

ISTE-470 Exercise 2: Data Scavenger Hunt

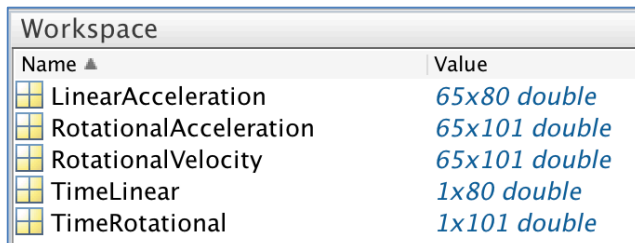
This exercise has two main purposes; first, you are given the opportunity to manipulate, display, and compute features from raw data and second, you will use the features you create to find trends in the data. Over the course of this exercise, you will be guided through a process of manually finding interesting phenomena in a data set. There is no specific process that will work for all data sets, but this specific process should give you a feel for how you would initially analyze your own data set that you find for your course project.

Part 1 – Loading Data in Python

The data set you'll be using for this exercise is on myCourses under Python Data Visualization and is stored in two files; waveforms.csv and times.csv. Save both files to your computer. Now download scavenger_hunt.py and place it in the same directory as the two data files.

In scavenger_hunt.py, you'll see two functions, read_waveforms and read_times, which are used to parse the data contained in waveforms.csv and times.csv, respectively.

After running the script, you'll see five numpy arrays called LA, RA, RV, TL, and TR, which correspond to the abbreviations of the data matrices shown below. Notice the dimensions of each matrix. How many instances of data do we have in the first three matrices? How do you know?



Name	Value
LinearAcceleration	65x80 double
RotationalAcceleration	65x101 double
RotationalVelocity	65x101 double
TimeLinear	1x80 double
TimeRotational	1x101 double

The matrices are defined as follows:

LinearAcceleration (LA): This matrix contains 65 instances of 80 linear acceleration readings taken by an accelerometer. The times at which each of these measurements occurred may be found in the TimeLinear matrix. Note that those two matrices have the same number of columns. The unit of each accelerometer reading is in g's (9.81 m/s^2).

RotationalVelocity (RV): This matrix contains 65 instances of 101 rotational velocity measurements taken by a gyroscope. The times at which each of these measurements

occurred may be found in the TimeRotational matrix. Note that those two matrices have the same number of columns. The unit of each gyroscope reading is in radians per second (rad/s).

RotationalAcceleration (RA): This matrix contains 65 instances of 101 rotational acceleration measurements that are derived from the rotational velocity measurements taken by the gyroscope. Rotational acceleration for our purposes is simply the difference between consecutive rotational velocity measurements and represents the rate of change in rotational velocity from reading to reading. Like the rotational velocity matrix, the measurement times come from the TimeRotational matrix. The unit for these measurements are in radians per second squared (rad/s^2).

TimeLinear (TL): This matrix has a single row of 80 times at which the accelerometer took its measurements, beginning at 0 seconds, and ranging to approximately 0.050 seconds, or 50 milliseconds (ms).

TimeRotational (TR): This matrix has a single row of 101 times at which the gyroscope took its measurements, beginning at 0 ms, and ranging to approximately 50 ms.

Now that you know what each of these matrices represent, let's look at the data!

Part 2 - Data Manipulation and Visualization

First of all, the two “time” matrices are not very pleasant to look at. Try printing out of the contents of TL and TR. You’ll notice the “min” and “max” values for these two matrices are very small numbers. The times are in seconds, but when we talk about the time scales at which the accelerometer and gyroscope measurements were taken, we really want this data to be in milliseconds. How can we convert from seconds to milliseconds? Modify two lines of code in the `read_times` function so that time will be expressed in milliseconds. After you do this, the lines should look like the following:

```
data[i] = float(data[i]) * <conversion constant>;
```

Now let’s see what the linear acceleration (LA), rotational velocity (RV), and rotational acceleration (RA) waveforms look like. Before we visualize the data, let’s note the “min”, “max”, and “mean” values for each of these matrices using `np.min(<matrix>)`, `np.max(<matrix>)`, and `np.mean(<matrix>)`, respectively, where `<matrix>` will be LA, RV, or RA. Note that these “summary statistics” of the data are the min, max, and mean for the entire 2D matrices. If we want to see the min, max, or mean for a single waveform instance, use the following code example for the matrix and summary statistic of interest.

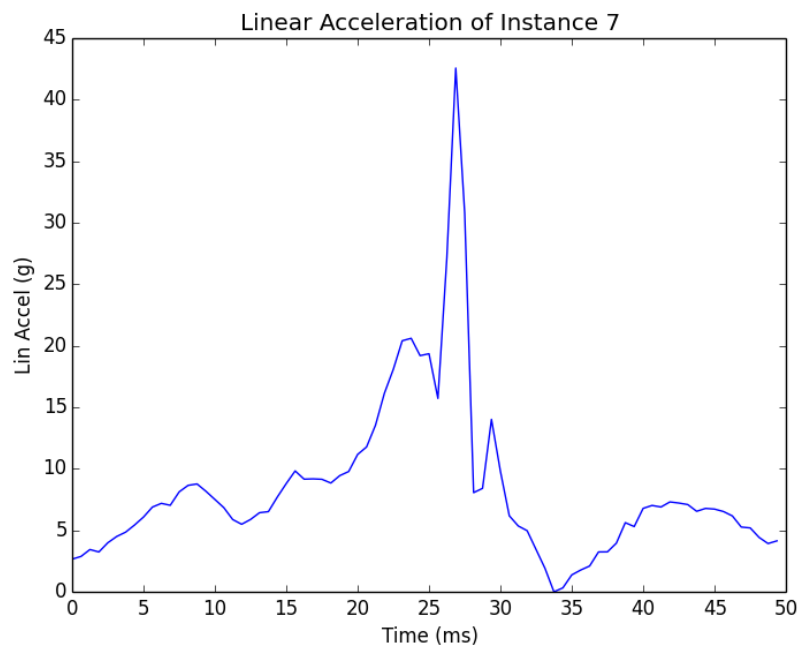
```
print(np.mean( LA[2, :] ))  
  
8.1695979875
```

This example computes the mean of all the 80 accelerometer readings from the 3rd instance of a linear acceleration waveform (since the instances are 0 indexed, we’re accessing row 2 of the matrix). Notice that the “:” operator is used in place of providing the range of indices (0:79) of the row of data. If Python does not output the same answer as shown in the example above, something is incorrect and you should seek assistance from your instructor.

What does a linear acceleration waveform look like?! Let’s plot one and see. Add the following code to your script and open the image file it creates, `Instance 7.png`.

```
plt.plot(TL, LA[6, :])  
plt.title('Linear Acceleration of Instance 7')  
plt.xlabel('Time (ms)')  
plt.ylabel('Lin Accel (g)')  
plt.xticks(np.arange(0, 55, step=5))  
plt.savefig('Instance 7.png')  
plt.close()
```

This set of commands plots the 7th linear acceleration waveform instance (stored in row 6 of the matrix) using a blue line and adds a meaningful title, axes labels, and x-axis tick marks every 5 milliseconds. It's always important to know what the units of measure are.



Your plot should look something like this.

Using a for loop and the subplot command, create an image file for every data instance that contains all three waveforms for that instance in a single image. The code should look something like below and should be placed in the `plot_waveforms` stub function.

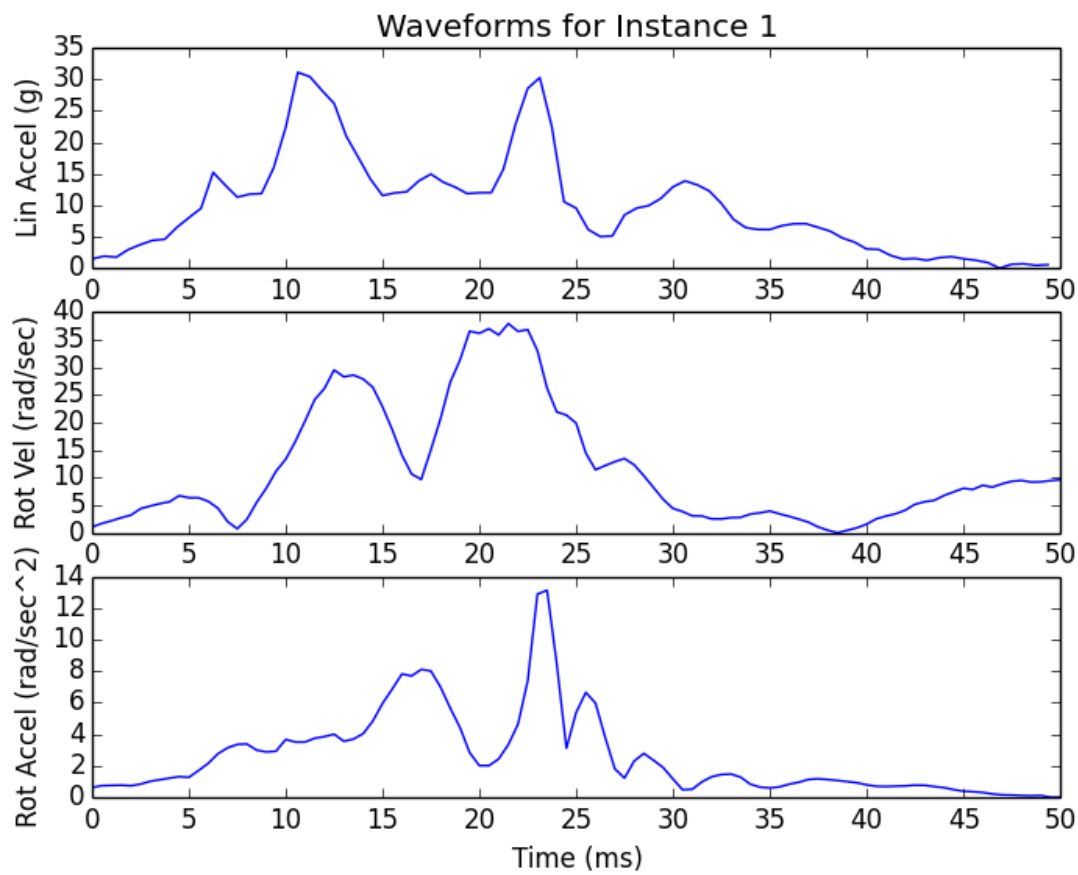
```
for i in range(0, num_instances) :
    plt.subplot( 311 )
    <plot commands for linear acceleration waveform>

    plt.subplot( 312 )
    <plot commands for rotational velocity waveform>

    plt.subplot( 313 )
    <plot command for rotational acceleration waveform>

    plt.savefig('Instance ' + str(i + 1) + '.png')
    plt.close()
```

You want your plots to look like the following example:



Notice that each waveform has x-axis tick marks and a y-axis label. Only the top subplot has a title and only the bottom subplot has a x-axis label. Once you've created the images for each data instance, take a look at all of the 65 images.

Do any of the waveforms stand out over the others? Compare the respective individual waveforms from each data instance as you go through the images one at a time. Is there a way to improve the visualizations to make the waveform comparisons more meaningful?

Part 3 – Creating, Analyzing, and Selecting Features

Out of the 65 data instances, 5 of them are statistically “different” than the other 60 instances. In this part of the exercise, you will be creating features that will help you and your team complete the data scavenger hunt. You’ll be generating several features and only some of them will turn out to be useful in finding the 5 data instances that are different than the rest.

Next, write code that will iterate across the three respective waveform matrices (LA, RV, and RA) and compute the min, max, and mean of each individual waveform. The easiest way to do this is to append the feature values to lists and then convert the lists to numpy arrays. Use the following names for your numpy arrays.

MLA: minimum linear acceleration
ALA: average (mean) linear acceleration
PLA: peak (max) linear acceleration

MRV: minimum rotational velocity
ARV: average (mean) rotational velocity
PRV: peak (max) rotational velocity

MRA: minimum rotational acceleration
ARA: average (mean) rotational acceleration
PRA: peak (max) rotational acceleration

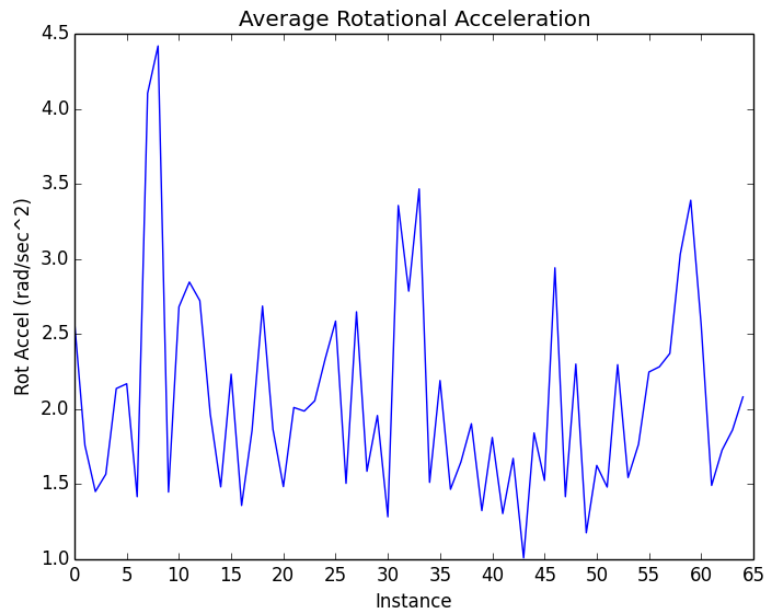
Ultimately, each feature should contain 65 values; one for each instance. Have your script output the summary statistics for each of these features. Specifically, note the range of values (minimum to maximum) of each feature and their mean. See the output below to make sure you are on the right track with your features.

```
MLA: min = 0.0, max = 0.0, avg = 0.0
ALA: min = 4.39519404, max = 22.2569035, avg = 10.9643273598
PLA: min = 16.8252, max = 151.7148, avg = 50.7157876923

MRV: min = 0.0, max = 0.0, avg = 0.0
ARV: min = 5.70672742574, max = 23.3975381287, avg = 10.9613332073
PRV: min = 18.1009, max = 55.3868, avg = 35.9690446154

MRA: min = 0.0, max = 0.0, avg = 0.0
ARA: min = 1.0092340495, max = 4.41835235644, avg = 2.07069517791
PRA: min = 10.3132, max = 44.5424, avg = 13.4938769231
```

Create line plots of each of the features and visually compare them. The plotting you do is up to you.



Above is an example of a feature plot, in this case, the average rotational acceleration (ARA). Note the range of values goes from $\sim 1 \text{ rad/sec}^2$ to $\sim 4.5 \text{ rad/sec}^2$. How does this range compare with other features?

After visualizing all nine of the features and pondering the summary statistics, three features will appear to be the most useful for differentiating between the “different” data instances and the rest of them. Make sure that you save all of your features plots for your next assignment. In the assignment, you will be investigating your three selected features by visualizing them in different ways.