finalReadMe

Jim Rhudy

10/27/2021

Although this project is my own work, I have freely made use of materials presented and/or linked within the Getting and Cleaning Data MOOC offered by Johns Hopkins University, especially: https://thoughtfulbloke.wordpress.com/2015/09/09/getting-and-cleaning-the-assignment/

A full description of the dataset is available at: http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

The data are available at: https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20HAR%20Dataset.zip

The data were downloaded and unzipped into a main project folder. The data and all supplemental files were found in a series of nested folders. For this reason each file specified in the provided README was copied into the main folder to avoid file path white space and to simplify file management. The download includes training and testing subsets for 10,299 observations of 30 subjects undertaking six activities while wearing a smart phone. The observations are reported in a multivariate time series dataset with 559 gyroscopic and accelerometric measurements in three axes and labelled by subject number and activity.

Setup

```
#setwd in tool bar to project folder
train<-read.table("X_train.txt")  #training subset
dim(train)     #   7352  561    corresponds to expected 70% of total instances
```

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

```
test<-read.table("X_test.txt")    #testing subset
dim(test)      #  2947  561    corresponds to expected 30% of total instances
```

```
## [1] 2947  561
```

Step One

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

The project documentation was read carefully until the purpose of every supplemental file was understood. The files were used to label, inspect, and merge the data back into the entire dataset as follows:

```r
# Windows users open files in notepad++ to prepare for the following steps

#label rows in train
subjectTrain<-read.table("subject_train.txt") #prepare to label subjects
activityTrain<-read.table("y_train.txt")      #prepare to label activities
trainLabelled<-cbind(subjectTrain, activityTrain, train) #label all rows

#label rows in test as above
subjectTest<-read.table("subject_test.txt")
activityTest<-read.table('y_test.txt')
testLabelled<-cbind(subjectTest, activityTest, test)

#rowbind train and test to entire combined dataset
entireRowLabelled<-rbind(trainLabelled, testLabelled)
#I now have the data combined with rows labelled

#label columns in combined dataset
givenColumnNames<-read.table("features.txt")
fullColumnNames<-c("subject", "activity", givenColumnNames$V2)

#length(fullColumnNames) #563
#ncol(entireRowLabelled) #563
entireFullyLabelled<-rbind(fullColumnNames, entireRowLabelled)
#dim(entireFullyLabelled) #10300   563
entireFullyLabelled[1:5, 1:5]
```

```
##        V1     V1.1              V1.2                V2                V3
## 1 subject activity tBodyAcc-mean()-X tBodyAcc-mean()-Y tBodyAcc-mean()-Z
## 2       1        5        0.28858451      -0.020294171       -0.13290514
## 3       1        5        0.27841883      -0.016410568       -0.12352019
## 4       1        5        0.27965306      -0.019467156       -0.11346169
## 5       1        5        0.27917394      -0.026200646       -0.12328257
```

The output of this step is a reassembled dataset with rows and columns labelled.

Step Two

2. Extracts only the measurements on the mean and standard deviation for each measurement.

Throughout, I have used grep() and grepl() to identify columns referring to either mean or standard deviation without regard to the position of the string within the variable label.

```r
#table(grepl("mean", entireFullyLabelled))  #inspect subset of "mean"
meanList<-grep("mean", entireFullyLabelled) #list all columns referring to mean
#meanList


#table(grepl("std", entireFullyLabelled))   #inspect subset of "std
stdList<-grep("std", entireFullyLabelled)   #list columns referring to std
#stdList

meanCols<-entireFullyLabelled[, meanList]   #select columns referring to mean
meanCols[1:5, 1:4]
```

```
##                   V1                V2                V3                   V41
## 1 tBodyAcc-mean()-X tBodyAcc-mean()-Y tBodyAcc-mean()-Z tGravityAcc-mean()-X
## 2        0.28858451      -0.020294171       -0.13290514           0.96339614
## 3        0.27841883      -0.016410568       -0.12352019           0.96656113
## 4        0.27965306      -0.019467156       -0.11346169            0.9668781
## 5        0.27917394      -0.026200646       -0.12328257           0.96761519
```

```
stdCols<-entireFullyLabelled[, stdList]     #select columns referring to std
stdCols[1:5, 1:4]
```

```
##                  V4               V5               V6                 V44
## 1 tBodyAcc-std()-X tBodyAcc-std()-Y tBodyAcc-std()-Z tGravityAcc-std()-X
## 2       -0.9952786      -0.98311061      -0.91352645         -0.98524969
## 3      -0.99824528      -0.97530022      -0.96032199         -0.99741134
## 4      -0.99537956      -0.96718701      -0.97894396         -0.99957395
## 5      -0.99609149       -0.9834027       -0.9906751         -0.99664558
```

```
selectEntire<-cbind(meanCols, stdCols)      #limit data to columns of interest
selectEntire[1:5, 1:4]
```

```
##                   V1                V2                V3                   V41
## 1 tBodyAcc-mean()-X tBodyAcc-mean()-Y tBodyAcc-mean()-Z tGravityAcc-mean()-X
## 2        0.28858451      -0.020294171       -0.13290514           0.96339614
## 3        0.27841883      -0.016410568       -0.12352019           0.96656113
## 4        0.27965306      -0.019467156       -0.11346169            0.9668781
## 5        0.27917394      -0.026200646       -0.12328257           0.96761519
```

```
#dim(selectEntire)

#relabel rows in selectEntire

#relabel subjects
# nrow(subjectTrain) #7352
# nrow(subjectTest)  #2947
nameSubject<-("subject")                    #prepare a column name
subjectEntire<-rbind(nameSubject, subjectTrain, subjectTest)
#nrow(subjectEntire) #10300, matching length of dataframe to be labelled

#relabel activities
# nrow(activityTrain) #7352
# nrow(activityTest)  #2947
nameActivity<-("activity")
activityEntire<-rbind(nameActivity, activityTrain, activityTest)
#nrow(activityEntire) #10300, matching length of dataframe to be labelled

#relabel all rows of selectEntire
#ncol(selectEntire) #79
labelledEntire<-cbind(subjectEntire, activityEntire, selectEntire)
#ncol(labelledEntire) #81
labelledEntire[1:5, 1:5]          #inspect result of procedure to label rows
```

```
##        V1     V1.1                 V1.2                 V2                V3
## 1 subject activity tBodyAcc-mean()-X tBodyAcc-mean()-Y tBodyAcc-mean()-Z
## 2       1        5        0.28858451      -0.020294171       -0.13290514
## 3       1        5        0.27841883      -0.016410568       -0.12352019
## 4       1        5        0.27965306      -0.019467156       -0.11346169
## 5       1        5        0.27917394      -0.026200646       -0.12328257
```

```
labelledEntire[10298:10300, 1:5]  #inspect result of procedure to label rows
```

```
##       V1 V1.1       V1.2           V2           V3
## 10298 24    2 0.34996609  0.030077442  -0.11578796
## 10299 24    2 0.23759383   0.01846687 -0.096498932
## 10300 24    2 0.15362719 -0.018436506  -0.13701846
```

```
#dim(labelledEntire)               #10300    81
```

The output of this step is a subset of the original dataset which includes only those columns which refer to a mean or standard deviation measurement.

Steps Three and Four

3. Uses descriptive activity names to name the activities in the data set
4. Appropriately labels the data set with descriptive variable names.

```
# head(labelledEntire)  #inspect data pre-transformation
# tail(labelledEntire)

#prepare for row_to_names to replace V1, V2, etc.

# rename columns meaningfully
relabelledEntire<-labelledEntire%>%
  janitor::row_to_names(1)
labelledEntire[1:5, 1:4]
```

```
##        V1     V1.1              V1.2                V2
## 1 subject activity tBodyAcc-mean()-X tBodyAcc-mean()-Y
## 2       1        5        0.28858451      -0.020294171
## 3       1        5        0.27841883      -0.016410568
## 4       1        5        0.27965306      -0.019467156
## 5       1        5        0.27917394      -0.026200646
```

```
#rename activity meaningfully
activityDescriptive<-recode(relabelledEntire$activity,
       "1"="walking",
       "2"="walking_upstairs",
       "3"="walking_downstairs",
       "4"="sitting",
       "5"="standing",
       "6"="laying"
       )

# head(activityDescriptive)   # inspect column names and data post-transformation
# tail(activityDescriptive)
#
# length(activityDescriptive) #10299
# dim(relabelledEntire) #10299 81

entire3_4<-cbind(activityDescriptive, relabelledEntire)
entire3_4[1:5, 1:4]
```

```
##   activityDescriptive subject activity tBodyAcc-mean()-X
## 2            standing       1        5        0.28858451
## 3            standing       1        5        0.27841883
## 4            standing       1        5        0.27965306
## 5            standing       1        5        0.27917394
## 6            standing       1        5        0.27662877
```

The output of this step is a dataset with descriptive names for each of the six activities undertaken by the subjects.

Step Five

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.

Components of tidy data specify that (1) each variable should be in one column and that (2) each observation should be in a different row. Inspection of a larger sample of the output of the previous step, shown below, demonstrates this. The names of the first three columns specify descriptive activity, subject, and activity labels; the other column names represent a list of 79 different measurements either collected or calculated during each experiment undertaken by a subject during a particular activity. The dataset resulting from the previous step is tidy because both the above conditions are met. Each unique variable completely occupies a single column and the outcomes of each unique experiment completely occupy a single row.

Given that each of 30 subjects undertook each of six different activities, there are 180 possible distinct combinations of subject and activity; however, the dataset contains 10,299 rows, each reporting measurements from a single experiment. This can be explained by the multivariate time-series design of the overall study. Each of the subject and activity combinations occurs many times in the dataset; this number ranges from 36 to 95 as shown below.There are 46 and 33 columns which report mean and standard deviation data, respectively. These findings would tend to hinder rectangular pivoting. The documentation accompanying the data download does not explicitly account for this variability, nor does the metadata available on the webpage. I do not have the substantive knowledge to fully interpret the given variable labels and render them more human-readable so I did not attempt this.

```
# four corners of a rectangular data.frame
entire3_4[1:3, 1:5]             # upper left corner
```

```
##   activityDescriptive subject activity tBodyAcc-mean()-X tBodyAcc-mean()-Y
## 2            standing       1        5        0.28858451      -0.020294171
## 3            standing       1        5        0.27841883      -0.016410568
## 4            standing       1        5        0.27965306      -0.019467156
```

```
entire3_4[10297:10299, 1:5]     # lower left corner
```

```
##       activityDescriptive subject activity tBodyAcc-mean()-X tBodyAcc-mean()-Y
## 10298    walking_upstairs      24        2        0.34996609       0.030077442
## 10299    walking_upstairs      24        2        0.23759383        0.01846687
## 10300    walking_upstairs      24        2        0.15362719      -0.018436506
```

```
entire3_4[1:4, 79:82]           # upper right corner
```

```
##   fBodyAccMag-std() fBodyBodyAccJerkMag-std() fBodyBodyGyroMag-std()
## 2       -0.95613397               -0.99375495            -0.96130944
## 3       -0.97586576               -0.99196029            -0.98332192
## 4       -0.98901548               -0.99086668            -0.98602772
## 5       -0.98674196               -0.99169983            -0.98783575
##   fBodyBodyGyroJerkMag-std()
## 2                -0.99069746
## 3                -0.99639947
## 4                -0.99512739
## 5                -0.99523693
```

```
entire3_4[10297:10299, 79:82]   # lower right corner
```

```
##       fBodyAccMag-std() fBodyBodyAccJerkMag-std() fBodyBodyGyroMag-std()
## 10298       -0.37724031               -0.35897998            -0.56332745
## 10299       -0.39020111               -0.38328201            -0.56591709
## 10300       -0.36259839               -0.38238729            -0.56394661
##       fBodyBodyGyroJerkMag-std()
## 10298                -0.61794841
## 10299                -0.65495993
## 10300                -0.62520727
```

```
varList<-names(entire3_4[4:82])
length(varList) #79
```

```
## [1] 79
```

```
# data are provided over a list of 79 different mean or std variables
length(meanList) #46 mean variables
```

```
## [1] 46
```

```
length(stdList) #33 std variables
```

```
## [1] 33
```

```
dim(entire3_4) #10299    82
```

```
## [1] 10299    82
```

```
prefinalDF<-entire3_4%>%
  group_by(subject, activityDescriptive)%>%
  summarise(count=n(), .groups = "keep")%>%
  arrange(subject)%>%
  glimpse
```

```
## Rows: 180
## Columns: 3
## Groups: subject, activityDescriptive [180]
## $ subject             <chr> "1", "1", "1", "1", "1", "1", "10", "10", "10", "1~
## $ activityDescriptive <chr> "laying", "sitting", "standing", "walking", "walki~
## $ count               <int> 50, 47, 53, 95, 49, 53, 58, 54, 44, 53, 38, 47, 57~
```

The output of steps 3 and 4 is a tidy dataset which fulfills the two specifications noted above; however, this dataset is not optimized to solve the assigned problem: calculation of the average of each variable by activity and subject. For this reason, I undertook the following step to aggregate the mean of each measurement grouped by subject and activity:

```
back<-apply(entire3_4[, 4:82], 2, as.numeric)

front<-entire3_4[, 1:3]

finalDF<-cbind(front, back)

# finalDF[1:3, 1:4]
# finalDF[10297:10299, 80:82]
# dim(finalDF) #10299    82

tidyMeanDF<-finalDF %>%

  group_by(subject, activityDescriptive) %>%

  #summarise(count=n()) %>%

  summarise(across(2:80, mean))
```

```
## `summarise()` has grouped output by 'subject'. You can override using the `.groups` argument.
```

```
# four corners of rectangular data.frame
tidyMeanDF[1:3, 1:4]       #upper left corner
```

```
## # A tibble: 3 x 4
## # Groups:   subject [1]
##   subject activityDescriptive `tBodyAcc-mean()-X` `tBodyAcc-mean()-Y`
##   <chr>   <chr>                             <dbl>               <dbl>
## 1 1       laying                            0.222            -0.0405
## 2 1       sitting                           0.261            -0.00131
## 3 1       standing                          0.279            -0.0161
```

```
tidyMeanDF[178:180, 1:4]  #lower left corner
```

```
## # A tibble: 3 x 4
## # Groups:   subject [1]
##   subject activityDescriptive ‘tBodyAcc-mean()-X‘ ‘tBodyAcc-mean()-Y‘
##   <chr>   <chr>                             <dbl>               <dbl>
## 1 9       walking                           0.279             -0.0181
## 2 9       walking_downstairs                0.296             -0.0204
## 3 9       walking_upstairs                  0.262             -0.0195
```

```
tidyMeanDF[1:3, 79:81]     #upper right corner
```

```
## # A tibble: 3 x 3
##   ‘fBodyBodyAccJerkMag-std()‘ ‘fBodyBodyGyroMag-std()‘ ‘fBodyBodyGyroJerkMag-st~
##                         <dbl>                    <dbl>                    <dbl>
## 1                      -0.922                   -0.824                   -0.933
## 2                      -0.982                   -0.932                   -0.987
## 3                      -0.993                   -0.978                   -0.995
```

```
finalDF[178:180, 79:81]    #lower right corner
```

```
##     fBodyAccMag-std() fBodyBodyAccJerkMag-std() fBodyBodyGyroMag-std()
## 179        -0.9585161                -0.9935696             -0.9625924
## 180        -0.9846613                -0.9913807             -0.9731317
## 181        -0.9751071                -0.9788202             -0.9533317
```

```
dim(tidyMeanDF) #180 81
```

```
## [1] 180  81
```

The output of this step is a data.frame which has been optimised to fulfill the assignment specifications as follows:

```
#names(tidyMeanDF)

meanMean<-tidyMeanDF%>%
  group_by(subject, activityDescriptive) %>%
  summarise(means=mean(c_across(1:46))) %>%
  glimpse
```

```
## ‘summarise()‘ has grouped output by ’subject’. You can override using the ‘.groups‘ argument.

## Rows: 180
## Columns: 3
## Groups: subject [30]
## $ subject             <chr> "1", "1", "1", "1", "1", "1", "10", "10", "10", "1~
## $ activityDescriptive <chr> "laying", "sitting", "standing", "walking", "walki~
## $ means               <dbl> -0.323455253, -0.313691419, -0.329807396, -0.06302~
```

```
dim(meanMean)
```

```
## [1] 180   3
```

```r
head(meanMean)
```

```
## # A tibble: 6 x 3
## # Groups:   subject [1]
##   subject activityDescriptive   means
##   <chr>   <chr>                 <dbl>
## 1 1       laying              -0.323
## 2 1       sitting             -0.314
## 3 1       standing            -0.330
## 4 1       walking             -0.0630
## 5 1       walking_downstairs  -0.0227
## 6 1       walking_upstairs    -0.220
```

```r
tail(meanMean)
```

```
## # A tibble: 6 x 3
## # Groups:   subject [1]
##   subject activityDescriptive   means
##   <chr>   <chr>                 <dbl>
## 1 9       laying              -0.400
## 2 9       sitting             -0.382
## 3 9       standing            -0.372
## 4 9       walking             -0.133
## 5 9       walking_downstairs  -0.0787
## 6 9       walking_upstairs    -0.232
```

```r
meanStd<-tidyMeanDF%>%
  group_by(subject, activityDescriptive) %>%
  summarise(means=mean(c_across(47:79))) %>%
  glimpse
```

```
## `summarise()` has grouped output by 'subject'. You can override using the `.groups` argument.
```

```
## Rows: 180
## Columns: 3
## Groups: subject [30]
## $ subject             <chr> "1", "1", "1", "1", "1", "1", "10", "10", "10", "1~
## $ activityDescriptive <chr> "laying", "sitting", "standing", "walking", "walki~
## $ means               <dbl> -0.8966993, -0.9625317, -0.9867985, -0.3025518, -0~
```

```r
dim(meanStd)
```

```
## [1] 180   3
```

```r
head(meanStd)
```

```
## # A tibble: 6 x 3
## # Groups:   subject [1]
##   subject activityDescriptive  means
##   <chr>   <chr>                <dbl>
## 1 1       laying               -0.897
## 2 1       sitting              -0.963
## 3 1       standing             -0.987
## 4 1       walking              -0.303
## 5 1       walking_downstairs  -0.264
## 6 1       walking_upstairs    -0.428
```

```
tail(meanStd)
```

```
## # A tibble: 6 x 3
## # Groups:   subject [1]
##   subject activityDescriptive  means
##   <chr>   <chr>                <dbl>
## 1 9       laying               -0.944
## 2 9       sitting              -0.928
## 3 9       standing             -0.951
## 4 9       walking              -0.420
## 5 9       walking_downstairs  -0.265
## 6 9       walking_upstairs    -0.491
```

The output of this step is a set of two tibbles which have aggregated the mean of all "mean" columns and the mean of all "std" columns in the final tidy data.frame. Having demonstrated that the final tidy dataset is optimized to perform the specified calculations, I wrote the dataset back to the project folder as follows:

```
#write.table(tidyMeanDF, "tidyMeanDF.txt", row.names = FALSE)
```