Practical Machine Learning - Course Project

Anastasios Vlaikidis

3/7/2021

Project details

By 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 types of devices are part of the quantified self movement – a group of enthusiasts who regularly take measurements about themselves to improve their health, find patterns in their behavior, or because they are tech geeks. People regularly 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 acquired from accelerometers placed on the belt, forearm, arm, and dumbbells of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

More information is available from this website:

```
http://groupware.les.inf.puc-rio.br/har
```

(see the section on the Weight Lifting Exercise Dataset).

The training data for this project are available here:

```
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
```

The test data are available here:

```
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.
```

If you use the document you create for this class for any purpose, please cite the corresponding references as this data has been generously provided for use for this kind of assignment. The goal of your project is to predict how they did the exercise. This is called the "classe" variable in the training set.

Import and prepare the Data

```r
# Workspace clean up
rm(list = ls())
```

```
DataT <- read.csv("pml-training.csv")
DataTesting <-read.csv("pml-testing.csv")
```

We see that we have some strange strings and we will replace them with NA.

```
# Replace with NA
DataT <- read.csv("pml-training.csv",na.strings =c("#DIV/0!","NA",""))
DataTesting <- read.csv("pml-testing.csv",na.strings =c("#DIV/0!","NA",""))
```

Removing NA.

```
#Remove columns with more than 95% of NA or "" values
limit<- dim(DataT)[1] * .95
gCols <-!apply(DataT, 2,
          function(x) sum(is.na(x)) > limit||sum(x=="")> limit)
DataT <- DataT[,gCols]

 limit2 <-dim(DataTesting)[1]*.95
 gCols2 <-!apply(DataTesting, 2,
           function(x) sum(is.na(x)) > limit2||sum(x=="")> limit2)
DataTesting<-DataTesting[,gCols2]
```

Dropping unwanted columns.

```
suppressMessages(library(dplyr))
DataT<-dplyr::select(DataT,-c(1,2,5))
DataTesting<-dplyr::select(DataTesting,-c(1,2,5))
```

We will also remove the variables from our training dataset with near-zero variance.

```
suppressMessages(library(caret))
bCols<- nearZeroVar(DataT,saveMetrics = T)
DataT <- DataT[,bCols$nzv==F]
DataTesting <- DataTesting[,-3]
```

Our target variable is called “classe” and its in the last column of our dataset DataT.It is a character variable.We will make it factor so we can easily see its levels.

```
class(DataT$classe)
```

```
## [1] "character"
```

```
DataT[,length(DataT)] <-as.factor(DataT$classe)
class(DataT$classe)
```

```
## [1] "factor"
```

```
levels(DataT$classe)
```

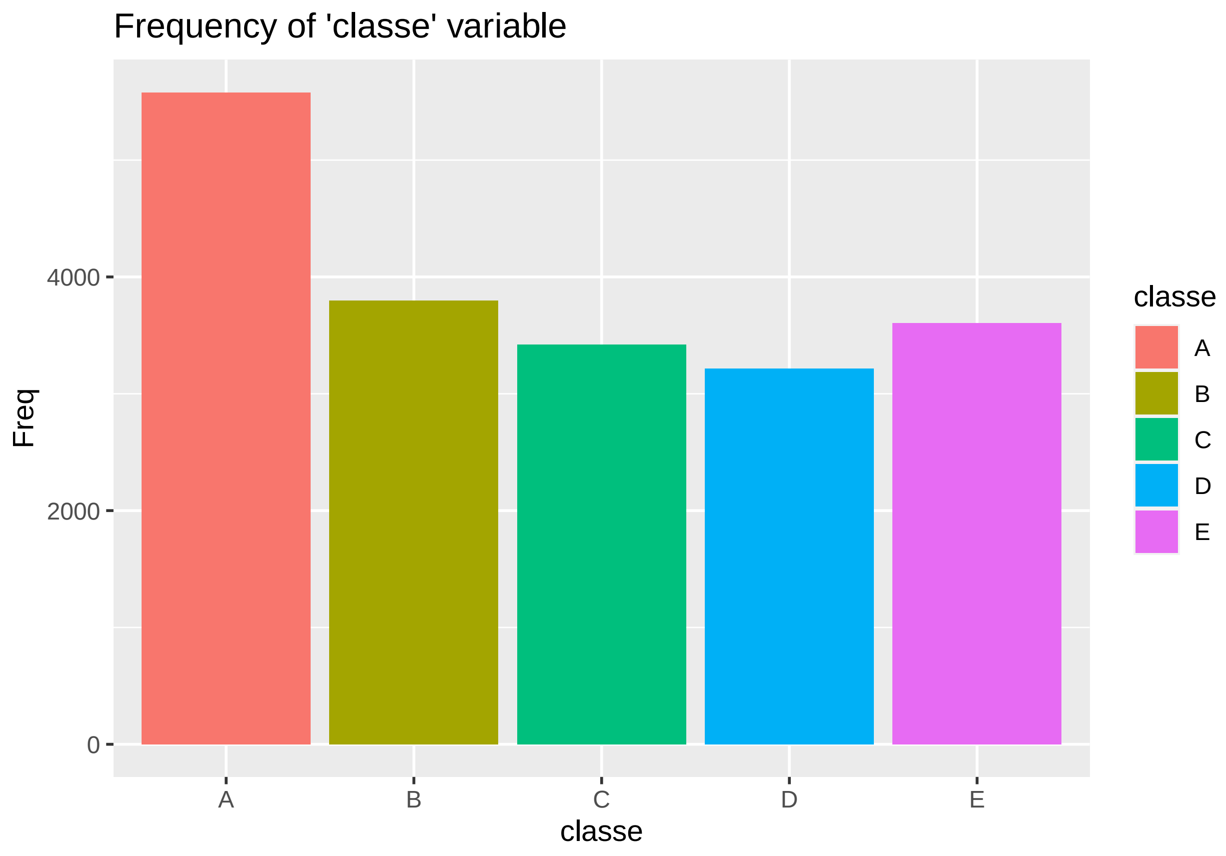
```
## [1] "A" "B" "C" "D" "E"
```

It has five levels: A,B,C,D,E. Since our target variable qualitative, it will be sensible to draw a barplot and a frequency table to represent our findings..

```r
suppressMessages(library(gmodels))
suppressMessages(library(ggplot2))
gmodels::CrossTable(DataT$classe,digits = 2,format = "SPSS")
```

```
##
##    Cell Contents
## |-------------------------|
## |                   Count |
## |             Row Percent |
## |-------------------------|
##
## Total Observations in Table:  19622
##
##           |         A  |         B  |         C  |         D  |         E  |
##           |-----------|-----------|-----------|-----------|-----------|
##           |      5580  |      3797  |      3422  |      3216  |      3607  |
##           |    28.44% |    19.35% |    17.44% |    16.39% |    18.38% |
##           |-----------|-----------|-----------|-----------|-----------|
##
##
```

```r
qplot(classe,data = DataT, geom = "bar", main ="Frequency of 'classe' variable",
      xlab = "classe",ylab ="Freq",fill=classe)
```



From the barplot we can see that, all levels have almost equal frequencies and level A has the higher frequency when compared with the rest.

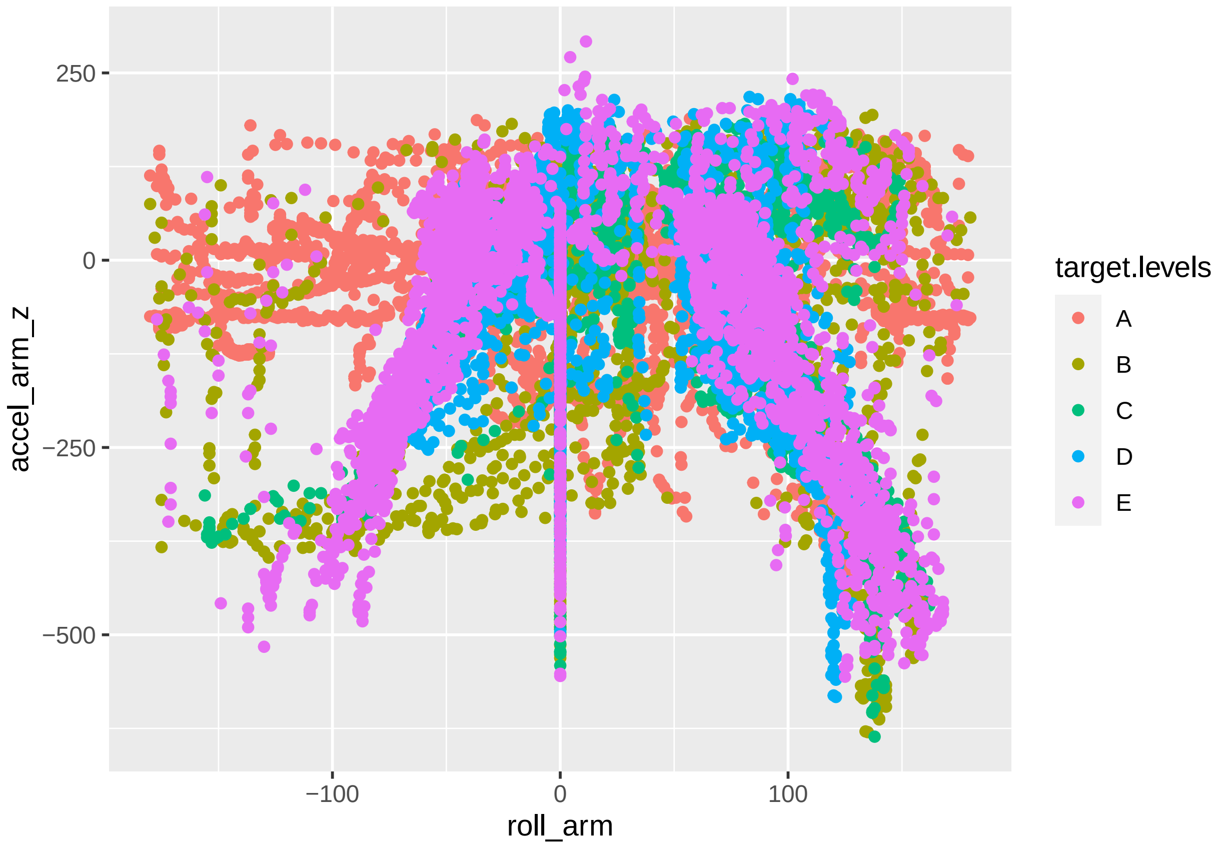
Exploratory analysis

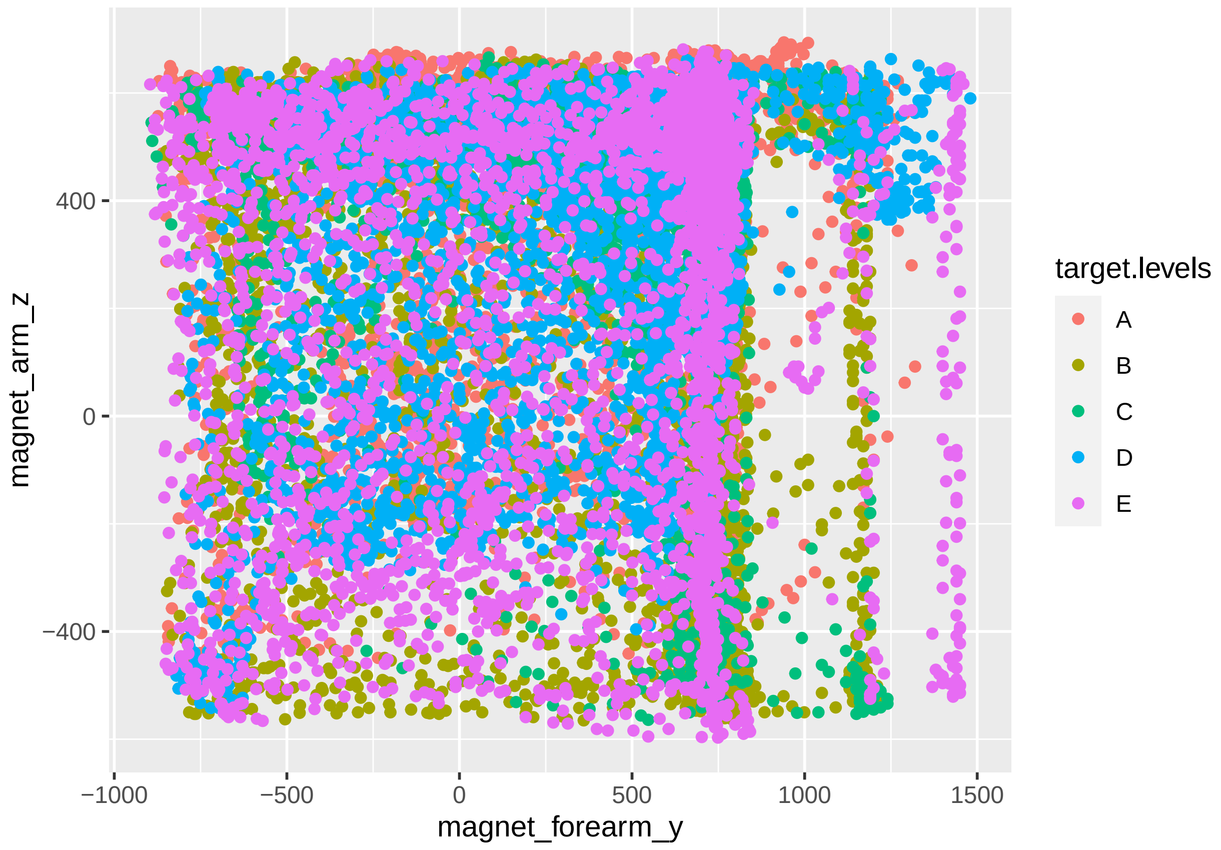
We will perform an exploratory analysis by using several visualization methods.

```r
#for reproducibility
set.seed(22)
#extract numeric columns
NumericVars <- dplyr::select(DataT,where(is.numeric))
```

Two random scaterplots from the NumericVars dataset, colored by out target variable.

```r
target.levels <- DataT$classe
for (i in 1:2){
x <-as.integer(runif(1,1,length(NumericVars)))
y <-as.integer(runif(1,1,length(NumericVars)))
print(qplot(NumericVars[,x],NumericVars[,y],col=target.levels,
      xlab=names(NumericVars)[x],ylab=names(NumericVars)[y]))
}
```

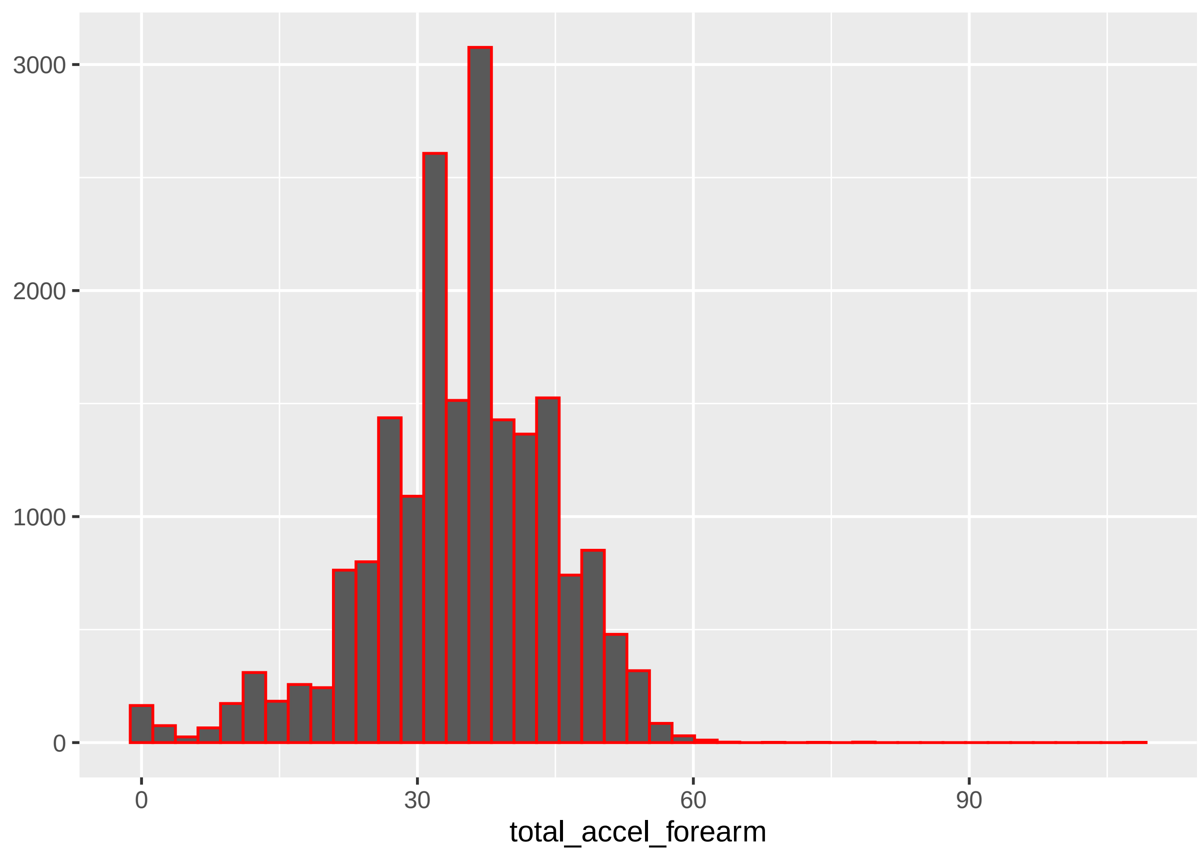


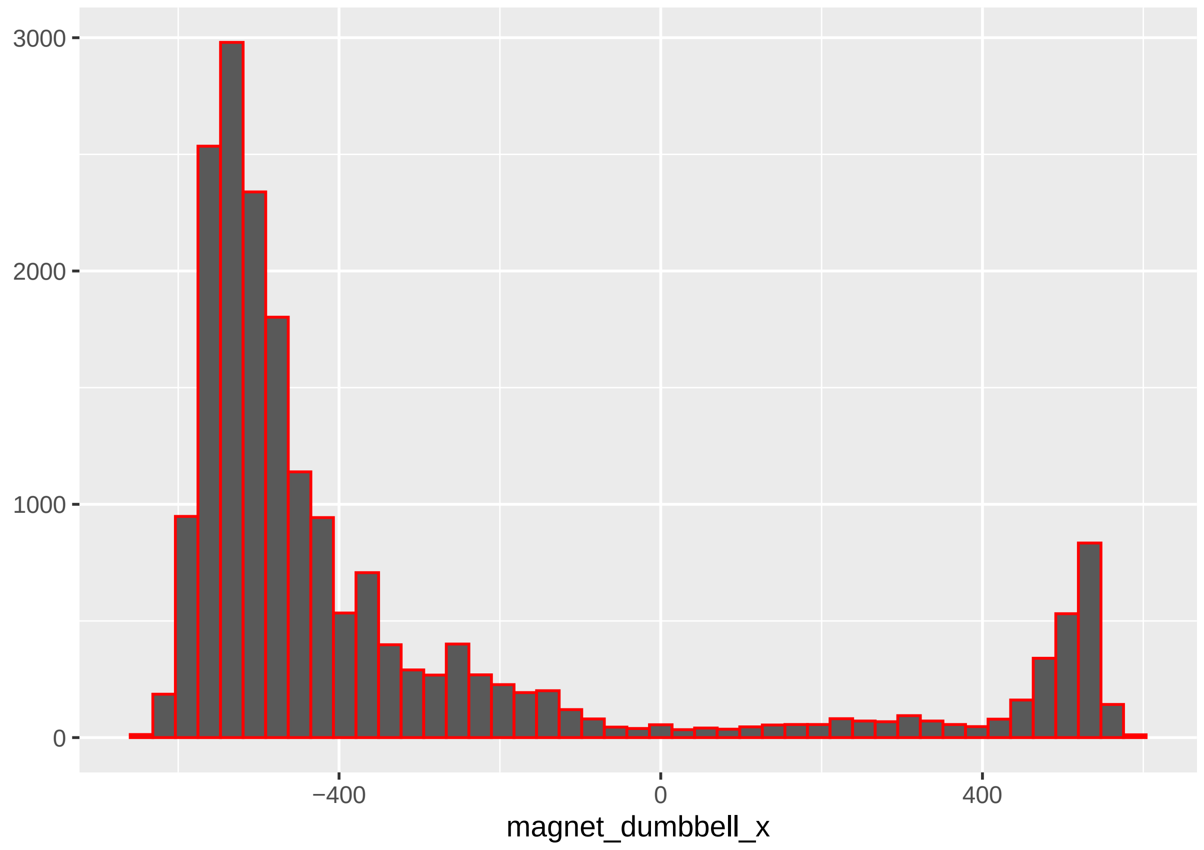


Now we will check the normality of our predictors, because if we have deviations from normality some algorithms maybe tricked. We will make 2 random histograms,density plots, QQ plots and hypothesis tests.We can repeat this process if we want to see more plots and generally better understand the shape of our data.

Histograms

```
Rnum <- as.integer(runif(2,1,length(NumericVars)))
for(i in Rnum) {
 print(qplot(NumericVars[,i], color=I("red"),bins = 45,
xlab = names(NumericVars)[i],geom ="histogram"))
}
```

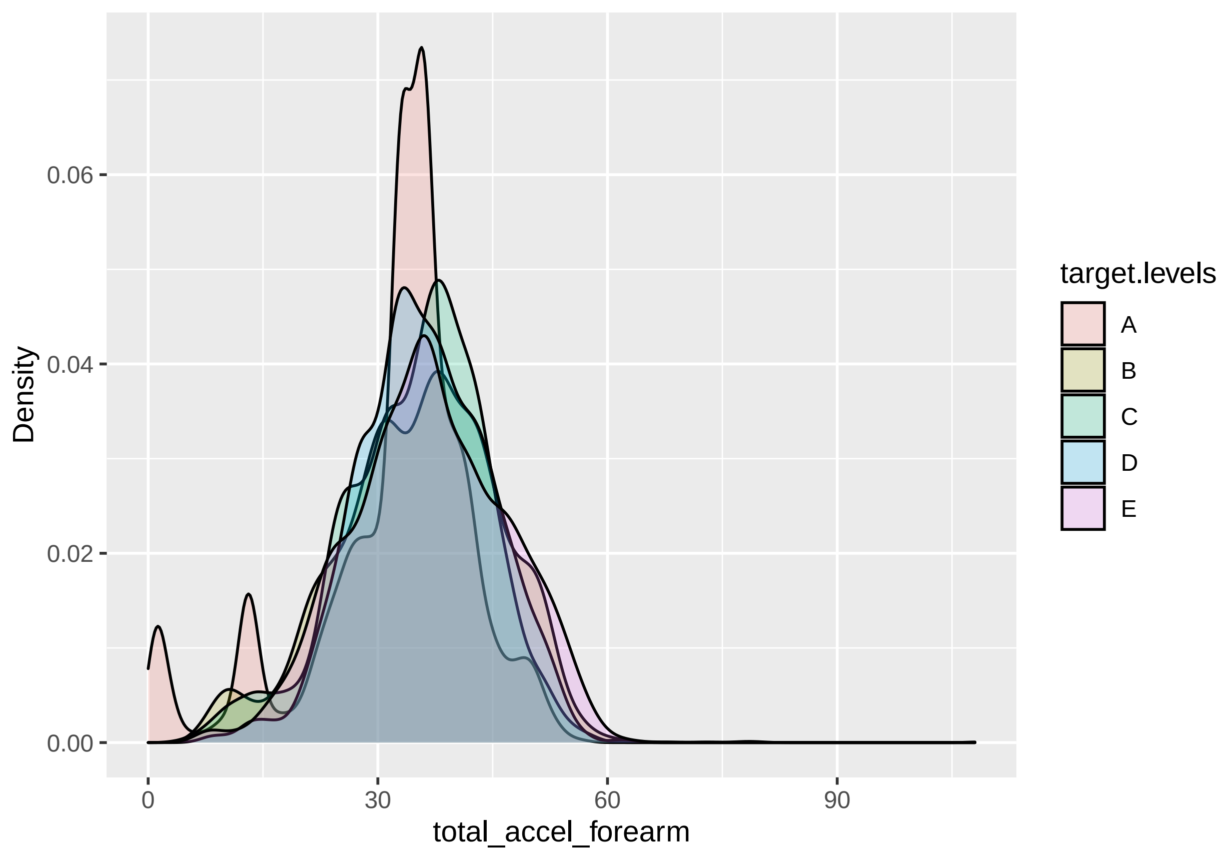

3000
2000
1000
0
0
30
60
90
total_accel_forearm

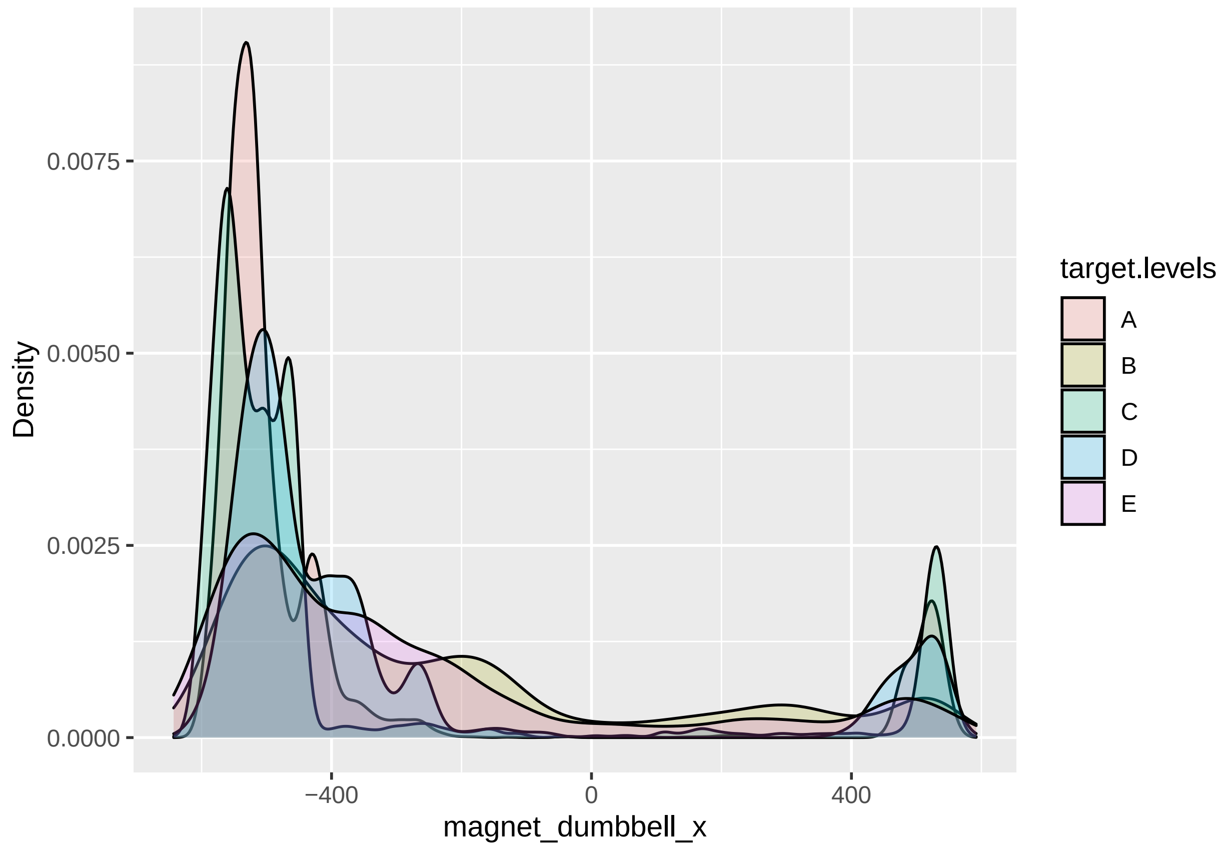


We see that some of our variables are skewed. In a normal distribution, the graph shows symmetry, meaning that there are about as many data values on the left side of the median as on the right side.

Now the density plots.

```r
for(i in Rnum) {
print(qplot(NumericVars[,i], fill=target.levels,alpha=I(.2),
ylab ="Density",xlab = names(NumericVars)[i],geom ="density"))
}
```

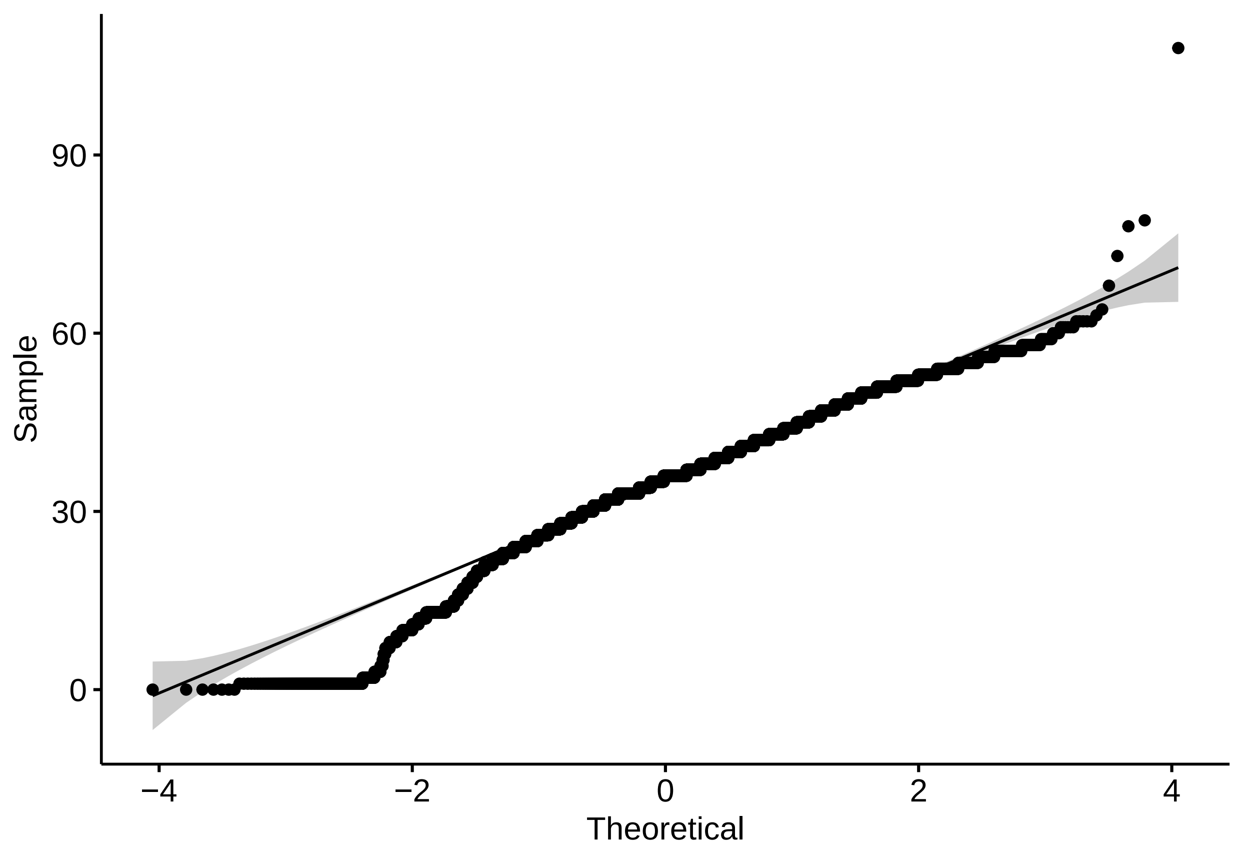

Density
0.00
0.02
0.04
0.06
0
30
60
90
total_accel_forearm
target.levels
A
B
C
D
E

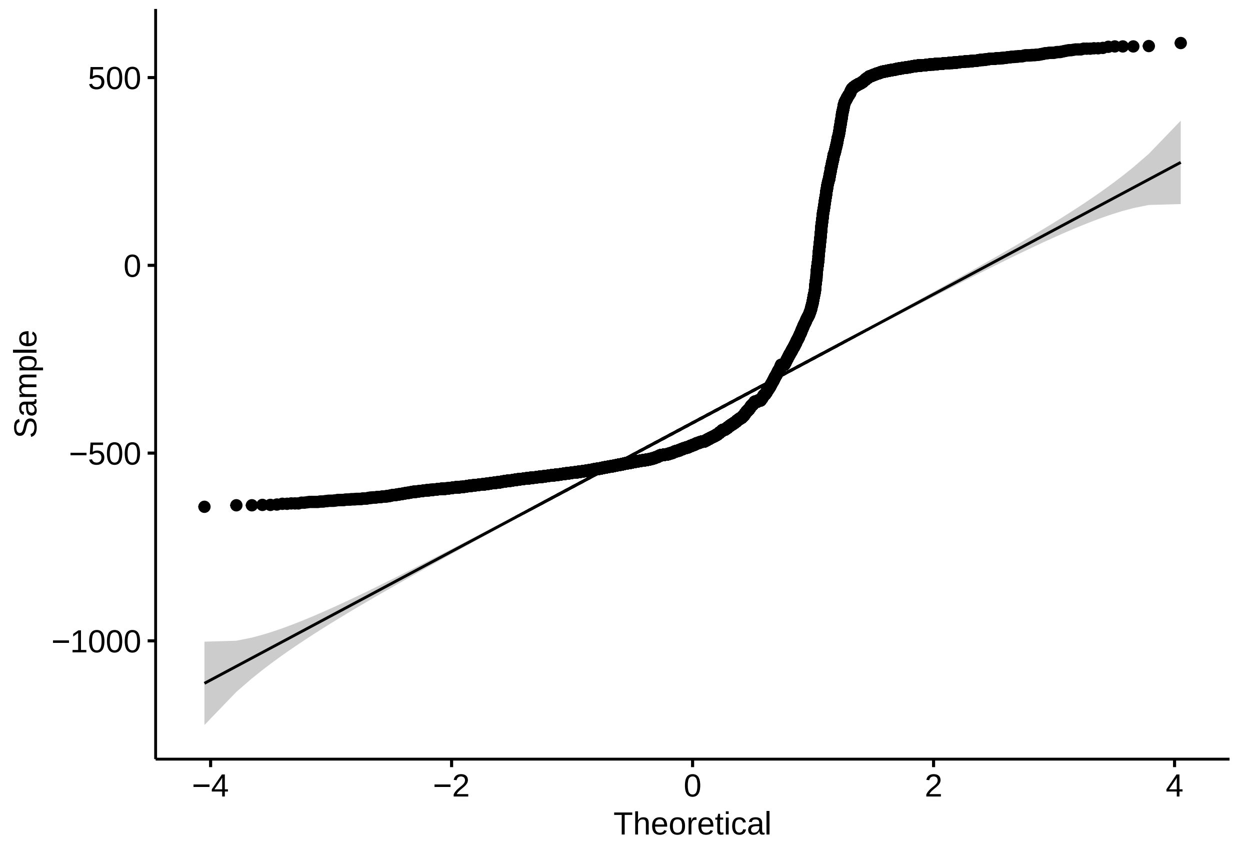


We do not have the characteristic bell curve of a normal distribution.

Quantile-Quantile plots.A 45-degree reference line is also plotted.QQ plots are used to visually check the normality of the data.

```r
suppressMessages(library(ggpubr))
for(i in Rnum) {
  print(ggqqplot(NumericVars[,i]))
}
```


Sample
90
60
30
0
−4
−2
0
2
4
Theoretical



If the data are consistent with a sample from a normal distribution, the points should lie close to the 45-degree reference line. We see that our data are not close to a normal distribution.

Hypothesis tests are also used to check the normality of the data.We will use Kolmogorov-Smirnov tests. The NULL hypothesis is that the data are normally distributed, with the alternative being that they will not be normally distributed.

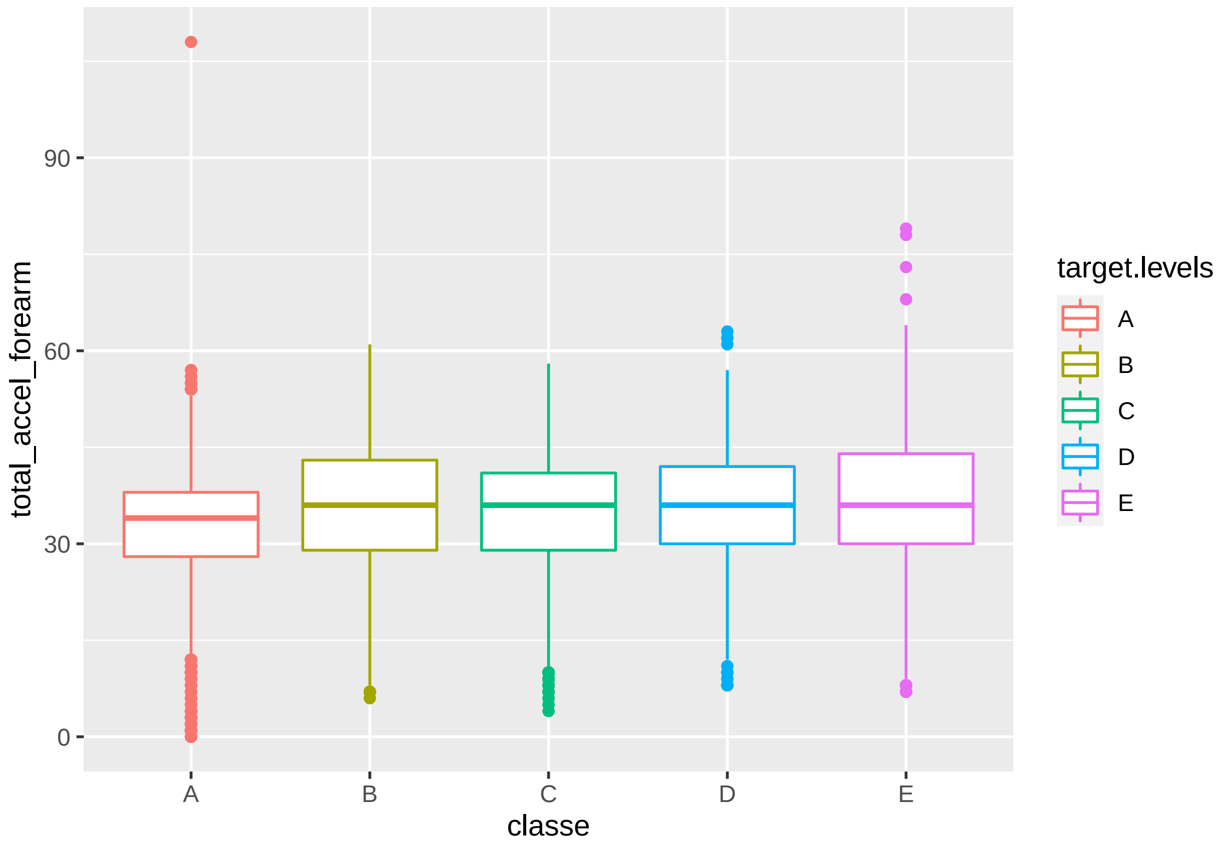
```
suppressMessages(library(nortest))
for ( i in Rnum){
print(lillie.test(NumericVars[,i])) #Kolmogorov-Smirnov test
}
```

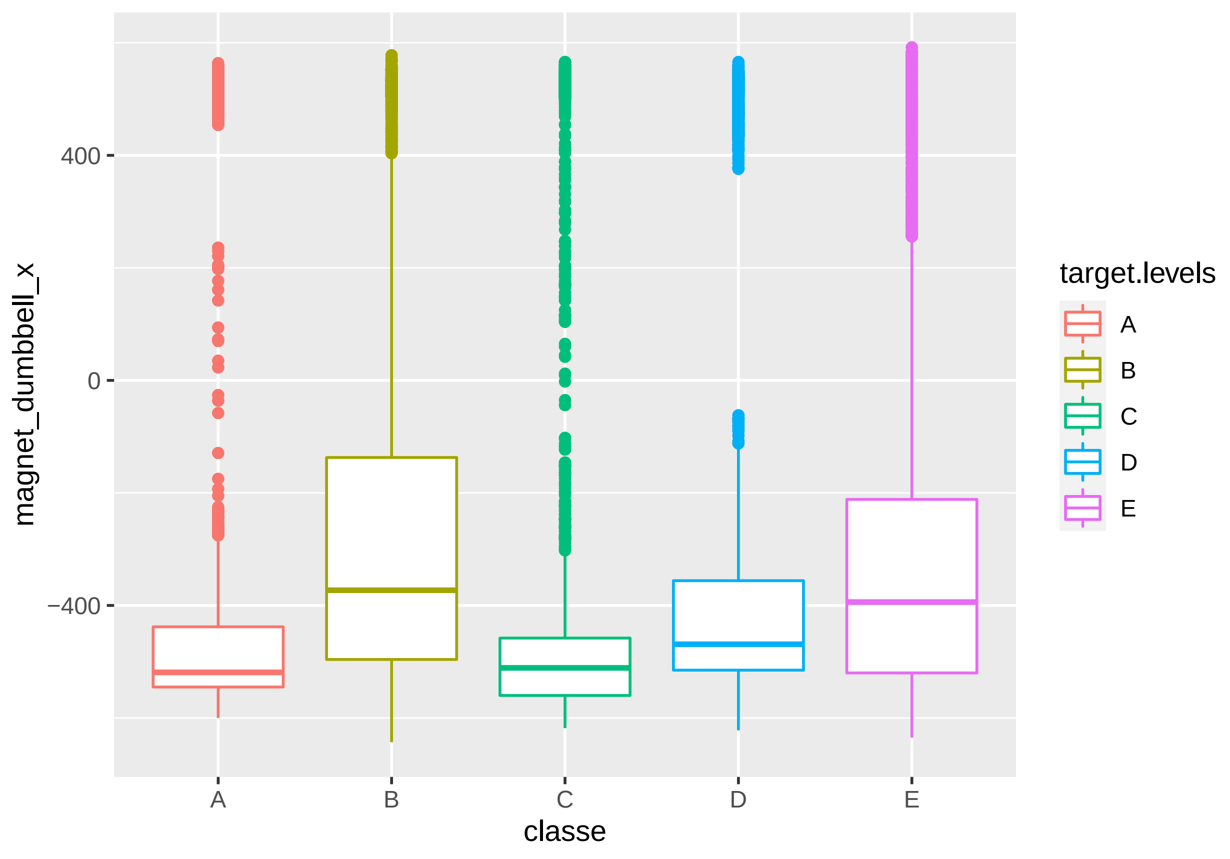
```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  NumericVars[, i]
## D = 0.078033, p-value < 2.2e-16
##
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  NumericVars[, i]
## D = 0.25045, p-value < 2.2e-16
```

We have a p-value < 0.05 , so we reject the NULL and the data we have are not normally distributed. Because of what we saw from our exploratory analysis, we may consider transforming our data.

We will also plot two random boxplots colored by each level of our target variable, because it is easy to see if we have outliers.

```
for(i in Rnum) {
  print(qplot(DataT$classe,NumericVars[,i],geom = "boxplot",
              ylab = names(NumericVars)[i],xlab =names(DataT)[length(DataT)],
              col=target.levels))
}
```





From the boxplots we see that we have outliers .

Correlated predictors

Now we will calculate the correlation between all the continuous predictors.The idea is that often we have multiple quantitative variables and sometimes they will be highly correlated with each other. In other words, they will be very similar to being the almost the exact same variable. In this case, it is not necessarily useful to include every variable in the model. We might want to include a summary that captures most of the information in those quantitative variables.

```
M <- abs(cor(NumericVars))
diag(M) <- 0
# which variables have correlation greater than 0.85
corM<-which(M >.85,arr.ind = T)
length(row.names(corM))
```

```
## [1] 26
```

```
row.names(corM)
```

```
##  [1] "total_accel_belt" "accel_belt_y"     "accel_belt_z"     "accel_belt_x"
##  [5] "magnet_belt_x"    "roll_belt"        "accel_belt_y"     "accel_belt_z"
##  [9] "pitch_belt"       "magnet_belt_x"    "roll_belt"        "total_accel_belt"
## [13] "accel_belt_z"     "roll_belt"        "total_accel_belt" "accel_belt_y"
## [17] "pitch_belt"       "accel_belt_x"     "gyros_arm_y"      "gyros_arm_x"
## [21] "gyros_dumbbell_z" "gyros_forearm_z"  "gyros_dumbbell_x" "gyros_forearm_z"
## [25] "gyros_dumbbell_x" "gyros_dumbbell_z"
```

It turns out that 26 of our predictors are highly correlated(the first with the second,the third with the fourth and so on).Therefore,including all of these predictors in the model might not necessarily be very useful.Depending on what algorithm we choose for our model, we might consider conducting a principal components analysis when we build it (PCA is most useful for linear type models like GLM, LDA).

Data spliting

```
# we use 75% to train the model and 25% to test it
 inTrain<-createDataPartition(y=DataT$classe, p=.75, list = F)
 training<-DataT[inTrain,]  # training set
 testing<-DataT[-inTrain,]  # testing set
```

Cross-Validation

```
# 10-fold cross validation
ctrl<-trainControl(method="cv",number = 10)
```

Fit a model

We choose Random Forest because it automatically selects important variables and is robust regarding correlated predictors and outliers in general.Tree models split numeric data into two groups by looking at conditional statistics of the target variable and will not benefit as much from exponential types of transformations(BoxCox, YeoJohnson, exponential).Monotone transformations(order unchanged) like log will produce the same splits.However, without proper cross-validation, the model can be over-fitted especially with large numbers of variables and results may differ from one run to the next.

```
modelFit <-caret::train(classe~.,data=training,method="rf",
        trControl = ctrl, ntree=10)
modelFit
```

```
## Random Forest
##
## 14718 samples
##    55 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13245, 13247, 13246, 13248, 13245, 13245, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9896728  0.9869359
##   28    0.9977577  0.9971639
##   55    0.9945644  0.9931241
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 28.
```

Model Predictions

```
# our predicted values
predictions<-predict(modelFit, newdata=testing,type = "raw")
```

Confusion matrix and Statistics

```
AC <-confusionMatrix(testing$classe,predictions)
AC
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  949    0    0    0
##          C    0    1  853    1    0
##          D    0    0    0  804    0
##          E    0    0    0    1  900
##
## Overall Statistics
##
##                Accuracy : 0.9994
##                  95% CI : (0.9982, 0.9999)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9992
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   1.0000   0.9975   1.0000
## Specificity            1.0000   1.0000   0.9995   1.0000   0.9998
## Pos Pred Value         1.0000   1.0000   0.9977   1.0000   0.9989
## Neg Pred Value         1.0000   0.9997   1.0000   0.9995   1.0000
## Prevalence             0.2845   0.1937   0.1739   0.1644   0.1835
## Detection Rate         0.2845   0.1935   0.1739   0.1639   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      1.0000   0.9995   0.9998   0.9988   0.9999
```

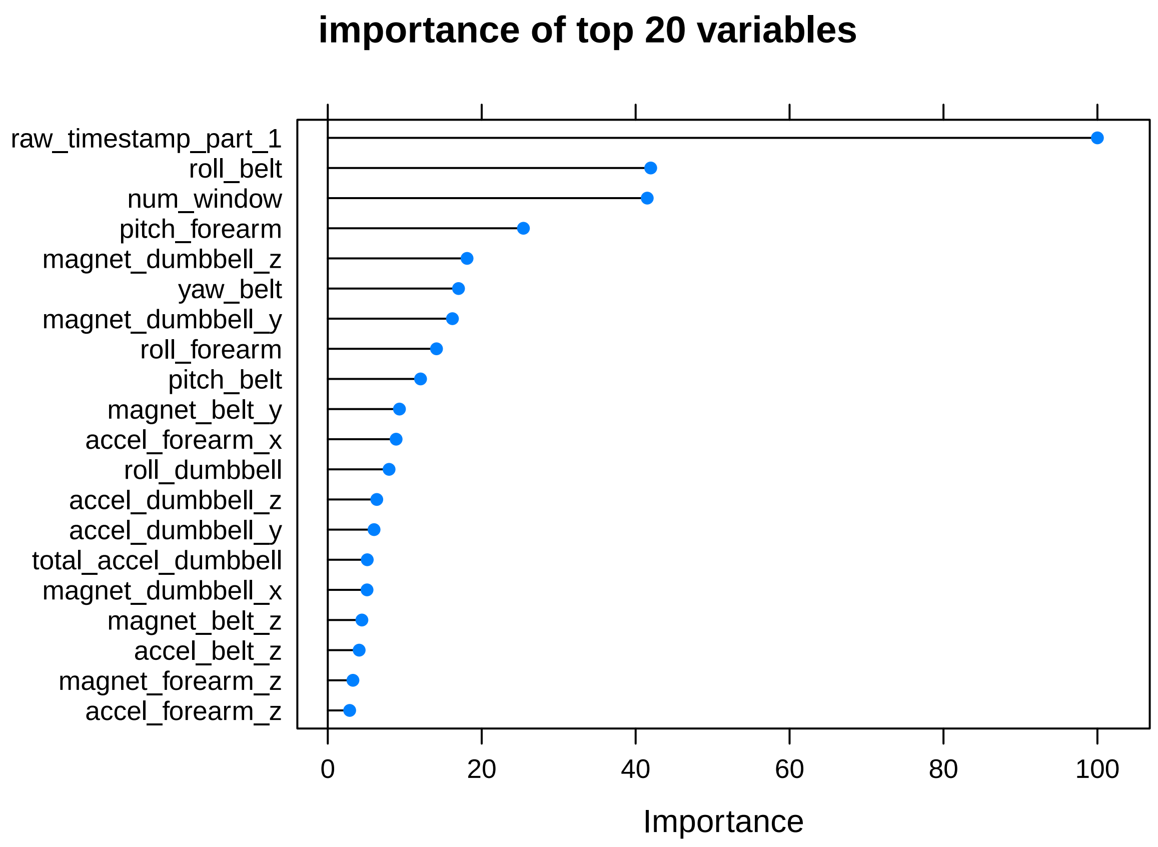
Out of sample error

```
t<-round(((1 - AC$overall[1])*100),2)
cat("out of sample error:",t,"%")
```

```
## out of sample error: 0.06 %
```

Variable importance

```
var.imp<-varImp(modelFit)
#plot the top 20 importance variables
plot(var.imp,main="importance of top 20 variables",top = 20)
```



Course Project Prediction Quiz Portion

```
results <-predict(modelFit ,newdata = DataTesting,type = "raw")
results
```

```
##  [1] 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
```

Version check and packages used

```
devtools::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.0.3 (2020-10-10)
##  os       Windows 10 x64
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United States.1252
##  ctype    English_United States.1252
##  tz       Europe/Istanbul
##  date     2021-03-10
##
## - Packages -------------------------------------------------------------------
##  package      * version    date       lib source
##  abind          1.4-5      2016-07-21 [1] CRAN (R 4.0.3)
##  assertthat     0.2.1      2019-03-21 [1] CRAN (R 4.0.3)
##  backports      1.2.1      2020-12-09 [1] CRAN (R 4.0.3)
##  broom          0.7.5      2021-02-19 [1] CRAN (R 4.0.3)
##  cachem         1.0.4      2021-02-13 [1] CRAN (R 4.0.3)
##  callr          3.5.1      2020-10-13 [1] CRAN (R 4.0.3)
##  car            3.0-10     2020-09-29 [1] CRAN (R 4.0.3)
##  carData        3.0-4      2020-05-22 [1] CRAN (R 4.0.3)
##  caret        * 6.0-86     2020-03-20 [1] CRAN (R 4.0.3)
##  cellranger     1.1.0      2016-07-27 [1] CRAN (R 4.0.3)
##  class          7.3-17     2020-04-26 [2] CRAN (R 4.0.3)
##  cli            2.3.1      2021-02-23 [1] CRAN (R 4.0.4)
##  codetools      0.2-16     2018-12-24 [2] CRAN (R 4.0.3)
##  colorspace     2.0-0      2020-11-11 [1] CRAN (R 4.0.3)
##  crayon         1.4.1      2021-02-08 [1] CRAN (R 4.0.3)
##  curl           4.3        2019-12-02 [1] CRAN (R 4.0.3)
##  data.table     1.14.0     2021-02-21 [1] CRAN (R 4.0.4)
##  DBI            1.1.1      2021-01-15 [1] CRAN (R 4.0.3)
##  desc           1.2.0      2018-05-01 [1] CRAN (R 4.0.3)
##  devtools       2.3.2      2020-09-18 [1] CRAN (R 4.0.3)
##  digest         0.6.27     2020-10-24 [1] CRAN (R 4.0.3)
##  dplyr        * 1.0.4      2021-02-02 [1] CRAN (R 4.0.3)
##  e1071          1.7-4      2020-10-14 [1] CRAN (R 4.0.3)
##  ellipsis       0.3.1      2020-05-15 [1] CRAN (R 4.0.3)
##  evaluate       0.14       2019-05-28 [1] CRAN (R 4.0.3)
##  fansi          0.4.2      2021-01-15 [1] CRAN (R 4.0.3)
##  farver         2.1.0      2021-02-28 [1] CRAN (R 4.0.4)
##  fastmap        1.1.0      2021-01-25 [1] CRAN (R 4.0.3)
##  forcats        0.5.1      2021-01-27 [1] CRAN (R 4.0.3)
##  foreach        1.5.1      2020-10-15 [1] CRAN (R 4.0.3)
##  foreign        0.8-80     2020-05-24 [2] CRAN (R 4.0.3)
##  fs             1.5.0      2020-07-31 [1] CRAN (R 4.0.3)
##  gdata          2.18.0     2017-06-06 [1] CRAN (R 4.0.3)
##  generics       0.1.0      2020-10-31 [1] CRAN (R 4.0.3)
##  ggplot2      * 3.3.3      2020-12-30 [1] CRAN (R 4.0.3)
##  ggpubr       * 0.4.0      2020-06-27 [1] CRAN (R 4.0.3)
##  ggsignif       0.6.1      2021-02-23 [1] CRAN (R 4.0.4)
##  glue           1.4.2      2020-08-27 [1] CRAN (R 4.0.3)
##  gmodels      * 2.18.1     2018-06-25 [1] CRAN (R 4.0.3)
##  gower          0.2.2      2020-06-23 [1] CRAN (R 4.0.3)
##  gtable          0.3.0      2019-03-25 [1] CRAN (R 4.0.3)
##  gtools          3.8.2      2020-03-31 [1] CRAN (R 4.0.3)
##  haven           2.3.1      2020-06-01 [1] CRAN (R 4.0.3)
##  highr           0.8        2019-03-20 [1] CRAN (R 4.0.3)
##  hms             1.0.0      2021-01-13 [1] CRAN (R 4.0.3)
##  htmltools       0.5.1.1    2021-01-22 [1] CRAN (R 4.0.3)
##  ipred           0.9-10     2021-03-04 [1] CRAN (R 4.0.3)
##  iterators       1.0.13     2020-10-15 [1] CRAN (R 4.0.3)
##  knitr           1.31       2021-01-27 [1] CRAN (R 4.0.3)
##  labeling        0.4.2      2020-10-20 [1] CRAN (R 4.0.3)
##  lattice       * 0.20-41    2020-04-02 [2] CRAN (R 4.0.3)
##  lava            1.6.8.1    2020-11-04 [1] CRAN (R 4.0.3)
##  lifecycle       1.0.0      2021-02-15 [1] CRAN (R 4.0.4)
##  lubridate       1.7.10     2021-02-26 [1] CRAN (R 4.0.4)
##  magrittr        2.0.1      2020-11-17 [1] CRAN (R 4.0.3)
##  MASS            7.3-53.1   2021-02-12 [1] CRAN (R 4.0.4)
##  Matrix          1.2-18     2019-11-27 [2] CRAN (R 4.0.3)
##  memoise         2.0.0      2021-01-26 [1] CRAN (R 4.0.3)
##  ModelMetrics    1.2.2.2    2020-03-17 [1] CRAN (R 4.0.3)
##  munsell         0.5.0      2018-06-12 [1] CRAN (R 4.0.3)
##  nlme            3.1-149    2020-08-23 [2] CRAN (R 4.0.3)
##  nnet            7.3-15     2021-01-24 [1] CRAN (R 4.0.3)
##  nortest       * 1.0-4      2015-07-30 [1] CRAN (R 4.0.3)
##  openxlsx        4.2.3      2020-10-27 [1] CRAN (R 4.0.3)
##  pillar          1.5.0      2021-02-22 [1] CRAN (R 4.0.4)
##  pkgbuild        1.2.0      2020-12-15 [1] CRAN (R 4.0.3)
##  pkgconfig       2.0.3      2019-09-22 [1] CRAN (R 4.0.3)
##  pkgload         1.2.0      2021-02-23 [1] CRAN (R 4.0.4)
##  plyr            1.8.6      2020-03-03 [1] CRAN (R 4.0.3)
##  prettyunits     1.1.1      2020-01-24 [1] CRAN (R 4.0.3)
##  pROC            1.17.0.1   2021-01-13 [1] CRAN (R 4.0.3)
##  processx        3.4.5      2020-11-30 [1] CRAN (R 4.0.3)
##  prodlim         2019.11.13 2019-11-17 [1] CRAN (R 4.0.3)
##  ps              1.6.0      2021-02-28 [1] CRAN (R 4.0.4)
##  purrr           0.3.4      2020-04-17 [1] CRAN (R 4.0.3)
##  R6              2.5.0      2020-10-28 [1] CRAN (R 4.0.3)
##  randomForest    4.6-14     2018-03-25 [1] CRAN (R 4.0.3)
##  Rcpp            1.0.6      2021-01-15 [1] CRAN (R 4.0.3)
##  readxl          1.3.1      2019-03-13 [1] CRAN (R 4.0.3)
##  recipes         0.1.15     2020-11-11 [1] CRAN (R 4.0.3)
##  remotes         2.2.0      2020-07-21 [1] CRAN (R 4.0.3)
##  reshape2        1.4.4      2020-04-09 [1] CRAN (R 4.0.4)
##  rio             0.5.26     2021-03-01 [1] CRAN (R 4.0.4)
##  rlang           0.4.10     2020-12-30 [1] CRAN (R 4.0.3)
##  rmarkdown       2.7        2021-02-19 [1] CRAN (R 4.0.3)
##  rpart           4.1-15     2019-04-12 [2] CRAN (R 4.0.3)
##  rprojroot       2.0.2      2020-11-15 [1] CRAN (R 4.0.3)
##  rstatix         0.7.0      2021-02-13 [1] CRAN (R 4.0.4)
##  scales          1.1.1      2020-05-11 [1] CRAN (R 4.0.4)
##  sessioninfo     1.1.1      2018-11-05 [1] CRAN (R 4.0.3)
##  stringi         1.5.3      2020-09-09 [1] CRAN (R 4.0.3)
##  stringr         1.4.0      2019-02-10 [1] CRAN (R 4.0.3)
##  survival        3.2-7      2020-09-28 [2] CRAN (R 4.0.3)
##  testthat        3.0.2      2021-02-14 [1] CRAN (R 4.0.4)
##  tibble         3.1.0      2021-02-25 [1] CRAN (R 4.0.3)
##  tidyr          1.1.3      2021-03-03 [1] CRAN (R 4.0.3)
##  tidyselect     1.1.0      2020-05-11 [1] CRAN (R 4.0.3)
##  timeDate       3043.102   2018-02-21 [1] CRAN (R 4.0.3)
##  usethis        2.0.1      2021-02-10 [1] CRAN (R 4.0.3)
##  utf8           1.1.4      2018-05-24 [1] CRAN (R 4.0.3)
##  vctrs          0.3.6      2020-12-17 [1] CRAN (R 4.0.3)
##  withr          2.4.1      2021-01-26 [1] CRAN (R 4.0.3)
##  xfun           0.21       2021-02-10 [1] CRAN (R 4.0.3)
##  yaml           2.2.1      2020-02-01 [1] CRAN (R 4.0.3)
##  zip            2.1.1      2020-08-27 [1] CRAN (R 4.0.3)
##
## [1] C:/Users/izzyd/Documents/R/win-library/4.0
## [2] C:/Program Files/R/R-4.0.3/library
```