ReadMe

Coleen Smith

4/2/2019

Assignment

This assignment will create one R script called run_analysis.R that does the following. 1. Merges the training and the test sets to create one data set. 2. Extracts only the measurements on the mean and standard deviation for each measurement. 3. Uses descriptive activity names to name the activities in the data set. 4. Appropriately labels the data set with descriptive variable names. 5. 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.

Raw Data

Data collected from the accelerometers from the Samsung Galaxy S smartphone was made available to download. 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 are the data for the project:

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

UCI HAR Dataset

After downloading and unzipping files, I renamed the directory from "UCI HAR Dataset" to "UCI_HAR_Dataset" to avoid possible issues with spaces in a pathname.

```
    dir("UCI_HAR_Dataset")
```

```
## [1] "activity_labels.txt" "features_info.txt"   "features.txt"
## [4] "README.txt"          "test"                "train"
```

activity_labels.txt - six activities, two columns (number & name), all uppercase, no headers
features.txt - 561 variables, two columns (number & name), mixed case, no headers
- 46 occurrences of "mean"
- 10 occurrences of "Mean", all as part of an ordered pair, rows 555-561
- 33 occurrences of "std", only single occurrence/row
features_info.txt - information on the variables and calculations listed in features.txt.
README.txt - overview of project
test (folder) - data from first 30 subjects
train (folder)- data from remaining 70 subjects

Based on the information included in features_info.txt, the files in the test and train and train folders are formatted and organized the same way. Since the test folder files are smaller with data from just 30 subjects, they became the starting point to understand the data and prepare the files for merging.

Test Folder

```
    dir("UCI_HAR_Dataset/test")
```

```
## [1] "Inertial Signals" "subject_test.txt" "X_test.txt"
## [4] "y_test.txt"
```

To get a sense of how the data were arranged, I also looked at each of the files using atom.

test(folder):
inertial signals (folder):
each .txt - a set of 35 element vectors.
subject_test.txt - a single column of numbers,no column headers.
x_test.txt - rows of 561 element vectors, no column headers.
y_test.txt - a single column of numbers, no column headers. 1 - 496 occurrences
2 - 471 occurrences
3 - 420 occurrences
4 - 491 occurrences
5 - 532 occurrences
6 - 537 occurrences
Total = 2947

Train Folder

```
    dir("UCI_HAR_Dataset/train")
```

```
## [1] "Inertial Signals"  "subject_train.txt" "X_train.txt"
## [4] "y_train.txt"
```

Train (folder) mirrors file structure and organization seen in test files, only larger because files represent 70 subjects rather than 30.

1. Merge the training and the test sets to create one data set

Based on this assignment's requirements, I decided to follow Wickham approach to tidying data when there is one type of data in multiple tables (Ref: Wickham, Section 3.5).

1. Read the files into a list of tables.
2. For each table, add a new column that records the original file name.
3. Combine all tables into a single table.

In this case, test and train data are one type of data observed for separate groups of people. Thinking ahead to tidy data,

```
1. Each variable forms a column.
    - subject (100)
    - activity (6)
    - features (561)
2. Each observation forms a row.
    - An observation is of a given subject
    - performing a specific activity
    - represented by 561 measured features
3. Each type of observational unit forms a table.
    - Interim: test and train in separate tables
    - Final: test and train data in single table
```

Since the subject, activity and measurements are recorded in separate tables within test and train, the first step would be to create a test data file with subject, activity and measurement observations. Then, do the same for the train data. Since they have the same variables, combine the interim test and train data files using rbind().

Read test files into a list of tables: subject_test, X_test, and Y_test data

From a bookkeeping perspective, a good place to start is with the number lines per file. The number of data lines in the source file should equal the number of lines in the destination file. [Back in the dark ages when file transfer protocol included a magnetic tape and FedEx Overnight, I spent a day looking for 13 "lost" lines out of 25,000 - lesson burned.]

```
## [1] 561
## attr(,"lastLineHasNewline")
## [1] TRUE
```

```
## [1] 2947
## attr(,"lastLineHasNewline")
## [1] TRUE
```

```
## [1] 2947
## attr(,"lastLineHasNewline")
## [1] TRUE
```

```
## [1] 2947
## attr(,"lastLineHasNewline")
## [1] TRUE
```

According to README.txt and features_info.txt, features.txt represents the names of the 561 variables recorded. Since the 561 variable list already contains the summary data (mean & std) required to comoplete the assignment, the merge for this assignment will not include the additional detail observations contained in the inertial signals test and train folders.

Using read.table, create a data frame from each file: subject_test.txt, X_test.txt and Y_test.txt. Create a character vector from features.txt to use later in naming the Y_test data frame columns.

```
    subject_test <- read.table("UCI_HAR_Dataset/test/subject_test.txt", header = FALSE)
    str(subject_test)
```

```
## 'data.frame':    2947 obs. of  1 variable:
##  $ V1: int  2 2 2 2 2 2 2 2 2 2 ...
```

```
    X_test <- read.table("UCI_HAR_Dataset/test/X_test.txt", header = FALSE)
    ## This is a looooong output, just need highlights
    str(X_test, list.len = 5)
```

```
## '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 ...
##  $ V4  : num  -0.938 -0.975 -0.994 -0.995 -0.994 ...
##  $ V5  : num  -0.92 -0.967 -0.97 -0.973 -0.967 ...
##   [list output truncated]
```

```
    Y_test <- read.table("UCI_HAR_Dataset/test/Y_test.txt", header = FALSE)
    str(Y_test)
```

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

```
##  $ V1: int  5 5 5 5 5 5 5 5 5 5 ...
```

```
    features <- readLines("UCI_HAR_Dataset/features.txt")
    str(features)
```

```
##  chr [1:561] "1 tBodyAcc-mean()-X" "2 tBodyAcc-mean()-Y" ...
```

For readibility, set column names. Use conventions described in README.txt and features_info.txt files. This will make it easier when need to filter "mean" and "std" variables.

```
    names(subject_test) <- "subject"
    names(Y_test) <- "activity"
    names(X_test) <- features
```

With the exception of adding column names, no data has been transformed or filtered. This is useful in case something goes wrong downstream and need to reset to original values. Each file (subject_test.txt, X_test.txt and Y_test.txt) has 2947 rows, and represent different variables of a single observation.

Combine test tables into a single table

Use cbind() to merge the three test data files. Bind in order of subject (subject_test), activity (Y_test), then recorded measurements (X_test). Review the result.

```
    test_merged <- cbind(subject_test, Y_test, X_test)
    str(test_merged, list.len = 5)
```

```
## 'data.frame':    2947 obs. of  563 variables:
##  $ subject                                : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ activity                               : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ 1 tBodyAcc-mean()-X                    : num  0.257 0.286 0.275 0.27 0.275 ...
##  $ 2 tBodyAcc-mean()-Y                    : num  -0.0233 -0.0132 -0.0261 -0.0326 -0.0278 ...
##  $ 3 tBodyAcc-mean()-Z                    : num  -0.0147 -0.1191 -0.1182 -0.1175 -0.1295 ...
##   [list output truncated]
```

With test_merged's 2947 observations, no rows lost. 563 column variables represent subject and activity columns added to 561 feature measurements.

Read train files into a list of tables: subject_train, X_train, and Y_train data

As mentioned before, the test and train files have identical structure and organization. Although larger, they can be processed in the same way: count lines, name the columns, create a data frame for each.

```
## [1] 7352
## attr(,"lastLineHasNewline")
## [1] TRUE
```

```
## [1] 7352
## attr(,"lastLineHasNewline")
## [1] TRUE
```

```
## [1] 7352
## attr(,"lastLineHasNewline")
## [1] TRUE
```

Each train data files has 7352 lines.

Using read.table, create a data frame from each file: subject_train.txt, X_train.txt and Y_train.txt.

```
    subject_train <- read.table("UCI_HAR_Dataset/train/subject_train.txt", header = FALSE)
    str(subject_train)
```

```
## 'data.frame':    7352 obs. of  1 variable:
##  $ V1: int  1 1 1 1 1 1 1 1 1 1 ...
```

```
    X_train <- read.table("UCI_HAR_Dataset/train/X_train.txt", header = FALSE)
    ## This is a looooong output, just need highlights
    str(X_train, list.len = 5)
```

```
## 'data.frame':    7352 obs. of  561 variables:
##  $ V1  : num  0.289 0.278 0.28 0.279 0.277 ...
##  $ V2  : num  -0.0203 -0.0164 -0.0195 -0.0262 -0.0166 ...
##  $ V3  : num  -0.133 -0.124 -0.113 -0.123 -0.115 ...
##  $ V4  : num  -0.995 -0.998 -0.995 -0.996 -0.998 ...
##  $ V5  : num  -0.983 -0.975 -0.967 -0.983 -0.981 ...
##   [list output truncated]
```

```
    Y_train <- read.table("UCI_HAR_Dataset/train/Y_train.txt", header = FALSE)
    str(Y_train)
```

```
## 'data.frame':    7352 obs. of  1 variable:
##  $ V1: int  5 5 5 5 5 5 5 5 5 5 ...
```

For readibility, set column names to match subject_test, X_test, Y_test. Since X_test and X_train have identical structure, use the features vector created earlier to name columns.

```
    names(subject_train) <- "subject"
    names(Y_train) <- "activity"
    names(X_train) <- features
```

With the exception of adding column names, no data has been transformed or filtered. This is useful in case something goes wrong downstream and need to reset to original values. Each file (subject_train.txt, X_train.txt and Y_train.txt) has 7352 rows, and represent different variables of a single observation.

Combine train tables into a single table

Use cbind() to merge the three train data files. Bind in order of subject (subject_train), activity (Y_train), then recorded measurements (X_train). Review the result.

```
    train_merged <- cbind(subject_train, Y_train, X_train)
    str(train_merged, list.len = 5)
```

```
## 'data.frame':    7352 obs. of  563 variables:
##  $ subject                               : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ activity                              : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ 1 tBodyAcc-mean()-X                   : num  0.289 0.278 0.28 0.279 0.277 ...
##  $ 2 tBodyAcc-mean()-Y                   : num  -0.0203 -0.0164 -0.0195 -0.0262 -0.0166 ...
##  $ 3 tBodyAcc-mean()-Z                   : num  -0.133 -0.124 -0.113 -0.123 -0.115 ...
##   [list output truncated]
```

With train_merged's 7352 observations, no rows lost. 563 variables represent subject and activity columns added to 561 feature measurements.

For each test_merge and train_merge table, add a new column that records the original file name.

Install and load dplyr packages. Use mutate() to create a new variable to identify the original file (test or train).

```
    test_add <- mutate(test_merged, original_file = "test")
    ## sanity check ... expecting 564
    ncol(test_add)
```

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

```
    tail(test_add$original_file)
```

```
## [1] "test" "test" "test" "test" "test" "test"
```

```
    train_add <- mutate(train_merged, original_file = "train")
    ## sanity check ... expecting 564
    ncol(test_add)
```

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

```
    tail(train_add$original_file)
```

```
## [1] "train" "train" "train" "train" "train" "train"
```

Using ncol confirms that each new table now has 564 columns. Explicitly specifying new column by name in tail operation, demonstrates original_name was added to test_merged and train_merged and was filled.

Combine test_add and train_add tables into a single table

Since the test_add and train_add tables have the same column organization, use rbind to combine into a single table.

```
    test_train <- rbind(test_add, train_add)
    ## 2947 from test + 7352 from train = 10299 in test_train
    ## matches the 10299 instances recorded in raw data
    nrow(test_train)
```

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

Set descriptive column names

Resources

The following Resources were very helpful in strategizing and working with the datasets: ## Packages

R.utils
dplyr

Tidy Data

David Hood's "thoughtfulbloke" blog at https://thoughtfulbloke.wordpress.com/2015/09/09/getting-and-cleaning-the-assignme (04/02/2019) Hadley Wickham's "Tidy Data" paper at https://vita.had.co.nz/papers/tidy-data.pdf (04/03/2019)
Hadley Wickam & Garrett Grolemund, R for Data Science

General R

R Markdown Cheatsheet at https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf