Classifying Movement in a Weight Lifting HAR Study

Douglas McLean, Peer Assessment, Machine Learning, Coursera

23rd December 2015

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

# Exective Summary

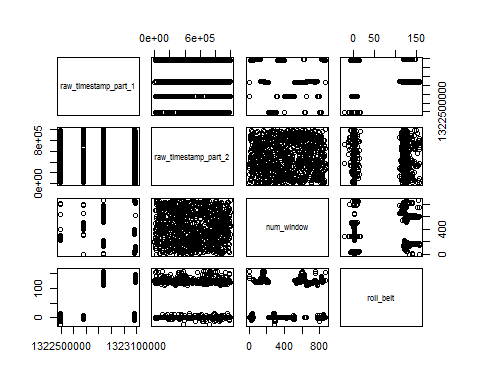
# Feature Selection

The training and test data: pml-training.csv and pml-testing.csv, respectively; were loaded. There were 19622 records in the training set and 20 records in the test set (that will be used for submitting predictions for grading and do not have an assigned classe). A considerable number of predictors in the test set are entirely composed of NA values and consequently we will be unable to predict with them. Despite the fact that they typically involve higher moments (e.g. skewness\_roll\_forearm) which may enable/encode information for variable transformations, we choose to discard these predictors since sample estimates of skewness and kurtosis are *notoriously* poorly estimated. Furthermore, since the variable new\_window takes the level no throughout the test set, then all records with level yes are discarded from the training set and the new\_window is discarded from both. This leaves the following predictors in the training set:

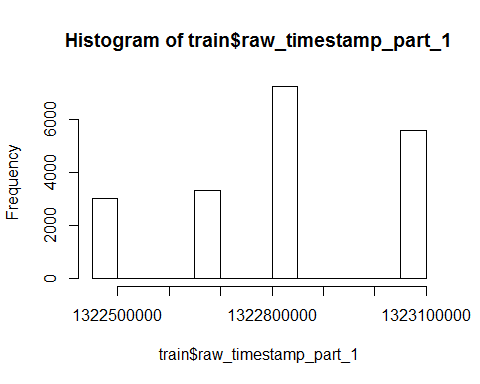
The date variable cvtd\_timestamp, once cast as a POSIXct date-class variable (and then written in seconds), has a *very* high correlation with the raw\_timestamp\_part\_1 variable: ; and a small correlation with the raw\_timestamp\_part\_2 variable: . It is therefore discarded from both training and test sets along with the variable X which is simply the row-index.

The raw\_timestamp\_part\_1 variable appears as though it could be modelled (approximately) as a factor variable in four levels:

s<-sample(1:nrow(train),1000)  
pairs(train[s,-1][,1:4])



hist(train$raw\_timestamp\_part\_1)



We fit a k-means cluster analysis to this variable in order to isolate the four levels:

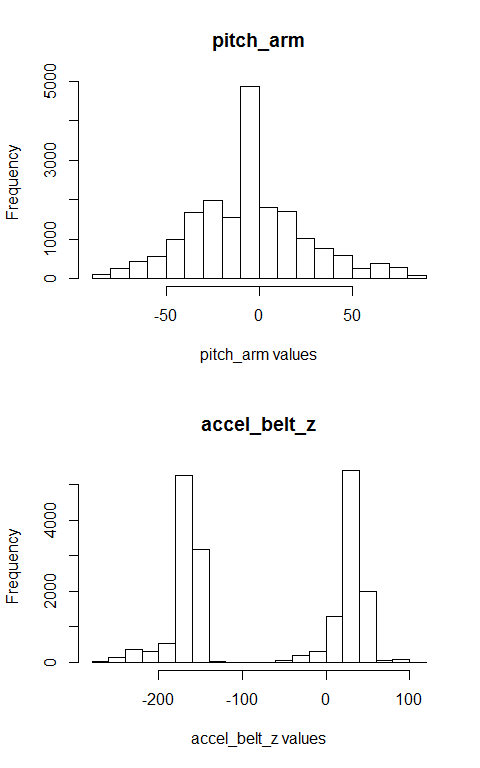
# K-Means Cluster Analysis  
fit <- kmeans(train$raw\_timestamp\_part\_1, 4)  
  
# 1. The 4 centers are:  
fit$centers

## [,1]  
## 1 1322489637  
## 2 1322673094  
## 3 1322489698  
## 4 1322945914

# 2. Overwrite raw\_timestamp\_part\_1 with the cluster centers  
raw\_timestamp\_part\_1 <- train$raw\_timestamp\_part\_1 # save  
train$raw\_timestamp\_part\_1 <- fit$cluster  
  
# 3. Cast as factor  
train$raw\_timestamp\_part\_1 <- as.factor(train$raw\_timestamp\_part\_1)  
  
# 4. Predict on test  
# First need to make a predict function :-(  
predict.kmeans <- function(km, data){  
 n <- length(km$centers)  
 m <- t(km$centers)[rep(1,length(data)),]  
 m <- abs(m-data)  
 apply(m,1,which.min)  
}  
test\_raw\_timestamp\_part\_1 <- test$raw\_timestamp\_part\_1 # save  
test$raw\_timestamp\_part\_1 <- predict.kmeans(fit,test$raw\_timestamp\_part\_1)  
test$raw\_timestamp\_part\_1 <- as.factor(test$raw\_timestamp\_part\_1)

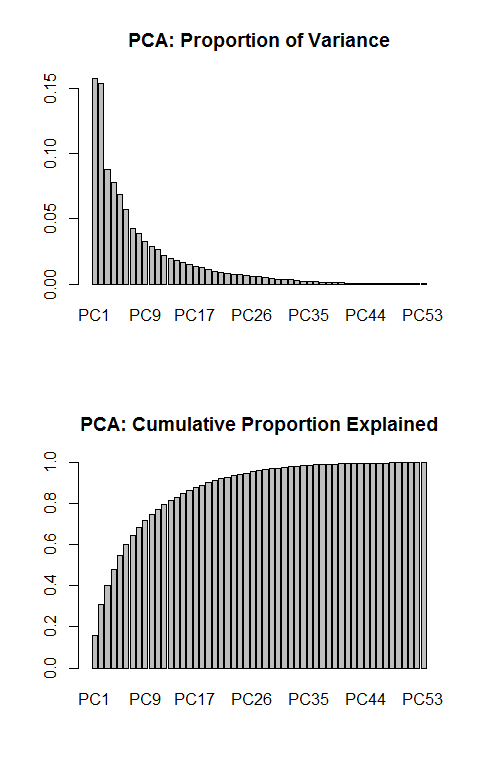
We now examine the remaining variables with histograms (omitted) to establish if there are any problem cases of repeated or missing values. Note that mean-centering and scaling is not necessary since we will be fitting a random forest model to the data. Regression trees are relatively robust to skewed data.

Although some predictors are unimodal in appearance: e.g. the pitch\_arm variable; some clearly are bi-modal: e.g. the accel\_belt\_z variable:



To the best of our knowledge, no variables have repeated zeros or NAs.

We assess the quantity of explanatory power in the predictors by computing a *principal components analysis*. This is applied to the train dataset and we drop the factor variables: user\_name, raw\_timestamp\_part\_1, raw\_timestamp\_part\_2; and outcome: classe; from the PCA.



We see that 95% of the variability is explained by the first 26 principal components (pc's). We might reasonably recast the machine learning problem of predicting classe by means of using these 26 pc's but in this work we use the full set of 56.

We now proceed to the fitting of regression trees using a stochastic gradient boosting technique.

# Model Fitting

The caret package will be used to fit the classification model which has classe as the outcome variable and the remaining 56 variables in the train (and also test) set as predictors. We begin by partitioning the train dataset into an *in sample* training set that we call training and an out-of-sample test set that we call testing (note the distinction between these and the prediction set test required to grade the final model by coursera).

inTrain <- createDataPartition(y=train$classe, p=0.7, list=FALSE)  
training <- train[ inTrain,]  
testing <- train[-inTrain,]

Now the gradient boosting model is fitted using the function caret::train and with method set to gbm with (the default) resampling setting of bootstrapping (boot):

model.Fitted <- TRUE  
if( model.Fitted ){  
 load("modFit.Rdat")  
} else {  
 modFit <- train(classe ~ ., method="gbm", data=training, verbose=FALSE)  
 save(modFit,file="modFit.Rdat")  
}  
print(modFit)

## Stochastic Gradient Boosting   
##   
## 13453 samples  
## 56 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
##   
## Summary of sample sizes: 13453, 13453, 13453, 13453, 13453, 13453, ...   
##   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa Accuracy SD  
## 1 50 0.7662212 0.7034306 0.004937523  
## 1 100 0.8306933 0.7855712 0.006028036  
## 1 150 0.8714580 0.8372538 0.005789650  
## 2 50 0.8894041 0.8598811 0.005112172  
## 2 100 0.9420651 0.9266577 0.004316847  
## 2 150 0.9628684 0.9529957 0.003750789  
## 3 50 0.9355478 0.9183755 0.003902703  
## 3 100 0.9699228 0.9619235 0.003350607  
## 3 150 0.9845484 0.9804421 0.002726602  
## Kappa SD   
## 0.006365679  
## 0.007714582  
## 0.007334571  
## 0.006490540  
## 0.005455057  
## 0.004739769  
## 0.004933203  
## 0.004248614  
## 0.003454432  
##   
## Tuning parameter 'shrinkage' was held constant at a value of 0.1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 150,  
## interaction.depth = 3 and shrinkage = 0.1.

n.repeats <- 3  
ctrl <- trainControl(method="repeatedcv",repeats=n.repeats)

To define -fold cross validation instead, we use the trainControl argument to caret::train function. In this instance, the 10-fold cross-validation is repeated 3 times:

model.Fitted.CV <- TRUE  
if( model.Fitted.CV ){  
 load("modFitCV.Rdat")  
} else {  
 modFitCV <- train(classe ~ ., method="gbm", data=training, verbose=FALSE, trControl=ctrl)  
 save(modFitCV,file="modFitCV.Rdat")  
}  
print(modFitCV)

## Stochastic Gradient Boosting   
##   
## 13453 samples  
## 56 predictor  
## 5 classes: 'A', 'B', 'C', 'D', 'E'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
##   
## Summary of sample sizes: 12109, 12108, 12107, 12109, 12107, 12107, ...   
##   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa Accuracy SD  
## 1 50 0.7700660 0.7084765 0.013647933  
## 1 100 0.8354283 0.7916655 0.011555192  
## 1 150 0.8778475 0.8453945 0.009388203  
## 2 50 0.8954894 0.8676445 0.009783814  
## 2 100 0.9467284 0.9325904 0.006290102  
## 2 150 0.9673934 0.9587359 0.006351012  
## 3 50 0.9408814 0.9251747 0.007192163  
## 3 100 0.9750742 0.9684573 0.004868617  
## 3 150 0.9888259 0.9858623 0.003755350  
## Kappa SD   
## 0.017430186  
## 0.014630157  
## 0.011894320  
## 0.012428738  
## 0.007965119  
## 0.008040969  
## 0.009101823  
## 0.006165180  
## 0.004752197  
##   
## Tuning parameter 'shrinkage' was held constant at a value of 0.1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 150,  
## interaction.depth = 3 and shrinkage = 0.1.

# Prediction

# Summary