Readme

D McMillan

October 20, 2015

Data Cleaning Course Project

Setup

This exercize required multiple steps. The first was obtaining the data. It was downloaded from

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

It consisted of a fair number of files, in multiple folders. In particular, the “Test” and “Train” sets of data were unnecessarily placed into separate folders, even though the file names had no name collisions.

For simplicity, therefore, all the files were expanded into a single folder named HAR.

Also, “dplyr” and “data.table” packages would be needed. These are included as `require()` Statements.

Processing

As there are numerous files, it’s good to get to know a little about them. One way is to load all of them into R as data tables, so that they can be easily examined (excluding the README file). To minimize confusion, each data table gets the same name as its source file name, using the `assign` function. This code accomplished that step:

```
filelist<-list.files(path="HAR")
nFiles<-length(filelist)
dtnames<-substr(filelist,start=1,stop = nchar(filelist)-4)

for(i in 1:nFiles)
  {
  print(paste(i,"Creating",dtnames[i]))  ## progress messages to console
  if(filelist[i]!="README.txt") assign(dtnames[i],fread(paste("HAR/",filelist[i],sep = "")))
}
```

A tables() command gives this list:

```
      NAME               NROW NCOL MB

 [1,] activity_labels       6    2  1
 [2,] body_acc_x_test   2,947  128  3
 [3,] body_acc_x_train  7,352  128  8
 [4,] body_acc_y_test   2,947  128  3
 [5,] body_acc_y_train  7,352  128  8
 [6,] body_acc_z_test   2,947  128  3
 [7,] body_acc_z_train  7,352  128  8
 [8,] body_gyro_x_test  2,947  128  3
 [9,] body_gyro_x_train 7,352  128  8
[10,] body_gyro_y_test  2,947  128  3
[11,] body_gyro_y_train 7,352  128  8
[12,] body_gyro_z_test  2,947  128  3
[13,] body_gyro_z_train 7,352  128  8
[14,] features            561    2  1
[15,] features_info         1   14  1
[16,] subject_test      2,947    1  1
[17,] subject_train     7,352    1  1
[18,] total_acc_x_test  2,947  128  3
[19,] total_acc_x_train 7,352  128  8
[20,] total_acc_y_test  2,947  128  3
[21,] total_acc_y_train 7,352  128  8
[22,] total_acc_z_test  2,947  128  3
[23,] total_acc_z_train 7,352  128  8
[24,] X_test            2,947  561 13
[25,] X_train           7,352  561 32
[26,] y_test            2,947    1  1
[27,] y_train           7,352    1  1
```

What is quickly apparent is that all the “test” files have 2,947 rows, and all the “train” files have 7,352. The workhorse files are X_test and X_train, having 561 colums. All the data we care about are in these data tables. The other data files, most of them with 128 columns of raw data, will be ignored, since we know we are eventually going to select only pre-processed columns anyway. Since some data tables have but one column (y_test, y_train, subject_test, and subject_train) a `table()` command gives a quick view of what is inside.

y_train

```
   1    2    3    4    5    6
1226 1073  986 1286 1374 1407
```

y_test

```
  1   2   3   4   5   6
496 471 420 491 532 537
```

subject_test

```
  2   4   9  10  12  13  18  20  24
302 317 288 294 320 327 364 354 381
```

subject_train

```
  1   3   5   6   7   8  11  14  15  16  17  19  21  22  23  25  26  27  28  29  30
347 341 302 325 308 281 316 323 328 366 368 360 408 321 372 409 392 376 382 344 383
```

The output shows that they have elements between 1 and 6, or 1 and 30. The activity identifiers and subject IDs have been found!

We also discover that of the 30 subjects, 9 were in the `test` group, the rest in `train`.

1. Combining `test` and `train`

To build the `test` and `train` data tables from these building blocks, we execute the following:

```
train <- data.table(subject_train$V1, y_train$V1, X_train)
test1 <- data.table(subject_test$V1,  y_test$V1,  X_test)
```

Now, combine them

```
alldata<-rbind(train,test1)
```

2. Extracting needed columns

Noting that the `features` file has 561 character-type values, we have found our column names for the primary data files. Loading these into a vector gives a list that can be pared down to those columns containing the word `mean` or `std`, meaning that they contain summary statistics of different measures.

```
cols<-features$V2                    ## all column names
cols<-gsub("-","_",cols)             ## get rid of hyphens in names
```

Now to `grep` the `mean` and `std` columns, then combine into a single set of column names to be used in the tidy final:

```
meancols<-cols[grep("[Mm]ean",cols)]  ## find columns with Means
stdcols<- cols[grep("[Ss]td",cols)]   ## find columns with Std Dev
                                      ## in case there are any column names containing both
reducedcols<- unique(c("ID","Activity",meancols,stdcols,"act_name"))
```

3. Give the Activities descriptive names

This is quickly done by adding a final variable in the dataset called `act_name` using the vector of names found earlier, and taking advantage of the fact that our `Activity` variable works as an index.

```
activity<-as.character(activity_labels$V2)
alldata<-mutate(alldata,act_name=activity[V2])
```

4. Assign descriptive names to our 561 variables

Now assign column names to dataset, and finish what we started in Step 2, ie, select the final, reduced data set, `all_ms` .

```
colnames(alldata)<-c("ID","Activity",cols,'act_name')
## now select only the needed columns
all_ms<-select(alldata,one_of(reducedcols))
```

Running another table() command counts the rows by ID and Activity, giving assurance that the data are organized as planned: 1 row per ID, Activity, measurement:

```
with(all_ms,table(act_name,ID))
sum(with(all_ms,table(act_name,ID)))  ## all rows accounted for
```

```
act_name              1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
  LAYING             50 48 62 54 52 57 52 54 50 58 57 60 62 51 72 70 71 65 83 68 90 72
  SITTING            47 46 52 50 44 55 48 46 50 54 53 51 49 54 59 69 64 57 73 66 85 62
  STANDING           53 54 61 56 56 57 53 54 45 44 47 61 57 60 53 78 78 73 73 73 89 63
  WALKING            95 59 58 60 56 57 57 48 52 53 59 50 57 59 54 51 61 56 52 51 52 46
  WALKING_DOWNSTAIRS 49 47 49 45 47 48 47 38 42 38 46 46 47 45 42 47 46 55 39 45 45 36
  WALKING_UPSTAIRS   53 48 59 52 47 51 51 41 49 47 54 52 55 54 48 51 48 58 40 51 47 42
                    ID
act_name              23 24 25 26 27 28 29 30
  LAYING             72 72 73 76 74 80 69 70
  SITTING            68 68 65 78 70 72 60 62
  STANDING           68 69 74 74 80 79 65 59
  WALKING            59 58 74 59 57 54 53 65
  WALKING_DOWNSTAIRS 54 55 58 50 44 46 48 62
  WALKING_UPSTAIRS   51 59 65 55 51 51 49 65

  [1] 10299
```

5. Creating a summarized tidy data set

Final step is to summarize all numeric variables by calculating the mean of each of the 86 variables that passed our screen. Using a series of `dplyr` commands, this is quickly accomplished:

```
means_HAR_ms<- all_ms %>%
    group_by(ID,act_name,Activity) %>%
    summarise_each(funs(mean)) %>%
    arrange(ID,act_name)
```

Our final dataset (mean_HAR_ms) should have a row for each subject and each activity (30 * 6 = 180). And it does (though ony the first 6 of the 89 variables are shown in columns below):

```
print(means_HAR_ms,width=95)
Source: local data table [180 x 89]
Groups: ID, act_name

ID                act_name Activity tBodyAcc_mean()_X tBodyAcc_mean()_Y tBodyAcc_mean()_Z
   (int)               (chr)    (int)             (dbl)             (dbl)             (dbl)
1      1             LAYING        6         0.2215982      -0.040513953        -0.1132036
2      1            SITTING        4         0.2612376      -0.001308288        -0.1045442
3      1           STANDING        5         0.2789176      -0.016137590        -0.1106018
4      1            WALKING        1         0.2773308      -0.017383819        -0.1111481
5      1 WALKING_DOWNSTAIRS        3         0.2891883      -0.009918505        -0.1075662
6      1   WALKING_UPSTAIRS        2         0.2554617      -0.023953149        -0.0973020
7      2             LAYING        6         0.2813734      -0.018158740        -0.1072456
8      2            SITTING        4         0.2770874      -0.015687994        -0.1092183
9      2           STANDING        5         0.2779115      -0.018420827        -0.1059085
10     2            WALKING        1         0.2764266      -0.018594920        -0.1055004
..   ...                ...      ...               ...               ...               ...
Variables not shown: tBodyAcc_mean()_Z (dbl), tGravityAcc_mean()_X (dbl), etc up to 89th
```