Getting and Cleaning Data Course Project

A. Used variables and data.

- **originDataColomnsNames** : Contains the column names of the dataset obtained from “features.txt”
- **activitiesLabels** : Contains the activities Labels that will be used in the dataset obtained from “activity_labels.txt”
- **originDataTrain** : Contains the training measurement part of the dataset. Obtained from “train/X_train.txt”
- **originDataTest** : Contains the test measurement part of the dataset. Obtained from “test/X_test.txt”
- **activitiesTrain** : Contains the list of activities corresponding to training measurement part of the dataset. Obtained from “test/y_train.txt”
- **activitiesTest** : Contains the list of activities corresponding to test measurement part of the dataset. Obtained from “test/y_test.txt”
- **subjectTrain** : Contains the list of subjects corresponding to training measurement part of the dataset. Obtained from “test/subject_train.txt”
- **subjectTest** : Contains the list of subjects corresponding to test measurement part of the dataset. Obtained from “test/subject_test.txt”
- **originDataTotal** : row bind of originDataTrain and originDataTest
- **activitiesTotal** : row bind of activitiesTrain and activitiesTest
- **subjectTotal** : row bind of subjectTrain and subjectTest
- **usefulColumnsId** : containts the columns ID of originDataTotal to extract, mean and standard deviation only.
- **extractDataTotal** : extract of originDataTotal
- **labelledActivitiesTotal** : activitiesTotal with format descriptive labels.
- **usefulColumnsName** : extract of originDataColomnsNames corresponding to usefulColumnsId.
- **finalDataSet** : column binding of extractDataTotal with appropriate variables names, labelledActivitiesTotal and subjectTotal
- **averageDataSet** : independent tidy data set with the average of each variable for each activity and each subject. Obtained from finalDataSet.

B. Loading and preprocessing the data.

Library loading

First of all, the needed libraries are loaded.

```
require(dplyr)
library(dplyr)
```

Useful data loading

Then, all needed data are retrieved and stored for future usage.

- Names of columns's Data and labels for activities.

```
originDataColomnsNames <- read.table("features.txt", stringsAsFactors = FALSE)
str(originDataColomnsNames)
```

```
## 'data.frame':    561 obs. of  2 variables:
##  $ V1: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ V2: chr  "tBodyAcc-mean()-X" "tBodyAcc-mean()-Y" "tBodyAcc-mean()-Z" "tBodyAcc-std()-X" ...
```

```
activitiesLabels <- read.table("activity_labels.txt", stringsAsFactors = FALSE)
str(activitiesLabels)
```

```
## 'data.frame':    6 obs. of  2 variables:
##  $ V1: int  1 2 3 4 5 6
##  $ V2: chr  "WALKING" "WALKING_UPSTAIRS" "WALKING_DOWNSTAIRS" "SITTING" ...
```

- Data in original format split in two data.frames.

```
originDataTrain <- read.table("train/X_train.txt", stringsAsFactors = FALSE)
str(originDataTrain, list.len = 10)
```

```
## 'data.frame':    7352 obs. of  561 variables:
##  $ V1  : num  0.289 0.278 0.28 0.279 0.277 ...
##  $ V2  : num  -0.0203 -0.0164 -0.0195 -0.0262 -0.0166 ...
##  $ V3  : num  -0.133 -0.124 -0.113 -0.123 -0.115 ...
##  $ V4  : num  -0.995 -0.998 -0.995 -0.996 -0.998 ...
##  $ V5  : num  -0.983 -0.975 -0.967 -0.983 -0.981 ...
##  $ V6  : num  -0.914 -0.96 -0.979 -0.991 -0.99 ...
##  $ V7  : num  -0.995 -0.999 -0.997 -0.997 -0.998 ...
##  $ V8  : num  -0.983 -0.975 -0.964 -0.983 -0.98 ...
##  $ V9  : num  -0.924 -0.958 -0.977 -0.989 -0.99 ...
##  $ V10 : num  -0.935 -0.943 -0.939 -0.939 -0.942 ...
##   [list output truncated]
```

```
originDataTest <- read.table("test/X_test.txt", stringsAsFactors = FALSE)
str(originDataTest, list.len = 10)
```

```
## 'data.frame':    2947 obs. of  561 variables:
##  $ V1  : num  0.257 0.286 0.275 0.27 0.275 ...
##  $ V2  : num  -0.0233 -0.0132 -0.0261 -0.0326 -0.0278 ...
##  $ V3  : num  -0.0147 -0.1191 -0.1182 -0.1175 -0.1295 ...
##  $ V4  : num  -0.938 -0.975 -0.994 -0.995 -0.994 ...
##  $ V5  : num  -0.92 -0.967 -0.97 -0.973 -0.967 ...
##  $ V6  : num  -0.668 -0.945 -0.963 -0.967 -0.978 ...
##  $ V7  : num  -0.953 -0.987 -0.994 -0.995 -0.994 ...
##  $ V8  : num  -0.925 -0.968 -0.971 -0.974 -0.966 ...
##  $ V9  : num  -0.674 -0.946 -0.963 -0.969 -0.977 ...
##  $ V10 : num  -0.894 -0.894 -0.939 -0.939 -0.939 ...
##   [list output truncated]
```

- Activities linked to Data in original format. Split in two data.frames too.

```r
activitiesTrain <- read.table("train/y_train.txt", stringsAsFactors = FALSE)
str(activitiesTrain)
```

```
## 'data.frame':    7352 obs. of  1 variable:
##  $ V1: int  5 5 5 5 5 5 5 5 5 5 ...
```

```r
activitiesTest <- read.table("test/y_test.txt", stringsAsFactors = FALSE)
str(activitiesTest)
```

```
## 'data.frame':    2947 obs. of  1 variable:
##  $ V1: int  5 5 5 5 5 5 5 5 5 5 ...
```

- Subjects linked to Data in original format. Split in two data.frames too.

```r
subjectTest <- read.table("test/subject_test.txt", stringsAsFactors = FALSE)
str(subjectTest)
```

```
## 'data.frame':    2947 obs. of  1 variable:
##  $ V1: int  2 2 2 2 2 2 2 2 2 2 ...
```

```r
subjectTrain <- read.table("train/subject_train.txt", stringsAsFactors = FALSE)
str(subjectTrain)
```

```
## 'data.frame':    7352 obs. of  1 variable:
##  $ V1: int  1 1 1 1 1 1 1 1 1 1 ...
```

---

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

At first, we apply rbind to test & training data to obtain only one data set

```
originDataTotal <- rbind(originDataTrain, originDataTest)
str(originDataTotal, list.len = 10)
```

```
## 'data.frame':    10299 obs. of  561 variables:
##  $ V1  : num  0.289 0.278 0.28 0.279 0.277 ...
##  $ V2  : num  -0.0203 -0.0164 -0.0195 -0.0262 -0.0166 ...
##  $ V3  : num  -0.133 -0.124 -0.113 -0.123 -0.115 ...
##  $ V4  : num  -0.995 -0.998 -0.995 -0.996 -0.998 ...
##  $ V5  : num  -0.983 -0.975 -0.967 -0.983 -0.981 ...
##  $ V6  : num  -0.914 -0.96 -0.979 -0.991 -0.99 ...
##  $ V7  : num  -0.995 -0.999 -0.997 -0.997 -0.998 ...
##  $ V8  : num  -0.983 -0.975 -0.964 -0.983 -0.98 ...
##  $ V9  : num  -0.924 -0.958 -0.977 -0.989 -0.99 ...
##  $ V10 : num  -0.935 -0.943 -0.939 -0.939 -0.942 ...
##   [list output truncated]
```

Then, we do the same for other data (activites & subject) for future usage.

```
activitiesTotal <- rbind(activitiesTrain, activitiesTest)
str(activitiesTotal)
```

```
## 'data.frame':    10299 obs. of  1 variable:
##  $ V1: int  5 5 5 5 5 5 5 5 5 5 ...
```

```
subjectTotal <- rbind(subjectTrain, subjectTest)
str(subjectTotal)
```

```
## 'data.frame':    10299 obs. of  1 variable:
##  $ V1: int  1 1 1 1 1 1 1 1 1 1 ...
```

---

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

The first step here is to find the ID of the column that we want to extract. Then, just select them.

```
usefulColumnsId <- originDataColomnsNames$V1[grepl('mean\\(\\)|std\\(\\)',
                                                   originDataColomnsNames$V2)]
extractDataTotal <- select(originDataTotal, usefulColumnsId)
str(extractDataTotal, list.len = 10)
```

```
## 'data.frame':    10299 obs. of  66 variables:
##  $ V1  : num  0.289 0.278 0.28 0.279 0.277 ...
##  $ V2  : num  -0.0203 -0.0164 -0.0195 -0.0262 -0.0166 ...
##  $ V3  : num  -0.133 -0.124 -0.113 -0.123 -0.115 ...
##  $ V4  : num  -0.995 -0.998 -0.995 -0.996 -0.998 ...
##  $ V5  : num  -0.983 -0.975 -0.967 -0.983 -0.981 ...
##  $ V6  : num  -0.914 -0.96 -0.979 -0.991 -0.99 ...
##  $ V41 : num  0.963 0.967 0.967 0.968 0.968 ...
##  $ V42 : num  -0.141 -0.142 -0.142 -0.144 -0.149 ...
##  $ V43 : num  0.1154 0.1094 0.1019 0.0999 0.0945 ...
##  $ V44 : num  -0.985 -0.997 -1 -0.997 -0.998 ...
##   [list output truncated]
```

---

3. Uses descriptive activity names to name the activities in the data set.

We start by obtaining a vector containing labelled activities. This vector will be added in the future to the big data set.

```
labelledActivitiesTotal <- activitiesLabels$V2[activitiesTotal$V1]
str(labelledActivitiesTotal)
```

```
##  chr [1:10299] "STANDING" "STANDING" "STANDING" "STANDING" ...
```

---

4. Appropriately labels the data set with descriptive variable names.

In this part, we firstly rename appropriately the variables names.

```
usefulColumnsName <- originDataColomnsNames$V2[usefulColumnsId]
colnames(extractDataTotal) <- usefulColumnsName
str(extractDataTotal, list.len = 10)
```

```
## 'data.frame':    10299 obs. of  66 variables:
##  $ tBodyAcc-mean()-X          : num  0.289 0.278 0.28 0.279 0.277 ...
##  $ tBodyAcc-mean()-Y          : num  -0.0203 -0.0164 -0.0195 -0.0262 -0.0166 ...
##  $ tBodyAcc-mean()-Z          : num  -0.133 -0.124 -0.113 -0.123 -0.115 ...
##  $ tBodyAcc-std()-X           : num  -0.995 -0.998 -0.995 -0.996 -0.998 ...
##  $ tBodyAcc-std()-Y           : num  -0.983 -0.975 -0.967 -0.983 -0.981 ...
##  $ tBodyAcc-std()-Z           : num  -0.914 -0.96 -0.979 -0.991 -0.99 ...
##  $ tGravityAcc-mean()-X       : num  0.963 0.967 0.967 0.968 0.968 ...
##  $ tGravityAcc-mean()-Y       : num  -0.141 -0.142 -0.142 -0.144 -0.149 ...
##  $ tGravityAcc-mean()-Z       : num  0.1154 0.1094 0.1019 0.0999 0.0945 ...
##  $ tGravityAcc-std()-X        : num  -0.985 -0.997 -1 -0.997 -0.998 ...
##   [list output truncated]
```

Then, we bind the 2 following columns : subject and labelled activities.

```
extractDataTotal <- cbind(extractDataTotal, "subject" = subjectTotal$V1)
finalDataSet <- cbind(extractDataTotal, "activity" = labelledActivitiesTotal)
str(finalDataSet, list.len = 10)
```

```
## 'data.frame':    10299 obs. of  68 variables:
##  $ tBodyAcc-mean()-X          : num  0.289 0.278 0.28 0.279 0.277 ...
##  $ tBodyAcc-mean()-Y          : num  -0.0203 -0.0164 -0.0195 -0.0262 -0.0166 ...
##  $ tBodyAcc-mean()-Z          : num  -0.133 -0.124 -0.113 -0.123 -0.115 ...
##  $ tBodyAcc-std()-X           : num  -0.995 -0.998 -0.995 -0.996 -0.998 ...
##  $ tBodyAcc-std()-Y           : num  -0.983 -0.975 -0.967 -0.983 -0.981 ...
##  $ tBodyAcc-std()-Z           : num  -0.914 -0.96 -0.979 -0.991 -0.99 ...
##  $ tGravityAcc-mean()-X       : num  0.963 0.967 0.967 0.968 0.968 ...
##  $ tGravityAcc-mean()-Y       : num  -0.141 -0.142 -0.142 -0.144 -0.149 ...
##  $ tGravityAcc-mean()-Z       : num  0.1154 0.1094 0.1019 0.0999 0.0945 ...
##  $ tGravityAcc-std()-X        : num  -0.985 -0.997 -1 -0.997 -0.998 ...
##   [list output truncated]
```

---

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. Save the result in averageDataSet.txt

By grouping by activities and subject, we can then easily apply mean function :

```
averageDataSet <- finalDataSet %>%
  group_by(activity, subject) %>%
  summarise_each(funs(mean))
str(averageDataSet, list.len = 10)
```

```
## Classes 'grouped_df', 'tbl_df', 'tbl' and 'data.frame':  180 obs. of  68 variables:
##  $ activity                   : Factor w/ 6 levels "LAYING","SITTING",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ subject                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ tBodyAcc-mean()-X          : num  0.222 0.281 0.276 0.264 0.278 ...
##  $ tBodyAcc-mean()-Y          : num  -0.0405 -0.0182 -0.019 -0.015 -0.0183 ...
##  $ tBodyAcc-mean()-Z          : num  -0.113 -0.107 -0.101 -0.111 -0.108 ...
##  $ tBodyAcc-std()-X           : num  -0.928 -0.974 -0.983 -0.954 -0.966 ...
##  $ tBodyAcc-std()-Y           : num  -0.837 -0.98 -0.962 -0.942 -0.969 ...
##  $ tBodyAcc-std()-Z           : num  -0.826 -0.984 -0.964 -0.963 -0.969 ...
##  $ tGravityAcc-mean()-X       : num  -0.249 -0.51 -0.242 -0.421 -0.483 ...
##  $ tGravityAcc-mean()-Y       : num  0.706 0.753 0.837 0.915 0.955 ...
##   [list output truncated]
##  - attr(*, "vars")=List of 1
##   ..$ : symbol activity
##  - attr(*, "drop")= logi TRUE
```

And finally we save it to the averageDataSet.txt

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