Predictive Machine Learning Project

Frederico Munoz fsmunoz@gmail.com

25-OCT-2015

Project Background

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. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset). This document uses the DLA dataset that is available there (Ugulino et al. 2012)

Objective

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. This document attempts to address this goal.

Setting up the environment

First, some initial housecleaning involving loading the needed libraries and importing the needed datasets.

```r
## Load used libraries
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
library(rpart)
library(ggplot2)
library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

The random seed is set for reproducibly purposes

```
set.seed(6677)
```

The datasets are available online and so we import them directly.

```
## Dataset URLs
pml_training_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
pml_testing_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

## Uncomment in the end to avoid fetching at every run
##pml_training<- read.csv(url(pml_training_url),header = T)
##pml_testing<- read.csv(url(pml_testing_url),header = T)

## Assign datasets
pml_testing<- read.csv("pml-testing.csv",header = T)
pml_training <- read.csv("pml-training.csv", header = T)
```

A quick look

Let us take a quick initial look at the training data we now have available

```
str(pml_training)
```

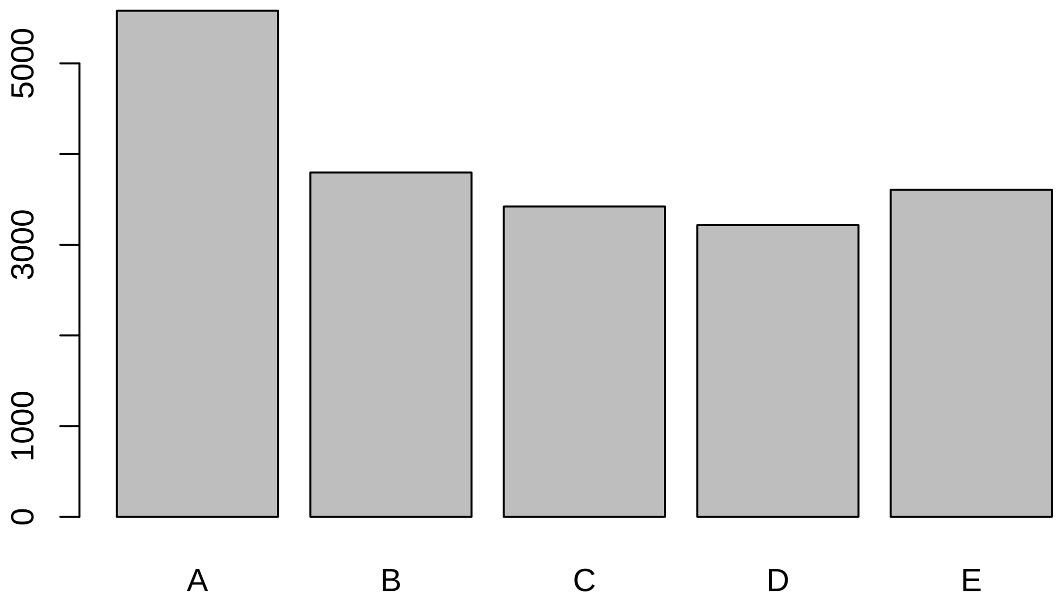
```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 4844
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","0.000673",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","0.006078",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","0.000000",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","0.000000",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","0.00","0.0000",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","0.01388","0.01574",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","0.00000","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ..
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","0.00000","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ..
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 398 levels "","0.0016","-0.0035",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","0.0045","0.0130",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","0.0011","0.0014",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","0.0063",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : Factor w/ 73 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : Factor w/ 73 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Lots of information there - 19622 observations of 160 variables to be precise. In particular we will focus on the “classe”" (which is Portuguese for “class”) variable which will be the one which we will be predicting.

```
summary(pml_training$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
plot(pml_training$classe)
```



Partitioning the data

We now split the training data into two different sets, training (60%) and testing (40%), using "classe"" as the outcome.

```
inTrain <- createDataPartition(y=pml_training$classe, p=0.6, list=FALSE)
training_set <- pml_training[inTrain, ]
testing_set <-  pml_training[-inTrain, ]
```

Cleaning the data

When we took a quick look at the data it was apparent that there were many variables with NA fields. One first approach to improve the quality of the data is to remove variables which are almost always NA (where mostly is determined to be 90%)

```
var_is_na <- sapply(training_set, function(x) mean(is.na(x))) > 0.9
training_set <- training_set[, var_is_na==F]
str(training_set)
```

```
## 'data.frame':    11776 obs. of  93 variables:
##  $ X                       : int  3 5 8 13 14 16 17 18 20 25 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 13
##  $ raw_timestamp_part_2    : int  820366 196328 440390 560359 576390 644302 692324 732306 788335 2831
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 12 12 12 12 12 12 12 12 13 ...
##  $ roll_belt               : num  1.42 1.48 1.42 1.42 1.42 1.48 1.51 1.55 1.59 1.53 ...
##  $ pitch_belt              : num  8.07 8.07 8.13 8.2 8.21 8.15 8.12 8.08 8.07 8.11 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","0.000673",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","0.006078",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","0.000000",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","0.000000",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","0.00","0.0000",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ gyros_belt_x            : num  0 0.02 0.02 0.02 0.02 0 0 0 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0.02 0 0 0 0 0 0.02 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 0 -0.02 0 -0.02 0 -0.02 0 ...
##  $ accel_belt_x            : int  -20 -21 -22 -22 -22 -21 -21 -21 -22 -19 ...
##  $ accel_belt_y            : int  5 2 4 4 4 4 4 5 5 4 ...
##  $ accel_belt_z            : int  23 24 21 21 21 23 22 21 22 21 ...
##  $ magnet_belt_x           : int  -2 -6 -2 -3 -8 0 -6 1 -1 -8 ...
##  $ magnet_belt_y           : int  600 600 603 606 598 592 598 600 604 605 ...
##  $ magnet_belt_z           : int  -305 -302 -313 -309 -310 -305 -317 -316 -314 -319 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -129 -129 -129 -129 -129 ...
##  $ pitch_arm               : num  22.5 22.1 21.8 21.4 21.4 21.3 21.3 21.2 21.1 20.7 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x             : num  0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.02 -0.02 ...
##  $ gyros_arm_y             : num  -0.02 -0.03 -0.02 -0.02 0 0 0 -0.02 -0.02 -0.02 ...
##  $ gyros_arm_z             : num  -0.02 0 0 -0.02 -0.03 -0.03 -0.02 -0.03 -0.02 0 ...
##  $ accel_arm_x             : int  -289 -289 -289 -287 -288 -289 -289 -288 -289 -289 ...
##  $ accel_arm_y             : int  110 111 111 111 111 109 110 108 109 109 ...
##  $ accel_arm_z             : int  -126 -123 -124 -124 -124 -121 -122 -124 -125 -123 ...
##  $ magnet_arm_x            : int  -368 -374 -372 -372 -371 -367 -371 -373 -373 -370 ...
##  $ magnet_arm_y            : int  344 337 338 338 331 340 337 336 335 340 ...
##  $ magnet_arm_z            : int  513 506 510 509 523 509 512 510 514 512 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","0.01388","0.01574",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","0.00000","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ..
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","0.00000","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ..
##  $ roll_dumbbell           : num  12.9 13.4 12.8 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.3 -70.4 -70.3 -70.8 -71 ...
##  $ yaw_dumbbell            : num  -85.1 -84.9 -85.1 -84.5 -84.3 ...
##  $ kurtosis_roll_dumbbell : Factor w/ 398 levels "","0.0016","-0.0035",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell: Factor w/ 401 levels "","0.0045","0.0130",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell  : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell : Factor w/ 401 levels "","0.0011","0.0014",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell: Factor w/ 402 levels "","-0.0053","0.0063",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell  : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_yaw_dumbbell       : Factor w/ 73 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_yaw_dumbbell       : Factor w/ 73 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_yaw_dumbbell : Factor w/ 3 levels "","0.00","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ total_accel_dumbbell   : int  37 37 37 37 37 37 37 36 37 37 ...
##  $ gyros_dumbbell_x       : num  0 0 0 0 0.02 0 0 0.02 0 0 ...
##  $ gyros_dumbbell_y       : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z       : num  0 0 0 -0.02 -0.02 0 0 -0.02 0 0 ...
##  $ accel_dumbbell_x       : int  -232 -233 -234 -234 -234 -233 -233 -231 -234 -234 ...
##  $ accel_dumbbell_y       : int  46 48 46 48 48 48 47 47 46 47 ...
##  $ accel_dumbbell_z       : int  -270 -270 -272 -269 -268 -271 -272 -268 -272 -271 ...
##  $ magnet_dumbbell_x      : int  -561 -554 -555 -552 -554 -554 -551 -557 -558 -555 ...
##  $ magnet_dumbbell_y      : int  298 292 300 302 295 297 296 292 302 290 ...
##  $ magnet_dumbbell_z      : num  -63 -68 -74 -69 -68 -73 -56 -62 -66 -68 ...
##  $ roll_forearm           : num  28.3 28 27.8 27.2 27.2 27.1 27.1 27 26.9 27.1 ...
##  $ pitch_forearm          : num  -63.9 -63.9 -63.8 -63.9 -63.9 -64 -64 -64 -64 -63.7 ...
##  $ yaw_forearm            : num  -152 -152 -152 -151 -151 -151 -151 -151 -151 -151 ...
##  $ kurtosis_roll_forearm  : Factor w/ 322 levels "","0.0128","-0.0227",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_forearm : Factor w/ 323 levels "","0.0012","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_forearm   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_forearm  : Factor w/ 323 levels "","-0.0004","-0.0013",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_forearm : Factor w/ 319 levels "","0.0000","-0.0113",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_forearm   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_yaw_forearm        : Factor w/ 45 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_yaw_forearm        : Factor w/ 45 levels "","0.0","-0.1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_yaw_forearm  : Factor w/ 3 levels "","0.00","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ total_accel_forearm    : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x        : num  0.03 0.02 0.02 0 0 0.02 0.02 0.02 0.02 0.05 ...
##  $ gyros_forearm_y        : num  -0.02 0 -0.02 0 -0.02 0 -0.02 0 -0.02 -0.03 ...
##  $ gyros_forearm_z       : num  0 -0.02 0 -0.03 -0.03 0 0 -0.02 0 0 ...
##  $ accel_forearm_x       : int  196 189 193 193 193 194 192 192 193 191 ...
##  $ accel_forearm_y       : int  204 206 205 205 202 204 204 206 205 202 ...
##  $ accel_forearm_z       : int  -213 -214 -213 -215 -214 -215 -213 -216 -215 -214 ...
##  $ magnet_forearm_x      : int  -18 -17 -9 -15 -14 -13 -13 -16 -9 -14 ...
##  $ magnet_forearm_y      : num  658 655 660 655 659 656 653 653 657 667 ...
##  $ magnet_forearm_z      : num  469 473 474 472 478 471 481 472 480 470 ...
##  $ classe                : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

This has reduced the number of variable to 93.

Another step we can take is to remove variables which don't actually vary that much and as such will not really be useful for predictions[1]

```
nzv <- nearZeroVar(training_set)
training_set <- training_set[, -nzv]
```

While many other improvements could be made (including stepwise testing, analysis of variance, etc) we will for this project only do something rather simple on top of the removing those who are mostly NA: remove variables which are obviously not relevant. This include X, user_name, and the timestamps and window variables, all concentrated near the beginning of the dataset whicn is useful for removing them.

```
training_set <- training_set[, -(1:7)]
str(training_set)
```

```
## 'data.frame':    11776 obs. of  52 variables:
##  $ pitch_belt          : num  8.07 8.07 8.13 8.2 8.21 8.15 8.12 8.08 8.07 8.11 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0.02 0.02 0.02 0 0 0 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0.02 0 0 0 0 0 0.02 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 0 -0.02 0 -0.02 0 -0.02 0 ...
##  $ accel_belt_x        : int  -20 -21 -22 -22 -22 -21 -21 -21 -22 -19 ...
##  $ accel_belt_y        : int  5 2 4 4 4 4 4 5 5 4 ...
##  $ accel_belt_z        : int  23 24 21 21 21 23 22 21 22 21 ...
##  $ magnet_belt_x       : int  -2 -6 -2 -3 -8 0 -6 1 -1 -8 ...
##  $ magnet_belt_y       : int  600 600 603 606 598 592 598 600 604 605 ...
##  $ magnet_belt_z       : int  -305 -302 -313 -309 -310 -305 -317 -316 -314 -319 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -129 -129 -129 -129 -129 ...
##  $ pitch_arm           : num  22.5 22.1 21.8 21.4 21.4 21.3 21.3 21.2 21.1 20.7 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.02 -0.02 ...
##  $ gyros_arm_y         : num  -0.02 -0.03 -0.02 -0.02 0 0 0 -0.02 -0.02 -0.02 ...
##  $ gyros_arm_z         : num  -0.02 0 0 -0.02 -0.03 -0.03 -0.02 -0.03 -0.02 0 ...
##  $ accel_arm_x         : int  -289 -289 -289 -287 -288 -289 -289 -288 -289 -289 ...
##  $ accel_arm_y         : int  110 111 111 111 111 109 110 108 109 109 ...
##  $ accel_arm_z         : int  -126 -123 -124 -124 -124 -121 -122 -124 -125 -123 ...
##  $ magnet_arm_x        : int  -368 -374 -372 -372 -371 -367 -371 -373 -373 -370 ...
##  $ magnet_arm_y        : int  344 337 338 338 331 340 337 336 335 340 ...
##  $ magnet_arm_z        : int  513 506 510 509 523 509 512 510 514 512 ...
##  $ roll_dumbbell       : num  12.9 13.4 12.8 13.4 13.4 ...
```

[1]This step is based on several contributions in the course forum

```
##  $ pitch_dumbbell      : num  -70.3 -70.4 -70.3 -70.8 -71 ...
##  $ yaw_dumbbell        : num  -85.1 -84.9 -85.1 -84.5 -84.3 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 36 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0.02 0 0 0.02 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 -0.02 0 0 -0.02 0 0 ...
##  $ accel_dumbbell_x    : int  -232 -233 -234 -234 -234 -233 -233 -231 -234 -234 ...
##  $ accel_dumbbell_y    : int  46 48 46 48 48 48 47 47 46 47 ...
##  $ accel_dumbbell_z    : int  -270 -270 -272 -269 -268 -271 -272 -268 -272 -271 ...
##  $ magnet_dumbbell_x   : int  -561 -554 -555 -552 -554 -554 -551 -557 -558 -555 ...
##  $ magnet_dumbbell_y   : int  298 292 300 302 295 297 296 292 302 290 ...
##  $ magnet_dumbbell_z   : num  -63 -68 -74 -69 -68 -73 -56 -62 -66 -68 ...
##  $ roll_forearm        : num  28.3 28 27.8 27.2 27.2 27.1 27.1 27 26.9 27.1 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.8 -63.9 -63.9 -64 -64 -64 -64 -63.7 ...
##  $ yaw_forearm         : num  -152 -152 -152 -151 -151 -151 -151 -151 -151 -151 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.02 0 0 0.02 0.02 0.02 0.02 0.05 ...
##  $ gyros_forearm_y     : num  -0.02 0 -0.02 0 -0.02 0 -0.02 0 -0.02 -0.03 ...
##  $ gyros_forearm_z     : num  0 -0.02 0 -0.03 -0.03 0 0 -0.02 0 0 ...
##  $ accel_forearm_x     : int  196 189 193 193 193 194 192 192 193 191 ...
##  $ accel_forearm_y     : int  204 206 205 205 202 204 204 206 205 202 ...
##  $ accel_forearm_z     : int  -213 -214 -213 -215 -214 -215 -213 -216 -215 -214 ...
##  $ magnet_forearm_x    : int  -18 -17 -9 -15 -14 -13 -13 -16 -9 -14 ...
##  $ magnet_forearm_y    : num  658 655 660 655 659 656 653 653 657 667 ...
##  $ magnet_forearm_z    : num  469 473 474 472 478 471 481 472 480 470 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

… and we're left with 52 variables.

Building our model

Armed with a cleaned dataset (or at least cleaner that the one we started with) it's time to build our model. We have decided on a Random Forest model as a starting point since it seemed appropriate to the task at hand.

```
## UNCOMMENT TO RUN THE MODEL FROM SCRATCH
##rf_model = train(classe ~ ., method="rf", data=training_set)
## Used once to cache the model, uncomment to overwrite it.
##saveRDS(rf_model, "rfmodel.rds")

## Load the previously saved model
rf_model = readRDS("rfmodel.rds")
```

The model is computed[2] and we can see the details

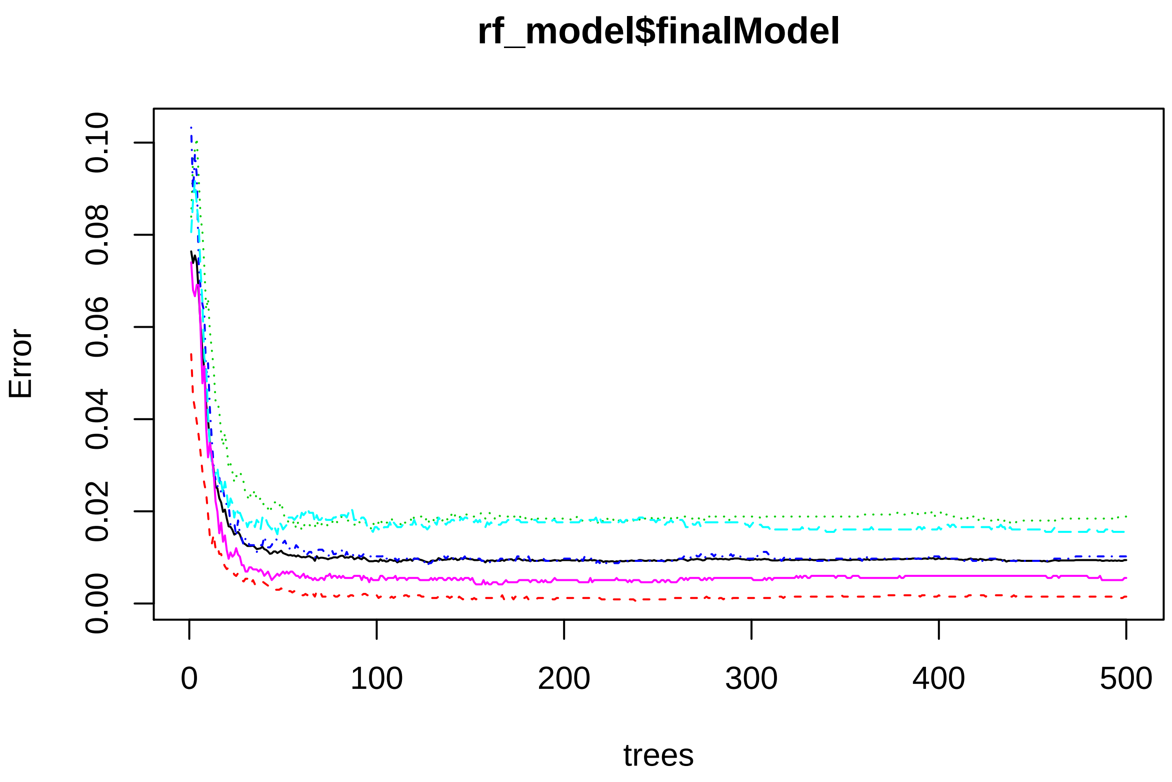
```
rf_model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
```

[2]And saved, again using a technic debated in the course forum and that avoid redoing the (expensive) calculation every time

```
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 26
##
##         OOB estimate of  error rate: 0.94%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3343    5    0    0    0 0.001493429
## B   26 2236   13    2    2 0.018867925
## C    0   17 2033    4    0 0.010223953
## D    0    1   26 1900    3 0.015544041
## E    0    0    4    8 2153 0.005542725
```

```
plot(rf_model$finalModel)
```



It chose 500 trees, 26 variables per split and a OOB estimate of error rate of 0.94%.

Model Evaluation

We will now use our model on the test date that we split earlier in order to gauge its accuracy: we will be using our model to guess the "classe" of each observation and then check if it was a correct guess.

A simple way to get an idea on the accuracy is simply to get a percentage of correct predictions

```
prediction <- predict(rf_model, testing_set)
mean(prediction == testing_set$class) * 100
```

```
## [1] 99.65588
```

Around 99.6%; we can complement this approach with a confusion matrix

```
confusionMatrix(testing_set$classe, prediction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    3    0    0    0
##          B    5 1512    1    0    0
##          C    0    5 1359    4    0
##          D    0    0    7 1279    0
##          E    0    0    0    2 1440
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.995, 0.9977)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9956
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9947   0.9941   0.9953   1.0000
## Specificity            0.9995   0.9991   0.9986   0.9989   0.9997
## Pos Pred Value         0.9987   0.9960   0.9934   0.9946   0.9986
## Neg Pred Value         0.9991   0.9987   0.9988   0.9991   1.0000
## Prevalence             0.2847   0.1937   0.1742   0.1638   0.1835
## Detection Rate         0.2841   0.1927   0.1732   0.1630   0.1835
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9986   0.9969   0.9964   0.9971   0.9998
```

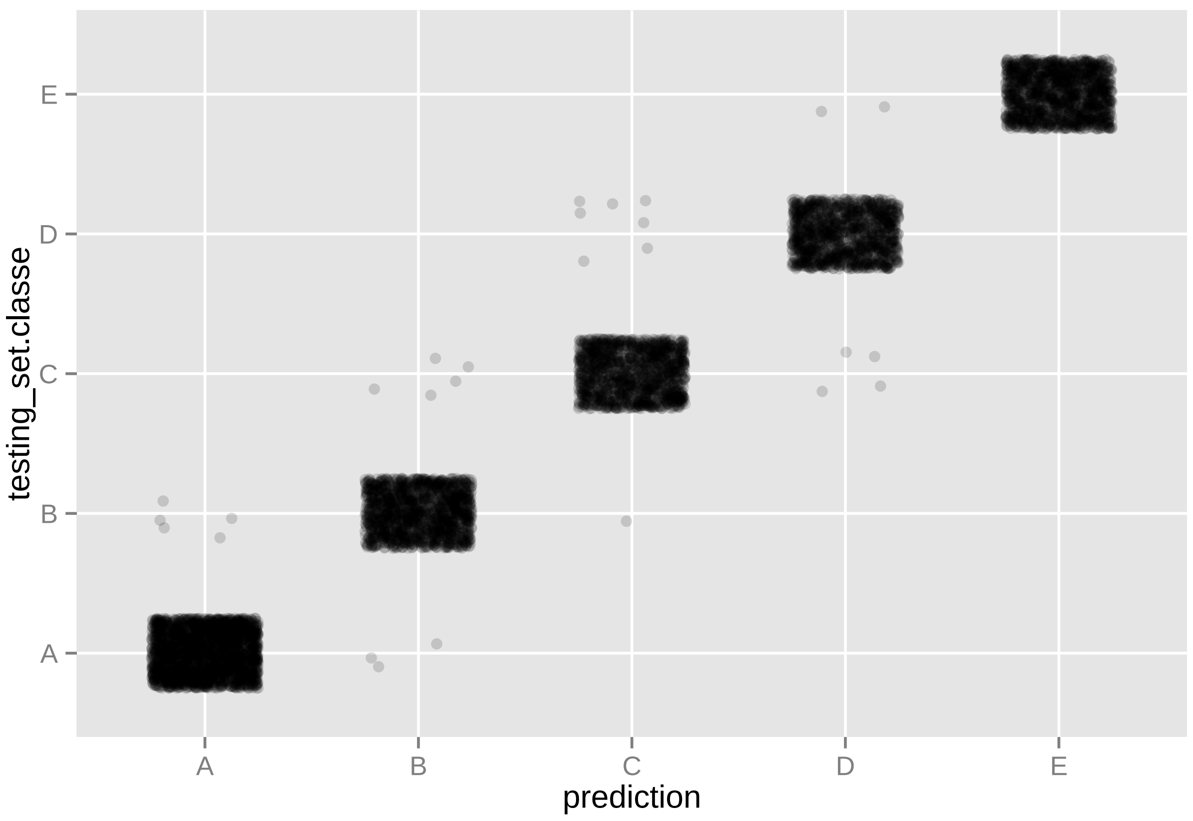
As we can see the values match and accuracy is around 99.6%. We perform additional validation by testing the original testing set (see Appendix).

Finally we compare the predictions that our model obtained with the real observations in the testing dataset.

```
results <- data.frame(prediction, testing_set$classe)
table(results)
```

```
##           testing_set.classe
## prediction    A    B    C    D    E
##          A 2229    5    0    0    0
##          B    3 1512    5    0    0
##          C    0    1 1359    7    0
##          D    0    0    4 1279    2
##          E    0    0    0    0 1440
```

```
ggplot(results, aes(prediction, testing_set.classe)) + geom_jitter(alpha = 0.15,position=position_jitte
```



The match is very good and reflects the high accuracy of the model.

Conclusion

The model we built to analye the data seems to be acurate to an error margin of ~0.5%. This is the percentage of predictions which were wrong. With this model we can very accuratly identify good and bad form, at least in terms of the data analysed.

Appendix 1: Course Project Submission of Test Cases

The second part of the assigment asks to make predictions for each of the 20 tests, write them to a file and submited. In the interest of completeness we have added the steps in this document, based on the instructions themselves[3] and comments in the course discussion forum (as well as individual exploration, of course).

```
## Use the original testing set
prediction <- predict(rf_model, pml_testing)
## Convert to char vector
prediction <- as.character(prediction)
## Function that writes to single files (from the instructions)
pml_write_files = function(x){
  n = length(x)
```

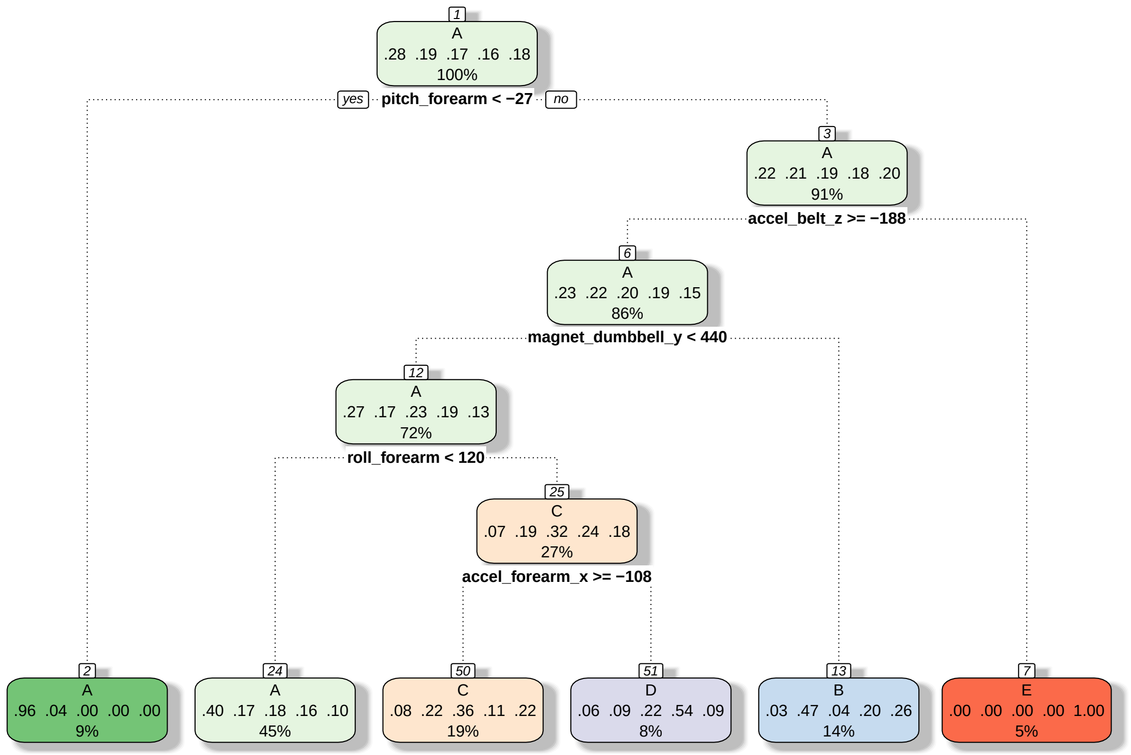
[3]cf. https://class.coursera.org/predmachlearn-033/assignment/view?assignment_id=5

```
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
## Write files
pml_write_files(prediction)
```

This should result in individual files ready to be submitted. # Appendix 2: An alternative model

As a complement to the main model we have added one additional approach which was attempted in the process but discarded in the early stages in favour or using Random Forests; this model uses regressive partitioning instead and can be useful to show the differences in model selection.

```
rp_model = train(classe ~ ., method="rpart", data=training_set)
fancyRpartPlot(rp_model$finalModel)
```



Rattle 2015−Oct−26 02:15:40 fsmunoz

This model has a much lower accuracy – below 50% actually.

```
prediction <- predict(rp_model, testing_set)
mean(prediction == testing_set$class) * 100
```

```
## [1] 48.38134
```

The confusion matrix clearly confirms this lack of accuracy.

```
confusionMatrix(testing_set$classe, prediction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2019   31  138   40    4
##          B  647  507  318   46    0
##          C  613   49  566  140    0
##          D  570  235  186  295    0
##          E  325  285  346   77  409
##
## Overall Statistics
##
##                Accuracy : 0.4838
##                  95% CI : (0.4727, 0.4949)
##     No Information Rate : 0.532
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3249
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4837  0.45799  0.36422  0.49331  0.99031
## Specificity            0.9420  0.84998  0.87254  0.86327  0.86103
## Pos Pred Value         0.9046  0.33399  0.41374  0.22939  0.28363
## Neg Pred Value         0.6161  0.90518  0.84748  0.95381  0.99938
## Prevalence             0.5320  0.14109  0.19806  0.07622  0.05264
## Detection Rate         0.2573  0.06462  0.07214  0.03760  0.05213
## Detection Prevalence   0.2845  0.19347  0.17436  0.16391  0.18379
## Balanced Accuracy      0.7129  0.65399  0.61838  0.67829  0.92567
```

References

Ugulino, Wallace, Débora Cardador, Katia Vega, Eduardo Velloso, Ruy Milidiú, and Hugo Fuks. 2012. "Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements." In *Advances in Artificial Intelligence-SBIA 2012*, 52–61. Springer.