Code Boook - Aggregation of Human Activity Recognition Using Smartphones

Mike Quashne

1/2/2020

Description

The purpose of the run_analysis.r script is to use provided datasets to create an table of the mean of summary statistics for smartphone accelerometer observations by subject and activity type. The following columns are included:

- SUBJECT — Integer Factor, 1 to 30. The number assigned to a specific smartphone from which observations were taken.
- ACTIVITY — String Factor. The motion activity being performed at the time of the observation
  - WALKING
  - WALKING_UPSTAIRS
  - WALKING_DOWNSTAIRS
  - SITTING
  - STANDING
  - LAYING
- NUMBERED SUMMARY STATISTICS (remaining columns) — Double, -1 to 1. Average of each variable for a given SUBJECT and ACTIVITY. These variables represent a summary statistic (mean or standard deviation) related to standardized and normalized measurements from a smartphone accelerometer at the time of observation.

Data Manipulation

Downloading Data

Data for this set was downloaded as a zip file from https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20HAR%20Dataset.zip

This zip file contains observations from the accelerometers of 30 Samsung Galaxy S smartphones during six different motion activities.

The following files were used to compose this dataset:

File	Description
activitylabels.txt	A lookup table providing text for each of the 6 numbered activities in the datasets.
features.txt	A list of the names of 561 variables gathered from the accelerometer data. These measurements correspond to the columns in the X datasets.
subject_test.txt	A column of integers representing which of the 30 test subjects each observation in the X_test dataset came from.

File	Description
subject_train.txt	A column of integers representing which of the 30 test subjects each observation in the X_train dataset came from.
Y_test.txt	A column of integers representing which of the six motion action activities corresponds to each observation in the X_test dataset
Y_train.txt	A column of integers representing which of the six motion action activities corresponds to each observation in the X_train dataset
X_text.txt	A table of measurements containing 561 variables for 2,947 discrete test observations from subjects during motion activities. The variables correspond to the names in features.txt
X_train.txt	A table of measurements containing 561 variables for 7,352 discrete training observations from subjects during motion activities. The variables correspond to the names in features.txt

The files were downloaded and stored in a folder in the working directory.

Requirement #1 - Merge the training and test sets

The following packages were loaded for use in the analysis:

```
library(dplyr)
library(tidyr)
library(readr)
```

Then, each text file was read into an r tibble variable using read_table from the dplyr package. No files contained column names, so the col_names option was set to false:

```
subject_test<-read_table("./UCI HAR Dataset/test/subject_test.txt", col_names=FALSE)
X_test<-read_table("./UCI HAR Dataset/test/X_test.txt", col_names=FALSE)
Y_test<-read_table("./UCI HAR Dataset/test/Y_test.txt", col_names=FALSE)
subject_train<-read_table("./UCI HAR Dataset/train/subject_train.txt", col_names=FALSE)
X_train<-read_table("./UCI HAR Dataset/train/X_train.txt", col_names=FALSE)
Y_train<-read_table("./UCI HAR Dataset/train/Y_train.txt", col_names=FALSE)
activity_labels<-read_table("./UCI HAR Dataset/activity_labels.txt", col_names=FALSE)
features<-read_table("./UCI HAR Dataset/features.txt", col_names=FALSE)
```

After reading in the data, the tibbles were formatted for a clean merge. The merge command rearranges the observations included in each dataset, so much of the pre-processing occurs before the merge.

First, activities in Y_test and Y_train were bound to the corresponding observations in X_test and X_train, respectively, and the column was named "Activity" in both tibbles:

```
colnames(Y_test)<-"Activity"
colnames(Y_train)<-"Activity"
X_test<-tbl_df(cbind(Y_test,X_test))
X_train<-tbl_df(cbind(Y_train,X_train))
```

Then, the corresponding subject from subject_test and subject_trained were bound to each observation in X_test and X_train, and the column named "Subject":

```
colnames(subject_test)<-"Subject"
colnames(subject_train)<-"Subject"
X_test<-tbl_df(cbind(subject_test,X_test))
X_train<-tbl_df(cbind(subject_train,X_train))
```

Prior to the merge, a column (Observation_Type) was added to each dataset using dplyr's mutate function with a string corresponding to which of the observation files the observation came from. "Test" was used for observations in X_test and "Train" was used for observations in X_train. This column was included to allow for later decomposition of the dataset if necessary:

```
X_test<-mutate(X_test,Observation_Type = "Test")
dim(X_test)
```

```
## [1] 2947  564
```

```
X_train<-mutate(X_train,Observation_Type = "Train")
dim(X_train)
```

```
## [1] 7352  564
```

Then, with each tibble formatted the same way, the X_test and X_train tibbles were merged into a single tibble called X_set using option all=TRUE to preserve all observations from both tables:

```
X_set<-tbl_df(merge(X_test,X_train,all=TRUE))
dim(X_set)
```

```
## [1] 10299   564
```

This fulfills assignment criteria #1

Requirement #4 - Descriptive variable names

Following the merge, a few cleanup operations were performed. First, column names were added to the X_set using already defined column names and a character vector made up of the column from the features tibble:

```
colnames(X_set)<-c("Subject","Activity",unlist(features[,1]),"Observation_Type")
```

This fulfills assignment criteria #4

Requirement #3 - Descriptive activity names

Then, the "Subject", "Activity", and "Observation_Type" columns were converted into factors from double, double, and character, respectively:

```
X_set<-mutate(X_set,Subject=as.factor(Subject),Activity=as.factor(Activity), Observation_Type=as.factor
```

Finally, the factor numbers in the "Activity" column were replaced with the corresponding text from the activity_lables tibble:

```
levels(X_set$Activity)<-unlist(activity_labels[,2])
```

This fulfills assignment criteria #3

The result is a fully merged dataset with descriptive variable names and activity names (requirements 1, 3, and 4).

Requirement #2 - Select only certain columns

To create the final product, only the factor variable columns, as well as those containing the mean and standard deviation of individual observations are needed. The mean and standard deviation columns all

end in either “mean()” or “std()”. A select function was performed to gather the factor variables (excluding Observation_Type), and any columns containing “mean()” or “std()”, bringing the total number of columns from 564 to 68:

```
X_agg<-select(X_set,Subject,Activity,contains("mean()"),contains("std()"))
dim(X_agg)
```

```
## [1] 10299    68
```

This fulfills assignment criteria #3

##Requirement 5 - Create a second dataset with the average of each variable for each activity and each subject

To create the final dataset, the aggregated set from the previous step was grouped by subject and activity, then the summarize_all function in dplyr was used to find the mean of all non-grouped variables:

```
Summary_Set<-group_by(X_agg,Subject,Activity) %>% summarize_all(funs(mean))
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
Summary_Set
```

```
## # A tibble: 180 x 68
## # Groups:   Subject [30]
##    Subject Activity `1 tBodyAcc-mea... `2 tBodyAcc-mea... `3 tBodyAcc-mea...
##    <fct>   <fct>               <dbl>            <dbl>            <dbl>
##  1 1       WALKING             0.277         -0.0174          -0.111
##  2 1       WALKING...            0.255         -0.0240          -0.0973
##  3 1       WALKING...            0.289         -0.00992         -0.108
##  4 1       SITTING             0.261         -0.00131         -0.105
##  5 1       STANDING            0.279         -0.0161          -0.111
##  6 1       LAYING              0.222         -0.0405          -0.113
##  7 2       WALKING             0.276         -0.0186          -0.106
##  8 2       WALKING...            0.247         -0.0214          -0.153
##  9 2       WALKING...            0.278         -0.0227          -0.117
## 10 2       SITTING             0.277         -0.0157          -0.109
## # ... with 170 more rows, and 63 more variables: `41 tGravityAcc-mean()-X` <dbl>,
## #   `42 tGravityAcc-mean()-Y` <dbl>, `43 tGravityAcc-mean()-Z` <dbl>, `81
## #   tBodyAccJerk-mean()-X` <dbl>, `82 tBodyAccJerk-mean()-Y` <dbl>, `83
## #   tBodyAccJerk-mean()-Z` <dbl>, `121 tBodyGyro-mean()-X` <dbl>, `122
## #   tBodyGyro-mean()-Y` <dbl>, `123 tBodyGyro-mean()-Z` <dbl>, `161
## #   tBodyGyroJerk-mean()-X` <dbl>, `162 tBodyGyroJerk-mean()-Y` <dbl>, `163
## #   tBodyGyroJerk-mean()-Z` <dbl>, `201 tBodyAccMag-mean()` <dbl>, `214
## #   tGravityAccMag-mean()` <dbl>, `227 tBodyAccJerkMag-mean()` <dbl>, `240
## #   tBodyGyroMag-mean()` <dbl>, `253 tBodyGyroJerkMag-mean()` <dbl>, `266
## #   fBodyAcc-mean()-X` <dbl>, `267 fBodyAcc-mean()-Y` <dbl>, `268
```

```
## #   fBodyAcc-mean()-Z` <dbl>, `345 fBodyAccJerk-mean()-X` <dbl>, `346
## #   fBodyAccJerk-mean()-Y` <dbl>, `347 fBodyAccJerk-mean()-Z` <dbl>, `424
## #   fBodyGyro-mean()-X` <dbl>, `425 fBodyGyro-mean()-Y` <dbl>, `426
## #   fBodyGyro-mean()-Z` <dbl>, `503 fBodyAccMag-mean()` <dbl>, `516
## #   fBodyBodyAccJerkMag-mean()` <dbl>, `529 fBodyBodyGyroMag-mean()` <dbl>,
## #   `542 fBodyBodyGyroJerkMag-mean()` <dbl>, `4 tBodyAcc-std()-X` <dbl>, `5
## #   tBodyAcc-std()-Y` <dbl>, `6 tBodyAcc-std()-Z` <dbl>, `44
## #   tGravityAcc-std()-X` <dbl>, `45 tGravityAcc-std()-Y` <dbl>, `46
## #   tGravityAcc-std()-Z` <dbl>, `84 tBodyAccJerk-std()-X` <dbl>, `85
## #   tBodyAccJerk-std()-Y` <dbl>, `86 tBodyAccJerk-std()-Z` <dbl>, `124
## #   tBodyGyro-std()-X` <dbl>, `125 tBodyGyro-std()-Y` <dbl>, `126
## #   tBodyGyro-std()-Z` <dbl>, `164 tBodyGyroJerk-std()-X` <dbl>, `165
## #   tBodyGyroJerk-std()-Y` <dbl>, `166 tBodyGyroJerk-std()-Z` <dbl>, `202
## #   tBodyAccMag-std()` <dbl>, `215 tGravityAccMag-std()` <dbl>, `228
## #   tBodyAccJerkMag-std()` <dbl>, `241 tBodyGyroMag-std()` <dbl>, `254
## #   tBodyGyroJerkMag-std()` <dbl>, `269 fBodyAcc-std()-X` <dbl>, `270
## #   fBodyAcc-std()-Y` <dbl>, `271 fBodyAcc-std()-Z` <dbl>, `348
## #   fBodyAccJerk-std()-X` <dbl>, `349 fBodyAccJerk-std()-Y` <dbl>, `350
## #   fBodyAccJerk-std()-Z` <dbl>, `427 fBodyGyro-std()-X` <dbl>, `428
## #   fBodyGyro-std()-Y` <dbl>, `429 fBodyGyro-std()-Z` <dbl>, `504
## #   fBodyAccMag-std()` <dbl>, `517 fBodyBodyAccJerkMag-std()` <dbl>, `530
## #   fBodyBodyGyroMag-std()` <dbl>, `543 fBodyBodyGyroJerkMag-std()` <dbl>
```

This fullfils assignment criteria #5, and finishes the assignment