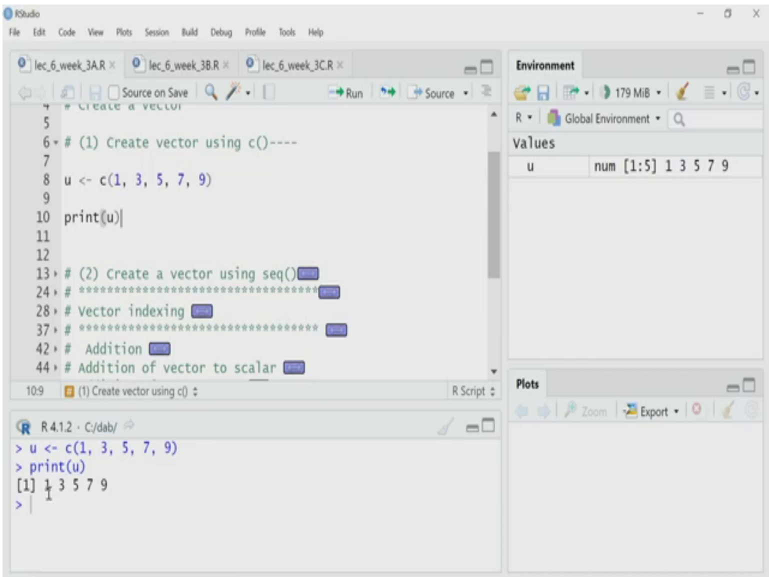
Data Analysis for Biologists
Professor Biplab Bose
Department of Biosciences and Bioengineering
Mehta Family School of Data Science and Artificial Intelligence
Indian Institute of Technology Guwahati
Lecture 19
Linear algebra using R

Hello everyone, welcome back. In week 2, we have learned about linear algebra. And in this lecture, I will imply implement those concepts that we have learned in linear algebra course using R. So, in this lecture, we will start with vectors, we will do vector operations, then we will move into matrices and do matrix operations like multiplication of matrices, calculating the determinant of a matrix something like that.

And eventually, we will solve a system of linear equations using matrix algebra, linear algebra. So, let us start with vectors, I will create vectors and then perform the vector additions like vectors multiplication, and all other things that we have learned in linear algebra on vectors.

(Refer Slide Time: 01:23)



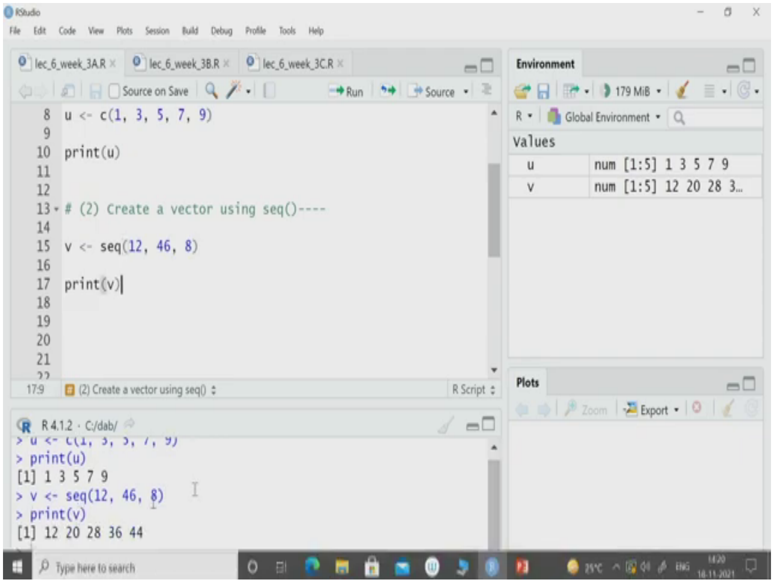
u ← c(1, 3, 5, 7)

print(u)

I will show you creation of vector using a two function one is c and other one is a sequence, both of these you have made earlier in the previous lectures. So, let us start with c function. I want to combine some data to create a vector. So, I have five data values 1, 3, 5, 7 and 9, I want to combine them together using the c function so that I will create a new variable u as written here in the code and that u will be vector, 1-dimensional list.

So, if I execute u assign to c and those values then u is created in the environment you can see u has been created. Now, I can print u and you can see, it is saying u is 1, is a list of 1, 3, 5, 7, 9. So, we can call it as a vector, it is a 1-dimensional list so I can call it a vector. Now, apart from c as we have learned earlier, I can create a vector using the sequence function also.

(Refer Slide Time: 02:28)



v ← seq(12, 46, 8)

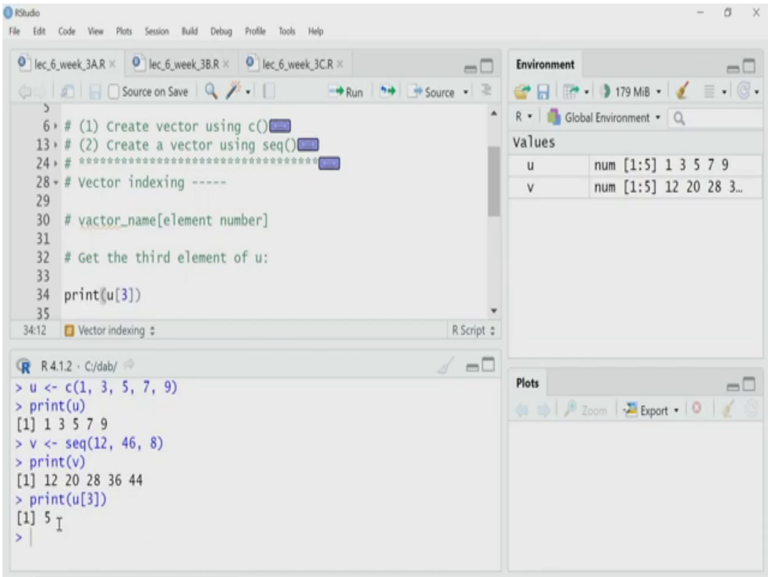
print(v)

So, let us do that. So, what I am doing here in this, I want to create a vector that will start with 12. And it will go up to 46 with an increment of 8, that is what I do in sequence function, we give the first value the lowest value or the first value and then we give the upper limit and we give the increment, so that is what I have given here, 12 is the first value, 46 is the last value, 8 is the increment.

So, let me execute that and that will give me a v vector because I assign that to v. So, if I print v, I can see it is starting with 12, then we are adding 8, so it is becoming 20 and it is going up to 44 because if I add 44 plus 8, it will be 52. So, I cannot go beyond 46. So, it stops at 44. So, I have got two vectors. So, I have created these two vectors, you can read a data file and extract vectors also, we have done that earlier.

Now, once I have got the vector, one important thing that we have to understand is indexing of it. What is a vector? If you remember we have said that vectors are stacked numbers. So, now, if I have stacked numbers, 10 numbers, 3 numbers or 4 numbers, I should be able to label them. So, in R level start from 1 to or 2, 3, 4 something like that. So, I may ask that okay, I want a third number in my vector. So, that is a third index. So, I should be able to get a particular value of having a particular index number to that. So, let us try to do that.

(Refer Slide Time: 04:11)



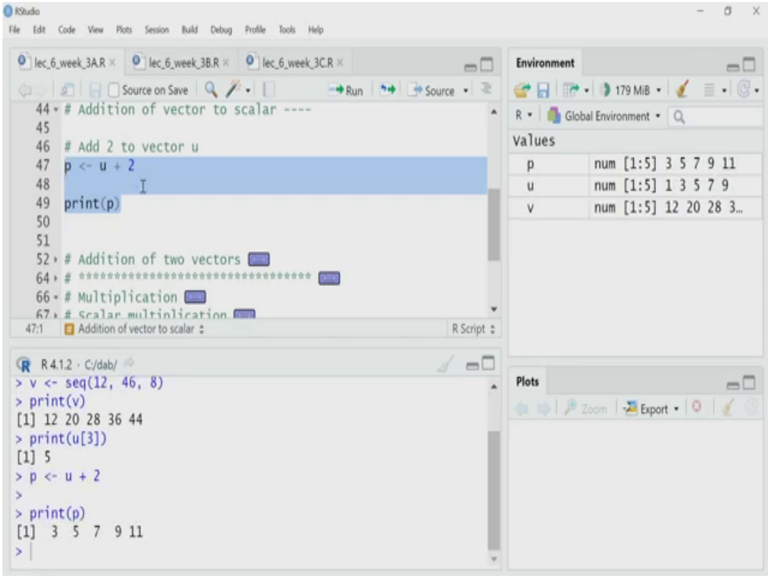
print(u[3])

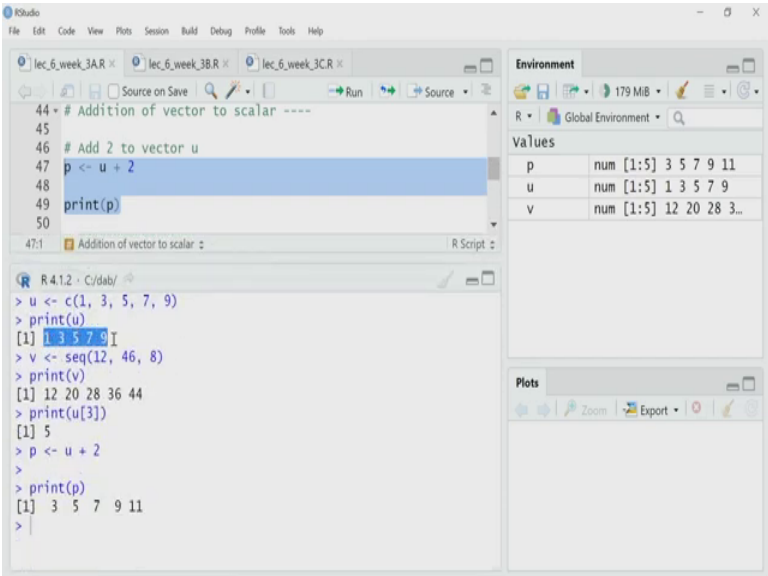
So, how would you write it? We write the vector name, and then we put the index number or element number. For example, as I have shown in the script, if I have created a u vector, few seconds back if you look here in the console, u is 1, 3, 5, 7, 9. These five numbers stacked. I want to get the third number, what will be the third number, third number will be 5. The first number is 1, second number is 3, the third number is 5.

So, I want to get the third number. So, how I should write? I should write it u and in the square bracket I should write that index 3. So, u in square bracket 3 means it tells R that okay, I want the third element of this u vector, and then I am putting it under the print so that I can print it. Let us print that. So, it says 5, obviously, because the 5 is the third element or the third number in my vector u.

So, this is how you actually use indexing of vector, you may want a part of that vector and do some further calculations. Now, in the lecture on vector and vector operations in the last week, we have learned about addition of vectors and multiplication of vector. So, let us start with addition. So, addition can be of two type, I may add two vectors, I may add a scalar to a vector also, the simplest one is always the addition of a scalar to a vector.

(Refer Slide Time: 05:43)





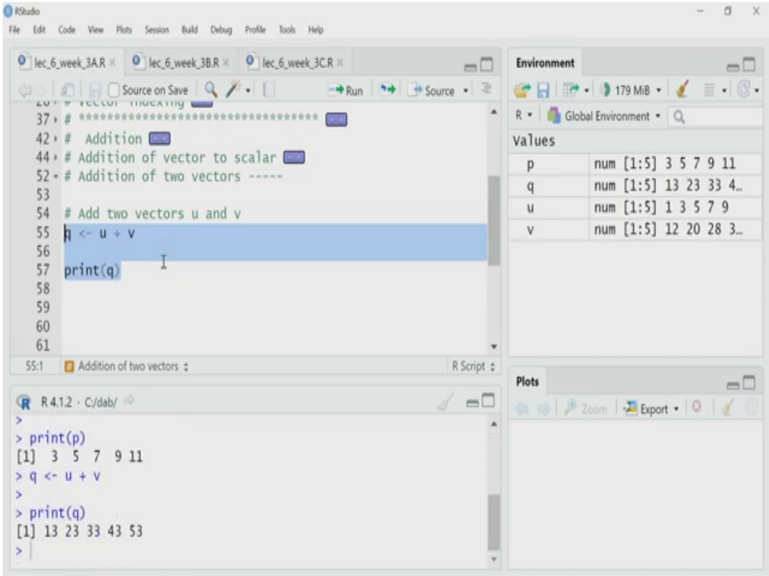
p ← u + 2

print(p)

So, suppose I want to add 2, 2 is a scalar value thing, to a vector u, to the vector u, u we have created few minutes back. So, how do I do a simple addition, I have to use the addition operator the plus sign here. So, u plus 2 will be summation of a vector u and the scalar 2, and it will give me a new vector, which I have assigned to p, and then I am printing this. So, let me execute both of them together.

So, what I have got, I have got a new vector 3, 5, 7, 9, 11, you can easily see my original vector u has 1, 3, 5, 7, 9. So, if I add 2 to them, the first element will become 3, the second element will become 5, and so on. So, I have done scalar addition. Now, I want to do vector addition. I want to add two vectors. So, it should be similar, I only add the addition operator here.

(Refer Slide Time: 06:47)



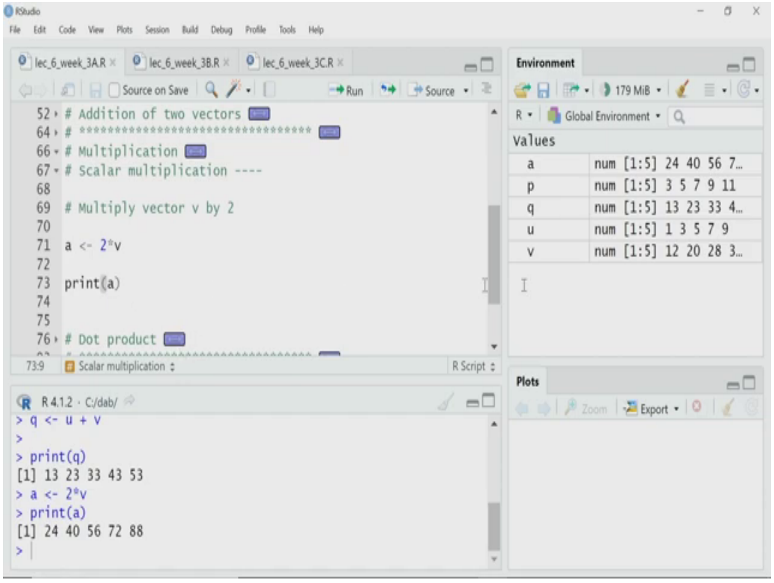
q ← u + v

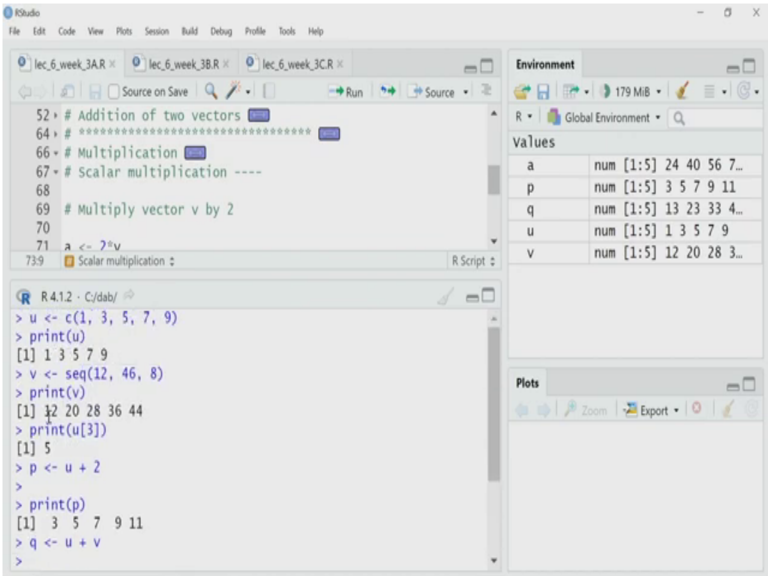
print(q)

So, here what I am doing, I have created earlier two vectors, u and v and I want to join them, sum them together using the addition operator. And I will assign this whole thing to a new vector q. And then in the next line, I am asking to print the q, so that I can check what is that vector created by addition.

So, that is what we have got, we have got a new vector q with elements of the values 13, 23, 33, 43 and 53. So, we have learned addition, it is very simple to do. Now, we move into multiplication, again, just like addition, I can have a scalar multiplication, and I can have multiplication between vectors. So, start with the scalar one, the simplest one.

(Refer Slide Time: 07:34)





a ← 2*v

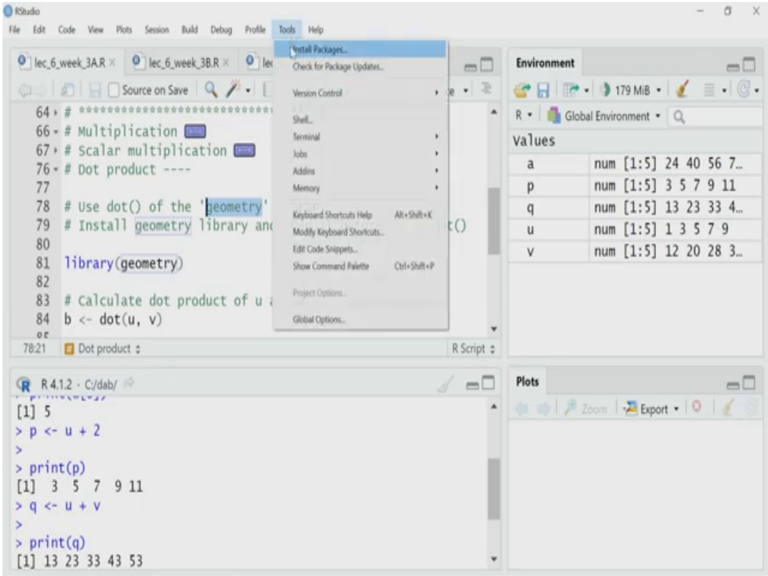
print(a)

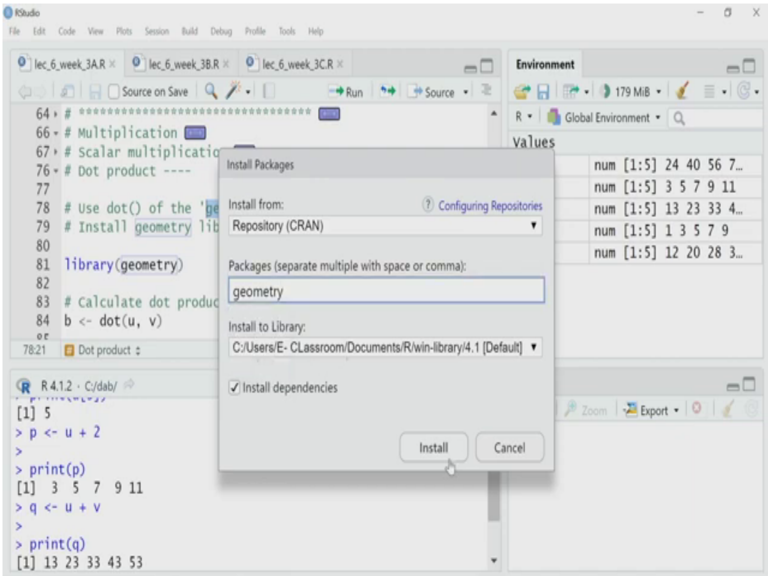
So, for scalar multiplication, suppose I want to multiply the v vector with 2. So, what I will write, I will write 2, and then I will give the star, the star is the operator for multiplication if you remember, and then the vector I have written and I am assigning that because this multiplication will give me another vector. So, I am assigning that vector to a, new variable, and then in the next line, I am printing a so that I can check it.

So, let us execute this. So, I have done the multiplication, scalar multiplication and let me print it. My v if you see, let me expand the console v is 12, 20, 28 and so on. So, if I multiply by 2, I should be, it should become 24, 40 something like that and that is what has happened here. You can see a is a vector with first element 24, second one is 40 and so on. So, I have performed scalar multiplication.

Now, I want to multiply two vectors. And when it comes to multiplication of two vectors, we talk, we are talking about here about the dot product. So, I want the dot product between two vectors. Now, you can do dot product by using the explicit definition how we have defined dot product and then you can use the base operator and base functions of R to do that.

(Refer Slide Time: 09:01)



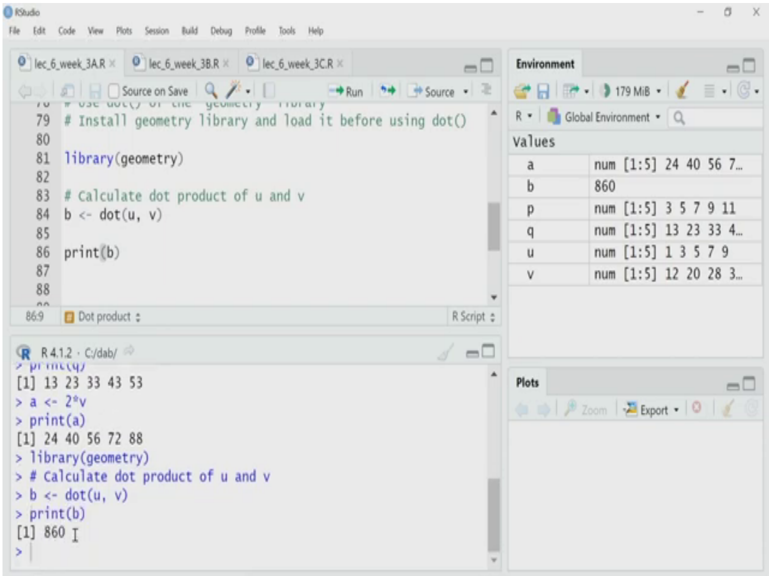


library(geometry)

But there is a library of R which is named as geometry, geometry library. And that library has a function called dot and that can do it in one line, the whole calculation of the dot product. So, just I have to give the two vectors as a argument to this dot function and it will give me the dot product. So, what I have done here, I have already installed this geometry library. If you do not know how to install, I will just show you briefly you go to this tool for this R Studio and then you click on this install package then it will ask what you want to install.

So, you can write here geometry and then you would simply click Install, I have already installed it so I will not do anything further. Now, once you have installed the library in R does not mean that library is now accessible to you while you are working. You have to make that library accessible to the session current session of work. So, how we do that for any library, if you have to incorporate in the current session, you have to call that library.

(Refer Slide Time: 10:08)



b ← dot(u, v)

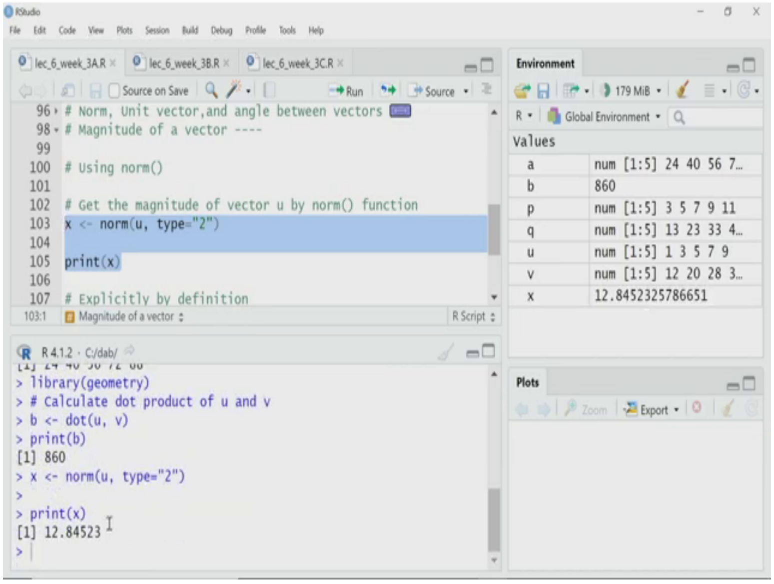
print(b)

So, we call that by using the function library, and then as the argument in the round bracket, you have to put the name of the library that you are calling and then you will use subsequently. So, I want to use the dot function, which is part of the geometry library, which I have installed earlier. Now, I want to use the dot, so I have to call the geometry library. So, I am calling geometry library here, in this line.

So, let me call that and the geometry library is now loaded in this session. So, now, what I will do, I will use this dot function to calculate dot product between u and v. So, u and v will be two arguments for this function dot. And if you remember, what should I get by the dot product? Do you remember? I should get a scalar. So, it is a scalar product. So, I am assigning that value to b.

So, let us perform that dot product. I have done the dot product calculation, now I want to print it, it is 860. So, I have shown you how to do simply the addition and multiplication in R for vectors. Now in vectors lecture, if you do not remember, go back and check we have also learned about magnitude or norm of a vector, we have learned what is called unit vectors, then we have learned what is the angle between two vectors. So, now, I will implement those in R.

(Refer Slide Time: 11:41)



x ← norm(u, type = “2”)

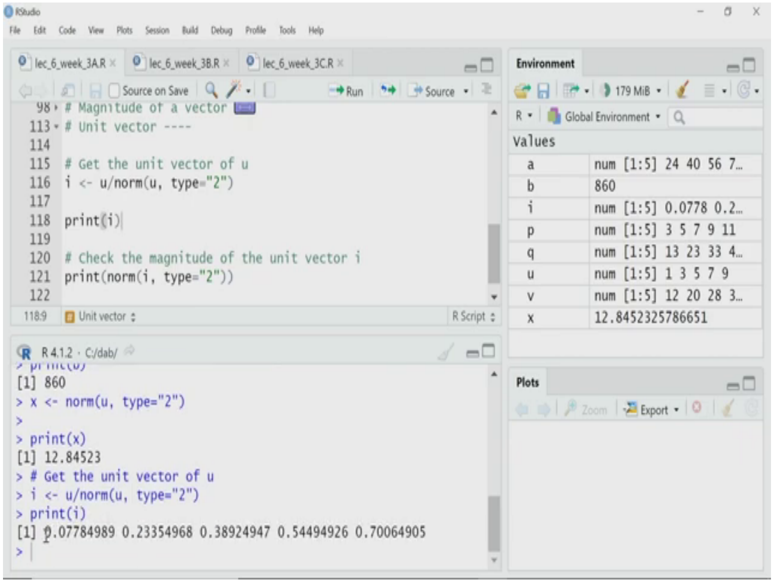
print(x)

So, the first thing that I will check is how to calculate the magnitude of a vector in R. So, what I will do, I will use norm function which is a base function present in the base R. So, these norm function will take two argument at least, that is one should be the vector for which you calculate the norm. And then you have to specify what type of norm. There are many types of norm, actually, the magnitude that we calculate for a vector is called the second norm, norm 2.

So, I have to specify that. So, what I am doing here in this line, I am saying that okay, you calculate norm, I am calling the norm function. And I am giving two inputs, two arguments, u is a vector for which I want to calculate the magnitude. And I am writing type is equal to 2. So, I am asking it to calculate the second norm, and it should assign that value, that will be a scalar value to x, and in the next line, I will print that value.

So, let me execute both of them together. So, it says the second norm or magnitude of my vector u is 12.84 something. So, I have calculated the norm. Now, I want to calculate unit vector. If you remember, unit vector is nothing but dividing a vector by its magnitude. So, you have taken a vector and you are normalizing with respect to its magnitude, so you will divide that vector by the second norm. So, that is what I will do simply.

(Refer Slide Time: 13:14)



i ← u/norm(u, type = “2”)

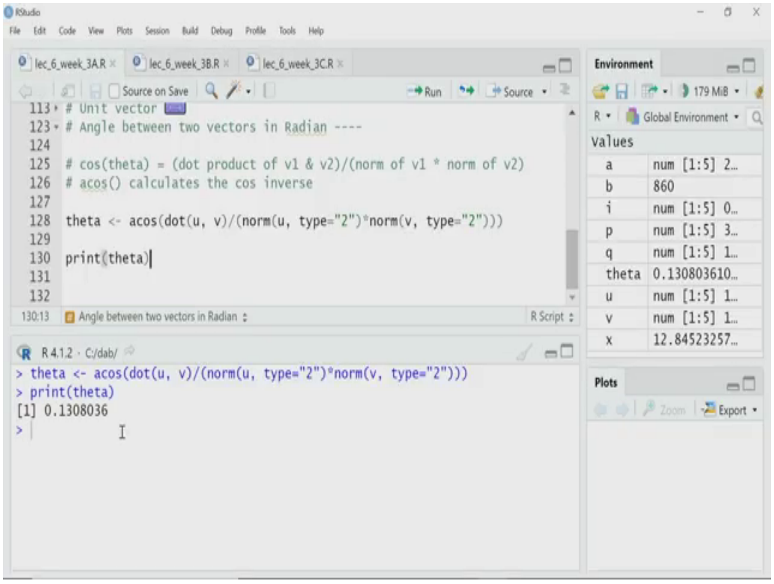
print(i)

print(norm(i, type = “2”))

So, that is what I am doing here in this line, I am taking that u vector, because I want to create a unit vector corresponding to u. So, I am taking that u vector, and then dividing it by its magnitude, just now I calculated the magnitude using the norm function. So, I am using the same norm function again.

And I will get a new vector, remember, unit vector is a vector, and I get a new vector that I assign to a variable called i. So, i will be my unit vector corresponding to u. So, I execute that, the i is created, but I want to see it. So, I write print i, and this is what I got. This is your unit vector corresponding to the vector u that we created earlier. So, just by simply dividing the norm, I got it.

(Refer Slide Time: 14:08)



theta ← acos(dot(u, v) / norm(u, type = “2”)* norm(v, type = “2”))

print(theta)

Now, I will go to the last operations for vectors that you have learned in our earlier lecture is that angle between two vectors. So, let me explain what is the angle, how we define the angle between two vectors. Usually, we know the relationship that the angle, the angle is theta, angle between two vectors is theta, then the Cos of that, Cos theta is equal to dot product of these two vectors.

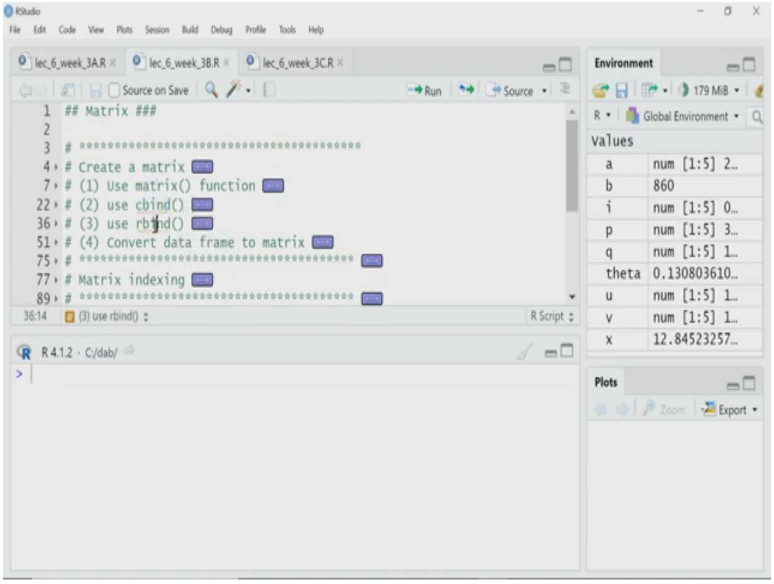
Suppose that two vectors are v1 and v2. So, you take a dot product of this vector and divide by the product of the magnitude of v1 and magnitude of v2. So, that is we know is the definition of Cos theta, when the theta is the angle between two vectors. So, I will use this formula to calculate Cos theta, and then I will inverse it, so that I will use Cos inverse to get the value of theta. So, how I am doing it, it is bit long, I will extend this.

So, the first thing is I have to take that dot product of v1 and v2 two vectors. So, I am using two vectors u and v. So, here I have taken it dot of u and v, so I am using the dot function to calculate the dot product, then I have to divide that by norm of these two vectors. So, the first thing I am taking and in the denominator here, you see the division operator, and then I am dividing by norm of u and the second norm of v, they are multiplied by this multiplication operator.

So, this whole thing, this whole thing will give me Cos theta, but I want to know the angle in radians. So, I will use Cos inverse. So, that is why I use a Cos function. And that Cos function will invert this and will give me the angle which I assigned to theta, theta is the angle in radian. And in the next line, I want to print that angle. So, it says the angle between these two vectors u and v is 0.1308 in radians.

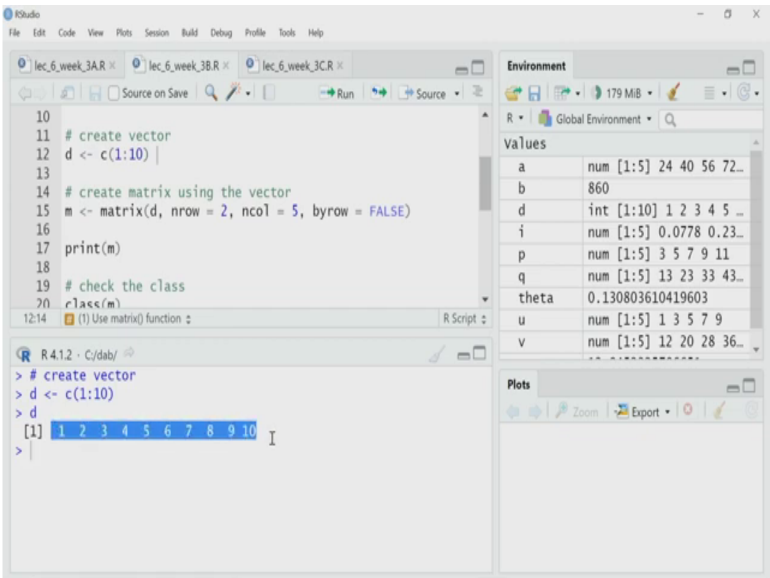
So, it is very easy to do, once I have the dot function and norm function, I can easily do that. That is all for vectors, there are many other things that you can play around with vectors, but we will not go into those, I am sticking to what we have discussed in our linear algebra classes. So, now I will move to matrices and I will create first matrices, I will show how to create matrices. And then we will do matrix operations like multiplication, addition, then we have to learn eigen value, how to calculate eigen value, eigen vectors, determinant of a matrix and so on.

(Refer Slide Time: 16:52)



So, let us start with creation of a matrix. How can I create a matrix? Here, I will show four methods. The first one will use the matrix function, then the second one I will show using cbind function, the third with rbind, and eventually I will show how to convert a data frame, if you know the table format of data into matrix. So, let us start with first the matrix function, how the matrix function helps us? Matrix function will convert a vector into a matrix. So, I will give a vector as an input to this matrix function and it will convert it into a matrix. So, let me explain with an example.

(Refer Slide Time: 17:37)



d← c(1 : 10)

```
m← matrix(d, nrow = 2, ncol = 5, byrow = FALSE)

print(m)
```

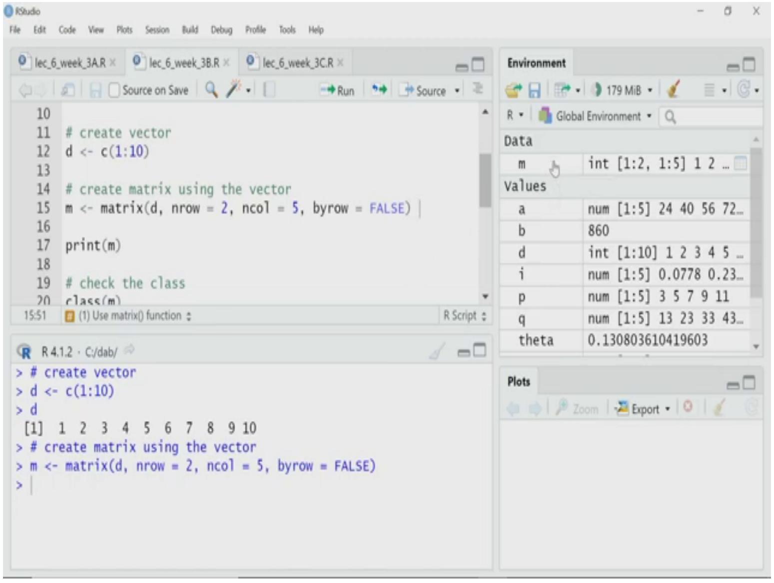
So, in the first line, here, I am creating a vector d using c function. And I am saying start with 1 end at 10. So, it will start at 1 and then take 2, 3 and 4 up to 10 and it will create a vector d. So, let me execute that. So, now it has created I can show you here this is d, it is written d and I have got those numbers. I can see also here in the console, if I write d I can see the vector.

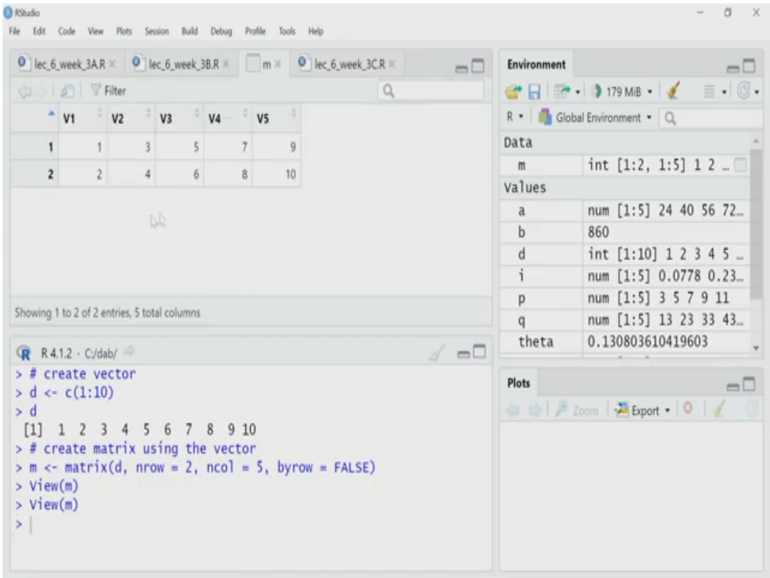
Now, I want to convert these vector which is values from 1 to 10 into a matrix, that means I have to break it in part and arrange them so that this 1-dimensional thing will become a 2-dimensional thing. So, that means I have to tell the function, who will do this job, the matrix function will do this job, that means I have to tell matrix function that how many rows I want and how many columns I want.

So, that is what I have done here. I am calling the matrix function. And the arguments are within the round bracket. The first argument is d of this is the vector which is the input, and then I am telling how many rows and how many columns. Here, I have said the n row, that is number of rows that I want in the new matrix is 2 and the number of columns ncol is equal to 5, 2 into 5 is 10. And I have 10 element, 10 numbers in my vector d.

So, that means I want to take these 10 numbers which are arranged in 1-dimension, and I want to break them to create two rows and five columns. Now, you can do it in two way. So, you can start like this. You can arrange them the 1, 2, 3, 4, 5 and 6, 7, 8, 9, 10, something like that. So, that will be row wise, or you can start with 1, 2, 3, 4, 5, 6, something like that, that will be column wise. So, that is why the last argument here is by row and I have said FALSE. That means I do not want to row wise, I want column wise. Let me execute it. It will be more clear.

(Refer Slide Time: 19:47)





d← c(1 : 10)

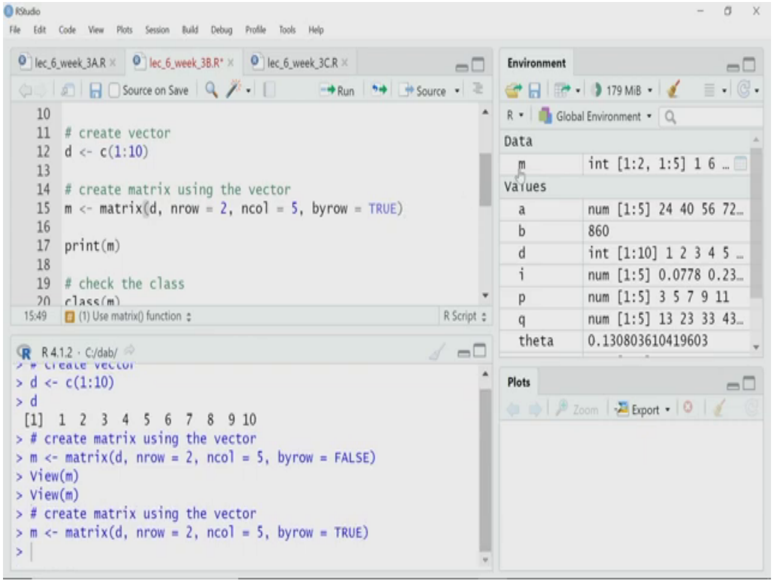
m← matrix(d, nrow = 2, ncol = 5, byrow = FALSE)

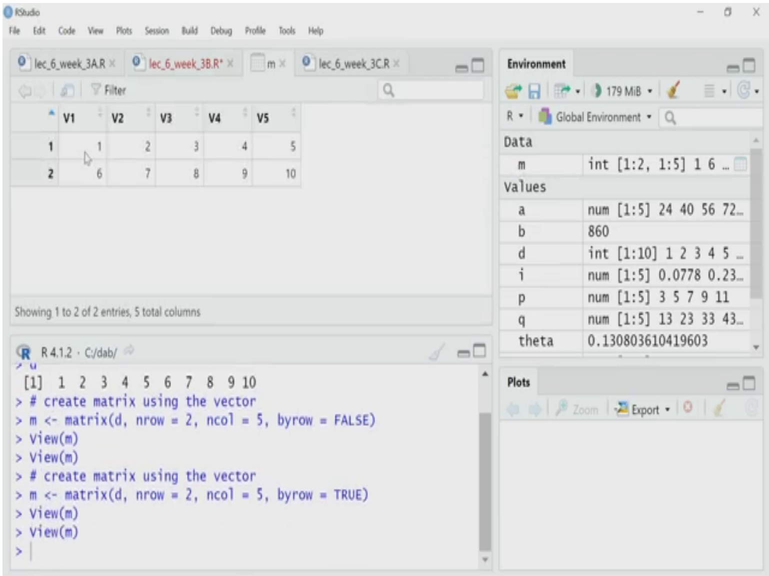
print(m)

class(m)

I have created m. You can go to this environment m double click it and you can see it here. See, originally I have 1 2 3 4 so on up to 10. Now, I have broken that in two row five column, but not row wise, I have said row wise is equal to FALSE, by row is FALSE. So, that means it is column row wise, that is why it is taken 1 2 3 4 5 6 7 8 9 10. So, that is how it has arranged.

(Refer Slide Time: 20:19)





d← c(1 : 10)

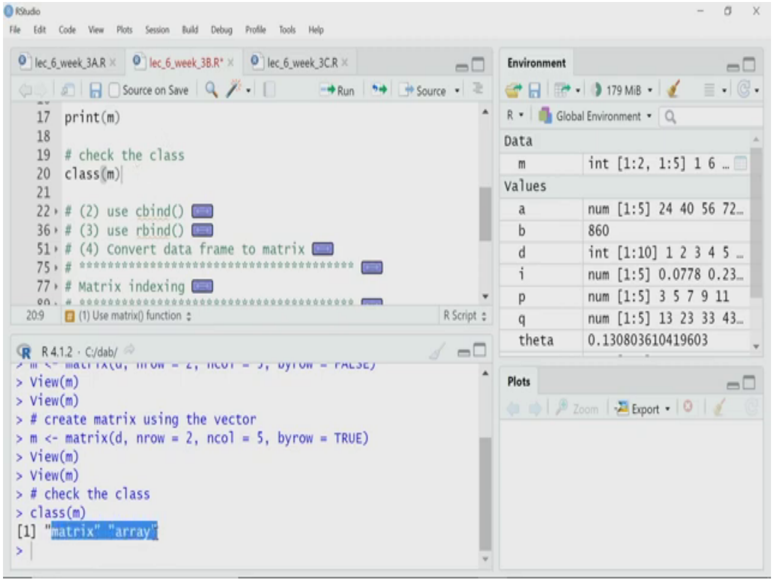
m← matrix(d, nrow = 2, ncol = 5, byrow = TRUE)

print(m)

class(m)

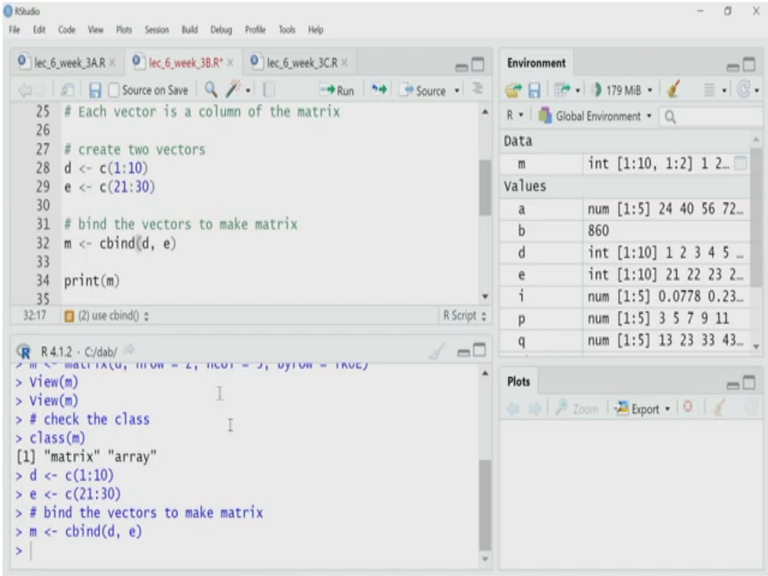
If I have said it equal to this equal to TRUE, then if I execute it again. Now, let me see m. Now, you see it is row wise 1 2 3 4 up to 5 in the first row, and the rest of the data or numbers are in the second row. This is how we actually can convert a vector into a matrix.

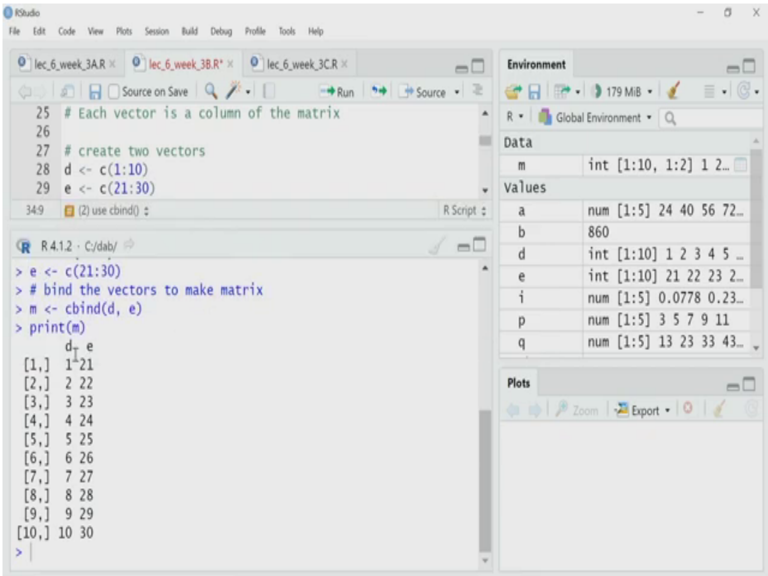
(Refer Slide Time: 20:49)



Now, let us see run one handy tool is to check that what we are doing. Are we really creating metrics or not is to call the class, class will tell me what data class is this variable belongs to. So, I have already created m. So, let me check what is this data class? And it says it is a matrix or array. That is what you want. You want a vector to convert into matrix or array. So, I have shown using matrix function.

(Refer Slide Time: 21:14)





d ← c(1 : 10)

e ← c(21 : 30)

m ← cbind(d, e)

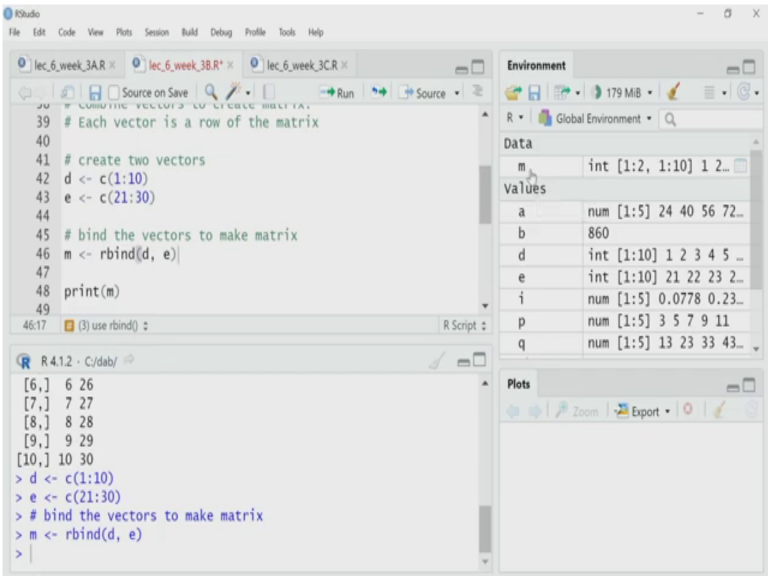
print(m)

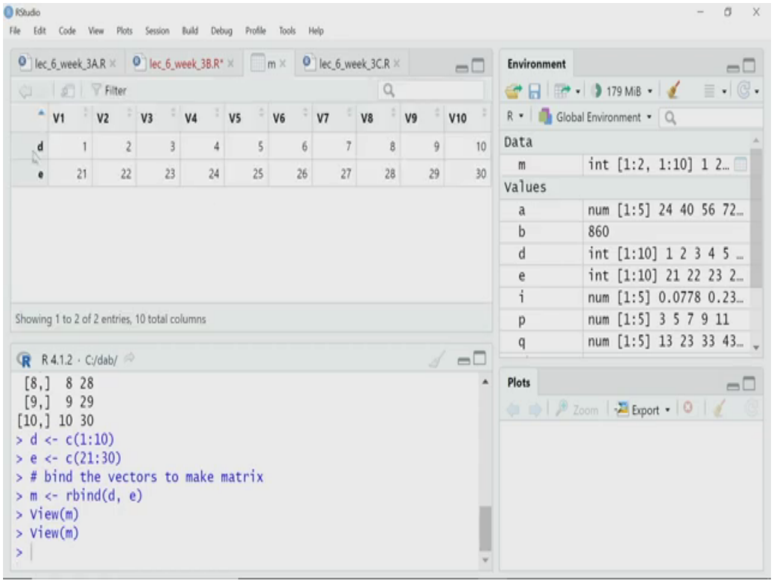
Now, I will use cbind function, column bind, bind by column. What is that? So here, I want to combine vectors to create matrices. In the earlier case, I had one vector, break it down in matrices. Now, I have multiple vectors, I want to join them to create matrix. And in this case, these original vectors, if you cbind, will be the columns of the new matrix. So, first, let me create two vectors to explain this.

So, I am creating two vectors, d and e here, in these two lines. It is a very simple, I am using the c function. So, I have created two vectors, d and e. Now, I want to combine them to create a matrix where these d and e will be columns. So, how I will do? I will call a cbind function. And the arguments are those vectors. So, the first vector will be d, the first column should be d, second column should be e.

So, if I execute it. So, I have created, let me print it. Here it goes. So, you see, the first column is d, second column is e, and vector d is there arranged as a column and e arranged as a column. So, I have created a new matrix m by binding column wise these two vectors. Now, I will use rbind, means I will bind vectors similarly, but row wise, the original vectors will become rows of the new matrix.

(Refer Slide Time: 22:56)





d ← c(1 : 10)

e ← c(21 : 30)

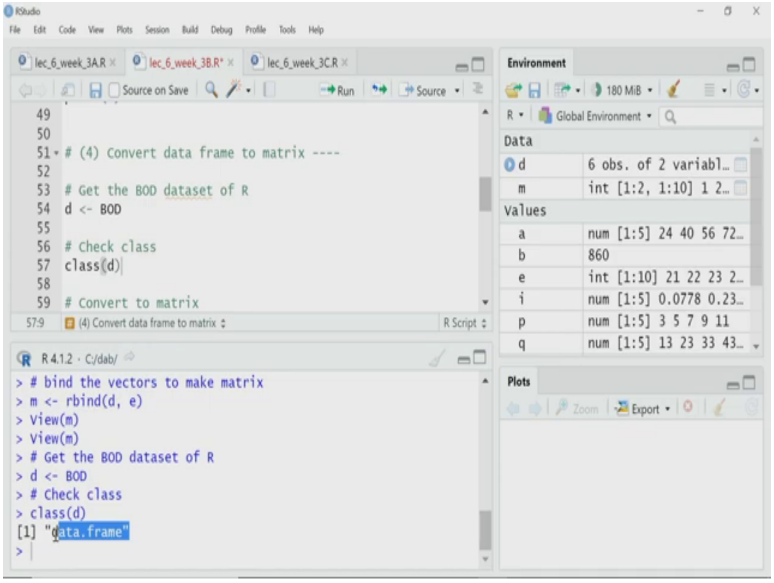
m ← rbind(d, e)

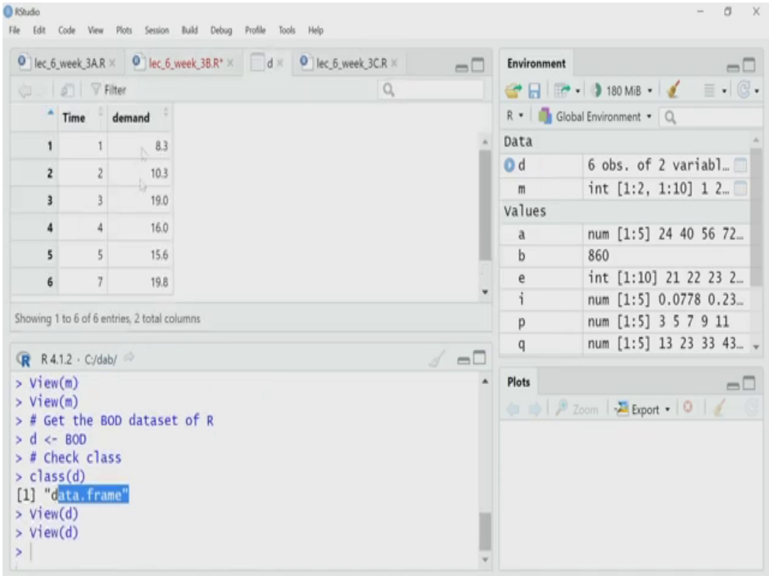
print(m)

So, to do that, I am creating again two vectors. And then I am joining them by calling the rbind function. Again, here, the arguments are d and e. So, d will be the first row, e will be the second row. So, I have created it. I can go to either print it or I can go and click on the environment pane on m, I should be able to see. Now, you see, d and e are not, no more columns, they are rows. So, the first row is d, vector originally 1 2 3 up to 10.

And the second row of the new matrix is the e vector starting with 21 up to 30. So, this is how using cbind and rbind I can create new matrices by combining vectors. Now, come to the last thing of creating metrics. I want to convert a data frame, means you have read a data in a table format from a CSV file or you have used inbuilt the data set whatever you are getting you have a data frame, a table. Now, you want to convert that into a matrix and so that you will use matrix operation on that.

(Refer Slide Time: 24:12)





d ← BOD

class(d)

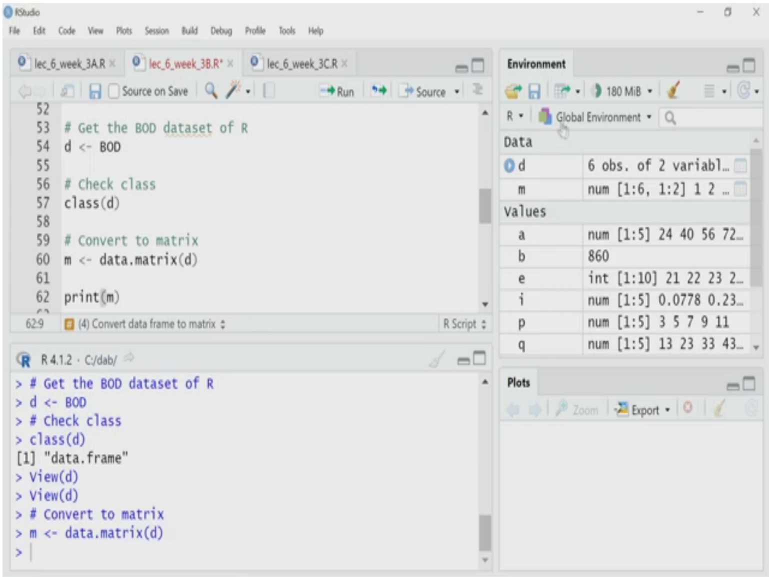
m ← data.matrix(d)

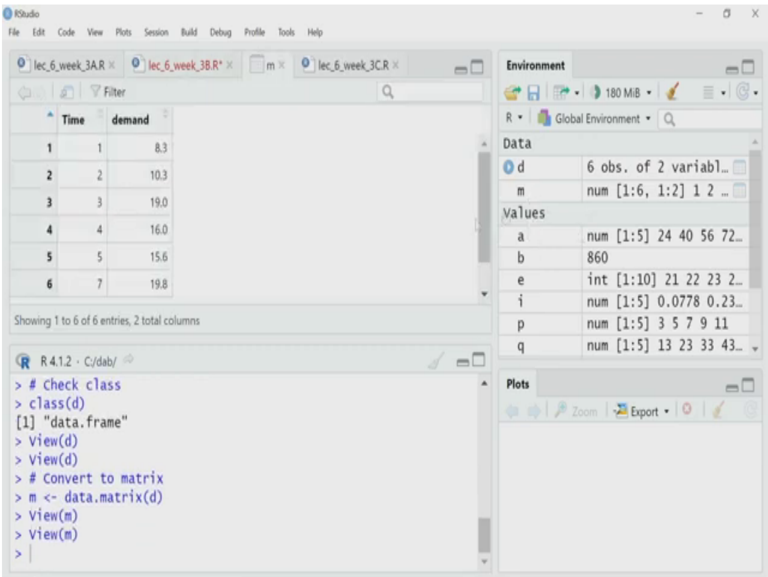
print(m)

So, for that what I will do, I use a function called data dot matrix. So, before doing that, I need some data in table format or a data frame format. So, what I will do I will use the BOD data set which is inbuilt in R. So, in your machine when you have installed R you have that, so I will call that data and assign it to a new variable d. So, you can see, let me check the class of these d what type of data is this, let me first read it, I have read the data.

Now, I want to check the class, the class it is saying d is a data frame. And you can double click here also to see here, it has time and demand, oxygen demand actually. So, you have this table. So, now I want to create a matrix out of this table. So, I want to convert these from data frame to matrix format.

(Refer Slide Time: 24:59)





d ← BOD

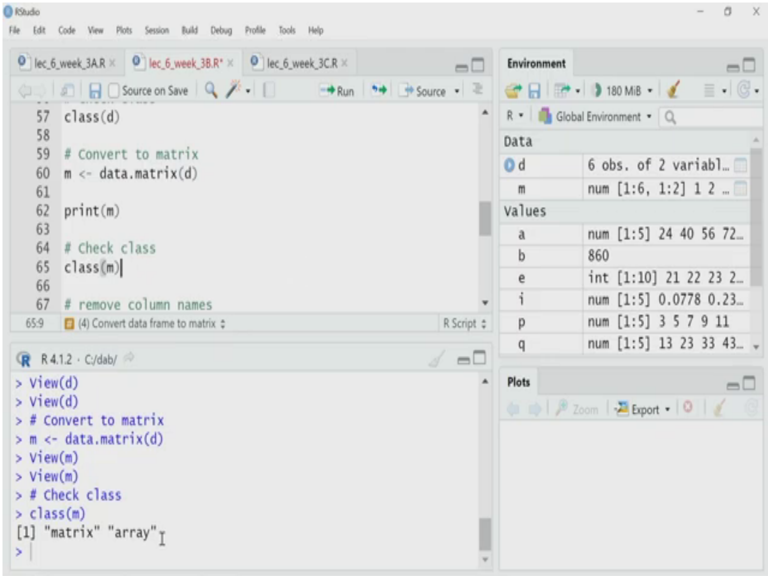
class(d)

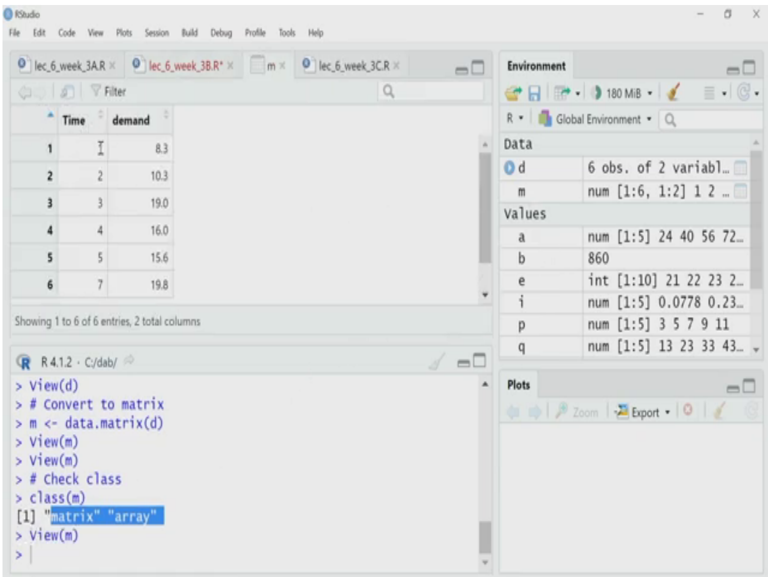
m ← data.matrix(d)

print(m)

So, what I will do, I will call these data dot matrix function, the argument should be this d, d is the data frame that I have. And the whole thing I will assign to a new matrix called m. So, if I execute that, and now if I check in the environment pane, you can easily see m has been created here. And let me double click it you can see I have time and demand as the header and I have the whole data as in a matrix format. But how do I know it is matrix, it looks like a table?

(Refer Slide Time: 25:35)

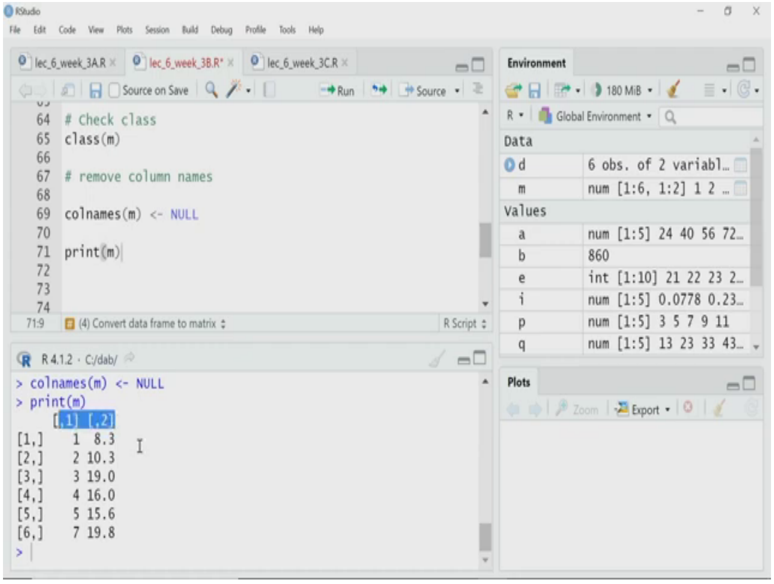


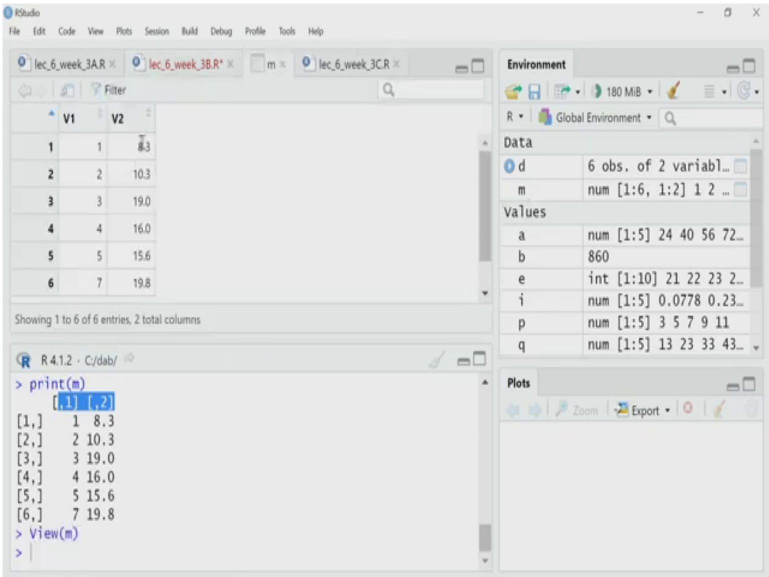


class(m)

So, I can check, I can go back and actually use that class option, class function to check the class of m now. So, let me check that and it is saying it is array is not a data frame. So, I have converted that data frame d, which is actually the BOD data into a matrix m. Now, this matrix has this header, column names. Suppose, you want to get rid of these column names. So, how can I do that?

(Refer Slide Time: 26:00)





columns(m) ← NULL

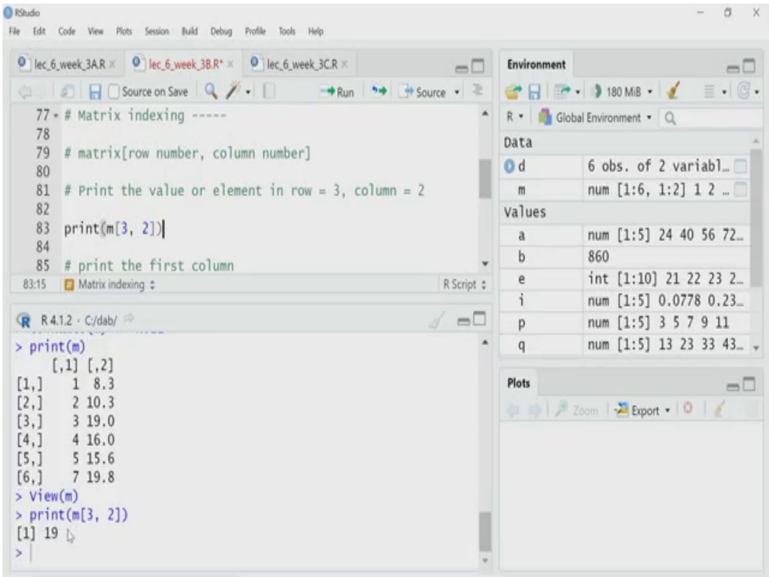
print(m)

I can do this way, very simple. Column names of m, this is what I am writing here. That is what I am telling to R, column names of m, m is the argument to this function m. And I am assigning NULL, NULL mean nothing, not zero, NULL mean nothing. I am assigning NULL to that. So, column names, column names of m, when I am writing m in the argument, column names function will extract the names of the header columns of that matrix.

And then it as I am assigning NULL to them, all of them will become NULL. So, let me execute that and I will show you the effect. So, here I have executed. Now, let me, I can print it. So, now, you see you do not have any header, only the column numbers are there or you can go to environment and you can see there is only v1, v2, because variable 1 variable 2 that is how it is saying, it does not have any columns, column names.

So, what I have done, I have learned till now, how to create matrices. Now, I should go for matrix operations, like addition, multiplication, but before that, I will go into what we have done for vector also, indexing issue. How to take a particular element of a matrix or a particular column or particular row of a matrix?

(Refer Slide Time: 27:25)



print(m[3, 2])

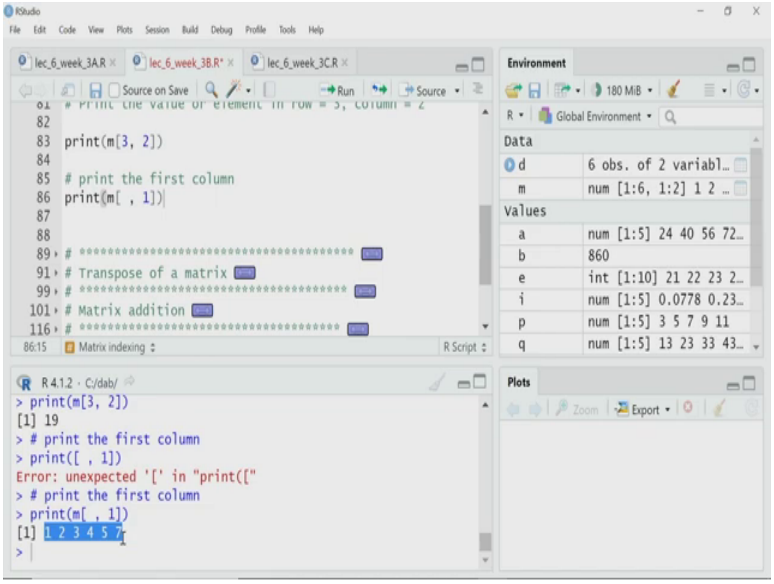
print(m[, 1])

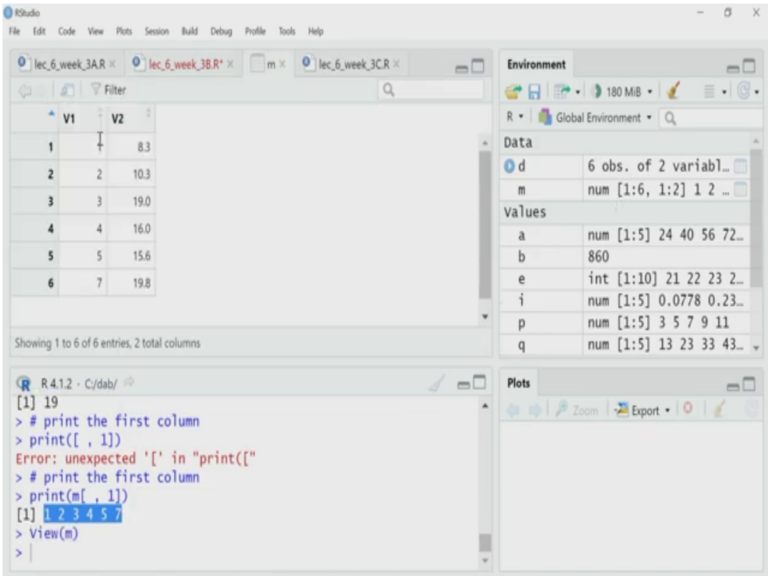
So, if you look into matrix indexing, the system is like this, you have should have a matrix name. And then in the square bracket, you should have column number and row numbers. The first number is row number. And the second number after comma is the column number. This is similar to vector in vector, we are using vector name in square bracket, I have the index of the element number position. Here, in these cases, a 2-dimensional matrix.

So, that means I have two things I would say, mentioned, I have to mention the row number as well as the column number. So, the first number in that square bracket should be the row number, then comma and then the column number. So, now suppose I have already created this m matrix from BOD data set, and I want to take the data or the element of row 3, column 2. So, let me check, row 3 is this one, and column 2, column 2 is this column.

So, row 3, column 2 should be 19. So, let us do it. I want to print it. So, what I am doing, in the print I am giving argument m square bracket 3, 2. That means 3 comma 2. That means 3 is the row number, 2 is the column number of whom, the m matrix and I want to print that. So, if I print. Yes, I get 19. Now, suppose I used to use the same indexing technique to not just get one value, but a whole column or whole row, I can also do that very easily.

(Refer Slide Time: 29:03)

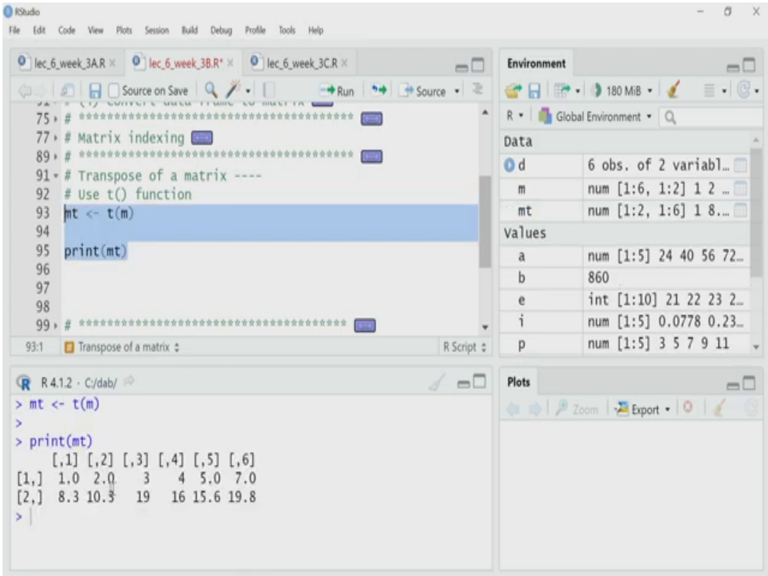


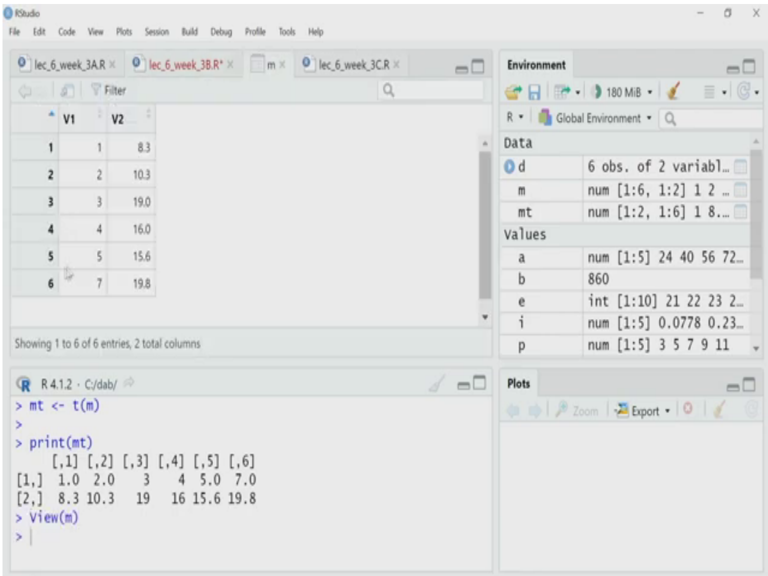


What I will do? So, suppose I want the first column, first column means I want these 1 2 3 4 5 up to 7. And I have to discard the second column. So, what I can say okay, I want the first column of m. So, m followed by the square bracket. And I do not put any more numbers because I know all, I want all the rows. So, I do not specify any row numbers. So, R will understand that I am asking it to give me all the rows data, and then after the comma, I am specifying the column number, the column number is 1.

And I want to print that. So, let me print it. Somewhere, I have missed the m while correcting, so it has to be m. So, print m in the square bracket, no row number and then the comma and the column number. Now, it should work. I have got the first column 1 2 3 4 5 6 7, let us check in m. First column is 1 2 3 4 5 6 7. So, indexing is done. Now, we will go into the next operation called transposing a matrix. If you remember, we can transpose. We can convert row to column, column to row, it can be done very easily in R.

(Refer Slide Time: 30:26)



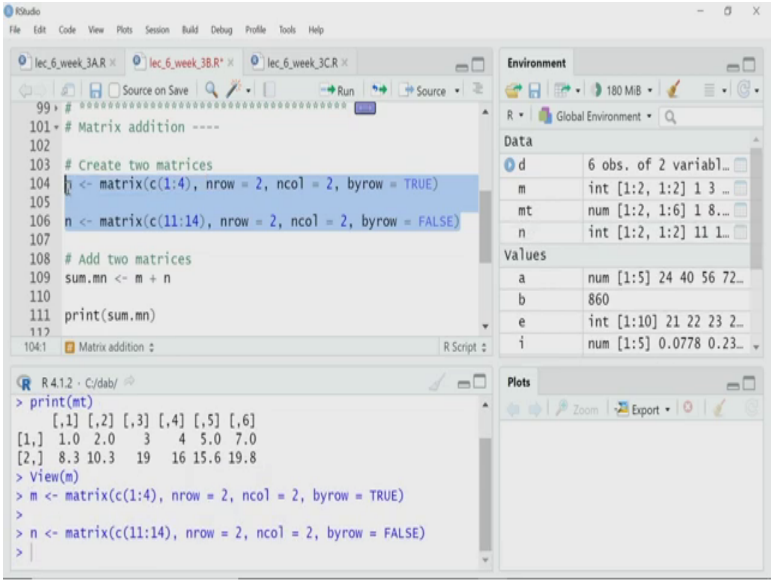


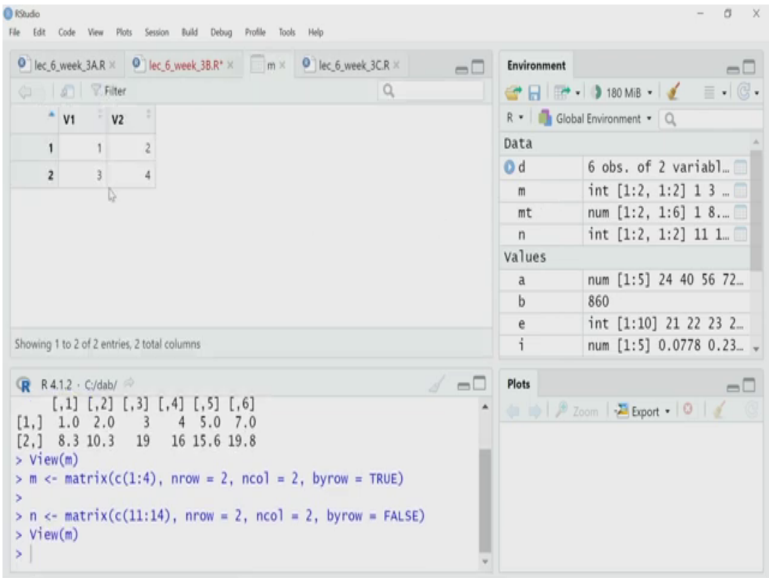
mt ← t(m)

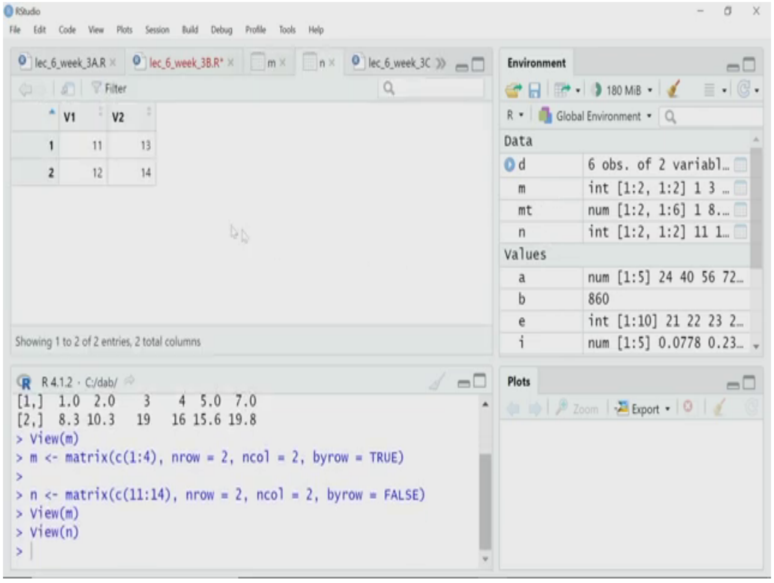
print(mt)

To do that, we have a function called t. And if you call that function and give the matrix as an argument, in this case, I will give m as an argument, I should be able to create the transpose of that. What I am doing in this line is that I am transposing m and creating and assigning that matrix to a new matrix called mt and I want to print it. So, let us do that. Now, you can see, the original m matrix is, how many rows we have, let us open, we have 6 rows and 2 columns. Now here, in case of mt, which is transpose, it has 2 rows and 6 columns. So, we have transposed m. Now, I will do matrix addition.

(Refer Slide Time: 31:14)







m ← matrix(c(1:4), nrow = 2, ncol = 2, byrow = TRUE)

n ← matrix(d(11:14), nrow = 2, ncol = 2, byrow = FALSE)

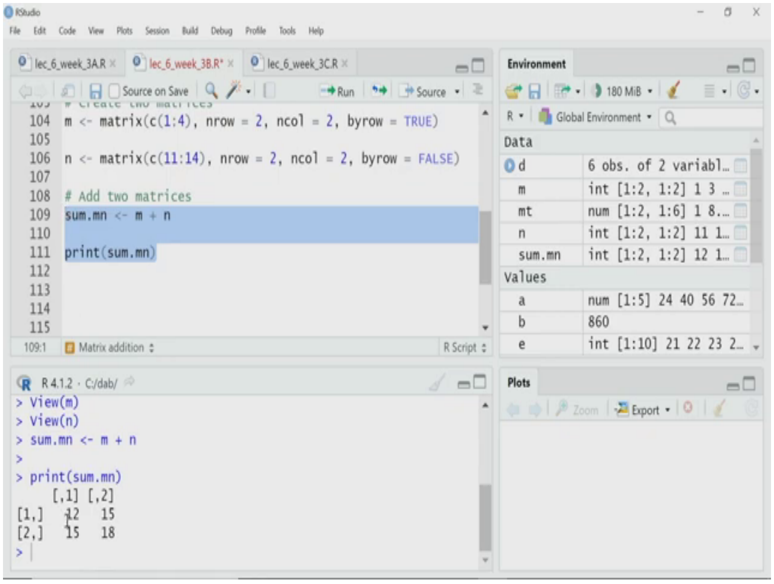
sum.mn← m + n

print(sum.mn)

Matrix addition, to do that, I will create two new matrices with a small r so that we can easily perform some other analysis easily. So, I am creating a matrix 2 by 2 matrix, the m. So, what I am doing it, I am using the matrix function, that means I have to give a vector. So, the first argument here I am creating a vector using c function, and that vector will have values from 1 to 4. And I want to divide this vector into 2 by 2 matrix, and I want to make it row wise.

Whereas here, I want to create a new matrix n for which I am giving a vector which starts from 11 and ends at 14 and it is also 2 by 2 and I want by row equal to FALSE, just like that I have given, you could, I could have taken something else as an example. So, let me create these two matrices. I have created them, you can check here m and n, 2 by 2 matrices.

(Refer Slide Time: 32:21)



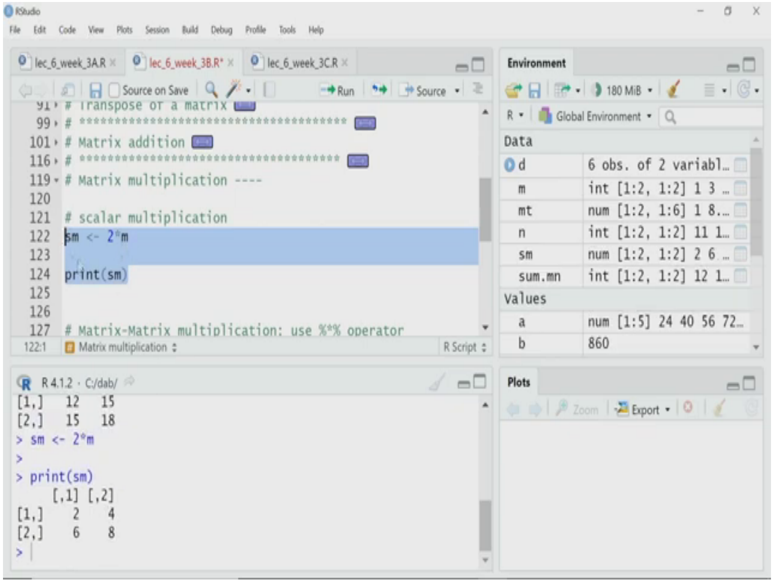
sum.mn← m + n

print(sum.mn)

Now, I will do all those vector operations on these two. So, I want to add, the first thing is that I want to add these two matrices, I will add m and n. So, just like addition of two vectors, I will use the plus operator and I should be able to add them. So, here what I am doing, I am adding m plus n, I am adding m and n matrices that those I made just now.

And I am assigning to a variable called sum dot mn. Obviously, that should also be a metrics. And I am printing it. So, let me do that, I have done it. So, I have added m and n. And I have got a new metric sum dot mn, which has these values 12 15 15 18. Now, as I have done the summation, obviously, you can easily do scalar addition to a matrix, I have not shown that, there is not an issue. Now, I will go and show you matrix multiplication.

(Refer Slide Time: 33:15)



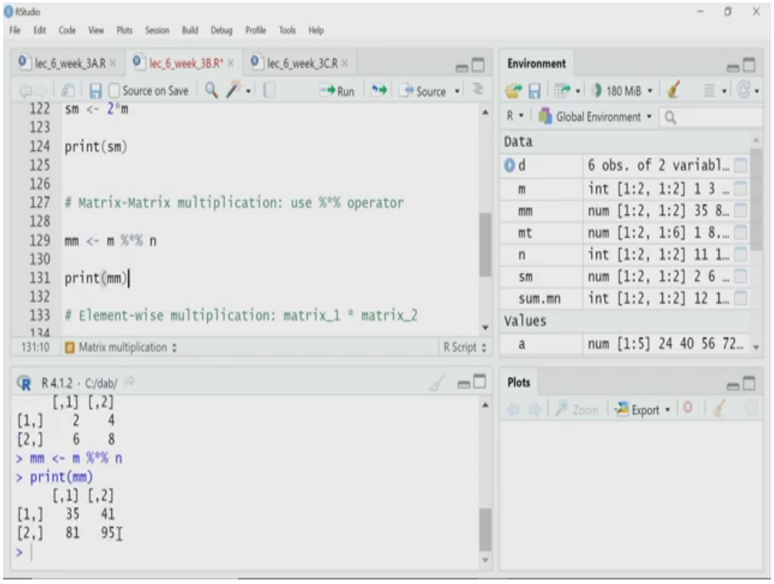
sm← 2*m

print(sm)

Just like vector multiplication matrix multiplication can be also of two types, at least two types. One is the scalar another one is a multiplication between two matrices that we have discussed in the lectures. So, first, I will multiply the m matrix that I created by 2, a scalar and then I will print it. So, I have multiplied it by 2 and then assign that to sm and I have printed sm here. And now I have got a new matrix 2 4 6 8.

Now, I want to do the complicated one, the matrix-matrix multiplication, two matrix has to multiply. And there is a rule to do that, you have to take the row and the column of the second one and do the multiplication. If you have forgotten, please go back and check the lecture on matrix multiplication to understand what I am saying, and then come back and look into this part of this video.

(Refer Slide Time: 34:07)



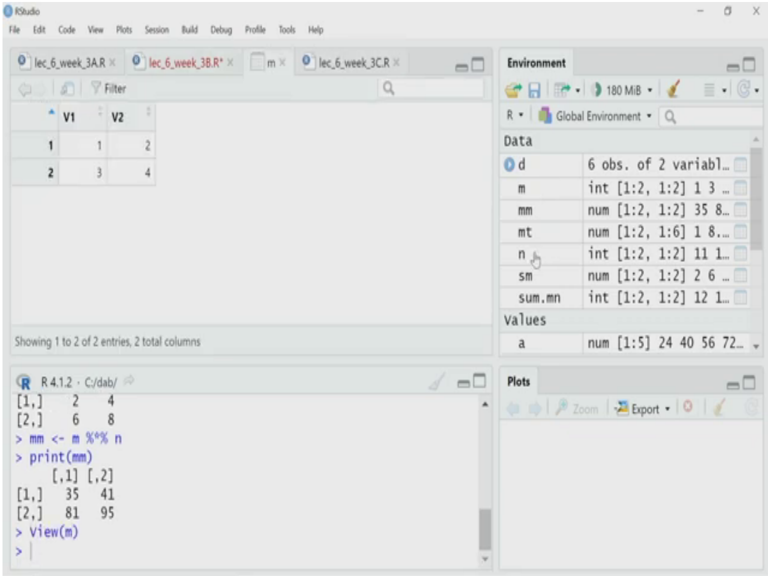
mn ← m%%n

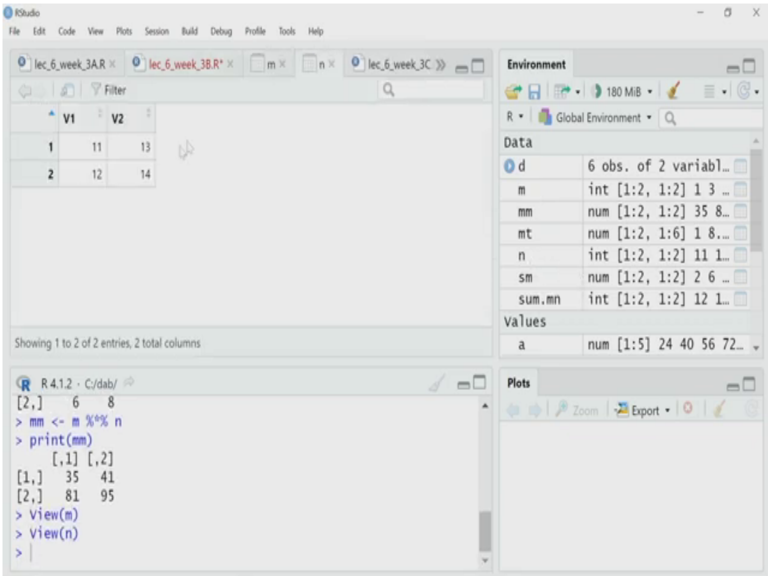
print(mn)

So, if you want to perform matrix-matrix multiplication, the way we have taught in that lecture, then you have to use this particular operator, you first put the percentage sign, then the star and then the percentage sign. So, percentage star percentage, this is the operator for matrix-matrix multiplication. And what I want to do here, I want to do matrix-matrix multiplication between m and n, m and n we had created, and I assign that to mn, a new matrix.

And that is why I am using here that particular operator percentage star percentage, data here I execute it, and then I print it. So, I got 35 41 81 95. You can go back and check the calculation using the m and n matrix you will find you will get the same matrix. So, by multiplying these two matrices, I have got another matrix. Now, what will happen if you use the star symbol which is the multiplication operator in R, you can do that, but that will give a different result. What it will perform? It will perform element by element multiplication. What do I mean by that?

(Refer Slide Time: 35:29)

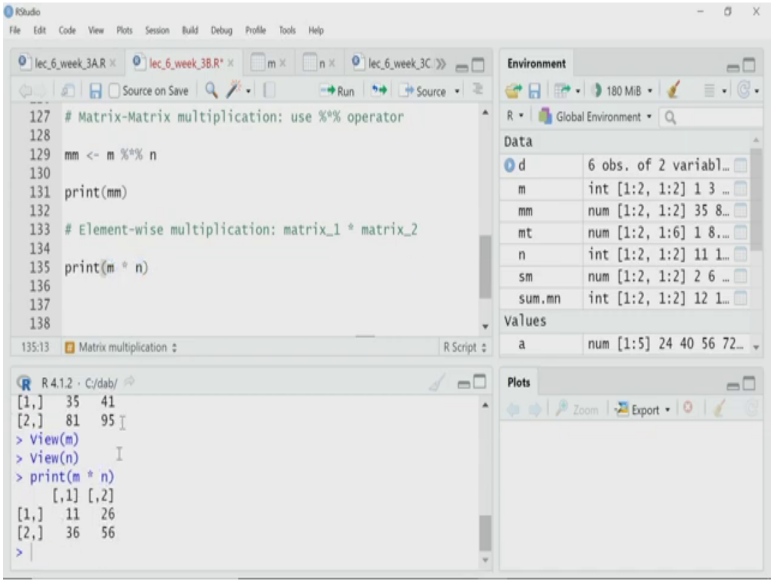




Let me open, m and n. This is m matrix; this is n matrix. So, if I use m star n then element by element multiplication will happen. That means, this 1 of m will get multiplied with 11 and that will remain in that position that value, 2 will get multiplied with 13. So, index numbers of these two matrices are matching, when I am taking the first row first column value and multiplying it with the first row first column value of the second matrix.

The second row second column value of the first matrix will be multiplied with second row second column value of the second matrix. So, this will eliminate wise I will multiply, I will show you how you, what will happen if I do multiplication like that.

(Refer Slide Time: 36:18)

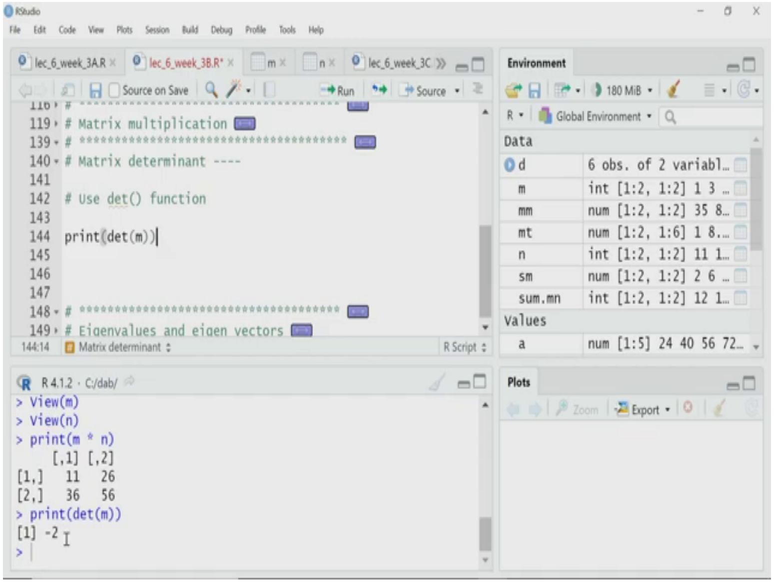


print(m * n)

So, here I am asking to print just using the star symbol, multiplication symbol. So, there should be element wise multiplication between m and n. Here, you can see 11 and this one is 26, it is this one is 11 because 1 into 11 is 11, then this one is 13. And the corresponding value here in the m is 2, 13 into 2 is 26.

So, if you want to do element wise multiplication, then only you use the star symbol, usually in the matrix multiplication requirement that we have in this data analysis course, we do not need that we require the original multiplication that we have learned in the linear algebra class. In that case, remember, you have to use these percentage star percentage operator to tell R that you want matrix-matrix multiplication, not element wise multiplication between the matrices.

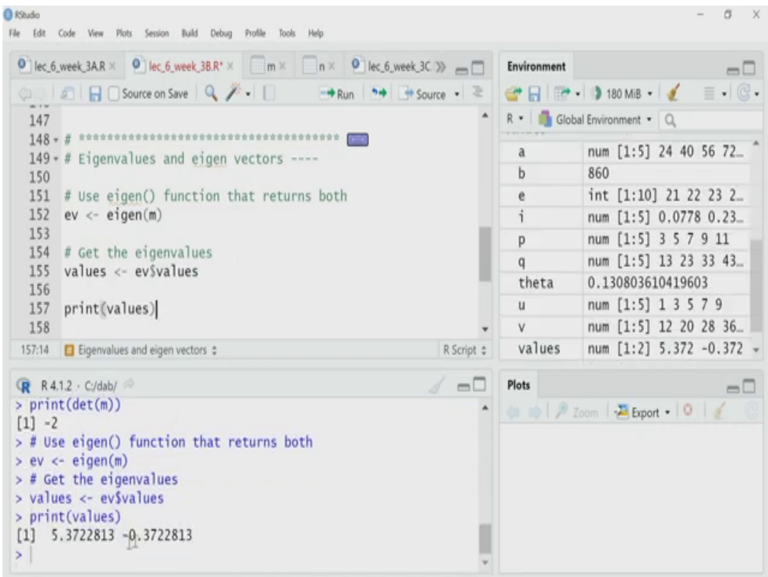
(Refer Slide Time: 37:18)



print(det(m))

So, we have learned the matrix multiplication. Now, I will go to determinant and eigen values. How can I calculate the determinant of a matrix? In R it is very easy, it has a determinant function, det function, which will take the matrix as an argument and it will output the determinant of that. So, I am printing it here in one line. So, I am saying print date of m and if m is the matrix we already know. So, if I do that, I get minus 2. So, that is the determinant of m matrix, that is the this one, this matrix, you can manually do and check that.

(Refer Slide Time: 37:57)



ev ← eigen(m)

```
values ← ev$$values

print(values)
```

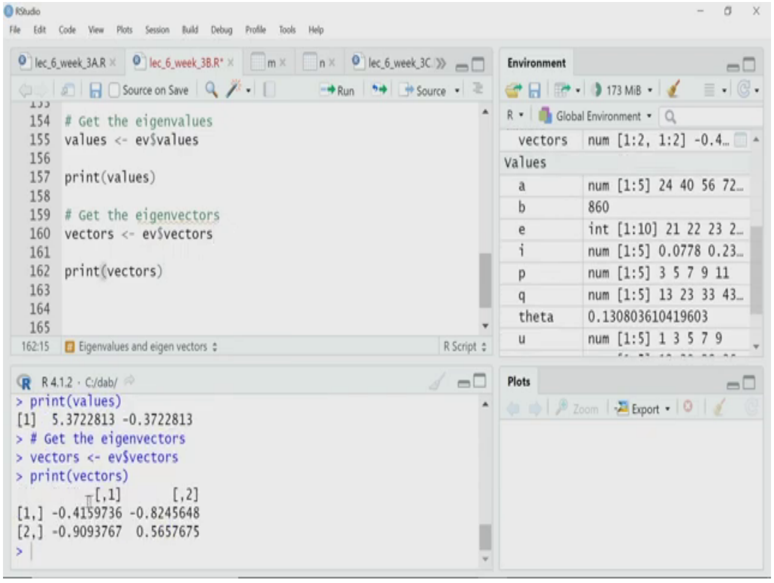
Now, I have done the determinant. Now, I will go to what we call eigen values and eigen vector, two very important quantities related to square matrices. So, luckily, R has an inbuilt function to calculate both eigen values and eigen vectors. That function is called eigen. It will take the matrix as an argument and it will return both eigen values and eigen vectors. Let me first ask it to calculate the eigen values and eigen vector and I will assign that whole data to this variable ev.

This ev will store, this variable will store both eigen values and eigen vector. Subsequently, I will separately collect the eigen vector and eigen value and print them. So, the first thing is I run the eigen function. So, it has calculated both the eigen values and eigen vectors for m and has assigned the whole data to ev. Now, I want to collect the eigen values only.

So, I am writing ev dollar sign eigen values. So, value is an object of the output comes from the eigen function, and that values object has all the eigen values in it. And I want to fetch that from ev. So, I am using this dollar symbol. So, I am saying give me the values that you are storing in value as values object in ev and I will assign those thing to values variables, those are the eigen values.

So, I do that and then I print them and the eigen values are 5.37 and minus 0.37 and so on. So, now, I can extract the eigen vectors in the same way. So, the output of an eigen function has an object called vectors. And I have stored the data in ev. So, I want ev to give me the vectors object.

(Refer Slide Time: 39:58)



values ← ev$$values

print(values)

vectors ← ev$vectors

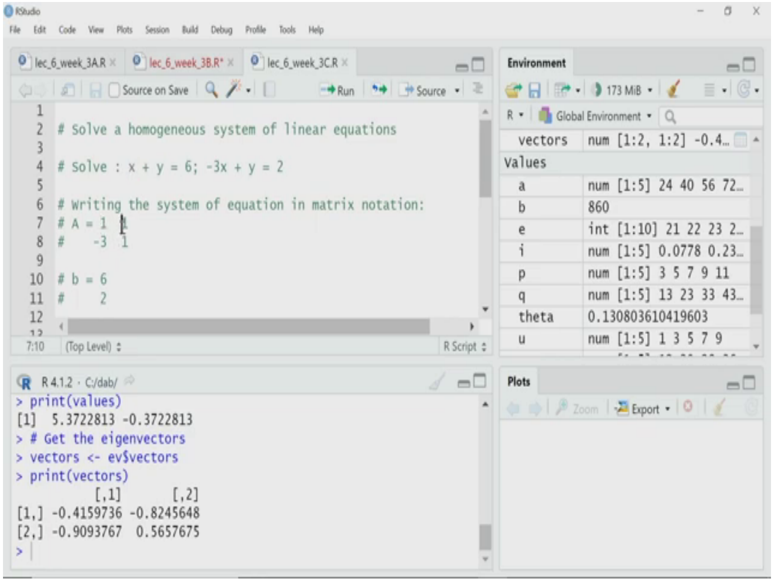
print(vectors)

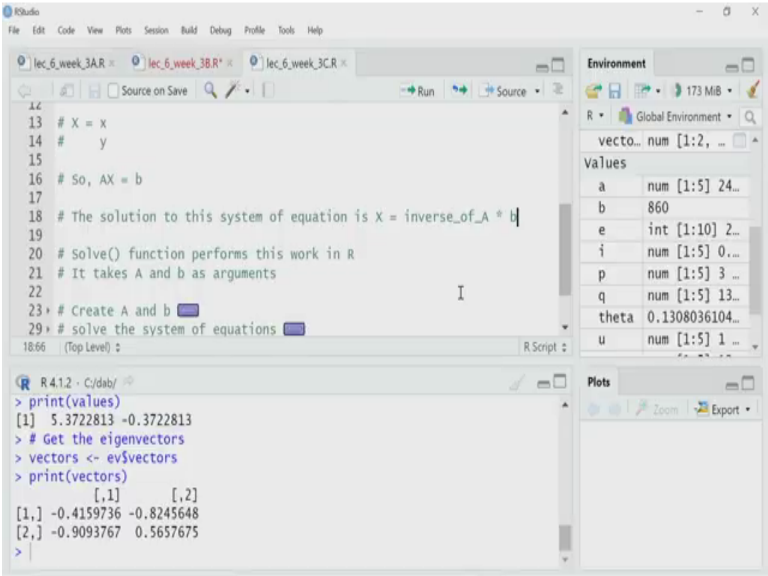
So, I am using ev, then the dollar sign and then the vectors. And if I run these and I will assign that to this new variable vectors, done that, I want to print the vectors. Now, you can see both of vectors. It is a two dimensional matrix, it is a two dimensional matrix, so it will have two eigen values two eigen vectors, check that it has two eigen values. So, correspondingly there should be two eigen vectors. The first eigen value is 5.37.

So, it is Victor is here, the first column minus 0.41, minus 0.9. So, this is the eigen vector corresponding to the first eigen value. And next one is the second eigen value here minus 0.37. So, the next column in the vectors is the second eigen vector corresponding to the second eigen value. That is all for matrices.

We have learned the how to create vectors how to create matrices and their operations, addition, multiplication, indexing, all these issues. One important thing that we have learned in linear algebra class in this course, is that, we can use linear algebra to solve a system of linear equations, I will end this lecture with that topic. So, let me recapitulate with how we solve a system of linear equation using linear algebra.

(Refer Slide Time: 41:21)





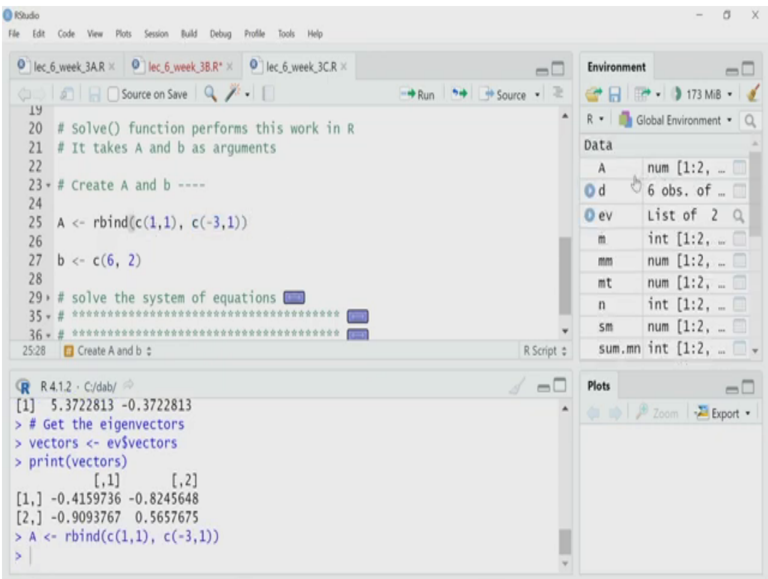
Suppose, you have told me that I have to solve these two equation x plus y equal to 6, minus 3x plus y equal to 2, I have to solve these two. That means, I have to get the value of x and y that we satisfy both this linear equation. So, how can I do that? I have to create some matrix and vectors. I have to create a coefficient matrix usually we call it A. So, in this case, the coefficient matrix will be 1, 1, minus 3, 1. 1, 1 will be the first row, minus 3, 1 will be the second row.

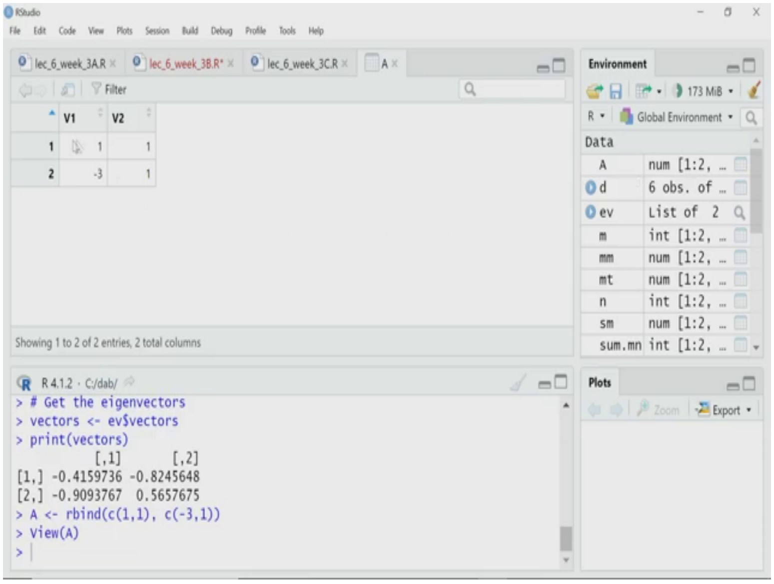
Why? Because I have 1 into x plus 1 into y in this first equation. So, I have 1, 1. In the second equation, I have minus 3 into x plus 1 into y. So, that is why I have minus 3 and 1. So, this is the coefficient matrix. Then everything on the right hand side of the equal to sign should be in another vector. So, I have 6 and 2. So, that will be vector b and I will have should have 6 and 2. And the solution should be another vector that is a capital X here and it should have the unknown x and y.

And I have to calculate this capital X vector. In other words, I have to calculate this small x and small y. How do we do that? We know this whole system of equation can be represented as A into X, A is the coefficient matrix, X is the unknown vector, and b is the right hand side vector. So, if I write A into X equal to b, then I can use linear algebra, and I can get a solution of these, if I do something like that the solution will be X, the vector, result vector will be equal to inverse of A, because I am taking A on the right hand side is equal to sign, inverse of A into b.

So, that means, I have to do inversion of A. Now, this is what you do to solve a system of linear equation by linear algebra. Now, I will not do manually inversion and all these things to solve this problem, I will call a solve function write a function which will solve this problem for me. So, R has that function, it is called solve. So, it should take two argument, A the coefficient matrix and the B vector and then it will spit out this x the solved values, solutions.

(Refer Slide Time: 43:49)



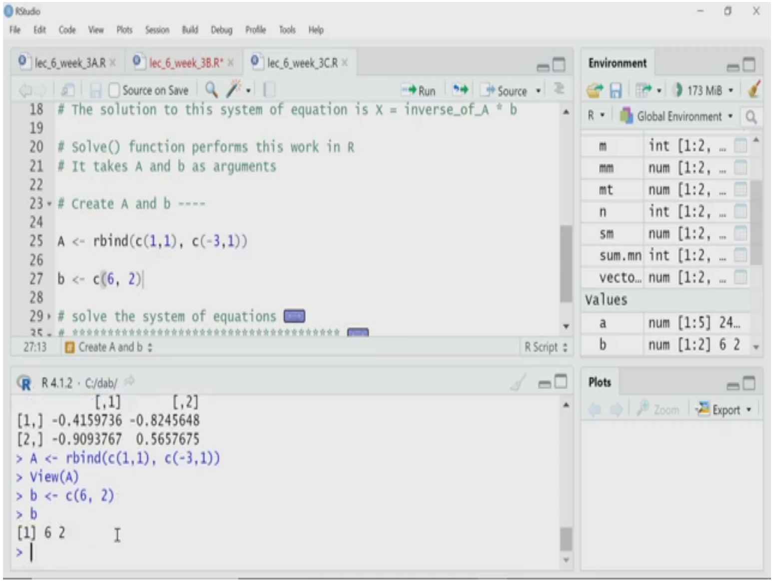


A← rbind(c(1, 1), c(-3, 1))

b← c(6, 2)

So, let me start with the creation of A and b, A is this matrix 1, 1, minus 3, 1. So, I am using rbind function. First, I am creating the vector 1, 1 here. And then I am creating the second vector minus 3, 1, and I am joining them, I am stacking them row wise, I am binding them row wise using rbind. So, I should get A, I have got A, I can double click and check. I have got 1, 1, minus 3, 1, I will use this A.

(Refer Slide Time: 44:21)

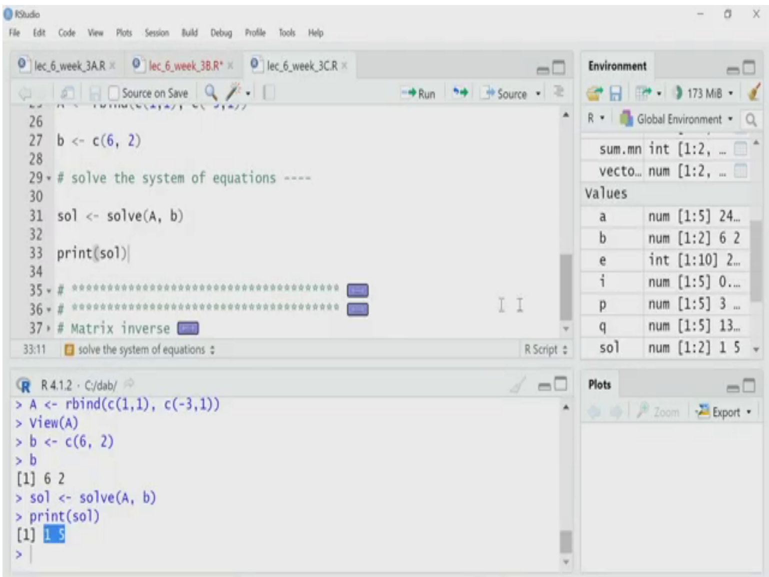


A← rbind(c(1, 1), c(-3, 1))

b← c(6, 2)

I want to keep the b vector, b vector should be 6, 2, as I have said here. So, that is what I am doing here, I am using c function to create b. So, b has been created. Let me print and see b, yes, b is 6, 2. So, now I have got a and b. So, now I can call solve function. So, I will do that.

(Refer Slide Time: 44:42)



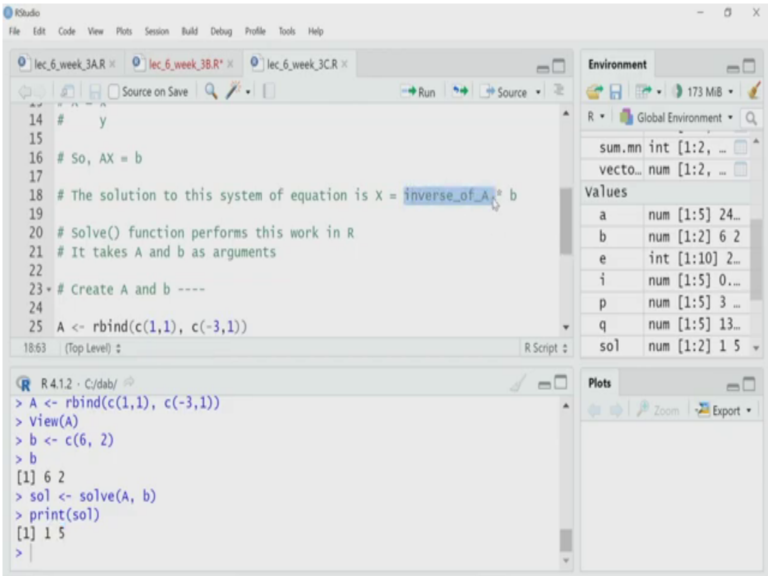
solv ← solve(A. b)

print(sol)

Here, I am calling solve function of r, I am giving A and b as argument. So, the solve function will solve and spit out the vector x, which has the solution for x and y. And I am assigning that to a new variable called solution, sol. I have done that. Now, I want to print a solution and the solution is 1 and 5. So, x equal to 1, y equal to 5, you can put these two value on those two equation, and you will see the solution will satisfy both this equation.

So, this is the way in one single line, I can solve a system of linear equations. Now, when I was discussing matrices, I had discussed determinant, transpose, eigen values, how to do those things in R. But in the lectures of linear algebra, we have also studied how to invert a matrix, inversion of a matrix.

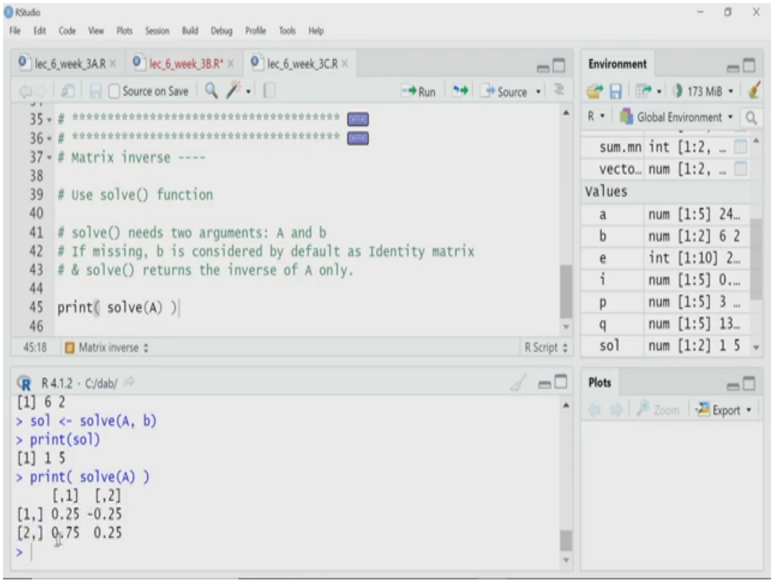
(Refer Slide Time: 45:36)



A← rbind(c(1, 1), c(-3, 1))

And as I said, here, to solve a linear equation, you have to do the inversion of a matrix. But today's lecture, I never discuss our inversion of a matrix, which we do very frequently in data analysis. So, how to do that? Actually, the solve function itself can do that. That is a good thing. So, how can I use the solve function to do that?

(Refer Slide Time: 45:58)



print(solve(A))

It is very easy, the solve function has been written in such a way that, see I have said I have to give two arguments, A is the coefficient matrix, b as the right hand side vector. Now, if you do not provide b, this function by default will consider b as the identity matrix. And if you remember that, if I multiply any matrix with an identity matrix, then I get the same matrix. So, in this case, it will multiply identity matrix with the inverse of A, so it will give me inverse of A.

So, if you want to get the inversion of a matrix, so what you simply do, you call the solve function, but only provide the matrix do not provide the b vector. So, that is what I am doing here. I am calling solve, only providing A not providing any b vector. So, it will only report me the inversion of A. So, if I run it and print it, this is the inversion of A, A inverse.

So, A inverse is 0.25, minus 0.25, and so on. That is all for this lecture. We have learned how to use R for all those things that we have learned in linear algebra course in this course, in the last week. Thank you for joining me today. Keep on learning. See you in the next video.