Assessing the Quality of Activity

by clearwriter, February 2016

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 on the original research is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Source

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Load Libraries and Prepare Datasets

Load libraries first.

```
library(caret)
```

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

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

```
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(sjPlot)
library(knitr)
library(captioner)
library(doMC)
require(data.table)
```

```
## Loading required package: data.table
```

```
set.seed(1234)
```

Download testing and training data to your working directory.

```
## Load training data.
url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
training <- fread(url)
url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
testing <- fread(url)
dim(training); dim(testing);
```

```
## [1] 19622   160
```

```
## [1]  20 160
```

Identify Predictor Candidates

Which variables in the test dataset have zero NAs? Use this tip: finding columns with all missing values in r.

Belt, arm, dumbbell, and forearm variables that do not have any missing values in the test dataset will be predictor candidates.

```
isAnyMissing <- sapply(testing, function (x) any(is.na(x) | x == ""))
isPredictor <- !isAnyMissing & grepl("belt|[^(fore)]arm|dumbbell|forearm", names(i
sAnyMissing))
predCandidates <- names(isAnyMissing)[isPredictor]
predCandidates
```

```
##  [1] "roll_belt"             "pitch_belt"            "yaw_belt"
##  [4] "total_accel_belt"      "gyros_belt_x"          "gyros_belt_y"
##  [7] "gyros_belt_z"          "accel_belt_x"          "accel_belt_y"
## [10] "accel_belt_z"          "magnet_belt_x"         "magnet_belt_y"
## [13] "magnet_belt_z"         "roll_arm"              "pitch_arm"
## [16] "yaw_arm"               "total_accel_arm"       "gyros_arm_x"
## [19] "gyros_arm_y"           "gyros_arm_z"           "accel_arm_x"
## [22] "accel_arm_y"           "accel_arm_z"           "magnet_arm_x"
## [25] "magnet_arm_y"          "magnet_arm_z"          "roll_dumbbell"
## [28] "pitch_dumbbell"        "yaw_dumbbell"          "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"      "gyros_dumbbell_y"      "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"      "accel_dumbbell_y"      "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"     "magnet_dumbbell_y"     "magnet_dumbbell_z"
## [40] "roll_forearm"          "pitch_forearm"         "yaw_forearm"
## [43] "total_accel_forearm"   "gyros_forearm_x"       "gyros_forearm_y"
## [46] "gyros_forearm_z"       "accel_forearm_x"       "accel_forearm_y"
## [49] "accel_forearm_z"       "magnet_forearm_x"      "magnet_forearm_y"
## [52] "magnet_forearm_z"
```

Next, we want to subset the primary dataset to include only the predictor candidates and the outcome variable, classe.

```
varToInclude <- c("classe", predCandidates)
training <- training[, varToInclude, with=FALSE]
dim(training)
```

```
## [1] 19622    53
```

```
names(training)
```

```
##  [1] "classe"                 "roll_belt"              "pitch_belt"
##  [4] "yaw_belt"               "total_accel_belt"       "gyros_belt_x"
##  [7] "gyros_belt_y"           "gyros_belt_z"           "accel_belt_x"
## [10] "accel_belt_y"           "accel_belt_z"           "magnet_belt_x"
## [13] "magnet_belt_y"          "magnet_belt_z"          "roll_arm"
## [16] "pitch_arm"              "yaw_arm"                "total_accel_arm"
## [19] "gyros_arm_x"            "gyros_arm_y"            "gyros_arm_z"
## [22] "accel_arm_x"            "accel_arm_y"            "accel_arm_z"
## [25] "magnet_arm_x"           "magnet_arm_y"           "magnet_arm_z"
## [28] "roll_dumbbell"          "pitch_dumbbell"         "yaw_dumbbell"
## [31] "total_accel_dumbbell"   "gyros_dumbbell_x"       "gyros_dumbbell_y"
## [34] "gyros_dumbbell_z"       "accel_dumbbell_x"       "accel_dumbbell_y"
## [37] "accel_dumbbell_z"       "magnet_dumbbell_x"      "magnet_dumbbell_y"
## [40] "magnet_dumbbell_z"      "roll_forearm"           "pitch_forearm"
## [43] "yaw_forearm"            "total_accel_forearm"    "gyros_forearm_x"
## [46] "gyros_forearm_y"        "gyros_forearm_z"        "accel_forearm_x"
## [49] "accel_forearm_y"        "accel_forearm_z"        "magnet_forearm_x"
## [52] "magnet_forearm_y"       "magnet_forearm_z"
```

And then we convert classe into a factor.

```
training <- training[, classe := factor(training[, classe])]
training[, .N, classe]
```

```
##    classe    N
## 1:      A 5580
## 2:      B 3797
## 3:      C 3422
## 4:      D 3216
## 5:      E 3607
```

As we’ve learned, we split the dataset into 60/40 training/probing.

```
inTrain <- createDataPartition(training$classe, p=0.6)
DTrain <- training[inTrain[[1]]]
DProbe <- training[-inTrain[[1]]]
```

We preprocess the prediction variables by centering and scaling.

```
X <- DTrain[, predCandidates, with=FALSE]
preProc <- preProcess(X)
preProc
```

```
## Created from 11776 samples and 52 variables
##
## Pre-processing:
##   - centered (52)
##   - ignored (0)
##   - scaled (52)
```

```
XCS <- predict(preProc, X)
DTrainCS <- data.table(data.frame(classe = DTrain[, classe], XCS))
```

And then apply the centering and scaling to our probing dataset.

```
X <- DProbe[, predCandidates, with=FALSE]
XCS <- predict(preProc, X)
DProbeCS <- data.table(data.frame(classe = DProbe[, classe], XCS))
```

We also need to check for near zero variance.

```
nzv <- nearZeroVar(DTrainCS, saveMetrics=TRUE)
if (any(nzv$nzv)) nzv else message("No variables with near zero variance")
```

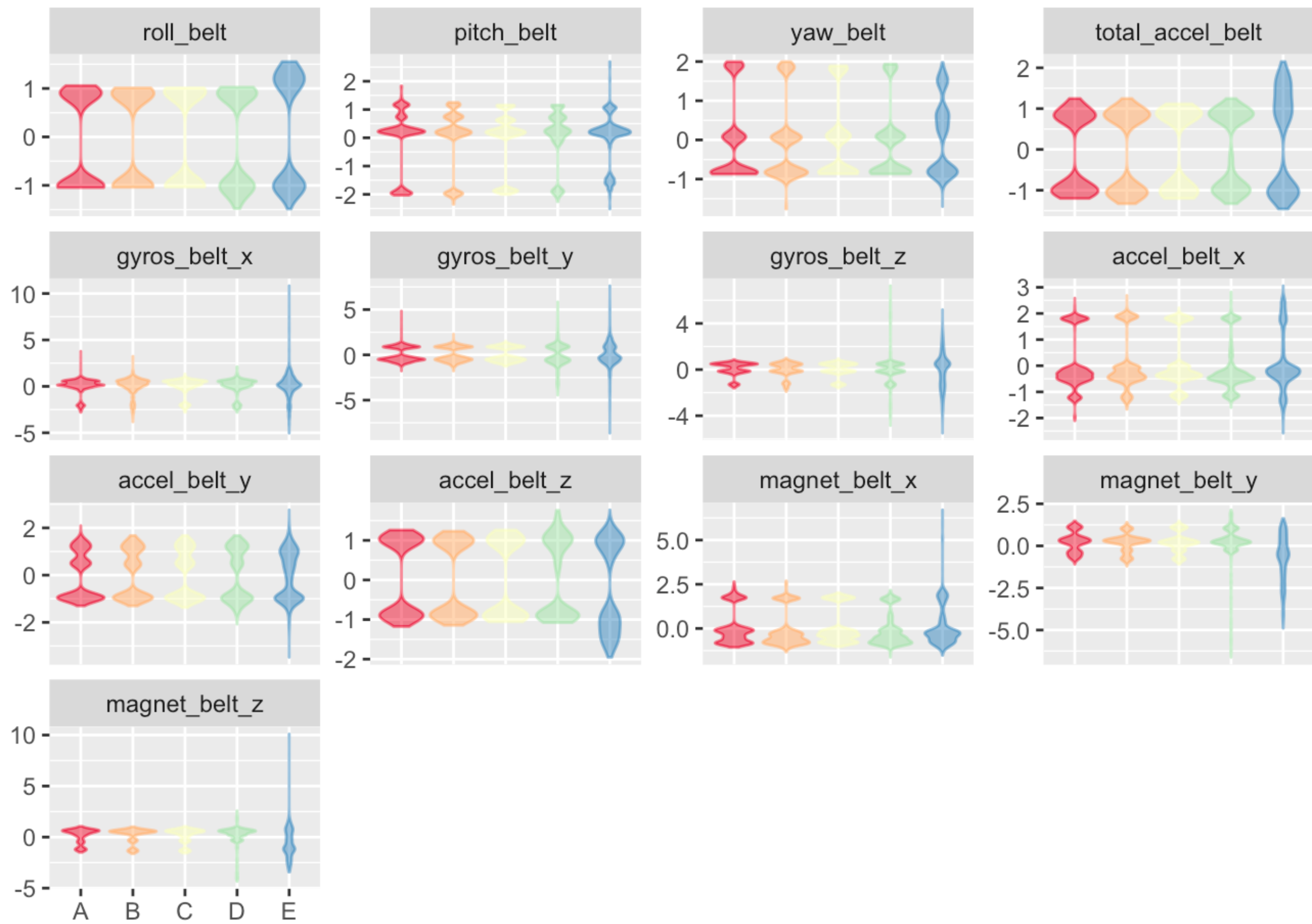
```
## No variables with near zero variance
```

Now, let’s examine our groups of prediction variables.

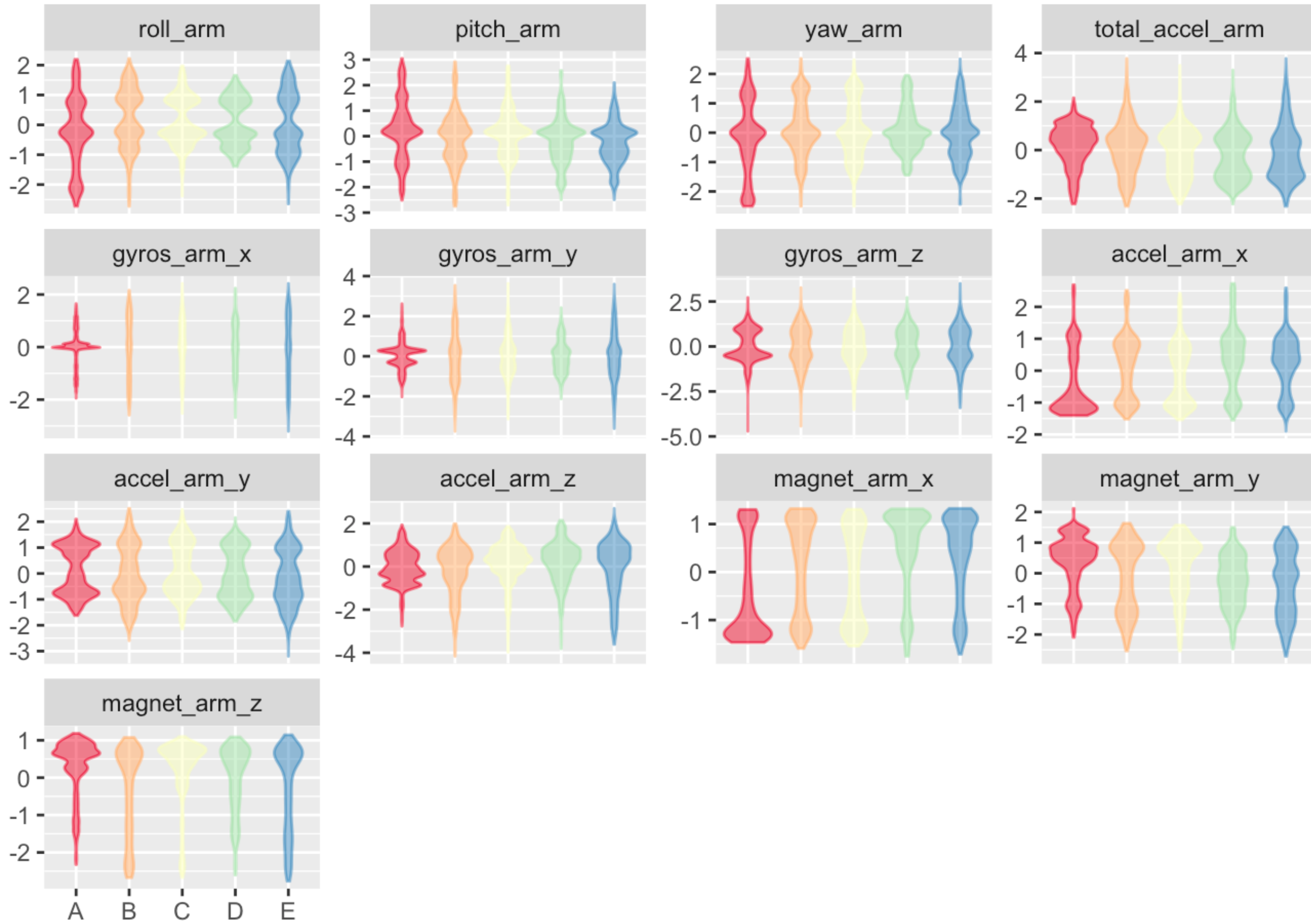
```
histGroup <- function (data, regex) {
  col <- grep(regex, names(data))
  col <- c(col, which(names(data) == "classe"))
  library(reshape2)
  n <- nrow(data)
  DMelted <- melt(data[, col, with=FALSE][, rownum := seq(1, n)], id.vars=c("rownu
m", "classe"))
  library(ggplot2)
  ggplot(DMelted, aes(x=classe, y=value)) +
    geom_violin(aes(color=classe, fill=classe), alpha=1/2) +
#     geom_jitter(aes(color=classe, fill=classe), alpha=1/10) +
#     geom_smooth(aes(group=1), method="gam", color="black", alpha=1/2, size=2) +
    facet_wrap(~ variable, scale="free_y") +
    scale_color_brewer(palette="Spectral") +
    scale_fill_brewer(palette="Spectral") +
    labs(x="", y="") +
    theme(legend.position="none")
}
histGroup(DTrainCS, "belt")
```

```
##
## Attaching package: 'reshape2'
```

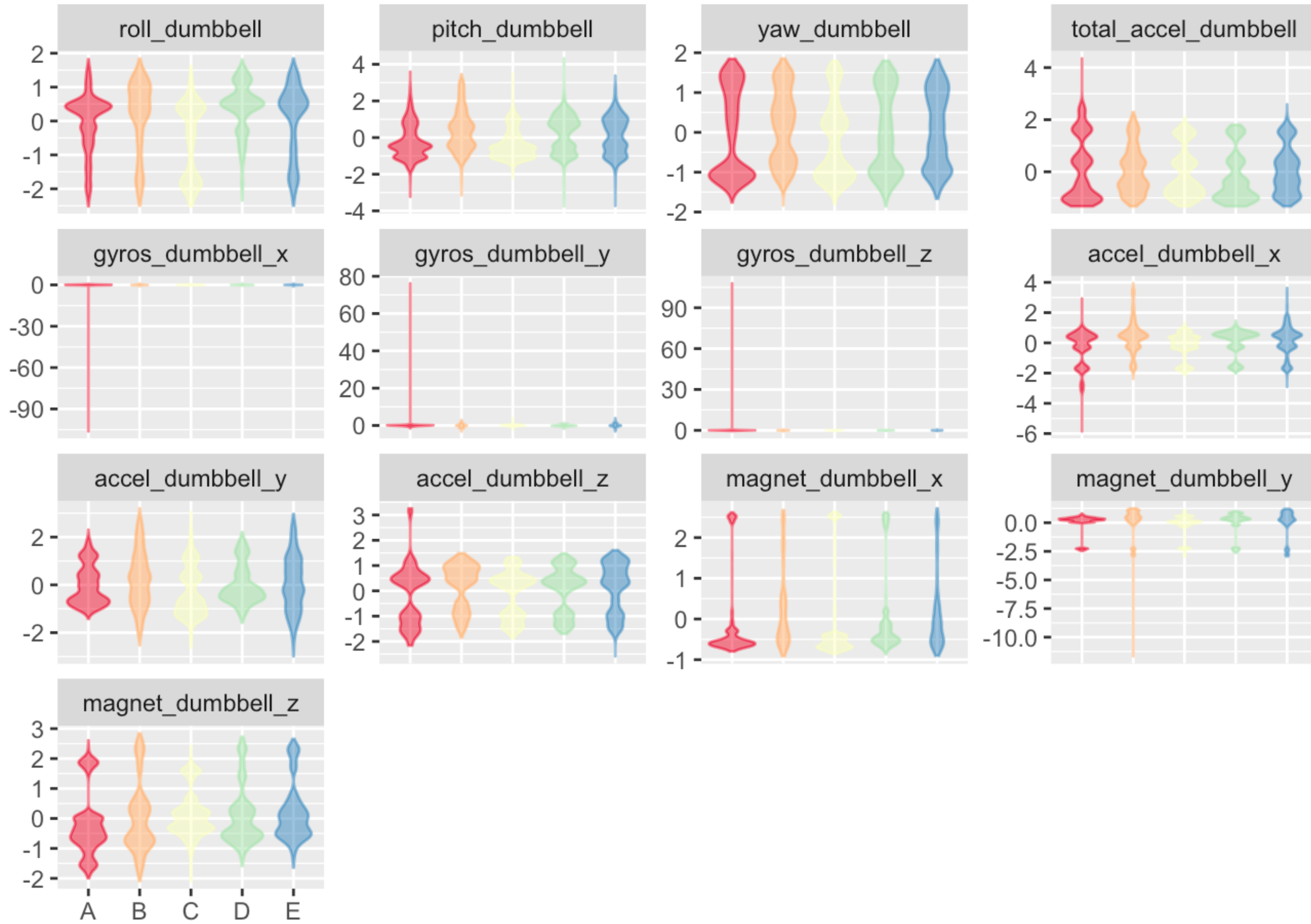
```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```



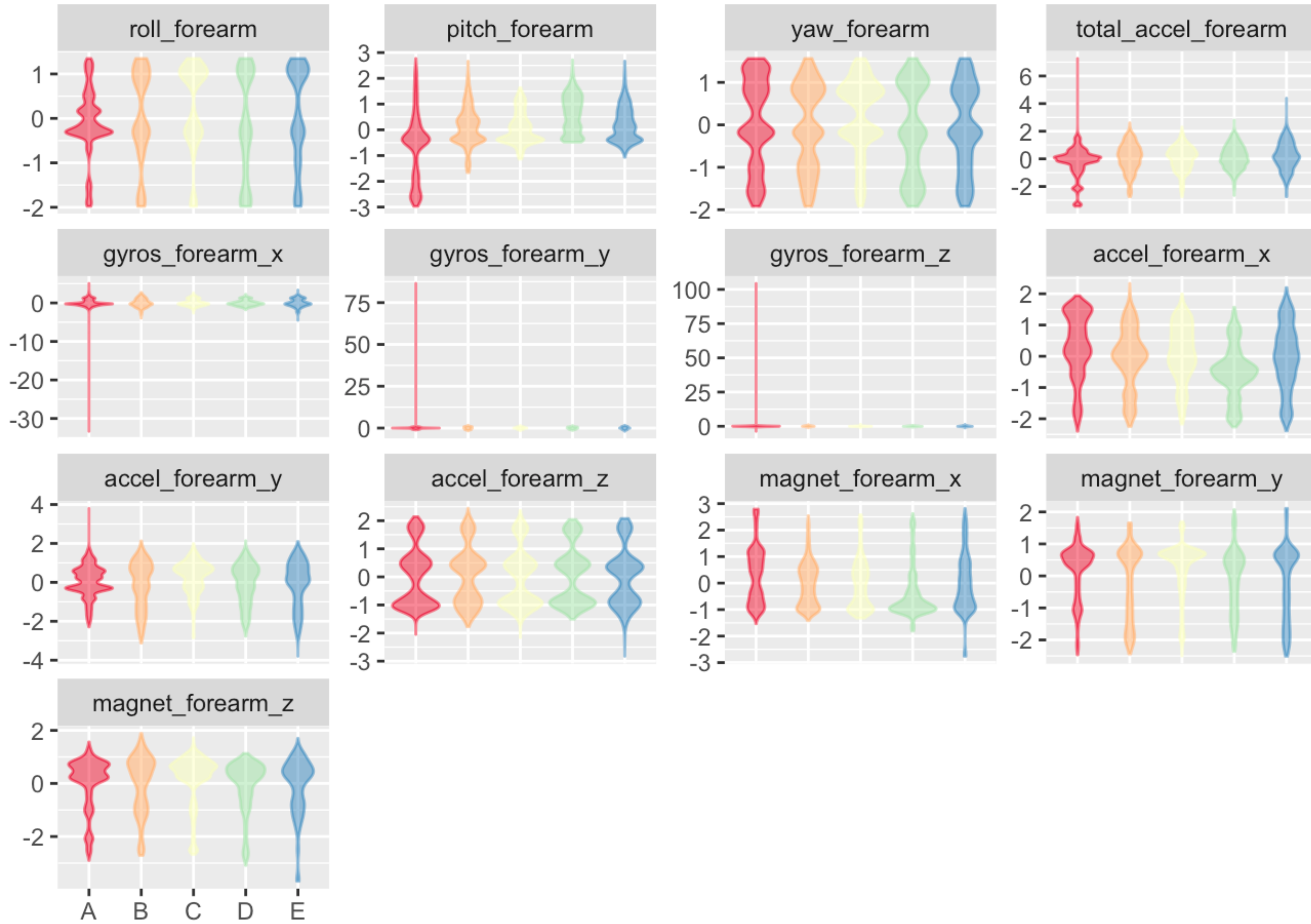

```
histGroup(DTrainCS, "[^(fore)]arm")
```




```
histGroup(DTrainCS, "dumbbell")
```




```
histGroup(DTrainCS, "forearm")
```



Training a Prediction Model

Using a random forest, the out-of-sample error should be small. We'll estimate the error using 40% probing sample.

Set up the parallel clusters.

```
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
```

Set the control parameters.

```
ctrl <- trainControl(classProbs=TRUE,
                     savePredictions=TRUE,
                     allowParallel=TRUE)
```

And fit our model over the training parameters. Note: this takes a while.

```
method <- "rf"
system.time(trainingModel <- train(classe ~ ., data=DTrainCS, method=method))
```

```
##     user   system  elapsed
##   34.518    1.085 1934.071
```

```
stopCluster(cl)
```

Evaluate the Training Model

```
trainingModel
```

```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776, 11776, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9861383  0.9824602  0.002500503  0.003172630
##   27    0.9857348  0.9819517  0.002154335  0.002727942
##   52    0.9771254  0.9710582  0.005047104  0.006397850
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
hat <- predict(trainingModel, DTrainCS)
confusionMatrix(hat, DTrain[, classe])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Evaluate the Model Using the Probing Dataset

```
hat <- predict(trainingModel, DProbeCS)
confusionMatrix(hat, DProbeCS[, classe])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231   11    0    0    0
##          B    1 1504   11    0    0
##          C    0    3 1353   31    3
##          D    0    0    4 1254    2
##          E    0    0    0    1 1437
##
## Overall Statistics
##
##                Accuracy : 0.9915
##                  95% CI : (0.9892, 0.9934)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9892
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9908   0.9890   0.9751   0.9965
## Specificity            0.9980   0.9981   0.9943   0.9991   0.9998
## Pos Pred Value         0.9951   0.9921   0.9734   0.9952   0.9993
## Neg Pred Value         0.9998   0.9978   0.9977   0.9951   0.9992
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1917   0.1724   0.1598   0.1832
## Detection Prevalence   0.2858   0.1932   0.1772   0.1606   0.1833
## Balanced Accuracy      0.9988   0.9944   0.9917   0.9871   0.9982
```

Final Model

```
varImp(trainingModel)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                     Overall
## roll_belt          100.00
## yaw_belt            78.03
## magnet_dumbbell_z   63.91
## magnet_dumbbell_y   62.02
## pitch_forearm       61.83
## pitch_belt          59.26
## magnet_dumbbell_x   51.14
## roll_forearm        50.20
## magnet_belt_z       43.52
## roll_dumbbell       42.57
## accel_dumbbell_y    42.52
## magnet_belt_y       42.42
## accel_belt_z        41.28
## accel_dumbbell_z    37.17
## roll_arm            34.57
## accel_forearm_x     30.70
## accel_dumbbell_x    29.08
## yaw_dumbbell        28.79
## gyros_dumbbell_y    27.33
## magnet_forearm_z    27.25
```

```
trainingModel$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.85%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3346    2    0    0    0 0.0005973716
## B   19 2255    5    0    0 0.0105309346
## C    1   23 2027    3    0 0.0131450828
## D    0    0   35 1893    2 0.0191709845
## E    0    0    5    5 2155 0.0046189376
```

We have an estimated error rate of less than 1%. Excellent. We'll save this training model for later.

```
save(trainingModel, file="trainingModel.RData")
```

Predictions

Load the training model.

```
load(file="trainingModel.RData", verbose=TRUE)
```

```
## Loading objects:
##   trainingModel
```

Predict and evaluate.

```
DTestCS <- predict(preProc, testing[, predCandidates, with=FALSE])
hat <- predict(trainingModel, DTestCS)
testing <- cbind(hat , testing)
subset(testing, select=names(testing)[grep("belt|[^(fore)]arm|dumbbell|forearm", n
ames(testing), invert=TRUE)])
```

```
##     hat V1 user_name raw_timestamp_part_1 raw_timestamp_part_2
##  1:   B  1     pedro           1323095002               868349
##  2:   A  2    jeremy           1322673067               778725
##  3:   B  3    jeremy           1322673075               342967
##  4:   A  4    adelmo           1322832789               560311
##  5:   A  5    eurico           1322489635               814776
##  6:   E  6    jeremy           1322673149               510661
##  7:   D  7    jeremy           1322673128               766645
##  8:   B  8    jeremy           1322673076                54671
##  9:   A  9  carlitos           1323084240               916313
## 10:   A 10   charles           1322837822               384285
## 11:   B 11  carlitos           1323084277                36553
## 12:   C 12    jeremy           1322673101               442731
## 13:   B 13    eurico           1322489661               298656
## 14:   A 14    jeremy           1322673043               178652
## 15:   E 15    jeremy           1322673156               550750
## 16:   E 16    eurico           1322489713               706637
## 17:   A 17     pedro           1323094971               920315
## 18:   B 18  carlitos           1323084285               176314
## 19:   B 19     pedro           1323094999               828379
## 20:   B 20    eurico           1322489658               106658
##       cvtd_timestamp new_window num_window problem_id
##  1: 05/12/2011 14:23         no         74          1
##  2: 30/11/2011 17:11         no        431          2
##  3: 30/11/2011 17:11         no        439          3
##  4: 02/12/2011 13:33         no        194          4
##  5: 28/11/2011 14:13         no        235          5
##  6: 30/11/2011 17:12         no        504          6
##  7: 30/11/2011 17:12         no        485          7
##  8: 30/11/2011 17:11         no        440          8
##  9: 05/12/2011 11:24         no        323          9
## 10: 02/12/2011 14:57         no        664         10
## 11: 05/12/2011 11:24         no        859         11
## 12: 30/11/2011 17:11         no        461         12
## 13: 28/11/2011 14:14         no        257         13
## 14: 30/11/2011 17:10         no        408         14
## 15: 30/11/2011 17:12         no        779         15
## 16: 28/11/2011 14:15         no        302         16
## 17: 05/12/2011 14:22         no         48         17
## 18: 05/12/2011 11:24         no        361         18
## 19: 05/12/2011 14:23         no         72         19
## 20: 28/11/2011 14:14         no        255         20
```