Quantifing Quality of Exercise

Wei Ann Lim

24 July 2015

Executive Summary

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. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely qualify that activity.

Data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants were collected while they were performing barbell lifts correctly and incorrectly in 5 different ways. Accelerometers data are then labelled as A, B, C, D or E to represent the quality of lifts performed. The quality of the lifts associated with each rows of data are stored in the variable **classe**.

This write up describes how a model was built to allow prediction of the quality of barbell lift performed based on accelerometer data.

The data for building the model comes from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har), and they are available here https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

Building the model

Data Preparation

Before building the model, Exploratory Data Analysis and data cleaning is done.

```
names(train)
```

```
##   [1] "X"                        "user_name"
##   [3] "raw_timestamp_part_1"     "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"           "new_window"
##   [7] "num_window"               "roll_belt"
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
```

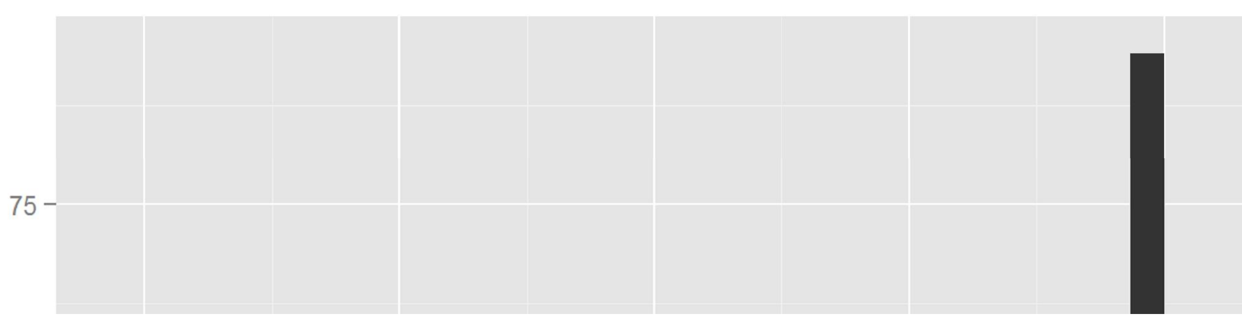
```
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

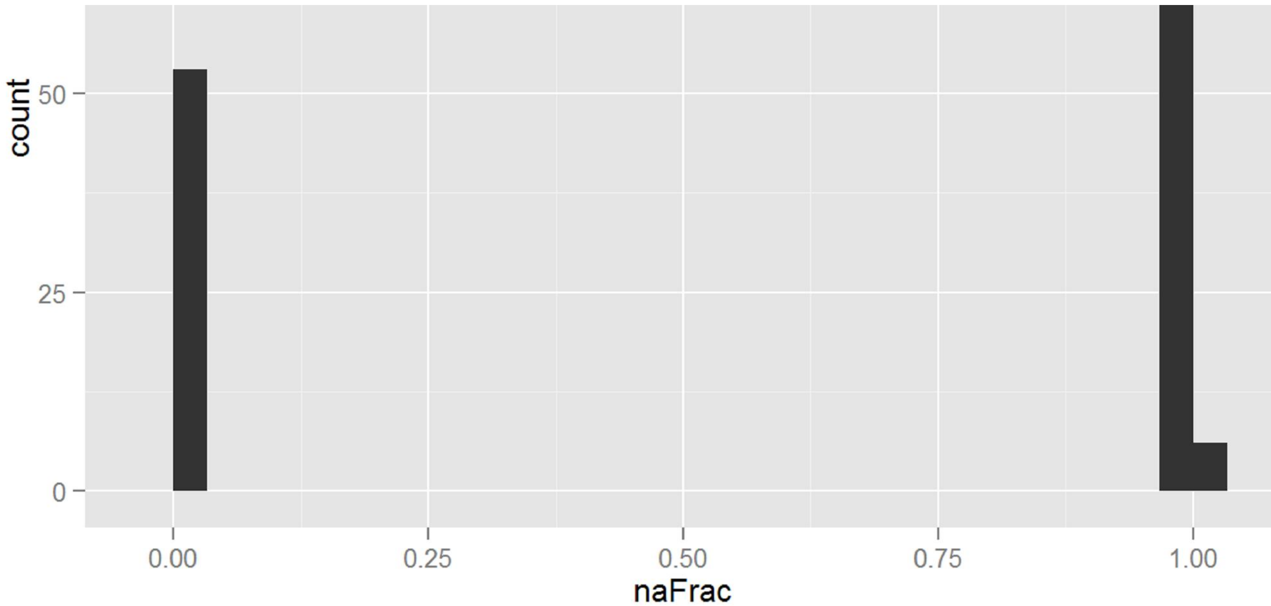
The first 7 variables, i.e. X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, num_window, are not sensor data. They will be remove from the analysis.

```
train <- train[, -(1:7)]
```

Now, let's look at how complete each column of data is.

```
# For each variable, calculate the percentage of NA values
naFrac <- apply(train, 2, function(x) {sum(is.na(x))/length(x)})
qplot(naFrac, geom = "histogram")
```





It was found that some variables consist of more than 90% NA values, while other variables have no NA values. Variables with more than 90% NA values, will not be useful for prediction, hence they will be removed from the dataset.

```
# Keep Variables with less than 90% NA values
train <- train[, naFrac < 0.9]
```

Highly correlated variables are also removed from the analysis. High correlated variables are found using the **findCorrelation()** function.

First, find the correlation between each variables.

```
corelation_mat <- cor(train[, -53]) # column 53 is where the "classe" variable is sto
red

# Set variable correlation with itself equal to 0
diag(corelation_mat) <- 0
```

Find variables with correlation > 0.75 and remove them.

```
# Cut off variables with corelation > 0.75
# Use the **findCorrelation** function
highlyCor <- findCorrelation(corelation_mat, 0.75)

# create a data.frame where the highly correlated variables are removed
train_sub <- train[, -highlyCor]
```

After these variables are removed, a **Random Forest** model is built with the remaining variables.

Random Forest

To estimate the test error, I used 5 fold cross validation. We set the parameters of the cross validation in the **trainControl()** function.

```
cvCtrl <- trainControl(method = "cv", number = 5)
```

Usually, the model will be more accurate when we increase the number of cross validation folds and the number of repeats. However, a bias-variance trade off is needed, so the number of folds cannot be infinitely large. Computing power also limits that number of folds. The default 5 folds is an appropriate compromise. Computing power also limits that number of folds.

With the **trainControl()** setup, I will train the model.

```
set.seed(2706)
rfFitCV <- train(classe ~ .,  data = train_sub, method = "rf", importance = TRUE, trC
ontrol = cvCtrl, allowParallel = TRUE)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
rfFitCV
```

```
## Random Forest
##
## 19622 samples
##    32 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 15696, 15698, 15699, 15697, 15698
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9931711  0.9913613  0.001614780  0.002042900
##   17    0.9922535  0.9902007  0.001437659  0.001818857
##   32    0.9859855  0.9822715  0.001770845  0.002240119
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

The expected out of sample error is 0.993, when the mtry is 2.