Practical Machine Learning - Write-Up Project

Carlos Alberto Guevara Díez

November, 2015

The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants and predict the manner in which people sis the exercise, to be more specific I´m going to predict the classe variable of the data sets, for further information about the description of the project and data used, please refer to the ReadMe file included in this repo.

Data Pre-Processing and Exploratory Analysis

The first thing to do is to load all the libraries needed to run the functions used in the project.

```
library(caret)
```

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

```
library(ggplot2)
library(corrplot)
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

Once loaded I´m going to load the data into R. Note: If you are going to run the code in your own environment please change the first line of the following chunk:

```
#Adjust this line to your own environment.
setwd("C:/Users/220194/Documents/Data Science Specialization/08 Practical Machine Learning/PracticalMachineLearn
#Read csv files provided
training <- read.csv("pml-training.csv",na.strings = c("NA", ""))
testing <- read.csv("pml-testing.csv",na.strings = c("NA", ""))
```

At this point I need to make some exploratory analysis on the data in order to identify those columns that do not support the analysis because they are non-numerical. Using what I´ve seen now I'm going to remove columns using the grep function an create a new vector. Note: To make the report easier to read I´m hiding the results of the sapply function, please refer to the appendix section (1) of this report to see the results.

```
#Explore the class of the columns
sapply(testing,class)
#Using previous exploration generate newe data sets removing non-numerical columns
training_aux <- training[, -(grep("timestamp|X|user_name|num_window|new_window", names(training)))]
testing_aux <- testing[, -(grep("timestamp|X|user_name|num_window|new_window", names(testing)))]
```

As part of the pre-processing tasks I need to get rid of the NA values, as you know The NA's records make our machine learning algorithm less precise, that´s why is so important to take care of it, once removed I'm updating the vectors I´m going to use in my analysis. Finally I'm going to create my working vectors using a 60%-40% relation between training and testing data sets.

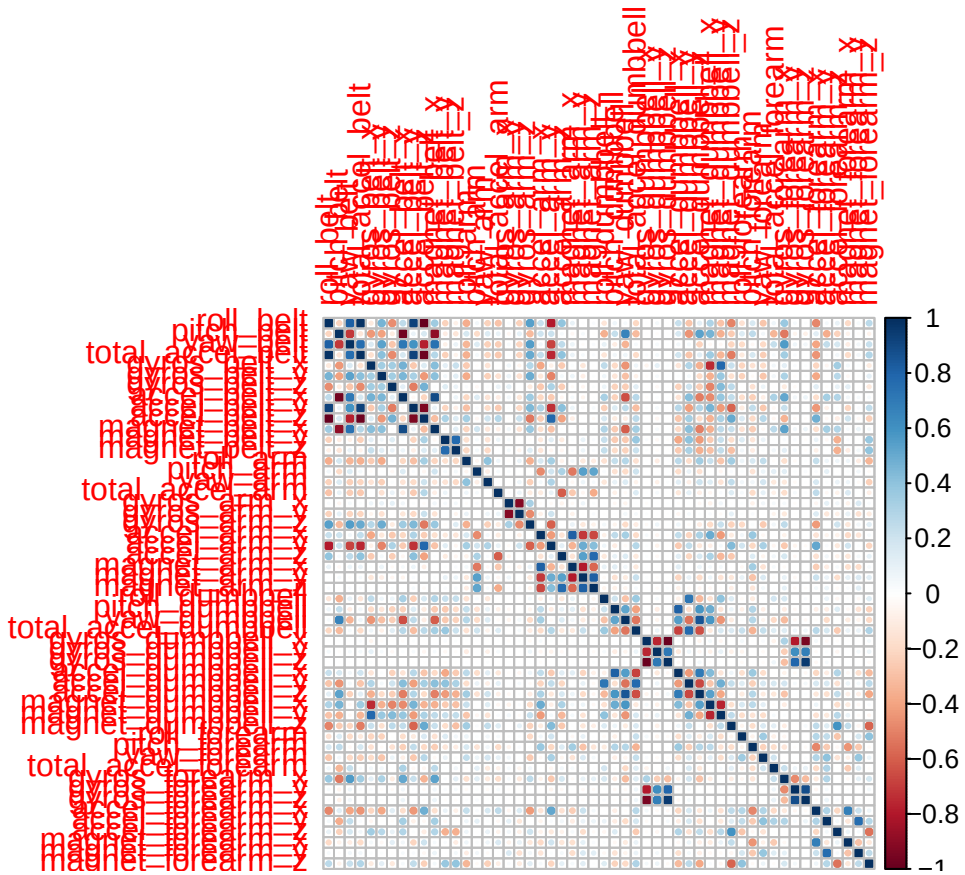
```
#Identify NAs
NAs <- apply(training_aux, 2, function(x) {
    sum(is.na(x))})
#Remove NAs
training_aux <- training_aux[, which(NAs == 0)]
testing_aux <- testing_aux[, which(NAs == 0)]
#Create Working training and testing data sets.
trainingWS <- training_aux[createDataPartition(y = training_aux$classe, p = 0.6, list = FALSE), ]
testingWS <- training_aux[-createDataPartition(y = training_aux$classe, p = 0.6, list = FALSE), ]
```

Now it´s time to look the changes that I've done, once again for more readability of the document I´m hiding the results, please refer to the appendix section (2) of this report to see the results.

```
#Review the changes in the DataSets
dim(trainingWS)
head(trainingWS)
```

To end with the exploratory tasks I´m showing the following graphic, this one is useful to understand the correlation among the variables, please note the code of color in which the dark blue indicates strong correlation and red negative correlation.

```
#Remove NAs and then plot a correlation graphic
Graph <- trainingWS
NAs <- apply(trainingWS, 2, function(x) {
    sum(is.na(x))
})
Graph<- trainingWS[, which(NAs == 0)]
CorrePlot = cor( Graph[,-c(grep("timestamp|X|user_name|num_window|new_window",names(Graph)), length(Graph))])
corrplot(CorrePlot, method="circle",tl.cex=1)
```



Model creation

Now that I have collected enough information to understand the datasets it´s time to build my model using Random Forest.

```
#Set seed for reproduction in other environments
set.seed(10)
#Create Model using trainControl and boot as method
rfModel<- train(trainingWS$classe ~ ., data = trainingWS, method = "rf",
    prof = TRUE, trControl = trainControl(method = "boot", number = 5, allowParallel = TRUE))
#Check for the model, final results and accuracy
rfModel
```

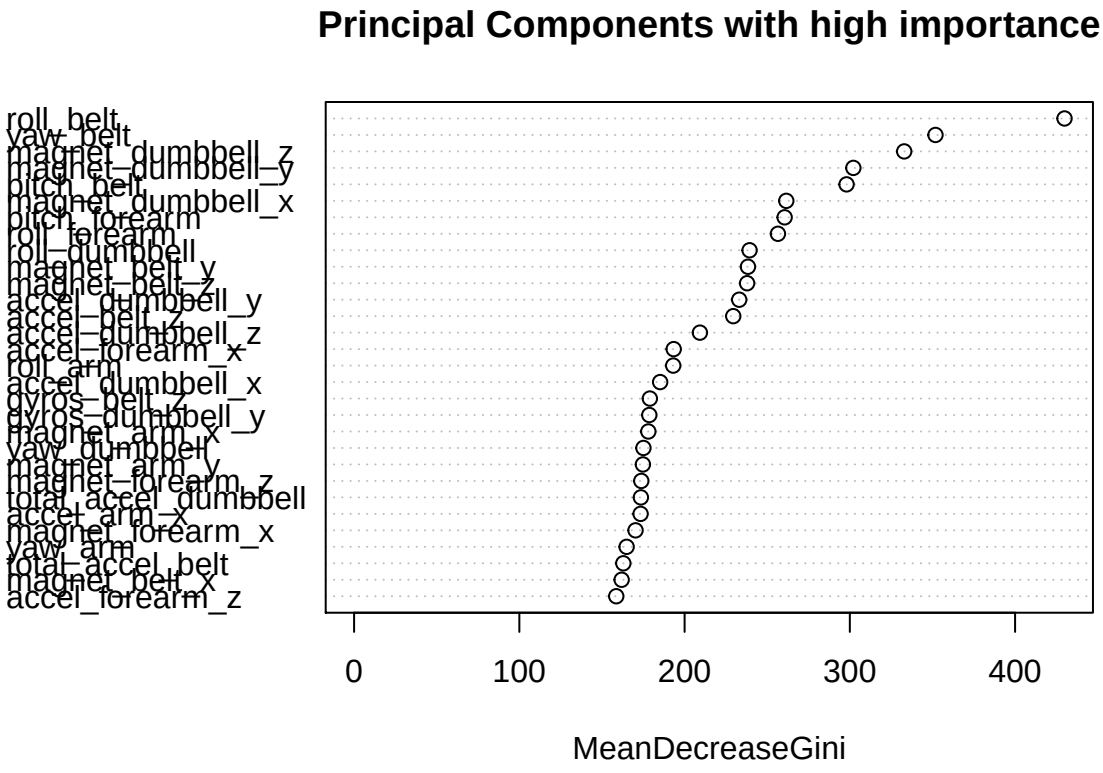
```
## Random Forest
##
## 11776 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (5 reps)
## Summary of sample sizes: 11776, 11776, 11776, 11776, 11776
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9878234  0.9845821  0.003614221  0.004572268
##   27    0.9867009  0.9831599  0.003091003  0.003923590
##   52    0.9763651  0.9700711  0.004059269  0.005164587
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
rfModel.Final<- rfModel$results
round(max(rfModel.Final$Accuracy), 3) * 100
```

```
## [1] 98.8
```

Now that I ran the model it´s possible to say that we have great results, the accuracy of the model is of 98.6 % which is a very nice value. Next I´m plotting a Variable Importance Plot which helps to understand the importance of variables (significance) measured by the Random Forest Model.

```
varImpPlot(rfModel$finalModel,
           main = "Principal Components with high importance")
```



Testing Evaluations

With our model built now it´s time to execute some validation to test the accuracy of our model, which will be the following:

Cross Validation.

Now let´s use cross validation to test accuracy on the testing data set created in the past section of this document, as we can see in the results I also got a high level of accuracy

```
testingWS$predRight <- (predict(rfModel, testingWS))== testingWS$classe
table(predict(rfModel, testingWS), testingWS$classe)
```

```
##
##        A    B    C    D    E
##   A 2232    6    0    0    0
##   B    0 1511    9    0    0
##   C    0    1 1358   11    0
##   D    0    0    1 1275    1
##   E    0    0    0    0 1441
```

```
CrossValidation<- postResample((predict(rfModel, testingWS)), testingWS$classe)
CrossValidation
```

```
##  Accuracy     Kappa
## 0.9963038 0.9953243
```

Confusion Matrix

Now it´s time to test the performance of the model, once again its showed that the results of the predictions are very close with the result of the matrix with accuracy of 99.6 %, all this test results shows that the model was built correctly.

```
#set seed for reproduction in opther environments
set.seed(10)
#build the matrix
CrossValidationError <- confusionMatrix((predict(rfModel, testingWS)), testingWS$classe)
CrossValidationError
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    6    0    0    0
##          B    0 1511    9    0    0
##          C    0    1 1358   11    0
##          D    0    0    1 1275    1
##          E    0    0    0    0 1441
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9947, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9953
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9954   0.9927   0.9914   0.9993
## Specificity            0.9989   0.9986   0.9981   0.9997   1.0000
## Pos Pred Value         0.9973   0.9941   0.9912   0.9984   1.0000
## Neg Pred Value         1.0000   0.9989   0.9985   0.9983   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1926   0.1731   0.1625   0.1837
## Detection Prevalence   0.2852   0.1937   0.1746   0.1628   0.1837
## Balanced Accuracy      0.9995   0.9970   0.9954   0.9956   0.9997
```

```
#Calculate Accuracy
postResample((predict(rfModel, testingWS)), testingWS$classe)[[1]]
```

```
## [1] 0.9963038
```

```
1- postResample((predict(rfModel, testingWS)), testingWS$classe)[[1]]
```

```
## [1] 0.003696151
```

Submission Part of The Project, prediction of the 20 cases

Once that I´ve built and tested my model it´s time to use the 20 cases test set to predict the behavior of the people using the jawbone devices, first I´m going to prepare the function to write the files that I´m going to submit to Coursera.

```
pml_write_files = function(x, directory="solutionfiles"){
  dir.create (directory)
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    filename=file.path(directory, filename)
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
```

Now that the function is written all I have to do is to use the predict function in the Random Forest model built using de TESTING data set provided (20 cases) and create the files, in the "solutionfiles"" directory provided in this repo are the 20 files that contains the prediction of each case provided in the TESTING data set.

```
Model.Prediction <- predict(rfModel, testing)
Model.Prediction
```

```
##  [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
```

```
pml_write_files(Model.Prediction)
```

```
## Warning in dir.create(directory): 'solutionfiles' already exists
```

Final Thoughts

In this assignment I´ve put in practice several concepts reviewed to the entire Data Science Specialization, finally I´ve put into practice them and understood the relation among training, testing and prediction datasets. This project really gave me a complete view of the data science goal and also gave a lot of ideas to put in practice in the real world, I really hope that you enjoy reading this report as much as I've enjoy it writing it, and of course that may help you to understand your own concepts and applications.

Apendix

Section (1) sapply function to explore the data

```
#Explore the class of the columns
sapply(testing,class)
```

```
##                        X                user_name     raw_timestamp_part_1
##                "integer"                 "factor"                "integer"
##     raw_timestamp_part_2           cvtd_timestamp               new_window
##                "integer"                 "factor"                 "factor"
##               num_window                roll_belt               pitch_belt
##                "integer"                "numeric"                "numeric"
##                 yaw_belt         total_accel_belt       kurtosis_roll_belt
##                "numeric"                "integer"                "logical"
##      kurtosis_picth_belt        kurtosis_yaw_belt       skewness_roll_belt
##                "logical"                "logical"                "logical"
##     skewness_roll_belt.1        skewness_yaw_belt            max_roll_belt
##                "logical"                "logical"                "logical"
##           max_picth_belt             max_yaw_belt            min_roll_belt
##                "logical"                "logical"                "logical"
##           min_pitch_belt             min_yaw_belt      amplitude_roll_belt
##                "logical"                "logical"                "logical"
##     amplitude_pitch_belt       amplitude_yaw_belt     var_total_accel_belt
##                "logical"                "logical"                "logical"
##            avg_roll_belt         stddev_roll_belt            var_roll_belt
##                "logical"                "logical"                "logical"
##           avg_pitch_belt        stddev_pitch_belt           var_pitch_belt
##                "logical"                "logical"                "logical"
##             avg_yaw_belt          stddev_yaw_belt             var_yaw_belt
##                "logical"                "logical"                "logical"
##             gyros_belt_x             gyros_belt_y             gyros_belt_z
##                "numeric"                "numeric"                "numeric"
##             accel_belt_x             accel_belt_y             accel_belt_z
##                "integer"                "integer"                "integer"
##            magnet_belt_x            magnet_belt_y            magnet_belt_z
##                "integer"                "integer"                "integer"
##                 roll_arm                pitch_arm                  yaw_arm
##                "numeric"                "numeric"                "numeric"
##          total_accel_arm            var_accel_arm            avg_roll_arm
##                "integer"                "logical"                "logical"
##          stddev_roll_arm             var_roll_arm            avg_pitch_arm
##                "logical"                "logical"                "logical"
##         stddev_pitch_arm            var_pitch_arm              avg_yaw_arm
##                "logical"                "logical"                "logical"
##           stddev_yaw_arm             var_yaw_arm             gyros_arm_x
##                "logical"                "logical"                "numeric"
##              gyros_arm_y             gyros_arm_z             accel_arm_x
##                "numeric"                "numeric"                "integer"
##              accel_arm_y             accel_arm_z            magnet_arm_x
##                "integer"                "integer"                "integer"
##             magnet_arm_y            magnet_arm_z       kurtosis_roll_arm
##                "integer"                "integer"                "logical"
##        kurtosis_picth_arm         kurtosis_yaw_arm        skewness_roll_arm
##                "logical"                "logical"                "logical"
##        skewness_pitch_arm         skewness_yaw_arm             max_roll_arm
##                "logical"                "logical"                "logical"
##             max_picth_arm              max_yaw_arm             min_roll_arm
##                "logical"                "logical"                "logical"
```

```
##            min_pitch_arm             min_yaw_arm       amplitude_roll_arm
##                "logical"               "logical"               "logical"
##      amplitude_pitch_arm       amplitude_yaw_arm            roll_dumbbell
##                "logical"               "logical"               "numeric"
##           pitch_dumbbell             yaw_dumbbell   kurtosis_roll_dumbbell
##                "numeric"               "numeric"               "logical"
##  kurtosis_picth_dumbbell   kurtosis_yaw_dumbbell   skewness_roll_dumbbell
##                "logical"               "logical"               "logical"
##  skewness_pitch_dumbbell   skewness_yaw_dumbbell        max_roll_dumbbell
##                "logical"               "logical"               "logical"
##       max_picth_dumbbell        max_yaw_dumbbell        min_roll_dumbbell
##                "logical"               "logical"               "logical"
##       min_pitch_dumbbell        min_yaw_dumbbell  amplitude_roll_dumbbell
##                "logical"               "logical"               "logical"
## amplitude_pitch_dumbbell  amplitude_yaw_dumbbell     total_accel_dumbbell
##                "logical"               "logical"               "integer"
##       var_accel_dumbbell        avg_roll_dumbbell     stddev_roll_dumbbell
##                "logical"               "logical"               "logical"
##        var_roll_dumbbell       avg_pitch_dumbbell    stddev_pitch_dumbbell
##                "logical"               "logical"               "logical"
##       var_pitch_dumbbell        avg_yaw_dumbbell      stddev_yaw_dumbbell
##                "logical"               "logical"               "logical"
##         var_yaw_dumbbell        gyros_dumbbell_x        gyros_dumbbell_y
##                "logical"               "numeric"               "numeric"
##         gyros_dumbbell_z        accel_dumbbell_x        accel_dumbbell_y
##                "numeric"               "integer"               "integer"
##         accel_dumbbell_z       magnet_dumbbell_x       magnet_dumbbell_y
##                "integer"               "integer"               "integer"
##        magnet_dumbbell_z            roll_forearm           pitch_forearm
##                "integer"               "numeric"               "numeric"
##             yaw_forearm   kurtosis_roll_forearm   kurtosis_picth_forearm
##                "numeric"               "logical"               "logical"
##     kurtosis_yaw_forearm   skewness_roll_forearm   skewness_pitch_forearm
##                "logical"               "logical"               "logical"
##     skewness_yaw_forearm        max_roll_forearm       max_picth_forearm
##                "logical"               "logical"               "logical"
##          max_yaw_forearm        min_roll_forearm       min_pitch_forearm
##                "logical"               "logical"               "logical"
##          min_yaw_forearm  amplitude_roll_forearm  amplitude_pitch_forearm
##                "logical"               "logical"               "logical"
##    amplitude_yaw_forearm     total_accel_forearm       var_accel_forearm
##                "logical"               "integer"               "logical"
##         avg_roll_forearm     stddev_roll_forearm        var_roll_forearm
##                "logical"               "logical"               "logical"
##        avg_pitch_forearm    stddev_pitch_forearm       var_pitch_forearm
##                "logical"               "logical"               "logical"
##         avg_yaw_forearm      stddev_yaw_forearm         var_yaw_forearm
##                "logical"               "logical"               "logical"
##         gyros_forearm_x         gyros_forearm_y         gyros_forearm_z
##                "numeric"               "numeric"               "numeric"
##         accel_forearm_x         accel_forearm_y         accel_forearm_z
##                "integer"               "integer"               "integer"
##        magnet_forearm_x        magnet_forearm_y        magnet_forearm_z
##                "integer"               "integer"               "integer"
##              problem_id
##                "integer"
```

```
#Using previous exploration generate newe data sets removing non-numerical columns
training_aux <- training[, -(grep("timestamp|X|user_name|num_window|new_window", names(training)))]
testing_aux <- testing[, -(grep("timestamp|X|user_name|num_window|new_window", names(testing)))]
```

Section (2) Review working data sets without NAs

```
#Review the changes in the DataSets
dim(trainingWS)
```

```
## [1] 11776    53
```

```
head(trainingWS)
```

```
##   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
## 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
## 7      1.42       8.09    -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
## 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
## 7        -0.02          -22            3           21            -4
##   magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm total_accel_arm
## 1           599          -313     -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
## 7           599          -311     -128      21.9    -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
## 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
## 7        0.00       -0.03        0.00        -289         111        -125
##   magnet_arm_x magnet_arm_y magnet_arm_z roll_dumbbell pitch_dumbbell
## 1         -368          337          516      13.05217      -70.49400
## 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
## 7         -373          336          509      13.12695      -70.24757
##   yaw_dumbbell total_accel_dumbbell gyros_dumbbell_x gyros_dumbbell_y
## 1    -84.87394                   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
## 7    -85.09961                   37                0            -0.02
##   gyros_dumbbell_z accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z
## 1             0.00             -234               47             -271
## 3             0.00             -232               46             -270
## 4            -0.02             -232               48             -269
## 5             0.00             -233               48             -270
## 6             0.00             -234               48             -269
## 7             0.00             -232               47             -270
##   magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z roll_forearm
```

```
## 1              -559              293               -65         28.4
## 3              -561              298               -63         28.3
## 4              -552              303               -60         28.1
## 5              -554              292               -68         28.0
## 6              -558              294               -66         27.9
## 7              -551              295               -70         27.9
##   pitch_forearm yaw_forearm total_accel_forearm gyros_forearm_x
## 1         -63.9       -153                  36            0.03
## 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
## 7         -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
## 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
## 7            0.00           -0.02             195             205
##   accel_forearm_z magnet_forearm_x magnet_forearm_y magnet_forearm_z
## 1            -215              -17              654              476
## 3            -213              -18              658              469
## 4            -214              -16              658              469
## 5            -214              -17              655              473
## 6            -215               -9              660              478
## 7            -215              -18              659              470
##   classe
## 1      A
## 3      A
## 4      A
## 5      A
## 6      A
## 7      A
```