Quantify how well they do it rather than how much

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 study, we explore how well they do it. We 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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har

Here are the phases of the study:

1. Processing the data.
2. Training Phase - fit a model.
3. Testing Phase - predicting the outcomes.

Data Processing

First, dowload train and test data. Second, pick the relevant columns without NAs.

Here is the code:

```
inData <- read.csv("./pml-training.csv")
tidyInData <- inData[,c(7:11,37:49,60:68,84:86,102,113:124,140,151:160)]
testing <- read.csv("./pml-testing.csv")
testing <- testing[,c(7:11,37:49,60:68,84:86,102,113:124,140,151:160)]
```

Phase 2

Creating training and validation data.

```
even_indexes<-seq(2,19622,2)
odd_indexes<-seq(1,19621,2)
training <- data.frame(tidyInData[odd_indexes,])
validation <- data.frame(tidyInData[even_indexes,])
```

Phase 3

Now, we will fit a model and validates it using the validation data. We will start using Rain-Forest method.

```
library(caret)
library(randomForest)
set.seed(62433)

modRf <- randomForest(classe ~ ., data = training)
predRf <- predict(modRf, validation)
confusionMatrix(predRf,validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2790    3    0    0    0
##          B    0 1894    3    0    0
##          C    0    1 1708   11    0
##          D    0    0    0 1597    3
##          E    0    0    0    0 1801
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9979
##                  95% CI : (0.9967, 0.9987)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9973
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9982   0.9932   0.9983
## Specificity            0.9996   0.9996   0.9985   0.9996   1.0000
## Pos Pred Value         0.9989   0.9984   0.9930   0.9981   1.0000
## Neg Pred Value         1.0000   0.9995   0.9996   0.9987   0.9996
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1839
## Detection Rate         0.2844   0.1930   0.1741   0.1628   0.1836
## Detection Prevalence   0.2847   0.1934   0.1753   0.1631   0.1836
## Balanced Accuracy      0.9998   0.9988   0.9984   0.9964   0.9992
```

It can be observed that the accuracy is 99.7% and the expected out-of-sample error is 0.3%. These results are excellent. Therefore, we will not examine other models now. We will continue to predict the test data. We will use other models or combine a few of them togther if the predictions will be poor. Moreover, we will not use any cross-validation processing behind those done inside the Random-Forest mehod unless our predictions will not be well enoght.

Now, we will test our model against the test data. Therefore, we will expand the test data by creating five observation from each observation by adding a classe variable, one from A thru E. Then, we will predict which one of the five clasee has better match. Accuracy

of 0 will indicates no matching, while 1 will indicates a match.

```
class <- c("A","B","C","D","E")
testing2 <- testing[i,]
testing2$classe <- as.factor("A")
for (i in 1:nrow(testing)) {
   t <- testing[i,]
   for (j in 1:5){
        t$classe <- as.factor(class[j])
        testing2 <- rbind(testing2, t)
   }

}
testing2 <- testing2[2:101,]

for (i in 1:(nrow(testing2))) {
        predRf <- predict(modRf, testing2[i,])
        print(testing2$classe[i])
        print(confusionMatrix(predRf,testing2$classe[i])$overall['Accuracy'])
        if (i%%5==0)print("======")
}
```

```
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
```

```
## Accuracy
##       1
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
## Levels: A B C D E
## Accuracy
##       0
## [1] E
## Levels: A B C D E
## Accuracy
##       0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##       1
## [1] B
## Levels: A B C D E
## Accuracy
##       0
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
```

```
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
## Accuracy
##        1
## [1] C
## Levels: A B C D E
## Accuracy
##        0
## [1] D
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
```

```
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##       1
## [1] B
## Levels: A B C D E
## Accuracy
##       0
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
## Levels: A B C D E
## Accuracy
##       0
## [1] E
## Levels: A B C D E
## Accuracy
##       0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##       1
## [1] B
## Levels: A B C D E
```

```
## Accuracy
##       0
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
## Levels: A B C D E
## Accuracy
##       0
## [1] E
## Levels: A B C D E
## Accuracy
##       0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##       0
## [1] B
## Levels: A B C D E
## Accuracy
##       0
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
```

```
## Levels: A B C D E
## Accuracy
##       0
## [1] E
## Levels: A B C D E
## Accuracy
##       1
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##       0
## [1] B
## Levels: A B C D E
## Accuracy
##       0
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
## Levels: A B C D E
## Accuracy
##       1
## [1] E
## Levels: A B C D E
## Accuracy
##       0
```

```
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
## Accuracy
##        1
## [1] C
## Levels: A B C D E
## Accuracy
##        0
## [1] D
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        1
## [1] B
## Levels: A B C D E
```

```
## Accuracy
##       0
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
## Levels: A B C D E
## Accuracy
##       0
## [1] E
## Levels: A B C D E
## Accuracy
##       0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##       1
## [1] B
## Levels: A B C D E
## Accuracy
##       0
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
```

```
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
## Accuracy
##        1
## [1] C
## Levels: A B C D E
## Accuracy
##        0
## [1] D
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
```

```
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
## Accuracy
##        0
## [1] C
## Levels: A B C D E
## Accuracy
##        1
## [1] D
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
```

```
## Accuracy
##       1
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
## Levels: A B C D E
## Accuracy
##       0
## [1] E
## Levels: A B C D E
## Accuracy
##       0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##       1
## [1] B
## Levels: A B C D E
## Accuracy
##       0
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
```

```
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
## Accuracy
##        0
## [1] C
## Levels: A B C D E
## Accuracy
##        0
## [1] D
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        1
```

```
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
## Accuracy
##        0
## [1] C
## Levels: A B C D E
## Accuracy
##        0
## [1] D
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        1
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        1
## [1] B
## Levels: A B C D E
```

```
## Accuracy
##       0
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
## Levels: A B C D E
## Accuracy
##       0
## [1] E
## Levels: A B C D E
## Accuracy
##       0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##       0
## [1] B
## Levels: A B C D E
## Accuracy
##       1
## [1] C
## Levels: A B C D E
## Accuracy
##       0
## [1] D
```

```
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
## Accuracy
##        1
## [1] C
## Levels: A B C D E
## Accuracy
##        0
## [1] D
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
```

```
## [1] "======"
## [1] A
## Levels: A B C D E
## Accuracy
##        0
## [1] B
## Levels: A B C D E
## Accuracy
##        1
## [1] C
## Levels: A B C D E
## Accuracy
##        0
## [1] D
## Levels: A B C D E
## Accuracy
##        0
## [1] E
## Levels: A B C D E
## Accuracy
##        0
## [1] "======"
```

After thought, there is a simply way to do that:

```
predict(modRf, testing)
```

```
##  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
```

These results achieve 20/20 matches.