Mikhail Lara - Practical Machine Learning Course Project

Loading Data

The raw training data and test data were loaded into the R workspace. However inspection of the top 6 rows of the training data revealed that a significant portion of the columns were blank or 'NA'. A trimmed version of the training data named "clean _ training"" that only included the 'classe' value and the accelerometer data was created. Trimming the data this way resulted in the number of possible predictors being reduced to 52 from the original 159. The same trimming operation was performed to make a corresponding 'clean _ testing' data set.

```
library(ggplot2)
library(lattice)
library(GGally)
library(survival)
library(splines)
library(parallel)
library(nlme)
library(caret)
library(e1071)
library(pgmm)
library(rpart)
library(gbm)
library(plyr)
library(MASS)
library(randomForest)
library(klaR)
library(mgcv)
library(nlme)

#Getting Data
  setwd('/Users/Mikey/Documents/ML-Case-Studies/Activity Accelerometer Study')
  list.files()
  training<-read.csv('pml-training.csv',header = TRUE)
  testing<-read.csv('pml-testing.csv',header = TRUE)

  #head(training)

  good_fields<-c('roll_belt','pitch_belt','yaw_belt','total_accel_belt',
                 'gyros_belt_x','gyros_belt_y','gyros_belt_z',
                 'accel_belt_x','accel_belt_y','accel_belt_z',
                 'magnet_belt_x','magnet_belt_y','magnet_belt_z',

                 'roll_arm','pitch_arm','yaw_arm',  'total_accel_arm',
                 'gyros_arm_x','gyros_arm_y','gyros_arm_z',
                 'accel_arm_x','accel_arm_y','accel_arm_z',
                 'magnet_arm_x','magnet_arm_y','magnet_arm_z',

                 'roll_dumbbell','pitch_dumbbell','yaw_dumbbell','total_accel_dumbbell',
                 'gyros_dumbbell_x' ,'gyros_dumbbell_y','gyros_dumbbell_z',
```

```
                'accel_dumbbell_x' ,'accel_dumbbell_y' ,'accel_dumbbell_z' ,
                'magnet_dumbbell_x'    ,'magnet_dumbbell_y'    ,'magnet_dumbbell_z'    ,

                'roll_forearm','pitch_forearm' ,'yaw_forearm','total_accel_forearm',
                'gyros_forearm_x'  ,'gyros_forearm_y', 'gyros_forearm_z',
                'accel_forearm_x'  ,'accel_forearm_y'  ,'accel_forearm_z',
                'magnet_forearm_x' ,'magnet_forearm_y' ,'magnet_forearm_z',

                'classe')

indx<-vector(mode='numeric')
for(i in c(1:length(names(training)))){
              indx<-c(indx,which(names(training)==good_fields[i]))
}

clean_training<-training[,indx]
clean_testing<- testing[,indx]
```

Exploratory Analysis & Cleaning Data

A simple exploratory analysis was performed to identify any outliers in the predictors. This is crucial because extreme outliers can have a deleterious effect on the performance of the iterative machine learning methods, such as random forest and boosting. The exploratory analysis procedure consisted of looking at scatterplots of each of the predictors and taking note if there were any accelerometer measurements that lay more than 5 standard deviations away from the mean. Two observations, located at rows 5373 & 9274 in 'clean_ train' were classified as outliers using this criterion and were removed from the 'clean_training' data set. The issues with row 5373 was that the gyroscope data for the accelerometers on the dumbbell and the forearm reported values with magnitudes in the hundreds when the remaining data was less than 20. A similar observation was made for row 9274 in the 'magnet_dumbbell_y' predictor. Plots of these observations are included.

Before proceeding to modeling, two more investigatory tests were performed to identify both significant correlation between predictors and predictors with statistically insignificant variance. To evaluate high correlation, correlation matrices were calculated for the data from each accelerometer. The results did not indicate any meaningful correlations that would merit eliminating any predictors. Statistically significant variance was calculated using the 'nearZeroVar' function on the entire 'clean _ training' data set. That analysis concluded that no predictors could be justifiably omitted from modeling on the basis of near zero variance.

```
#Exploratory Analysis & Cleaning Data

  ##Look for Fields with Outliers {Random Forest & Tree Base Methods are Sensitive to Outliers}
  summary(clean_training)

  #belt data
    for(i in c(1:13)){
      plot(clean_training[,i],main = names(clean_training)[i])
    }
```

```
  #arm data
    for(i in c(1:13)){
      plot(clean_training[,i+13],main = names(clean_training)[i+13])
    }
```

```
#dumbbell data
  for(i in c(1:13)){
    plot(clean_training[,i+26],main = names(clean_training)[i+26])
  }
```

```
#forearm data
  for(i in c(1:13)){
    plot(clean_training[,i+39],main = names(clean_training)[i+39])
  }
```

```
#Sample 5373 had extreme outliers for all dumbbell & forearm gyroscope data
#Sample 9274 had an extreme outlier for ***
#>>>>Single Composite Plot for 5373 <PLOT1>
#>>>>Single Plot for 9274 <PLOT2>

cor(clean_training[,1:13])
cor(clean_training[,14:26])
cor(clean_training[,27:39])
cor(clean_training[,39:52])
#No Strong Correlation that Merited Removal of Predictor

clean_training<-clean_training[-c(5373,9274),]

#length(nearZeroVar(clean_train_dat))
#No Near Zero Variance Predictors in Cleaned Data
```

- Modeling Data & Cross-Validation

Training & validation datasets were created from the trimmed 'clean_training' data set using the standard createDataPartition function in caret. The training set consisted of 75% of the observations in the 'clean_training' data using the 'classe' data as the outputs. The validation set was created for model cross-validation to provide an estimate of out-of-sample accuracy.

Modeling the data was performed by successfully fitting 4 different prediction methods to the training data 'clean_train_dat'. The specific methods used were 1) recursive partitioning/regression tree, 2)linear discriminant analysis, 3)support vector machine, and 4) random forest. Boosting & Naive Bayes methods were also attempted but were abandoned because the analysis took too long, making it impractical for optimization/tuning. It is worth noting that the random forest 'mtry' value had to be set to 10 to make it produce a model in a reasonable amount of time.

```
#Modeling - Create Training/Validation Data Sets & Test Different Prediction Models

  inTrain = createDataPartition(clean_training$classe,p=3/4,list=FALSE)
  clean_train_dat = clean_training[inTrain,]
  validate = clean_training[-inTrain,]

  # Single Tree - Low Accuracy
  set.seed(12345)
  tree1<-train(classe~.,data = clean_train_dat, method='rpart')
  tree_pred_train<-predict(tree1,clean_train_dat)
  tree_pred_validate<-predict(tree1,validate)
  confusionMatrix(tree_pred_train,clean_train_dat$classe)
  confusionMatrix(tree_pred_validate,validate$classe)
  #Test Set Accuracy:49.21
  #Validation Set Accuracy:50.6
```

```
#Linear Discriminant Analysis - Mediocre Accuracy
set.seed(12345)
lda<- train(classe~.,data = clean_train_dat, method='lda')
plda_train<-predict(lda,clean_train_dat)
plda_validate<-predict(lda,validate)
confusionMatrix(plda_train,clean_train_dat$classe)
confusionMatrix(plda_validate,validate$classe)
#Test Set Accuracy:70.92
#Validation Set Accuracy:69.9

#Support Vector Machine - Very Accurate
set.seed(12345)
svmFit <- svm(classe~ ., data = clean_train_dat)
svmPred_train <- predict(svmFit,clean_train_dat)
svmPred_validate<-predict(svmFit,validate)
confusionMatrix(svmPred_train,clean_train_dat$classe)
confusionMatrix(svmPred_validate,validate$classe)
#Test Set Accuracy:95.86
#Validation Set Accuracy:95.25

#Random Forest - Extremely Accurate
set.seed(12345)
forest<- randomForest(classe~.,data = clean_train_dat, mtry = 10)
forest_pred_train<-predict(forest,clean_train_dat)
forest_pred_validate<-predict(forest,validate)
confusionMatrix(forest_pred_train,clean_train_dat$classe)
confusionMatrix(forest_pred_validate,validate$classe)
#In-Sample Accuracy:100%
#Validation Set Accuracy:99.56%
```

- Final Model Selection

The accuracies for each of the 4 resulting models were evaluated by simply predicting on both the 'clean_train_dat' and 'validate' data sets and looking at the resulting confusion matrix. This procedure resulted in one in-sample accuracy measurement and one estimate of out-of-sample accuracy. These pairs of accuracy values were virtually identical for each model. The top two models of these 4 were the random forest and support vector machine models. The in-sample accuracy and estimated out-of-sample accuracy for the random forest model were 100% & 99.56% respectively. The in-sample accuracy and estimated out-of-sample accuracy for the support vector machine model were 95.86% & 95.25% respectively.

A comparison of the training set predictions of the top 2 models demonstrates that the values they agree upon are 100% accurate. With regards to the validation set, the agreed predictions of the top 2 models are 99.6% accurate. That is the values they mutually predict fail to accurately predict 1 out of 4681 observations. This degree of accuracy suggests that although the random forest method has a higher in-sample accuracy, the 2 models have nearly identical predictive power.

When used to predict on the testing set, the two models agreed on all but 1 out of 20 observations. Ultimately, the final model chosen was the random forest model due to the slightly higher in-sample accuracy.

```
#Final Model Selection

#Check Accuracy of Agreed Forest & SVM Points
agree_train<-forest_pred_train[c(forest_pred_train==svmPred_train)]
agree_validate<-forest_pred_validate[c(forest_pred_validate==svmPred_validate)]
confusionMatrix(agree_train,clean_train_dat$classe[c(forest_pred_train==svmPred_train)]) #In-Sample A
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4179    0    0    0    0
##          B    0 2633    0    0    0
##          C    0    0 2508    0    0
##          D    0    0    0 2207    0
##          E    0    0    0    0 2614
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2955
##     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             0.2955   0.1862   0.1774   0.1561   0.1849
## Detection Rate         0.2955   0.1862   0.1774   0.1561   0.1849
## Detection Prevalence   0.2955   0.1862   0.1774   0.1561   0.1849
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```
confusionMatrix(agree_validate,validate$classe[c(forest_pred_validate==svmPred_validate)]) #Est. Out
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1388    4    0    0    0
##          B    0  860    4    0    0
##          C    0    5  828    4    2
##          D    0    0    0  728    3
##          E    0    0    0    0  866
##
## Overall Statistics
##
##                Accuracy : 0.9953
##                  95% CI : (0.9929, 0.9971)
##     No Information Rate : 0.2958
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.994
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9896   0.9952   0.9945   0.9943
## Specificity            0.9988   0.9990   0.9972   0.9992   1.0000
## Pos Pred Value         0.9971   0.9954   0.9869   0.9959   1.0000
## Neg Pred Value         1.0000   0.9976   0.9990   0.9990   0.9987
## Prevalence             0.2958   0.1852   0.1773   0.1560   0.1856
## Detection Rate         0.2958   0.1833   0.1765   0.1552   0.1846
## Detection Prevalence   0.2967   0.1841   0.1788   0.1558   0.1846
## Balanced Accuracy      0.9994   0.9943   0.9962   0.9969   0.9971
```

```
  #Results that Random forest & Supported Vector Machine Methods Agree On Are 100% Accurate

#Predict on Test Data
forest_test<-predict(forest,clean_testing)
svm_test<-predict(svmFit,clean_testing)

print(forest_test)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

```
print(svm_test)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  A  A  B  A  A  E  D  B  A  A  A  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

```
confusionMatrix(svm_test,forest_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction A B C D E
##          A 7 2 0 0 0
##          B 0 6 0 0 0
##          C 0 0 1 0 0
##          D 0 0 0 1 0
##          E 0 0 0 0 3
##
## Overall Statistics
##
##                Accuracy : 0.9
##                  95% CI : (0.683, 0.9877)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : 5.041e-06
##
##                   Kappa : 0.8561
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.7500     1.00     1.00     1.00
## Specificity            0.8462   1.0000     1.00     1.00     1.00
```

```
## Pos Pred Value         0.7778   1.0000     1.00     1.00     1.00
## Neg Pred Value         1.0000   0.8571     1.00     1.00     1.00
## Prevalence             0.3500   0.4000     0.05     0.05     0.15
## Detection Rate         0.3500   0.3000     0.05     0.05     0.15
## Detection Prevalence   0.4500   0.3000     0.05     0.05     0.15
## Balanced Accuracy      0.9231   0.8750     1.00     1.00     1.00
```

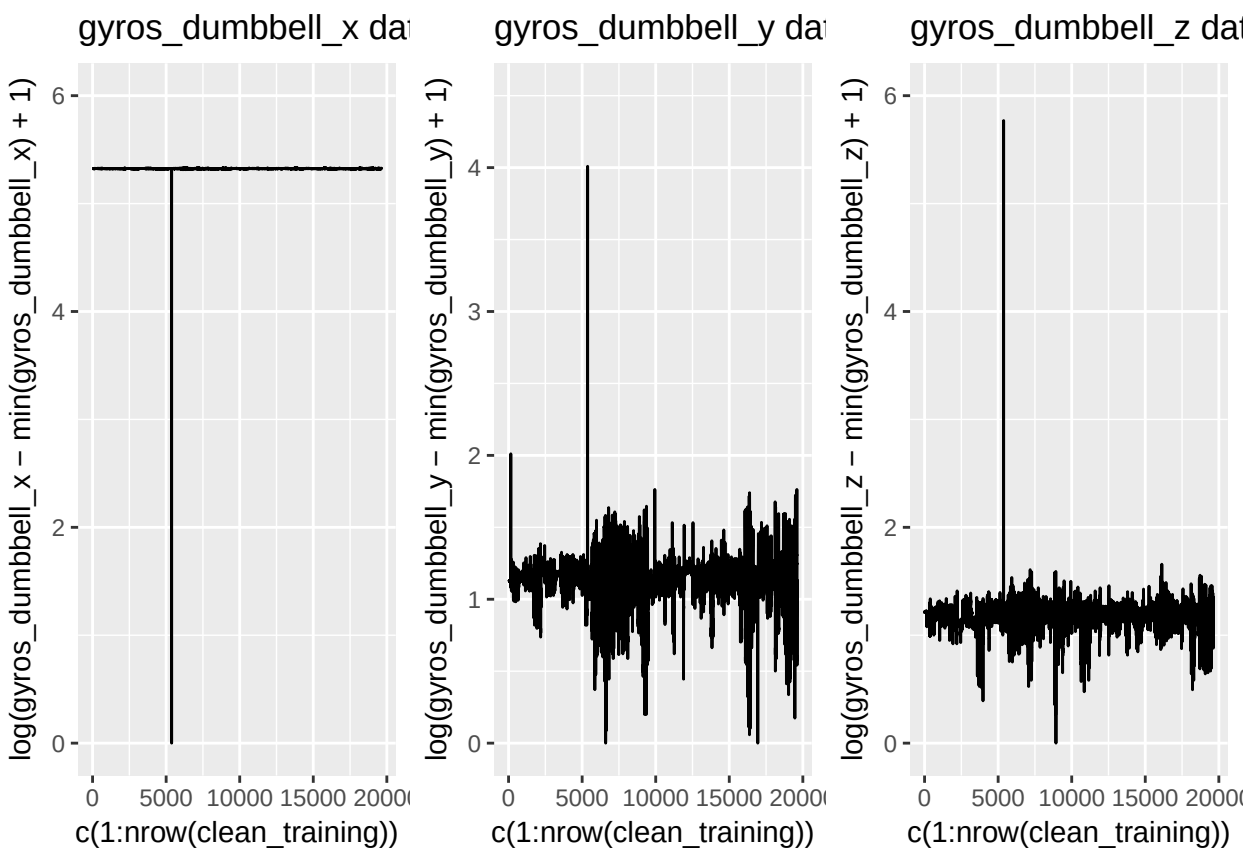
```
#They agree for all but 1 sample out of 20 {Both are Very Good, Random Forest is Slightly Better}
```

PLOTS

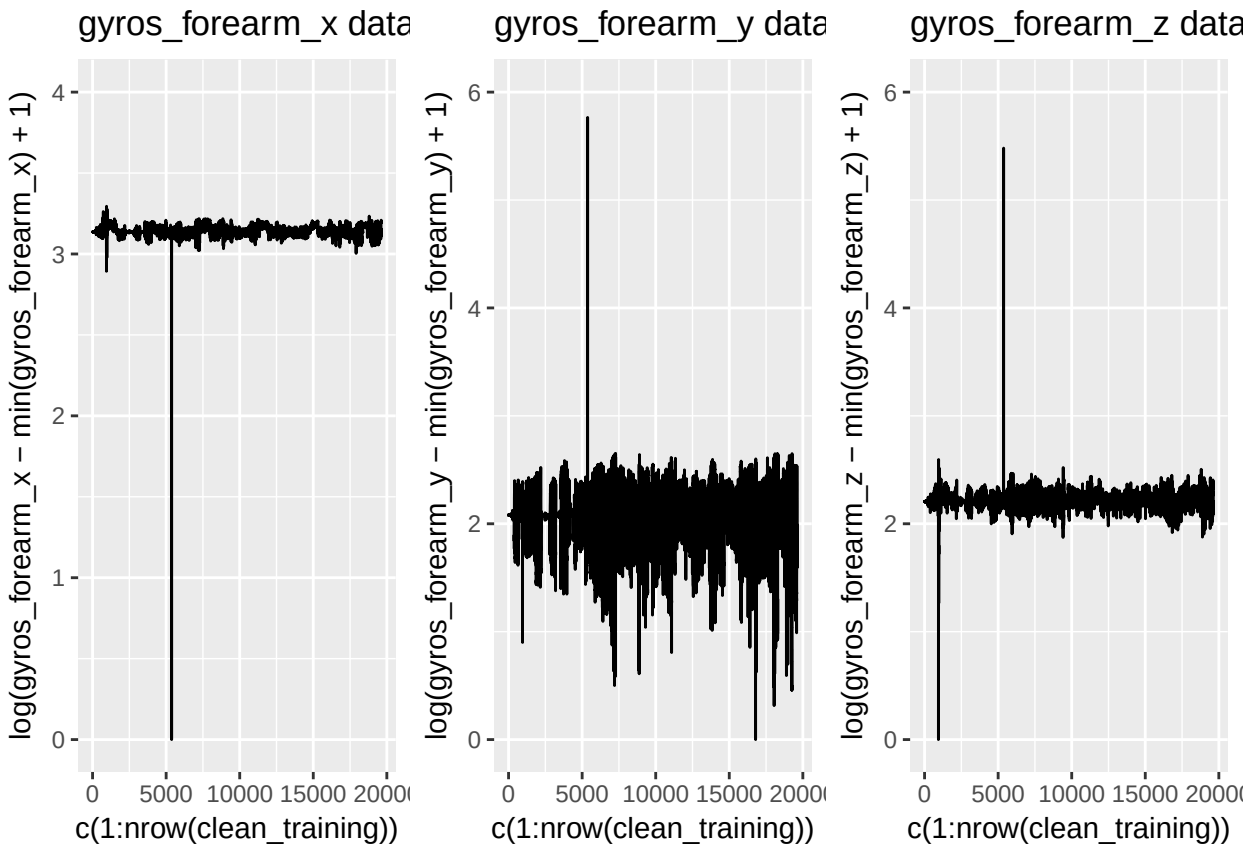
1) Dumbbell Outliers corresponding to row 5373
2) Forearm Outliers corresponding to row 5373
3) Outier corresponding to row 9274

```
clean_training<-training[,indx]

 multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
    library(grid)

    # Make a list from the ... arguments and plotlist
    plots <- c(list(...), plotlist)

    numPlots = length(plots)

    # If layout is NULL, then use 'cols' to determine layout
    if (is.null(layout)) {
      # Make the panel
      # ncol: Number of columns of plots
      # nrow: Number of rows needed, calculated from # of cols
      layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                       ncol = cols, nrow = ceiling(numPlots/cols))
    }

   if (numPlots==1) {
      print(plots[[1]])

    } else {
      # Set up the page
      grid.newpage()
      pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

      # Make each plot, in the correct location
      for (i in 1:numPlots) {
        # Get the i,j matrix positions of the regions that contain this subplot
        matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

        print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                        layout.pos.col = matchidx$col))
      }
    }
  }

  #Log Transforms of Row 5373 Outliers
```

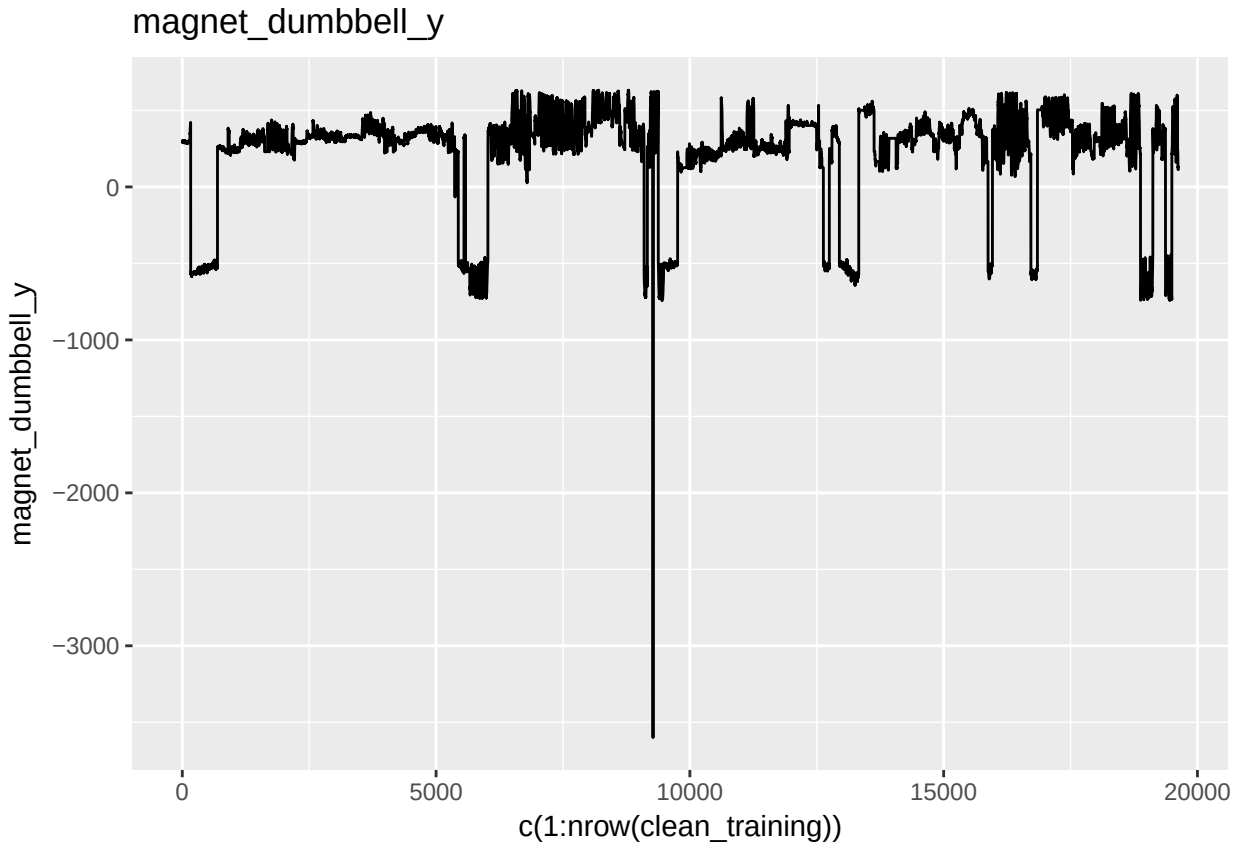
```
g1<-qplot(data = clean_training,x=c(1:nrow(clean_training)),y=log(gyros_dumbbell_x-min(gyros_dumbbell
g2<-qplot(data = clean_training,x=c(1:nrow(clean_training)),y=log(gyros_dumbbell_y-min(gyros_dumbbell
g3<-qplot(data = clean_training,x=c(1:nrow(clean_training)),y=log(gyros_dumbbell_z-min(gyros_dumbbell

g4<-qplot(data = clean_training,x=c(1:nrow(clean_training)),y=log(gyros_forearm_x-min(gyros_forearm_x
g5<-qplot(data = clean_training,x=c(1:nrow(clean_training)),y=log(gyros_forearm_y-min(gyros_forearm_y
g6<-qplot(data = clean_training,x=c(1:nrow(clean_training)),y=log(gyros_forearm_z-min(gyros_forearm_z

multiplot(g1,g2,g3,cols=3)
```




```
multiplot(g4,g5,g6,cols=3)
```




```
#Row 9274 Outlier
g7<-qplot(data = clean_training,x=c(1:nrow(clean_training)),y=magnet_dumbbell_y,geom='line')+ggtitle(
g7
```


magnet_dumbbell_y
0
−1000
−2000
−3000
magnet_dumbbell_y
0
5000
10000
15000
20000
c(1:nrow(clean_training))