Practical Machine Learning - Assignment

Sébastien Lievain

03/01/2017

Executive Summary

In this project, we will 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.
The goal is to predict the manner in which they did the exercise.

Data Loading and Cleaning

The data for this project comes from this source: http://groupware.les.inf.puc-rio.br/har.

```
library(caret)
library(randomForest)

dir.create("data", showWarnings = FALSE)

if(!file.exists("data/pml-training.csv") | !file.exists("data/pml-training.csv")) {
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
                  "data/pml-training.csv", method = "curl")
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
                  "data/pml-testing.csv", method = "curl")
}

pmlTraining <- read.csv("data/pml-training.csv", na.strings = c("#DIV/0!", "NA"))
testing <- read.csv("data/pml-testing.csv", na.strings = c("#DIV/0!", "NA"))
```

The `cvtd_timestamp` variable was loaded as a `factor` instead of a `Date`.
The `X` variable won't be of any interest in our study:

```
library(lubridate)

# transforming cvtd_timestamp variable into a Date
pmlTraining$cvtd_timestamp <- dmy_hm(pmlTraining$cvtd_timestamp)
testing$cvtd_timestamp <- dmy_hm(testing$cvtd_timestamp)

# Removing first column
pmlTraining <- pmlTraining[, !(names(pmlTraining) %in% c("X"))]
testing <- testing[, !(names(testing) %in% c("X"))]
```

Data has been captured by sensors during the practice of barbell lifts. Data is therfore temporal!
Two approaches are possible to analyse such data:

- Analysing all data points during the practice
- Aggregating data on small windows of time (2.5 sec here) and only analyse these aggregated data points.

The second approach can be challenging:

- How long should windows be?
- Should they all be of the same length?
- When to start and stop a window?

- Should windows overlap?

but also rewarding:

- Timely patterns can better be captured: e.g. the amplitude of the movement of the dumbbell can detect two wrong habits:
  - lifting the dumbbell only halfway (Class C),
  - lowering the dumbbell only halfway (Class D).
- Once data points have been aggregated (initial cost), computation cost is much cheaper for future analyses.

In the training dataset, these aggregated data points are indicated by the `new_window` variable. Unfortunatelly, no aggregated data point is present in the testing dataset!! Making impossible this second approach.

Therefore, all 100 aggregated features can be removed as well as ones about windows:

```
agg_features <- c(
    "min_roll_belt", "min_roll_dumbbell", "min_roll_arm", "min_roll_forearm",
    "min_pitch_belt", "min_pitch_dumbbell", "min_pitch_arm", "min_pitch_forearm",
    "min_yaw_belt", "min_yaw_dumbbell", "min_yaw_arm", "min_yaw_forearm",
    "max_roll_belt", "max_roll_dumbbell", "max_roll_arm", "max_roll_forearm",
    "max_picth_belt", "max_picth_dumbbell", "max_picth_arm", "max_picth_forearm",
    "max_yaw_belt", "max_yaw_dumbbell", "max_yaw_arm", "max_yaw_forearm",
    "amplitude_roll_belt", "amplitude_roll_dumbbell", "amplitude_roll_arm",
    "amplitude_roll_forearm", "amplitude_pitch_belt", "amplitude_pitch_dumbbell",
    "amplitude_pitch_arm", "amplitude_pitch_forearm", "amplitude_yaw_belt",
    "amplitude_yaw_dumbbell", "amplitude_yaw_arm", "amplitude_yaw_forearm",
    "kurtosis_roll_belt", "kurtosis_roll_dumbbell", "kurtosis_roll_arm",
    "kurtosis_roll_forearm", "kurtosis_picth_belt", "kurtosis_picth_dumbbell",
    "kurtosis_picth_arm", "kurtosis_picth_forearm", "kurtosis_yaw_belt",
    "kurtosis_yaw_dumbbell", "kurtosis_yaw_arm", "kurtosis_yaw_forearm",
    "skewness_roll_belt", "skewness_roll_dumbbell", "skewness_roll_arm",
    "skewness_roll_forearm", "skewness_roll_belt.1", "skewness_pitch_dumbbell",
    "skewness_pitch_arm", "skewness_pitch_forearm", "skewness_yaw_belt",
    "skewness_yaw_dumbbell", "skewness_yaw_arm", "skewness_yaw_forearm",
    "avg_roll_belt", "avg_roll_dumbbell", "avg_roll_arm", "avg_roll_forearm",
    "avg_pitch_belt", "avg_pitch_dumbbell", "avg_pitch_arm", "avg_pitch_forearm",
    "avg_yaw_belt", "avg_yaw_dumbbell", "avg_yaw_arm", "avg_yaw_forearm",
    "stddev_roll_belt", "stddev_roll_dumbbell", "stddev_roll_arm",
    "stddev_roll_forearm", "stddev_pitch_belt", "stddev_pitch_dumbbell",
    "stddev_pitch_arm", "stddev_pitch_forearm", "stddev_yaw_belt",
    "stddev_yaw_dumbbell", "stddev_yaw_arm", "stddev_yaw_forearm",
    "var_roll_belt", "var_roll_dumbbell", "var_roll_arm", "var_roll_forearm",
    "var_pitch_belt", "var_pitch_dumbbell", "var_pitch_arm", "var_pitch_forearm",
    "var_yaw_belt", "var_yaw_dumbbell", "var_yaw_arm", "var_yaw_forearm",
    "var_total_accel_belt", "var_accel_dumbbell", "var_accel_arm", "var_accel_forearm")

pmlTraining <- pmlTraining[, !(names(pmlTraining) %in% c(agg_features,
                                                          "new_window", "num_window"))]
```

Furthermore, Euler angles (roll, pitch and yaw) have been calculated from IMUs records (accelerometer, gyroscope and magnetometer). Therefore, only features from one out of the two systems should be used. I decided to use those from the Euler system:

```
IMU_features <- c(
    "gyros_belt_x", "gyros_dumbbell_x", "gyros_arm_x", "gyros_forearm_x",
```

```
    "gyros_belt_y", "gyros_dumbbell_y", "gyros_arm_y", "gyros_forearm_y",
    "gyros_belt_z", "gyros_dumbbell_z", "gyros_arm_z", "gyros_forearm_z",
    "accel_belt_x", "accel_dumbbell_x", "accel_arm_x", "accel_forearm_x",
    "accel_belt_y", "accel_dumbbell_y", "accel_arm_y", "accel_forearm_y",
    "accel_belt_z", "accel_dumbbell_z", "accel_arm_z", "accel_forearm_z",
    "magnet_belt_x", "magnet_dumbbell_x", "magnet_arm_x", "magnet_forearm_x",
    "magnet_belt_y", "magnet_dumbbell_y", "magnet_arm_y", "magnet_forearm_y",
    "magnet_belt_z", "magnet_dumbbell_z", "magnet_arm_z", "magnet_forearm_z")

pmlTraining <- pmlTraining[, !(names(pmlTraining) %in% c(IMU_features))]
```

Model creation

After several tests, random forest seems to be the best algorithm for the current application.

```
# First approach: all data points are taken into account
formula <- as.formula("classe ~ roll_belt + pitch_belt + yaw_belt + roll_dumbbell + roll_arm + roll_fore

modelFit <- train(formula, method = "rf", data = pmlTraining, importance = TRUE,
                  trControl = trainControl(formula, method = "cv", number = 10))
```

Cross Validation

By applying cross validation, the best tuning parameters were looked at.
For the random forest algorithm, only the `mtry` parameter (number of variables randomly sampled as candidates at each split) can be optimized.

```
print(modelFit$results)
```

```
##   mtry  Accuracy     Kappa  AccuracySD     KappaSD
## 1    2 0.9890425 0.9861399 0.002359333 0.002983538
## 2    7 0.9901639 0.9875588 0.002242806 0.002835999
## 3   12 0.9872084 0.9838214 0.003032257 0.003834458
```

We can see that the best accuracy (99.02%) is obtained with the `mtry` parameter set to 7.

Expected Out of Sample Error

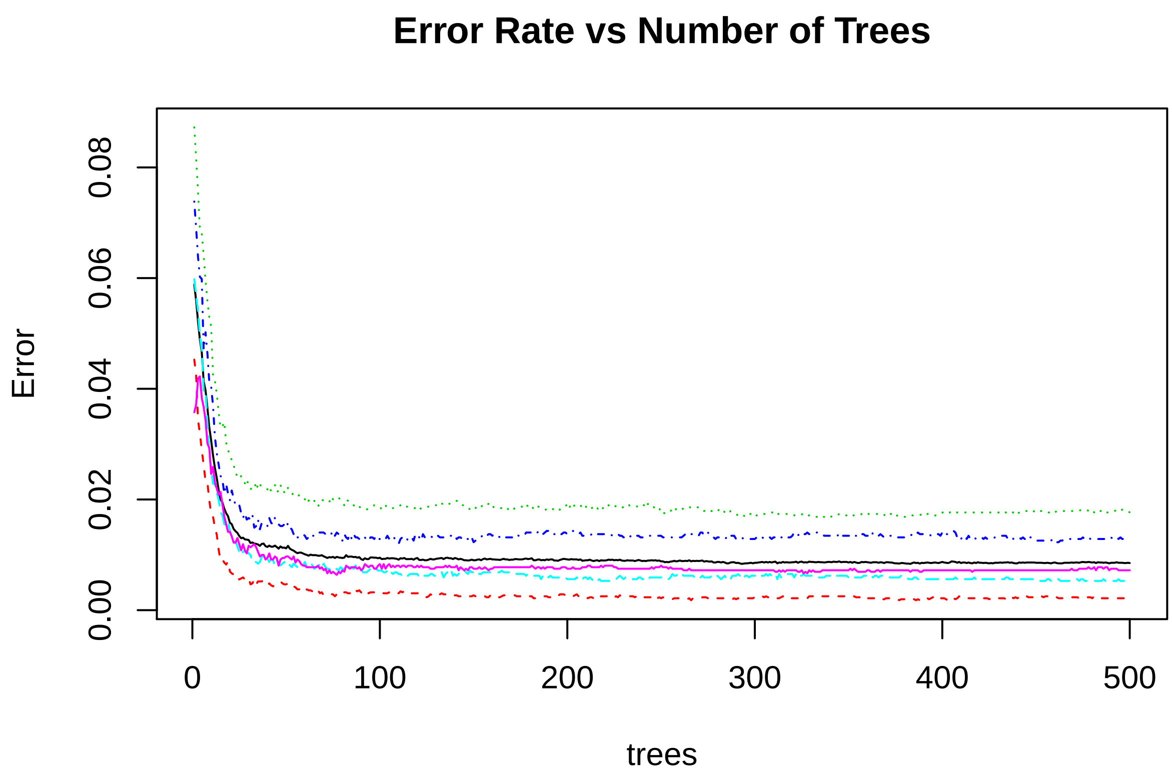
Our final Model has an out-of-bag error rate of 0.85%.
The associated confusion matrix is displayed below:

```
modelFit$finalModel$confusion
```

```
##      A    B    C    D    E class.error
## A 5568    9    0    2    1 0.002150538
## B   18 3730   44    4    1 0.017645510
## C    0   23 3377   19    3 0.013150205
## D    2    4    9 3199    2 0.005286070
## E    0    7   12    7 3581 0.007208206
```

For informational purposes a plot of the error rate versus number of trees is also shown.

```
plot(modelFit$finalModel, main = "Error Rate vs Number of Trees")
```

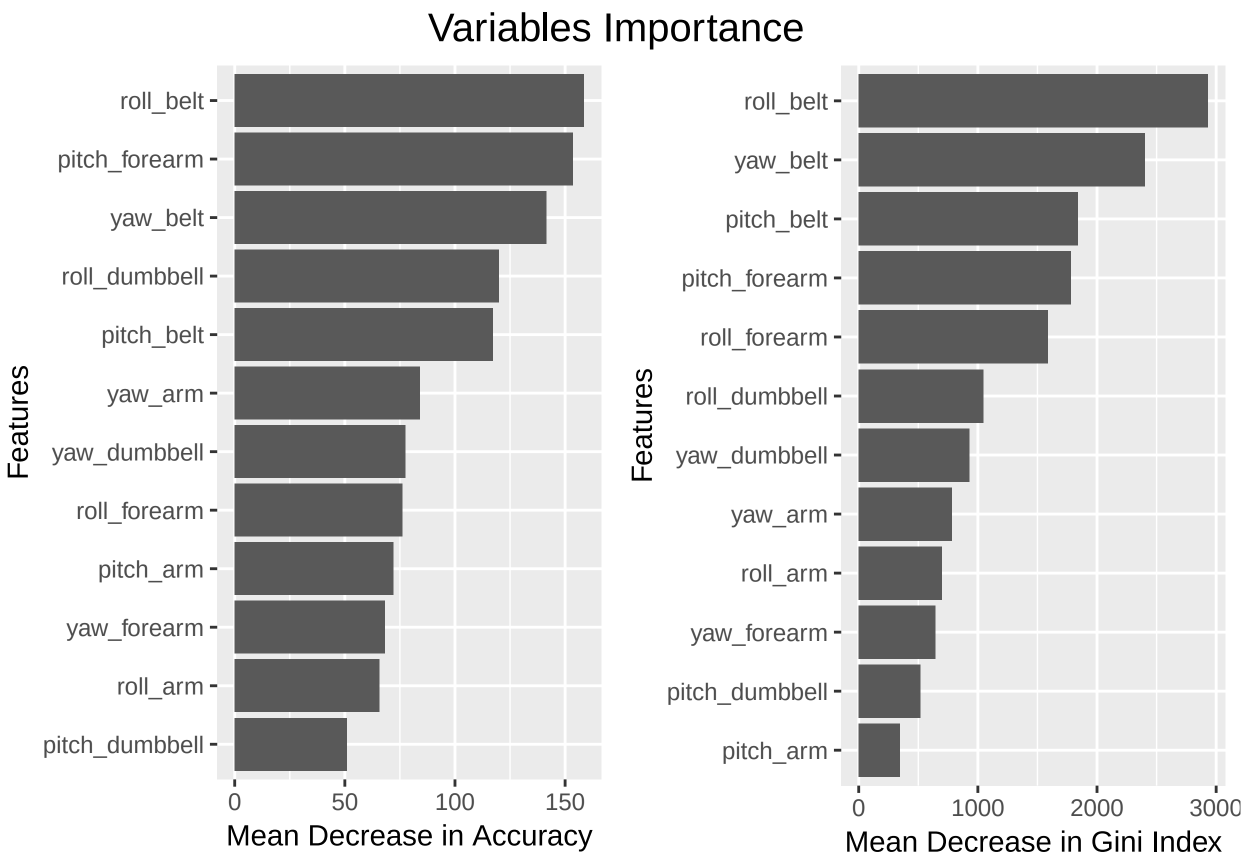


Variables importance

The importance of each variable can be represented by the Mean Decrease in Accuracy or Gini index:

```
library(grid)
library(gridExtra)
library(ggplot2)

importance <- as.data.frame(importance(modelFit$finalModel))
importance$features <- rownames(importance)

plot1 <- ggplot(importance, aes(
        x = reorder(features, MeanDecreaseAccuracy),
        y = MeanDecreaseAccuracy)
    ) +
    geom_bar(stat = "identity") +
    xlab("Features") +
    ylab("Mean Decrease in Accuracy") +
    coord_flip()

plot2 <- ggplot(importance, aes(
        x = reorder(features, MeanDecreaseGini),
        y = MeanDecreaseGini)
    ) +
    geom_bar(stat = "identity") +
    xlab("Features") +
    ylab("Mean Decrease in Gini Index") +
    coord_flip()
```

```r
grid.arrange(plot1, plot2, ncol = 2,
             top = textGrob("Variables Importance", gp = gpar(fontsize = 15, font = 8)))
```



Predicting classes from testing data

We can use the model we previously built to predict the class from testing data:

```r
predict(modelFit, newdata = testing)
```

```
##  [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
```

This ends up the assignment!

Annexe: Second Approach

For those who are interested in the second approach. I will cover it as much as possible below.

First, let's filter non-aggregated lines:

```r
# Second approach: only new window points are taken into account
# Reading file again
pmlTraining <- read.csv("data/pml-training.csv", na.strings = c("#DIV/0!", "NA"))
pmlTraining$cvtd_timestamp <- dmy_hm(pmlTraining$cvtd_timestamp)
pmlTraining <- pmlTraining[, !(names(pmlTraining) %in% c("X"))]

training_agg_lines <- pmlTraining[pmlTraining$new_window == "yes", ]
not_agg_features <- names(training_agg_lines)[!(names(training_agg_lines) %in% agg_features)]
```

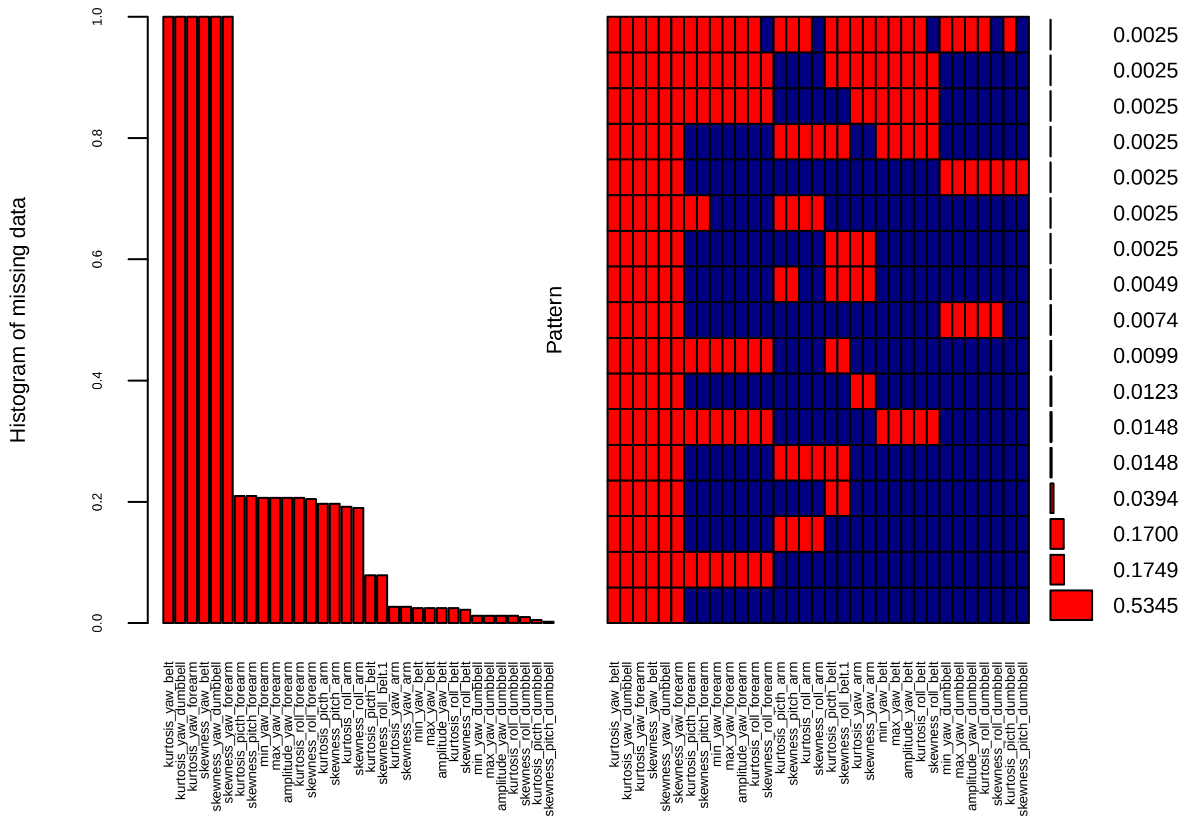
During the aggregation process certain statistics have not been calculated resulting in NAs:

```r
library(mice)
library(VIM)
library(pander)

#exploring missing data
countNA <- t(t(sapply(agg_features, function(feature) {
    sum(is.na(training_agg_lines[, feature]))
})))
featuresToInvestigate <- agg_features[countNA != 0]

pander(t(md.pattern(training_agg_lines[, featuresToInvestigate])), split.table = 100)
```

	217	16	5	69	1	3	6	1	2	1	71	4	1	6	1	1	1	
skewness_pitch_dumbbell	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
kurtosis_picth_dumbbell	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	2
skewness_roll_dumbbell	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	4
min_yaw_dumbbell	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	5
max_yaw_dumbbell	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	5
amplitude_yaw_dumbbell	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	5
kurtosis_roll_dumbbell	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	5
skewness_roll_belt	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	9
min_yaw_belt	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	10
max_yaw_belt	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	10
amplitude_yaw_belt	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	10
kurtosis_roll_belt	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	10
kurtosis_yaw_arm	1	1	0	1	0	1	1	1	0	1	1	1	1	1	0	0	0	11
skewness_yaw_arm	1	1	0	1	0	1	1	1	0	1	1	1	1	1	0	0	0	11
kurtosis_picth_belt	1	0	1	1	0	1	0	1	0	1	1	0	0	1	1	0	0	32
skewness_roll_belt.1	1	0	1	1	0	1	0	1	0	1	1	0	0	1	1	0	0	32
skewness_roll_arm	1	1	1	0	1	1	0	0	1	1	1	1	0	1	1	1	1	77
kurtosis_roll_arm	1	1	1	0	1	1	0	0	1	1	1	1	0	1	1	1	0	78
kurtosis_picth_arm	1	1	1	0	1	1	0	0	0	1	1	1	0	1	1	1	0	80
skewness_pitch_arm	1	1	1	0	1	1	0	0	0	1	1	1	0	1	1	1	0	80
skewness_roll_forearm	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	1	83
min_yaw_forearm	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	84
max_yaw_forearm	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	84
amplitude_yaw_forearm	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	84
kurtosis_roll_forearm	1	1	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	84

	217	16	5	69	1	3	6	1	2	1	71	4	1	6	1	1	1	
kurtosis_picth_forearm	1	1	1	1	1	1	1	0	1	1	0	0	1	0	0	0	0	85
skewness_pitch_forearm	1	1	1	1	1	1	1	0	1	1	0	0	1	0	0	0	0	85
kurtosis_yaw_belt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	406
kurtosis_yaw_dumbbell	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	406
kurtosis_yaw_forearm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	406
skewness_yaw_belt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	406
skewness_yaw_dumbbell	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	406
skewness_yaw_forearm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	406
	6	8	8	10	10	11	12	12	12	13	13	15	17	18	20	22	28	3502

```r
aggr(training_agg_lines[, featuresToInvestigate],
     col = c('navyblue','red'),
     numbers = TRUE, prop = TRUE, sortVars = TRUE,
     labels = names(training_agg_lines[, featuresToInvestigate]),
     cex.axis = .4, cex.lab = .7, cex.numbers = .7, gap = 1,
     ylab = c("Histogram of missing data","Pattern"))
```



The random forest algorithm cannot cope with NAs. Corresponding lines should be filtered out or missing values should be imputed.
Using the `mice` package, we can try to imput them:

```r
#imputing missing data
init <- mice(training_agg_lines, maxit = 0)
meth = init$method
predM = init$predictorMatrix
```

```r
predM[, c("user_name")] = 1
predM[, not_agg_features] = 0
predM[, agg_features] = 1

meth[not_agg_features] = ""
meth[agg_features] = "pmm"

imputed <- mice(training_agg_lines,
                method = meth, predictorMatrix = predM,
                m = 5, maxit = 50, seed = 500, print = FALSE)
completedData <- complete(imputed, 1)

countNA <- t(t(sapply(agg_features, function(feature) {
    sum(is.na(completedData[, feature]))
})))
featuresToDrop <- agg_features[countNA != 0]

cleanData <- completedData[, !(names(completedData) %in% featuresToDrop)]
```

Columns for which NAs remains have been filtered out.
We can now fit a model:

```r
featuresToKeep <- agg_features[ !(agg_features %in% featuresToDrop)]
formula <- as.formula(paste0("classe ~ ", paste(featuresToKeep, collapse = " + ")))

modelFit <- train(formula, method = "rf", data = cleanData,
                  trControl = trainControl(formula, method = "cv", number = 10))
```

Using cross-validation, we can see that we obtain an accuracy of 81.76% with the `mtry` parameter set to 2.

Our final Model has an out-of-bag error rate of 19.7%.
The associated confusion matrix is displayed below:

```r
modelFit$finalModel$confusion
```

```
##     A  B  C  D  E class.error
## A 100  5  2  1  1  0.08256881
## B  19 54  3  0  3  0.31645570
## C   7  9 51  2  1  0.27142857
## D   6  0  4 56  3  0.18840580
## E   3  7  2  2 65  0.17721519
```