

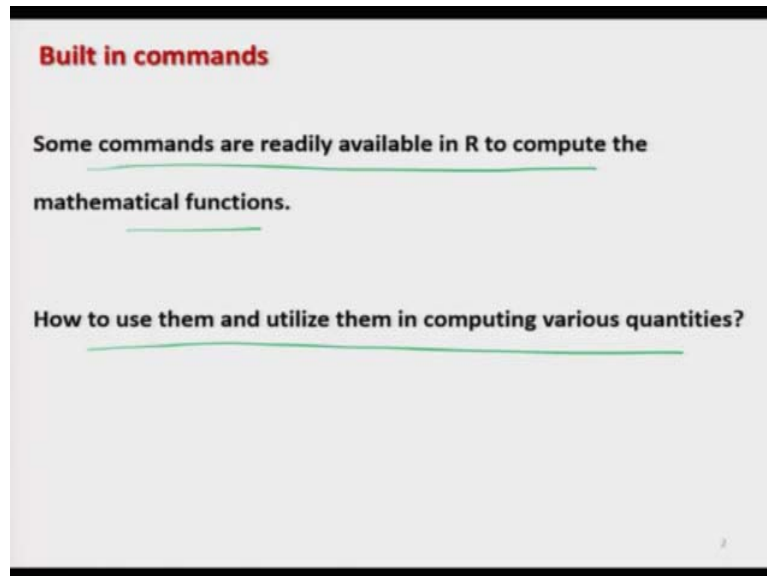
Essentials of Data Science with R Software - 1
Professor Shalabh
Department of Mathematics & Statistics
Indian Institute of Technology Kanpur
Lecture No. 06
Built-in Commands and Bivariate Plots

Hello, friends welcome to the course Essentials of Data Science with R software 1. In which we are going to handle the topics of Probability theory and Statistical inference. So, you can recall that in the last couple of lectures, we have learnt the very basic operations of the R software like addition, multiplication, and other types of operations on a scalar or vector and so on.

Now, in this lecture and in the next lecture, I will try to pick up some more topics of the R software, which I am going to use in the forthcoming lectures. So, in this lecture, I am going to take up the topic of Built-in commands and as simply a plot command. You know, that R has a wonderful capability for creating beautiful graphics and very informative graphics. And it depends actually on you that what type of graphics do you think is going to convey the hidden information inside the data. That will be your choice, that will be your decision, but in this course, just to make the things very simple, I am just going to use here one type of simple plot.

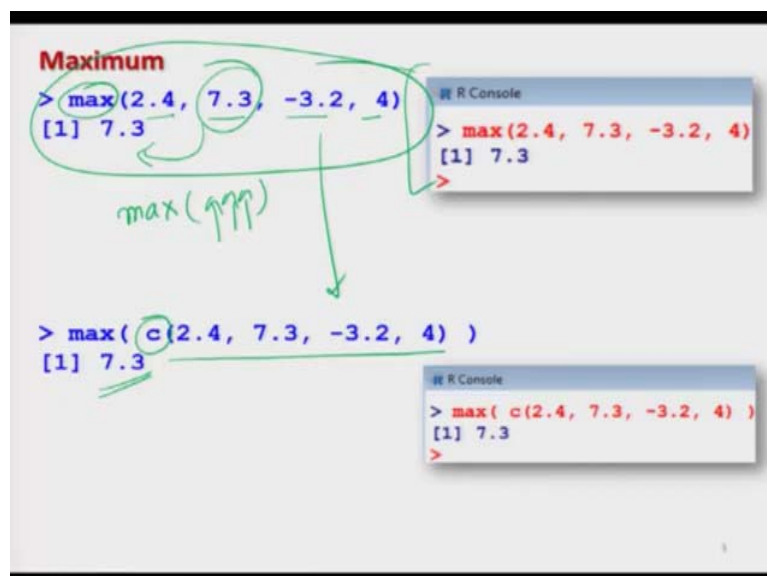
So, in this lecture, very quickly, I will try to discuss that how are we going to use the Built-in commands and how are we going to use the plot command to create some simple graphics. Well, I know that most of you have done these things in the R software, but it is important for us to have a quick review. So, definitely, I will not be going into much detail, but I will very quickly brush up the knowledge. So, let us begin our lecture.

(Refer Slide Time: 01:59)



So, you see in R software, we have two options that either I can write a command on the R console and execute the program or there are some Built-in commands which are available or which have been programmed by some other people. So, those commands are readily available inside the R software to compute different types of mathematical functions. For example, if you want to multiply, or if you want to sum, you want to find out the arithmetic mean etc. So, my objective here is very simple that I want to show you that how are you going to use them and how you have to be careful while using those operations.

(Refer Slide Time: 02:35)



For example, that we try to take care a couple of examples to illustrate that how are we going to handle it and how you have to be careful. Well, there is a long list of the such functions,

but definitely, it is not possible to take up all the commands which are available inside the R software, but I am sure that if you try to learn these basic commands after that, there should not be any problem for you to understand those commands.

Now, let me try to take here one simple example. Suppose you want to find out the maximum value for a given data set. Suppose, I have two option here, the first option is this, those people who are good in programming, they can write up complete program for find out the maximum values. But those who are not good in programming, it is advised and preferable that you can or they can use the built-in function, which can directly compute the maximum value out of a given data set.

And similar is the justification for all other commands also. So, in case if you want to find out the maximum of a given data set, then the command here is `max`. And the rule here is that you have to write down the values inside the parentheses. So, if you write here `max` and inside the parentheses, if you try to write down the values, then it will give you the maximum among the available values which are given inside the parenthesis.

But now there is a question. In order to answer that question, first, let me try to pose here two examples. You can see here I want to find out here the maximum of four values 2.4, 7.3, -3.2, and 4. So, you know that 7.3 is the maximum value and if you try to operate it on the R console, it will try to give you the value here 7.3. And you can see here, this is a screenshot. But if you try to see here, that in this example, I have not used the `c` Command. `c` command means when I am trying to give here the values 2.4, 7.3, -3.2, and 4, I am not using here the `c` command to combine these values but I am simply writing these values.

Now, I tried to do the same operation using the `c` command and I tried to write down the same values here, but I now use the `c` command and I try to write down here all the data values inside the parentheses, but without `c` command and answer is once again 7.3. So, you can see here that now, here, in this case, I have here two options that I can input the data using the `c` command or without using the `c` command and it is giving us the same answer.

But that is the place where I would like to make you cautious. In this maximum command, it is happening that if you are trying to give these values with or without using the `c` command both are working, but that may not really be true with many other functions. Now, you can ask me why this is happening? One of the possible reason I can think is that because this R software was developed by a team and when it is started different people all over the world,

they started contributing in this R package somebody wrote the program for maximum, somebody wrote the program for sum, somebody wrote the program for variance etc.

It might be possible that when they are trying to write down the program, then the person who has written the program for finding out the maximum he has written the program in which the data has to be given without the `c` command. But in case if somebody is writing a program for finding out the sum, then that person may have written the program using the `c` command that means the data has to be given with the `c` command. But later on, when people tried to work together, they started converging, then they try to make such function at the same level. So, they also introduced that if you try to give the data using the `c` command in the maximum function that will also work.

So, now what is happening that this `c` command is working on the maximum with both, with `c` and without `c`, but that will not happen always I will try to show you. So, now what you have to do. My simple advice is that whenever you are trying to give us a set of data in the format of a data vector, always use the `c` command just to avoid all the confusion. Because what will happen, I will show you that in some functions, if you do not give the data in the format of `c` command, even then the output will be there. That output might be calculated in a very different way than what you want.

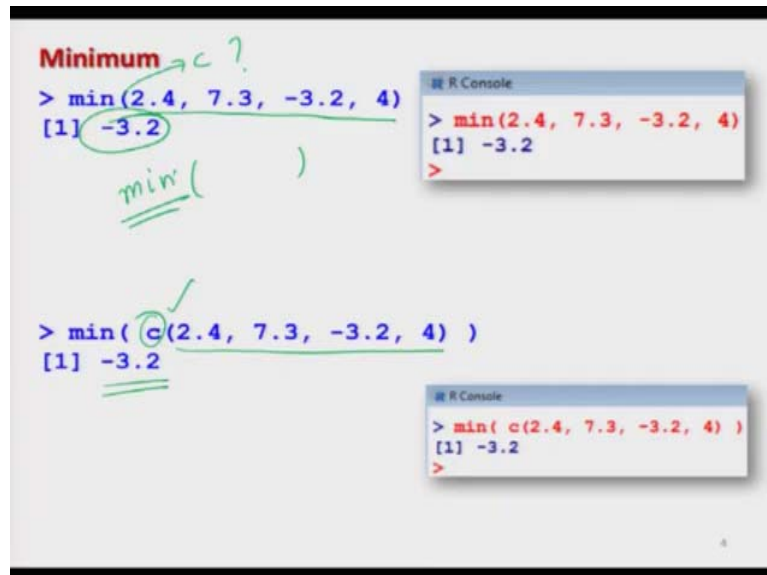
Suppose you want to find out the sum of 3 numbers 1, 2, and 3. And if you simply write say `sum` within parentheses 1, 2, and 3, possibly it may give you an answer only 1, that may not add 2 and 3. But if you try to use the command `sum` and within parentheses you try to give the data in terms of `c`, `c` within parentheses 1, 2, 3, then possibly it will give you the answer 6. So, now, you can see what will be the consequence, when you are trying to write down the bigger program in the data science, you want to compute a complicated issue you want to compute it and somewhere you are trying to use this function like `sum` or `maximum`.

And if you did not give the `c` command it will not show you any type of mistake or error, the R will compute the thing and R will not even inform you that there is any mistake or problem. And once you are trying to do such complicated calculations at the end, you will not be able to judge whether R has done the correct computations or not because you have given the command correctly to `sum` that is `sum`.

So, that is why in order to make the life simple and to avoid any type of problem in the programming, my very simple advice and sincere advice to you all is that whenever you are

trying to use a Built-in function try to give the data using the c command that is the safest. So now, let us come back to our slides and try to see some more operation I will try to show you these operations on the R console also.

(Refer Slide Time: 10:01)



So, now, similarly, if you try to want to find out the minimum of the given values, then just like this maximum, I have the command `min` minimum. And inside the pair of parentheses, you have to give the values and it will try to give you the minimum value out of the values inside the parentheses and then minimal also I have that same issue, if you try to use the command `c` or do not use the command `c` to input the data, it will give you the correct value. For example, you can see here that I am trying to find out the minimum of 2.4, 7.3, -3.2, and 4 and it is giving me the answer -3.2.

But here I am not using the `c` command. But in case if you are trying to do the `c` command here, then the minimum of the values 2.4, 7.3, -3.2, and 4, this minimum is coming out we -3.2. But here you are trying to use the `c` command. So once again, I will suggest you that do not get into this complication and simply try to use the `c` command when you are trying to enter the data.

(Refer Slide Time: 11:02)

<u>abs()</u>	Absolute value
<u>sqrt()</u>	Square root
<u>round()</u> , <u>floor()</u> , <u>ceiling()</u>	Rounding, up and down
<u>sum()</u> , <u>prod()</u>	Sum and product
<u>log()</u> , <u>log10()</u> , <u>log2()</u>	Logarithms
<u>exp()</u> e^x	Exponential function
<u>sin()</u> , <u>cos()</u> , <u>tan()</u> , <u>asin()</u> , <u>acos()</u> , <u>atan()</u>	Trigonometric functions
<u>sinh()</u> , <u>cosh()</u> , <u>tanh()</u> , <u>asinh()</u> , <u>acosh()</u> , <u>atanh()</u>	Hyperbolic functions

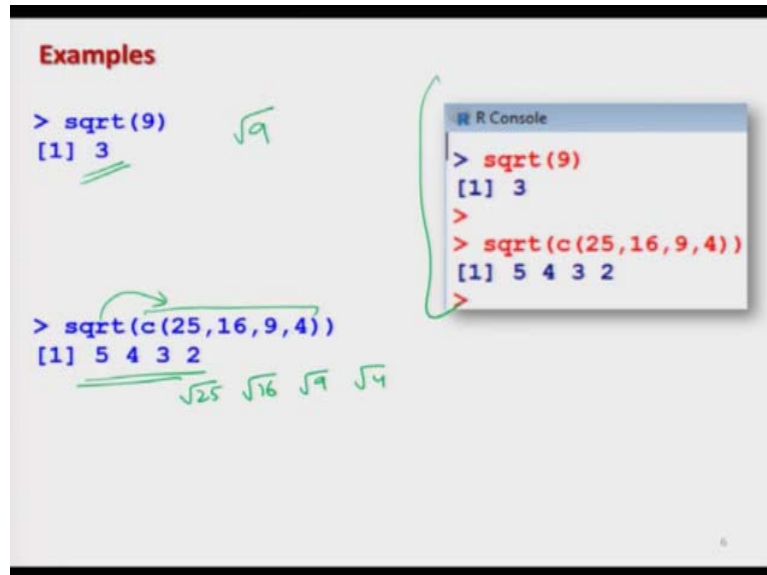
Now, similar to maximum-minimum we have here different types of command, there is a long list, but I just want to give you a glimpse and I am just trying to give you here a simple function which we expect that possibly we will be using them or I expect that you will be using when you are trying to write down the program for your data science, that is your ultimate objective that is why you are here. So, now, in case if you want to find out the absolute values, then there is a function here 'abs' and inside the parentheses, you have to write down the data as a rule I will always say that just try to use the c commands.

Similarly, in case if you want to find out the square root you have the command 'sqrt'. Similarly, for finding out rounding up and down of the number we have the command 'round, floor, ceiling'. And similarly, to find out the sum and product we have the 'sum' and 'prod' commands, that will mean if you try to write down the number inside the parenthesis, they will give you the sum or product of those number inside the parentheses. Similarly, if you try to find out different types of log functions, we have the command here 'log, log10, and log 2' etc.

And similarly, if you want to find out exponential function like it e raised power of here x, then you have the command here 'exp'. And similarly, if you want to find out here at any trigonometric function, we have the Built-in commands here 'sin, cos, tan', for cosec it is 'asin', for a sec it is 'acos', and for cot it is 'atan'. And similarly, for the hyperbolic functions, we have the 'sinh, cosh, tanh, asinh, acosh, atanh'. And then there is a long list, but

instead of going through these commands that we try to hear, take here some example to show you that how they are going to work.

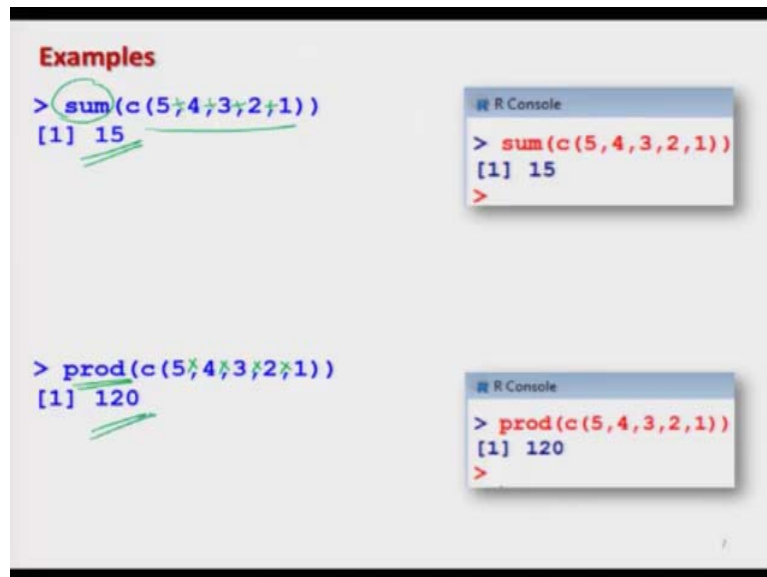
(Refer Slide Time: 12:45)



And then what is the advantage of this Built-in function when you are trying to work in the R software that is my ultimate goal. So, suppose if I try to take a scalar, and if I try to find out the square root, suppose I want to find out the square root of 9 like this. So, you can see here, this is coming out to be simply here 3. And now I try to use the same sqrt command over a data vector. Suppose my data vector here is 25, 16, 9, 4.

So, you will see here when this square root function is operating on the data vector, this will become like a square root of 25, square root of 16, square root of 9, and the square root of 4. And you will get an answer here 4,5,3,2. And this is the screenshot. So, we can see here that there is a Built-in function that is also operating over the data as a vector.

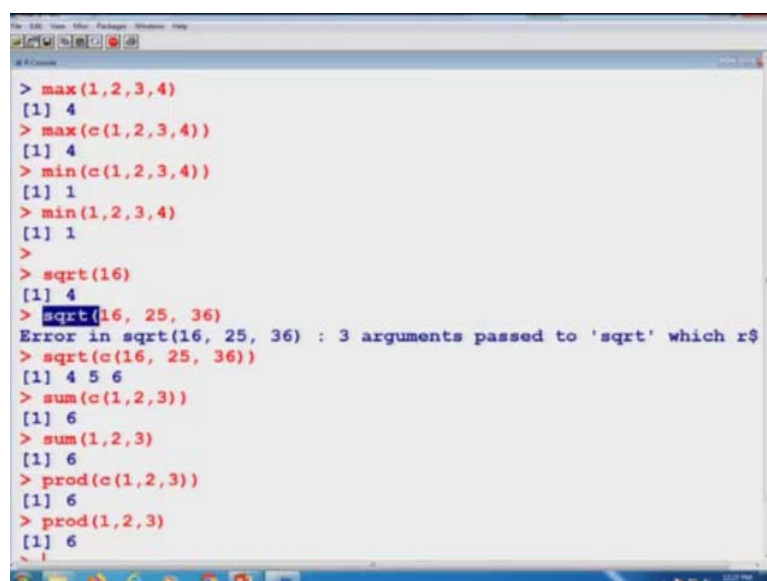
(Refer Slide Time: 13:38)



And similarly, if you want to find out here the sum and product function, suppose you want to find out the sum of a set of numbers, so you simply try to give those numbers using the c commands and try to use the Built-in command here sum, s u m will give you the sum of all the numbers which are written inside the parentheses using the c command. And you will see here that if I try to write down here, sum of c 5,4,3,2,1, this will give me the sum of 5 plus 4 plus 3 plus 2 plus 1 and this answer will come out here to 15.

And similarly, if you want to find out the product of the numbers just try to give those numbers inside the parenthesis just like sum. You use here prod product function, and this will give you the output of 5 into 4 into 3 into 2 into 1 which is here 120. So, before we try to go further, let me try to show you these things on the R console itself.

(Refer Slide Time: 14:35)



So, if I try to take here the first example of maximum, so if you try to see here if I try to take the maximum of 1,2,3,4. This is giving me the 4, but if I try to give here the values here like maximum of c 1,2,3,4, that is the same data set, but I am trying to give the values inside the parentheses using the c command. So, this will give me the value here 4. So, you can see here it is not making any difference.

And similarly, if you want to find out here the minimum then instead of here m a x you try to use the command here minimum, minimum of c 1,2,3,4, will be 1 that we know and if you remove the c command from this minimum command even then you will see the answer is coming out to be here 1,2,3,4.

Now, in case if you try to find out here the square root of some value here say 16 you can see here that answer is 4. Well, I am trying to take an example where you know the answer so, that you can verify. That whatever R is doing that is convincing with what you know. And similarly, if I want to find out say here is square root of 16, 25, and 36. Suppose, I do not give here the c operator and you can see here it is not working, but if you try to give here these three values using the c command over here, then you will see that this is working.

And similarly, if you try to use here the sum command suppose if I want to sum here 1, 2, and 3, three number 1, 2, and 3. So, I try to use the c command, and if I try to give here s u m it is giving me 1 plus 2 plus 3 which is 6. But, in case if I try to remove here the c command. Let us try to see what happens it is again giving me the number here 6. And similarly, if you want to want to find out here the product of here c 1, 2, 3.

So, you can see here that we know 1 into 2 is 2, 2 into 3 is 6. So, the answer is coming out to be 6 and you can see here whether what will happen if I do not use here the c command that is still working. So, it may happen that in some cases c command is working and in some cases, the c command is not working. For example, you have seen here in this square root command, this is not working. So, now, let us try to consider here some more commands over here.

(Refer Slide Time: 17:11)

Assignments

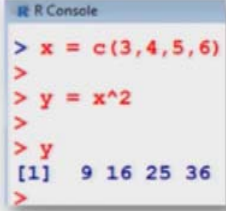
An assignment can also be used to save values in variables:

```
> x = c(3,4,5,6)
```

```
> y = x^2
```

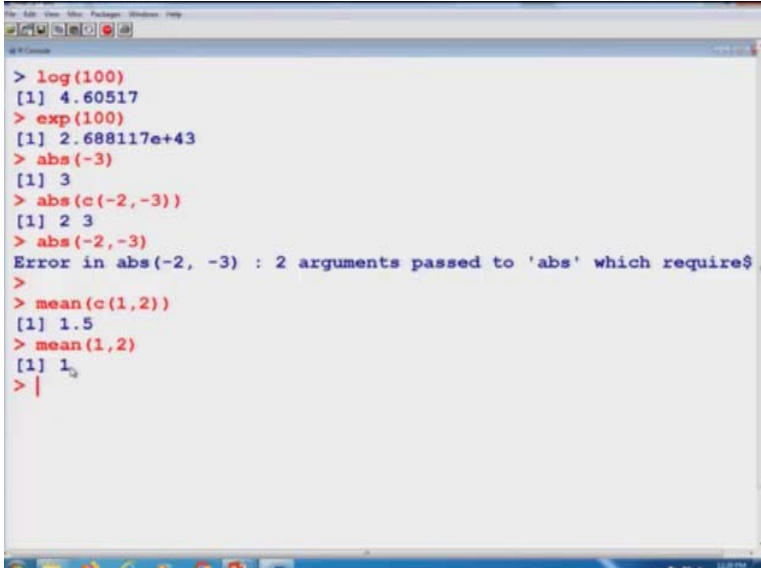
```
> y
```

```
[1] 9 16 25 36
```



Now, suppose I have a variable, which is also in the form of a data vector, where the inputs are going to be in the form of a data vector. So, now, you can see here that x is equal to here 3,4,5,6 that is my data vector, and if I try to define here a new variable y, which is simply here x hat 2 that is x square. So, this also can be done. So, y will become here 9, 16, 25, 36. So, that means when we are trying to define the variables in terms of data vectors, that is also possible that is what I am trying to show you here.

(Refer Slide Time: 18:10)



```
> log(100)
```

```
[1] 4.60517
```

```
> exp(100)
```

```
[1] 2.688117e+43
```

```
> abs(-3)
```

```
[1] 3
```

```
> abs(c(-2,-3))
```

```
[1] 2 3
```

```
> abs(-2,-3)
```

```
Error in abs(-2, -3) : 2 arguments passed to 'abs' which require$
```

```
>
```

```
> mean(c(1,2))
```

```
[1] 1.5
```

```
> mean(1,2)
```

```
[1] 1
```

```
> |
```

And this is how things will work without any problem. And similarly, if I try to show you here, some more operations, like as. Suppose, if I want to find out here, see here, log of some value here 100 you can see here, this is not the log base 10 but this is log base. And similarly,

if you want to find out exponential of 100 you can see here because it is coming out to be here 2.688.

And similarly, if you want to find out here the absolute value of here -3, this will come out to be here 3 and if you want to find out here the absolute value of here some data vector here -2, -3, and you can see here this is again coming out to be 2, 3, and if you want to know that, whether the `c` command will work in this command or not, you can see here this is not working.

For example, I can show you here one thing. Suppose if I want to find out the arithmetic mean of the 2 values 1 and 2, then the command here is `mean`. And if I try to give here `mean c 1, 2`, you can see here because it will come on to be 1 plus 2 divided by 2 which is 1.5. Now, means, I do not know whether the `c` command is needed here or not. So, I tried to give you here `mean 1, 2`, now you can see where this is going to give you only here the value 1.

Do you think that means 1 and 2 is 1. No, the mean of 1 and 2 is 1.5. But what it is doing here, it is giving you an answer 1 that means it is trying to take only here the first value. This is exactly what I was trying to tell you in the beginning that if you do not use the `c` command, for example, in the case of `absolute` it is giving you an error so you will come to know but in the case of `mean` this is not giving you the correct value, but it is giving you some value, this value is wrong, but when you are trying to execute this function inside a bigger program, then it will not be possible for you to check each and every step for such a big data set and then it will be computing something and you will never come to know that the value of `mean` has been computed as 1.5 or 1.

Whatever is the outcome that will be used as an input to another function. So, that is what I said that in the beginning that you have to be very careful when you are trying to use this Built-in function. So, my ultimate methods for using this Built-in function is this handle with care they are very useful, but you have to be very careful when you are trying to use them.



(Refer Slide Time: 20:51)

Bivariate plots:

Provide first hand visual information about the nature and degree of relationship between two variables.

Relationship can be linear or nonlinear.

We discuss several types of plots through examples.

Now, I try to come to another topic which we are going to use in our lecture, this is about the Bivariate Plots. You know about the plots that plots will give you different types of display and they also help us in retrieving a different type of information which is hidden inside that data values and R has a wonderful capability to create very beautiful informative graphics. And believe me, when you are trying to really venture into the topics of data sciences or the subject of data science, you have to be very good in exploring different types of graphics.

Graphics will play a very important role when you are trying to work in the data sciences because many times the information which can be understood by a graphic that is more difficult to explain by a formula or words. For example, if I try to create here a smiley like this one, you can see very clearly that this person looks to be happy, and if you try to write down this the same thing, but if I try to write down here this face to be upward, instead of upward it is downward, it will show that the person is not happy.

So, that is the advantage of using the graphics that you know very well, but instead of going through with all possible topics or all possible graphics, I am trying to take here only 1 plot which is a bivariate plot which I have used in the further lectures and rest depends on you that how much you want to understand and learn the graphics in the R software. So, these bivariate plots, they provide first-hand visual information about the nature and degree of the relationship between the two variables.

Now, this relationship can be linear or this relationship can be nonlinear. This means if you try to plot here a dataset, it may look like this or if a dataset is like this, it may look like this. So, there are different types of ways by which you can determine whether the relationship is

linear or nonlinear that we will try to learn when we are trying to understand the different topics of data sciences, but here I try to take 1 particular type of plot.

(Refer Slide Time: 23:17)

Bivariate plots: Scatter plot

Plot command:

x, y: Two data vectors

plot(x, y)

plot(x, y, type)

type	
"p" for <u>points</u>	"l" for <u>lines</u>
"b" for <u>both</u>	"c" for the <u>lines part alone of "b"</u>
"o" for both ' <u>overplotted</u> '	"s" for <u>stair steps</u> .
"h" for ' <u>histogram</u> ' like (or ' <u>high-density</u> ') vertical lines	

And I try to show you with the different types of examples that how this is really going to work. So, I am going to talk about the Bivariate Scatter plot in which I have two-dimensional graphics in the form of that we have here X-axis, Y-axis and then we have data here in the form of pair or something like x_i, y_i and then we try to plot this data here x_i and here y_i and then we try to plot these values and then they will try to give us a two-dimensional plot.

So, suppose I have a data set into data vectors x and y which are given here like this, then the command here is `plot(x, y)` inside the parentheses and after this, there are different types of options which are available and that can make the graphics as informative as you want. For example, 1 option here is to use the command here `type`. So now, in case if you try to use the type is equal to `p`, then the graphic will be in the form of points, which is actually the default.

And if you try to use here the type is equal to `'l'`, then the graphic is in the form of line and if you try to use here the type is equal to `'b'`, then you will have your points in line both if you try to use here `'c'` as type, then there will be lines which are only the part, alone of only `b` means out of this `b` only lines will be there. And then if you try to use type is equal to `'o'`, then both points and lines, they are overplotted and if you try to use type is equal to `'s'` then you will have a graphic in the form of stair steps and if you try to use here the type is equal to `'h'`, then the graphic is in the form of histogram or like high-density vertical lines.

(Refer Slide Time: 25:10)

Bivariate plots: Scatter plot

Plot command:

`x, y`: Two data vectors

`plot(x, y)`

`plot(x, y, type)`

Get more details from help: `help("type")`

Other options:

<u>main</u>	an overall title for the plot.
<u>suba</u>	sub title for the plot.
<u>xlaba</u>	title for the x axis.
<u>ylaba</u>	title for the y axis.
<u>aspthe</u>	y/x aspect ratio.

So, let us try to take a simple example and try to show you but before that, I just want to give you one information that. Well, in this slide, I am trying to give you some basic operation, but definitely, there are many more operations which are possible with this plot command and my request will be that you please try to look into the help menu to understand those commands, but some of the popular options which are needed to create such bivariate Plots.

Suppose you want to give the overall title of the plot then I have to use the command here ‘main’ in case if you want to have the subtitle title for the plot, I have to use here the command here ‘suba’. And similarly, if you want to have here the title on the X-axis, then you have to use the command ‘xlaba’ and if you want to use the title on the Y axis, then you have to give the command here ‘ylaba’ and if you want to control the aspect ratio, then you have to give the command here a s p t h e, ‘aspthe’ and then you have to give all these things over here.

Mean inside the parenthesis separated by commas, just try to write down these names, and then you try to give these options. Similarly, you can also change the color by using the command c o l etc. But once again, I will request you to try to look into the help and try to see the help plot or say help type that will give you more information.

(Refer Slide Time: 26:31)

Bivariate plots: Example

Number of marks obtained by students depend upon the number of hours of study.

Data on marks out of 500 maximum marks and number of hours per week for 20 students are collected as follows:

Marks out of 500 maximum marks
`marks <- c(337, 316, 334, 327, 340, 360, 374, 330, 352, 353, 370, 380, 384, 398, 413, 428, 430, 438, 439, 450)`

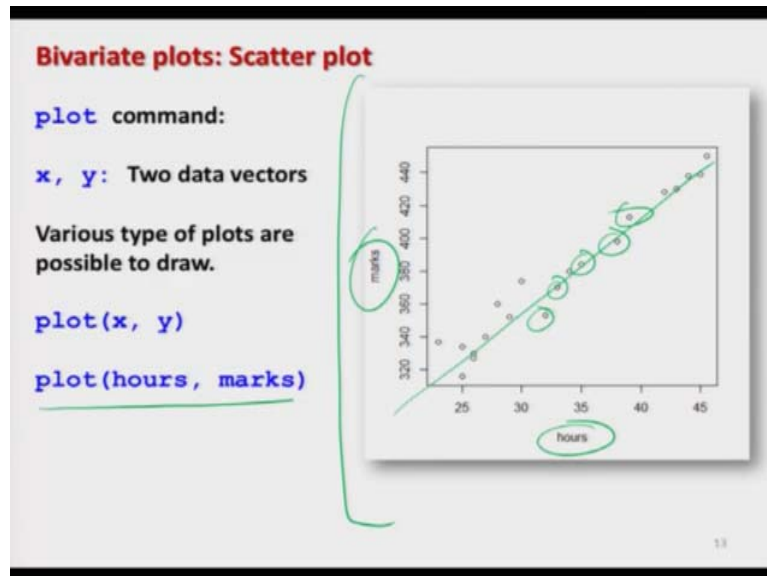
Number of hours per week
`hours <- c(23, 25, 25, 26, 27, 28, 30, 26, 29, 32, 33, 34, 35, 38, 39, 42, 43, 44, 45, 45.5)`

(Handwritten annotations in green: (23, 337) and (25, 316) are circled, with lines connecting them to the corresponding values in the code blocks above.)

So, now, in order to explain the use of these options in the plot command, we can try to take here a very simple example to explain to you that how are we going to work. Suppose I take here a simple example, where we have collected the data on 20 students on their marks and these marks have been given means out of 500 maximum marks. And you know the marks of a student they essentially depend that how many hours the student has studied. So, that is our belief. So, we have collected the appeared observation on the number of hours of study of a student and the marks.

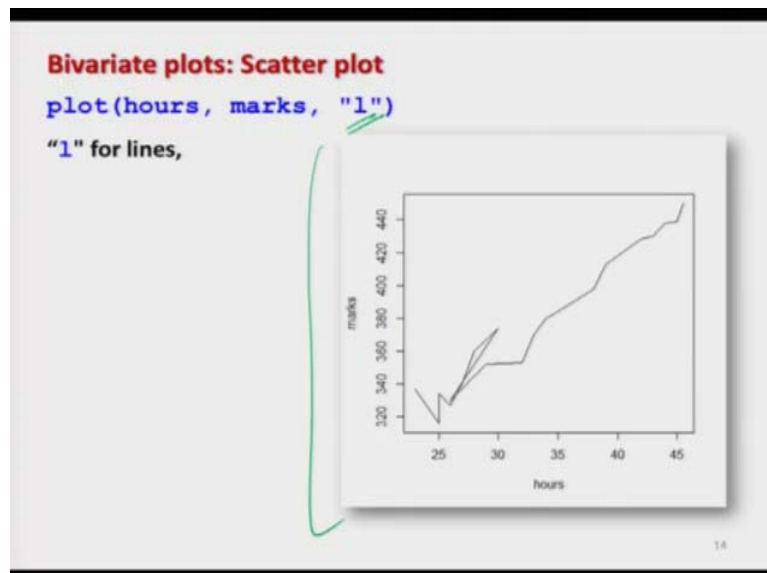
So, these 20 observations are here under the two data vectors. See here marks and here hours. So, if a student number 1 has studied 23 hours then it is evident from this data that this is a 23 in the data vector hours and then we have here 337 in the marks that is the first element in the data vector marks. So, this student who studied 23 hours has got 337 marks and this is here like this. And similarly, the second student has studied for 25 hours and the student has got 316 marks. So, it has been written in 25, 316 and so on and this data has been stored exactly in the same way in these two data vectors.

(Refer Slide Time: 28:08)



Now, I tried to use here the command `plot`. If you simply try to use here the command `plot x y` by default it will give you the points. So, I try to use here these two data vectors and if I write `plot, hours, marks`. And you will get here this type of graphics you can see here that this is here hours, this is here marks and you can see here these are their different data points. So, this is indicating that as if there is an approximately linear relationship between the number of hours of study and marks, so that is giving you this type of information.

(Refer Slide Time: 28:44)



Bivariate plots: Scatter plot

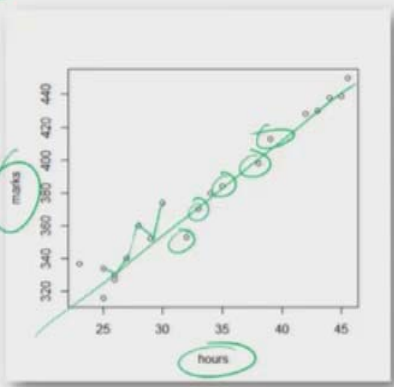
`plot` command:

`x, y`: Two data vectors

Various type of plots are possible to draw.

`plot(x, y)`

`plot(hours, marks)`

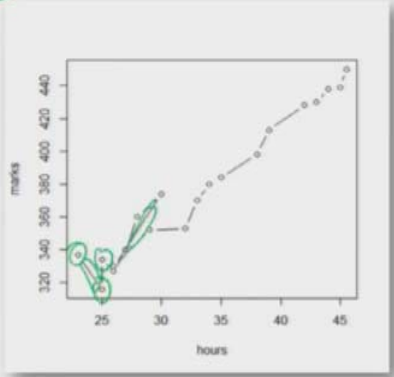


13

Bivariate plots: Scatter plot

`plot(hours, marks, "b")` *type = "b"*

"b" for both – line and point

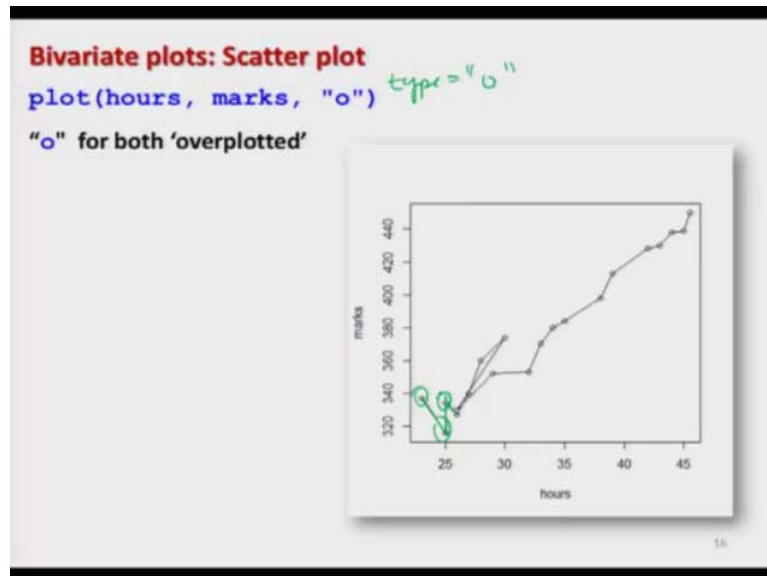


15

And similarly, if you try to use here, the type is equal to 1 then what will happen all these points which are here they are joined together like this one and so on and this is the output here. So, there are no points, but those points have been joined by lines and that is why the type here is 1 which means lines. And similarly, if you want to have here points and lines both then you have to give here the type is equal to 'b'.

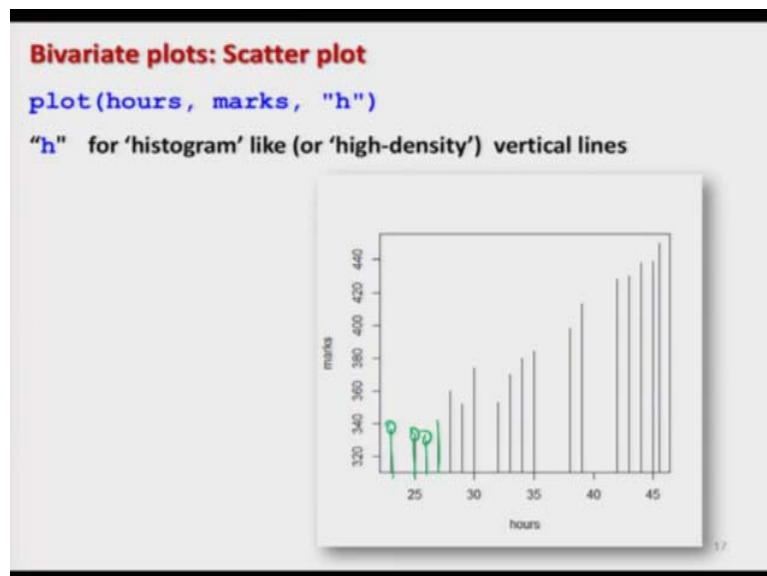
You can also write down type is equal to within double quotes try to write down here 'b'. So, b means both lines and points both are there. For example, you can see here these are my points and these are my lines and so on.

(Refer Slide Time: 29:34)



So, now, we take 1 more option that you can see here, that here these lines and here points they are separated, but if you want to join them together, then the option here is you try to use here, type is equal to 'o', 'o' means overplotted, that you can see here, that there are points here and these points are joined by the line and these lines are going over the points. So, this option will give you this type of graphic.

(Refer Slide Time: 30:09)



And then in case if you use here the type is equal to h, h means high-density vertical lines also histogram type of thing. So, all these points which are here they have been dropped with a line on the X-axis and different type of information, we are going to use it. So, you have to understand, how they are going to work and how they are going to interpret in the Decision

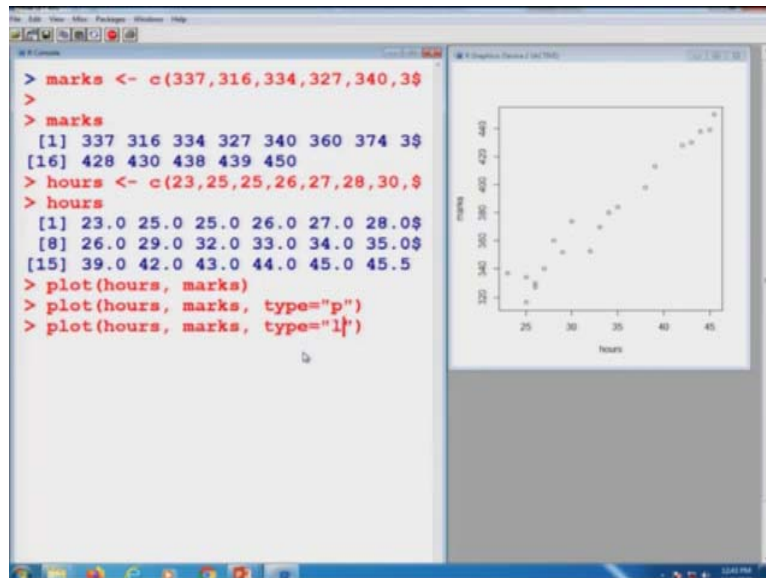
Sciences that we will try to discuss when we try to talk about the probability mass functions. So, I just told you this thing, so, that you could be confident enough that we are going to use these topics over there.

(Refer Slide Time: 30:48)



And similarly, in case if you want to use here that type equal to 's', then you will get here this stair type of curve. You know, what is the meaning of a stair step, do you know that stairs are like this and if somebody walks over here, it can go to the roof or some height and so, on. So, far that you have to use here the type is equal to small 's' and this will give you this type of plot over here. So, now, what I try to do here that I try to use this data and I try to show you these things on the R console also. So, let me try to just copy this data to avoid or to save some time on this R console.

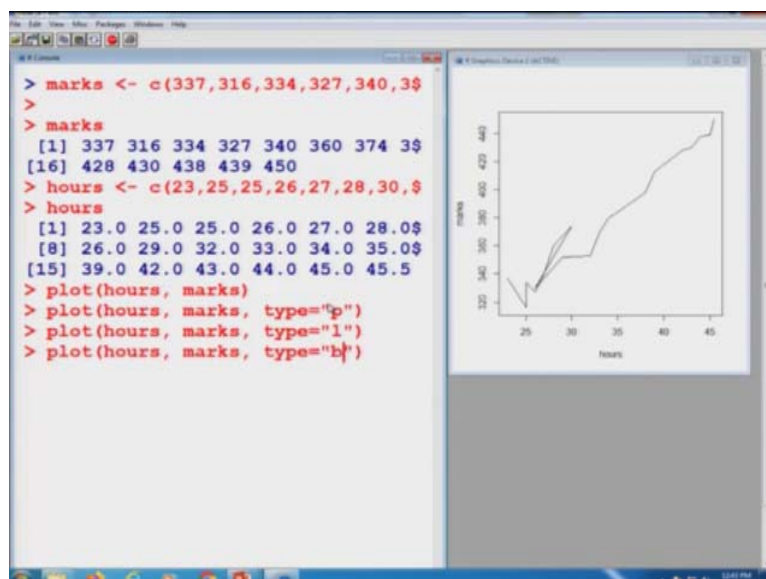
(Refer Slide Time: 31:38)



So, you can see here this is my marks and then I tried to see here the data on hours. So, you can see here, this is our hour's data and I tried to create here some space so, that I can show you the graphic very clearly as this happened. So, you can see here now, this hour's data is like this. And if you try to plot here, plot between say hours here marks. So, you can see here these graphics created like as here.

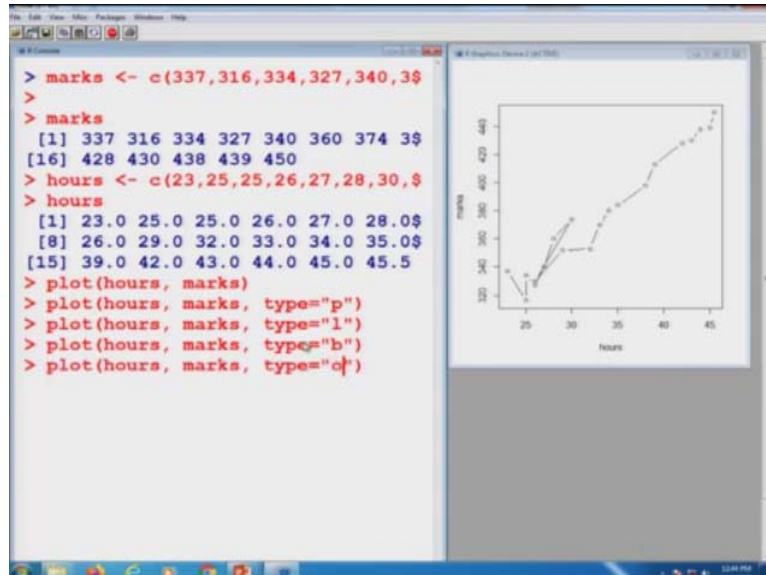
And similarly, if you try to change the type for example, you as I said that here that type is equal to if you try to use it here point well this is the default and you already have got the point. So, it will not make any change. I tried to make this graph a little bit smaller that you can see both the things.

(Refer Slide Time: 32:30)



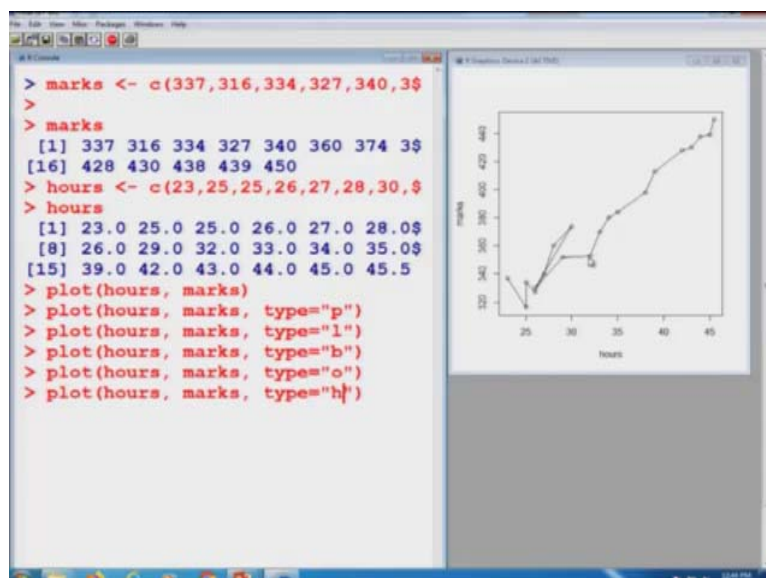
Now, I try to change here that this type you can see here very clearly both the windows together I tried to make it here type is equal to l that means line you can see here that the same data has been plotted using the command l type plot.

(Refer Slide Time: 32:46)



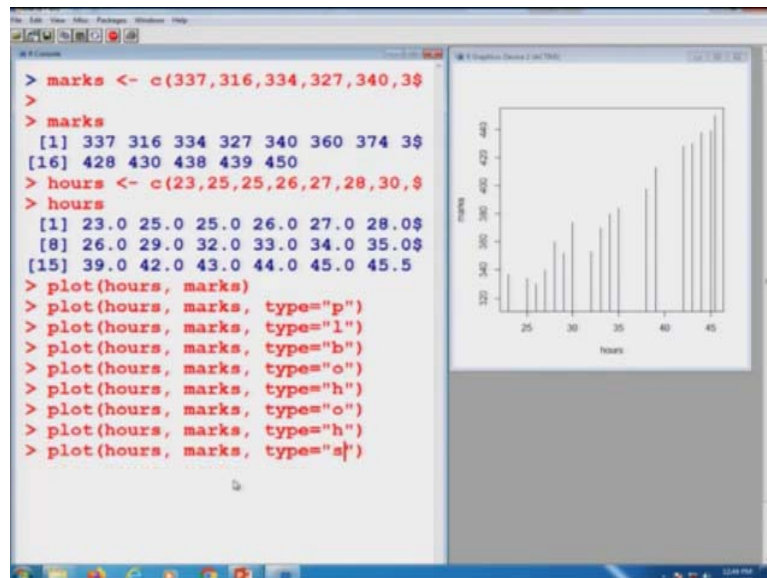
And similarly, if you want to have here points and lines here both you simply have to type here type is equal to b that means both and you can see here this graph is changed and then you have here points and lines both.

(Refer Slide Time: 33:00)



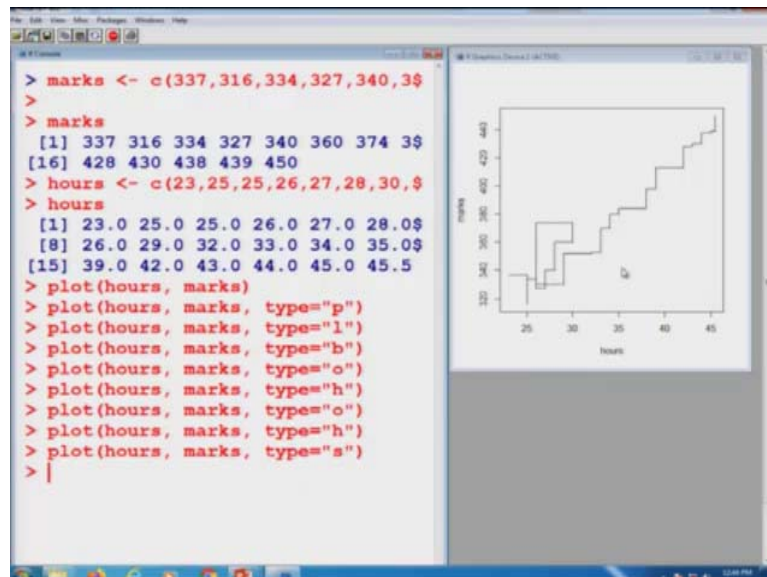
And similarly, but they are separated, but if you want to over plot them, just try to use the option type is equal to o and you can see here this is the figure in which the points and lines they are overplotted.

(Refer Slide Time: 33:13)



And similarly, if you try to use here the option H, then you will get your histogram type or say high-density vertical lines that you can see here formation of the point means you can see here if I try to show you here this thing and if I try to show you here this thing from the same point these lines will be dropped you can see here.

(Refer Slide Time: 33:33)



And similarly, if you want to have the plot of stairs, steps type, so you have to use that type equal to s and you can see here that this command is producing this type of graphic. So, now, we come to an end to this lecture and I have given you a basic idea, basic background about

the plot command and the built-in functions. But definitely, my idea was very simple. I just wanted to sensitize you that we are going to use some built-in functions and some graphics.

Well, when we are trying to use the built-in function, definitely we are going to introduce many more built-in functions. But my idea of telling you this built-in function was that how to use those new functions. So, the moral of the story what we have learned in the built-in function that you have to just write down the function and then you have to write down the data inside the parenthesis, but the rule what we are going to follow that we are always going to write down the data using the c command. Otherwise, I do not want to make the life complicated or confusing. As simple as that and similarly for the graphic also.

Well, I am going to use only these graphics because I will be working at a very basic fundamental level, but when you are trying to work in the Decision Sciences the data size is very big. And this is only you who is going to take a call that which of that tool whether this is analytical tool or graphical tool has to be used, your basic objective will be only that whatever is the hidden information inside the data that you have to bring out.

And for that what you have to do, nobody will come from the sky to inform you this is only you who has to think and who has to take a call and that is how you will become a data scientist. So, you think about this, try to have a quick revision of these commands, try to practice this, and I will see you in the next lecture with some more topics. Till then, goodbye.