CodeBook

Mike Holmquest

5/14/2020

Project Introduction

One of the most exciting areas in all of data science right now is wearable computing - see for example this article . Companies like Fitbit, Nike, and Jawbone Up are racing to develop the most advanced algorithms to attract new users. The data linked to from the course website represent data collected from the accelerometers from the Samsung Galaxy S smartphone. A full description is available at the site where the data was obtained:

http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

Here is the data for the project:

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

Review criteria

1. The submitted data set is tidy.
2. The Github repo contains the required scripts.
3. GitHub contains a code book that modifies and updates the available codebooks with the data to indicate all the variables and summaries calculated, along with units, and any other relevant information.
4. The README that explains the analysis files is clear and understandable.
5. The work submitted for this project is the work of the student who submitted it.

Getting and Cleaning Data Course Project

The purpose of this project is to demonstrate your ability to collect, work with, and clean a data set. The goal is to prepare tidy data that can be used for later analysis. You will be graded by your peers on a series of yes/no questions related to the project. You will be required to submit: 1) a tidy data set as described below, 2) a link to a Github repository with your script for performing the analysis, and 3) a code book that describes the variables, the data, and any transformations or work that you performed to clean up the data called CodeBook.md. 4) You should also include a README.md in the repo with your scripts. This repo explains how all of the scripts work and how they are connected. 5) You should create one R script called run_analysis.R that does the following.

Merges the training and the test sets to create one data set. Extracts only the measurements on the mean and standard deviation for each measurement. Uses descriptive activity names to name the activities in the data set Appropriately labels the data set with descriptive variable names. From the data set in step 4, creates a second, independent tidy data set with the average of each variable for each activity and each subject. Good luck!

Load data

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
```

```
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

The files now need to be downloaded from the internet from a zip file and loaded into the files folder.

Download Zip File

```
if (!file.exists("UCIDataset.zip")){
        fileURL <- "https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20HAR%20Dataset.
        download.file(fileURL, destfile = "UCIDataset.zip" , method="curl")
}

if (!file.exists("UCIDataset")) {
        unzip("UCIDataset.zip")
}
```

Once all the files are downloaded they need to be loaded into the environment to use in R Studio

Files are in new folder names "UCI HAR Dataset". What are the file names that we are interested in?

```
list.files("UCI HAR Dataset", recursive=TRUE)
```

```
##  [1] "activity_labels.txt"
##  [2] "features_info.txt"
##  [3] "features.txt"
##  [4] "README.txt"
##  [5] "test/Inertial Signals/body_acc_x_test.txt"
##  [6] "test/Inertial Signals/body_acc_y_test.txt"
##  [7] "test/Inertial Signals/body_acc_z_test.txt"
##  [8] "test/Inertial Signals/body_gyro_x_test.txt"
##  [9] "test/Inertial Signals/body_gyro_y_test.txt"
## [10] "test/Inertial Signals/body_gyro_z_test.txt"
## [11] "test/Inertial Signals/total_acc_x_test.txt"
## [12] "test/Inertial Signals/total_acc_y_test.txt"
## [13] "test/Inertial Signals/total_acc_z_test.txt"
## [14] "test/subject_test.txt"
## [15] "test/X_test.txt"
## [16] "test/y_test.txt"
## [17] "train/Inertial Signals/body_acc_x_train.txt"
## [18] "train/Inertial Signals/body_acc_y_train.txt"
## [19] "train/Inertial Signals/body_acc_z_train.txt"
## [20] "train/Inertial Signals/body_gyro_x_train.txt"
## [21] "train/Inertial Signals/body_gyro_y_train.txt"
## [22] "train/Inertial Signals/body_gyro_z_train.txt"
## [23] "train/Inertial Signals/total_acc_x_train.txt"
## [24] "train/Inertial Signals/total_acc_y_train.txt"
## [25] "train/Inertial Signals/total_acc_z_train.txt"
## [26] "train/subject_train.txt"
## [27] "train/X_train.txt"
## [28] "train/y_train.txt"
```

These are all .txt files Load all data frames from project files and add in column names

```r
subject_test <- read.table("UCI HAR Dataset/test/subject_test.txt")
x_test <- read.table("UCI HAR Dataset/test/X_test.txt")
y_test <- read.table("UCI HAR Dataset/test/y_test.txt")
subject_train <- read.table("UCI HAR Dataset/train/subject_train.txt")
x_train <- read.table("UCI HAR Dataset/train/X_train.txt")
y_train <- read.table("UCI HAR Dataset/train/y_train.txt")
```

Now download and look at the features and activity tables.

```r
features <- read.table("UCI HAR Dataset/features.txt")
activities <- read.table("UCI HAR Dataset/activity_labels.txt")
head(features)
```

```
##   V1                V2
## 1  1 tBodyAcc-mean()-X
## 2  2 tBodyAcc-mean()-Y
## 3  3 tBodyAcc-mean()-Z
## 4  4  tBodyAcc-std()-X
## 5  5  tBodyAcc-std()-Y
## 6  6  tBodyAcc-std()-Z
```

```r
head(activities)
```

```
##   V1                 V2
## 1  1            WALKING
## 2  2   WALKING_UPSTAIRS
## 3  3 WALKING_DOWNSTAIRS
## 4  4            SITTING
## 5  5           STANDING
## 6  6             LAYING
```

Add column names to the data frames using the features and activity data frames

```r
colnames(subject_train) <- "subjectID"
colnames(x_train) <- features[,2]
colnames(y_train) <- "activityID"

colnames(subject_test) <- "subjectID"
colnames(x_test) <- features[,2]
colnames(y_test) <- "activityID"

## Check dimensions on
dim(y_train)
```

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

```r
dim(x_train)
```

```
## [1] 7352  561
```

Merge Data frames

The first task is to merge the data sets together. To do this you either need to column bind first and then row bind, or row bind first, and then column bind. I chose to row bind first. and then create a new file named "Merged_UCI_Data" that had all of the needed raw data in it.

Now all of the data sets need to be combined Merge all training and test data sets

```r
Subject_Data <- rbind(subject_train, subject_test)
X_Data <- rbind(x_train, x_test)
Y_Data <- rbind(y_train, y_test)
Merge_UCI_Data <- cbind(Subject_Data, Y_Data, X_Data)
```

Extract Mean and Standard Deviation

Once the raw data is merged you need to then extract the identification data (subject and activity) and also any variables that have mean, or standard deviation data.

The next step extracts some valid information from the data frame. Extract only the mean and standard deviation measurement

```r
FinalData <- Merge_UCI_Data %>%
        select(subjectID, activityID, contains("mean"), contains("std"))
##Look at data to make sure the right data was collected
head(FinalData, 1)
```

```
##   subjectID activityID tBodyAcc-mean()-X tBodyAcc-mean()-Y tBodyAcc-mean()-Z
## 1         1          5         0.2885845       -0.02029417        -0.1329051
##   tGravityAcc-mean()-X tGravityAcc-mean()-Y tGravityAcc-mean()-Z
## 1            0.9633961           -0.1408397            0.1153749
##   tBodyAccJerk-mean()-X tBodyAccJerk-mean()-Y tBodyAccJerk-mean()-Z
## 1            0.07799634           0.005000803           -0.06783081
##   tBodyGyro-mean()-X tBodyGyro-mean()-Y tBodyGyro-mean()-Z
## 1       -0.006100849        -0.03136479          0.1077254
##   tBodyGyroJerk-mean()-X tBodyGyroJerk-mean()-Y tBodyGyroJerk-mean()-Z
## 1             -0.0991674            -0.05551737             -0.0619858
##   tBodyAccMag-mean() tGravityAccMag-mean() tBodyAccJerkMag-mean()
## 1         -0.9594339            -0.9594339             -0.9933059
##   tBodyGyroMag-mean() tBodyGyroJerkMag-mean() fBodyAcc-mean()-X
## 1          -0.9689591              -0.9942478        -0.9947832
##   fBodyAcc-mean()-Y fBodyAcc-mean()-Z fBodyAcc-meanFreq()-X
## 1        -0.9829841        -0.9392687             0.2524829
##   fBodyAcc-meanFreq()-Y fBodyAcc-meanFreq()-Z fBodyAccJerk-mean()-X
## 1             0.1318358           -0.05205025            -0.9923325
##   fBodyAccJerk-mean()-Y fBodyAccJerk-mean()-Z fBodyAccJerk-meanFreq()-X
## 1            -0.9871699            -0.9896961                 0.8703845
##   fBodyAccJerk-meanFreq()-Y fBodyAccJerk-meanFreq()-Z fBodyGyro-mean()-X
## 1                  0.210697                 0.2637079         -0.9865744
##   fBodyGyro-mean()-Y fBodyGyro-mean()-Z fBodyGyro-meanFreq()-X
## 1         -0.9817615         -0.9895148             -0.2575489
##   fBodyGyro-meanFreq()-Y fBodyGyro-meanFreq()-Z fBodyAccMag-mean()
## 1             0.09794711               0.547151         -0.9521547
##   fBodyAccMag-meanFreq() fBodyBodyAccJerkMag-mean()
## 1            -0.08843612                 -0.9937257
##   fBodyBodyAccJerkMag-meanFreq() fBodyBodyGyroMag-mean()
## 1                      0.3469885              -0.9801349
##   fBodyBodyGyroMag-meanFreq() fBodyBodyGyroJerkMag-mean()
## 1                  -0.1289889                  -0.9919904
##   fBodyBodyGyroJerkMag-meanFreq() angle(tBodyAccMean,gravity)
## 1                     -0.07432303                  -0.1127543
##   angle(tBodyAccJerkMean),gravityMean) angle(tBodyGyroMean,gravityMean)
## 1                           0.03040037                       -0.4647614
##   angle(tBodyGyroJerkMean,gravityMean) angle(X,gravityMean)
## 1                          -0.01844588           -0.8412468
##   angle(Y,gravityMean) angle(Z,gravityMean) tBodyAcc-std()-X tBodyAcc-std()-Y
## 1            0.1799406          -0.05862692       -0.9952786       -0.9831106
##   tBodyAcc-std()-Z tGravityAcc-std()-X tGravityAcc-std()-Y tGravityAcc-std()-Z
## 1       -0.9135264          -0.9852497          -0.9817084           -0.877625
##   tBodyAccJerk-std()-X tBodyAccJerk-std()-Y tBodyAccJerk-std()-Z
## 1           -0.9935191             -0.98836            -0.993575
##   tBodyGyro-std()-X tBodyGyro-std()-Y tBodyGyro-std()-Z tBodyGyroJerk-std()-X
## 1        -0.9853103        -0.9766234        -0.9922053            -0.9921107
##   tBodyGyroJerk-std()-Y tBodyGyroJerk-std()-Z tBodyAccMag-std()
## 1            -0.9925193            -0.9920553        -0.9505515
##   tGravityAccMag-std() tBodyAccJerkMag-std() tBodyGyroMag-std()
## 1           -0.9505515            -0.9943364         -0.9643352
##   tBodyGyroJerkMag-std() fBodyAcc-std()-X fBodyAcc-std()-Y fBodyAcc-std()-Z
## 1             -0.9913676       -0.9954217        -0.983133        -0.906165
##   fBodyAccJerk-std()-X fBodyAccJerk-std()-Y fBodyAccJerk-std()-Z
## 1           -0.9958207           -0.9909363            -0.9970517
##   fBodyGyro-std()-X fBodyGyro-std()-Y fBodyGyro-std()-Z fBodyAccMag-std()
## 1        -0.9850326        -0.9738861        -0.9940349         -0.956134
##   fBodyBodyAccJerkMag-std() fBodyBodyGyroMag-std() fBodyBodyGyroJerkMag-std()
## 1                 -0.993755             -0.9613094                 -0.9906975
```

```r
dim(FinalData)
```

```
## [1] 10299    88
```

Appropriately label the data set with descriptive names

Change the Activity ID codes into descriptive names with the activities data table

```r
FinalData$activityID <- activities[FinalData$activityID, 2]
```

Now that we have the data within the columns tidy, our next task is to tidy the labels for the columns. I decided to first look at the current names of the columns to find out what I could change. I found a lot of abbreviations were being used so I tried to relabel what I could to clarify the variables.

What do some of the column names look like?

```r
head(colnames(FinalData), 10)
```

```
##  [1] "subjectID"              "activityID"             "tBodyAcc-mean()-X"
##  [4] "tBodyAcc-mean()-Y"      "tBodyAcc-mean()-Z"      "tGravityAcc-mean()-X"
##  [7] "tGravityAcc-mean()-Y"   "tGravityAcc-mean()-Z"   "tBodyAccJerk-mean()-X"
## [10] "tBodyAccJerk-mean()-Y"
```

From this information I used gsub for string substitutions.

```r
names(FinalData) <- gsub ("Acc", "Accelerometer", names(FinalData))
names(FinalData) <- gsub ("Gyro", "Gyroscope", names(FinalData))
names(FinalData) <- gsub ("BodyBody", "Body", names(FinalData))
names(FinalData) <- gsub ("Mag", "Magnitude", names(FinalData))
names(FinalData) <- gsub ("^t", "Time", names(FinalData))
names(FinalData)<- gsub("^f", "Frequency", names(FinalData))
names(FinalData) <- gsub("tBody", "TimeBody", names(FinalData))
names(FinalData) <- gsub("-mean()", "Mean", names(FinalData),
        ignore.case = TRUE)
names(FinalData) <- gsub("std", "STD", names(FinalData),
        ignore.case = TRUE)
```

```r
names(FinalData) <- gsub("-freq()", "Frequency", names(FinalData),
        ignore.case = TRUE)
names(FinalData) <- gsub("freq()", "Frequency", names(FinalData),
                         ignore.case = TRUE)
names(FinalData) <- gsub("angle", "Angle", names(FinalData))
names(FinalData) <- gsub("gravity", "Gravity", names(FinalData))
```

Summary Data

The now that our data is Tidy, our next task is to create a new data table that has the summary statistics for mean and standard deviation. I am going to create a new data table called "TidyFinalData" that has only the means of the mean and standard devation data. I will then write this file so that I can hand it in for the assignment.

Format data set to create a new data table with the average of each variable

```r
TidyFinalData <- FinalData %>%
        group_by(subjectID, activityID) %>%
        summarise_all(funs(mean))
```

```
## Warning: `funs()` was deprecated in 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))
```

Format data set to create a new data table with the average of each variable

```r
TidyFinalData <- FinalData %>%
        group_by(subjectID, activityID) %>%
        summarise_all(funs(mean))
```

Write the summary output to a text file

```r
write.table(TidyFinalData, "TidyFinalData.txt", row.name = FALSE)
```