# Code Book for Project

This file is the codebook for producing the tidy data set required in

the project for the cousera course Getting and Cleaning Data.

## Contents

## Introduction:

This is the codebook for the tidy data set project.

## Process:

1. Load R packages:
    a. dplyr
    b. stringr
2. Create a data directory if does not already exist in the current working directory.
3. If the file does not already exist in ./data/Dataset.zip, then download it using download.file with the mode set to "wb" from https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20HAR%20Dataset.zip" and unzip it. Otherwise, do not download it.
4. The data will be in a directory in the current folder called "UCI HAR Dataset". It contains the files and folders for the analysis.
    a. README.txt
    b. **activity_labels.txt**: These are the factors used to describe the activity the tester was engaging in at the time the measurement was made.
    c. **features.txt**: The variable names for the measurements in X_train.txt and X_test.txt.
    d. **features_info.txt**: A description of the variables in features.txt.
    e. folder – train: contains the training data sets.
        i. **subject_train**: A file listing which tester conducted the test in each row.
        ii. **y_train**: A file listing the activity the tester was engaged each row.
        iii. **X_train**: A file containing the measurements as described in the README.txt file downloaded with the data set.
    f. folder –test : contains the testing datasets:
        i. **subject_test**: A file listing which tester conducted the test in each row.
        ii. **y_test**: A file listing the activity the tester was engaged each row.
        iii. **X_test**: A file containing the measurements as described in the README.txt file
5. Read the **activity_labels.txt** file into the data frame **activity_labels** using **read.csv**.

    a. Assign it the column names **Record** and **ActivityType**.
    b. Print the file to view it contents.

6. Read the activity file **y_train.txt** and assign it to the data frame **y_train**.
    a. Assign it the column name **Activity**
    b. The number of levels in this file is 6 listed as the numbers 1:6.
    c. Rename the levels using the function factor with the levels and names coming from the **activity_labels** data set. As these are the column names that we will use for our data, we will condition these names in the data frame **features. Note:** this will not affect the raw data set file as we are not writing this file back out.
        i. Remove unwanted characters using the local function **f_remove_chars**. This function removes the following characters:
            1. (
            2. )
            3. ,
            4. 1 or more spaces in a row
            5. Hyphen –
            6. Underscore _
            7. Trims any leading and training spaces using **str_trim** from the **stringr** package.
        ii. Since CamelCode seems to the agreed on standard for variable names, then variable names in the df **y_train** will be changed. Let it be noted that some people prefer all lower case. I changed to camel code because of the forums for the course. Also note that after we subset the columns to only the ones that we want, later there will only be three words which need to be fixed.
            1. mean to Mean
            2. std to Std and
            3. gravity to Gravity
        iii. I have chosen to leave the prefixed t and f on all variable names as it is a single character and represents the domain of the variable as to whether it is t for the time domain or f for the frequency domain.
7. The request is to keep only the columns which represent the mean and frequency of the measurements. In order to do this, the conditioned features data frame is subset using grep and assigned to a variable called **columns_to_keep**. In order to help create the code book **columns_to_ keep** is written to a file called **df_columns_to_keep.txt**.
8. Next we read in the subject data into **subject_train** and assign it the column name **tester.**
9. Now we read in the actual measurement variables in **X_table.txt** into **x_train**. Note that this data set is white space delimited. Sometime there is more than one space between variables. Therefore, we use read.table as it allows for any white space between variables if the separator is set to “”.
10. Subset the data into **sub_x_train** using the variable we created above **columns_to_keep**. Now we do not have any issues with duplicate column names. Note that if you do not remove the

characters from the feature variables as we did above the R will treat some of the column names as duplicates.

11. Now we check **sub_x_train** for NA values. I found that there are none in the dataset.
12. Now, the data set **data_train** is created by column binding **subject_train**, **y_train**, and **sub_x_train**. This dataset has 7352 observations and 86 variables.
13. Next we read in the subject data into **subject_test** and assign it the column name **tester.**
14. Now we read in the actual measurement variabes in **X_test.txt** into **x_test**. Note that this data set is white space delimited. Sometime there is more than one space between variables. Therefor we use read.table as it allows for any white space between variables if the separator is set to “”.
15. Subset the data into **sub_x_test** using the variable we created above **columns_to_keep**. Now we do not have any issues with duplicate column names. Note that if you do not remove the characters from the feature variables as we did above the R will treat some of the column names as duplicates.
16. Now we check s**ub_x_test** for NA values. I found that there are none in the dataset.
17. Now, the data set **data_test** is created by column binding **subject_test**, **y_test**, and **sub_x_test**. This dataset has 2947 observations and 86 variables.
18. The next step is to append **rbind data_test** to **data_train. B**efore verified that:
    a. The dimensions of the data sets are the same.
    b. The tester is unique between the two files.
19. Now the two files are combined, so we want to get the mean of the columns. To do this use the **dplyr** package functions to create the tidy data set **out_data**:
    a. Use **tbl_df** to convert data to a data from table which is required for the **dplyr** tools. The output dataset is **tbl_data.** The table has 10,299 observations and 88 variables.
    b. Use **group_by** to group the table by **tester** and **activity**. The output dataset is **g_data**
    c. Use **summarise_each** to apply the mean to the **non grouped** columns. The output dataset is the final dataset **out_data.** The final dataset has 180 observations of 88 variables.
20. Write out **out_data** to the tidy data set file **tidy_data_set.txt.**

The final data set has is of the form:

| Testor | Activity | Mean of 1st variable | Mean of 2nd variable |
|---|---|---|---|
| 1 | WALKING | *value* | *value* |
| 1 | STANDING | *Value* | *value* |

21.

## Data Dictionary: Description of Variables:

**Activity**

Explanation of the variable and valid options:

1. Tester – The person who conducted the test.
    a. Tester has values of 1 to 30 each one representing a different testor.
2. ActivityType -
    a. WALKING
    b. WALKING_UPSTAIRS
    c. WALKING_DOWNSTAIRS
    d. SITTING
    e. STANDING
    f. LAYING
3. tBodyAccMeanX
4. tBodyAccMeanY
5. tBodyAccMeanZ
6. tBodyAccStdX
7. tBodyAccStdY
8. tBodyAccStdZ
9. tGravityAccMeanX
10. tGravityAccMeanY
11. tGravityAccMeanZ
12. tGravityAccStdX
13. tGravityAccStdY
14. tGravityAccStdZ
15. tBodyAccJerkMeanX
16. tBodyAccJerkMeanY
17. tBodyAccJerkMeanZ
18. tBodyAccJerkStdX
19. tBodyAccJerkStdY
20. tBodyAccJerkStdZ
21. tBodyGyroMeanX
22. tBodyGyroMeanY
23. tBodyGyroMeanZ
24. tBodyGyroStdX
25. tBodyGyroStdY
26. tBodyGyroStdZ
27. tBodyGyroJerkMeanX

28. tBodyGyroJerkMeanY
29. tBodyGyroJerkMeanZ
30. tBodyGyroJerkStdX
31. tBodyGyroJerkStdY
32. tBodyGyroJerkStdZ
33. tBodyAccMagMean
34. tBodyAccMagStd
35. tGravityAccMagMean
36. tGravityAccMagStd
37. tBodyAccJerkMagMean
38. tBodyAccJerkMagStd
39. tBodyGyroMagMean
40. tBodyGyroMagStd
41. tBodyGyroJerkMagMean
42. tBodyGyroJerkMagStd
43. fBodyAccMeanX
44. fBodyAccMeanY
45. fBodyAccMeanZ
46. fBodyAccStdX
47. fBodyAccStdY
48. fBodyAccStdZ
49. fBodyAccMeanFreqX
50. fBodyAccMeanFreqY
51. fBodyAccMeanFreqZ
52. fBodyAccJerkMeanX
53. fBodyAccJerkMeanY
54. fBodyAccJerkMeanZ
55. fBodyAccJerkStdX
56. fBodyAccJerkStdY
57. fBodyAccJerkStdZ
58. fBodyAccJerkMeanFreqX
59. fBodyAccJerkMeanFreqY
60. fBodyAccJerkMeanFreqZ
61. fBodyGyroMeanX
62. fBodyGyroMeanY
63. fBodyGyroMeanZ
64. fBodyGyroStdX
65. fBodyGyroStdY
66. fBodyGyroStdZ
67. fBodyGyroMeanFreqX
68. fBodyGyroMeanFreqY
69. fBodyGyroMeanFreqZ

70. fBodyAccMagMean
71. fBodyAccMagStd
72. fBodyAccMagMeanFreq
73. fBodyBodyAccJerkMagMean
74. fBodyBodyAccJerkMagStd
75. fBodyBodyAccJerkMagMeanFreq
76. fBodyBodyGyroMagMean
77. fBodyBodyGyroMagStd
78. fBodyBodyGyroMagMeanFreq
79. fBodyBodyGyroJerkMagMean
80. fBodyBodyGyroJerkMagStd
81. fBodyBodyGyroJerkMagMeanFreq
82. angletBodyAccMeanGravity
83. angletBodyAccJerkMeanGravityMean
84. angletBodyGyroMeanGravityMean
85. angletBodyGyroJerkMeanGravityMean
86. angleXGravityMean
87. angleYGravityMean
88. angleZGravityMean