# README

*Ed Rogers*

*05/20/2015*

This is the README document for the "Getting and Cleaning Data" Course Project. This document can be read directly as an R Markdown document, or as a PDF, which is included in the repository (README.pdf). In this document, the run_analysis.R script is explained step-by-step. This script relies on the dplyr package, v0.4.1 or later.

## Original Data Source

After unzipping the UCI HAR Dataset source file, multiple files and directories of data are available. Data in the Inertial Signals sub-folders can be ignored as feature labels are not made available, and they will not be used in the Course Project.

Between the two folders – "train" and "test" – there are 6 files of data to be loaded into R. There are also 2 files of metadata.

### 6 Files of Data:

- UCI HAR Dataset/train/X_train.txt
- UCI HAR Dataset/train/subject_train.txt
- UCI HAR Dataset/train/y_train.txt
- UCI HAR Dataset/test/X_test.txt
- UCI HAR Dataset/test/subject_test.txt
- UCI HAR Dataset/test/y_test.txt

### 2 Files of Metadata:

- UCI HAR Dataset/features.txt
- UCI HAR Dataset/activity_labels.txt

## Step 1: Merging the files to one dataset with descriptive names for features and activities

### Reading/prepping the metadata

The features.txt file gives the names of each feature in the X_train.txt and X_test.txt files. The activity_labels.txt file gives better factor labels for the y_train.txt and y_test.txt files. The metadata is prepared with the following code:

```
suppressPackageStartupMessages(library(dplyr,quietly = TRUE))

# Read in the names of each of the columns for our data.frame first
featureNames <- c("Subject",
```

```
                  "Activity",
                  read.table(file = "UCI HAR Dataset/features.txt",
                             header = FALSE,
                             stringsAsFactors = FALSE)[,"V2"]
                  )

# Read in the names of the activities to be used as factors
activities <- read.table(file = "UCI HAR Dataset/activity_labels.txt",
                         header = FALSE,
                         stringsAsFactors = FALSE)[,"V2"]
```

### Merging the data into 1 tidy data.frame

The first step of merging the 6 files of data is to combine "train" and "test" data with rbind().

```
# Read in the test and train data from the 6 relevant files
trainData <- read.table("UCI HAR Dataset/train/X_train.txt")
trainSubj <- read.table("UCI HAR Dataset/train/subject_train.txt",colClasses = "factor")
trainActv <- read.table("UCI HAR Dataset/train/y_train.txt",colClasses = "factor")
testData  <- read.table("UCI HAR Dataset/test/X_test.txt")
testSubj  <- read.table("UCI HAR Dataset/test/subject_test.txt",colClasses = "factor")
testActv  <- read.table("UCI HAR Dataset/test/y_test.txt",colClasses = "factor")

# rbind the 6 train and test tables into 3 data.frames:
combinedData <- rbind(trainData,testData)
combinedSubj <- rbind(trainSubj,testSubj)
combinedActv <- rbind(trainActv,testActv)
```

Next, the columns of "Subject", "Activity", and the 561 features are merged with cbind(). This results in a tidy data.frame of 10299 observations of 563 variables.

```
# cbind the three data.frames into one
dataHAR <- cbind(combinedSubj,combinedActv,combinedData)
```

### Applying the metadata

```
# Name the columns
colnames(dataHAR) <- featureNames

# Rename the Activity factors
levels(dataHAR$Activity) <- activities
```

## Step 2: Extract "only the measurements on the mean and standard deviation for each measurement."

For this step, only those features with the word "mean" or "std" are kept in the data.frame. Features based on mean vector angles are also dropped, despite having the word "mean" in their name; they are not strictly speaking averages. Features that give an average in the frequency domain ("meanfreq") are kept. The subsetting is primarily performed using regex.

```r
# Extract "only the measurements on the mean
# and standard deviation for each measurement."
namesWithMean             <- grep("mean",  names(dataHAR),value=TRUE,ignore.case = TRUE)
namesWithMeanButNotAngles <- grep("^angle",namesWithMean, value=TRUE,invert=TRUE)
namesWithStd              <- grep("std",   names(dataHAR),value=TRUE,ignore.case = TRUE)
dataHAR <- dataHAR[,c("Subject","Activity",namesWithMeanButNotAngles,namesWithStd)]
```

## Steps 3 & 4: Descriptive Activity and Variable Names

These steps are already applied during the merging of the data.

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

After grouping, the summarise_each() function will condense each feature to its average for each Subject and Activity. Thus, only 180 values (= 30 Subjects * 6 Activities) are needed for each of the 79 features. The resulting data.frame is 81 columns (= 2 factors + 79 features) by 180 rows.

```r
# Note: As of dplyr 0.4.1, "summarize_each()" (with an
# American spelling) does not exist. Only summarise_each()

dataHARMeansOnly <- dataHAR %>% group_by(Subject,Activity) %>% summarise_each(funs(mean))
```

## Demonstrate what's in this final data.frame by subsetting

This data.frame has 2 factors and 79 features making 81 total columns. There are 180 rows, one for each possible combination of Subject and Activity factors. Below, a few rows are shown for the two factors and two of the features.

```r
featuresOfInterest <- c("Subject","Activity","tBodyAcc-mean()-X","tBodyAcc-std()-X")
dataHARMeansOnly[1:10,featuresOfInterest]
```

```
## Source: local data frame [10 x 4]
## Groups: Subject
##
##    Subject           Activity tBodyAcc-mean()-X tBodyAcc-std()-X
## 1        1            WALKING         0.2773308      -0.28374026
## 2        1   WALKING_UPSTAIRS         0.2554617      -0.35470803
## 3        1 WALKING_DOWNSTAIRS         0.2891883       0.03003534
## 4        1            SITTING         0.2612376      -0.97722901
## 5        1           STANDING         0.2789176      -0.99575990
## 6        1             LAYING         0.2215982      -0.92805647
## 7       11            WALKING         0.2718219      -0.42284207
## 8       11   WALKING_UPSTAIRS         0.2637759      -0.23880296
## 9       11 WALKING_DOWNSTAIRS         0.2916056       0.14245656
## 10      11            SITTING         0.2765902      -0.98279656
```

## Finally, write the data to a text file

```
# For this activity, we will only save the smaller
# 180x81 data.frame.

write.csv(dataHARMeansOnly,file = "TidyAveragesHAR.csv",row.names=FALSE)
```

Note: write.csv replaces special characters in feature names with "." For example "tBodyAcc-mean()-X" becomes "tBodyAcc.mean...X"

## Instructions to read in this file from text

Special care is required in handling the column classes.

```
myColClasses <- c(rep("factor",times=2),rep("numeric",times=79))
fromFileData <- read.csv("TidyAveragesHAR.csv",colClasses=myColClasses)
```

### Demonstrating that data.frame loaded from the CSV looks the same

```
# Demonstrate what's in the CSV by printing
# a subset of the loaded data.frame
featuresOfInterest <- c("Subject","Activity","tBodyAcc.mean...X","tBodyAcc.std...X")
fromFileData[1:10,featuresOfInterest]
```

```
##    Subject           Activity tBodyAcc.mean...X tBodyAcc.std...X
## 1        1            WALKING         0.2773308      -0.28374026
## 2        1   WALKING_UPSTAIRS         0.2554617      -0.35470803
## 3        1 WALKING_DOWNSTAIRS         0.2891883       0.03003534
## 4        1            SITTING         0.2612376      -0.97722901
## 5        1           STANDING         0.2789176      -0.99575990
## 6        1             LAYING         0.2215982      -0.92805647
## 7       11            WALKING         0.2718219      -0.42284207
## 8       11   WALKING_UPSTAIRS         0.2637759      -0.23880296
## 9       11 WALKING_DOWNSTAIRS         0.2916056       0.14245656
## 10      11            SITTING         0.2765902      -0.98279656
```