

**Affective Computing**  
**Shrivatsa mishra**  
**Ritik Vatsal**  
**Department of Computer Science and Engineering**  
**Indraprastha Institute of Information Technology, Delhi**

**Week - 07**  
**Lecture - 22**  
**Tutorial Emotions in Physiological Signals**

Hello, I am Shrivatsa Mishra. In the last lecture, we saw how we could use PsychoPy to present stimuli as well as collect sensor data. PsychoPy allows us to integrate several devices to collect multiple different types of data including physiological signals. Since we have also seen how emotions can be extrapolated from physiological data, in this lecture, I shall explain how we can analyze Electrodermal Activity or EDA using Python.

EDA has been closely linked to autonomic emotions and cognitive processing. And EDA is widely used as a sensitive index for emotional processing and sympathetic activity. Investigations of EDA have also been used to eliminate wider areas of inquiry such as physiology, personality disorders, conditioning and neuropsychology.

We shall use sample EDA data collected through an Empatica E4 device. This is a device that can be worn and streamed data through Bluetooth or store that data locally. We can extract that data using an Empatica E4 app and for analysis, we will use flirt module in Python.

(Refer Slide Time: 01:33)



(Refer Slide Time: 01:41)



Hello, I am Ritik Vatsal and I will be introducing you to the Empatica E4 wristband data collection device. The Empatica E4 is a wristband that collects your physiological data. It has a single button and a LED light on the surface and two sensor points on the back with electrodes that need to touch your skin when you wear the device. To wear the device, just place it on your wrist with this side on the top and just make sure that it is comfortable enough that you can do daily task.

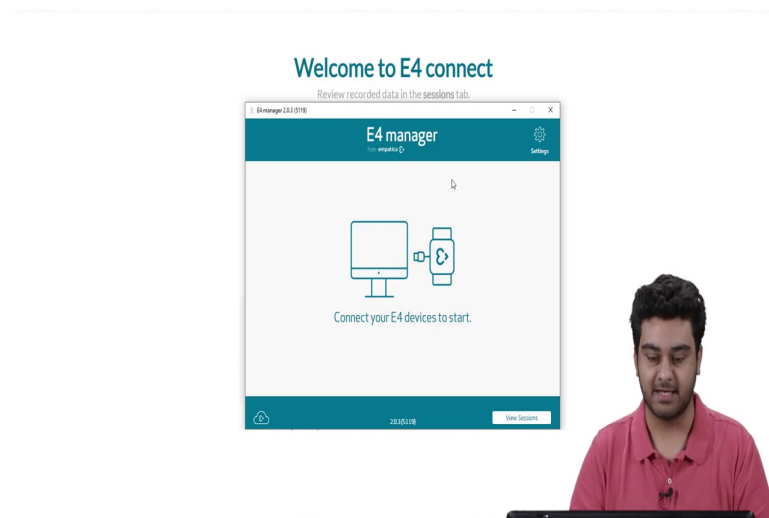
But it is tight enough that both the electrodes at the bottom touch your skin at all times. After that, we are ready to start the device. To start the device, just press and hold the top button for three seconds. A blue light would come on indicating that the device has started.

The blue light would start blinking for about 15 to 20 seconds while the device initializes the setup and checks all the sensors and all. After the watch has finished blinking for up to a minute, the light turns red. When the light is red, you know that the recording has started.

After some time, the red light would fade off and the light would turn black. That would mean that the data is still being recorded, but the watch has turned off the light to preserve battery. While recording, you can single press the button to mark events in real time on the watch like this.

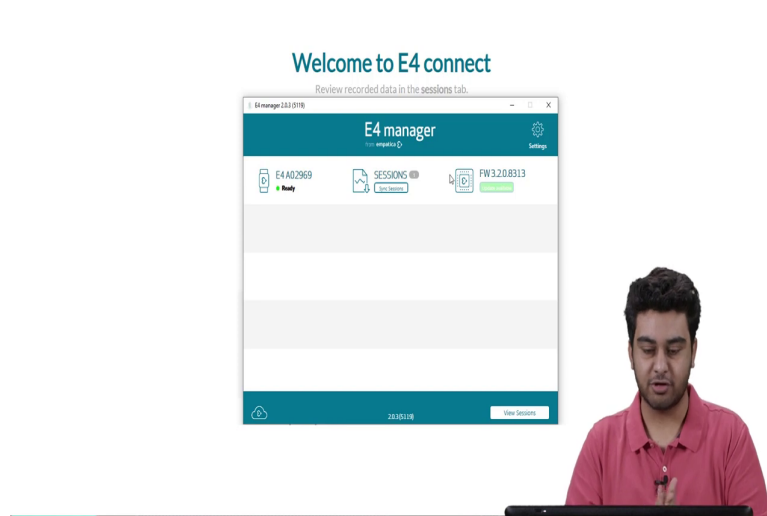
The watch light would again turn on for a small time. Finally, to turn off the watch and stop the data from stop data recording, you can just press and hold the button for three seconds and that is it. The watch has now stopped data recording.

(Refer Slide Time: 03:15)



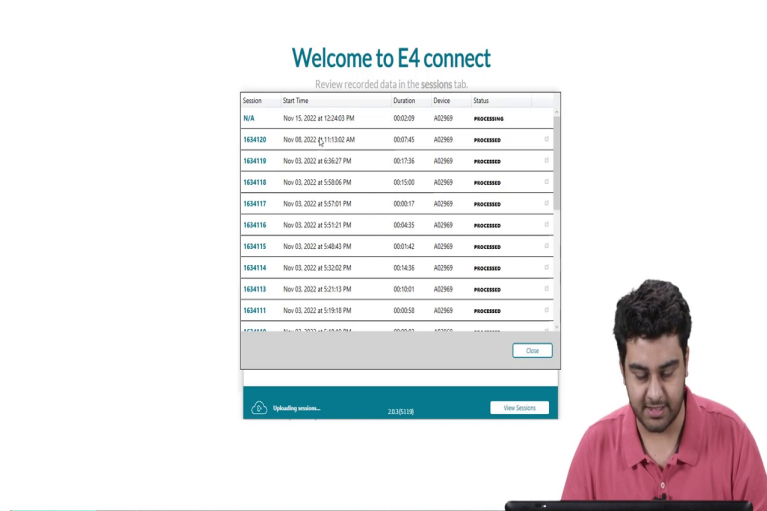
This is the splash screen of the E4 manager device that you would be greeted by when you will first log into the application. On this screen, you now need to connect your device to the USB port on your laptop or PC. I will do that now, ok.

(Refer Slide Time: 03:29)



Now, as you can see, my watch is ready and I see the one session that I just recorded. To move forward with this, I just simply click the Sync Sessions button and all the sessions would be synced and would be available for me to view, ok.

(Refer Slide Time: 03:50)



Now, we can click the view session button and we can see all the sessions that have been recorded. This is the session for today's date which has just been processed, ok. By clicking this button, we can view more about the session, yeah.

(Refer Slide Time: 04:01)

## Welcome to E4 connect

Review recorded data in the sessions tab.

We are sunsetting the E4 to make room for more powerful technology, bringing our real-world research suite to EmbracePlus.

The E4 workflow software suite (E4 connect, E4 manager, E4 real-time, E4 streaming server, and E4 SDQ) will remain available until August 2024.

The software suite will continue to undergo regular maintenance until the software is fully phased out, and all E4 devices within the warranty period will be serviced as normal. For any questions please visit our [support center](#).

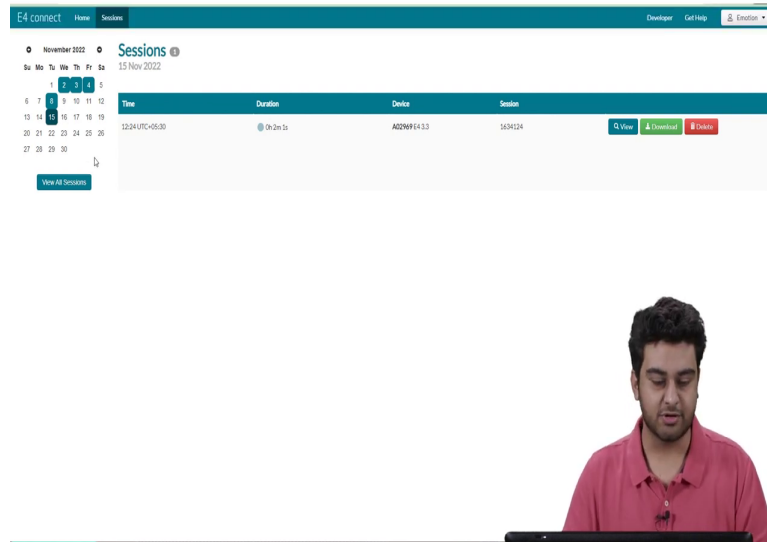


**Unlock the future of real-world research**  
Order our new suite for real-world and academic research, and claim your early bird discount.

[Learn more](#)



(Refer Slide Time: 04:06)



The screenshot displays the 'E4 connect' website interface. At the top, there is a navigation bar with 'Home' and 'Sessions' tabs, and a search bar containing 'Emotion'. Below the navigation bar, a calendar for November 2022 is visible, with the 15th of November highlighted. To the right of the calendar, the 'Sessions' section shows a table with the following data:

Time	Duration	Device	Session	View	Download	Delete
12:24 UTC+05:30	01:2m 1s	A0296F1433	5634124	<a href="#">View</a>	<a href="#">Download</a>	<a href="#">Delete</a>

Below the table, there is a button labeled 'View All Sessions'. In the bottom right corner of the slide, a man in a red shirt is shown from the chest up, looking at a laptop screen.

So, this is the E4 website where all the session details are stored. You can just click on the sessions icon and there. The sessions are sorted in a date wise manner which you can see in the top left menu. We just click on today's date and we can see today's session which was of 2 minutes and 1 second. We can either download this or view this in more detail.

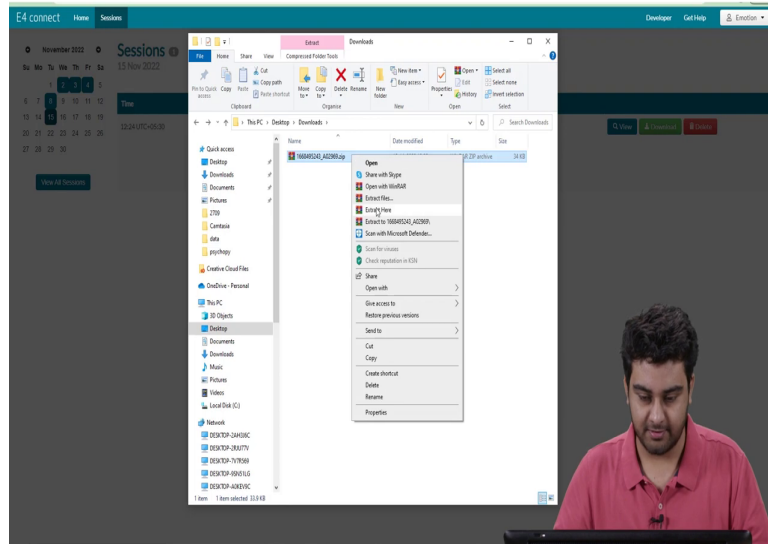


(Refer Slide Time: 04:22)

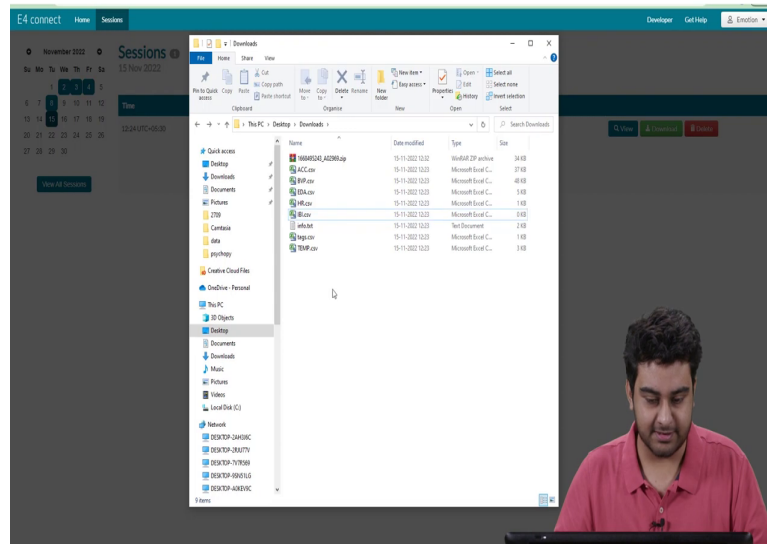


The red lines denote the markers that we placed on the watch by pressing the button. By going back to the Sessions menu, we can download this whole session in a zip file format.

(Refer Slide Time: 04:35)

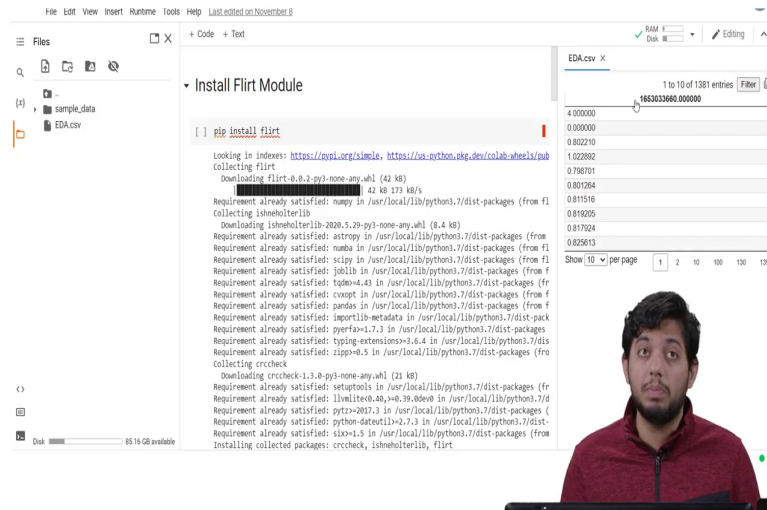


(Refer Slide Time: 04:41)



So, after we download the zip file, we can simply extract the file, extract here and we will see all the different modalities that the device has recorded in a simple and convenient CSV format. Shrivatsa will explain how to extract the data further from this. Thank you.

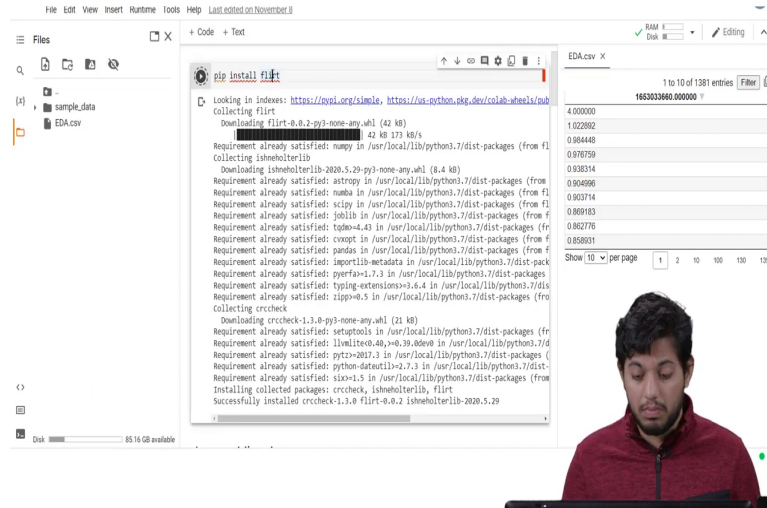
(Refer Slide Time: 04:50)



The Empatica E4 device provides us with multiple types of data such as the heart rate variability; inter beat intervals as well as electrodermal activity. For this assignment, we should just be using the electrodermal activity. Now, this data is stored in a CSV file. The first value in the CSV file is the start time of the entire data. This is stored locally on the Empatica E4 or using the computer if you are streaming it on (Refer Time: 05:13)

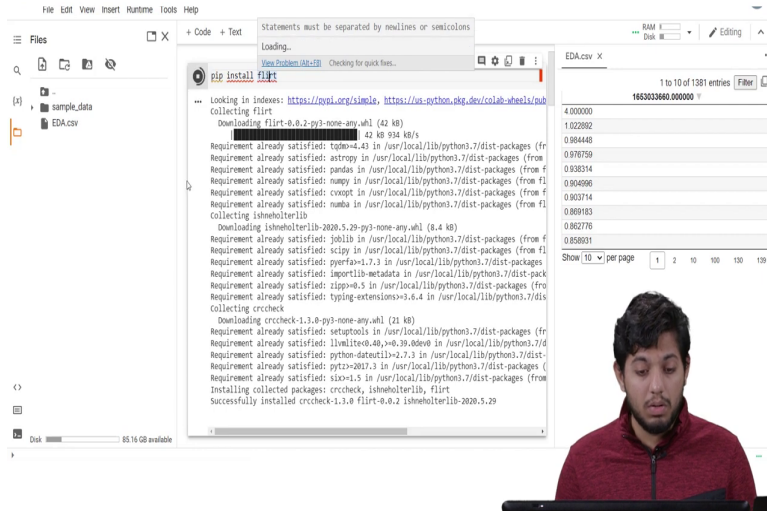
You can reset the clock according to whatever system you want by just connecting it and using the Empatica app. The second value is the frequency at which the data is collected. This is in hertz. So, since the example for the EDA is 4, it means 4 data points are collected every second. Now, let us move on to the code. For this, I shall be using Google Colab as it is easy to use and readily available.

(Refer Slide Time: 05:45)



The data used is 5 minute sample collected during another study using the Empatica E4 itself. We will start by installing flirt module in Python using the pip command. This will take some time, but flirt is a library that will allow you to extract the data very easily as well as extract the features from the same data.

(Refer Slide Time: 06:03)



The image shows a terminal window with the following output:

```
pip install flirt
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-data/python/pypi/
Collecting flirt
  Downloading flirt-0.0.2-py3-none-any.whl (42 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42 kB 934 kB/s
Requirement already satisfied: tqdm<4.43 in /usr/local/lib/python3.7/dist-packages (from flirt)
Requirement already satisfied: astropy in /usr/local/lib/python3.7/dist-packages (from flirt)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from flirt)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from flirt)
Requirement already satisfied: cvxopt in /usr/local/lib/python3.7/dist-packages (from flirt)
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from flirt)
Collecting ishholterlib
  Downloading ishholterlib-2020.5.29-py3-none-any.whl (8.4 kB)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from ishholterlib)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from ishholterlib)
Requirement already satisfied: pyerfa<1.7.3 in /usr/local/lib/python3.7/dist-packages (from ishholterlib)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from ishholterlib)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from typing-extensions)
Collecting crccheck
  Downloading crccheck-1.3.0-py3-none-any.whl (21 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from crccheck)
Requirement already satisfied: llvmlite<0.40.0, >=0.39.0dev0 in /usr/local/lib/python3.7/dist-packages (from llvmlite)
Requirement already satisfied: python-dateutil<2.7.3 in /usr/local/lib/python3.7/dist-packages (from python-dateutil)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from python-dateutil)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil)
Installing collected packages: crccheck, ishholterlib, flirt
Successfully installed crccheck-1.3.0 flirt-0.0.2 ishholterlib-2020.5.29
```

On the right side of the terminal, there is a table with the following data:

1 to 10 of 1381 entries	
165303866.000000	▼
4.000000	
1.022892	
0.964448	
0.976759	
0.932034	
0.904696	
0.903714	
0.869183	
0.862776	
0.858931	

Below the terminal window, there is a video feed of a person with dark hair and a beard, wearing a red jacket, looking at the screen.

(Refer Slide Time: 06:07)

The screenshot displays a Jupyter Notebook environment. The top panel shows the installation of the 'flirt' library using pip. The output indicates that several dependencies are already satisfied, and the 'flirt' library is successfully installed. Below the installation output, the notebook is divided into sections for 'Import Libraries' and 'Import Data'. The 'Import Libraries' section contains the following code:

```
import flirt;
import flirt.reader.empatica
import flirt.with_
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

The 'Import Data' section contains the following code:

```
[ ] data_loc = "EDA.csv"
eda = flirt.reader.empatica.read_eda_file_into_df(data_loc)
print(eda)
```

The bottom right corner of the screenshot shows a small video feed of a person in a red hoodie, likely the presenter, looking at the screen.

Now, that this has been installed, we will import the library. We will be importing flirt library as well as its reader. We will also be importing NumPy, C-Born as well as Matplotlib to graph the data itself.

(Refer Slide Time: 06:22)

The screenshot displays a Jupyter Notebook environment. The left sidebar shows a file explorer with 'sample\_data' and 'EDA.csv'. The main area is divided into three sections: 'Code + Text', 'Import Data', and 'Graph Data'. The 'Code + Text' section contains the following Python code:

```
import time as time
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

The 'Import Data' section shows the execution of the following code:

```
data_loc = "EDA.csv"
eda = flirt.reader.empatica.read_edq_file_into_df(data_loc)
print(eda)
```

The output of the code is a data frame with two columns: 'datetime' and 'eda'. The data frame contains 1380 rows. The first few rows are:

datetime	eda
2022-05-20 08:01:00+00:00	0.000000
2022-05-20 08:01:00.250000+00:00	0.802210
2022-05-20 08:01:00.500000+00:00	1.022092
2022-05-20 08:01:00.750000+00:00	0.756701
2022-05-20 08:01:01+00:00	0.801264
...	...
2022-05-20 08:08:43.750000+00:00	0.793506
2022-05-20 08:08:44+00:00	0.904996
2022-05-20 08:08:44.250000+00:00	0.938314
2022-05-20 08:08:44.500000+00:00	0.976739
2022-05-20 08:08:44.750000+00:00	0.904442

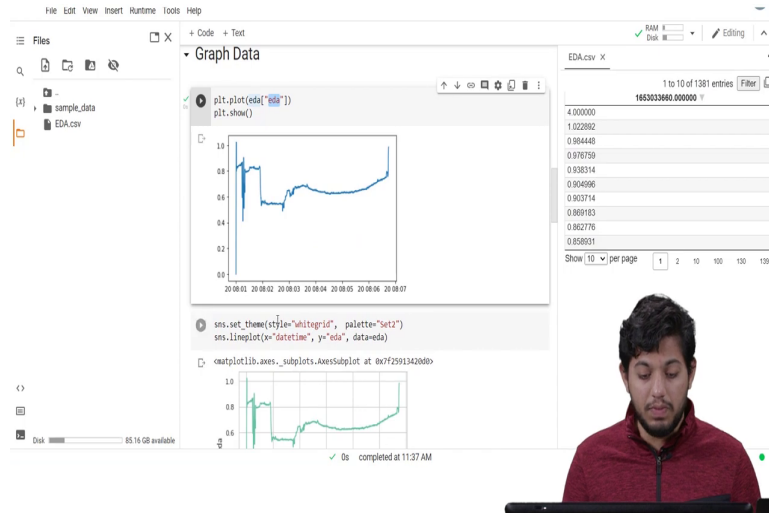
The 'Graph Data' section is currently empty. The bottom right corner of the notebook shows a person's face, indicating a live recording or presentation.

Importing these libraries usually takes some time. Now, that they have been imported let us import the data itself. The data is stored in an EDA dot csv file and we can simply get it through an reader function in the flirt module itself. Upon running it and printing it, we find that this is stored in a data frame.

A data frame is a pandas data type and in this, there are two columns. The date and time of when it was recorded as well as the EDA value for that. Since we know as we know the frequency is 4, therefore, as we see every successive frame is at a difference of point two fifth of a second.

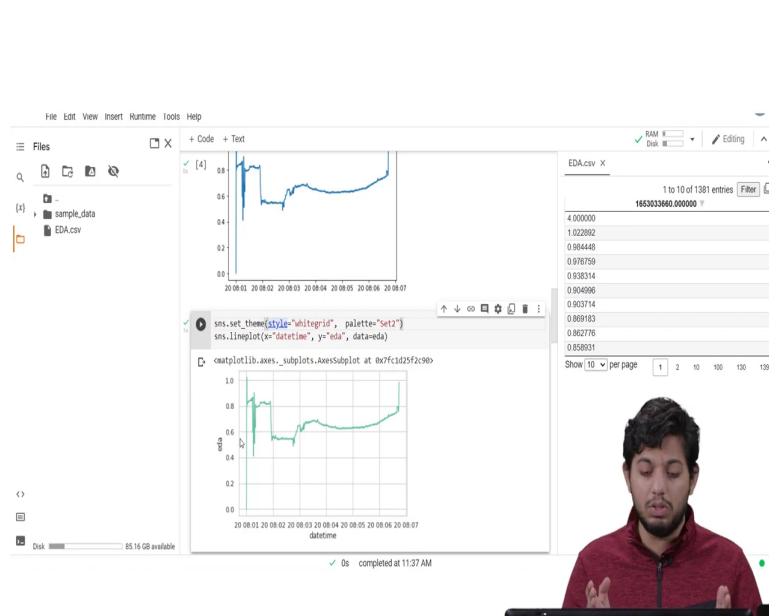


(Refer Slide Time: 07:01)



Now, let us just graph this data naturally. So, there are two ways we can graph. We could use Matplotlib or C-Born. I can show you using both. So, for Matplotlib, we will just use the plt dot plot function and plot the eda value in this as well as using the C-Born function.

(Refer Slide Time: 07:22)



As we can see at the start, there is a large variation. This can be ignored as we usually take smaller chunk.

(Refer Slide Time: 07:27)

The screenshot displays a Jupyter Notebook interface. The main window shows a code cell with the following code:

```
eda_features = flirt.get_eda_features(eda["eda"])
print(eda_features)
```

The output of the code is a DataFrame with the following columns: `dateTime`, `phasic_n_sign_changes`, `phasic_lqr`, and `phasic_lqr_5_95`. The data is as follows:

dateTime	phasic_n_sign_changes	phasic_lqr	phasic_lqr_5_95
2022-05-20 08:02:00:00:00	77	163	1.038182
2022-05-20 08:02:02:00:00	90	150	1.484463
2022-05-20 08:02:03:00:00	87	153	0.565341
2022-05-20 08:02:04:00:00	88	152	0.516457
...	...	...	...
2022-05-20 08:07:40:00:00	5	15	0.816598
2022-05-20 08:07:41:00:00	4	12	0.807556
2022-05-20 08:07:42:00:00	6	6	0.873419
2022-05-20 08:07:43:00:00	4	4	0.869902
2022-05-20 08:07:44:00:00	3	1	0.110796

The notebook also shows a file explorer on the left with 'sample\_data' and 'EDA.csv'. On the right, there is a console output window showing a large number of entries (1 to 10 of 1381 entries) and a status bar at the bottom indicating '0s completed at 11:37 AM'.

A video inset in the bottom right corner shows a man with dark hair and a beard, wearing a red jacket, speaking.

Next, let us move on to extracting the features. We can simply use the get eda features function in the flirt module to get the features. This will take some time. Now, that we have got the data.

(Refer Slide Time: 07:46)

The screenshot displays a Jupyter Notebook environment. The main area contains a code cell with the following content:

```
2 0.8734
2 0.8699
2 0.86407 0.821785

phasic_pct_5 phasic_pct_95 phasic_entropy \
datetime
2022-05-20 08:02:00:00 0.006957 1.045872 4.837986
2022-05-20 08:02:01:00 0.004936 1.007258 4.798759
2022-05-20 08:02:02:00 0.011720 0.978270 4.929440
2022-05-20 08:02:03:00 0.012622 0.967034 5.019536
2022-05-20 08:02:04:00 0.009359 0.971215 5.023832
... ..
2022-05-20 08:07:40:00 0.000000 0.230211 2.136263
2022-05-20 08:07:41:00 -0.009638 0.211146 -Inf
2022-05-20 08:07:42:00 -0.078855 0.067746 -Inf
2022-05-20 08:07:43:00 -0.031849 0.079747 -Inf
2022-05-20 08:07:44:00 -0.037427 0.004358 NaN

phasic_genm_entropy phasic_svid_entropy
datetime
2022-05-20 08:02:00:00 0.002347 0.399805
2022-05-20 08:02:01:00 0.998968 0.349558
2022-05-20 08:02:02:00 0.998968 0.328863
2022-05-20 08:02:03:00 0.991959 0.318918
2022-05-20 08:02:04:00 0.990617 0.311875
... ..
2022-05-20 08:07:40:00 0.977418 0.645935
2022-05-20 08:07:41:00 0.995727 0.687010
2022-05-20 08:07:42:00 0.921076 0.989560
2022-05-20 08:07:43:00 -0.000000 0.908110
2022-05-20 08:07:44:00 NaN NaN
```

Below the code cell, the status bar shows: [345 rows x 44 columns] /usr/local/lib/python3.7/dist-packages/Python/core/interactiveshell.py:3326: UserWarning: user/rode.ohi: call user alobal re: call user re) ✓ 21s completed at 11:39 AM

On the right side, a preview of the data is shown, indicating 1 to 10 of 1381 entries. The first entry is 165303860.000000.

In the bottom right corner, there is a video feed of a person with dark hair and a beard, wearing a red jacket, looking down at a laptop screen.

(Refer Slide Time: 07:52)

```
[345] run > All changes saved
[6] 2022-05-20 08:02:03:00:00 0.991959 0.318918
2022-05-20 08:02:04:00:00 0.990617 0.331475
...
2022-05-20 08:07:40:00:00 0.977418 0.645935
2022-05-20 08:07:41:00:00 0.995727 0.687819
2022-05-20 08:07:42:00:00 0.991076 0.989969
2022-05-20 08:07:43:00:00 -0.000000 0.900110
2022-05-20 08:07:44:00:00 NaN NaN

[345] run > All changes saved
In [345]: DataFrame: eda_features
Python/core/interactiveshell.py:3336: UserWarning
exec
f.user_ns

Dataframe with shape (345, 44)
print(eda_features.columns)

Index(['tonic_mean', 'tonic_std', 'tonic_min', 'tonic_max', 'tonic_gtp',
       'tonic_sum', 'tonic_energy', 'tonic_skewness', 'tonic_kurtosis',
       'tonic_peaks', 'tonic_rms', 'tonic_lineintegral', 'tonic_n_above_mean',
       'tonic_n_below_mean', 'tonic_n_sign_changes', 'tonic_lqr',
       'tonic_lqr_5_95', 'tonic_pct_5', 'tonic_pct_95', 'tonic_entropy',
       'tonic_perm_entropy', 'tonic_svd_entropy', 'phasic_mean', 'phasic_std',
       'phasic_min', 'phasic_max', 'phasic_ppp', 'phasic_sum', 'phasic_energy',
       'phasic_skewness', 'phasic_kurtosis', 'phasic_peaks', 'phasic_rms',
       'phasic_lineintegral', 'phasic_n_above_mean', 'phasic_n_below_mean',
       'phasic_n_sign_changes', 'phasic_lqr', 'phasic_lqr_5_95',
       'phasic_pct_5', 'phasic_pct_95', 'phasic_entropy',
       'phasic_perm_entropy', 'phasic_svd_entropy'],
      dtype='object')

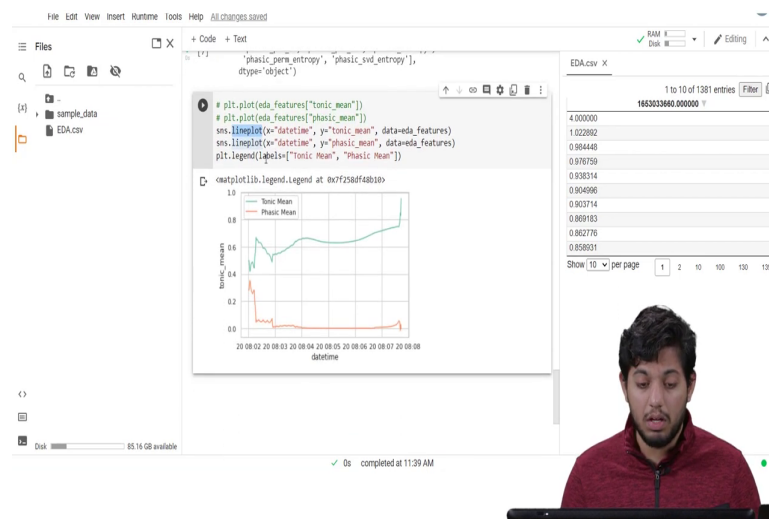
[] # plt.plot(eda_features['tonic_mean'])
[] # plt.plot(eda_features['phasic_mean'])
sns.linplot(x='datetime', y='tonic_mean', data=eda_features)
✓ 0s completed at 11:39 AM
```

We can look at what all features it has extracted. There are two main components to the overall complex refer to as the eda. The first is the tonic level eda. This relates to the signals slower acting components as well as background characteristics. The most common measure of this component is the skin conducting level on SEL.

And changes in the SEL are thought to reflect general changes in the autonomic response. The other components is the phasic component, which refers to the signals faster changing elements. The Skin Conductance Response or the SCR is what is the major component in this.

Recent evidences suggest that both components are important and rely on different neural mechanisms. Crucially, it is important to be aware that the phasic SCR, which often receives the most attention, only makes up a small proportion of the overall EDA complex.

(Refer Slide Time: 08:56)



Now, let us graph the two different values. First, we have the tonic value as well, tonic mean as well as the phasic mean. We graph this using C-Born, this lineplot function. Over here, we put the x value as the datetime in the y value as the tonic mean. Upon graphing it, in our case, we obtain this graph. As we can see, the tonic value is always greater than the phasic value. This is because as stated, phasic values make up a smaller percentage of the total value as compared to the tonic values.

In summary, what we have done is imported the library for flirt, imported the data which stored in a CSV file, read it, graph the base EDA data, extracted the phasic and tonic levels

from the EDA data, as well as graph them. Using just these two basic datas, we are able to extract a lot of different things about the data itself and we will need to look into greater depth into these to understand more. Even now, there is much research being done on the field and new advances are being made.

Thank you for listening to me.