Objective of the Project

1. To read several files from an online Source related to experiments with a group of 30 volunteers. The files give the average measurements based on 6 different types of activities and a range of features associated with motion.
2. Reshape and transform the data to create a ‘Tidy’ Data as per best practice.
3. Create 2 output files that satisfy the project requirement.
   1. A comprehensive Tidy data set of test and train data
   2. A average of ‘Mean’ and STD metrics by Subject and Activity.

Data Source:

About the data

Human Activity Recognition Using Smartphones Dataset

Version 1.0

==================================================================

Jorge L. Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto.

Smartlab - Non Linear Complex Systems Laboratory

DITEN - Universit‡ degli Studi di Genova.

Via Opera Pia 11A, I-16145, Genoa, Italy.

activityrecognition@smartlab.ws

www.smartlab.ws

The list of files available:

'features\_info.txt': Shows information about the variables used on the feature vector.

-1’features.txt': List of all features.

-2 'activity\_labels.txt': Links the class labels with their activity name.

-3 'train/X\_train.txt': Training set.

-4 'train/y\_train.txt': Training labels.

-5’test/X\_test.txt': Test set.

-6 test/y\_test.txt': Test labels.

The following files are available for the train and test data. Their descriptions are equivalent.

-7 'train/subject\_train.txt': Each row identifies the subject who performed the activity for each window sample. Its range is from 1 to 30.

- 8 ’train/Inertial Signals/total\_acc\_x\_train.txt': The acceleration signal from the smartphone accelerometer X axis in standard gravity units 'g'. Every row shows a 128 element vector. The same description applies for the 'total\_acc\_x\_train.txt' and 'total\_acc\_z\_train.txt' files for the Y and Z axis.

- 9 ’train/Inertial Signals/body\_acc\_x\_train.txt': The body acceleration signal obtained by subtracting the gravity from the total acceleration.

-10 'train/Inertial Signals/body\_gyro\_x\_train.txt': The angular velocity vector measured by the gyroscope for each window sample. The units are radians/second.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CODE Design Summary**

**Approach**

After reading in and examining the dimension I chose the following approach to code

1. Append row labels or key/ ID columns to the individual Train or test data

2. Set up the formats for the activity labels using gsub function and the filter index using Grep function

2a: we undertake the transformation of the activity description labels to make them more reader friendly

3. Create a new column to identify the source - Train or Test. While this was not required, I have added it to enable reconstructing the original data.

4. Append the test and train data using row bind.

5. Apply the filter index to subset the measures to those with STD or MEAN.

6. Update the column names

7. Replace the activity ID with activity labels

8. Create a new data set with Average of each variable for each activity - using the dplyer commands: Melt and Dcast

9. Write the two files to Txt formats with space seperators.

**Reading in the Files**

The Files 1- 10 were read in and their dimensions and structure understood using

Dim() and Str(). In cases of factor and character strings Table() were used to see the distributions.

Here were the preliminary results:

**File1:** features = read.table("features.txt", col.names= c(“FeatureNum","Feature"),stringsAsFactors=FALSE)# Dim(561, 2)

FeatureNum FeatureNm Description of Measured Features

1 1 tBodyAcc-mean()-X Mean of time based body acceleration along X axis

2 2 tBodyAcc-mean()-Y

3 3 tBodyAcc-mean()-Z

4 4 tBodyAcc-std()-X

5 5 tBodyAcc-std()-Y Std Deviation of time based body acceleration along Y axis

6 6 tBodyAcc-std()-Z

Feature Selection

=================

The features selected for this database come from the accelerometer and gyroscope 3-axial raw signals tAcc-XYZ and tGyro-XYZ. These time domain signals (prefix 't' to denote time) were captured at a constant rate of 50 Hz. Then they were filtered using a median filter and a 3rd order low pass Butterworth filter with a corner frequency of 20 Hz to remove noise. Similarly, the acceleration signal was then separated into body and gravity acceleration signals (tBodyAcc-XYZ and tGravityAcc-XYZ) using another low pass Butterworth filter with a corner frequency of 0.3 Hz.

Subsequently, the body linear acceleration and angular velocity were derived in time to obtain Jerk signals (tBodyAccJerk-XYZ and tBodyGyroJerk-XYZ). Also the magnitude of these three-dimensional signals were calculated using the Euclidean norm (tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag, tBodyGyroJerkMag).

Finally a Fast Fourier Transform (FFT) was applied to some of these signals producing fBodyAcc-XYZ, fBodyAccJerk-XYZ, fBodyGyro-XYZ, fBodyAccJerkMag, fBodyGyroMag, fBodyGyroJerkMag. (Note the 'f' to indicate frequency domain signals).

These signals were used to estimate variables of the feature vector for each pattern:

'-XYZ' is used to denote 3-axial signals in the X, Y and Z directions.

tBodyAcc-XYZ

tGravityAcc-XYZ

tBodyAccJerk-XYZ

tBodyGyro-XYZ

tBodyGyroJerk-XYZ

tBodyAccMag

tGravityAccMag

tBodyAccJerkMag

tBodyGyroMag

tBodyGyroJerkMag

fBodyAcc-XYZ

fBodyAccJerk-XYZ

fBodyGyro-XYZ

fBodyAccMag

fBodyAccJerkMag

fBodyGyroMag

fBodyGyroJerkMag

The set of variables that were estimated from these signals are:

mean(): Mean value

std(): Standard deviation

mad(): Median absolute deviation

max(): Largest value in array

min(): Smallest value in array

sma(): Signal magnitude area

energy(): Energy measure. Sum of the squares divided by the number of values.

iqr(): Interquartile range

entropy(): Signal entropy

arCoeff(): Autorregresion coefficients with Burg order equal to 4

correlation(): correlation coefficient between two signals

maxInds(): index of the frequency component with largest magnitude

meanFreq(): Weighted average of the frequency components to obtain a mean frequency

skewness(): skewness of the frequency domain signal

kurtosis(): kurtosis of the frequency domain signal

bandsEnergy(): Energy of a frequency interval within the 64 bins of the FFT of each window.

angle(): Angle between to vectors.

Additional vectors obtained by averaging the signals in a signal window sample. These are used on the angle() variable:

gravityMean

tBodyAccMean

tBodyAccJerkMean

tBodyGyroMean

tBodyGyroJerkMean

The complete list of variables of each feature vector is available in 'features.txt'

**File2:**  activityLabels = read.table("activity\_labels.txt", col.names= c("ActivityID", “Activity”))#DIM (6 ,2)

# ActivityID Activity

1 1 WALKING

2 2 WALKING\_UPSTAIRS

3 3 WALKING\_DOWNSTAIRS

4 4 SITTING

5 5 STANDING

6 6 LAYING

**File 3:** Xtrain = read.table("X\_train.txt") # Dim(7352,561)

Individual subject readings by activity type across 561 feature measures as defined in Features.txt .

**File 4:** trainLabel = read.table("y\_train.txt",col.names="ActivityType") # Dim(7352,1)

Row labels providing the Activity vector that goes with each of the Xtrain recording.

**File5:**  Xtest = read.table(“X\_test.txt") # Dim(2947, 561)

Note: Default col names V1...v561 will require replacement using Features vector.

'data.frame': 2947 obs. of 561 variables:

$ V1 : num 0.257 0.286 0.275 0.27 0.275 ...

$ V2 : num -0.0233 -0.0132 -0.0261 -0.0326 -0.0278 ...

$ V3 : num -0.0147 -0.1191 -0.1182 -0.1175 -0.1295

**File6:**  testLabel = read.table("y\_test.txt",col.names="ActivityType")

Dim(2947,1) .Values from 1:6

testLabel distribution

1 2 3 4 5 6

496 471 420 491 532 537

**File7a:** subjecTest = read.table("subject\_test.txt", col.names=“SubjectActivity")

The subject vector (2947,1)

table() = 2 4 9 10 12 13 18 20 24

302 317 288 294 320 327 364 354 381

'data.frame': 2947 obs. of 1 variable:

$ SubjectActivity: int 2 2 2 2 2 2 2 2 2 2 ...

Comment: there are 30 subjects, but only 9 have recorded measures and features in the table as seen from file subjecTest.

**File7b:** subjecTrain = read.table("subject\_train.txt", col.names="SubjectActivity")

# Dim(7352,1) should range from 1- 30 subjects but not all present in the training set table below.

# 1 3 5 6 7 8 11 14 15 16 17 19 21 22 23 25 26 27 28 29 30

# 347 341 302 325 308 281 316 323 328 366 368 360 408 321 372 409 392 376 382 344 383

Remaining Files were not used in this Project.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Background about the Data Source**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

About the data

Human Activity Recognition Using Smartphones Dataset

Version 1.0

==================================================================

Jorge L. Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto.

Smartlab - Non Linear Complex Systems Laboratory

DITEN - Universit‡ degli Studi di Genova.

Via Opera Pia 11A, I-16145, Genoa, Italy.

activityrecognition@smartlab.ws

www.smartlab.ws

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING\_UPSTAIRS, WALKING\_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain. See 'features\_info.txt' for more details.

**CODE Design**

*# libraries that will be used in accomplishing the tasks*

library(jsonlite)

library(curl)

library(gzip)

require(curl)

library(reshape2)

library(dplyr)

***#1 connect to the zipped file and download data, then close connection***

fileurl ="https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20HAR%20Dataset.zip"

download.file(fileurl, destfile="projectfile.zip", method="curl")

zipfile ="projectfile.zip"

unzip(zipfile, compressed = 'gzip', exdir="./Rclass/DataClass/Project")

topdir= "/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset"

list.files(topdir) *# checking for the unzipped files within the folder*

***#2 read in the Common data files File 1, File 2 from ReadMe.doc***

dir\_act="/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/activity\_labels.txt"

dir\_fea="/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/features.txt"

activityLabels = read.table(dir\_act, col.names= c("ActivityID", "Activity"))

Features = read.table(dir\_fea, col.names= c("FeatureNum","Feature"),stringsAsFactors=FALSE)

*#Do not want the character variables to be treated as factors so use stringsAsFactors*

*#Reformat Features set up is applied to create the final tidy data variable selection.*

*# Selecting Features associated with a 'mean' or a 'std' measure. In addition reformating the feature names.*

Feature\_reformat= function(cols) {

cols= gsub("-mean\\(\\)",".Mean",cols)

cols=gsub ("-std\\(\\)", ".Std",cols)

cols=gsub ("-Y", ".Y",cols)

cols=gsub ("-X", ".X",cols)

cols=gsub ("-Z", ".Z",cols)

}

***#3 Reading "test" Files 5,6, 7b from ReadMe.doc***

test\_sub="/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/test/subject\_test.txt"

test\_x="/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/test/X\_test.txt"

test\_y="/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/test/y\_test.txt"

subjecTest = read.table(test\_sub, col.names="Subject")

*# Subjects range between 1-30 refer to the experiment participants*

Xtest = read.table(test\_x)

*# matrix table of measures of the 561 features, for Test subjects, across 6 different* *activities*

testLabel = read.table(test\_y,col.names="ActivityType")

*# activity corresponding to row in Xtest*

Sample= c("Test") *# created an additional factor variable to ensure that we can distinguish test and train sources.*

*# appended to each row of the Test Data Table.*

XYtest\_subject = cbind(Sample,testLabel,subjecTest,Xtest)

*# 2947 x (561 +1 +1) Created a data frame for the test sample.*

***#4 Reading the Train Files 3,4,7a from ReadMe.doc***

train\_sub="/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/train/subject\_train.txt"

train\_x="/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/train/X\_train.txt"

train\_y="/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/train/y\_train.txt"

subjecTrain = read.table(train\_sub, col.names="Subject")  *# Dim[7352,1]*

Xtrain = read.table(train\_x)  *# Dim[7352,561]*

trainLabel = read.table(train\_y,col.names="ActivityType")  *# Dim[7352,1]*

*# add test and train flag before adding the data together - best practice for model building exercises.*

Sample= c("Train")

XYtrain\_subject = cbind(Sample,trainLabel,subjecTrain,Xtrain)  *#Dim[7352,1+2+ 561]*

***#4 Check dimensions, nrows, tables or summary to understand the components of each table noted against each file read***

# *used nrow(filename), head(filename), tail(filename) table() to validate each file structure and distribution*

***#5 Merge Test and train***

Test\_Train = rbind(XYtest\_subject,XYtrain\_subject)  *# Dim(10299x 564)*

subTestTrain= Test\_Train[, 4:564] *# subsetting the last 561* variables to apply the mean and std subset

names(subTestTrain)=NULL #*set default labels V1:V561 to null*

*# creating an Index vector to subset the required columns*

*# the filter index vector is created using pattern match 'grep()'*

automatic= grep("std\\(|mean\\(", Features$Feature)

subTestTrain=subTestTrain[ ,automatic]

***#6 Applying the Mean and Std filter and description reformats. Project Requirement #2***

*# Reduced dimensionality from 564 columns to Dim(10299x 66)*

names(subTestTrain) =

Feature\_reformat(Features$Feature[grep("std\\(|mean\\(", Features$Feature)])

TidyData= cbind(Test\_Train[, 1:3],subTestTrain)

*# TidyData: Dimension returned to 10299x 69*

*#( In the discussion forum the typical number of columns is 68 - I have an extra column for retaining the ability to distinguish Train vs Test records.)*

***#7 Replacing activity names in the dataset - Project Requirement #3 & #4***

add\_TidyLabel= merge(TidyData,activityLabels,by.x="ActivityType",by.y="ActivityID")

add\_TidyLabel$Activity=gsub("(\\w)(\\w\*)","\\U\\1\\L\\2",add\_TidyLabel$Activity,perl=TRUE )  *# creating a mixed case activity description*

TidyData= "/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/TidyData.txt"

write.table(add\_TidyLabel,TidyData,sep="\t", row.names=FALSE)

***#8 calculating Average values by activities - Project Requirement #5***

names(add\_TidyLabel)=tolower(names(add\_TidyLabel)) *#dim(add\_TidyLabel) (10299,70)*

*# Mixed case colonnades are not accepted by melt procedure so revert to lower case*

Tidytemp= melt(add\_TidyLabel,id=c("sample","subject","activitytype","activity"),na.rm=TRUE)

*#add\_TidyLabel - Dim(679734,60)*

*# I have used the Wide format to create the final TIDY data,*

*# since the focus of the analysis is the 30 Subjects and the activities that the subjects have been* measured against.

*# Visually it is easy to read across and use it to create graphs by column selection.*

TidyAvg= dcast(Tidytemp,subject+activity ~ variable, mean)

*# Dim(180,68) Variables ( retained activitytype, sample )*

***#9 writing out Text file Project Requirement #5***

TidyAvg= "/Users/priyasarathy/Documents/Rclass/DataClass/UCI HAR Dataset/TidyAvg.txt"

write.table(TidyAvg,TidyAvg,sep="\t", row.names=FALSE)