fuel train

Chris Peck

December 21, 2016

Description

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E). More information 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). 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).

The goal of this project is to create an algorithm to identify how well participants performed the dumbbbell bicep curls.

Load the data

Load the training and test data sets

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

filetest<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

traindata <- read.csv(file=filetrain, header=TRUE, sep=",")
testdata <- read.csv(file=filetest, header=TRUE, sep=",")
```

clean the data

After viewing the data, it was evident that many fields contained blanks and NAs for the majority of the measurements (19,216 out of 19.622 observations). These fields were eliminated from the data we use since they can't be used in the prediction algorithm and to help speed up the calculations on the data set.

```
##convert blanks to NA
traindatana <- read.csv(file=filetrain, header=TRUE, sep=",",na.strings=c(""," ","NA"
))
na_count <-sapply(traindatana, function(y) sum(length(which(is.na(y)))))
na_count <- data.frame(na_count)
##na_count

##we see 19,216 NAs for many of the variables out of the 19,622 observations so we re
move these columns to reduce the number of variables for our model fitting.  Statisti
cs not shown to conserve space in the report.
traindataclean<-traindatana[ , ! apply( traindatana , 2 , function(x) any(is.na(x)) )
]

##remove the first 7 columns which aren't part of the measurements taken by the monit
or
traindataclean<-traindataclean[,c(8:60)]

##we can see there are no more variables that are blank or NA
head(traindataclean)
```

```
##   roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x gyros_belt_y
## 1      1.41       8.07    -94.4                3         0.00         0.00
## 2      1.41       8.07    -94.4                3         0.02         0.00
## 3      1.42       8.07    -94.4                3         0.00         0.00
## 4      1.48       8.05    -94.4                3         0.02         0.00
## 5      1.48       8.07    -94.4                3         0.02         0.02
## 6      1.45       8.06    -94.4                3         0.02         0.00
##   gyros_belt_z accel_belt_x accel_belt_y accel_belt_z magnet_belt_x
## 1        -0.02          -21            4           22            -3
## 2        -0.02          -22            4           22            -7
## 3        -0.02          -20            5           23            -2
## 4        -0.03          -22            3           21            -6
## 5        -0.02          -21            2           24            -6
## 6        -0.02          -21            4           21             0
##   magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm total_accel_arm
## 1           599          -313     -128      22.5    -161              34
## 2           608          -311     -128      22.5    -161              34
## 3           600          -305     -128      22.5    -161              34
## 4           604          -310     -128      22.1    -161              34
## 5           600          -302     -128      22.1    -161              34
## 6           603          -312     -128      22.0    -161              34
##   gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x accel_arm_y accel_arm_z
## 1        0.00        0.00       -0.02        -288         109        -123
## 2        0.02       -0.02       -0.02        -290         110        -125
## 3        0.02       -0.02       -0.02        -289         110        -126
## 4        0.02       -0.03        0.02        -289         111        -123
## 5        0.00       -0.03        0.00        -289         111        -123
## 6        0.02       -0.03        0.00        -289         111        -122
##   magnet_arm_x magnet_arm_y magnet_arm_z roll_dumbbell pitch_dumbbell
```

```
## 1        -368          337          516      13.05217      -70.49400
## 2        -369          337          513      13.13074      -70.63751
## 3        -368          344          513      12.85075      -70.27812
## 4        -372          344          512      13.43120      -70.39379
## 5        -374          337          506      13.37872      -70.42856
## 6        -369          342          513      13.38246      -70.81759
##   yaw_dumbbell total_accel_dumbbell gyros_dumbbell_x gyros_dumbbell_y
## 1    -84.87394                   37                0            -0.02
## 2    -84.71065                   37                0            -0.02
## 3    -85.14078                   37                0            -0.02
## 4    -84.87363                   37                0            -0.02
## 5    -84.85306                   37                0            -0.02
## 6    -84.46500                   37                0            -0.02
##   gyros_dumbbell_z accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z
## 1             0.00             -234               47             -271
## 2             0.00             -233               47             -269
## 3             0.00             -232               46             -270
## 4            -0.02             -232               48             -269
## 5             0.00             -233               48             -270
## 6             0.00             -234               48             -269
##   magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z roll_forearm
## 1              -559               293               -65         28.4
## 2              -555               296               -64         28.3
## 3              -561               298               -63         28.3
## 4              -552               303               -60         28.1
## 5              -554               292               -68         28.0
## 6              -558               294               -66         27.9
##   pitch_forearm yaw_forearm total_accel_forearm gyros_forearm_x
## 1         -63.9        -153                  36            0.03
## 2         -63.9        -153                  36            0.02
## 3         -63.9        -152                  36            0.03
## 4         -63.9        -152                  36            0.02
## 5         -63.9        -152                  36            0.02
## 6         -63.9        -152                  36            0.02
##   gyros_forearm_y gyros_forearm_z accel_forearm_x accel_forearm_y
## 1            0.00           -0.02             192             203
## 2            0.00           -0.02             192             203
## 3           -0.02            0.00             196             204
## 4           -0.02            0.00             189             206
## 5            0.00           -0.02             189             206
## 6           -0.02           -0.03             193             203
##   accel_forearm_z magnet_forearm_x magnet_forearm_y magnet_forearm_z
## 1            -215              -17              654              476
## 2            -216              -18              661              473
## 3            -213              -18              658              469
## 4            -214              -16              658              469
## 5            -214              -17              655              473
## 6            -215               -9              660              478
##   classe
## 1      A
```

```
## 2       A
## 3       A
## 4       A
## 5       A
## 6       A
```

Split the training data into a training set and a test set

We split the training data into a training subset and a test subset so that we can do a cross-validation of the model prior to applying it to the 20 samples in the test data

```
library(caret)
```

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

```
trainIndex <- createDataPartition(traindataclean$classe, p=0.7, list=FALSE)
data_train_train_subset <- traindataclean[ trainIndex,]
data_train_test_subset <- traindataclean[-trainIndex,]
```

Create various models for prediction

Try rpart model for prediction

```
install.packages("rattle")
```

```
## package 'rattle' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\cpeck\AppData\Local\Temp\RtmpYnjBKK\downloaded_packages
```

```
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

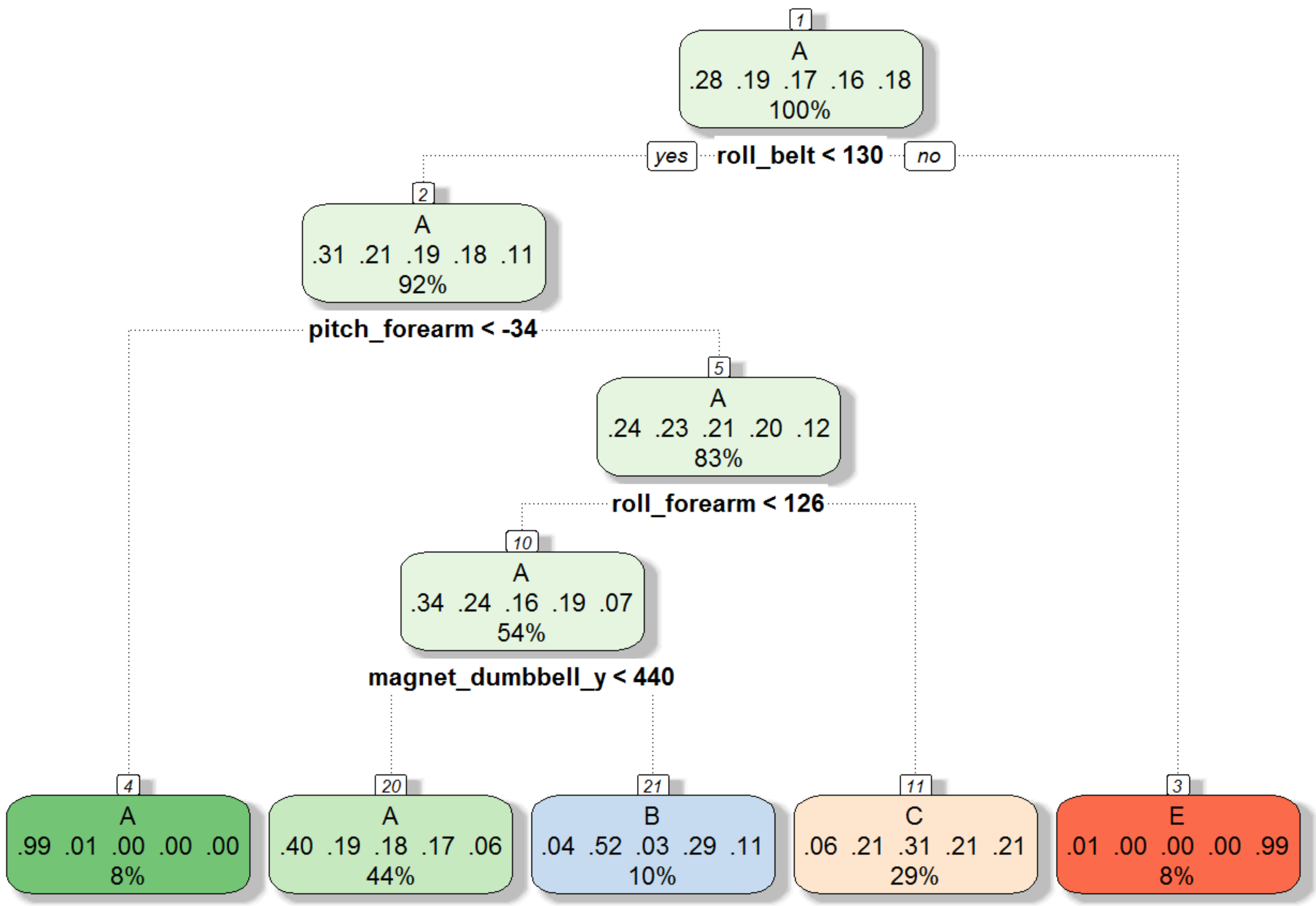
```
modelfittree<-train(classe~.,method="rpart", data=data_train_train_subset, control =
rpart.control(maxdepth = 5))
```

```
## Loading required package: rpart
```

```
print(modelfittree$finalModel)
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12570 8672 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1123    7 A (0.99 0.0062 0 0 0) *
##      5) pitch_forearm>=-33.95 11447 8665 A (0.24 0.23 0.21 0.2 0.12)
##       10) roll_forearm< 126.5 7410 4882 A (0.34 0.24 0.16 0.19 0.067)
##         20) magnet_dumbbell_y< 439.5 6103 3633 A (0.4 0.19 0.18 0.17 0.058) *
##         21) magnet_dumbbell_y>=439.5 1307  630 B (0.044 0.52 0.033 0.29 0.11) *
##       11) roll_forearm>=126.5 4037 2801 C (0.063 0.21 0.31 0.21 0.21) *
##    3) roll_belt>=130.5 1167    8 E (0.0069 0 0 0 0.99) *
```

```
fancyRpartPlot(modelfittree$finalModel)
```



Rattle 2016-Dec-31 15:23:06 cpeck

Try predicting with the rpart model. We can see it doesn't work very well based on the table, which contains many misclassifications in the predtree variable compared to the actual classe.

```
predtree<-predict(modelfittree,data_train_test_subset)
table(predtree,data_train_test_subset$classe)
```

```
##
## predtree    A    B    C    D    E
##        A 1507  490  498  431  174
##        B   22  282   10  161   62
##        C  139  367  518  372  374
##        D    0    0    0    0    0
##        E    6    0    0    0  472
```

Try random forest model. We can see that this works well based on the confusion matrix, which has very low class errors. The variable importance plot shows that the yaw belt, roll belt, magnet dumbbell z and pitch belt are the most important variables in predicting the classe.

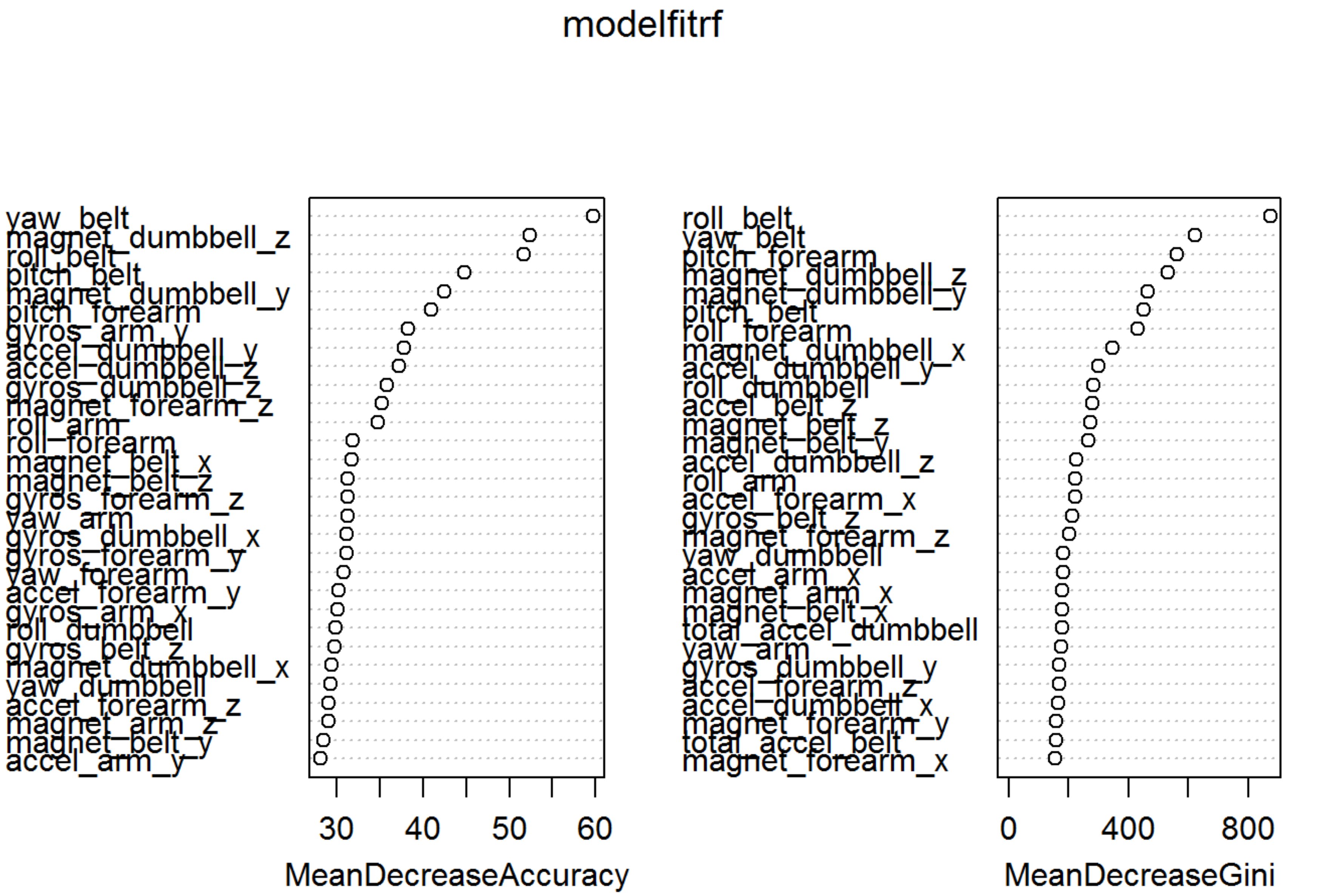
```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
modelfitrf <- randomForest(classe ~ ., data=data_train_train_subset, importance=TRUE)
modelfitrf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = data_train_train_subset,      importanc
e = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.5%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    4    1    0    0 0.001280082
## B   13 2639    6    0    0 0.007148232
## C    0   10 2383    3    0 0.005425710
## D    0    0   22 2228    2 0.010657194
## E    0    0    2    5 2518 0.002772277
```

```
varImpPlot(modelfitrf)
```



Try predicting with the random forest on the training test data. We can see it works well as there are very few predictions that don’t match the actual classe.

```
predrf<-predict(modelfitrf,data_train_test_subset)
predtab<-table(predrf,data_train_test_subset$classe)
confusionMatrix(predtab)
```

```
## Confusion Matrix and Statistics
##
##
## predrf    A    B    C    D    E
##      A 1672    3    0    0    0
##      B    0 1134    1    0    0
##      C    2    2 1024   14    2
##      D    0    0    1  949    0
##      E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9956   0.9981   0.9844   0.9982
## Specificity            0.9993   0.9998   0.9959   0.9998   0.9998
## Pos Pred Value         0.9982   0.9991   0.9808   0.9989   0.9991
## Neg Pred Value         0.9995   0.9989   0.9996   0.9970   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1927   0.1740   0.1613   0.1835
## Detection Prevalence   0.2846   0.1929   0.1774   0.1614   0.1837
## Balanced Accuracy      0.9990   0.9977   0.9970   0.9921   0.9990
```

Expected out of sample error

From the confusion matrix above, we can see that the out of sample error is very low based on the accuracy of 99.5%. Out of sample error is calculated as 1 - accuracy.

Predict on the test data

We apply the random forest model to predict the classe of the 20 test observations

```
predrftest<-predict(modelfitrf,testdata)
predrftest
```

```
##  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
```

```
table(predrftest)
```

```
## predrftest
## A B C D E
## 7 8 1 1 3
```