

**Privacy and Security in Online Social Networks**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Lecture – 31**  
**Tutorial 7: Visualization – Highcharts**

Hello everyone, welcome to a new tutorial session for the course. Until now we have seen tutorials for collecting data **from** different online social networks and visualizing networks using tools such as Gephi.

In this tutorial, I will introduce highcharts **the** interactive java script charts which help in creating beautiful visualizations for the data that we have been collecting. In particular, we will look at two things; one is creating visualizations using a python script, and second is a cloud hosting platform that helps in automatically building these visualizations.

In this tutorial, I will talk about four types of graphs that one can build from the data that is being generated. Let us start with the basic bar chart. A bar chart is a chart that presents group data with rectangular bars. The length of the bars is proportional to the value that they represent. Now to start coding in python, we would need a python **wrapper**, which **aids** in creating these visualizations. The wrapper is a simple translation layer between python and java script, which helps in creating visualizations, it really makes things easy to use.

(Refer Slide Time: 01:42)

```
srishti@ubuntu:~/nptel_highcharts$ sudo pip install python-highcharts
[sudo] password for srishti:
The directory '/home/srishti/.cache/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please check the permissions and owner of the
t directory. If executing pip with sudo, you may want sudo's -H flag.
The directory '/home/srishti/.cache/pip/' or its parent directory is not owned by the current user and caching wheels has been disabled. check the permissions and owner of that direc
tory. If executing pip with sudo, you may want sudo's -H flag.
Collecting python-highcharts
  /usr/local/lib/python2.7/dist-packages/pip-8.1.2-py2.7.egg/pip/_vendor/requests/packages/urllib3/util/ssl_.py:315: SNIMissingWarning: An HTTPS request has been made, but the SNI (Su
bject Name Indication) extension to TLS is not available on this platform. This may cause the server to present an incorrect TLS certificate, which can cause validation failures. Fo
r more information, see https://urllib3.readthedocs.org/en/latest/security.html#sni-in-missing-warning.
    SNIMissingWarning
  /usr/local/lib/python2.7/dist-packages/pip-8.1.2-py2.7.egg/pip/_vendor/requests/packages/urllib3/util/ssl_.py:120: InsecurePlatformWarning: A true SSLContext object is not available
. This prevents urllib3 from configuring SSL appropriately and may cause certain SSL connections to fail. For more information, see https://urllib3.readthedocs.org/en/latest/securit
y.html#insecureplatformwarning.
    InsecurePlatformWarning
Collecting future (from python-highcharts)
Requirement already satisfied (use --upgrade to upgrade): Jinja2 in /usr/lib/python2.7/dist-packages (from python-highcharts)
Requirement already satisfied (use --upgrade to upgrade): markupsafe in /usr/lib/python2.7/dist-packages (from Jinja2->python-highcharts)
Installing collected packages: future, python-highcharts
Successfully installed future-0.15.2 python-highcharts-0.2.0
You are using pip version 8.1.2, however version 9.0.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
srishti@ubuntu:~/nptel_highcharts$
```

Now, to install this wrapper, let us go to the terminal and type `sudo pip install python-highcharts` and press enter. This has successfully installed the first python script to generate a bar chart. Suppose, we would like to represent the top 5 users which had the maximum posts which we could extract from Facebook graph `api` in the last four days; we can represent this data in the form of a bar chart.

(Refer Slide Time: 02:49)

```
from highcharts import Highchart
chart = Highchart()
options = {
    'chart':{
        'type':'bar'},
    'title':{
        'text':'Highchart bar'
    },
    'legend':{
        'enabled':True
    },
    'xaxis':{
        'categories':['User 1', 'User 2', 'User 3', 'User 4', 'User 5']},
    'yaxis':{
        'title':{
            'text':'Number of followers (thousands)'
        }
    },
}
data1 = [107,31,839,209,2]
data2 = [133,156,847,906,6]
data3 = [897,914,4894,732,34]
data4 = [1852,954,4259,749,38]

chart.set_dict_options(options)

chart.add_data_set(data1, 'bar', 'day1')
chart.add_data_set(data2, 'bar', 'day2')
chart.add_data_set(data3, 'bar', 'day3')
chart.add_data_set(data4, 'bar', 'day4')

chart.save_file('./bar-highcharts')
```

To start writing code in python, let us start the vi editor, we need to type `vi` and the name of the file, let us name it as `vi bar highcharts dot p y`. We would start by importing the wrapper that we just installed, so the command goes like `from highcharts import highchart`. Next, we define a container that will render our data. We do this by writing `chart is equal to highchart`. Next we define a set of options that will specify our graph. So, we create a dictionary of options, we specify the chart type as bar chart. Next, we define the title of the chart, this is defined in the text, let us write name it as `highchart bar`.

Now, for any chart, we need to define the legends. So, we need to first enable the legends. Now we wanted to represent each of the 5 users on the bar chart. So, we will define the x-axis to be the categories of these users. And we initialize the categories as user 1, user 2, user 3, user 4 and user 5. Let us define the y-axis as well. So, the y-axis would define the number of followers for each of these categories of users. So, the way we define the title is going into the text, and we can specify the text as number of followers and can probably write it in thousands.

So, next we will add the data or the post which we need to be represented in the bar graph. For each day, we will create a data array for the number of posts for each of these 5 users. So, for instance, data 1 would be our first array and we can define it as some random values. Similarly, we will define data 2 for the second user; similarly, we will define the data values for all the 5 users for each of the four days.

Once we have created these options, we will add them to the container that we created before, **now** I do it by writing `chart` and set the dictionary options by passing the variables options that we just created. Finally, we will add the data sets to the chart container by writing `chart dot add data set`. We will pass our first data array. Specify the type that is bar chart and write the day as day 1. We will repeat the same thing for all the four days. Now this code will generate an html file, which we can save in our local directory by writing `chart dot save file`. And we can specify the name as `bar highcharts`, now to quit the vi editor and save the file; let us first press escape and then write colon w q.

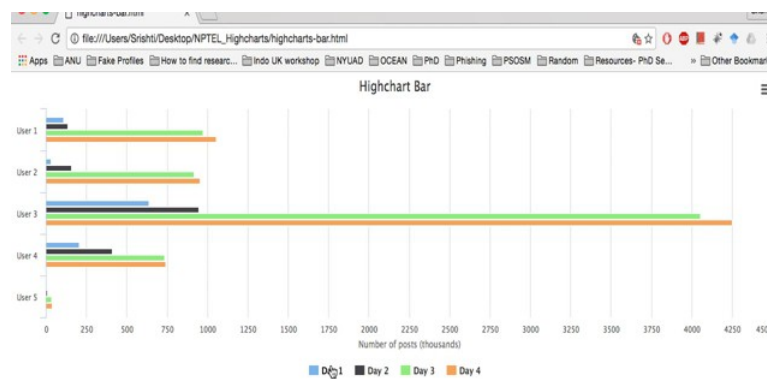
(Refer Slide Time: 09:55)

```
srishti@ubuntu:~/nptel_highcharts$ vi bar-highcharts.py
srishti@ubuntu:~/nptel_highcharts$ python bar-highcharts.py
srishti@ubuntu:~/nptel_highcharts$
```



Now to run this file, we will type the command python and the name of the file that is bar highcharts. So, this would have created a file with a name of bar highcharts in your local directory.

(Refer Slide Time: 10:19)



Now to view this file, you can right click on the file name and see it on the chrome

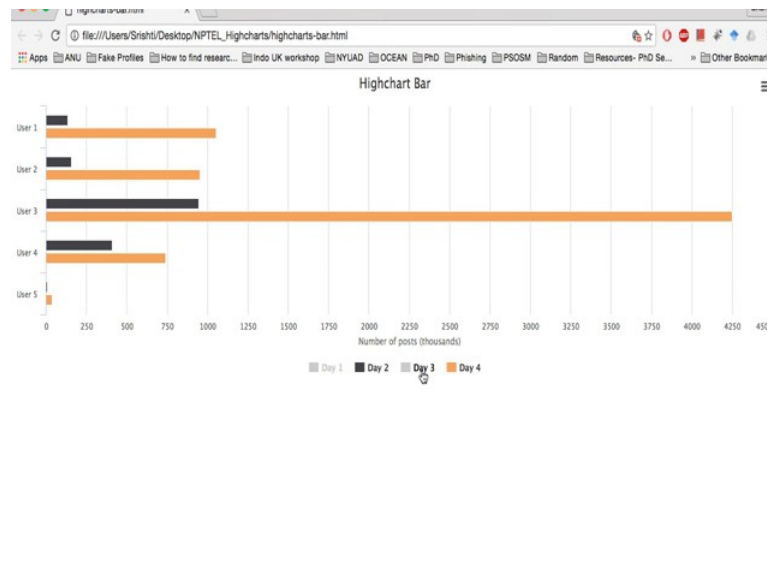
browser. So, this is the bar graph, where you can **hover** to each of these users and see the values. So, day 1 its 1 0 7, and day 4 its 1 0 5 2, so each of the days are represented by different color of bars.

(Refer Slide Time: 10:45)



You can also filter out **a** particular day that you do not want to see. So, if I click on day one, the graph now shows the values for 3 other days.

(Refer Slide Time: 10:50)



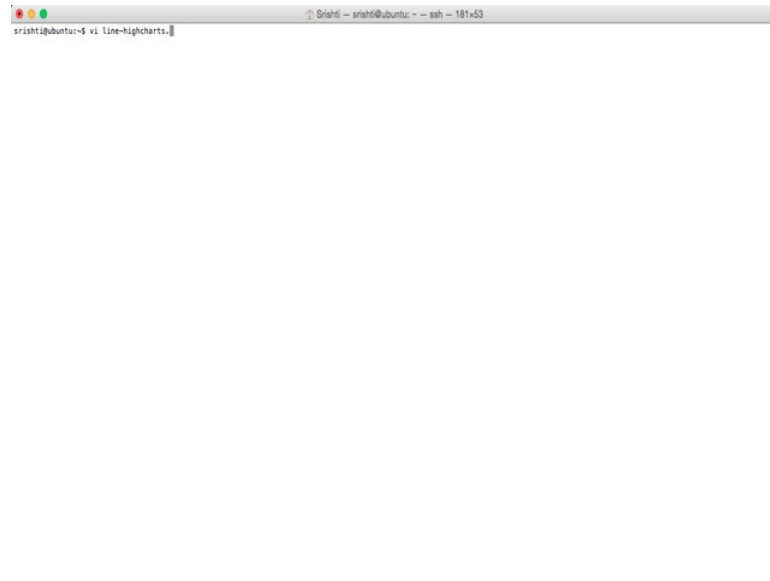
Similarly, I can skip day 3 and see only for day 4 and day 2.

(Refer Slide Time: 11:02)



We can also download and save this graph as a png image, this is how a bar chart can be created using python script.

(Refer Slide Time: 11:53)



Next we look at a second type of graph that is called a line chart. Line chart is the chart which shows series of data points that are connected using a straight line. They are most often used to visualize data, where the data changes over time. Now suppose we would like to see the follow patterns of a user over time. Assuming that we recorded the number of followers for each month of the year, we can represent it using a line chart. To create the python script let us start with the vi editor and let us name the file as line highcharts dot py.

(Refer Slide Time: 12:03)

```
from highcharts import Highchart
chart = Highchart()

options = {
    'chart':{
        'type':'line',
        'title':{
            'text':'Highcharts line'
        }
    },
    'legend':{
        'enabled':True,
        'xaxis':{
            'categories':['Jan', 'Feb', 'Mar', 'Apr', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
            'title':{
                'text':'Months of the year'
            }
        },
        'yaxis':{
            'title':{
                'text':'Number of followers'
            }
        }
    },
}

data1 = [7,7,9,14,18,21,25,26,29,35,42,85]
data2 = [11,17,22, 25, 36, 36, 91, 182, 258]
data3 = [3, 4, 5, 8, 11, 15, 17, 36, 42, 183, 226, 348]

chart.set_dict_options(options)
chart.add_data_set(data1, 'line', 'User 1')
chart.add_data_set(data2, 'line', 'User 2')
chart.add_data_set(data3, 'line', 'User 3')

chart.save_file('./line-highcharts')
```

We would need to follow the same steps as before. So, we will first write the import statement from highcharts import highchart. Then we define the container to render our chart. Next, we define a certain set of options. The first one would be the type of the chart; you name it as a line chart. Next, we need to define the title using the text field, keep it as highcharts line. Similarly, we need to define the legend, so write enabled as true. Again, here we would be defining the categories for x-axis to be the months of the year. So, the way we define it is and we can write all the months and December.

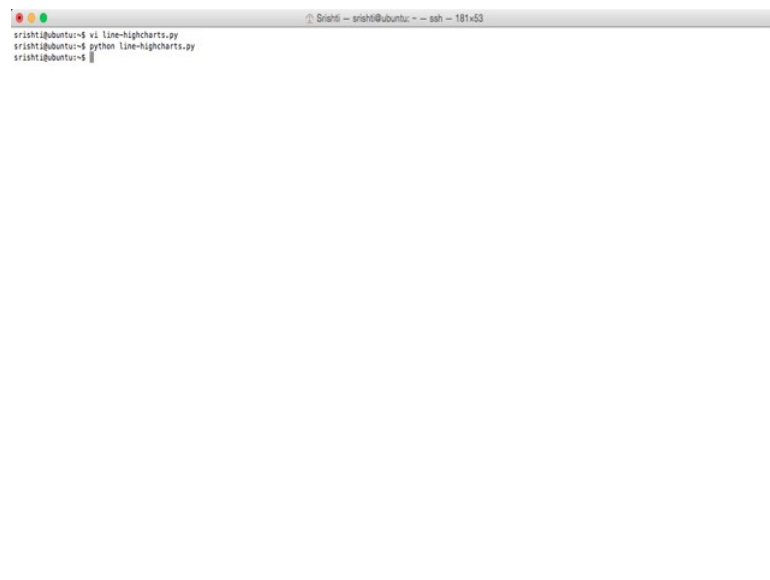
We can define the title of the x-axis by writing the text field and we can define it as months of the year, close the brackets. And for y-axis, we can simple define what the axis is going to represent. So, we can define the title as number of followers, so we would close all the brackets now. And finally, we would define our data array. Let us write some dummy values here. Let us define the data array for three users.

Once we have defined the data arrays, the next step is to add it to the container by writing chart dot set dictionary options to be options. And then finally, add our data sets to the container by writing add data set, first is the data 1; the type of the chart is a line chart and it is for user 1. Similarly, we can write it for all the users. Finally, save the file by writing the command chart dot save file. And we can name the file as line highcharts.



Before saving the file, there is we need to change this. We need to change this line and write it as options, sorry for this mistake. Now to save the file, we first press escape and write colon w q.

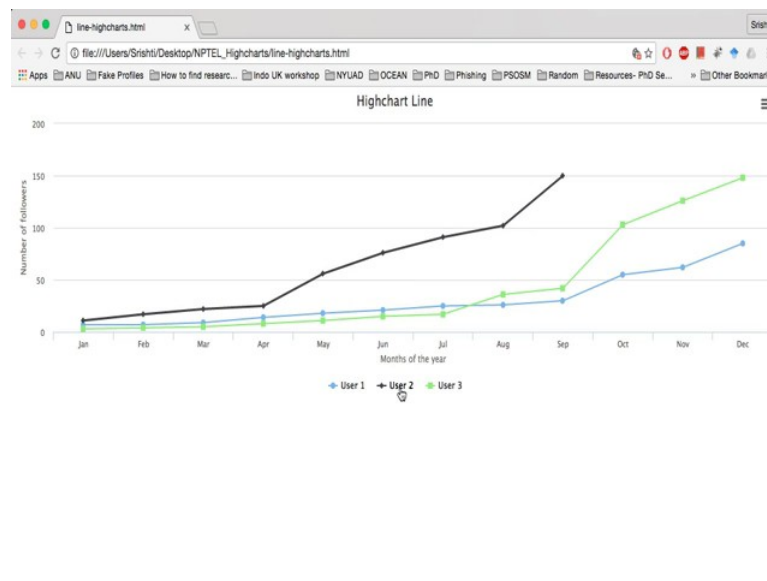
(Refer Slide Time: 17:34)

A screenshot of a terminal window with a title bar that reads "Brash - srish0@ubuntu: ~ - ssh - 181x53". The terminal shows a sequence of commands: first, "srish0@ubuntu:~\$ vi line-highcharts.py", followed by "srish0@ubuntu:~\$ python line-highcharts.py", and finally "srish0@ubuntu:~\$". The prompt changes from "\$" to "~" after the first command.

```
srish0@ubuntu:~$ vi line-highcharts.py
srish0@ubuntu:~$ python line-highcharts.py
srish0@ubuntu:~$
```

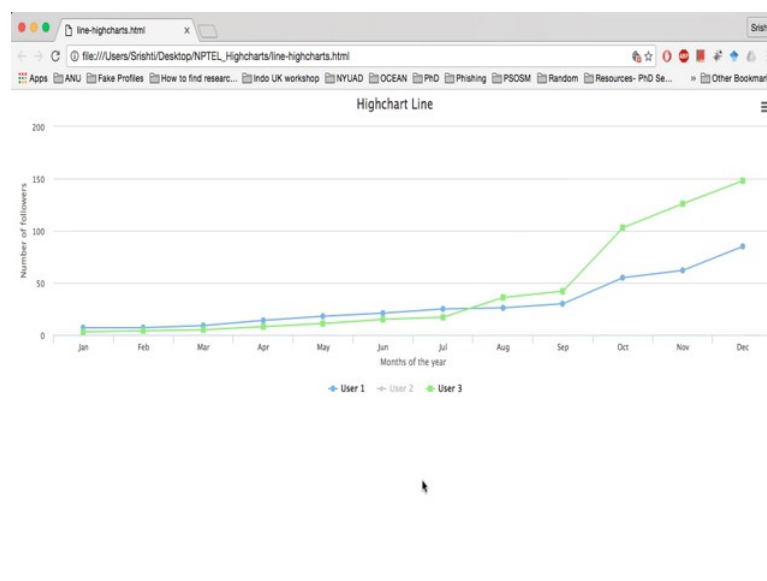
To run the file lets type python line highcharts. This would have created an html file with the name line highcharts. And we can now open it using the chrome browser, again right click on the file and open it in chrome browser.

(Refer Slide Time: 18:00)



This represents the line chart for each of the three users. The beauty of these visualizations **highlight the fact** that they are really interactive; one can play around with a lot of things.

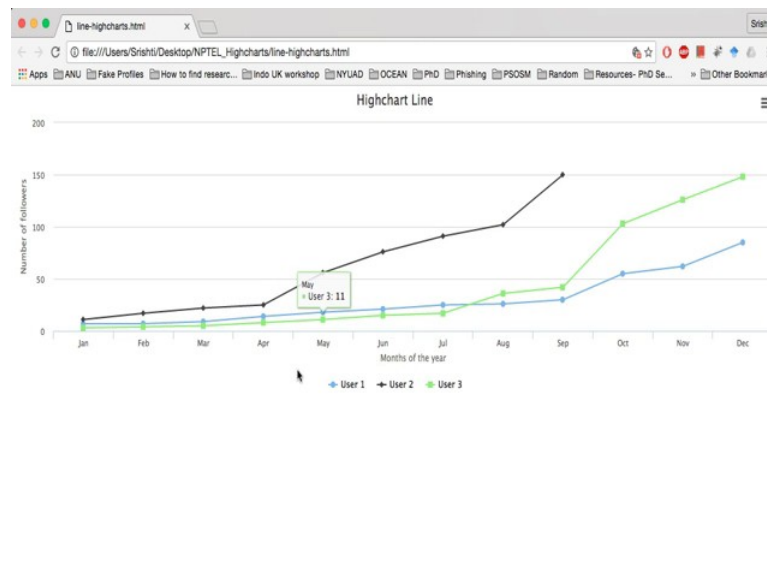
(Refer Slide Time: 18:18)



For instance, I can skip the user 2 from the graph, and see the patterns for user 1 and user

2.

(Refer Slide Time: 18:28)



When I **hover on** the graphs or the data points, I can see the values for each of the users. So, while the bar graph shows the data points as a rectangular bars, line charts connects each of the data points and shows it as a straight line, ok good.

So, now we look at another plot which is called **a** scatter plot where the plot draws a point for each of the data points in the array without really connecting them. Now to create this kind of plot, suppose we have two sets of users on twitter, one is a set of celebrity accounts and the other is a set of normal accounts. Each of these set of accounts have five users each. Now we would like to see the friends to followers' patterns of all the five users in each of the two sets. Just **a** reminder, followers would be the set of people who are following user x and friends would be the set of people who x **follows**.

(Refer Slide Time: 19:28)

```
from highcharts import Highchart
chart = Highchart()

options = {
    'chart': {
        'type': 'scatter'
    },
    'title': {
        'text': 'Highchart scatter'
    },
    'legend': {
        'enabled': True
    },
    'xaxis': {
        'title': {
            'text': 'Followers (thousands)'
        }
    },
    'yaxis': {
        'title': {
            'text': 'Friends (thousands)'
        }
    },
}

data1 = [[161, 51], [167, 59], [159, 49], [167, 63], [155, 53]]
data2 = [[174, 165], [175, 171], [193, 180], [186, 172], [187, 170]]

chart.set_dict_options(options)
chart.add_data_set(data1, 'celebrity accounts', color = 'rgb(223,83,8,0.5)')
chart.add_data_set(data2, 'scatter', 'Normal user accounts', color='rgb(119, 152, 191,0.5)')

chart.save_file('/scatter-highcharts')
```

So, now, to write the script for the scatter plot, we will write vi editor and let us name the file to be scatter highcharts. We would start by importing highcharts. Next, we define the container **for** a highchart. Next is defining the set of options for the chart. The first one would be the chart type; and the type for this **chart** is scatter.

Next, we would define the title of **the** chart and the text can be highcharts scatter, we would want to enable the legend, so we would write enabled as true. Now define the x-axis, which are the followers. We would start by writing it as x-axis. Then we need to **define** the title. Then we would define the text, let us keep it as the number of followers which are in **thousands**. Similarly, we need to define the y-axis, let us keep it as friends again in **thousands**; we need to close the brackets.

Next, we need to define the data array. So, we have two sets of users; one is the celebrity user and the other is a normal user. And each of these sets has 5 users each. Each of the 5 users now **have** followers and friends count. So, data set 1 would be suppose user 1 has 161 followers and 51 friends. The next array would be 167 followers and 59 friends and so on for all the 5 users. Once we have the data array created, next step is to add to the container, so chart dot set dictionary options and we pass our dictionary object.

Then next we add the data set, so it would be data one, type of chart is scatter, and the set is celebrity accounts. And we can also define the color, so let us define it as red. So, you do not need to worry about the color combination, we can always Google it, these are the standard color codes. Similarly, we would define for the next data set. Finally we would save our file as let's name it as scatter highcharts. So, to save this python file, we need to first press escape then write colon w q and press enter.

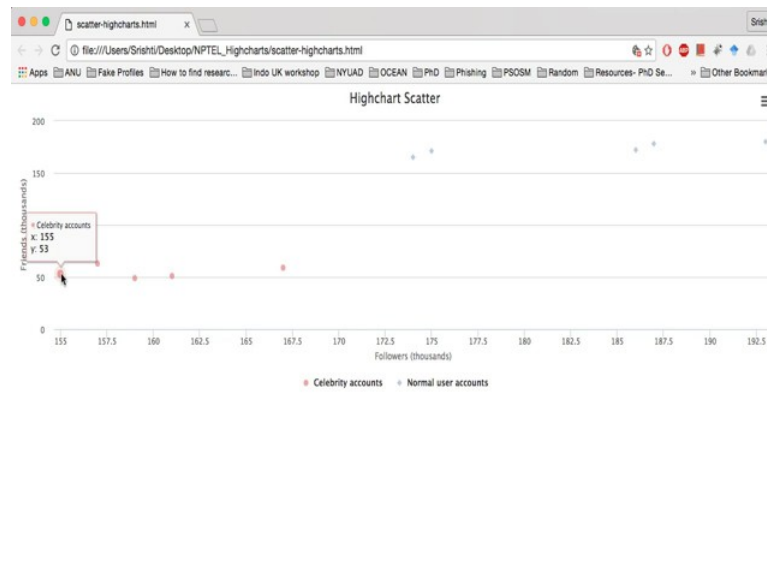
(Refer Slide Time: 24:49)

A terminal window with a title bar showing 'Brahm - srisht@ubuntu: ~ - ssh - 181x53'. The terminal content shows the following commands and their outputs:

```
srisht@ubuntu:~$ vi scatter-highcharts.py
srisht@ubuntu:~$ python scatter-highcharts.py
srisht@ubuntu:~$
```

Now, to run the file let us type python.

(Refer Slide Time: 25:02)



To view the html file, let us go to the chrome browser. In scatter plots, each of the x and the y-axis represents some value. So, this point essentially means that x is 155, and y is 53, similarly for all other values. As mentioned earlier scatter plots are similar to line plots; in line plots each of these data points is connected by a line, whereas in scatter plots it is just represented as a different set of data points and not really connected by a line

Finally, I will cover the last chart type, which is a bubble chart. Similar to scatter plots, each point, they are replaced by a bubble. And we have an additional dimension which is represented by the size of the bubble. So, if the dimension is large, larger would be the size of the bubble; and if the dimension is small then the size of the bubble will be small. And just like these scatter plots, bubble charts, they represent values on both the axis.

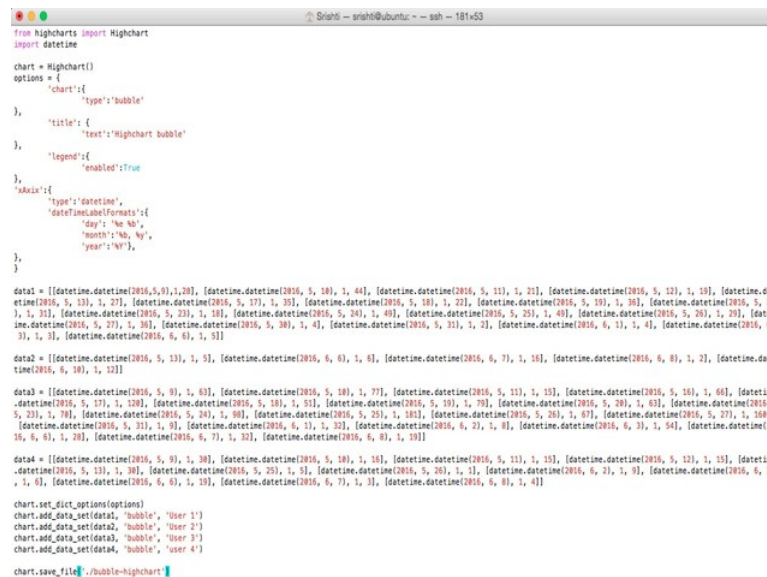
Now suppose we know a twitter user x **who had** posted y number of tweets on a particular day, the way we would code it is that x represents the time when the user posted, y would indicate one if there is a tweeting pattern on that particular day, and 0 if there is no tweet on that particular day, and z would represent the number of tweets that are posted on a particular day. So, we would have three dimensions x, y, z which would now represent on a bubble chart.

(Refer Slide Time: 26:43)



Now, to write the python **script**, we will again go to the vi editor and name the file as bubble highcharts.

(Refer Slide Time: 27:03)



In this code, we will also learn how to represent **date** formats while creating the graphs, but to start with, let us write the standard inputs statements, which is from highcharts

import highchart. Now since we have to represent the values as date formats, we need to include another inbuilt python package called as `datetime`. So, we would write `import datetime`. We now declare our container by writing `chart` is equal to `highchart`.

Next is the set of options that define our graph; first would be the type of the chart. So, here we have let's name it as bubble chart. Next, we would need to define the title for the graph, which is represented as text; and we can write it as `highchart bubble`. Now we would need to enable the legend for the graph.

Now comes the part, which is different from the other graphs that we have been looking before. So, here x-axis is `the` format of date time. So, now the way `we` would represent these axes is we start with x-axis we first defined the type of the format which is now date time. Now in each of these axes, we need that we have particular labels, which is that for each point we need to know that what that particular point represents. For instance, if a person tweets on first January 2016, I need that point on x-axis to be written as first January 2016, so for that we need to define labels, and we write date time label formats.

So, there are some standard ways to represent these date time objects. So, for instance, the way we can define day is percentage e percentage b. Now here percentage e represents the day of the month, so that is 1 through 31; and percentage b represents the short months like January would be jan, February would be feb and so on. After day, we need to define the month and we can represent it as percentage b, which is the short code for each of the months. And then we have percentage y, which represents two digits for the year. So, if the year is 2016 then we will write it as 16. And finally, the way we would write the year would be capital Y which represents all the four digits of the year. So, now we need to close the brackets.

Next, we need to add data arrays. Suppose we have four users, who have different `tweeting` patterns. So, for data user 1, the way we define the data array is the date time object so each of the time stamp is represented as a date time object by writing date time dot date time. And we specify the parameters, for instance, if I write 2016 specify the month and the date. Since, the user a is tweeting at that particular time, we define Y as 1;



and we define number of tweets to be 28. So, this is the first object for user 1 at time stamp 9, 5, 2016.

Similarly, we can have different time stamp values and we will just randomly hard code some of the values. After creating the data objects, we need to add **it** to the container. So, first we would add the options by writing chart dot set **dict** options and passing the options array. Next we would add the data sets. So, it will be data one, type is a bubble chart and we are doing it for the user 1. Finally, we would save the chart by writing save file and we can name it as bubble highchart. Now, to save the file we first press escape and then we write colon w q and press enter.

(Refer Slide Time: 34:23)

A terminal window titled 'Srihari - srishti@ubuntu: ~ - ssh - 181x53'. The terminal shows the following commands and output:

```
srishti@ubuntu:~$ vi bubble-highcharts.py
srishti@ubuntu:~$ python bubble-highcharts.py
srishti@ubuntu:~$
```

To run the file we write python and the name of the file.

(Refer Slide Time: 34:35)



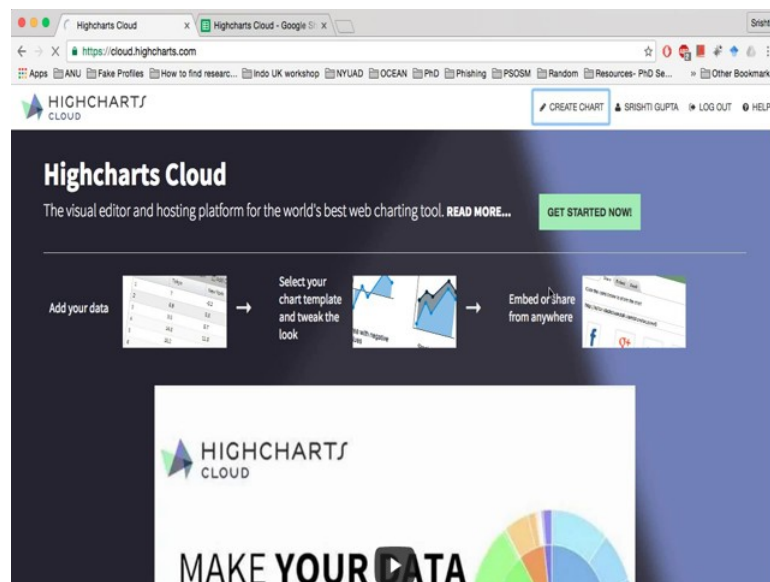
To view the file, again let us open the html that is created by right clicking and opening it in the chrome browser. So, the x axis represents the **timestamp** value starting from 10th May to somewhere around some values in July, where we can see some **overlap** between the user.

(Refer Slide Time: 34:59)



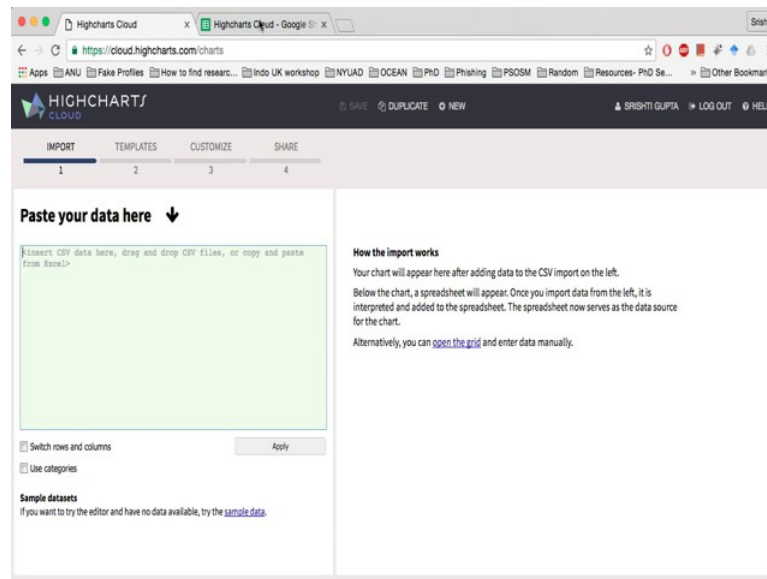
And if **we** want to remove the overlap we can unselect some of the users for instance if I remove the user 4 and I remove the user 1, then I can see the patterns for user 2 and user 3. So, here the size of this bubble is 120, which is greater than the size for this bubble which is 51. So, in bubble charts we can represent three values; the x-axis being the time; y-axis being whether user is posting the tweet or not; and the z-axis which represents what is the volume of the tweets that has been posted. So, that is how we would create four charts that is bar chart, line chart, scatter chart and bubble chart using the python script.

(Refer Slide Time: 35:57)



Now I would speak about another cloud platform called as highcharts cloud, where we can just copy paste our text and it will create the charts on its own. This cloud platform can be reached at <https://cloud.highcharts.com>. You can create a user account, so that you can save the graphs that **we generate** using this platform. So, we would first go to create chart.

(Refer Slide Time: 36:17)



Here we can either paste our data or drag and drop a csv file.

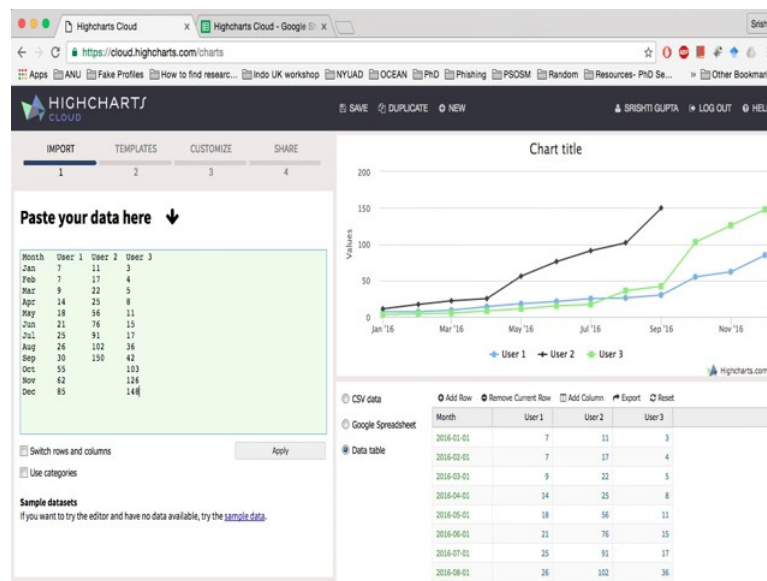
(Refer Slide Time: 36:29)

The screenshot shows a Google Spreadsheet with the following data:

Month	User 1	User 2	User 3
Jan	7	11	3
Feb	7	17	4
Mar	9	22	5
Apr	14	25	8
May	18	56	11
Jun	21	76	15
Jul	25	91	17
Aug	26	102	36
Sep	30	150	42
Oct	55		103
Nov	62		126
Dec	85		148

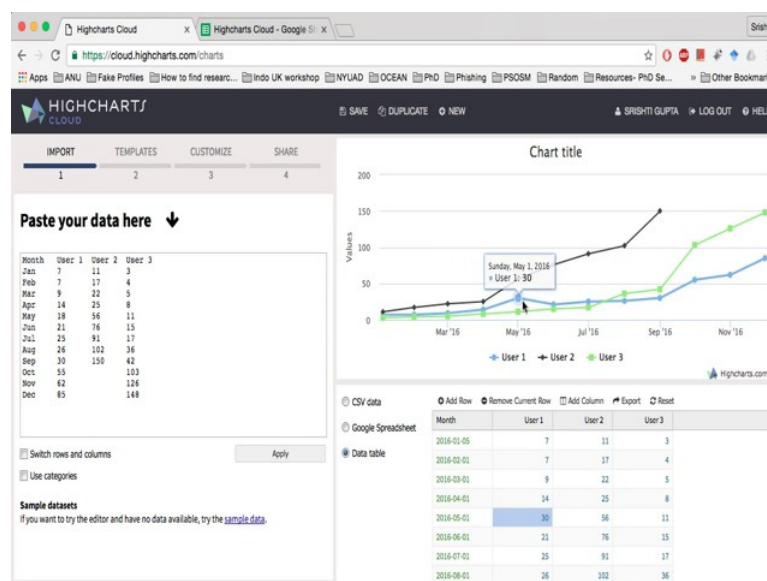
So, this is the data that we used to create the line graph using the python's script. I just added it to csv file, and we can just copy paste all the data from here.

(Refer Slide Time: 36:45)



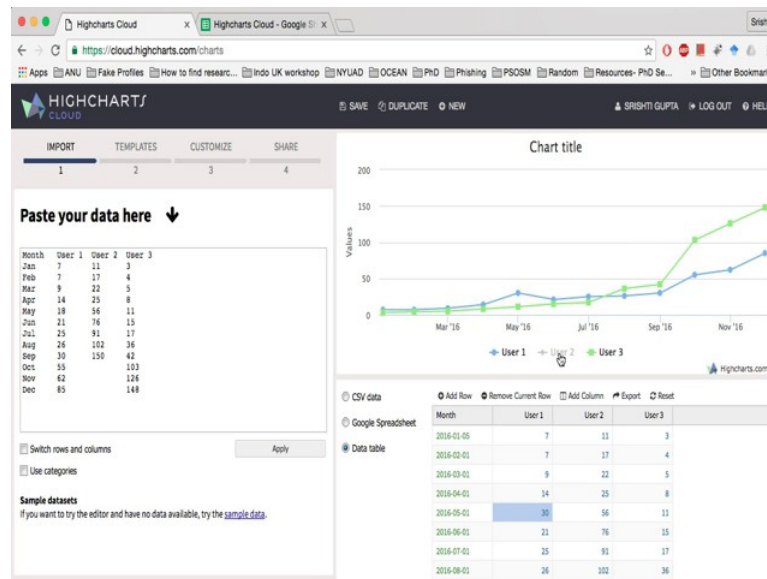
And paste it to the cloud platform. And it automatically creates the line chart that we have created before using the python' script. I can change the value, if I want for instance if I want to make it as **five**, the graph would automatically change its values.

(Refer Slide Time: 37:07)



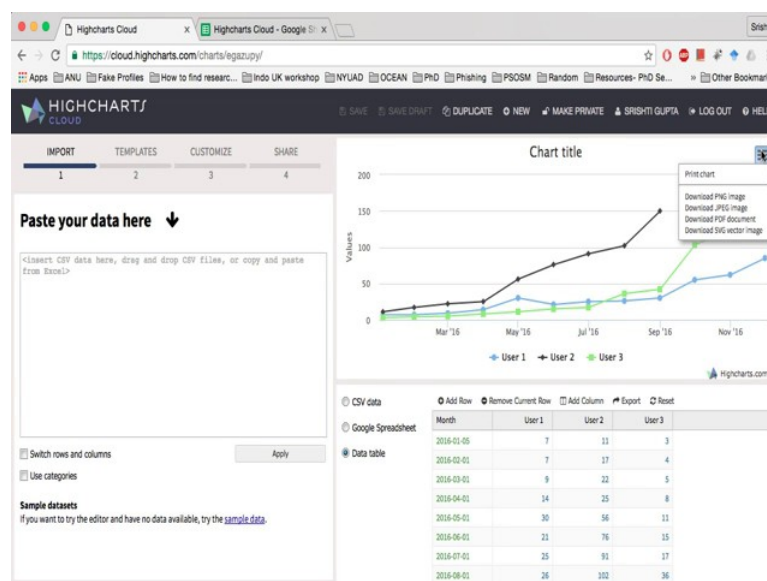
So, here I can change the value to be **say** 30 and the value gets reflected.

(Refer Slide Time: 37:19)



As we observed from the python script, similarly, here we can filter out the users for instance I can filter out the user 2, and see the graph. And each of the tool tips tells us the value for each of the data points. After creating this chart, we can actually save the chart.

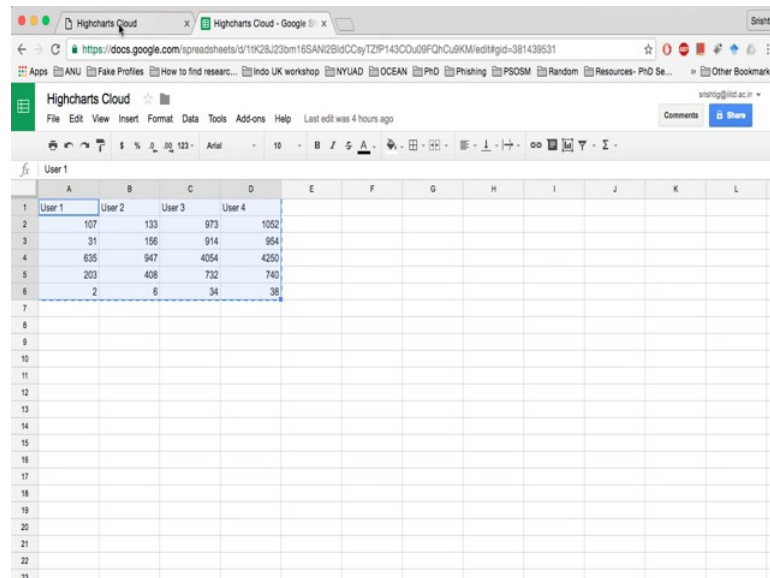
(Refer Slide Time: 37:42)



Once the chart gets saved, it gives us an option to download the chart as a png image

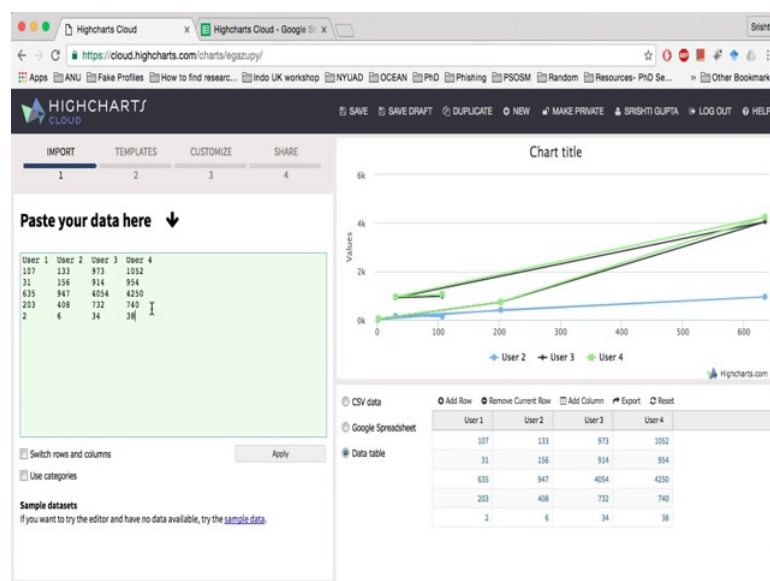
which we can probably use later. So, this was the example of a line chart.

(Refer Slide Time: 37:56)



Let us look at another bar chart example that we developed using a python script.

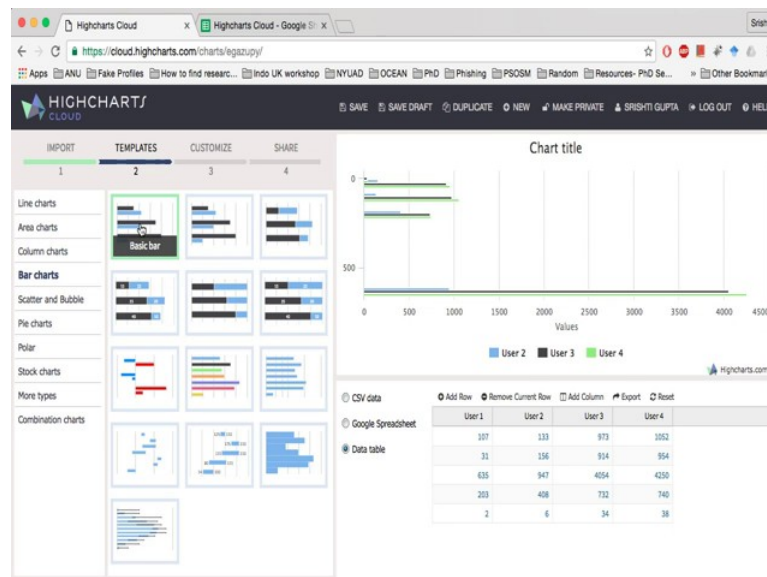
(Refer Slide Time: 38:00)



So, we can just copy paste the data and paste it here. So, here we can customize our

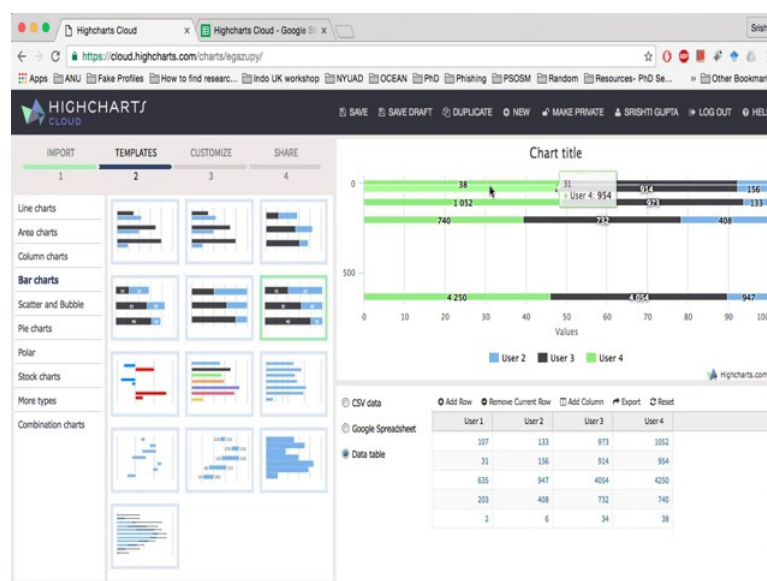
chart.

(Refer Slide Time: 38:08)



So, I go to the templates, I write, go to bar chart, and I click this bar chart. Now it gets converted into a bar chart.

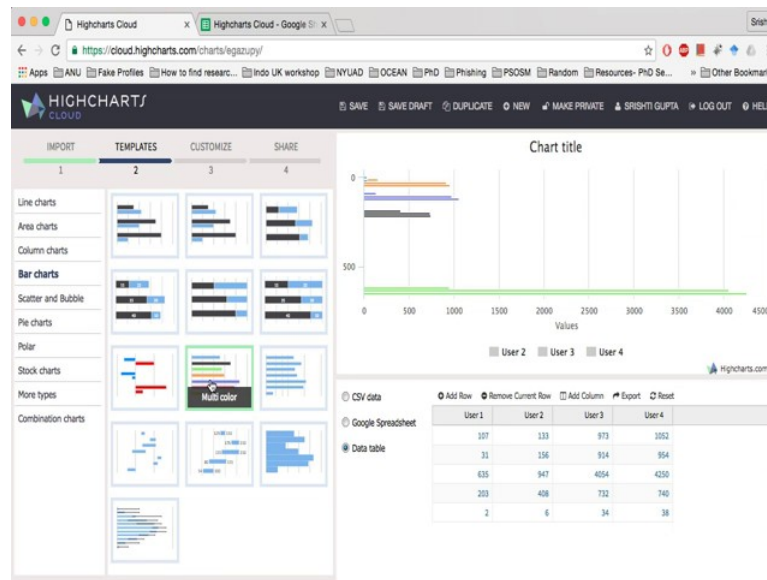
(Refer Slide Time: 38:20)





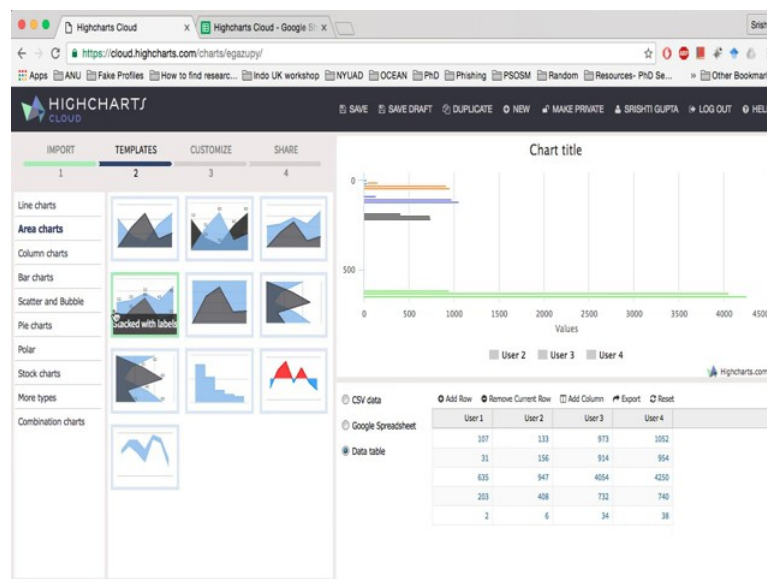
I can have several variations, for instance I can have this where it tells me the value at each of the bars.

(Refer Slide Time: 38:22)



Or I can have it as multi colored bars. You can actually play around with lot of options, which are available on these cloud platforms.

(Refer Slide Time: 38:37)



You can have several kinds of scatters and bubble charts, column charts and linear charts. This would be **it** for the tutorial today. If you have any questions, please feel free to drop it at the discussion forum, and we will be happy to take them.

Thank you.