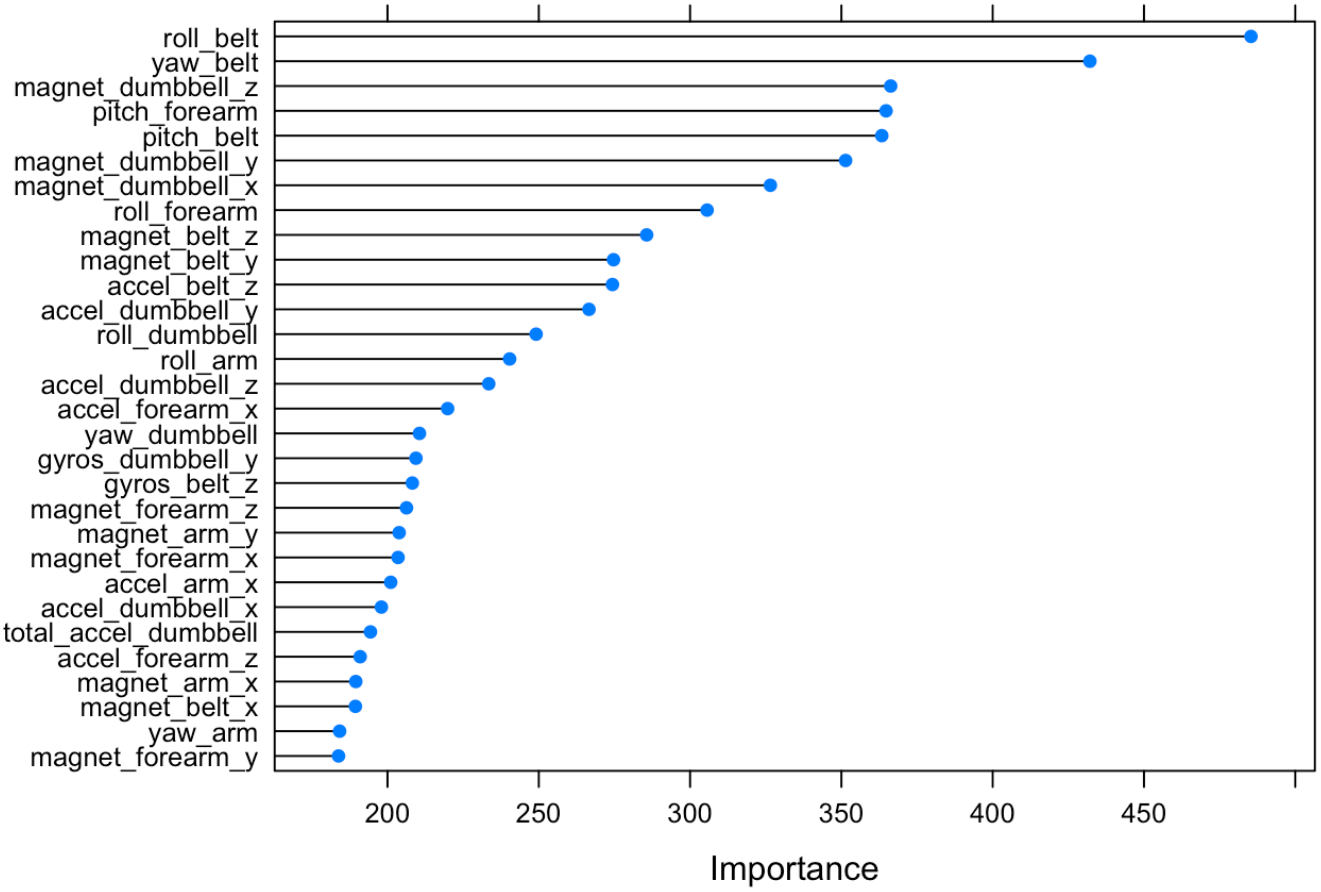



Figure 5: Plot of the variable importance

```
varImpRF <- varImp(rf.fit, scale = FALSE)
plot(varImpRF, top = 30, main = "Variable importance - Random Forest model - top 30
predictors")
```


Variable importance - Random Forest model - top 30 predictors
roll_belt
yaw_belt
magnet_dumbbell_z
pitch_forearm
pitch_belt
magnet_dumbbell_y
magnet_dumbbell_x
roll_forearm
magnet_belt_z
magnet_belt_y
accel_belt_z
accel_dumbbell_y
roll_dumbbell
roll_arm
accel_dumbbell_z
accel_forearm_x
yaw_dumbbell
gyros_dumbbell_y
gyros_belt_z
magnet_forearm_z
magnet_arm_y
magnet_forearm_x
accel_arm_x
accel_dumbbell_x
total_accel_dumbbell
accel_forearm_z
magnet_arm_x
magnet_belt_x
yaw_arm
magnet_forearm_y
200
250
300
350
400
450
Importance