Human Activity Recognition Using Smartphones Dataset

Peter Vennel

Monday, January 19, 2015

I have renamed the original data directory from **UCI HAR Dataset** to **UCI_HAR**. This is because on some operating systems, spaces in file names and directory names can cause issues.

The script **run_analysis.R** has to be copied in the same directory where **UCI_HAR** exist. This is so that we do not combine code and data in same folder.Helps easy management of versions control of code, by keeping data and code separate.

The original data files that were used for the analysis is as follows:

- UCI_HAR/activity_labels.txt
- UCI_HAR/test/subject_test.txt
- UCI_HAR/test/X_test.txt
- UCI_HAR/test/y_test.txt
- UCI_HAR/train/subject_train.txt
- UCI_HAR/train/X_train.txt
- UCI_HAR/train/y_train.txt

Task#1 Merges the training and the test sets to create one data set

R Packages I used the following R packages

```
library("dplyr", lib.loc="~/R/win-library/3.1")
library("data.table", lib.loc="~/R/win-library/3.1")
library("stringr", lib.loc="~/R/win-library/3.1")
library("plyr", lib.loc="~/R/win-library/3.1")
library("tidyr", lib.loc="~/R/win-library/3.1")
```

UCI_HAR/features.txt file had some characters in the feature names, that would cause error when used in R code. I cleaned up the file by making some changes and creating a new file **UCI_HAR/features-clean.txt**

I made the following changes in NotePad++

1. removed "()" with ""
2. replaced "(" with "_"
3. replaced "," with "-"

This was done directly using the fins and replace command of the editor.

After the changes it should be noted that, "__" signify function and "**-**" signifies separation between function arguments.

I created a table of the features from the cleaned file from above.Since the file did not have column names, I set the header to FALSE.

```
featuresDF <- read.table("UCI_HAR/features-clean.txt", header=F)
```

After reading the file **UCI_HAR/activity_labels.txt**, I had to added the column names **ACTIVITY_ID** and **ACTIVITY_DESC** to add clarity.

```
activity <- read.table("UCI_HAR/activity_labels.txt")
names(activity) <- c("ACTIVITY_ID","ACTIVITY_DESC")
```

I created the test subject DataFrame and training subject Data Frame

```
subject_test <- read.csv("UCI_HAR/test/subject_test.txt", header=F)
subject_train <- read.csv("UCI_HAR/train/subject_train.txt", header=F)
```

I added the column name **SUBJECT_ID** to both the Data Frames for clarity.

```
names(subject_test) <- c("SUBJECT_ID")
names(subject_train) <- c("SUBJECT_ID")
```

I created the **Test** DataFrames. Since both the files did not have column names, I set the header to FALSE. *X_test.txt* contained the Test set. *y_test.txt* contains the Test labels

```
x_data_test <- read.table("UCI_HAR/test/X_test.txt", header=F)
y_data_test <- read.table("UCI_HAR/test/y_test.txt", header=F)
```

Now since both the files did not have the column names, I had to explicitly add column names to the files.

```
names(y_data_test) <- c("ACTIVITY_ID")
names(x_data_test) <- featuresDF$V2
```

I then performed the same steps to get the **Training** data.

```
x_data_train <- read.table("UCI_HAR/train/X_train.txt", header=F)
y_data_train <- read.table("UCI_HAR/train/y_train.txt", header=F)
names(y_data_train) <- c("ACTIVITY_ID")
names(x_data_train) <- featuresDF$V2
```

My next step is to combine the respective Test and Training Data Frames.This can be easliy accomplished using by appending the row of similar files using rbind().

```
subjectDF <- rbind(subject_test, subject_train)
y_dataDF <- rbind(y_data_test, y_data_train)
x_dataDF <- rbind(x_data_test, x_data_train)
```

The final step for task#1 was to combine all the t3 files into one file by merging the column. This is accomplished using cbind().

```
SubjectTrainingDF <- cbind(subjectDF, y_dataDF, x_dataDF)
```

Task#2 Extracts only the measurements on the mean and standard deviation for each measurement.

I am creating a list of column names that contain the word "std" or "mean" in any case in anywhere in the column name. Setting **ignore.case=T** makes it check for string irrespective of the case(lower or upper).

```
stdCol <- grep("mean|std|SUBJECT_ID|ACTIVITY_ID", names(SubjectTrainingDF), ignore.case = T)
```

Using the list of filtered column name, I am creating a subset of the original Data Frame.

```
filterSubjectDF <- SubjectTrainingDF[stdCol]
```

Task#3 Uses descriptive activity names to name the activities in the data set

I wanted to retain the *ACTIVITY_ID* column so that I can verify if the values were correctly applied to the description column. Hence I created a new column (**ACTIVITY_DESC**) to store Activity Description

```
filterSubjectDF["ACTIVITY_DESC"] <- NA
```

The above command creates a new column with values = "NA"

I used data.table to map the Activity Description to the Activity Id in filterSubjectDF. For this I have to first convert the Data Frame to Data Table.

```
tempTbl1 <- data.table(filterSubjectDF)
tempTbl2 <- data.table(activity)
tempTbl1[, ACTIVITY_DESC:=tempTbl2[ACTIVITY_ID==tempTbl1$ACTIVITY_ID, ACTIVITY_DESC]]
```

Nolw I need to convert the object class from Data Table to Data Frame.

```
filterSubjectDF <- as.data.frame.matrix(tempTbl1)
```

Task#4 Apropriate labels of data set with descriptive variable names

My goal was to add some more meaning to the variable names in the data set, but at the same time not make the variable name too long.With that in mind, I used the following transformation:

- Acc => Accel
- t => time
- f => freq

The CodeBook has the full description of the variable in the data set.

I create a character loist of the modified column names based on the above assumptions I made.

```
newColNames <- c("SUBJECT_ID", "ACTIVITY_ID", "timeBodyAccel-mean-X", "timeBodyAccel-mean-Y", "timeBody
                 "timeBodyAccel-std-X", "timeBodyAccel-std-Y", "timeBodyAccel-std-Z", "timeGravityAccel-
                 "timeGravityAccel-mean-Y", "timeGravityAccel-mean-Z", "timeGravityAccel-std-X", "timeG
                 "timeGravityAccel-std-Z", "timeBodyAccelJerk-mean-X", "timeBodyAccelJerk-mean-Y", "tim
                 "timeBodyAccelJerk-std-X", "timeBodyAccelJerk-std-Y", "timeBodyAccelJerk-std-Z", "time
                 "timeBodyGyro-mean-Y", "timeBodyGyro-mean-Z", "timeBodyGyro-std-X", "timeBodyGyro-std-
```

```
                  "timeBodyGyro-std-Z", "timeBodyGyroJerk-mean-X", "timeBodyGyroJerk-mean-Y", "timeBodyGy
                  "timeBodyGyroJerk-std-X", "timeBodyGyroJerk-std-Y", "timeBodyGyroJerk-std-Z", "timeBody
                  "timeBodyAccelMag-std", "timeGravityAccelMag-mean", "timeGravityAccelMag-std", "timeBod
                  "timeBodyAccelJerkMag-std", "timeBodyGyroMag-mean", "timeBodyGyroMag-std", "timeBodyGy
                  "timeBodyGyroJerkMag-std", "freqBodyAccel-mean-X", "freqBodyAccel-mean-Y", "freqBodyAc
                  "freqBodyAccel-std-X", "freqBodyAccel-std-Y", "freqBodyAccel-std-Z", "freqBodyAccel-mea
                  "freqBodyAccel-meanFreq-Z", "freqBodyAccelJerk-mean-X", "freqBodyAccelJerk-mean-Y", "f
                  "freqBodyAccelJerk-std-X", "freqBodyAccelJerk-std-Y", "freqBodyAccelJerk-std-Z", "freq
                  "freqBodyAccelJerk-meanFreq-Y", "freqBodyAccelJerk-meanFreq-Z", "freqBodyGyro-mean-X",
                  "freqBodyGyro-mean-Z", "freqBodyGyro-std-X", "freqBodyGyro-std-Y", "freqBodyGyro-std-Z
                  "freqBodyGyro-meanFreq-X", "freqBodyGyro-meanFreq-Y", "freqBodyGyro-meanFreq-Z", "freq
                  "freqBodyAccelMag-std", "freqBodyAccelMag-meanFreq", "freqBodyBodyAccelJerkMag-mean",
                  "freqBodyBodyAccelJerkMag-meanFreq", "freqBodyBodyGyroMag-mean", "freqBodyBodyGyroMag-
                  "freqBodyBodyGyroMag-meanFreq", "freqBodyBodyGyroJerkMag-mean", "freqBodyBodyGyroJerkM
                  "freqBodyBodyGyroJerkMag-meanFreq", "angle-timeBodyAccelMean_gravity", "angle-timeBody
                  "angle-timeBodyGyroMean-gravityMean", "angle-timeBodyGyroJerkMean_gravityMean", "angle-
                  "angle-Y_gravityMean", "angle-Z_gravityMean", "ACTIVITY_DESC")
```

Nos I assign the character vector to replace the column names.

```
names(filterSubjectDF) <- newColNames
```

Task#5 Create a second, independent tidy data set with the average of each variable for each activity and each subject

I am using ddply to group by Subject and Activity. So with 30 subject and 6 activites, we will see 180 records that represents the Mean(aka Average) values.

```
SummarySubjectDF <- ddply(filterSubjectDF, .(SUBJECT_ID, ACTIVITY_ID, ACTIVITY_DESC), numcolwise(mean))
```

To tidy up, I am now removing column ACTIVITY_ID as there is already ACTIVITY_DESC. This follows Tidy principle.

```
SummarySubjectDF["ACTIVITY_ID"] <- NULL
```

Finally I am creating a outfile of this summary data.

```
write.table(SummarySubjectDF, "SummarySubjectData.txt", sep = " ", row.name= FALSE)
```

Please note that once you download the file I created, you can view it using the following command. MAke sure that the file is downloaded to your current working directory. You can check the current working directory using the command *getwd()*

```
data <- read.table("SummarySubjectData.txt", header = TRUE)
View(data)
```