Introduction and Assumptions

Goal of Project

The Getting and Cleaning Data Course Project involves rearranging the information contained in a number of text files so that they are embodied in a single data set.

The project also places certain restrictions on the resulting dataset. In particular, it requires that only the mean and standard deviation measures for each row of a subject’s time series and activity be included in the result.

As an adjunct to the data set, the project requires that an independent tidy data set be created that contains only the average of the resulting values calculated over all time series for each subject and activity combination.

In order to properly describe the nature of the effort some background on the source data is necessary.

Source Data Background

The nature of the source data is a collection of measurements made by certain technical operations of a cell phone. The collection of these measurements constitute a study. That study was engineered to provide both training and test datasets, by separating the population of all study participants into test and training, “Study Groups.”

Assumptions made from Reviewing the Source Data

More than the project assignment description, the approach to completing the project came from what seem like a number of inferences made by reviewing the data and the README.txt file for the project. By seeing the correlation between the number of records in the files it was possible to assume the relationships among the files. For example in the training data, after reading in the data:

```
> subjects <- read.table("UCI HAR Dataset/train/subject_train.txt")

> nrow(subjects)

[1] 7352

> xtrain <- read.table("UCI HAR Dataset/train/X_train.txt")

> nrow(xtrain)

[1] 7352

>> ytrain <- read.table("UCI HAR Dataset/train/y_train.txt")

> nrow(ytrain)
```

```
[1] 7352
```

It seems obvious that these files are all ways of describing the same information. What's more, the README.txt states, "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." And we can test to see if that is the actual breakdown of the subjects in the subjects files:

```
> sub_train <- read.table("UCI HAR Dataset/train/subject_train.txt")
> length(unique(sub_train$V1))
[1] 21
> sub_test <- read.table("UCI HAR Dataset/test/subject_test.txt")
> length(unique(sub_test$V1))
[1] 9
```

So, 9:21 = 3:7(test:train); 30% test:70% train, the subjects file in each of the Study_groups seems to identify the participants. And since there are exactly the same number of lines in the subject_train, x_train, and y_train data, it seems reasonable to assume that each line of the subject's data identifies the associated participant on the x_train/y_train data. As to what is contained in that data, a review of those data may provide a way to proceed with understanding the nature of the input (again, with a great deal of assumption and speculation).If we read in those files:

```
> ncol(xtrain)
[1] 561
>
[1] 561
> ncol(ytrain)
[1] 1
> head(ytrain)
  V1
1  5
2  5
3  5
```

```
4  5

5  5

6  5

>
```

Figuring out what y_train represents may be easier than x_train. The README.txt file indicates that ../train/y_train.txt contains training labels. And if we look at {ziproot}/activity_labels.txt:

```
> labels <- read.table("UCI HAR Dataset/activity_labels.txt")

> labels

  V1                 V2
1  1            WALKING
2  2   WALKING_UPSTAIRS
3  3 WALKING_DOWNSTAIRS
4  4            SITTING
5  5           STANDING
6  6             LAYING
```

It seems reasonable to assume that y_train represents the activity identifier for each line of x_train.
This assumption is strengthened by the fact that exactly 6 unique values exist in y_train:

```
> unique(ytrain$V1)
[1] 5 4 6 1 3 2
```

AS for x_train.txt, the 561 variables happen to be exactly the number of identifiers in at {ziproot}/ features.txt:

```
> features <- read.table("UCI HAR Dataset/features.txt")
> nrow(features)
[1] 561
> head(features)
  V1                V2
1  1 tBodyAcc-mean()-X
2  2 tBodyAcc-mean()-Y
3  3 tBodyAcc-mean()-Z
4  4  tBodyAcc-std()-X
5  5  tBodyAcc-std()-Y
```

```
6  6  tBodyAcc-std()-Z
>
```

From having read the information available about the study it now seems reasonable to propose a hypothesis that describes the information we are trying to encapsulate. The x_{study group}.txt files represent the capture measurements of the data for each time series value; The y_{study group}.txt files contain the identification of the activity that is being measured for the time series point; the subject_{study group}.txt files indicate the participant for each time series point. All that seems to be missing is the actual time that the activity was measured for the participant, and that is not necessary to accomplish the project. If that hypothesis is correct we should see the same relationships between the record counts of the test files that we saw in the train files.

```
> xtest <- read.table("UCI HAR Dataset/test/X_test.txt")
> nrow(xtest)
[1] 2947

> sub_test <- read.table("UCI HAR Dataset/test/subject_test.txt")
> nrow(sub_test)
[1] 2947

> act_test <- read.table("UCI HAR Dataset/test/y_test.txt")
> nrow(act_test)
[1] 2947
>
```

And the number of features should be the same between the two study groups:

```
> ncol(xtest)
[1] 561
>
```

This implies a strategy to create a single data set that places all this information in structure that will make it useful to extract it for our needs.

Combined Human Activity Project Data Store

The requirements for this Class project are stated as this:

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

2. Extract only the measurements on the mean and standard deviation for each measurement.
3. Use descriptive activity names to name the activities in the data set
4. Appropriately label the data set with descriptive variable names.
5. Create a second, independent tidy data set with the average of each variable for each activity and each subject.

I propose the following structure to support the requirements:

Human Activity Project Data Store Data Dictionary

The data store contains the following fields. Each line represents one point in a time series, the same level of granularity as the incoming X_train.txt and X_test.txt data:

Field	Variable Name	Type	Description	Valid Values
1	subject	Factor	Numeric ID of subject as indicated on {studygroup}/subject_{studygroup}.txt	1 :30
2	study_group	Factor (2)	Identifies the study group (test/train) to which the subject belongs	"Test", "Train"
3	Activity	Char (18)	Activity for this time series point	"WALKING" "WALKING_UPSTAIRS" "WALKING_DOWNSTAIRS" "SITTING" "STANDING" "LAYING"
4	tBodyAcc-mean()-X	num	measure	(-1,1)
5	tBodyAcc-mean()-Y	num	measure	(-1,1)
6	tBodyAcc-mean()-Z	num	measure	(-1,1)
7	tBodyAcc-std()-X	num	measure	(-1,1)
8	tBodyAcc-std()-Y	num	measure	(-1,1)
9	tBodyAcc-std()-Z	num	measure	(-1,1)
10	tGravityAcc-mean()-X	num	measure	(-1,1)
11	tGravityAcc-mean()-Y	num	measure	(-1,1)
12	tGravityAcc-mean()-Z	num	measure	(-1,1)
13	tGravityAcc-std()-X	num	measure	(-1,1)
14	tGravityAcc-std()-Y	num	measure	(-1,1)
15	tGravityAcc-std()-Z	num	measure	(-1,1)
16	tBodyAccJerk-mean()-X	num	measure	(-1,1)
17	tBodyAccJerk-mean()-Y	num	measure	(-1,1)

18	tBodyAccJerk-mean()-Z	num	measure	(-1,1)
19	tBodyAccJerk-std()-X	num	measure	(-1,1)
20	tBodyAccJerk-std()-Y	num	measure	(-1,1)
21	tBodyAccJerk-std()-Z	num	measure	(-1,1)
22	tBodyGyro-mean()-X	num	measure	(-1,1)
23	tBodyGyro-mean()-Y	num	measure	(-1,1)
24	tBodyGyro-mean()-Z	num	measure	(-1,1)
25	tBodyGyro-std()-X	num	measure	(-1,1)
26	tBodyGyro-std()-Y	num	measure	(-1,1)
27	tBodyGyro-std()-Z	num	measure	(-1,1)
28	tBodyGyroJerk-mean()-X	num	measure	(-1,1)
29	tBodyGyroJerk-mean()-Y	num	measure	(-1,1)
30	tBodyGyroJerk-mean()-Z	num	measure	(-1,1)
31	tBodyGyroJerk-std()-X	num	measure	(-1,1)
32	tBodyGyroJerk-std()-Y	num	measure	(-1,1)
33	tBodyGyroJerk-std()-Z	num	measure	(-1,1)
34	tBodyAccMag-mean()	num	measure	(-1,1)
35	tBodyAccMag-std()	num	measure	(-1,1)
36	tGravityAccMag-mean()	num	measure	(-1,1)
37	tGravityAccMag-std()	num	measure	(-1,1)
38	tBodyAccJerkMag-mean()	num	measure	(-1,1)
39	tBodyAccJerkMag-std()	num	measure	(-1,1)
40	tBodyGyroMag-mean()	num	measure	(-1,1)
41	tBodyGyroMag-std()	num	measure	(-1,1)
42	tBodyGyroJerkMag-mean()	num	measure	(-1,1)
43	tBodyGyroJerkMag-std()	num	measure	(-1,1)
44	fBodyAcc-mean()-X	num	measure	(-1,1)
45	fBodyAcc-mean()-Y	num	measure	(-1,1)
46	fBodyAcc-mean()-Z	num	measure	(-1,1)
47	fBodyAcc-std()-X	num	measure	(-1,1)
48	fBodyAcc-std()-Y	num	measure	(-1,1)
49	fBodyAcc-std()-Z	num	measure	(-1,1)
50	fBodyAccJerk-mean()-X	num	measure	(-1,1)
51	fBodyAccJerk-mean()-Y	num	measure	(-1,1)
52	fBodyAccJerk-mean()-Z	num	measure	(-1,1)
53	fBodyAccJerk-std()-X	num	measure	(-1,1)
54	fBodyAccJerk-std()-Y	num	measure	(-1,1)
55	fBodyAccJerk-std()-Z	num	measure	(-1,1)
56	fBodyGyro-mean()-X fBodyGyro-	num	measure	(-1,1)

57	mean()-Y	num	measure	(-1,1)
58	fBodyGyro-mean()-Z	num	measure	(-1,1)
59	fBodyGyro-std()-X	num	measure	(-1,1)
60	fBodyGyro-std()-Y	num	measure	(-1,1)
61	fBodyGyro-std()-Z	num	measure	(-1,1)
62	fBodyAccMag-mean()	num	measure	(-1,1)
63	fBodyAccMag-std()	num	measure	(-1,1)
64	fBodyBodyAccJerkMag-mean()	num	measure	(-1,1)
65	fBodyBodyAccJerkMag-std()	num	measure	(-1,1)
66	fBodyBodyGyroMag-mean()	num	measure	(-1,1)
67	fBodyBodyGyroMag-std()	num	measure	(-1,1)
68	fBodyBodyGyroJerkMag-mean()	num	measure	(-1,1)
69	fBodyBodyGyroJerkMag-std()	num	measure	(-1,1)

Aesthetic Approach to the Data Store and script

One of the requirements of the project is to produce a, “Tidy,” data set for the store. I believe the project data store represents an efficient way to retain the human activity data assigned for the project. It is not highly normalized, but my experience from dealing with in service data is that high degrees of normalization serve more to save of storage space than to support efficiency of processing. While it is true that the project puts a bit of strain on the memory space used by R, it is unlikely that moving an equivalent amount of data into memory would not be required no matter how the process was engineered if the same information was required. I have found that there is a great deal of efficiency that is gained by having the the categorical variables (subject, study_group and Activity) on each time series point. For example, looking at the code, you can notice that a single dplyr function; avetestdata <- project_data_store %>% group_by (subject, study_group, activity) %>% summarise_each(funs(mean)); completes the entire last requirement of the project (apart from a more reasonable naming of the output variables). I believe this decision would yield similar efficiencies when the store was queried for any other information. And, as it does not violate any of the requirements for, “Tidy,” data sets (each variable represents exactly one item of information,etc.,...) I feel it is a strong design to meet the needs of the project.

License:

========

Use of the source dataset in publications must be acknowledged by referencing the following publication [1]

[1] 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

This dataset is distributed AS-IS and no responsibility implied or explicit can be addressed to the authors or their institutions for its use or misuse. Any commercial use is prohibited.

Jorge L. Reyes-Ortiz, Alessandro Ghio, Luca Oneto, Davide Anguita. November 2012.