1 Getting data

First of all, we load all the packages we are going to use and we look at the data.
There are three types of NAs in the data sets; "NA", "#DIV/0!" and an empty space "". We classify all of those as "NA".
Moreover, the first seven columns of the data sets are just information about users and time/dates, which are not important for the prediction of classe, therefore we do not need those.

```
> library(MASS)
> library(caret)
> library(rattle)
> library(foreach)
> library(rpart)
> library(randomForest)
> library(ppcor)
> trainData <- read.csv("pml-training.csv", header = T,
+                       na.strings = c("NA", "#DIV/0!", ""))[,-(1:7)]
> testData <- read.csv("pml-testing.csv", header = T,
+                      na.strings = c("NA", "#DIV/0!", ""))[,-(1:7)]
```

2 Cleaning data

Firstly, we get rid of the columns that with more than 40% NA.

```
> data_NA <- apply(is.na(trainData), 2, sum) > 0.4*nrow(trainData)
> train <- (trainData[,!data_NA])
> test <- (testData[,!data_NA])
```

Secondly, we check the correlation between the predictors. We are interested in pairs of predictors with correlation over 0.80.

```
> train_matrix <- as.matrix(train[,-53])
> correlation <- pcor(train_matrix, method="pearson")$estim
> diag(correlation) <- 0
> which((abs(correlation))>0.80, arr.ind=TRUE)
```

```
                 row col
gyros_arm_y       19  18
gyros_arm_x       18  19
magnet_arm_z      26  23
accel_arm_z       23  26
gyros_dumbbell_z  33  31
gyros_dumbbell_x  31  33
magnet_forearm_x  50  47
accel_forearm_x   47  50
```

Since we work out ourselves and have read the paper Wearable Qualitative Activity Recognition of Weight Lifting Exercises as well as about methods and devices used for the measurements, we decide to eliminate following predictors: "$gyros_arm_x$","$magnet_arm_z$","$gyros_dumbbell_x$"and"$magnet_forearm_x$".

```
> ## we eliminate gyros_arm_x, magnet_arm_z, gyros_dumbbell_x and magnet_forearm_x
> train_noncorr <- train[,-c(18,26,31,47)]
```

We use this dataset with non-correlated columns to build our model.

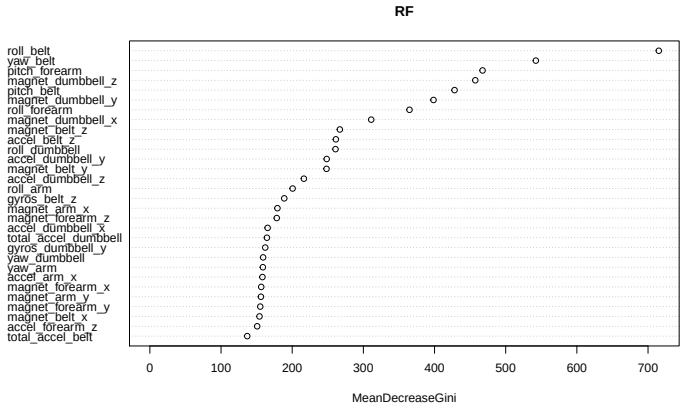
3 Building the model

For the cross validation we use random subsampling approach, i.e. splitting our "train" data into two subsets: 60% used for the model building and 40% for the model testing. (the "test" data set will be used later for the prediction). Also, we change "classe" to factor variable.

```
> set.seed(123)
> inTrain <- createDataPartition(train_noncorr$classe, p=0.6, list=F)
> training  <- train_noncorr[inTrain,]
> testing  <- train_noncorr[-inTrain,]
> dim(training); dim(testing)

[1] 11776    49

[1] 7846   49

> training$classe <- as.factor(training$classe)
```

We decided to use random forest approach since it produces the most accurate results.
After eliminating NAs and correlated predictors we end up with 49 predictors, which is still quite a lot. Therefore we order them according to their importance and decide whether we can simplify the model.

```
> set.seed(123)
> RF <- randomForest(classe~., data=training, ntree=500)
> varImpPlot(RF)
```



There is a significant break after the 13th predictor, thus we use the first 13 of them. We create the final dataset with the predictors and "classe".

```
> df <- as.data.frame(RF$importance)
> TRAINING <- training[,c(order(df$MeanDecreaseGini, decreasing = T)[1:13], 49)]
```

Finally, we build the model. In order to improve the performance of random forest approach, we have decided to use bagging.

```
> ## bootstrapped sample size 1/p of the original number of observations,
> ## "iter" is number of iterations
> bagg_RF <- function(training_data,testing_data,p,iter)
+ {
+   predictions <- foreach(m=1:iter,.combine=cbind) %do% {
+     sampled_positions <- sample(nrow(training_data), size=floor((nrow(training_data)/p)))
+     training_positions <- 1:nrow(training_data) %in% sampled_positions
+     set.seed(123)
+     ## we have chosen randomForest instead of train(..., method="rf",...)
+     ## because it is quicker and we get slightly better results
+     RF_fit <- randomForest(classe~., data=training_data[training_positions,], ntree=500)
+     predict(RF_fit,newdata=testing_data)
+   }
+   Predictions_num <- round(rowMeans(predictions))
+   Predictions <- as.vector(length(Predictions_num))
+   for(i in 1:length(Predictions_num)){
+     if(Predictions_num[i]==1) Predictions[i]="A"
+     if(Predictions_num[i]==2) Predictions[i]="B"
+     if(Predictions_num[i]==3) Predictions[i]="C"
+     if(Predictions_num[i]==4) Predictions[i]="D"
+     if(Predictions_num[i]==5) Predictions[i]="E"
+   }
+   Predictions
+ }
```

We use our model to predict on the "testing" dataset for accuracy.

```
> testing_pred <- bagg_RF(TRAINING,testing,p=4,iter=100)
> confusionMatrix(testing$classe, testing_pred)

Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 2184   44    4    0    0
         B   41 1386   70   19    2
         C    0   52 1290   26    0
         D    4    3   48 1231    0
         E    2    2   13   33 1392

Overall Statistics

               Accuracy : 0.9537
                 95% CI : (0.9489, 0.9583)
    No Information Rate : 0.2843
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9415
 Mcnemar's Test P-Value : 1.993e-12

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9789   0.9321   0.9053   0.9404   0.9986
Specificity            0.9915   0.9792   0.9879   0.9916   0.9923
Pos Pred Value         0.9785   0.9130   0.9430   0.9572   0.9653
Neg Pred Value         0.9916   0.9840   0.9792   0.9881   0.9997
Prevalence             0.2843   0.1895   0.1816   0.1668   0.1777
Detection Rate         0.2784   0.1767   0.1644   0.1569   0.1774
Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
Balanced Accuracy      0.9852   0.9557   0.9466   0.9660   0.9954
```

We get the accuracy 95.37% on the "testing" set, which is satisfactory.
The very last thing to do on this project is to predict "classe" for the "testData".

```
> bagg_RF(TRAINING,testData,p=4,iter=100)

 [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A" "B" "B"
[20] "B"
```