Strategy and Code Description

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

We need to bring the data into R data frames.

First, we load the TRAINING files:

Note: The suffix [_df] denotes a Data Frame

- Subject Train (with validation of the number of lines loaded)

```
subject_train_df <- read.table("./Project/UCI HAR Dataset/train/subject_train.txt")
print("subject_train_df: Number of lines and columns loaded")
dim(subject_train_df)
print("subject_train_df: List of unique values")
unique(subject_train_df)
print("subject_train_df: Number of unique values")
nrow(unique(subject_train_df))
```

- X Train (with validation of the number of lines loaded)

```
x_train_df <- read.table("./Project/UCI HAR Dataset/train/x_train.txt")
print("x_train_df: Number of lines and columns loaded")
dim(x_train_df)
```

- Y Train (with validation of the number of lines loaded)

```
y_train_df <- read.table("./Project/UCI HAR Dataset/train/y_train.txt")
print("y_train_df: Number of lines and columns loaded")
dim(y_train_df)
print("y_train_df: List of unique values")
unique(y_train_df)
print("y_train_df: Number of unique values")
nrow(unique(y_train_df))
```

By looking at the unique values of the data frame subject_train_df (values within 1 and 30) and the number of these values (21, which is 70% of 30 subjects) we can safely assume that this file contains the ids of the subjects that were used for training the model.

By looking at the number of rows (7352) and columns (561) in the data frame x_train_df we can safely assume that this data frame contains the observations for the 561 features analyzed in the experiment

Finally, by looking at the unique values of the data frame y_train_df (from 1 to 6) and the number of these values (6) we can safely assume that this file contains the ids of the activities that were used in the experiment (i.e. WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING and LAYING)...

Also, since the number of lines is the same in the 3 data frames, we can also assume that each line in each data frame, goes along the same line number in the other data frames. With this in mind, we can safely proceed to merge these 3 files together to get a tidy dataset for training where:

- Each variable forms a column (i.e. the 561 features from the x_train_df are in their respective columns).
- Each observation forms a row (each one of the 7352 observations in the 3 files are all contained in the merged tidy data set).
- Each type of observational unit forms a table (All the features are all related to the same observation).

Now, we can merge the training data together: By using cbind() we can safely merge all 3 files without messing with the original order, thus not changing the experiment results

```
train_merged_df <- cbind(subject_train_df, y_train_df, x_train_df)
```

Now is the turn to load the TEST files into R data frames

- Subject Test (with validation of the number of lines loaded)

```
subject_test_df <- read.table("./Project/UCI HAR Dataset/test/subject_test.txt")

print("subject_test_df: Number of lines and columns loaded")

dim(subject_test_df)

print("subject_test_df: List of unique values")

unique(subject_test_df)

print("subject_test_df: Number of unique values")

nrow(unique(subject_test_df))
```

- X Test (with validation of the number of lines loaded)

```
x_test_df <- read.table("./Project/UCI HAR Dataset/test/x_test.txt")

print("x_test_df: Number of lines and columns loaded")

dim(x_test_df)
```

- Y Test (with validation of the number of lines loaded)

```
y_test_df <- read.table("./Project/UCI HAR Dataset/test/y_test.txt")

print("y_test_df: Number of lines and columns loaded")

dim(y_test_df)

print("y_test_df: List of unique values")

unique(y_test_df)

print("y_test_df: Number of unique values")

nrow(unique(y_test_df))
```

By looking at the unique values of the data frame subject_test_df (values within 1 and 30) and the number of these values (9, which is 30% of 30 subjects) we can safely assume that this file contains the ids of the subjects that were used for testing the model...

By looking at the number of rows (2947) and columns (561) in the data frame x_test_df we can safely assume that this data frame contains the observations for the 561 features analyzed in the experiment

Finally, by looking at the unique values of the data frame y_test_df (from 1 to 6) and the number of these values (6) we can safely assume that this file contains the ids of the activities that were used in the experiment (i.e. WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING and LAYING)...

Also, since the number of lines is the same in the 3 data frames, we can also assume that each line in each data frame, goes along the same line number in the other data frames. With this in mind, we can safely proceed to merge these 3 files together to get a tidy dataset for training where:

- Each variable forms a column (i.e. the 561 features from the x_test_df are in their respective columns).

- Each observation forms a row (each one of the 2947 observations in the 3 files are all contained in the merged tidy data set).
- Each type of observational unit forms a table (All the features are all related to the same observation).

Now, we can merge the test data together: By using cbind() we can safely merge all 3 files without messing with the original order, thus not changing the experiment results

```
test_merged_df <- cbind(subject_test_df, y_test_df, x_test_df)
```

Finally, we can merge the test and training data sets together to form one data set:

Since both data frames have the exact same variables (i.e. subject id, activity id and the 561 features) we can safely merge them by using the rbind() command

```
merged_df <- rbind(train_merged_df, test_merged_df)
```

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

In order to do this, we will assume that the order of the columns in the X_train_df and x_test_df is exactly the same as the list of features from the file "features.txt". With this in mind, we can load that file into an R data frame so we can use it to extract the right columns from the merged data frame

First, let's load the list of features from the features file

```
features_df <- read.table("./Project/UCI HAR Dataset/features.txt",
stringsAsFactors = FALSE)
```

Once we have this data frame, we can convert it to a vector, as it will be easier to use for renaming the columns of the merged data frame. The idea is to have a vector with all the column names in the desired order; so we need to initialize it with the names of the first 2 columns, i.e. subjectid and activityid

Now, let’s create a vector with the column names

```
colnames_vector <- c("subjectid", "activityid")
```

Then, we append the names of the 561 features that come from the data frame that was used to load the features file

```
colnames_vector <- append(colnames_vector, features_df[,2])
```

Now, by listing the names of the columns of the merged data frame, we can see that the first 3 columns are named, respectively, "V1","V2" and "V1".

```
names(merged_df)
```

Oops, the first and the third columns have the same name... we have to fix this because it doesn't comply with the 3 tidy data principles, and it will confuss R, so:

We need to change the name of the first 2 columns of the merged data frame

```
names(merged_df)[1:2] <- colnames_vector[1:2]
```

Extracts only the means and the std devs into a new data frame

Finally, we can now extract the data we need from the merged data frame. Here, we are really going to use the vector that we created before, since it will aloow us to use the grep() command to match the column names against regular expressions, which will assure that we gett all and only those columns that have the mean and the standard deviation for each variable (these columns are labeled with a mean() and std() suffix as explained in the features_info.txt file)

Let`s create a new data frame with only the columns needed

```
mean_std_features_df <- select(merged_df, 1:2, grep("mean()|std()",
colnames_vector))
```

By combining the select() command from the dplyr library with the grep() command we were able to get only those columns that we were interested on

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

Our strategy here is to bring the names of the activities from the data provided, and use this list to find the ids in the merged data frame and add the right activity name (i.e. WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING and LAYING)

Load the activities data into R data frames

```
activity_labels_df <- read.table("./Project/UCI HAR
Dataset/activity_labels.txt", stringsAsFactors = FALSE)
```

In order to bring the activity names into the merged data frame, we can use the function inner_join(). This command, by default, uses the columns with the same name in the tables to be joined, so we let's make sure that we have the same names for the activity id column

Change the column names of the activities data loaded

```
colnames(activity_labels_df) <- c("activityid", "activityname")
```

Now, we can join both tables by the column activityid

```
mean_std_features_df <- inner_join(mean_std_features_df,
activity_labels_df)
```

The function inner_join() appends all the columns of the second table to the right of the first table... This works, but it would be nicer if we had the columns activity id and activity name together, so we'll use the select() function from the dplyr package to reorganize our resulting table

Reorganize the table

```
mean_std_features_df <- select(mean_std_features_df, 1,2,82,3:81)
```

What we did was to bring column 83 (i.e. activity name) to the third position, and then everything from there to the right. Now we have a nicer resulting data frame

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

Gives the names to the columns. Is in this step that we will make use of the vector with the names of the features that we created in a previous step. We just pass the names of the columns that we want from this vector to the colnames function, taking into account to use the grep() command to select the right ones.

```
colnames(mean_std_features_df)[4:82] <-
colnames_vector[grep("mean()|std()", colnames_vector)]
```

5. From the data set in step 4, create a second, independent tidy data set with the average of each variable for each activity and each subject.

First, let's create a dyplr data frame so we can group it by the subject and activity

```
mean_std_features_dt <- tbl_df(mean_std_features_df)
```

Then, let's create another data frame that's already grouped by the subject and activity

```
features_by_activity_subject <- group_by(mean_std_features_dt, subjectid,
activityid, activityname)
```

Finally, let's calculate the mean for every variable and create a new data frame with the results

```
summ_features_by_activity_subject <-
summarize_all(features_by_activity_subject, mean)
```

Variables Description

subjectid 1

Subject ID

1..30 .Unique Identifier of the experiment subject that produced the observation

activityid 1

Activity ID

1..6 .Unique Identifier of the activity that experiment subject that produced the observation was doing

activityname 20

Activity Name

1 WALKING

2 WALKING_UPSTAIRS

3 WALKING_DOWNSTAIRS

4 SITTING

5 STANDING

6 LAYING

tBodyAcc-mean()-X

Mean by subject and activity of the Mean of X-axis Body Linear Acceleration (time)

tBodyAcc-mean()-Y

Mean by subject and activity of the Mean of Y-axis Body Linear Acceleration (time)

tBodyAcc-mean()-Z

Mean by subject and activity of the Mean of Z-axis Body Linear Acceleration (time) tBodyAcc-std()-X

Mean by subject and activity of the Standard Deviation of X-axis Body Linear Acceleration (time)

tBodyAcc-std()-Y

Mean by subject and activity of the Standard Deviation of Y-axis Body Linear Acceleration (time)

tBodyAcc-std()-Z

Mean by subject and activity of the Standard Deviation of Z-axis Body Linear Acceleration (time)

tGravityAcc-mean()-X

Mean by subject and activity of the Mean of X-axis Gravitational Acceleration (time)

tGravityAcc-mean()-Y

Mean by subject and activity of the Mean of Y-axis Gravitational Acceleration (time)

tGravityAcc-mean()-Z

Mean by subject and activity of the Mean of Z-axis Gravitational Acceleration (time)

tGravityAcc-std()-X

Mean by subject and activity of the Standard Deviation of X-axis Gravitational Acceleration (time)

tGravityAcc-std()-Y

Mean by subject and activity of the Standard Deviation of X-axis Gravitational Acceleration (time)

tGravityAcc-std()-Z

Mean by subject and activity of the Standard Deviation of X-axis Gravitational Acceleration (time)

tBodyAccJerk-mean()-X

Mean by subject and activity of the Mean of Jerk signal of X-axis Body Linear Acceleration (time)

tBodyAccJerk-mean()-Y

Mean by subject and activity of the Mean of Jerk signal of Y-axis Body Linear Acceleration (time)

tBodyAccJerk-mean()-Z

Mean by subject and activity of the Mean of Jerk signal of Z-axis Body Linear Acceleration (time)

tBodyAccJerk-std()-X

Mean by subject and activity of the Standard Deviation of Jerk signal of X-axis Body Linear Acceleration (time)

tBodyAccJerk-std()-Y

Mean by subject and activity of the Standard Deviation of Jerk signal of Y-axis Body Linear Acceleration (time)

tBodyAccJerk-std()-Z

Mean by subject and activity of the Standard Deviation of Jerk signal of Z-axis Body Linear Acceleration (time)

tBodyGyro-mean()-X

Mean by subject and activity of the Mean of X-axis Angular Velocity (time)

tBodyGyro-mean()-Y

Mean by subject and activity of the Mean of Y-axis Angular Velocity (time)

tBodyGyro-mean()-Z

Mean by subject and activity of the Mean of Z-axis Angular Velocity (time)

tBodyGyro-std()-X

Mean by subject and activity of the Standard Deviation of X-axis Angular Velocity (time)

tBodyGyro-std()-Y

Mean by subject and activity of the Standard Deviation of Y-axis Angular Velocity (time)

tBodyGyro-std()-Z

Mean by subject and activity of the Standard Deviation of Z-axis Angular Velocity (time)

tBodyGyroJerk-mean()-X

Mean by subject and activity of the Mean of Jerk signal of X-axis Angular Velocity (time)

tBodyGyroJerk-mean()-Y

Mean by subject and activity of the Mean of Jerk signal of Y-axis Angular Velocity (time)

tBodyGyroJerk-mean()-Z

Mean by subject and activity of the Mean of Jerk signal of Z-axis Angular Velocity (time)

tBodyGyroJerk-std()-X

Mean by subject and activity of the Standard Deviation of Jerk signal of X-axis Angular Velocity (time)

tBodyGyroJerk-std()-Y

Mean by subject and activity of the Standard Deviation of Jerk signal of Y-axis Angular Velocity (time)

tBodyGyroJerk-std()-Z

Mean by subject and activity of the Standard Deviation of Jerk signal of Z-axis Angular Velocity (time)

tBodyAccMag-mean()

Mean by subject and activity of the Mean of the magnitud (calculated using Euclidean norm) for Body Linear Acceleration (time)

tBodyAccMag-std()

Mean by subject and activity of the Standard Deviation of the magnitud (calculated using Euclidean norm) for Body Linear Acceleration (time)

tGravityAccMag-mean()

Mean by subject and activity of the Mean of the magnitud (calculated using Euclidean norm) for Gravity Acceleration (time)

tGravityAccMag-std()

Mean by subject and activity of the Standard Deviation of the magnitud (calculated using Euclidean norm) for Gravity Acceleration (time)

tBodyAccJerkMag-mean()

Mean by subject and activity of the Mean of the magnitud (calculated using Euclidean norm) for the Jerk signal of Body Linear Acceleration (time)

tBodyAccJerkMag-std()

Mean by subject and activity of the Standard Deviation of the magnitud (calculated using Euclidean norm) for the Jerk signal of Body Linear Acceleration (time)

tBodyGyroMag-mean()

Mean by subject and activity of the Mean of the magnitud (calculated using Euclidean norm) for Angular Velocity (time)

tBodyGyroMag-std()

Mean by subject and activity of the Standard Deviation of the magnitud (calculated using Euclidean norm) for Angular Velocity (time)

tBodyGyroJerkMag-mean()

Mean by subject and activity of the Mean of the magnitud (calculated using Euclidean norm) for the Jerk signal of Angular Velocity (time)

tBodyGyroJerkMag-std()

Mean by subject and activity of the Standard Deviation of the magnitud (calculated using Euclidean norm) for the Jerk signal of Angular Velocity (time)

fBodyAcc-mean()-X

Mean by subject and activity of the Mean of X-axis Body Linear Acceleration (frequency)

fBodyAcc-mean()-Y

Mean by subject and activity of the Mean of Y-axis Body Linear Acceleration (frequency)

fBodyAcc-mean()-Z

Mean by subject and activity of the Mean of Z-axis Body Linear Acceleration (frequency)

fBodyAcc-std()-X

Mean by subject and activity of the Standard Deviation of X-axis Body Linear Acceleration (frequency)

fBodyAcc-std()-Y

Mean by subject and activity of the Standard Deviation of Y-axis Body Linear Acceleration (frequency) fBodyAcc-std()-Z

Mean by subject and activity of the Standard Deviation of Z-axis Body Linear Acceleration (frequency)

fBodyAcc-meanFreq()-X

Mean by subject and activity of the Mean of the weighted average of the frequency components of the X-axis Body Linear Acceleration (frequency)

fBodyAcc-meanFreq()-Y

Mean by subject and activity of the Mean of the weighted average of the frequency components of the Y-axis Body Linear Acceleration (frequency)

fBodyAcc-meanFreq()-Z

Mean by subject and activity of the Mean of the weighted average of the frequency components of the Z-axis Body Linear Acceleration (frequency)

fBodyAccJerk-mean()-X

Mean by subject and activity of the Mean of Jerk signal of X-axis Body Linear Acceleration (frequency)

fBodyAccJerk-mean()-Y

Mean by subject and activity of the Mean of Jerk signal of Y-axis Body Linear Acceleration (frequency)

fBodyAccJerk-mean()-Z

Mean by subject and activity of the Mean of Jerk signal of Z-axis Body Linear Acceleration (frequency)

fBodyAccJerk-std()-X

Mean by subject and activity of the Standard Deviation of Jerk signal of X-axis Body Linear Acceleration (frequency) fBodyAccJerk-std()-Y

Mean by subject and activity of the Standard Deviation of Jerk signal of Y-axis Body Linear Acceleration (frequency)

fBodyAccJerk-std()-Z

Mean by subject and activity of the Standard Deviation of Jerk signal of Z-axis Body Linear Acceleration (frequency)

fBodyAccJerk-meanFreq()-X

Mean by subject and activity of the Mean of the weighted average of the frequency components of the Jerk signal for the X-axis Body Linear Acceleration (frequency)

fBodyAccJerk-meanFreq()-Y

Mean by subject and activity of the Mean of the weighted average of the frequency components of the Jerk signal for the Y-axis Body Linear Acceleration (frequency)

fBodyAccJerk-meanFreq()-Z

Mean by subject and activity of the Mean of the weighted average of the frequency components of the Jerk signal for the Z-axis Body Linear Acceleration (frequency)

fBodyGyro-mean()-X

Mean by subject and activity of the Mean of X-axis Angular Velocity (frequency)

fBodyGyro-mean()-Y

Mean by subject and activity of the Mean of Y-axis Angular Velocity (frequency)

fBodyGyro-mean()-Z

Mean by subject and activity of the Mean of Z-axis Angular Velocity (frequency) fBodyGyro-std()-X

Mean by subject and activity of the Standard Deviation of X-axis Angular Velocity (frequency)

fBodyGyro-std()-Y

Mean by subject and activity of the Standard Deviation of Y-axis Angular Velocity (frequency)

fBodyGyro-std()-Z

Mean by subject and activity of the Standard Deviation of Z-axis Angular Velocity (frequency)

fBodyGyro-meanFreq()-X

Mean by subject and activity of the Mean of the weighted average of the frequency components of the X-axis Angular Velocity (frequency)

fBodyGyro-meanFreq()-Y

Mean by subject and activity of the Mean of the weighted average of the frequency components of the Y-axis Angular Velocity (frequency)

fBodyGyro-meanFreq()-Z

Mean by subject and activity of the Mean of the weighted average of the frequency components of the Z-axis Angular Velocity (frequency)

fBodyAccMag-mean()

Mean by subject and activity of the Mean of the magnitud (calculated using Euclidean norm) of Body Linear Acceleration (frequency)

fBodyAccMag-std()

Mean by subject and activity of the Standard Deviation of the magnitud (calculated using Euclidean norm) of Body Linear Acceleration (frequency)

fBodyAccMag-meanFreq()

Mean by subject and activity of the Mean of the weighted average of the frequency components of magnitud (calculated using Euclidean norm) of the Body Linear Acceleration (frequency)

fBodyBodyAccJerkMag-mean()

Mean by subject and activity of the Mean of the magnitud (calculated using Euclidean norm) of the Jerk signal for the Body Linear Acceleration (frequency)

fBodyBodyAccJerkMag-std()

Mean by subject and activity of the Standard Deviation of the magnitud (calculated using Euclidean norm) of the Jerk signal for the Body Linear Acceleration (frequency)

fBodyBodyAccJerkMag-meanFreq()

Mean by subject and activity of the Mean of the weighted average of the frequency components of magnitud (calculated using Euclidean norm) of the Jerk signal for the Body Linear Acceleration (frequency)

fBodyBodyGyroMag-mean()

Mean by subject and activity of the Mean of the magnitud (calculated using Euclidean norm) of the Angular Velocity (frequency)

fBodyBodyGyroMag-std()

Mean by subject and activity of the Standard Deviation of the magnitud (calculated using Euclidean norm) of the Angular Velocity (frequency)

fBodyBodyGyroMag-meanFreq()

Mean by subject and activity of the Mean of the weighted average of the frequency components of magnitud (calculated using Euclidean norm) of the Angular Velocity (frequency)

fBodyBodyGyroJerkMag-mean()

Mean by subject and activity of the Mean of the magnitud (calculated using Euclidean norm) of the Jerk signal for the Angular Velocity (frequency) fBodyBodyGyroJerkMag-std()

Mean by subject and activity of the Standard Deviation of the magnitud (calculated using Euclidean norm) of the Jerk signal for the Angular Velocity (frequency)

fBodyBodyGyroJerkMag-meanFreq()

Mean by subject and activity of the Mean of the weighted average of the frequency components of magnitud (calculated using Euclidean norm) of the Jerk signal for the Angular Velocity (frequency)