Packages, Libraries, Seed

Installing packages, loading libraries, and setting the seed for reproduceability:

```
#install.packages("caret")
#install.packages("randomForest")
#install.packages("rpart")
library(caret)
## Loading required package: lattice
## Loading required package: ggplot2
library(randomForest) #Random forest for classification and regression
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
library(rpart) # Regressive Partitioning and Regression trees
library(rpart.plot) # Decision Tree plot

# setting the overall seed for reproduceability
set.seed(1234)
```

Loading data sets and preliminary cleaning

First we want to load the data sets into R and make sure that missing values are coded correctly.
Irrelevant variables will be deleted.
Results will be hidden from the report for clarity and space considerations.

```
# After saving both data sets into my working directory
# Some missing values are coded as string "#DIV/0!" or "" or "NA" - these
will be changed to NA.
# We notice that both data sets contain columns with all missing values -
these will be deleted.

# Loading the training data set into my R session replacing all missing with
"NA"
trainingset <- read.csv("C:/Users/Sandrine/ML_Project/trainingdata.csv",
na.strings=c("NA","#DIV/0!", ""))

# Loading the testing data set
testingset <- read.csv('C:/Users/Sandrine/ML_Project/testingdata.csv',
na.strings=c("NA","#DIV/0!", ""))

# Check dimensions for number of variables and number of observations
dim(trainingset)
dim(testingset)

# Delete columns with all missing values
trainingset<-trainingset[,colSums(is.na(trainingset)) == 0]
testingset <-testingset[,colSums(is.na(testingset)) == 0]

# Some variables are irrelevant to our current project: user_name,
raw_timestamp_part_1, raw_timestamp_part_,2 cvtd_timestamp, new_window, and
num_window (columns 1 to 7). We can delete these variables.
trainingset   <-trainingset[,-c(1:7)]
testingset <-testingset[,-c(1:7)]
```

```
# and have a look at our new datasets:
dim(trainingset)
dim(testingset)
head(trainingset)
head(testingset)
```

Partitioning the training data set to allow cross-validation

The training data set contains 53 variables and 19622 obs.
The testing data set contains 53 variables and 20 obs.
In order to perform cross-validation, the training data set is partionned into 2 sets: subTraining (75%) and subTest (25%).
This will be performed using random subsampling without replacement.

```
subsamples <- createDataPartition(y=trainingset$classe, p=0.75, list=FALSE)
subTraining <- trainingset[subsamples, ]
subTesting <- trainingset[-subsamples, ]
dim(subTraining)
dim(subTesting)
head(subTraining)
head(subTesting)
```

A look at the Data

The variable "classe" contains 5 levels: A, B, C, D and E. A plot of the outcome variable will allow us to see the frequency of each levels in the subTraining data set and compare one another.

```
plot(subTraining$classe, col="blue", main="Bar Plot of levels of the variable
classe within the subTraining data set", xlab="classe levels",
ylab="Frequency")
```

From the graph above, we can see that each level frequency is within the same order of magnitude of each other. Level A is the most frequent with more than 4000 occurrences while level D is the least frequent with about 2500 occurrences.

First prediction model: Using Decision Tree

```
model1 <- rpart(classe ~ ., data=subTraining, method="class")

# Predicting:
prediction1 <- predict(model1, subTesting, type = "class")

# Plot of the Decision Tree
rpart.plot(model1, main="Classification Tree", extra=102, under=TRUE,
faclen=0)
```

```
# Test results on our subTesting data set:
```

```
confusionMatrix(prediction1, subTesting$classe)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1235  157   16   50   20
##          B   55  568   73   80  102
##          C   44  125  690  118  116
##          D   41   64   50  508   38
##          E   20   35   26   48  625
##
## Overall Statistics
##
##                Accuracy : 0.739
##                  95% CI : (0.727, 0.752)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.67
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.885    0.599    0.807    0.632    0.694
## Specificity             0.931    0.922    0.900    0.953    0.968
## Pos Pred Value          0.836    0.647    0.631    0.725    0.829
## Neg Pred Value          0.953    0.905    0.957    0.930    0.933
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.252    0.116    0.141    0.104    0.127
## Detection Prevalence    0.301    0.179    0.223    0.143    0.154
## Balanced Accuracy       0.908    0.760    0.854    0.792    0.831
```

Second prediction model: Using Random Forest

```
model2 <- randomForest(classe ~. , data=subTraining, method="class")

# Predicting:
prediction2 <- predict(model2, subTesting, type = "class")

# Test results on subTesting data set:
confusionMatrix(prediction2, subTesting$classe)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    3    0    0    0
##          B    1  944   10    0    0
##          C    0    2  843    6    0
##          D    0    0    2  798    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.995
##                  95% CI : (0.993, 0.997)
```

```
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.994
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.999    0.995    0.986    0.993    1.000
## Specificity             0.999    0.997    0.998    1.000    1.000
## Pos Pred Value          0.998    0.988    0.991    0.997    1.000
## Neg Pred Value          1.000    0.999    0.997    0.999    1.000
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.172    0.163    0.184
## Detection Prevalence    0.285    0.195    0.174    0.163    0.184
## Balanced Accuracy       0.999    0.996    0.992    0.996    1.000
```

Decision

As expected, Random Forest algorithm performed better than Decision Trees.
Accuracy for Random Forest model was 0.995 (95% CI: (0.993, 0.997)) compared to 0.739 (95% CI: (0.727, 0.752)) for Decision Tree model. **The random Forest model is choosen**. The accuracy of the model is 0.995. The expected out-of-sample error is estimated at 0.005, or **0.5%**. The expected out-of-sample error is calculated as 1 - accuracy for predictions made against the cross-validation set. Our Test data set comprises 20 cases. With an accuracy above 99% on our cross-validation data, we can expect that very few, or none, of the test samples will be missclassified.