Data Science::3::Getting & Cleaning Data: Final Project

John Tiede

08/23/2014

Begin Project Statement

Getting and Cleaning Data

by Jeff Leek, PhD, Roger D. Peng, PhD, Brian Caffo, PhD

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. 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.

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 are the data for the project:

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

You should 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) Creates a second, independent tidy data set with the average of each variable for each activity and each subject.

End Project Statement

```
# Setup commands for this R-session:
# Code to set working directory (enabled):
if (TRUE) {
    working.directory <- paste0(Sys.getenv("HOME"), "/School/Coursera/DataScience_JohnsHopkins/03_GetCl
    setwd(working.directory)
    getwd()
}
```

```
## [1] "/home/jwt/School/Coursera/DataScience_JohnsHopkins/03_GetCleanData/Project"
```

The following block downloads the data. It is disabled because this only has to be done once. For the final script, I assume that this step has been done and the data has been unzipped.

```
# 0) Download data:
# Code to download data (disabled: assuming zip file has been downloaded and unzipped)
if (FALSE) {
    dataURL <- "https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20HAR%20Dataset.zip"

    if (!file.exists("./getdata_projectfiles_Dataset.zip")) {
        download.file(dataURL, "./getdata_projectfiles_Dataset.zip", method="curl")
        dateDownloaded <- date()
        # It is possible to write 'dataDownloaded' to a file, but I used the modification time of the z
    } else {
        cat("Data file already exists.\n")
    }
    cat(paste("Zip file last modified: ", file.info(list.files(pattern="getdata_projectfiles_Dataset.zi

    # Reference: http://stackoverflow.com/questions/8986818/automate-zip-file-reading-in-r
    # Command below shows the files in the zip file.
    #zipFileInfo <- unzip("./getdata_projectfiles_Dataset.zip", list=TRUE)
}
```

Next, I want to read the data into R. Based on a forum post (https://class.coursera.org/getdata-006/forum/thread?thread_id=43), I will not merge the raw data from the 'Inertial Signals' directories.

From the forum post, CTA David Hood said the following:

"The general understanding is that you don't need the original inertial files, as they are not going to fit the pattern of the columns you are going to keep, and working with them is going to be rather painful (they are the original sensor readings that the values in x were calculated from)."

Here is a very helpful figure created by David Hood:

My first attempt was to use the 'read.fwf()' function. Applied to just 'filename.1', it crashed my system (OS: Linux/Kubuntu 14.04; Processor=4 X Intel Core i3-3225CPU @ 3.30GHz; RAM=4GB). The total file size is about 26MB, but the function uses recursive techniques to perform its activities. This somehow (memory leaks?) uses all available system memory. After researching this, I decided to use the 'sqldf' package to read the data into R.

Here are some useful references for my approach:

- http://www.ats.ucla.edu/stat/r/modules/raw_data.htm
- https://stackoverflow.com/questions/1727772/quickly-reading-very-large-tables-as-dataframes-in-r
- http://sqlite.awardspace.info/syntax/sqlitepg08.htm
- https://sites.google.com/site/timriffepersonal/DemogBlog/newformetrickforworkingwithbigishdatainr
- http://stackoverflow.com/questions/2247045/chopping-a-string-into-a-vector-of-fixed-width-character-elements

```
filename.1 <- "./UCI\ HAR\ Dataset/test/X_test.txt"
filename.2 <- "./UCI\ HAR\ Dataset/test/subject_test.txt"
filename.3 <- "./UCI\ HAR\ Dataset/test/y_test.txt"
filename.4 <- "./UCI\ HAR\ Dataset/train/X_train.txt"
filename.5 <- "./UCI\ HAR\ Dataset/train/subject_train.txt"
filename.6 <- "./UCI\ HAR\ Dataset/train/y_train.txt"

file.columns <- 561  # number of columns in file
```

```
column.width <- 16   # fixed width of each column

suppressPackageStartupMessages(library(sqldf))

# Create a SQL command to read in fixed width format file:
sql.1 <- "SELECT "
for (col in 1:(file.columns-1)) { sql.1 <- paste0(sql.1, " SUBSTR(V1,", column.width*(col-1)+1, ",16) AS
sql.1 <- paste0(sql.1, " SUBSTR(V1,", column.width*(file.columns-1)+1, ",16) AS V", file.columns, " FROM

# Read in test data:
f <- file(filename.1)
X.test <- sqldf(sql.1,
                dbname=tempfile(),
                file.format=list(header=FALSE, row.names=FALSE))
```

```
## Loading required package: tcltk
```

```
# Check data:
class(X.test)
```

```
## [1] "data.frame"
```

```
dim(X.test)
```

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

```
class(X.test[1,1])
```

```
## [1] "character"
```

```
X.test[1,1]
```

```
## [1] "  2.5717778e-001"
```

```
X.test[1,561]
```

```
## [1] " -5.7978304e-002"
```

```
# Read in the subject data:
subject.test <- read.table(filename.2, header=FALSE)

# Check data:
class(subject.test)
```

```
## [1] "data.frame"
```

```
dim(subject.test)
```

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

```
# Read in classification data:
y.test <- read.table(file=filename.3, header=FALSE)

# Check data:
class(y.test)
```

```
## [1] "data.frame"
```

```
dim(y.test)
```

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

```
# Column bind 'X.test' & 'subject.test' and 'y.test' into 'test' data frame:
test <- cbind(X.test, subject.test, y.test)

# Check data:
class(test)
```

```
## [1] "data.frame"
```

```
dim(test)
```

```
## [1] 2947  563
```

```
class(test[1,1])
```

```
## [1] "character"
```

```
test[1,1]
```

```
## [1] "  2.5717778e-001"
```

```
test[1,file.columns+1]
```

```
## [1] 2
```

```
test[1,file.columns+2]
```

```
## [1] 5
```

```
# Read in training data:
f <- file(filename.4)
X.train <- sqldf(sql.1,
                 dbname=tempfile(),
                 file.format=list(header=FALSE, row.names=FALSE))
```

```
## Warning: closing unused connection 5 (./UCI HAR Dataset/test/X_test.txt)
```

```r
# Check data:
class(X.train)
```

```
## [1] "data.frame"
```

```r
dim(X.train)
```

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

```r
class(X.train[1,1])
```

```
## [1] "character"
```

```r
X.train[1,1]
```

```
## [1] "  2.8858451e-001"
```

```r
X.train[1,561]
```

```
## [1] " -5.8626924e-002"
```

```r
# Read in the subject data:
subject.train <- read.table(filename.5, header=FALSE)

# Check data:
class(subject.train)
```

```
## [1] "data.frame"
```

```r
dim(subject.train)
```

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

```r
# Read in classification data:
y.train <- read.table(file=filename.6, header=FALSE)

# Check data:
class(y.train)
```

```
## [1] "data.frame"
```

```r
dim(y.train)
```

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

```
# Column bind 'X.train' & 'subject.train' & 'y.train' into 'train' data frame:
train <- cbind(X.train, subject.train, y.train)

# Check data:
class(train)
```

```
## [1] "data.frame"
```

```
dim(train)
```

```
## [1] 7352  563
```

```
class(train[1,1])
```

```
## [1] "character"
```

```
train[1,1]
```

```
## [1] "  2.8858451e-001"
```

```
train[1,file.columns+1]
```

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

```
train[1,file.columns+2]
```

```
## [1] 5
```

Problem 1: Merge the training & test sets to create one data set.

I am using the data frame layout shown above.

```
# Row bind 'train' & 'test' into 'all.data' data frame:
all.data <- rbind(train, test)

# Check data:
class(all.data)
```

```
## [1] "data.frame"
```

```
dim(all.data)
```

```
## [1] 10299   563
```

```
# Column 562 is the subject for which the data was collected and column 563 is
#   the activity.  These columns should probably be converted to a factor type,
#   but I am not doing that now.
# For columns 1:561, convert column classes from 'character' to 'numeric':
for (i in 1:file.columns) { all.data[,i] <- as.numeric(all.data[,i]) }

# Check data:
class(all.data)
```

```
## [1] "data.frame"
```

```
dim(all.data)
```

```
## [1] 10299   563
```

```
class(all.data[1,1])
```

```
## [1] "numeric"
```

```
class(all.data[1,file.columns+1])
```

```
## [1] "integer"
```

```
class(all.data[1,file.columns+2])
```

```
## [1] "integer"
```

```
all.data[1,1]
```

```
## [1] 0.2886
```

```
# Name the columns:
filename.7 <- "./UCI\ HAR\ Dataset/features.txt"
features <- read.table(filename.7, header=FALSE, stringsAsFactors=FALSE)
colnames(all.data) <- c(features[,2], "subject", "activity")

# Check data:
colnames(all.data)[1:4]
```

```
## [1] "tBodyAcc-mean()-X" "tBodyAcc-mean()-Y" "tBodyAcc-mean()-Z"
## [4] "tBodyAcc-std()-X"
```

```
colnames(all.data)[(dim(all.data)[2]-4):dim(all.data)[2]]
```

```
## [1] "angle(X,gravityMean)" "angle(Y,gravityMean)" "angle(Z,gravityMean)"
## [4] "subject"              "activity"
```

Problem 2: Extract only the mean and standard deviation for each measurement.

What does this mean? Looking in the 'UCI HAR Dataset' directory, there is a 'features.txt' file. This file describes what each column of the measurements are. Some columns have the (*regular expression notation*) **'.+-mean.+'** or **'.+-std.+'** as part of the labels. These are the columns that should be extracted from 'all.data' and put into their own data frame. Please note that there are columns names that have the string 'Mean' in them. I do not believe that these are valid for problem 2 based on my reading of the codebook. In addition, the 'subject' & 'activity' columns should be retained. My assumption is this exercise is a process from one step to the next. The columns 'subject' & 'activity' are required by the following problems. (This is not very clear in the problem statements.)

```r
# Determine the columns where '-mean()' & '-std()' appear:
mean.cols <- grep(".+-mean.+", colnames(all.data), perl=TRUE)
std.cols <- grep(".+-std.+", colnames(all.data), perl=TRUE)

# Combine 'mean.cols' & 'std.cols' & sort:
extract.cols <- sort(c(mean.cols, std.cols))
# Get column numbers for 'subject' & 'activity':
subject.col <- match("subject", colnames(all.data))
activity.col <- match("activity", colnames(all.data))
# Add these to columns to extract:
extract.cols <- c(extract.cols, subject.col, activity.col)

# Create new data frame based on 'extract.cols':
extract.df <- all.data[, extract.cols]

# Check data:
length(extract.cols)
```

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

```r
dim(extract.df)
```

```
## [1] 10299    81
```

```r
colnames(extract.df)
```

```
##  [1] "tBodyAcc-mean()-X"              "tBodyAcc-mean()-Y"
##  [3] "tBodyAcc-mean()-Z"              "tBodyAcc-std()-X"
##  [5] "tBodyAcc-std()-Y"               "tBodyAcc-std()-Z"
##  [7] "tGravityAcc-mean()-X"           "tGravityAcc-mean()-Y"
##  [9] "tGravityAcc-mean()-Z"           "tGravityAcc-std()-X"
## [11] "tGravityAcc-std()-Y"            "tGravityAcc-std()-Z"
## [13] "tBodyAccJerk-mean()-X"          "tBodyAccJerk-mean()-Y"
## [15] "tBodyAccJerk-mean()-Z"          "tBodyAccJerk-std()-X"
## [17] "tBodyAccJerk-std()-Y"           "tBodyAccJerk-std()-Z"
## [19] "tBodyGyro-mean()-X"             "tBodyGyro-mean()-Y"
## [21] "tBodyGyro-mean()-Z"             "tBodyGyro-std()-X"
## [23] "tBodyGyro-std()-Y"              "tBodyGyro-std()-Z"
## [25] "tBodyGyroJerk-mean()-X"         "tBodyGyroJerk-mean()-Y"
## [27] "tBodyGyroJerk-mean()-Z"         "tBodyGyroJerk-std()-X"
## [29] "tBodyGyroJerk-std()-Y"          "tBodyGyroJerk-std()-Z"
## [31] "tBodyAccMag-mean()"             "tBodyAccMag-std()"
## [33] "tGravityAccMag-mean()"          "tGravityAccMag-std()"
## [35] "tBodyAccJerkMag-mean()"         "tBodyAccJerkMag-std()"
## [37] "tBodyGyroMag-mean()"            "tBodyGyroMag-std()"
## [39] "tBodyGyroJerkMag-mean()"        "tBodyGyroJerkMag-std()"
## [41] "fBodyAcc-mean()-X"              "fBodyAcc-mean()-Y"
## [43] "fBodyAcc-mean()-Z"              "fBodyAcc-std()-X"
## [45] "fBodyAcc-std()-Y"               "fBodyAcc-std()-Z"
## [47] "fBodyAcc-meanFreq()-X"          "fBodyAcc-meanFreq()-Y"
## [49] "fBodyAcc-meanFreq()-Z"          "fBodyAccJerk-mean()-X"
## [51] "fBodyAccJerk-mean()-Y"          "fBodyAccJerk-mean()-Z"
```

```
## [53] "fBodyAccJerk-std()-X"            "fBodyAccJerk-std()-Y"
## [55] "fBodyAccJerk-std()-Z"            "fBodyAccJerk-meanFreq()-X"
## [57] "fBodyAccJerk-meanFreq()-Y"       "fBodyAccJerk-meanFreq()-Z"
## [59] "fBodyGyro-mean()-X"              "fBodyGyro-mean()-Y"
## [61] "fBodyGyro-mean()-Z"              "fBodyGyro-std()-X"
## [63] "fBodyGyro-std()-Y"               "fBodyGyro-std()-Z"
## [65] "fBodyGyro-meanFreq()-X"          "fBodyGyro-meanFreq()-Y"
## [67] "fBodyGyro-meanFreq()-Z"          "fBodyAccMag-mean()"
## [69] "fBodyAccMag-std()"               "fBodyAccMag-meanFreq()"
## [71] "fBodyBodyAccJerkMag-mean()"      "fBodyBodyAccJerkMag-std()"
## [73] "fBodyBodyAccJerkMag-meanFreq()"  "fBodyBodyGyroMag-mean()"
## [75] "fBodyBodyGyroMag-std()"          "fBodyBodyGyroMag-meanFreq()"
## [77] "fBodyBodyGyroJerkMag-mean()"     "fBodyBodyGyroJerkMag-std()"
## [79] "fBodyBodyGyroJerkMag-meanFreq()" "subject"
## [81] "activity"
```

Problem 3: Use descriptive activity names to name the activities in the data set.

What does this mean? Looking in the 'UCI HAR Dataset' directory, there is a 'activity_label.txt' file. This file describes the code for each activity for which the measurements were made. Replace the code number in the activity column with the corresponding string in this file. This should be done for the 'all.data' data frame, because this column does not exist in the 'extract.df' data frame.

```
# Read the 'activity_labels.txt' file:
filename.8 <- "./UCI\ HAR\ Dataset/activity_labels.txt"
activity <- read.table(filename.8, header=FALSE, stringsAsFactors=FALSE)

# Here is the brute force, ugly way to do it (takes a very, very long time):
if (FALSE) {
    for (row in 1:dim(all.data)[1]) {
        for (act in 1:dim(activity[1])) {
            if (all.data$activity[row] == activity[act,1]) { all.data$activity[row] <- activity[act,2];
        }
    }
}

# Here's a somewhat (faster) better method:
for (act in 1:dim(activity)[1]) {
    extract.df$activity[extract.df$activity == activity[act,1]] <- activity[act,2]
}

# Check data:
class(extract.df$activity[1])
```

```
## [1] "character"
```

```
extract.df$activity[1:4]
```

```
## [1] "STANDING" "STANDING" "STANDING" "STANDING"
```

```
extract.df$activity[(dim(extract.df)[1]-4):dim(extract.df)[1]]
```

```
## [1] "WALKING_UPSTAIRS" "WALKING_UPSTAIRS" "WALKING_UPSTAIRS"
## [4] "WALKING_UPSTAIRS" "WALKING_UPSTAIRS"
```

Problem 4: Appropriately label the data set with descriptive variable names.

What does this mean? I labeled the data columns with the original column names from 'features.txt'. It does not seem like a good idea to change these names. However, the '-' characters may be a problem, so I will change them to '.'.

```
# Repair column names:
extract.df.colnames <- gsub("-", ".", colnames(extract.df), perl=TRUE)

# Rename columns:
colnames(extract.df) <- extract.df.colnames

# Check data:
dim(extract.df)
```

```
## [1] 10299    81
```

```
head(colnames(extract.df))
```

```
## [1] "tBodyAcc.mean().X" "tBodyAcc.mean().Y" "tBodyAcc.mean().Z"
## [4] "tBodyAcc.std().X"  "tBodyAcc.std().Y"  "tBodyAcc.std().Z"
```

Problem 5: Create a second, independent tidy data set with the average of each variable for each activity and each subject.

What does this mean? Take the average of each variable for each subject while he is doing a particular activity - that is, the average for subject 1 while he is lying, the average for subject 1 while he is standing... and so on until you reach the average for each variable for subject 30 while he is lying, standing, ... 30 subjects doing 6 activities -> 180 combinations.

Reference: http://martinsbioblogg.wordpress.com/2014/03/25/using-r-quickly-calculating-summary-statistics-from-a-data-fra

```
# Step 1: Melt the data frame:
suppressPackageStartupMessages(library(reshape2))

extract.df.melt <- melt(extract.df, id=c("subject","activity"), measure.vars=colnames(extract.df)[1:(di

# Check data:
class(extract.df.melt)
```

```
## [1] "data.frame"
```

```
dim(extract.df.melt)
```

```
## [1] 813621      4
```

```
colnames(extract.df.melt)
```

```
## [1] "subject"  "activity" "variable" "value"
```

```
head(extract.df.melt)
```

```
##   subject activity          variable  value
## 1       1 STANDING tBodyAcc.mean().X 0.2886
## 2       1 STANDING tBodyAcc.mean().X 0.2784
## 3       1 STANDING tBodyAcc.mean().X 0.2797
## 4       1 STANDING tBodyAcc.mean().X 0.2792
## 5       1 STANDING tBodyAcc.mean().X 0.2766
## 6       1 STANDING tBodyAcc.mean().X 0.2772
```

```
# Step 2: Cast for each subject:
extract.df.melt.dcast <- dcast(extract.df.melt, subject + activity ~ variable, mean)

# Check data:
class(extract.df.melt.dcast)
```

```
## [1] "data.frame"
```

```
dim(extract.df.melt.dcast)
```

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

```
head(extract.df.melt.dcast)
```

```
##   subject           activity tBodyAcc.mean().X tBodyAcc.mean().Y
## 1       1             LAYING            0.2216         -0.040514
## 2       1            SITTING            0.2612         -0.001308
## 3       1           STANDING            0.2789         -0.016138
## 4       1            WALKING            0.2773         -0.017384
## 5       1 WALKING_DOWNSTAIRS            0.2892         -0.009919
## 6       1   WALKING_UPSTAIRS            0.2555         -0.023953
##   tBodyAcc.mean().Z tBodyAcc.std().X tBodyAcc.std().Y tBodyAcc.std().Z
## 1           -0.1132         -0.92806         -0.83683         -0.82606
## 2           -0.1045         -0.97723         -0.92262         -0.93959
## 3           -0.1106         -0.99576         -0.97319         -0.97978
## 4           -0.1111         -0.28374          0.11446         -0.26003
## 5           -0.1076          0.03004         -0.03194         -0.23043
## 6           -0.0973         -0.35471         -0.00232         -0.01948
##   tGravityAcc.mean().X tGravityAcc.mean().Y tGravityAcc.mean().Z
## 1              -0.2489               0.7055              0.44582
## 2               0.8315               0.2044              0.33204
## 3               0.9430              -0.2730              0.01349
## 4               0.9352              -0.2822             -0.06810
## 5               0.9319              -0.2666             -0.06212
## 6               0.8934              -0.3622             -0.07540
##   tGravityAcc.std().X tGravityAcc.std().Y tGravityAcc.std().Z
## 1             -0.8968             -0.9077             -0.8524
## 2             -0.9685             -0.9355             -0.9490
## 3             -0.9938             -0.9812             -0.9763
## 4             -0.9766             -0.9713             -0.9477
## 5             -0.9506             -0.9370             -0.8959
```

```
## 6             -0.9564             -0.9528             -0.9124
##   tBodyAccJerk.mean().X tBodyAccJerk.mean().Y tBodyAccJerk.mean().Z
## 1               0.08109             0.0038382              0.010834
## 2               0.07748            -0.0006191             -0.003368
## 3               0.07538             0.0079757             -0.003685
## 4               0.07404             0.0282721             -0.004168
## 5               0.05416             0.0296504             -0.010972
## 6               0.10137             0.0194863             -0.045563
##   tBodyAccJerk.std().X tBodyAccJerk.std().Y tBodyAccJerk.std().Z
## 1             -0.95848              -0.9241              -0.9549
## 2             -0.98643              -0.9814              -0.9879
## 3             -0.99460              -0.9856              -0.9923
## 4             -0.11362               0.0670              -0.5027
## 5             -0.01228              -0.1016              -0.3457
## 6             -0.44684              -0.3783              -0.7066
##   tBodyGyro.mean().X tBodyGyro.mean().Y tBodyGyro.mean().Z
## 1           -0.01655           -0.06449            0.14869
## 2           -0.04535           -0.09192            0.06293
## 3           -0.02399           -0.05940            0.07480
## 4           -0.04183           -0.06953            0.08494
## 5           -0.03508           -0.09094            0.09009
## 6            0.05055           -0.16617            0.05836
##   tBodyGyro.std().X tBodyGyro.std().Y tBodyGyro.std().Z
## 1           -0.8735          -0.951090           -0.9083
## 2           -0.9772          -0.966474           -0.9414
## 3           -0.9872          -0.987734           -0.9806
## 4           -0.4735          -0.054608           -0.3443
## 5           -0.4580          -0.126349           -0.1247
## 6           -0.5449           0.004105           -0.5072
##   tBodyGyroJerk.mean().X tBodyGyroJerk.mean().Y tBodyGyroJerk.mean().Z
## 1               -0.10727               -0.04152               -0.07405
## 2               -0.09368               -0.04021               -0.04670
## 3               -0.09961               -0.04406               -0.04895
## 4               -0.09000               -0.03984               -0.04613
## 5               -0.07396               -0.04399               -0.02705
## 6               -0.12223               -0.04215               -0.04071
##   tBodyGyroJerk.std().X tBodyGyroJerk.std().Y tBodyGyroJerk.std().Z
## 1               -0.9186               -0.9679               -0.9578
## 2               -0.9917               -0.9895               -0.9879
## 3               -0.9929               -0.9951               -0.9921
## 4               -0.2074               -0.3045               -0.4043
## 5               -0.4870               -0.2388               -0.2688
## 6               -0.6148               -0.6017               -0.6063
##   tBodyAccMag.mean() tBodyAccMag.std() tGravityAccMag.mean()
## 1           -0.84193          -0.79514              -0.84193
## 2           -0.94854          -0.92708              -0.94854
## 3           -0.98428          -0.98194              -0.98428
## 4           -0.13697          -0.21969              -0.13697
## 5            0.02719           0.01988               0.02719
## 6           -0.12993          -0.32497              -0.12993
##   tGravityAccMag.std() tBodyAccJerkMag.mean() tBodyAccJerkMag.std()
## 1              -0.79514               -0.95440              -0.92825
## 2              -0.92708               -0.98736              -0.98412
## 3              -0.98194               -0.99237              -0.99310
```

```
## 4             -0.21969              -0.14143             -0.07447
## 5              0.01988              -0.08945             -0.02579
## 6             -0.32497              -0.46650             -0.47899
##   tBodyGyroMag.mean() tBodyGyroMag.std() tBodyGyroJerkMag.mean()
## 1            -0.87476           -0.8190                -0.9635
## 2            -0.93089           -0.9345                -0.9920
## 3            -0.97649           -0.9787                -0.9950
## 4            -0.16098           -0.1870                -0.2987
## 5            -0.07574           -0.2257                -0.2955
## 6            -0.12674           -0.1486                -0.5949
##   tBodyGyroJerkMag.std() fBodyAcc.mean().X fBodyAcc.mean().Y
## 1                -0.9358          -0.93910          -0.86707
## 2                -0.9883          -0.97964          -0.94408
## 3                -0.9947          -0.99525          -0.97707
## 4                -0.3253          -0.20279           0.08971
## 5                -0.3065           0.03823           0.00155
## 6                -0.6486          -0.40432          -0.19098
##   fBodyAcc.mean().Z fBodyAcc.std().X fBodyAcc.std().Y fBodyAcc.std().Z
## 1           -0.8827         -0.92444         -0.83363         -0.81289
## 2           -0.9592         -0.97641         -0.91728         -0.93447
## 3           -0.9853         -0.99603         -0.97229         -0.97794
## 4           -0.3316         -0.31913          0.05604         -0.27969
## 5           -0.2256          0.02433         -0.11296         -0.29793
## 6           -0.4333         -0.33743          0.02177          0.08596
##   fBodyAcc.meanFreq().X fBodyAcc.meanFreq().Y fBodyAcc.meanFreq().Z
## 1              -0.15879               0.09753               0.08944
## 2              -0.04951               0.07595               0.23883
## 3               0.08652               0.11748               0.24486
## 4              -0.20755               0.11309               0.04973
## 5              -0.30740               0.06322               0.29432
## 6              -0.41874              -0.16070              -0.52011
##   fBodyAccJerk.mean().X fBodyAccJerk.mean().Y fBodyAccJerk.mean().Z
## 1              -0.95707              -0.92246               -0.9481
## 2              -0.98660              -0.98158               -0.9861
## 3              -0.99463              -0.98542               -0.9908
## 4              -0.17055              -0.03523               -0.4690
## 5              -0.02766              -0.12867               -0.2883
## 6              -0.47988              -0.41344               -0.6855
##   fBodyAccJerk.std().X fBodyAccJerk.std().Y fBodyAccJerk.std().Z
## 1             -0.96416              -0.9322              -0.9606
## 2             -0.98749              -0.9825              -0.9883
## 3             -0.99507              -0.9870              -0.9923
## 4             -0.13359               0.1067              -0.5347
## 5             -0.08633              -0.1346              -0.4017
## 6             -0.46191              -0.3818              -0.7260
##   fBodyAccJerk.meanFreq().X fBodyAccJerk.meanFreq().Y
## 1                    0.1324                   0.02451
## 2                    0.2566                   0.04754
## 3                    0.3142                   0.03916
## 4                   -0.2093                  -0.38624
## 5                   -0.2532                  -0.33759
## 6                   -0.3770                  -0.50950
##   fBodyAccJerk.meanFreq().Z fBodyGyro.mean().X fBodyGyro.mean().Y
## 1                  0.024388            -0.8502            -0.9522
```

```
## 2                  0.092392           -0.9762           -0.9758
## 3                  0.138581           -0.9864           -0.9890
## 4                 -0.185530           -0.3390           -0.1031
## 5                  0.009372           -0.3524           -0.0557
## 6                 -0.551104           -0.4926           -0.3195
##   fBodyGyro.mean().Z fBodyGyro.std().X fBodyGyro.std().Y fBodyGyro.std().Z
## 1           -0.90930           -0.8823          -0.95123           -0.9166
## 2           -0.95132           -0.9779          -0.96235           -0.9439
## 3           -0.98077           -0.9875          -0.98711           -0.9823
## 4           -0.25594           -0.5167          -0.03351           -0.4366
## 5           -0.03187           -0.4954          -0.18141           -0.2384
## 6           -0.45360           -0.5659           0.15154           -0.5717
##   fBodyGyro.meanFreq().X fBodyGyro.meanFreq().Y fBodyGyro.meanFreq().Z
## 1              -0.003547               -0.09153              0.0104581
## 2               0.189153                0.06313             -0.0297839
## 3              -0.120293               -0.04472              0.1006076
## 4               0.014784               -0.06577              0.0007733
## 5              -0.100454                0.08255             -0.0756762
## 6              -0.187450               -0.47357             -0.1333739
##   fBodyAccMag.mean() fBodyAccMag.std() fBodyAccMag.meanFreq()
## 1           -0.86177           -0.7983                0.08641
## 2           -0.94778           -0.9284                0.23666
## 3           -0.98536           -0.9823                0.28456
## 4           -0.12862           -0.3980                0.19064
## 5            0.09658           -0.1865                0.11919
## 6           -0.35240           -0.4163               -0.09774
##   fBodyBodyAccJerkMag.mean() fBodyBodyAccJerkMag.std()
## 1                   -0.93330                   -0.9218
## 2                   -0.98526                   -0.9816
## 3                   -0.99254                   -0.9925
## 4                   -0.05712                   -0.1035
## 5                    0.02622                   -0.1041
## 6                   -0.44265                   -0.5331
##   fBodyBodyAccJerkMag.meanFreq() fBodyBodyGyroMag.mean()
## 1                        0.26639                -0.8622
## 2                        0.35185                -0.9584
## 3                        0.42222                -0.9846
## 4                        0.09382                -0.1993
## 5                        0.07649                -0.1857
## 6                        0.08535                -0.3260
##   fBodyBodyGyroMag.std() fBodyBodyGyroMag.meanFreq()
## 1                -0.8243                 -0.1397750
## 2                -0.9322                 -0.0002622
## 3                -0.9785                 -0.0286058
## 4                -0.3210                  0.2688444
## 5                -0.3984                  0.3496139
## 6                -0.1830                 -0.2193034
##   fBodyBodyGyroJerkMag.mean() fBodyBodyGyroJerkMag.std()
## 1                     -0.9424                    -0.9327
## 2                     -0.9898                    -0.9870
## 3                     -0.9948                    -0.9947
## 4                     -0.3193                    -0.3816
## 5                     -0.2820                    -0.3919
## 6                     -0.6347                    -0.6939
```

```
##   fBodyBodyGyroJerkMag.meanFreq()
## 1                          0.1765
## 2                          0.1848
## 3                          0.3345
## 4                          0.1907
## 5                          0.1900
## 6                          0.1143
```

```
# Step 3: Make 'extract.df.melt.dcast' tidy:
final.df <- melt(extract.df.melt.dcast, id=c("subject","activity"), measure.vars=colnames(extract.df.mel

# Check data:
class(final.df)
```

```
## [1] "data.frame"
```

```
dim(final.df)
```

```
## [1] 14220     4
```

```
head(final.df, n=24)
```

```
##    subject           activity          variable  value
## 1        1             LAYING tBodyAcc.mean().X 0.2216
## 2        1            SITTING tBodyAcc.mean().X 0.2612
## 3        1           STANDING tBodyAcc.mean().X 0.2789
## 4        1            WALKING tBodyAcc.mean().X 0.2773
## 5        1 WALKING_DOWNSTAIRS tBodyAcc.mean().X 0.2892
## 6        1   WALKING_UPSTAIRS tBodyAcc.mean().X 0.2555
## 7        2             LAYING tBodyAcc.mean().X 0.2814
## 8        2            SITTING tBodyAcc.mean().X 0.2771
## 9        2           STANDING tBodyAcc.mean().X 0.2779
## 10       2            WALKING tBodyAcc.mean().X 0.2764
## 11       2 WALKING_DOWNSTAIRS tBodyAcc.mean().X 0.2776
## 12       2   WALKING_UPSTAIRS tBodyAcc.mean().X 0.2472
## 13       3             LAYING tBodyAcc.mean().X 0.2755
## 14       3            SITTING tBodyAcc.mean().X 0.2572
## 15       3           STANDING tBodyAcc.mean().X 0.2800
## 16       3            WALKING tBodyAcc.mean().X 0.2756
## 17       3 WALKING_DOWNSTAIRS tBodyAcc.mean().X 0.2924
## 18       3   WALKING_UPSTAIRS tBodyAcc.mean().X 0.2608
## 19       4             LAYING tBodyAcc.mean().X 0.2636
## 20       4            SITTING tBodyAcc.mean().X 0.2715
## 21       4           STANDING tBodyAcc.mean().X 0.2805
## 22       4            WALKING tBodyAcc.mean().X 0.2786
## 23       4 WALKING_DOWNSTAIRS tBodyAcc.mean().X 0.2800
## 24       4   WALKING_UPSTAIRS tBodyAcc.mean().X 0.2709
```

For submission, please upload your data set as a txt file created with write.table() using row.name=FALSE.

```
write.table(final.df, file="./submission_file.table", row.name=FALSE)
```