

# Readme

Whitchurch Muthumani

1/25/2020

## Introduction

**Important information:** *To view a cleaner version of this file open `Readme.pdf`*

This document (`Readme.Rmd`) will guide the reader through the various sections that comprise this week's assignment. It will explain the context of the data; the script to analyze the data; the resultant dataset generated; justify why the resultant dataset is tidy. Finally, it will provide a script to view the tidydata which was generated.

This document is broken up into 4 sections

- Section 1 : Context of the data.
- Section 2 : File types in the project.
- Section 3 : Walkthrough of the `run_Analysis.r` script.
- Section 4 : Justification as to why the `tidydata.txt` is Tidy.
- Section 5 : Script to view the tidy data: `tidydata.txt`
- Explanation for `DataFrameCreator` function

## Section 1 : Context of the data.

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING\_UPSTAIRS, WALKING\_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.<sup>1</sup>

## Section 2: File types in the project.

### Sources of data:

---

<sup>1</sup>Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine. International Workshop of Ambient Assisted Living (IWAAL 2012). Vitoria-Gasteiz, Spain. Dec 2012

File	Description
features_info.txt	Shows information about the variables used on the feature vector.
features.txt	List of all features.
activity_labels.txt	Links the class labels with their activity name.
X_train.txt	Training set.
y_train.txt	Training labels.
X_test.txt	Test set
y_test.txt	Test labels.

### Analysis Script file

This is the script, that takes the tables listed in “Sources of data” and performs analysis on them to generate the tidy data output file.

File	Description
run_analysis.R	Script to generate the tidy data

### Output file

This is the final output generated by the run\_analysis.R script. The file generated is names: tidydata.txt.

File	Description
tidydata.txt	This file contains the tidydata

### Codebook file

This file contains the description of the variables, that make up the columns of the output tidy data set: tidydata.txt. It also contains the units for each of the variables as applicable It contains a description of the variables.

## Section 2: Walkthrough of the run\_Analysis.r script.

**Note to Grader:** Location of file: Week4Project/scripts/run\_analysis.R

**Step 0: Clear the environment variables and load the dplyr package**

```
rm(list = ls())
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
## filter, lag
```

```
## The following objects are masked from 'package:base':
##
## intersect, setdiff, setequal, union
```

This is to ensure that , we have a clean environment, before beginning the execution of our code.

### Step 1: Execute the DataFrameCreator function

This is to initialize the DataFrameCreator function. The function will be explained in greater depth below:

### Step 2: Create the Training Dataset by calling the function DataFrameCreator

```
pathofinterest <- getwd()
pathtolaad1 <- gsub("Week4Project/output", "/Week4Project/data/dataset/train/X_train.txt", pathofinterest)
pathtolaad2 <- gsub("Week4Project/output", "/Week4Project/data/dataset/train/y_train.txt", pathofinterest)
pathtolaad3 <- gsub("Week4Project/output", "/Week4Project/data/dataset/train/subject_train.txt", pathofinterest)
featurestoload <- gsub("Week4Project/output", "/Week4Project/data/dataset/features.txt", pathofinterest)
```

*#Call DataFrameCreator function to get the Trained Dataset*

```
TrainDataSet <- DataFrameCreator(path1 = pathtolaad1, path2 = pathtolaad2, path3 = pathtolaad3, features =
```

We will not look at the top 5 rows of the TrainDataSet: (Number of columns displayed have been cut down for the sake of brevity)

```
## volunteer_id activities tBodyAcc-mean()-X tBodyAcc-mean()-Y tBodyAcc-mean()-Z
## 1 1 5 0.2885845 -0.02029417 -0.1329051
## 2 1 5 0.2784188 -0.01641057 -0.1235202
## 3 1 5 0.2796531 -0.01946716 -0.1134617
## 4 1 5 0.2791739 -0.02620065 -0.1232826
## 5 1 5 0.2766288 -0.01656965 -0.1153619
```

```
## [1] "The dimensions of the TrainDataSet: Rows = 7352 ; Columns = 563;"
```

### Step 3: Create the Test Dataset by calling the function DataFrameCreator

```
pathofinterest <- getwd()
pathtolaad1 <- gsub("Week4Project/output", "/Week4Project/data/dataset/test/X_test.txt", pathofinterest)
pathtolaad2 <- gsub("Week4Project/output", "/Week4Project/data/dataset/test/y_test.txt", pathofinterest)
pathtolaad3 <- gsub("Week4Project/output", "/Week4Project/data/dataset/test/subject_test.txt", pathofinterest)
featurestoload <- gsub("Week4Project/output", "/Week4Project/data/dataset/features.txt", pathofinterest)
```

*#Call DataFrameCreator function to get the Trained Dataset*

```
TestDataSet <- DataFrameCreator(path1 = pathtolaad1, path2 = pathtolaad2, path3 = pathtolaad3, features =
```

We will not look at the top 5 rows of the TestDataSet: (Number of columns displayed have been cut down for the sake of brevity)

```
## volunteer_id activities tBodyAcc-mean()-X tBodyAcc-mean()-Y tBodyAcc-mean()-Z
## 1 2 5 0.2571778 -0.02328523 -0.01465376
## 2 2 5 0.2860267 -0.01316336 -0.11908252
## 3 2 5 0.2754848 -0.02605042 -0.11815167
## 4 2 5 0.2702982 -0.03261387 -0.11752018
## 5 2 5 0.2748330 -0.02784779 -0.12952716
```

```
## [1] "The dimensions of the TestDataSet: Rows = 2947 ; Columns = 563;"
```

#### Step 4: Create the Merged Data Set; Merging Training and Test Datasets

```
MergedDataSet <- rbind(TrainDataSet,TestDataSet)
a=nrow(MergedDataSet)
b= ncol(MergedDataSet)

sprintf("The dimensions of the MergedDataSet: Rows = %i ; Columns = %i;", a, b)
```

```
## [1] "The dimensions of the MergedDataSet: Rows = 10299 ; Columns = 563;"
```

#### Step 5: Extract the mean and SD columns:-

```
str(v1 <- grep("std|mean",names(MergedDataSet)))
```

```
## int [1:79] 3 4 5 6 7 8 43 44 45 46 ...
```

```
ColumnsToSelect <- names(MergedDataSet[v1])
print(ColumnsToSelect)
```

## [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()"
```

**Note:** I have included *MeafFrequency*, as it calculates Mean of Frequencies. This was done willfully and not as an oversight. There has been debate about whether *MEanFrequency* should be included or not. I decided to Err in the side of caution by including it. As it is better to have more variables, than accidentally neglect variables which maybe required in the future.

**Cleaning up of duplicate columns:** There were duplicate columns the code below exposes those

```
## [1] "Total Number of Duplicate Columns = 84"

## [1] "Are there any mean std columns that are duplicated= 0"
```

There are no columns of interests which have been duplicated, we can proceed by just dropping the duplicate columns Therefore instead of wrangling the data further, I take only the columns of interest, ignore the rest.

## Step 6: Use descriptive activity names for activities in the dataset

Before Converting *activityID* to Activity Names

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

Applying the code to convert ActivityID to Activity Names

```
pathofinterest <- getwd()
pathtoload <- gsub("Week4Project/output","Week4Project/data/dataset/activity_labels.txt",pathofinterest)
#pathtoload <- "../Week4Project/data/dataset/activity_labels.txt"
activityDF <- read.csv(pathtoload,header = FALSE,sep=" ")
activityDF <- activityDF%>%rename("activity_id" = "V1")
activityDF <- activityDF%>%rename("activity" = "V2")

#head(MergedDataSet$activities)
#head(activityDF)

#res <- vector(mode="character",length = length(MergedDataSet$activities))

for(X in activityDF$activity_id)
{
  if(grep(X,activityDF$activity_id))
  {
    #print(X)
    #print(as.character(activityDF[X,"activity"]))
    MergedDataSet$activities <- gsub(X,as.character(activityDF[X,"activity"]),MergedDataSet$activities)
```

```
}
}
```

**End Result** *Activity Names should appear in the output*

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

## Step 7: Give more descriptive names for the variables so non-domain experts understand it

*This is achieved by grepping for regular Expression patterns and then subbing them with more verbose wording I used to approach to generate, long verbose variable names, which are easy to understand but large in length The code below demonstrates the grep and sub pattern of string replacement*

```
names(MergedDataSet) <- gsub("-", "", names(MergedDataSet))
names(MergedDataSet) <- gsub("X$", "_AlongXAxis", names(MergedDataSet))
names(MergedDataSet) <- gsub("Y$", "_AlongYAxis", names(MergedDataSet))
names(MergedDataSet) <- gsub("Z$", "_AlongZAxis", names(MergedDataSet))
names(MergedDataSet) <- gsub("std()", "_StandardDeviation", names(MergedDataSet))
names(MergedDataSet) <- gsub("meanFreq()", "_Meanfrequency", names(MergedDataSet))
names(MergedDataSet) <- gsub("[()]", "", names(MergedDataSet))
names(MergedDataSet) <- gsub("mean", "_Mean", names(MergedDataSet))
names(MergedDataSet) <- gsub("Acc", "Acceleration", names(MergedDataSet))
names(MergedDataSet) <- gsub("Mag", "Magnitude", names(MergedDataSet))
names(MergedDataSet) <- gsub("Gyro", "AngularVelocity", names(MergedDataSet))
names(MergedDataSet) <- gsub("~t", "Time", names(MergedDataSet))
names(MergedDataSet) <- gsub("~f", "Frequency", names(MergedDataSet))
names(MergedDataSet) <- gsub("Jerk", "JerkSignal", names(MergedDataSet))
names(MergedDataSet) <- gsub("BodyBody", "Body", names(MergedDataSet))
```

## Step 8: Creating the tidy dataset

*# We group by Volunteer\_id ad activities.*

```
MergedDataSet <- group_by(MergedDataSet, volunteer_id)
MergedDataSet <- group_by(MergedDataSet, activities, add = TRUE)
```

*#Now if we apply summarize\_at, with means as the function to apply. it will summarize the columns of in #by breaking them into sections. The sections are demarcated by (Volunteer\_id and Activity) which is wh #did for us.*

```
vec <- names(MergedDataSet)
vec <- tail(vec, -2)
str(vec)
```

```
## chr [1:79] "TimeBodyAcceleration_Mean_AlongXAxis" ...
```

```
tidySet <- summarize_at(MergedDataSet, .vars = vec , .funs = mean)
```

## Step 9: View the tidydata

```
knitr::kable(tidySet[1:3], caption = "The Tidy Data Set")
```

Table 4: The Tidy Data Set

volunteer_id	activities	TimeBodyAcceleration_Mean_AlongXAxis
1	LAYING	0.2215982
1	SITTING	0.2612376
1	STANDING	0.2789176
1	WALKING	0.2773308
1	WALKING_DOWNSTAIRS	0.2891883
1	WALKING_UPSTAIRS	0.2554617
2	LAYING	0.2813734
2	SITTING	0.2770874
2	STANDING	0.2779115
2	WALKING	0.2764266
2	WALKING_DOWNSTAIRS	0.2776153
2	WALKING_UPSTAIRS	0.2471648
3	LAYING	0.2755169
3	SITTING	0.2571976
3	STANDING	0.2800465
3	WALKING	0.2755675
3	WALKING_DOWNSTAIRS	0.2924235
3	WALKING_UPSTAIRS	0.2608199
4	LAYING	0.2635592
4	SITTING	0.2715383
4	STANDING	0.2804997
4	WALKING	0.2785820
4	WALKING_DOWNSTAIRS	0.2799653
4	WALKING_UPSTAIRS	0.2708767
5	LAYING	0.2783343
5	SITTING	0.2736941
5	STANDING	0.2825444
5	WALKING	0.2778423
5	WALKING_DOWNSTAIRS	0.2935439
5	WALKING_UPSTAIRS	0.2684595
6	LAYING	0.2486565
6	SITTING	0.2767785
6	STANDING	0.2803462
6	WALKING	0.2836589
6	WALKING_DOWNSTAIRS	0.2770453
6	WALKING_UPSTAIRS	0.2682294
7	LAYING	0.2501767
7	SITTING	0.2846746
7	STANDING	0.2827235
7	WALKING	0.2755930
7	WALKING_DOWNSTAIRS	0.2803071
7	WALKING_UPSTAIRS	0.2487069
8	LAYING	0.2612543
8	SITTING	0.2674915
8	STANDING	0.2796210
8	WALKING	0.2746863

volunteer_id	activities	TimeBodyAcceleration_Mean_AlongXAxis
8	WALKING_DOWNSTAIRS	0.2834841
8	WALKING_UPSTAIRS	0.2588802
9	LAYING	0.2591955
9	SITTING	0.2483267
9	STANDING	0.2823101
9	WALKING	0.2785028
9	WALKING_DOWNSTAIRS	0.2959234
9	WALKING_UPSTAIRS	0.2624365
10	LAYING	0.2802306
10	SITTING	0.2706121
10	STANDING	0.2766503
10	WALKING	0.2785741
10	WALKING_DOWNSTAIRS	0.2904016
10	WALKING_UPSTAIRS	0.2671219
11	LAYING	0.2805930
11	SITTING	0.2765902
11	STANDING	0.2777156
11	WALKING	0.2718219
11	WALKING_DOWNSTAIRS	0.2916056
11	WALKING_UPSTAIRS	0.2637759
12	LAYING	0.2601134
12	SITTING	0.2750072
12	STANDING	0.2774058
12	WALKING	0.2771287
12	WALKING_DOWNSTAIRS	0.2815211
12	WALKING_UPSTAIRS	0.2729703
13	LAYING	0.2767164
13	SITTING	0.2743285
13	STANDING	0.2777584
13	WALKING	0.2758831
13	WALKING_DOWNSTAIRS	0.2949076
13	WALKING_UPSTAIRS	0.2582039
14	LAYING	0.2332754
14	SITTING	0.2799906
14	STANDING	0.2805456
14	WALKING	0.2719596
14	WALKING_DOWNSTAIRS	0.2934221
14	WALKING_UPSTAIRS	0.2624211
15	LAYING	0.2894757
15	SITTING	0.2729034
15	STANDING	0.2789158
15	WALKING	0.2738992
15	WALKING_DOWNSTAIRS	0.2801989
15	WALKING_UPSTAIRS	0.2701876
16	LAYING	0.2742272
16	SITTING	0.2807686
16	STANDING	0.2834974
16	WALKING	0.2760236
16	WALKING_DOWNSTAIRS	0.2955868
16	WALKING_UPSTAIRS	0.2559861
17	LAYING	0.2697801
17	SITTING	0.2773570



volunteer_id	activities	TimeBodyAcceleration_Mean_AlongXAxis
17	STANDING	0.2779425
17	WALKING	0.2723419
17	WALKING_DOWNSTAIRS	0.2939183
17	WALKING_UPSTAIRS	0.2526048
18	LAYING	0.2746916
18	SITTING	0.2772700
18	STANDING	0.2784588
18	WALKING	0.2738878
18	WALKING_DOWNSTAIRS	0.2884395
18	WALKING_UPSTAIRS	0.2654012
19	LAYING	0.2726537
19	SITTING	0.2738303
19	STANDING	0.2781723
19	WALKING	0.2739312
19	WALKING_DOWNSTAIRS	0.2626881
19	WALKING_UPSTAIRS	0.2421188
20	LAYING	0.2395079
20	SITTING	0.2780454
20	STANDING	0.2780769
20	WALKING	0.2725893
20	WALKING_DOWNSTAIRS	0.2961444
20	WALKING_UPSTAIRS	0.2520983
21	LAYING	0.2713255
21	SITTING	0.2775396
21	STANDING	0.2769522
21	WALKING	0.2791835
21	WALKING_DOWNSTAIRS	0.3014610
21	WALKING_UPSTAIRS	0.2651945
22	LAYING	0.2799597
22	SITTING	0.2735838
22	STANDING	0.2790539
22	WALKING	0.2788646
22	WALKING_DOWNSTAIRS	0.2844590
22	WALKING_UPSTAIRS	0.2483915
23	LAYING	0.2740380
23	SITTING	0.2733513
23	STANDING	0.2778993
23	WALKING	0.2732119
23	WALKING_DOWNSTAIRS	0.2898974
23	WALKING_UPSTAIRS	0.2499952
24	LAYING	0.2728505
24	SITTING	0.2734757
24	STANDING	0.2803489
24	WALKING	0.2769808
24	WALKING_DOWNSTAIRS	0.2886312
24	WALKING_UPSTAIRS	0.2698811
25	LAYING	0.2507918
25	SITTING	0.2785415
25	STANDING	0.2780137
25	WALKING	0.2789928
25	WALKING_DOWNSTAIRS	0.2913297
25	WALKING_UPSTAIRS	0.2779954

volunteer_id	activities	TimeBodyAcceleration_Mean_AlongXAxis
26	LAYING	0.2716459
26	SITTING	0.2582435
26	STANDING	0.2811270
26	WALKING	0.2792644
26	WALKING_DOWNSTAIRS	0.2792846
26	WALKING_UPSTAIRS	0.2726914
27	LAYING	0.2741025
27	SITTING	0.2739413
27	STANDING	0.2795669
27	WALKING	0.2768495
27	WALKING_DOWNSTAIRS	0.2975442
27	WALKING_UPSTAIRS	0.2657703
28	LAYING	0.2759135
28	SITTING	0.2769776
28	STANDING	0.2777951
28	WALKING	0.2812282
28	WALKING_DOWNSTAIRS	0.2936421
28	WALKING_UPSTAIRS	0.2620058
29	LAYING	0.2872952
29	SITTING	0.2771800
29	STANDING	0.2779651
29	WALKING	0.2719999
29	WALKING_DOWNSTAIRS	0.2931404
29	WALKING_UPSTAIRS	0.2654231
30	LAYING	0.2810339
30	SITTING	0.2683361
30	STANDING	0.2771127
30	WALKING	0.2764068
30	WALKING_DOWNSTAIRS	0.2831906
30	WALKING_UPSTAIRS	0.2714156

```
sprintf("The tidy dataset dimensions are; rows = %i, columns = %i",nrow(tidySet),ncol(tidySet))
```

```
## [1] "The tidy dataset dimensions are; rows = 180, columns = 81"
```

*I am only printing the first 4 columns as, the table is too large to fit all columns inside the page I have printed the dimensions so that the grader can check that the number of rows and columns are correct*

#### Step 10: Writing the tidydata to a file caled tidydata.txt

```
pathofinterest <- getwd()
pathtolaad <- gsub("Week4Project/output","Week4Project/output/tidydata.txt",pathofinterest)
write.csv(tidySet,pathtolaad,row.names = FALSE)
```

## Section 4 : Justification as to why the tidydata.txt is Tidy.

**Note :** To faciitate ease of displaying the table in this section, i will be making the columnnames shorter

```
tidysetcopy <- tidySet
names(tidysetcopy) <- abbreviate(names(tidysetcopy), minlength=2)
knitr::kable(tidysetcopy[1:10,1:5], caption = "Abbreviated column names for ease of display")
```

Table 5: Abbreviated column names for ease of display

v_	ac	TBA_M_AX	TBA_M_AY	TBA_M_AZ
1	LAYING	0.2215982	-0.0405140	-0.1132036
1	SITTING	0.2612376	-0.0013083	-0.1045442
1	STANDING	0.2789176	-0.0161376	-0.1106018
1	WALKING	0.2773308	-0.0173838	-0.1111481
1	WALKING_DOWNSTAIRS	0.2891883	-0.0099185	-0.1075662
1	WALKING_UPSTAIRS	0.2554617	-0.0239531	-0.0973020
2	LAYING	0.2813734	-0.0181587	-0.1072456
2	SITTING	0.2770874	-0.0156880	-0.1092183
2	STANDING	0.2779115	-0.0184208	-0.1059085
2	WALKING	0.2764266	-0.0185949	-0.1055004

```
sprintf("The tidy dataset dimensions are; rows = %i, columns = %i",nrow(tidySet),ncol(tidySet))
```

```
## [1] "The tidy dataset dimensions are; rows = 180, columns = 81"
```

Inorder to satisfy the Tidy Principle 3 things should be satisfied:-

- I. Each variable forms a column.
- II. Each observation forms a row.
- III. Each type of observational unit forms a table.

#### I. Each variable forms a column:

```
knitr::kable(tidysetcopy[1:2,1:5], caption = "Abbreviated column names for ease of display")
```

Table 6: Abbreviated column names for ease of display

v_	ac	TBA_M_AX	TBA_M_AY	TBA_M_AZ
1	LAYING	0.2215982	-0.0405140	-0.1132036
1	SITTING	0.2612376	-0.0013083	-0.1045442

In our dataset that has been output. Every one of the variables:

- TimeBodyAcceleration\_Mean\_AlongXAxis -> abbreviated as -> TBA\_M\_AX
- TimeBodyAcceleration\_Mean\_AlongYAxis -> abbreviated as -> TBA\_M\_AY
- TimeBodyAcceleration\_Mean\_AlongZAxis -> abbreviated as -> TBA\_M\_AZ
- and so on. . . . , has a separate column. Therefore , the 1st rule has been satisfied

## II. Each observation forms a row:

```
knitr::kable(tidysetcopy[10,1:5], caption = "Abbreviated column names for ease of display")
```

Table 7: Abbreviated column names for ease of display

v_	ac	TBA_M_AX	TBA_M_AY	TBA_M_AZ
2	WALKING	0.2764266	-0.0185949	-0.1055004

By observing a single row of the dataset we see that, every observation, has a single row. And each row contains all the attribute variables. Therefore the 2nd rule has been satisfied.

## III. Each type of observational unit forms a table:

```
tibblerepresentation <- as_tibble(tidySet)
print(tibblerepresentation)
```

```
## # A tibble: 180 x 81
##   volunteer_id activities TimeBodyAcceler... TimeBodyAcceler... TimeBodyAcceler...
##         <int> <chr>          <dbl>          <dbl>          <dbl>
## 1             1 LAYING          0.222          -0.0405         -0.113
## 2             1 SITTING          0.261          -0.00131        -0.105
## 3             1 STANDING          0.279          -0.0161        -0.111
## 4             1 WALKING          0.277          -0.0174        -0.111
## 5             1 WALKING_D...      0.289          -0.00992        -0.108
## 6             1 WALKING_U...      0.255          -0.0240        -0.0973
## 7             2 LAYING          0.281          -0.0182        -0.107
## 8             2 SITTING          0.277          -0.0157        -0.109
## 9             2 STANDING          0.278          -0.0184        -0.106
## 10            2 WALKING          0.276          -0.0186        -0.106
## # ... with 170 more rows, and 76 more variables:
## #   TimeBodyAcceleration_StandardDeviation_AlongXAxis <dbl>,
## #   TimeBodyAcceleration_StandardDeviation_AlongYAxis <dbl>,
## #   TimeBodyAcceleration_StandardDeviation_AlongZAxis <dbl>,
## #   TimeGravityAcceleration_Mean_AlongXAxis <dbl>,
## #   TimeGravityAcceleration_Mean_AlongYAxis <dbl>,
## #   TimeGravityAcceleration_Mean_AlongZAxis <dbl>,
## #   TimeGravityAcceleration_StandardDeviation_AlongXAxis <dbl>,
## #   TimeGravityAcceleration_StandardDeviation_AlongYAxis <dbl>,
## #   TimeGravityAcceleration_StandardDeviation_AlongZAxis <dbl>,
## #   TimeBodyAccelerationJerkSignal_Mean_AlongXAxis <dbl>,
## #   TimeBodyAccelerationJerkSignal_Mean_AlongYAxis <dbl>,
## #   TimeBodyAccelerationJerkSignal_Mean_AlongZAxis <dbl>,
## #   TimeBodyAccelerationJerkSignal_StandardDeviation_AlongXAxis <dbl>,
## #   TimeBodyAccelerationJerkSignal_StandardDeviation_AlongYAxis <dbl>,
## #   TimeBodyAccelerationJerkSignal_StandardDeviation_AlongZAxis <dbl>,
## #   TimeBodyAngularVelocity_Mean_AlongXAxis <dbl>,
## #   TimeBodyAngularVelocity_Mean_AlongYAxis <dbl>,
```

```

## # TimeBodyAngularVelocity_Mean_AlongZAxis <dbl>,
## # TimeBodyAngularVelocity_StandardDeviation_AlongXAxis <dbl>,
## # TimeBodyAngularVelocity_StandardDeviation_AlongYAxis <dbl>,
## # TimeBodyAngularVelocity_StandardDeviation_AlongZAxis <dbl>,
## # TimeBodyAngularVelocityJerkSignal_Mean_AlongXAxis <dbl>,
## # TimeBodyAngularVelocityJerkSignal_Mean_AlongYAxis <dbl>,
## # TimeBodyAngularVelocityJerkSignal_Mean_AlongZAxis <dbl>,
## # TimeBodyAngularVelocityJerkSignal_StandardDeviation_AlongXAxis <dbl>,
## # TimeBodyAngularVelocityJerkSignal_StandardDeviation_AlongYAxis <dbl>,
## # TimeBodyAngularVelocityJerkSignal_StandardDeviation_AlongZAxis <dbl>,
## # TimeBodyAccelerationMagnitude_Mean <dbl>,
## # TimeBodyAccelerationMagnitude_StandardDeviation <dbl>,
## # TimeGravityAccelerationMagnitude_Mean <dbl>,
## # TimeGravityAccelerationMagnitude_StandardDeviation <dbl>,
## # TimeBodyAccelerationJerkSignalMagnitude_Mean <dbl>,
## # TimeBodyAccelerationJerkSignalMagnitude_StandardDeviation <dbl>,
## # TimeBodyAngularVelocityMagnitude_Mean <dbl>,
## # TimeBodyAngularVelocityMagnitude_StandardDeviation <dbl>,
## # TimeBodyAngularVelocityJerkSignalMagnitude_Mean <dbl>,
## # TimeBodyAngularVelocityJerkSignalMagnitude_StandardDeviation <dbl>,
## # FrequencyBodyAcceleration_Mean_AlongXAxis <dbl>,
## # FrequencyBodyAcceleration_Mean_AlongYAxis <dbl>,
## # FrequencyBodyAcceleration_Mean_AlongZAxis <dbl>,
## # FrequencyBodyAcceleration_StandardDeviation_AlongXAxis <dbl>,
## # FrequencyBodyAcceleration_StandardDeviation_AlongYAxis <dbl>,
## # FrequencyBodyAcceleration_StandardDeviation_AlongZAxis <dbl>,
## # FrequencyBodyAcceleration_Meanfrequency_AlongXAxis <dbl>,
## # FrequencyBodyAcceleration_Meanfrequency_AlongYAxis <dbl>,
## # FrequencyBodyAcceleration_Meanfrequency_AlongZAxis <dbl>,
## # FrequencyBodyAccelerationJerkSignal_Mean_AlongXAxis <dbl>,
## # FrequencyBodyAccelerationJerkSignal_Mean_AlongYAxis <dbl>,
## # FrequencyBodyAccelerationJerkSignal_Mean_AlongZAxis <dbl>,
## # FrequencyBodyAccelerationJerkSignal_StandardDeviation_AlongXAxis <dbl>,
## # FrequencyBodyAccelerationJerkSignal_StandardDeviation_AlongYAxis <dbl>,
## # FrequencyBodyAccelerationJerkSignal_StandardDeviation_AlongZAxis <dbl>,
## # FrequencyBodyAccelerationJerkSignal_Meanfrequency_AlongXAxis <dbl>,
## # FrequencyBodyAccelerationJerkSignal_Meanfrequency_AlongYAxis <dbl>,
## # FrequencyBodyAccelerationJerkSignal_Meanfrequency_AlongZAxis <dbl>,
## # FrequencyBodyAngularVelocity_Mean_AlongXAxis <dbl>,
## # FrequencyBodyAngularVelocity_Mean_AlongYAxis <dbl>,
## # FrequencyBodyAngularVelocity_Mean_AlongZAxis <dbl>,
## # FrequencyBodyAngularVelocity_StandardDeviation_AlongXAxis <dbl>,
## # FrequencyBodyAngularVelocity_StandardDeviation_AlongYAxis <dbl>,
## # FrequencyBodyAngularVelocity_StandardDeviation_AlongZAxis <dbl>,
## # FrequencyBodyAngularVelocity_Meanfrequency_AlongXAxis <dbl>,
## # FrequencyBodyAngularVelocity_Meanfrequency_AlongYAxis <dbl>,
## # FrequencyBodyAngularVelocity_Meanfrequency_AlongZAxis <dbl>,
## # FrequencyBodyAccelerationMagnitude_Mean <dbl>,
## # FrequencyBodyAccelerationMagnitude_StandardDeviation <dbl>,
## # FrequencyBodyAccelerationMagnitude_Meanfrequency <dbl>,
## # FrequencyBodyAccelerationJerkSignalMagnitude_Mean <dbl>,
## # FrequencyBodyAccelerationJerkSignalMagnitude_StandardDeviation <dbl>,
## # FrequencyBodyAccelerationJerkSignalMagnitude_Meanfrequency <dbl>,
## # FrequencyBodyAngularVelocityMagnitude_Mean <dbl>,

```

```
## #   FrequencyBodyAngularVelocityMagnitude_StandardDeviation <dbl>,
## #   FrequencyBodyAngularVelocityMagnitude_Meanfrequency <dbl>,
## #   FrequencyBodyAngularVelocityJerkSignalMagnitude_Mean <dbl>,
## #   FrequencyBodyAngularVelocityJerkSignalMagnitude_StandardDeviation <dbl>,
## #   FrequencyBodyAngularVelocityJerkSignalMagnitude_Meanfrequency <dbl>
```

The entire dataframe represents the readings that have been taken from the sensor of the Samsung Galaxy phone. As such it represents a single observational unit of data. Therefore the 3rd rule has been satisfied.

**Therefore this tidy data set form we have achieved after running the script is a valid tidy data representation**

## Section 5 : Script to view the tidy data: tidydata.txt

**Note to Grader :**

- In order to view the entire cleaned up tidy data table.
- Run the script: showTidyData.R (located in the output folder path: Week4Project/output/showTidyData.R)
- This will display the entire tidy data set table

## Explanation for DataFrameCreator function

The DataFrameCreator function has: 3 parameters: path1, path2, path3: for Training, Test and Subject text files respectively

```
DataFrameCreator <- function(path1,path2,path3,features {}
```

The first section of the function is for loading the input Data set that contains all the feature variables, using the path1 variable.

```
# ==Load the input Data Set=====#
path1laad <- path1
traininputDF <- read.csv(path1laad,header = FALSE,sep = "")

#Get an idea about the dataFrame structure
head(traininputDF)
nrow(traininputDF)
ncol(traininputDF)
```

The second section of the function is for loading the Activity Data set that contains activities information using the path 2 variable.

```
# ===Load the output/Activity Dataset===#
path2laad <- path2
trainoutputDF <- read.csv(path2laad,header = FALSE,sep = "")
```

The third section of the function is for loading the subject data set using the path 3 variable

```
# ==Load the Volunteer Dataset===#
pathload <- path3
subjecttDF <- read.csv(pathload,header = FALSE,sep = "")
```

The fourth section of the function is for assigning proper names to the DataFrame columns

```
# =====Assign appropriate Variable names to Dataframe columns=====

subjecttDF <- subjecttDF%>%rename("volunteer_id" = names(subjecttDF))
trainoutputDF <- trainoutputDF%>%rename("activities" = names(trainoutputDF))
```

The fifth section is for assigning the proper variable names for each column replacing the V1, V2 V3 etc

```
=====Load the 561 feature vector names from the features.txt file=====
pathload <- features
featureVariableNamesDF <- read.csv(pathload,header = FALSE,sep="\\"")

#Inorder to remove the numerics and perform string operations we convert it to a character type
featureVariableNamesDF$V1 <- as.character(featureVariableNamesDF$V1)
resultantSplit <- strsplit(featureVariableNamesDF$V1," ")
featureVariableNameVector <- sapply(resultantSplit, function(x) x[-1]) # we got back a vector with all

#Finally we apply the variable names to the columns of the appropriate data set
names(traininputDF)[1:ncol(traininputDF)] <- featureVariableNameVector[1:length(featureVariableNameVector)]
```

The sixth section is for combining all 3 dataframes: subject, input and output to create the final dataframe

```
#=Finally combine all 3 dataframes to create the Training Data Set =====#

FinalTrainingDF <- cbind(subjecttDF,trainoutputDF,traininputDF)
```