CodeBook

Introduction

The purpose of this project is to obtain raw data, merge data from multiple files, select a subset of that data and convert it into a tidy data format, including the tidy data files and a code book, along with the raw data.

Hardware & Software used

Apple MacBook Pro 2.66 GHz Intel Core 2 Duo 8 GiB RAM
OS X v. 10.11.3 El Capitan
R version 3.2.3 (2015-12-10) – "Wooden Christmas-Tree"
Platform: x86_64-apple-darwin13.4.0 (64-bit)
TextWrangler v. 5.0.2 (3786) (for writing, saving R scripts)
RStudio v. 0.99.491 (for creating Markdown, Rmarkdown, html, and PDF documents)

R packages:
dplyr: A Grammar of Data Manipulation
(Dependencies have been omitted.)

Study Design

From the Web page for the data set:

"*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.*

"*Check the README.txt file for further details about this dataset.*"

Code book

The files are in a directory named *UCI HAR Dataset*. Within that directory are four files and two more folders. The files are *activity_labels.txt*, *features_info.txt*, *features.txt*, and *README.txt*. The two directories are called *test* and *train*. The *test* directory contains three files and an additional directory The files are *subject_test.txt*, *X_test.txt*, and *y_test.txt*. The directory is named *Inertial Signals*. The *train* directory

is structured identically to the *test* directory, but the files are named *subject_train.txt*, *X_train.txt*, and *y_train.txt*.

README.txt provides a brief description of the experiments, as well as descriptions of some of the files.

features_info.txt provides explanations of the variables contained in *X_test.txt* and *X_train.txt* files. Additionally, it contains information helpful in understanding the formatting used in the names provided in *features.txt*. This latter aspect is the more useful information for this project.

features.txt provides formatted names for each of the 561 variables in the *X_test.txt* and *X_train.txt* files.

activity_labels.txt matches the integer activity identifier to the activity name.

X_train.txt provides the training data, while *X_test.txt* provides the test data set. These are the data from which we obtain a subset for our this project. Values are normalized unit-less values, with a range of [-1, 1]. (Note: The author of this report lacks the contextual knowledge to understand the study's authors' explanations of exactly what many the variables of interest represent, so readers are referred to the *features_info.txt* file.)

y_train.txt provides the activity, denoted by an integer, for each observation in the training set. *y_test.txt* is the corresponding file for the test set.

subject_train.txt identifies the subject for each observation in the training set with an integer. *subject_test.txt* is the corresponding file for the test set. The range of the two files, combined, is 1 to 30.

Though not used in this project, each *Inertial Signals* directory contains nine files. We quote the descriptions from the *README.txt* file.

total_acc_x_train.txt, *total_acc_y_train.txt*, and *total_acc_z_train.txt* contain "the acceleration signal from the smartphone accelerometer X[, Y, and Z, respectively] axis in standard gravity units 'g'. Every row shows a 128 element vector."

body_acc_x_train.txt, *body_acc_y_train.txt*, and *body_acc_z_train.txt* contain "the body acceleration signal obtained by subtracting the gravity from the total acceleration." Presumably, the units are also standard gravity units.

body_gyro_x_train.txt, *body_gyro_y_train.txt*, and *body_gyro_z_train.txt* contain "the angular velocity vector measured by the gyroscope for each window sample. The units are radians/second."

The test directory contains a corresponding *Inertial Samples* directory with equivalent contents.

The files are space delineated.

From the raw data, we build two tidy data files, *tidy.data.1.Rdata* and *tidy.data.2.Rdata*. These files are also provided in comma separated value formats.

tidy.data.1 is a data frame in which the first column is a factor variable identifying the subject of each observation, the second column is a factor variable identifying the activity for each observation, and the remaining columns are a subset of the numeric variables and values provided by *X_test.txt* and *X_train.txt* files, with meaningful column (variable) names added. The variables selected for the subset were the variables containing means and standard deviations for each observation. Like the superset from which they are selected, the values are normalized, unit-less, and have a range of [-1, 1].

tidy.data.2 is a data frame with the average value of each numeric variable for each treatment, i.e., each of 180 combinations of subject and activity.

Instructions for obtaining tidy data

The data are available from https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20HAR%20Dataset.zip. We download the data with R. (The data were downloaded on December 23, 2015.)

Load the data. We first load the dplyr library.

```
library( dplyr)
```

We then identify our current working directory. It can be changed as appropriate with the *setwd* function.

```
getwd()
```

We download and uncompress the raw data.

```
fileUrl =
  "https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20HAR%20Dataset.zip"
    # The URL where the file is located

filePath = "./RawData.zip"

download.file( fileUrl, filePath, method = "curl")
    # download .zip file to working directory as RawData.zip

unzip( filePath,  exdir = "./")
    # uncompress raw data file
```

The uncompressed files are read into R using *read.table*.

```
    # Test Data

subject.test = read.table( "./UCI HAR Dataset/test/subject_test.txt")
    # 2947 x 1 matrix identifying subject by integer identifier

X.test = read.table( "./UCI HAR Dataset/test/X_test.txt")
    # 2947 x 561 matrix variable values

y.test = read.table( "./UCI HAR Dataset/test/y_test.txt")
    # 2947 x 1 matrix identifying acivities by integer 1,...,6



    # Training Data

subject.train = read.table( "./UCI HAR Dataset/train/subject_train.txt")
    # 7352 x 1

X.train = read.table( "./UCI HAR Dataset/train/X_train.txt")
    # 7352 x 561

y.train = read.table( "./UCI HAR Dataset/train/y_train.txt")
    # 7351 x 1
```

```
    # Variable and Activity labels

features = read.table( "./UCI HAR Dataset/features.txt",
                       row.names = 1,
                       stringsAsFactors = FALSE)
    # 561 x 1 variable names

activity.labels = read.table( "./UCI HAR Dataset/activity_labels.txt",
                              row.names = 1,
                              stringsAsFactors = FALSE)
    # 6 x 1 activity names
```

Merge the training data with the test data. We combine the training set and test set into a single data set.

```
subject.master = rbind( subject.train, subject.test)

colnames( subject.master) = "Subject"

subject.master$Subject = paste( "subject.",
                                formatC( subject.master$Subject, width = 2, flag = "0"),
                                sep = "")
    # prepend "subject." to integer subject identifier after adding leading zero
    #   to single digit integers.

subject.master$Subject = as.factor( subject.master$Subject)
    # convert subject strings to factors

X.master = rbind( X.train, X.test)
    # will provide variable names later, after using regular expression
    #   manipulation on features

y.master = rbind( y.train, y.test)

colnames( y.master) = "Activity"
```

We've added column names *Subject* and *Activity* to our newly created data frames *subject.master* and *y.master*, respectively. We wait to provide meaningful variable names to *X.master*, because we would first like to reformat them to a more readable and descriptive form.

Extract the mean and standard deviation for each measurement. The *X.master* data frame includes columns for mean and standard deviation for each measurement. We wish to extract these. Instead of searching through 561 different variables manually, we use regular expressions and the *grep* function to identify the variables in which we are interested.

Note that the *features* data frame includes seven variables which include the character string "Mean" in which we are not interested. These are the last seven values of our *features* data frame, *angle(tBodyAccMean,gravity)*, *angle(tBodyAccJerkMean),gravityMean)*, *angle(tBodyGyroMean,gravityMean)*,

angle(tBodyGyroJerkMean,gravityMean), *angle(X,gravityMean)*, *angle(Y,gravityMean)*, and *angle(Z,gravityMean)*. We are careful to create our search strings in such a way that these variables are not matched.

```
index.mean = grep( "mean()", features$V2, fixed = TRUE)
    # identify and save index values of variables representing means

index.std = grep( "std()", features$V2, fixed = TRUE)
    # identify and save index values of variables representing standard
    #   deviations

index = c( index.mean, index.std)

index = sort( index)
    # index is a vector containing the column indices of the variables of
    #   interest.

features.subset = features[ index,]
    # this vector will be used for column names of variables of interest after
    #   being modified with regular expressions. Note that features.subset is a
    #   vector, not a data frame.

X.subset = X.master[ , index]
    # this is a matrix of the values of our variables of interest
```

Use descriptive names to label the activities in the data set. We replace the integers used to identify the activity for each observation with the nominal name of the activity. This is fairly straightforward with a for loop with six iterations, one for each activity. We use the activity names in our *activity.labels* data frame.

Additionally, we change the case of the activity names to lowercase. We also change the activity to a factor variable.

```
activity.labels$V2 = tolower( activity.labels$V2)
    # convert activity names to lowercase

for (i in 1 : nrow( activity.labels)) {

    y.master$Activity[ y.master$Activity == i] = activity.labels$V2[ i]
        # replace integer identifiers with descriptive character strings

}

y.master$Activity = as.factor( y.master$Activity)
    # convert descriptive character strings to factors
```

Appropriately label the data set with descriptive variable names. The variable names provided in *features.txt* are reasonably descriptive, but we can improve them. (An example of a provided name:

fBodyBodyGyroJerkMag-std().) We use a combination of literal strings and regular expressions, along with the *sub* and *gsub* functions to make the desired changes.

The names provided in *features.txt* contain a prefix of "t" or "f" to denote whether the measurements are obtained from time or frequency domain signals. We expand these prefixes to "time" and "freq".

The "t" or "f" prefixes are followed by a descriptive name of the type of measurement, provided in CamelCase. We retain this part of the names as is.

The next portion of the name is either "mean()" or "std()". We change these to "mean" and "sd", respectively. We choose these since they are the names of the corresponding R functions and will hopefully be informative while being shorter.

Some of the names have an additional suffix of "X", "Y", or "Z", indicating the axis on which the measurement was obtained.

Except for the "t" or "f" prefixes, the different portions of the provided names are separated with a hyphen. We replace this with a dot (period) character. We also add a dot character immediately after "time" or "freq".

We reject the strategy of all lowercase names with no underscore or dot characters for spaces, since it would result in many names that are unreadable, defeating the purpose of tidy data. Consider *freq.BodyGyroJerkMag.sd* compared to *freqbodygyrojerkmagsd*. Our strategy of retaining CamelCase and using dot characters provides a better result.

The frequency domain magnitude variables appear to have an extra "Body" in the description. We remove this.

```
features.subset = gsub( "^t", "time.", features.subset)

features.subset = gsub( "^f", "freq.", features.subset)

features.subset = gsub( "[()]", "", features.subset)
    # parentheses placed inside brackets because parentheses are otherwise used in
    #   regex syntax

features.subset = gsub( "-", ".", features.subset)

features.subset = sub( "BodyBody", "Body", features.subset)
    # frequency magnitude variables appear to have extra "Body" in name

features.subset = sub( "std", "sd", features.subset)

colnames( X.subset) = features.subset
```

Combine subject identifier, activity identifier and variable values into single data frame. We still build a data frame that has columns for the subject, activity, and each measurement for each observation. We also group the observations by subject and activity and save the resulting data frame in three formats: a .csv file, a .txt file, and a .Rdata file.

```
dataset = cbind( subject.master, y.master, X.subset)

tidy.data = group_by( dataset, Subject, Activity)

write.csv( tidy.data, "Tidy_Data_1.csv", row.names = FALSE)
```

```
    # save tidy.data as a .csv file in the working directory

write.table( tidy.data, "Tidy_Data_1.txt", row.names = FALSE)
    # # save tidy.data as a .txt file in the working directory

save( tidy.data, file = "tidy.data.1.Rdata")
    # save tidy.data as an .Rdata file in working directory
```

Create a second, independent tidy data set. We create a second tidy data set with the average of each variable with the average for each subject and activity combination. We have 30 subjects and 6 activities, so our resulting data frame will have 180 rows. The *summarize_each* function in the *dplyr* package allows us to create the desired data frame very easily.

As before, we save our new data frame in three file formats.

```
tidy.data.2 = summarize_each( tidy.data, "mean")

write.csv( tidy.data.2, "Tidy_Data_2.csv", row.names = FALSE)
    # save tidy.data.2 as a .csv file in the working directory

write.table( tidy.data.2, "Tidy_Data_2.txt", row.names = FALSE)
    # # save tidy.data.2 as a .txt file in the working directory

save( tidy.data.2, file = "tidy.data.2.Rdata")
    # save tidy.data.2 as a .Rdata file in the working directory
```

Citations

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition Using Smartphones. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.

R Core Team (2015). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Hadley Wickham and Romain Francois (2015). dplyr: A Grammar of Data Manipulation. R package version 0.4.3. https://CRAN.R-project.org/package=dplyr

Yihui Xie (2016). knitr: A General-Purpose Package for Dynamic Report Generation in R. R package version 1.12.