Weight Lifting Exercise Data Analysis

Summary

“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” Through data analysis we will try to determine if the users performed the exercise correctly

Data

We are using the data provided by coursera of the study by Groupware@LES (mailto:Groupware@LES)

Training Data - CSV (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv) Test Data - CVS (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Data Processing

The first thing that we need to do is to remove columns with NA’s in it, and the first 7 columns that have identifiers to leave the columns we will be working with. While it might hinder the accuracy of the model I’m taking a 99.9% confident sample of the data,this will allow my 4GB of RAM process the data in less time (I tried it the first time with 19K observations and 60 columns, after a couple of hours I found my model was overfitted) . I’m setting a seed of the sample I’m taking to allow reproducibility of the model

I’m setting a 70% training & 30% testing data sample split.

```
#Remove NA's
trainingD<-trainingD[ , apply(trainingD, 2, function(x) !any(is.na(x)))]
#Remove identifier columns
trainingD <- trainingD[,-c(1:7)]
table(trainingD$classe)
```

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

```
## GET SAMPLE
size<-as.integer((as.character(nrow(trainingD))))

sample.size = function(c.lev, margin=.5,
                       c.interval=.05, population) {
  z.val = qnorm(.5+c.lev/200)
  ss = (z.val^2 * margin * (1-margin))/c.interval^2
  p.ss = round((ss/(1 + ((ss-1)/population))), digits=0)
  METHOD = paste("Recommended sample size for a population of ",
                 population, " at a ", c.lev,
                 "% confidence level", sep = "")
  structure(list(Population = population,
                 "Confidence level" = c.lev,
                 "Margin of error" = c.interval,
                 "Response distribution" = margin,
                 "Recommended sample size" = p.ss,
                 method = METHOD),
            class = "power.htest")
}

sampleSize<-sample.size(99.9, , , size)



ss<-as.integer(sampleSize[5])

set.seed(1)
trainS<-trainingD[sample (nrow(trainingD), ss),]
table(trainS$classe)
```

```
##
##   A   B   C   D   E
## 288 222 164 162 190
```

```
inTrain<-createDataPartition(y=trainS$classe, p=.7, list=F)
training<-trainS[inTrain,]
testing<-trainS[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 720  53
```

```
## [1] 306  53
```

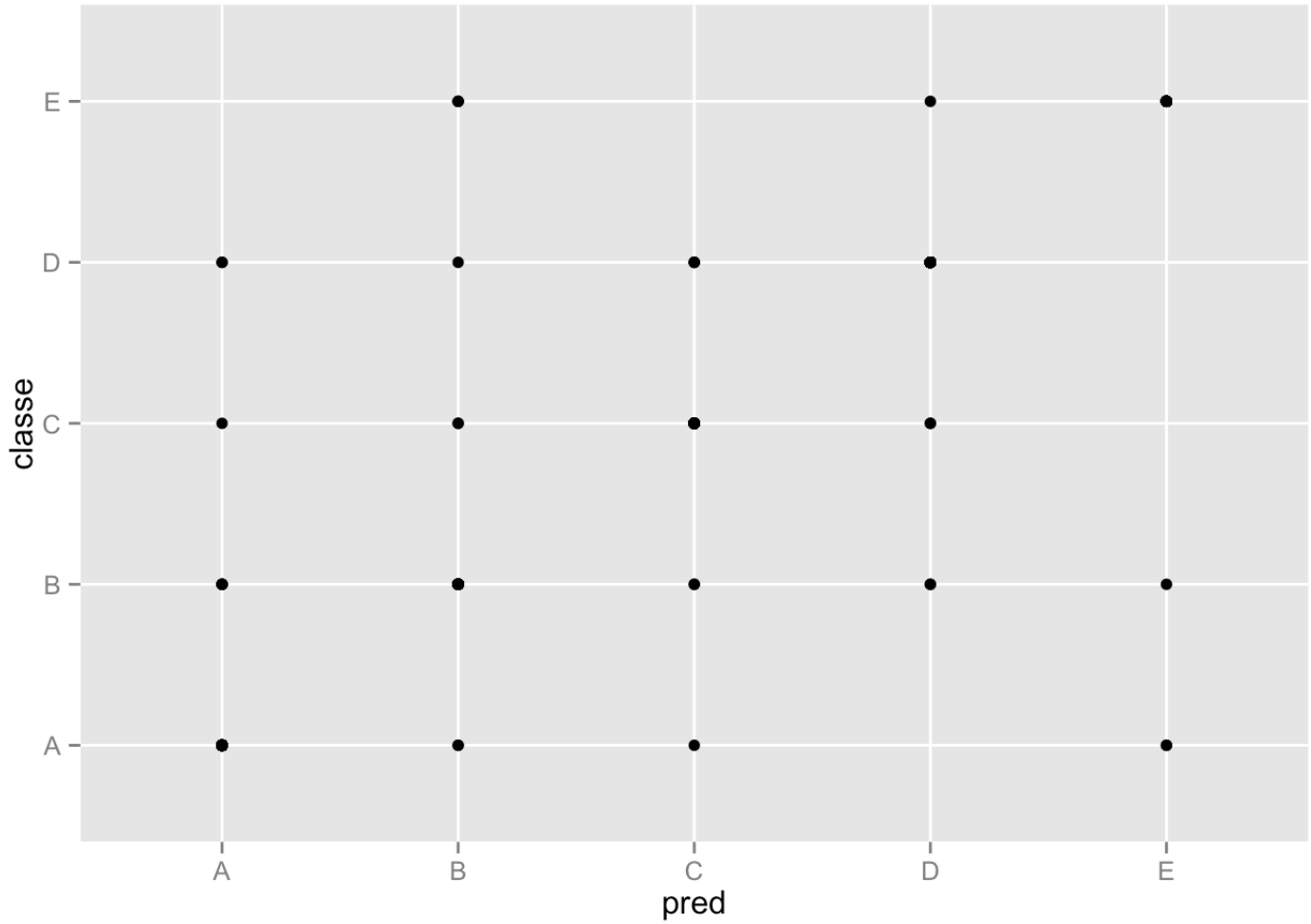
Model Building & testing

I'm using a Stochastic Gradient Boosting model with the sample that resulted in a 80% accuracy and 20% error rate. While the model is not perfect it does provide a good rate of accuracy

```
modFit<-train(classe~.,method="gbm", data=training, verbose=F)
print(modFit)
```

```
## Stochastic Gradient Boosting
##
## 720 samples
##  52 predictor
##   5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 720, 720, 720, 720, 720, 720, ...
##
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa   Accuracy SD  Kappa SD
##   1                   50      0.6612    0.5694  0.03512      0.04464
##   1                  100      0.7142    0.6377  0.03040      0.03803
##   1                  150      0.7424    0.6734  0.02741      0.03383
##   2                   50      0.7384    0.6681  0.02945      0.03663
##   2                  100      0.7792    0.7200  0.02766      0.03444
##   2                  150      0.7929    0.7374  0.02459      0.03060
##   3                   50      0.7688    0.7068  0.02542      0.03144
##   3                  100      0.7955    0.7405  0.02370      0.02949
##   3                  150      0.8057    0.7535  0.02276      0.02824
##
## 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.
```

```
pred<-predict(modFit,testing);
qplot(pred, classe, data=testing)
```




```
table(pred, testing$classe)
```

```
##
## pred  A  B  C  D  E
##    A 81  4  1  2  0
##    B  2 57  2  1  3
##    C  1  2 44  3  0
##    D  0  2  2 42  1
##    E  2  1  0  0 53
```

```
testingD<-read.csv("~/Dropbox/Coursera - R/Practical Machine Learning/Proy/pml-testing.csv" ,
na.strings=c("","NA"), header=T)
testingD<-testingD[ , apply(testingD, 2, function(x) !any(is.na(x)))]
finalTesting <- testingD[,-c(1:7)]
finalPrediction<-predict(modFit,finalTesting);
print(finalPrediction)
```

```
##  [1] B A B A C E D D A A C A B A E E A B A B
## Levels: A B C D E
```

```
```

Final Test

As expected I got 18 out of 20 correct results in the final test.