Predicting exercise performance with machine learning

Executive 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. The goal is to predict the manner in which participants did the exercise by using data from accelerometers on the belt, forearm, arm, and dumbell to train a classifier with machine learning. Following data exploration and cleaning (removing NA variables, removing variables with low variance), various models were used to train classifiers: Decision trees, Linear Discriminant Analysis, Generalized Boosted Regression, and Random Forest. The accuracy of the trained classifiers was highest for the Random Forest model. Using the Random Forest based prediction model, a prediction is made for the manner in which the exercises were done by 20 different test cases.

The data

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. Data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants were recorded while participants performed barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har First, the data sets for training and choosing the best classifier, as well as the data set for the final prediction of exercise performance are loaded:

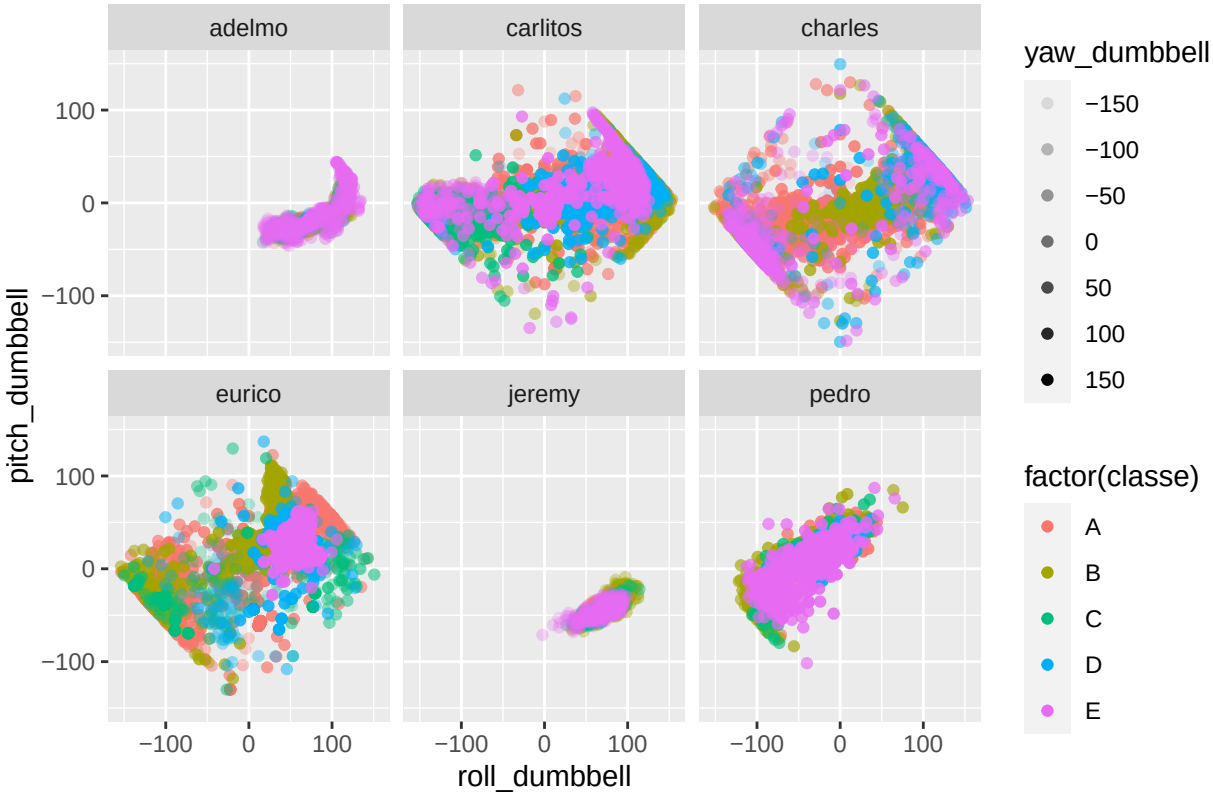
```
setwd("/Users/huib/Documents/Huib/Coursera/DataScience/8-Practical Machine Learning/CourseProject")
tmptrain <- read.csv("pml-training.csv", header=TRUE)
tmptest <- read.csv("pml-testing.csv", header=TRUE)
#dim(tmptrain);dim(tmptest);summary(tmptrain)
```

Exploratory data analysis

What does the data look like? The data consist of measurements from accelerometers on the belt, forearm, arm, and dumbell of 6 participants: Adelmo, Carlito, Charles, Eurico, Jeremy and Pedro. Figures 1 and 2 show that arm movements during dumbell weight lifting varied across the different participants ## Figure 1

```
library(ggplot2)
#knitr::opts_chunk$set(fig.width=5, fig.height=3)
g <- ggplot(tmptrain, aes(x=roll_dumbbell, y=pitch_dumbbell))
g <- g + geom_point(aes(colour = factor(classe), alpha=yaw_dumbbell))
g <- g + facet_wrap(~user_name)
g <- g+  labs(title="Fig 1: Dumbell movement per subject in training data")
g
```

Fig 1: Dumbell movement per subject in training data



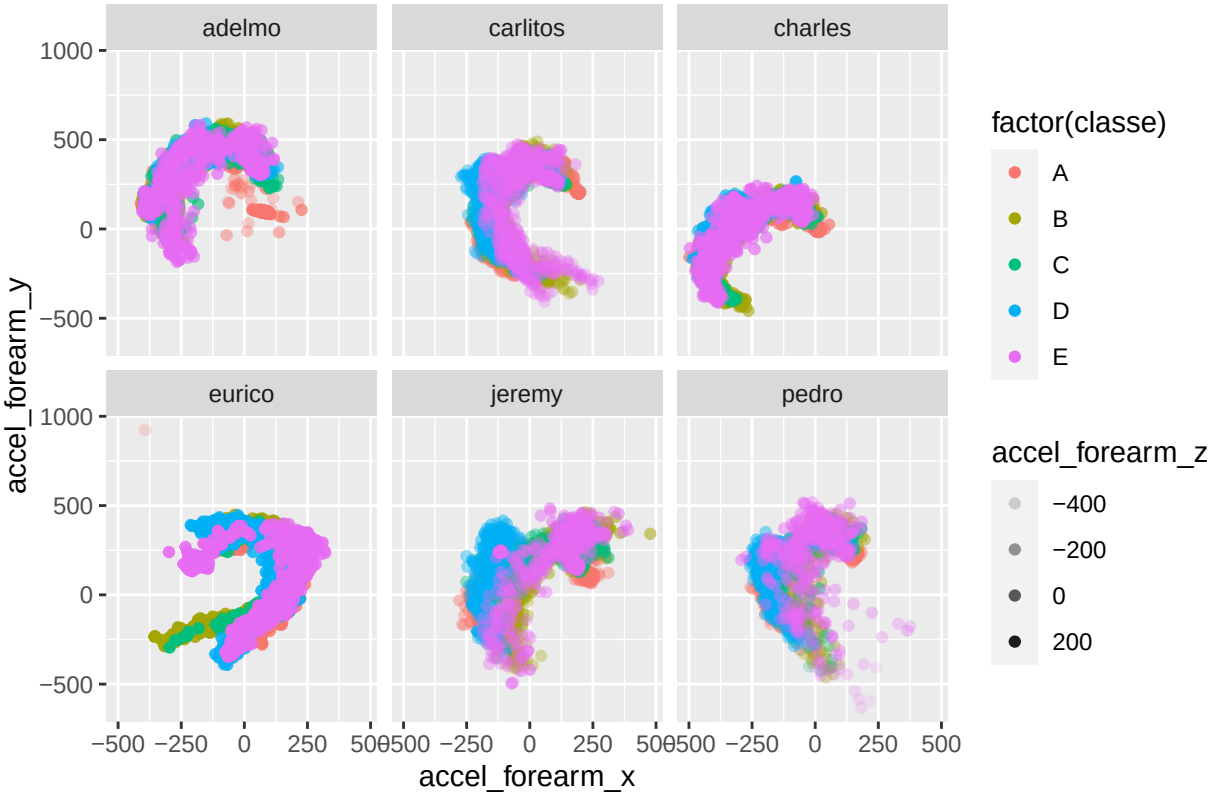
Looking through the summary(tmptrain) data shows that many variables had 'NA' values. These variables were removed from the train and test data sets. Also the first seven columns of the data set contained names, dates and time stamps, which are also not helpful for training classifiers. These first seven columns were also removed from the train and test (prediction) data sets.

```
dftrain <- tmptrain[, colSums(is.na(tmptrain)) == 0]
dftest <- tmptest[, colSums(is.na(tmptrain)) == 0]
dftrain <- dftrain[, -c(1:7)]
dftest <- dftest[, -c(1:7)]
```

Figure 2

```
g <- ggplot(tmptrain, aes(x=accel_forearm_x, y=accel_forearm_y))
g <- g + geom_point(aes(colour = factor(classe), alpha=accel_forearm_z))
g <- g + facet_wrap(~user_name)
g <- g+  labs(title="Fig 2: Forearm accelleration per subject in training data")
g
```

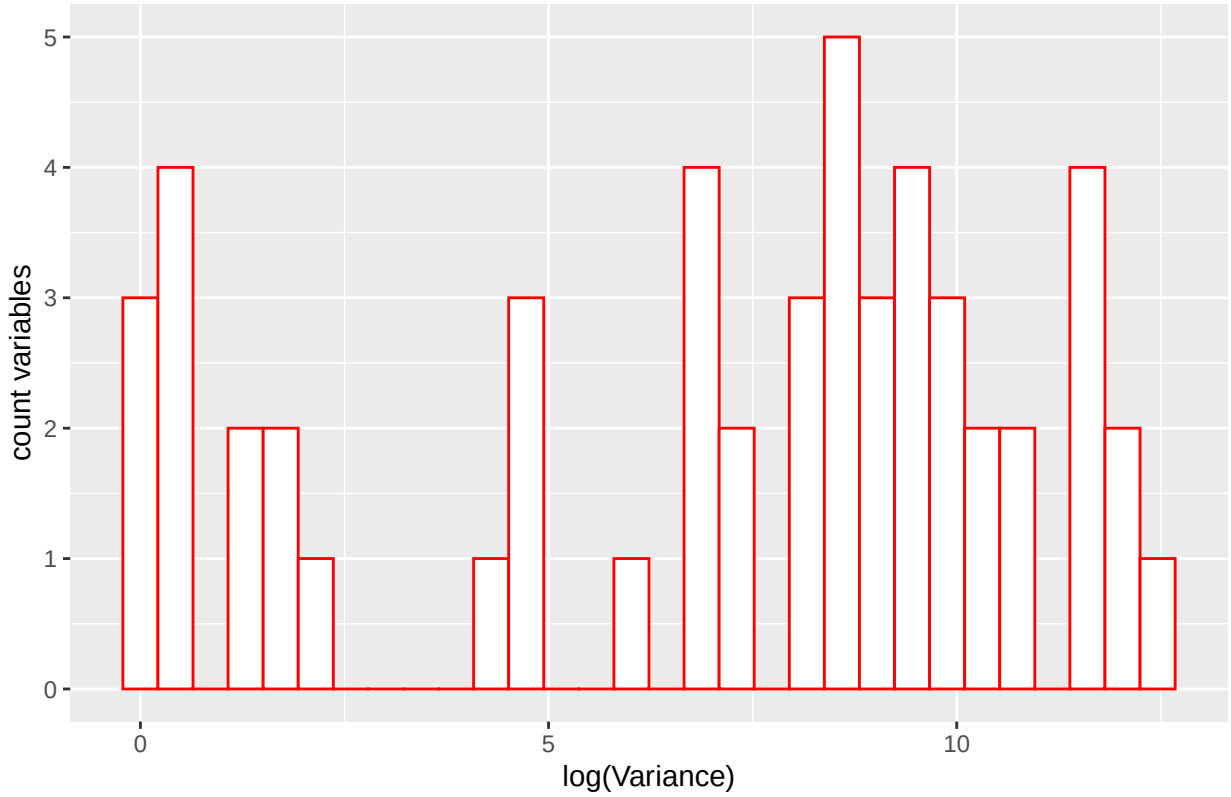
Fig 2: Forearm accelleration per subject in training data



Next, there were several variables that showed very low variance in the data. Figure 3 shows the log transformed variance of each of the variables. The histogram shows that there are many variables with a log(Variance) below or close to 1. These will carry little information for a classifier to train on.
Figure 3

```
x <- data.frame(log(apply(dftrain,2,var)+1))
g <- ggplot(x, aes(x=x[,1], na.rm=TRUE))
g <- g + geom_histogram(color="red", fill="white")
g <- g + labs(title="Fig 3: Variance distribution across variables", x="log(Variance)", y="count variabl
g
```

Fig 3: Variance distribution across variables



The R package 'caret' contains the function 'nearZeroVar' that identifies variables with low variance. This function was used to remove these low variance variables from the train and test data.

```
# find and remove variables with little variance (nearZeroVar in caret package)
library(caret)
NZV <- nearZeroVar(dftrain)
dftrain <- dftrain[, -NZV]
dftest  <- dftest[, -NZV]
```

Cross validation

To train the classifiers, test the 'accuracy' and 'out of sample error' various classifiers, and to validate the performance of the best classifier, the cleaned 'pml-training' data set was partitioned into three subsets:
- a training set
- a testing set
- validation set

The classifiers are first trained on the training set, their performance tested on the test data set, and the best model is tested on the validation set.

```
set.seed(1212)
inBuild <- createDataPartition(y=dftrain$classe,p=0.7, list=FALSE)
validation <- dftrain[-inBuild,]
buildData <- dftrain[inBuild,]
inTrain <- createDataPartition(y=buildData$classe, p=0.7, list=FALSE)
training <- buildData[inTrain,]
testing <- buildData[-inTrain,]
```

Fit decision tree models with and without PCA preprocessing:

Since the variables that measure the movements of the participants can result in correlated data, first it is tested whether reducing the number of variables in the data by dimension reduction with principal component analysis (PCA) will benefit training a decision tree based classifier: ### decision tree with PCA

```
set.seed(2323)
mod_treePCA <- train(classe~.,method="rpart", preProcess="pca", data=training)
pred_mod_treePCA <- predict(mod_treePCA, testing)
confmatrtreePCA <- confusionMatrix(pred_mod_treePCA, testing$classe)
round(confmatrtreePCA$overall,3)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##          0.385          0.176          0.370          0.400          0.284
## AccuracyPValue  McnemarPValue
##          0.000            NaN
```

decision tree

```
mod_tree <- train(classe~.,method="rpart",data=training)
pred_mod_tree <- predict(mod_tree, testing)
confmatrtree <- confusionMatrix(pred_mod_tree, testing$classe)
round(confmatrtree$overall,3)
```

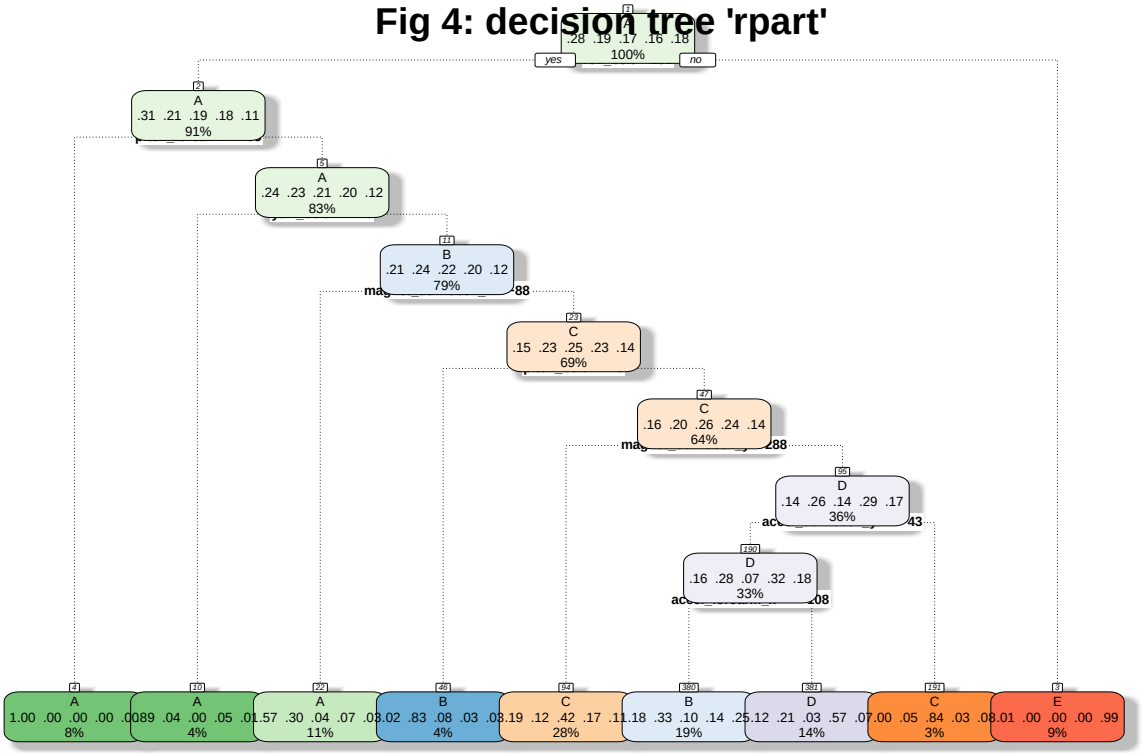
```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##          0.554          0.442          0.539          0.570          0.284
## AccuracyPValue  McnemarPValue
##          0.000          0.000
```

Figure 4 shows the result of the decision tree fitting to the training data. The confusion matrices show that the PCA preprocessing did not improve the accuracy and out of sample error (below 0.5, calculated as 1-accuracy). Overall, the performance of this model is very low (accuracy < 0.6)

Figure 4

```
library(rattle)
fancyRpartPlot(mod_tree$finalModel, main="Fig 4: decision tree 'rpart'", cex=.4)
```

Fig 4: decision tree 'rpart'



Rattle 2020−Jun−04 16:09:42 huib

Fit LDA, GBR and RF models:

To test other approaches, the following classifiers were trained: - Linear Discriminant Analysis (LDA) - Generalized Boosted Regression (GBR) - Random Forest (RF) ### linear discriminant analysis

```
library(mgcv)
library(nlme)
mod_lda  <- train(classe ~ ., data=training, method = "lda", verbose = FALSE)
pred_mod_lda <- predict(mod_lda, newdata=testing)
confmatrLDA <- confusionMatrix(pred_mod_lda, testing$classe)
round(confmatrLDA$overall,3)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##          0.695          0.615          0.681          0.709          0.284
## AccuracyPValue  McnemarPValue
##          0.000          0.000
```

Generalized Boosted Regression

```
cntrlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
mod_gbm  <- train(classe ~ ., data=training, method = "gbm", trControl = cntrlGBM, verbose = FALSE)
pred_mod_gbm <- predict(mod_gbm, newdata=testing)
confmatrGBM <- confusionMatrix(pred_mod_gbm, testing$classe)
round(confmatrGBM$overall,3)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##          0.958          0.947          0.952          0.964          0.284
## AccuracyPValue  McnemarPValue
##          0.000          0.001
```

Random Forest:

```
controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)
mod_rf <- train(classe ~ ., data=training, method="rf", trControl=controlRF)
pred_mod_rf <- predict(mod_rf, newdata=testing)
confmatrRF <- confusionMatrix(pred_mod_rf, testing$classe)
round(confmatrRF$overall,3)
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##          0.990          0.987          0.986          0.992          0.284
## AccuracyPValue  McnemarPValue
##          0.000            NaN
```

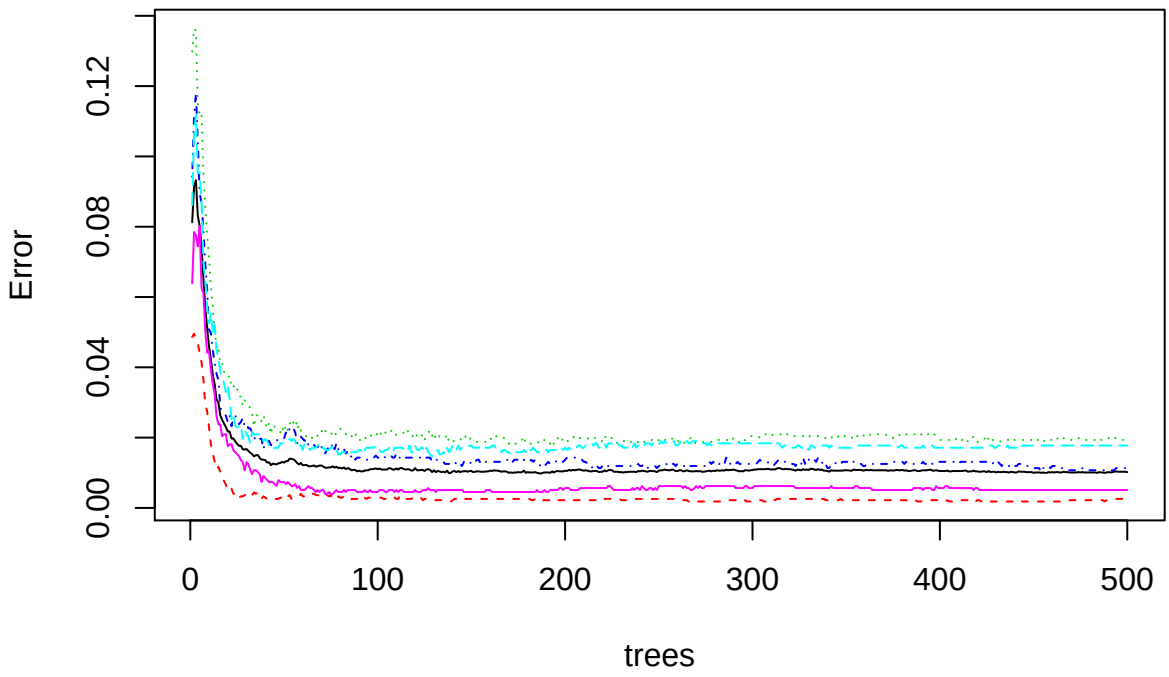
expected out of sample error of Random Forest

The results from the confusion matrices show that boosting regresion strongly improves performance. The accuracy is well above 0.95. The best classifier was based on random forests with an out of sample error of about 1% (1-accuracy). Figure 5 shows that progress during training of the Random Forest classifier.

Figure 5

```
plot(mod_rf$finalModel, main="Fig 5: Random Forest result, black line = average error")
```

Fig 5: Random Forest result, black line = average error



Random Forest performance in the validation data set.

The table below shows the prediction performance of the trained Random Forest classifier, with predictions on validation set versus the true values of the 'classe' variable.

```
val_mod_rf <- predict(mod_rf, newdata=validation)
confmatrRFval <- confusionMatrix(val_mod_rf, validation$classe)
confmatrRFval$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9886151      0.9855971      0.9855639      0.9911663      0.2844520
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

```
table(val_mod_rf,validation$classe)
```

```
##
## val_mod_rf    A    B    C    D    E
##          A 1671   10    0    0    0
##          B    2 1124    8    0    0
##          C    0    5 1011   24    3
##          D    0    0    7  937    4
##          E    1    0    0    3 1075
```

Use the best model (Random Forest) to predict the scores on the pml-testing data

Finally, these are the predictions of the best performing classifier of the 'pml-testing' data for the Quiz

```
QuizPrediction <- predict(mod_rf, newdata=dftest)
QuizPrediction
```

```
##  [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
```