MachineLearningNovReport.Rmd

Jim Callahan

October 18-25, 2015

Executive Summary

People in the "quantified self movement" often quantify how much of a particular activity they do, but they rarely quantify how well they do it.

Telemetry data from weight lifters was gathered in a project described in "**Qualitative Activity Recognition of Weight Lifting Exercises**" a paper presented by Velloso, Bulling, Gellersen, Ugulino and Fuks at the **Augmented Human '13** ACM conference held in Stuttgart, Germany. Their paper is available on the web at: http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf

The goal of their project and this project is to build a model to predict the manner in which 6 participants did a weight lifting exercise from data gathered from from accelerometers (and other sensors) on the belt, arm and forearm of the participants as well as on the dumbell they lifted.

The participants were asked to perform barbell lifts correctly and then incorrectly in 5 different ways. The correct lift (labeled "A") and the four incorrect lifts (labeled "B" through "E") form the "**classe**" variable in the training set. The model predicting the "**classe**" variable was built using a subset of the variables in the training set.

The authors describe their instrumentation as: "For data recording we used four 9 degrees of freedom **Razor inertial measurement units (IMU)**, which provide three-axes acceleration, gyroscope and magnetometer data at a joint sampling rate of 45 Hz."

Exploratory Data Analysis (EDA)

The first step in working with new data is to verify the data matches any description supplied with the data. In this project there was a substantial mismatch between the published article and the data supplied so an extensive reconcilation process had to be undertaken.

The training data file provided on the website consisted of 160 variables. There was no data dictionary provided for the training or test files.

Therefore the data file had to be read in and selectively dumped, so reasonable assumptions could be made about what data to include in the analysis.

```
# read in data
rawtraining <- read.csv("~\\GitHub\\MachineLearning\\Data\\pml-training.csv",
                        stringsAsFactors = FALSE, na.strings = c("NA", ""))
# dump the names of the variables
# names(rawtraining)
rawtesting <- read.csv("~\\GitHub\\MachineLearning\\Data\\pml-testing.csv",
                        stringsAsFactors = FALSE, na.strings = c("NA", ""))
```

The first seven variables appear to be the "coordinates" of the rest of the data. In order to subset the data, the first seven variables will be refered to as "Block0" (block zero). Zero because there will be four additional data blocks one through four.

```r
Block0 <- c(1:7)
# str(rawtraining[ , Block0])
```

X: sequence number
The variable "**X**" appears to be a sequence number.

```r
# str(rawtraining$X)
# summary(rawtraining$X)
```

The data set has 19,622 observations (rows) and the **X** variable only runs from 1 to 19,620 (off by 2), but we can't sweat the small stuff. We probably need to exclude the sequence number from the analysis anyway to avoid potential "data leakage".

User name: the six participants
The six participants are identified in the variable, "**user_name**"

```r
summary(as.factor(rawtraining$user_name))
```

```
##   adelmo carlitos  charles   eurico   jeremy    pedro
##     3892     3112     3536     3070     3402     2610
```

The numbers underneath each name are the number of observations (rows) associated with each name. Although the numbers are not the same, it appears to reasonably well balanced. There does not appear to be an impossible to overcome class imbalance.

The target: classe
The last variable, "**classe**", should be the labels "A" through "E" indicating the correct ("A") and incorrect ("B" through "E") weight lifts.

```r
summary(as.factor(rawtraining$classe))
```

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

According to the paper, class "**A**" corresponds to the [correct] specified execution of the exercise, while the other 4 classes correspond to common mistakes."

A. Correct – "exactly according to the specification"
B. Incorrect – "throwing the elbows to the front"
C. Incorrect – "lifting the dumbbell only halfway"
D. Incorrect – "lowering the dumbbell only halfway"
E. Incorrect – "throwing the hips to the front"

The "**user_name**", "**classe**" and "**new_window**" variables should be permanently converted to factors.

```r
rawtraining$user_name <- as.factor(rawtraining$user_name)
rawtraining$classe    <- as.factor(rawtraining$classe)
rawtraining$new_window <- as.factor(rawtraining$new_window)
```

We should be able to tabulate "**user_name**" by "**classe**"

```r
t1 <- table(rawtraining$user_name, rawtraining$classe)
t1
```

```
##
##              A    B    C    D    E
##   adelmo   1165  776  750  515  686
##   carlitos  834  690  493  486  609
##   charles   899  745  539  642  711
##   eurico    865  592  489  582  542
##   jeremy   1177  489  652  522  562
##   pedro     640  505  499  469  497
```

```r
# row proportions
# round(prop.table(t1, 1),2)
# column proportions
# round(prop.table(t1, 2),2)
```

Window Number?
Each unique exercise (for which there may be serveral measurement observations) is presumably tracked by the "**num_window**" variable:

```r
length(unique(rawtraining$num_window))
```

```
## [1] 858
```

There seeem to be 858 "windows" numbered 1 through 864 with 6 "windows" not included. We should be able to tabulate "**user_name**" by "**num_window**" (omitted).

```r
# Multiple pages of output omitted
# t2 <- table(rawtraining$user_name, rawtraining$num_window)
# t2
```

Only one participant appears to be active in each "window" (all of the other observations are zero).

We can also look at "**classe**" by "**num_window**" (omitted).

```r
# Multiple pages of output omitted
# t3 <- table(rawtraining$classe, rawtraining$num_window)
# t3
```

Date and Time
There are three date or time variables in columns #3, #4 and #5.

```r
str(rawtraining[ , 3:5])
```

```
## 'data.frame':    19622 obs. of  3 variables:
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1323
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484434
##  $ cvtd_timestamp      : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 1
```

```
#
# Look at the time and "window" and variables.
# timewindowvars <- c("new_window", "num_window", "raw_timestamp_part_1", "raw_timestamp_part_2", #
# head(rawtraining[ , timewindowvars], 20)
```

Even the first six rows are problematic. Notice that the "**cvtd__timestamp**" and "**new__window**" are the same, but the window number, "**num__window**" changes. Why?

The change in "**num__window**" seems to be correlated with a change in part 1 of the the timestamp, "**raw__timestamp__part__1**". Unclear, what if anything, this means.

```
# summary(rawtraining[ , timewindowvars])
```

(Other) :11007

The converted timestamp variable, "**cvtd__timestamp**" has 11,007 "Other" values. What is "Other"? and why does the converted timestamp variable have "Other" when the part_1 and part_2 variables appear to be complete? Is "Other" blank, do I need to change how I read in the converted timestamp variable?

The other 152 variables
In the 160 variable data set, beyond "**classe**" and the 7 identifcation variables, the remaining 152 variables seem to be made up of **4 blocks** of **38 variables** each. This is different from the **96 features** described in the paper, but we have to work with the data we have and not rely on (differing) data descriptions in the paper.

```
4*38
```

```
## [1] 152
```

The **first block of 38 variables**, variables #8 through #45 appear to relate to the sensor on the **belt** of the participant.

```
# First Block of 38
# str(rawtraining[ , 8:45])
```

We want the raw accelerometer, magnetometer and gyroscopic data as well as the euler angle data (which may be synthesized from several measurements, but is not otherwise transformed).

So, from the first block we want the first four (#8, #9, #10 and #11) and the last nine (#37, #38, #39, #40, #41, #42, #43, #44, #45).

```
# First Block: untransformed data and Euler angle data
Block1 <- c(8:11, 37:45)
# str(rawtraining[ , Block1])
```

The **second block of 38 variables**, variables #46 through #83 appear to relate to the armband sensor on the **arm** of the participant.

```
# str(rawtraining[ , 46:83])
```

From the second block we again want the first four (#46, #47, #48 and #49) and but, the order of varaibles has changed. We want to skip 10 and the pick up the next nine (#60, #61, #62, #63, #64, #65, #66, #67 and #68), then skip the rest.

```
# Second Block: untransformed data and Euler angle data
Block2 <- c(46:49, 60:68)
# str(rawtraining[ , Block2])
```

The **third block of 38 variables**, variables #84 through #121 appear to relate to the sensor on the **dumbell** weight lifted by the participant.

```
# str(rawtraining[ , 84:121])
```

From the third block the order of variables has changed again, so we only want the first three (#84, #85 and #86) and then We want to skip 15 and the pick up only one (#102) and then skip another 10 and pick up the last 9 (#113, #114, #115, #116, #117, #118, #119, #120 and #121).

```
# Third Block: untransformed data and Euler angle data
Block3 <- c(84:86, 102, 113:121)
# str(rawtraining[ , Block3])
```

The **fourth block of 38 variables**, variables #122 through #159 appear to relate to the glove sensor on the **forearm** (wrist) of the participant.

```
# str(rawtraining[ , 122:159])
```

From the fourth block we only want the first three (#122, #123 and #124) and then we want to skip 15 and the pick up only one (#140) and then skip another 10 and pick up the last 9 (#151, #152, #153, #154, #155, #156, #157, #158 and #159) and the **classe** variable (#160).

```
# Fourth Block: untransformed data and Euler angle data
Block4 <- c(122:124, 140, 151:159, 160)
# str(rawtraining[ , Block4])
```

Revised Training Set
So, our revised training set will consist of blocks one through four.

```
# decided not to include Block zero (timestamp and ID variables)
training <- rawtraining[ ,c(Block1, Block2, Block3, Block4)]
str(training)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 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 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# Check for NAs (No NAs found in this subset other than datetime in Block0)
# summary(training)
```

Do the Random Forest

```
# Random Forests -- Template
## S3 method for class 'formula'
# randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
## Default S3 method:
# randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
# mtry=if (!is.null(y) && !is.factor(y))
# max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
# replace=TRUE, classwt=NULL, cutoff, strata,
# sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
# nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
# maxnodes = NULL,
# importance=FALSE, localImp=FALSE, nPerm=1,
# proximity, oob.prox=proximity,
# norm.votes=TRUE, do.trace=FALSE,
# keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
# keep.inbag=FALSE, ...)

# Proximity = Should proximity measure among the rows be calculated?

# Random Forests
library(caret)
```

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

```
# Set Seed
set.seed(1234)
## 10-fold Cross-Validation
# fitControl <- trainControl(## 10-fold CV
#                             method = "cv",
#                            number = 10)

# To Normalize or Not?   preProcess = c("center","scale"),

# This step ran more than 45 minutes with more than 7 gigs of RAM!

# modFit <- train(classe ~ ., data=training, method="rf",
#                  trControl = fitControl,
#                  importance = TRUE,
#                 proximity  = TRUE)
# modFit

## Look at variable importance:
# round(importance(modFit), 2)

# Get One Tree
# Tree2 <- getTree(modFit$finalModel,k=2)
```

```r
# library(rattle)
# fancyRpartPlot(Tree2)

# Predict New Values
# pred <- predict(modFit, testing)
# testing$predRight <- pred == testing$classe
# table(pred,testing$classe)

# Confusion Matrix
# confusionMatrix(data = pred, testing$classe)
```

```r
library(lattice)
library(ggplot2)
library(caret)
summary(training$classe)
```

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

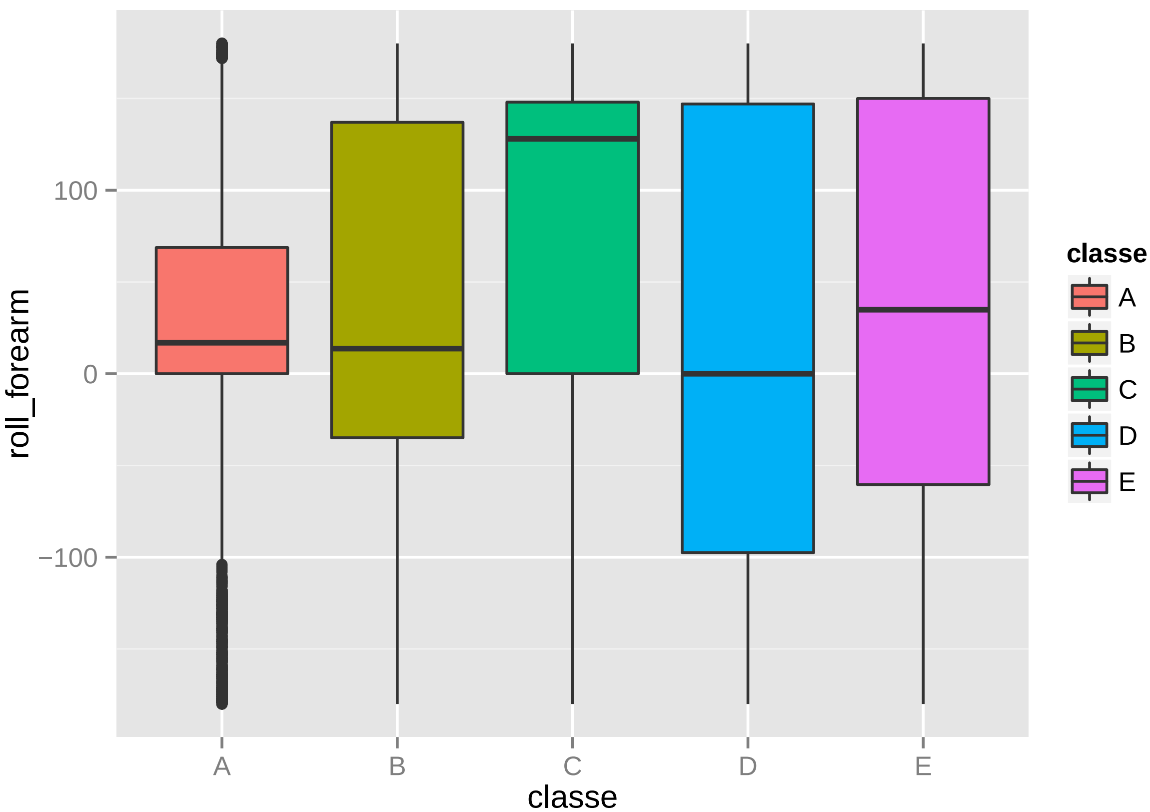
```r
ClassA <- training$classe == "A"
ClassB <- training$classe == "B"
summary(ClassA)
```

```
##    Mode   FALSE    TRUE    NA's
## logical   14042    5580       0
```

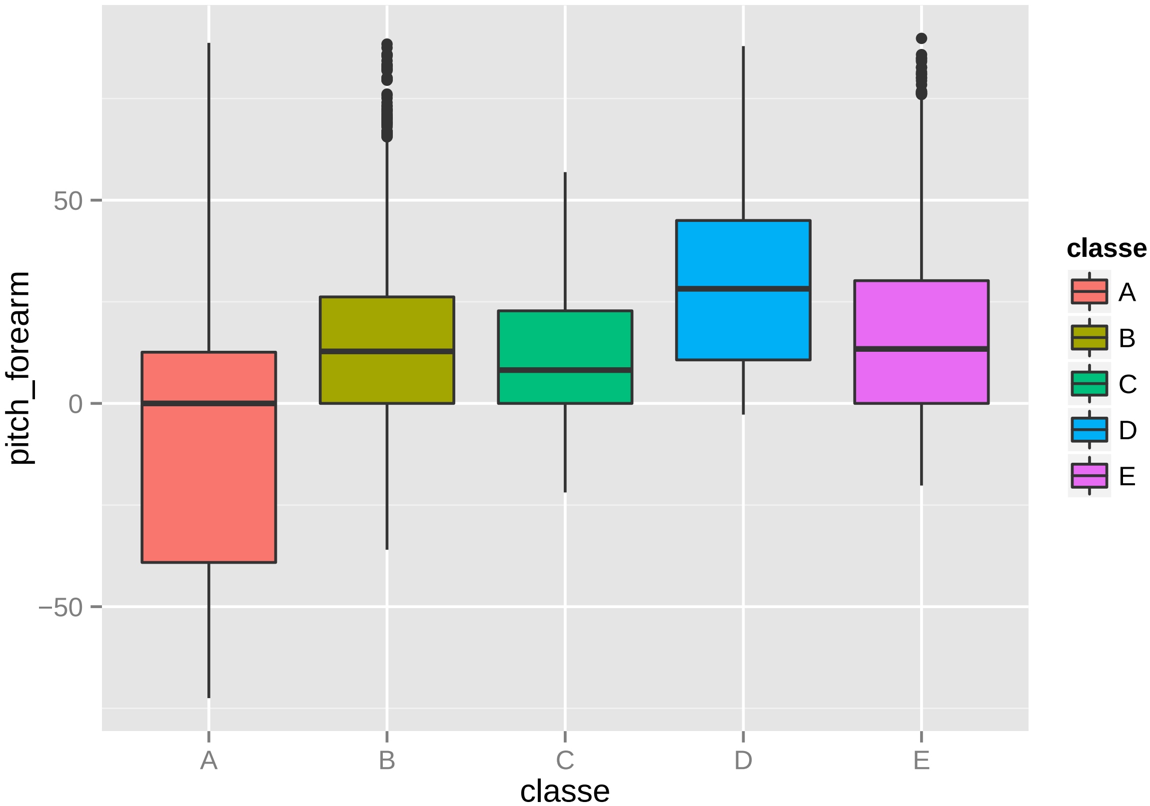
```r
summary(ClassB)
```

```
##    Mode   FALSE    TRUE    NA's
## logical   15825    3797       0
```

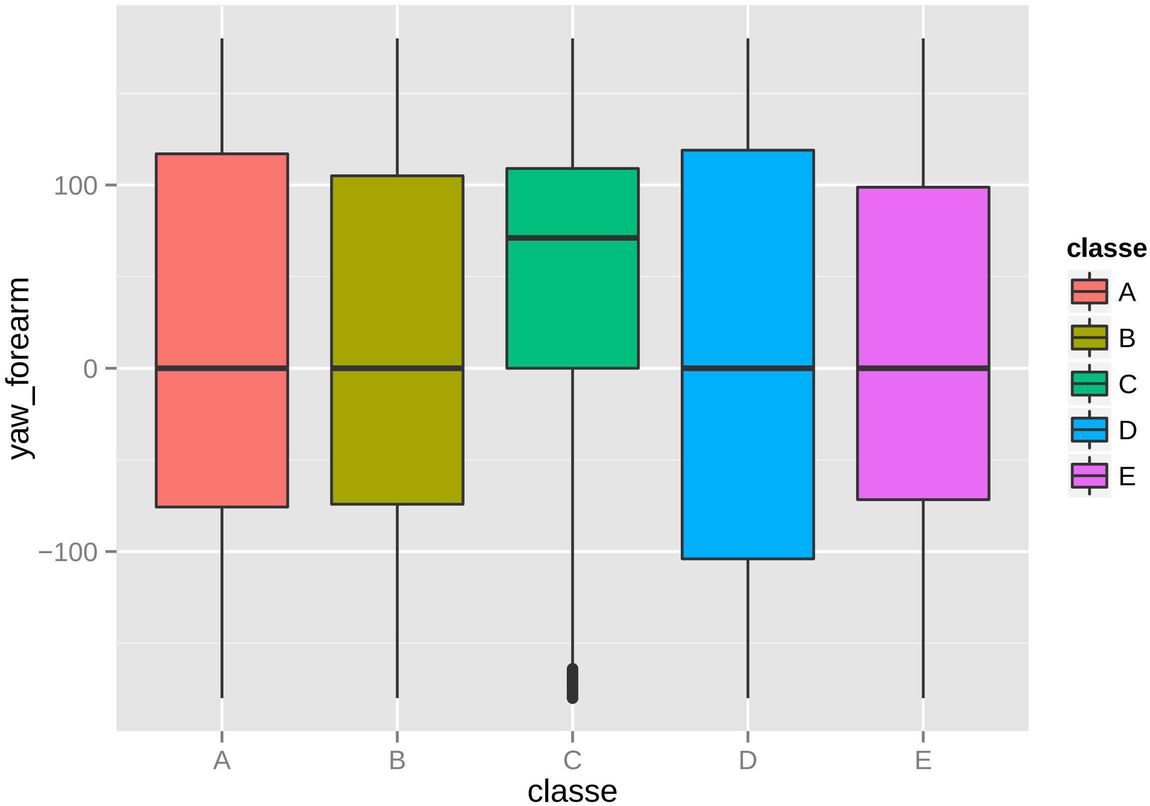
```r
# featurePlot(x=training[ClassA, c("roll_forearm","pitch_forearm","yaw_forearm")],
# y = training$classe[ClassA],
# plot="pairs")

# featurePlot(x=training[ClassB, c("roll_forearm","pitch_forearm","yaw_forearm")],
# y = training$classe[ClassB],
# plot="pairs")

p1 <- qplot(classe, roll_forearm, data=training, fill=classe,
geom=c("boxplot"))
p1
```




```
p2 <- qplot(classe, pitch_forearm, data=training, fill=classe,
geom=c("boxplot"))
p2
```




```
p3 <- qplot(classe, yaw_forearm, data=training, fill=classe,
geom=c("boxplot"))
p3
```



Preliminary Analysis

1. Fix converted timestamp
2. Normalize – preprocess – scale
3. Principal Components?
4. 10 fold cross-validation? ### Model Building ###
5. Naive Bayes / library(klaR) / nb() – did well in "Doing Data Science" NYT
6. KNN
7. Recursive Partitioning / libary(party) / ctree() – blog post http://www.r-bloggers.com/party-with-the-first-tribe/
8. Random Forest / library(randomForest) / rf() -or- cforest() a fancy method

"RF [Random Forests] thrives on variables–the more the better. There is no need for variable selection ,On a sonar data set with 208 cases and 60 variables, the RF error rate is 14%. Logistic Regression has a 50% error rate." Leo Breiman http://www.stat.berkeley.edu/~breiman/wald2002-2.pdf

Model Validation

1. confusion matrix – both training and test
2. AUC
3. Picture of Tree
4. 10 fold cross-validation
5. 20 predictions
6. Short paper

Conclusion

1. Subset the columns
2. Subset the rows (apparently not necessary in this project – I wasted a lot of time trying to understand the windows variables)
3. Check for NAs and impute values if necessary
4. Check whether numbers are of similar magnitude (Principal Components and KNN let biggest number dominate)
5. Split data for cross-validation (even though we have training and test data; I believe the peer evaluation asks about "cross validation")
6. Run the machine learning algorithm (the literature says the exact algorithm doesn't make much difference as long as your algorithm is appropriate to the task – supervised classification vs. unsupervised clustering, etc)
7. Generate the confusion matrix
8. Generate AUC curve
9. Generate graphics (picture of tree if you did a tree algorithm – data graph otherwise)
10. Do the 20 predictions
11. Write a very short paper describing what you did and the reasons for the choices you made.

Bibliography

DATA SOURCE:
Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino W.; Fuks, H.
"Qualitative Activity Recognition of Weight Lifting Exercises"
Proceedings of the 4th International Conference in Cooperation with SIGCHI
(Augmented Human '13), Stuttgart, Germany
ACM SIGCHI, 2013.
Available at:
http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf

SOFTWARE:

R
R Core Team (2015).
"**R: A language and environment for statistical computing**".
R Foundation for Statistical Computing, Vienna, Austria.
https://www.R-project.org/

Caret (R package)
by Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams,
Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton
Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew
Ziem, Luca Scrucca, Yuan Tang and Can Candan. (2015).
"**caret: Classification and Regression Training**".

R package version 6.0-57.
"**Building Predictive Models in R Using the caret Package**" Journal of Statistical Software 28(5), 1-26. http://www.jstatsoft.org/v28/i05/paper
http://CRAN.R-project.org/package=caret http://topepo.github.io/caret/index.html

kernlab (R package)
by Alexandros Karatzoglou, Alex Smola, Kurt Hornik, Achim Zeileis (2004).
"**kernlab - An S4 Package for Kernel Methods in R**".
Journal of Statistical Software 11(9), 1-20.
http://www.jstatsoft.org/v11/i09/
http://CRAN.R-project.org/package=kernlab

"**ggplot2 (R package)**"
by H. Wickham.
"**ggplot2: elegant graphics for data analysis**". Springer New York, 2009.
https://cran.r-project.org/package=ggplot2
http://ggplot2.org/book/

"**R Graphics Cookbook**"
by Winston Chang (O'Reilly).
Copyright 2013 Winston Chang, ISBN 978-1-449-31695-2.
http://oreil.ly/R_Graphics_Cookbook
http://www.cookbook-r.com/Graphs/

"**Doing Data Science**"
by Cathy O'Neil and Rachel Schutt (O'Reilly).
Copyright 2014 Cathy O'Neil and Rachel Schutt, ISBN 978-1-449-35865-5
http://oreil.ly/doing_data_science
http://mathbabe.org/

"**Data analysis with Open Source Tools**"
by Phillip K. Janert (O'Reilly).
Copyright 2011 Phillip K. Janert, ISBN 978-0-596-80235-6.
http://shop.oreilly.com/product/9780596802363.do
http://www.beyondcode.org/

"**DISTRIBUTION BASED TREES ARE MORE ACCURATE**" by Nong Shang and Leo Breiman ?
https://www.stat.berkeley.edu/~breiman/DB-CART.pdf

"**OUT-OF-BAG ESTIMATION**" by Leo Breiman

"**WALD LECTURE II: LOOKING INSIDE THE BLACK BOX**" by Leo Breiman http://www.stat.berkeley.edu/~breiman/wald2002-2.pdf

"**Boosting Tutorial**" by Ron Meir Machine Learning Summer School 2002 Technion Univerity, Israel http://webee.technion.ac.il/people/rmeir/BoostingTutorial.pdf

Wikipedia
https://en.wikipedia.org/wiki/AdaBoost
https://en.wikipedia.org/wiki/Autoencoder
https://en.wikipedia.org/wiki/Bootstrap_aggregating
https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation
https://en.wikipedia.org/wiki/Decision_tree_learning
https://en.wikipedia.org/wiki/Delphi_method
https://en.wikipedia.org/wiki/Design_matrix
https://en.wikipedia.org/wiki/Ensemble_learning
https://en.wikipedia.org/wiki/Linear_discriminant_analysis
https://en.wikipedia.org/wiki/Probably_approximately_correct_learning
https://en.wikipedia.org/wiki/Recommender_system

https://en.wikipedia.org/wiki/Quadratic_classifier
https://en.wikipedia.org/wiki/Random_forest