

**Foundations of R Software**  
**Prof. Shalabh**  
**Department of Mathematics and Statistics**  
**Indian Institute of Technology, Kanpur**

**Basics of Calculations**

**Lecture - 10**

**R as a Calculator with Scalars and Data Vectors: Power Operations, Integer and Modulo divisions**

Hello friends, welcome to the course Foundations of R Software. Now you can recall that in the last two lectures we had discussed different types of mathematical operations in the R software and we had considered the mathematical operations when we are trying to consider scalars, data vectors and a combination of them.

And we had essentially learnt that how to do addition, subtraction, multiplication and division with the data vectors and scalars. So, now in this a lecture we are going to move forward and we are going to consider some more operations. So, as we have done in the last two lectures similar type of thing we are going to do in this lecture also and we are going to understand how are does the power operation, modulo division, integer division.

Well, these things are important and you know that when you are trying to do the mathematical computations, these things are required in many situations. So, this is what we are going to do, we are exactly follow going to follow the same pattern what we have done in the last two lectures and we will try to understand that how are performs these operations in the software.

So, we start our lecture.

(Refer Slide Time: 01:39)

**R as a calculator**  $2^3$

```
> 2^3      # Command for power operator
[1] 8      # Output

> 2**3     # Command for power operator
[1] 8      # Output
```

R Console

```
> 2^3
[1] 8
> 2**3
[1] 8
```

$2^3 = 2 \times 2 \times 2$   
Power operation  
^ : Power operation  
\* : Multiplication

$2^3 = 2^3 \Leftrightarrow 2**3$

Now, you can see here that in case if you want to compute the expression like 2 cube means you are trying to multiply 2, 3 times, then how are you going to do it.

So, in the R software there are two notations to indicate such power operation. One here is hat, hat you can see this is one of the key on your keyboard and second option is two stars, you can recall that when you are trying to use only one star this is indicating the multiplication and when you are trying to indicate two stars this is going to indicate the power operation.

So, now suppose if you want to compute 2 cube. So, in R you have to write it like 2 hat 3 or other alternative is you can write down here 2 star star 3 like this. Now you can see here this is what example, I have taken here I want to compute here 2 cube and I have written this here as a 2 hat 3 and which is giving you an answer 8.

And the same thing I have written as 2 double star 3, which is giving you an answer here 8 and this is here the screenshot. So, you can see this is how you can do the power operation when you are trying to consider a scalar.

(Refer Slide Time: 02:55)

**R as a calculator**

```
> 2^0.5 # Command for power operator
[1] 1.414214 # Output

> 2**0.5 # Command for power operator
[1] 1.414214 # Output

> 2^-0.5 # Command for power operator
[1] 0.7071068 # Output
```

$\sqrt{2} = 2^{\frac{1}{2}} = 2^{0.5} = 2^{*0.5}$

$\frac{1}{\sqrt{2}} = 2^{-\frac{1}{2}} = 2^{-0.5} = 2^{*-0.5}$

The screenshot shows an R console window with the following commands and outputs:

```
R Console
> 2^0.5
[1] 1.414214
> 2**0.5
[1] 1.414214
> 2^-0.5
[1] 0.7071068
>
```

Handwritten mathematical derivations are shown to the right of the console output:

$$\sqrt{2} = 2^{\frac{1}{2}} = 2^{0.5} = 2^{*0.5}$$
$$\frac{1}{\sqrt{2}} = 2^{-\frac{1}{2}} = 2^{-0.5} = 2^{*-0.5}$$

Similarly, in case if you want to find out the square root of something say 2. So, you know this square root of 2 can be written as 2 raise to the power of 1 upon 2 or 2 raise to the power of here 0.5. So now, after this you can very easily write this quantity in R as 2

hat 0.5 or 2 double star 0.5 and this is what I have shown you here, that if you try to see this is the value of square root of 2 and this is giving you 1.414214.

And similarly the same operation you can do with the double star also this is also giving you the same value. And similarly if you want to find out 1 upon square root of 2, then this can be written as 2 raise to the power of minus 1 upon 2 which is 2 raise to the power minus 0.5. And then you can write down here this thing here, is say 2 hat minus 0.5 or 2 double star minus 0.5.

Yeah, do not get confused that this minus is going to be operated first because you are going to follow the rule of BODMAS, right. So, in case if you want to compute 1 upon square root of 2 then you can write it like this 2 hat minus 0.5 and this will give you the value 0.7071068. And this is here the screenshot of the same operation.

I hope it is not difficult and it is very easy for you to understand after understanding the earlier lectures.

(Refer Slide Time: 04:16)

**Power operator with scalar**

```
> c(2,3,5,7)^2 # command: application to a vector
[1] 4 9 25 49 # output
```

Handwritten annotations:  $2^2$ ,  $3^2$ ,  $5^2$ ,  $7^2$  are shown below the output, with arrows pointing from the corresponding numbers in the command. A diagram on the right shows 'Value' pointing to 'data vector' and 'Scalar' pointing to 'Scalar'.

R Console screenshot:

```
> c(2,3,5,7)^2
[1] 4 9 25 49
>
```

So, now these operations are related to when you are trying to do the power operation with respect to a scalar, right. Now I try to do the power operation when one of the value is a data vector and the power is a scalar. You know that for example, when you write 2 cube, so then this is one value this is another value suppose I call it say here value 1 and this here as a value 2.

Now you have different combination. Value 1 value 2 both are scalars, value 1 is a data vector and value 2 is a scalar or third option can be both value 1 and value 2 are data vectors. So, we will try to see all sorts of combination and you try to understand how R function.

So now, I try to take it here this data vector say 2, 3, 5, 7 and then it has here a power 2. So now, in case if you try to operate it the same rule operates here. What happened in the mathematical operation? That this hat 2 is going to be operated over all the elements. So, this will become like 2 square, 3 square, 5 square, and 7 square and this will give you the value here 4 9 25 and 49.

This is as simple as that, as you have done with addition, subtraction, multiplication, division. Just like that power operator will also be entering inside the data vector and it will make the operation on each and every element in the data vector. You can see here this is the screenshot of the same thing.

(Refer Slide Time: 05:42)

**Power operation with vector**

```
> c(2,3,5,7) ^ c(2,3) # !!ATTENTION! Observe the operation
[1] 4 27 25 343 # output
```

$2^2, 3^3, 5^2, 7^3$

R Console

```
> c(2,3,5,7) ^ c(2,3)
[1] 4 27 25 343
>
```

Now, after this I try to take a similar example. So, I try to take here two data vectors let us call it here say data vector 1 and another data vector here DV2. So, first data vector this has got a the number of elements which is four and the second data vector has number of elements two.

So, the number of elements in the second data vector is the multiple of the number of elements in the first data vector. So, what will happen here that if you try to recall the earlier operations the this will be your here position say here 1, 2, 3 and here 4. So, in the DV1 you have 4 values here 2, 3, 5 and 7 and say DV2 you have here 2 values 2 and 3.

So, first of all the operation will be like as here, hat and so for this block is going to be operated and you will get here 2 hat 2 and 3 hat 3 which is here 4 and here 27, a 2 square and 3 cube. Now after this when R moves further it does not find any value over here. So, what it does it will copy this value here 2 and 3 and then the values at the third and fourth positions will be operated with these 2 values and you will get here 5 square and say here 7 cube.

And this will give you here the value 25 and 343. So, you can see it was working in the multiplication, division, addition, subtraction the same way power operation is also being done in this case.

(Refer Slide Time: 07:05)

**Power operation with vector**

```
> c(1,2,3,4,5,6)^c(2,3,4) # command: application
# output
[1] 1 8 81 16 125 1296
```

$1^2, 2^3, 3^4, 4^2, 5^3, 6^4$

R Console

```
> c(1,2,3,4,5,6)^c(2,3,4)
[1] 1 8 81 16 125 1296
>
```

And now in case if you try to take care one more example where once again I am trying to increase the number of data values in the 2 data vectors, but I am trying to keep them as a multiple. You can see here now this 1, 2, 3, 4, 5, 6 there are 6 values and in the second data vector there are 2, 3, 4. So, now, you will see here this 2, 3, 4 will come over here and it will be like this here 2, 3 and here 4.

And then once here this 2, 3, 4 will come here and it will become here 2, 3 and here 4 like this, right. And then you will get here the value here 1 square which is here 2 cube which is here 3 to the power of 4 which is here, 4 square which is here 5 cube which is here and 6 power of a 4 which is here. And if you try to get this value in the R software they will come like this no issues and this is the screenshot of the same operation ok.

(Refer Slide Time: 07:57)

**Power operation with vector**

```
> c(2,3,5,7)^c(2,3,4)      #error message
[1] 4 27 625 49           # output
Warning message:
longer object length is not a multiple of
shorter object length in: c(2,3,5,7)^c(2,3,4)
```

$2^2, 3^3, 5^4, 7^2$

DV1: 2 3 5 7  
 DV2: 2 3 4

Diagram illustrating the power operation with vectors. DV1 (2, 3, 5, 7) and DV2 (2, 3, 4) are shown. The operation results in 2<sup>2</sup>, 3<sup>3</sup>, 5<sup>4</sup>, and 7<sup>2</sup>. The diagram shows the alignment of the vectors and the resulting values.

```
R Console
> c(2,3,5,7)^c(2,3,4)
[1] 4 27 625 49
Warning message:
In c(2, 3, 5, 7)^c(2, 3, 4) :
  longer object length is not a multiple of shorter object length
>
```

Now, I try to take one more example where I am trying to keep the length of the second data vector, which is not the exact multiple of the length of the first data vector.

So, this is my here DV1 which has here 4 values and this is my here DV2 which has only 3 values. So, the same thing will happen here also if I try to write down here DV1 and DV2 values, so DV1 is here 2, 3, 5, 7 and DV2 here is 2, 3, 4 and then its positions are here like as here 1, 2, 3 and here 4.

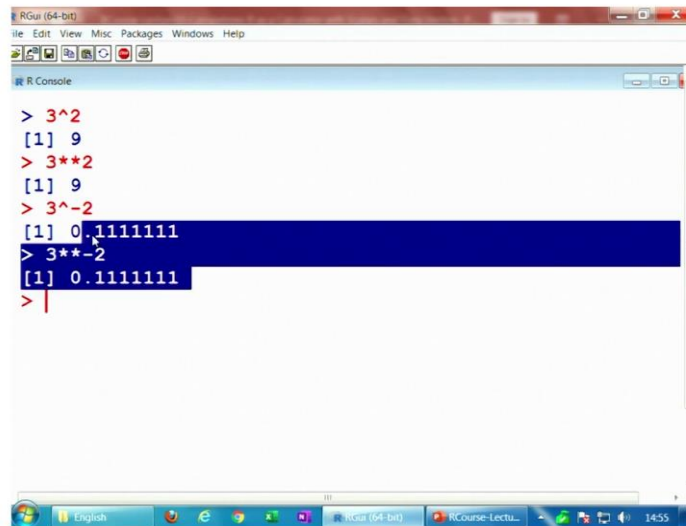
So, now, what will happen with this power operation that this block will be operated first and you will get here the values 2 square, 3 cube, 5 raise to the power of here 4. And after that R will get confused that there is no value here, so what it will do it will try to copy the same block here the same value will come here 2, 3 and here 4.

So, R will try to operate here with the first combination 7 and 2 and it will make it here 7 square, but after that what to do with 3 and 4 there are no values in the data vector 1 which R is getting to operate with the value 3 and 4 here. So, it will stop here, but it will

give you here a warning that the longer object length is not a multiple of the shorter object length in this vector and this is here the screenshot.

So, you can see here it is not a very difficult thing, means that is exactly in the same way the other operations were happening.

(Refer Slide Time: 09:22)



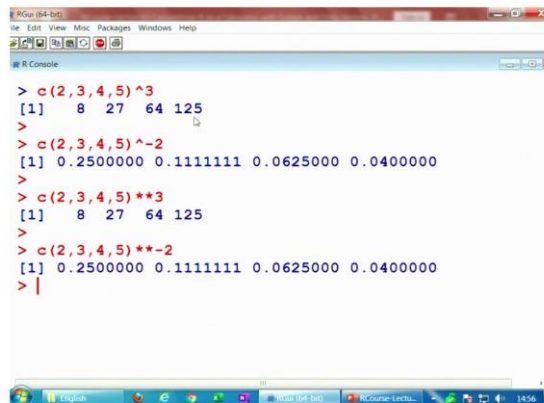
```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console

> 3^2
[1] 9
> 3**2
[1] 9
> 3^-2
[1] 0.1111111
> 3**^-2
[1] 0.1111111
> |
```

So, before I move further I try to give you this illustration in the R software. So, you can see here if you want to make it want to find out here 3 square that is 3 hat 2 and the same thing if you try to make it here with the two star symbol it will become here 3 double star 2 it is again 9.

Now, in case if you want to find out the square root of 3 you can express it like this 3 hat minus 2 this is 0.111 and so on. And if you try to find out here the same value using the double star notation, you can simply replace hat with double star and still you will find the same operations you can see here.

(Refer Slide Time: 09:59)



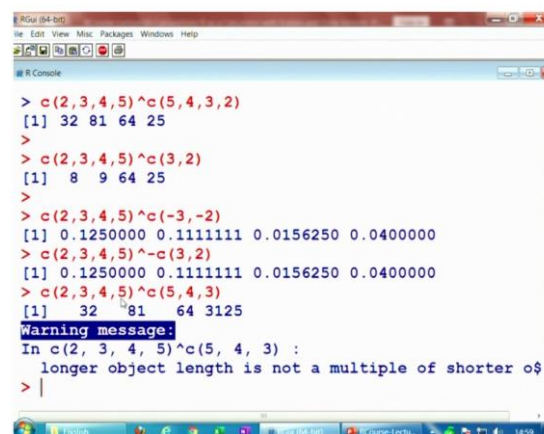
```
R Console
> c(2,3,4,5)^3
[1] 8 27 64 125
>
> c(2,3,4,5)^-2
[1] 0.2500000 0.1111111 0.0625000 0.0400000
>
> c(2,3,4,5)**3
[1] 8 27 64 125
>
> c(2,3,4,5)**-2
[1] 0.2500000 0.1111111 0.0625000 0.0400000
> |
```

Now, I try to take here one more example 2, 3, 4 and here 5 and I try to take it here say cube.

So, now this is hat three, so this is cube. So, this cube operation is going to be operated on each of this element inside this data vector having values 2, 3, 4 and 5. So, this will become here 2 cube, 3 cube, 4 cube, 5 cube and you can see here the values 2 cube is 8, 3 cube is 27, 4 cube is 64 and 5 cube is 125. And now, in case if I try to do here the same operation with here with some negative power also that will also work here yeah that you can see right there is no issue at all.

And the same thing if I try to replace it with the double star operation, you can see here it is giving you the same value right.

(Refer Slide Time: 10:44)



```
R Console
> c(2,3,4,5)^c(5,4,3,2)
[1] 32 81 64 25
>
> c(2,3,4,5)^c(3,2)
[1] 8 9 64 25
>
> c(2,3,4,5)^c(-3,-2)
[1] 0.1250000 0.1111111 0.0156250 0.0400000
> c(2,3,4,5)^-c(3,2)
[1] 0.1250000 0.1111111 0.0156250 0.0400000
> c(2,3,4,5)^c(5,4,3)
[1] 32 81 64 3125
Warning message:
In c(2, 3, 4, 5)^c(5, 4, 3) :
longer object length is not a multiple of shorter o$
> |
```



Now, I try to take an example where you are trying to work with the data vectors. So, if I try to take here one example to c 2, 3, 4, 5 and suppose just for the sake of illustration, I try to take here one more data vector of the same length. Suppose I can take it here 5, 4, 3 and here 2. So, this will become here what 2 raise to the power of 5, 3 raise to the power of here 4, 4 raise to the power of here 3 that is 4 cube and 5 square. So, now if you enter, this will give you this value 2 raise to the power of 5 is 2 into 2 is 4, 5 2s are 8, 8 2s are 16 and 16 2s are 32. Similarly 3 3s are 9, 9 3s are 27 and 27 3s are 81.

Then 4 into 4 into 4 is 64 and 5 square is 25. Now in case if I try to change this value and if I try to remove the two values, so that the second data vector has only two values so, what will happen here this 3 and 2 will be operated on the first 2 values. So, this will become 2 cube and 3 square and then once again 3 and 2 will be operated in the remaining 2 values and this will become here 4 cube and 5 square and you can verify these values without any problem.

And now in case if I try to show you here one operation that if I try to make it here -3 and -2 what will happen, and instead of having making here -3 and -2 if I make it here what will happen. You can see here both are the same values that that is correct. That because once you are trying to do here for example, -3 and -2 as to the power on all these numbers and here you are just trying to take this minus sign common and inside the bracket there are only 3 and 2.

But on the other hand in case if you try to make here the second vector of the length which is not the exact multiple of the first data vectors length, this will be here like this. 5 4 3 they will be operated over 2 3 4, so this will become 2 raise to the power of 5, 3 raise to the power of here 4 and 4 cube and then 5 raise to the power of here 5 which is 3125. But there will be a warning message also.

So, that is how it is going to work. So, now, you can see here that is power operations in the R console there also not very difficult to operate and they can be done without any problem.

(Refer Slide Time: 12:51)

**Integer division with scalar**

**Integer Division:** Division in which the fractional part (remainder) is discarded

Operator: `%%` `/%`

```
> 2 %% 2  
[1] 1
```

```
> 5 %% 2  
[1] 2
```

```
> 7 %% 3  
[1] 2
```

```
R Console  
> 2 %/% 2  
[1] 1  
> 5 %/% 2  
[1] 2  
> 7 %/% 3  
[1] 2  
> |
```

So, now after this we try to take here one more operation that is integer division.

What is this integer division? For example, if you try to divide well I will use the primary class arithmetic, if you try to divide 2 by 2 this will become here 2 1s are 2 and remainder here is 0. And similarly if you try to divide here 5 by 2, so I can write down here 2 2s are 4 and the remainder here will be here 1. And similarly, if I try to divide 7 by 3, so I can write down here 3 2s are 6 and the remainder here is 1.

So, that is how I was taught the arithmetic in my childhood, so that is the same thing. So, now, you can see here when you are trying to divide any value with something then you have here 2 answer one is here and another here is remainder. So now, when you are talking about the integer division in this integer division, the fractional part which is the remainder this is discarded and only this value is going to be here.

So, similarly if you try to see this division here is 2 and here 2. So, that is how we actually work in many many programming sometimes you want to do such operations. So, and these are very common operations in mathematics. So, in order to do the integer division the command here is like this one which is percentage sign on your keyboard then this backslash and then percentage sign.

So, if you try to operate here like this here 2 integer division that is 2 percentage backslash percentage 2 then it is giving you the same value and this value 1 is given

here. And similarly if you try to do here with 5 integer division by 2 like here this; so, this will go here like this and this value is given here, right.

And then similarly if you try to divide here 7 by 3 using integer division like this then this value here is given here like this and this 2 value here is obtained here like this. So, this is the concept of integer division in R software and here you can see the screenshot of the same operation.

So, this is what you have to understand.

(Refer Slide Time: 14:51)

**Integer division with scalar**

```
> c(2, 3, 5, 7) %% 2
[1] 1 1 2 3
```

Handwritten annotations show the calculation of remainders for each element:  $2 \div 2 = 1$ ,  $3 \div 2 = 1$ ,  $5 \div 2 = 2$ , and  $7 \div 2 = 3$ . Below the R console output, the individual operations are listed:  $2\%/\%2$ ,  $3\%/\%2$ ,  $5\%/\%2$ , and  $7\%/\%2$ .

```
R Console
> c(2, 3, 5, 7) %% 2
[1] 1 1 2 3
```

And similarly if you try to do this integer division when you try to choose 1 as a data vector and another as a scalar, the same thing will happen what has happened in the earlier operation; just like addition, subtraction, division, multiplication that this operator this will operate on each of this value 2, 3, 5 and 7.

So, this will be like as this 2 integer division by 2, 3 integer division by 2, 5 integer division by 2 and 7 integer division by 2. And you can see here 2 integer division by 2 will give you answer 1 and 3 integer division by 2 will also give you answer 1, 5 integer division 2 will be like this 2 2s are 4. So, remainder here is 1.

So, this value comes over here is 2 and when you are trying to divide 7 by 2, 2 3s are 6 remainder here is 1, so this 3 will come here. So, you can see here this is the same way as

it was happening in other operation the same process will be followed here also in the case of integer division.

(Refer Slide Time: 15:50)

**Integer division with data vector**

```
> c(2,3,5,7) %/% c(2,3)
[1] 1 1 2 2
```

2%/%2, 3%/%3, 5%/%2, 7%/%3

```
R Console
> c(2,3,5,7) %/% c(2,3)
[1] 1 1 2 2
```

10

And yeah in case if you try to take a two data vectors, so if you try to take the data vectors of the same length then each of the value in the respective position in the two data vectors will have the integer division operation. That is now very obvious for you, so just for the sake of example I have taken here two data vectors, one is of length 4 having 4 values and another is of length 2 which has 2 values 2 and 3.

So now, this integer division will be operated over this set of values and then this set of values. So, in case if you try to do here like this, 2 integer division 2, and 3 integer division 3 right. So, this value will be say here 1 and 1. And now in case if you try to have integer division over this 5 and 7. So, it will come here, so 5 integer division by 2 and 7 integer division by 3.

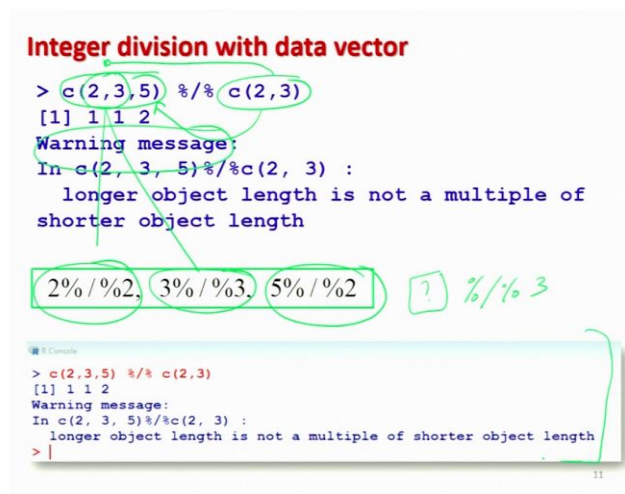
So, in both the cases the values are going to be 2 2s are 4 and 3 3s are 6. So, these 2 values are here and this is the screenshot of the same operation, right.

(Refer Slide Time: 16:50)

**Integer division with data vector**

```
> c(2,3,5) %/% c(2,3)
[1] 1 1 2
Warning message:
In c(2, 3, 5) %/% c(2, 3) :
  longer object length is not a multiple of
  shorter object length
```

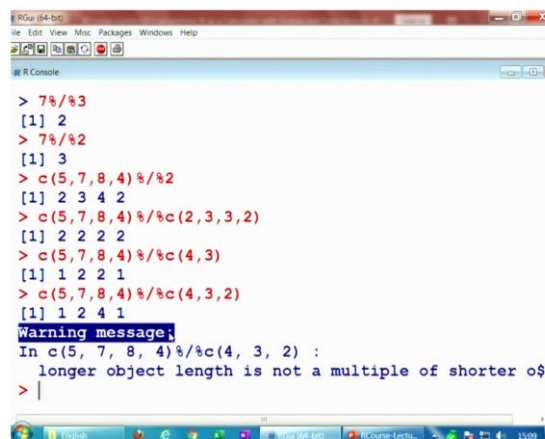
2%/%2, 3%/%3, 5%/%2    ? %/%3



And similarly if you try to take here a similar example where the first data vector has a length 3 and second data vector has a length 2, which is not the exact multiple of the length of the first data vector. So, then the same thing will happen these 2 values c 2, 3 will be operating over the first 2 values in the data vector 2 and 3.

So, this will become here like this 2 integer division by 2, 3 integer division by 3 and then this 2, 3 will come here once again over the 5. So, 5 will have the integer division by 2, but there is no other value with which something can be divided here using integer division. So, this will give you here a warning message you can see here this is the screenshot of the same operation.

(Refer Slide Time: 17:39)



```
> 7%/%3
[1] 2
> 7%/%2
[1] 3
> c(5,7,8,4) %/% 2
[1] 2 3 4 2
> c(5,7,8,4) %/% c(2,3,3,2)
[1] 2 2 2 2
> c(5,7,8,4) %/% c(4,3)
[1] 1 2 2 1
> c(5,7,8,4) %/% c(4,3,2)
[1] 1 2 4 1
Warning message:
In c(5, 7, 8, 4) %/% c(4, 3, 2) :
  longer object length is not a multiple of shorter object length
> |
```

So, let me try to show you these operations over the R console. So, that you get here more confidence. So, you can see here if I try to see here 7 integer division say 3 this will be here 2 and if you try to take it by 2. So, then 2 3s are 6 the answer will come out to be a 3.

And in case if you try to take here value here 5, 7, 8 and say here 4 and if you try to have the integer division by some scalar. So, this you can see here like this. So, this will be like here 2 2s are 4 2 3s are 6, 4, 7, 2 4s are 8 and 2 2s are 4, so this is the value here and similarly if you try to use here some data vector also.

So, if I try to say here 2, 3, 3 and here 2 like this data vector. So, what will happen the corresponding positions in the data vector 1 will have the integer division with respect to respective positions of the data vector 2. So, if you try to see here this will be 5 integer division by 2 this is 2, 2 2s are 4 7 integer division by 3 which is here 2, 8 integer division by 3 which is 3 2s are 6 this 2 and 4 integer division by 2 which is here 2, 2 2s are 4.

And similarly if I try to make it here like this say 4 and here 3 and if I try to remove the other 2 values, such that, the length of the second data vector is an exact multiple of the first data vector this will be here 1 2 2 1 how this 5 integer division by 4, 1, 7 integer division by this 3 this is 3 2s are 6 and then 8 integer division by 4 which is 4 2s are 8 and 4 inter division by 3 which is here 3 1s are 3 this is 1.

And in case if you try to say here 4, 3 and 2 here then you will get here like this 4, 3, 2 they are going to be integer division for the 3 number 5, 7 and 8 and for the 4 there is only one value here 4. So, it will give here 4 1s are 4, but the for the remaining 2 value 3 and 2 there are no numbers here in this data vector where the integer division can be are formed.

So, it will give you here a warning. So, you can see here it is not a very difficult thing which you cannot do and it is very easy to understand after learning all these topics.

(Refer Slide Time: 19:45)

### Modulo Division ( $x \bmod y$ ) with scalars

**Modulo Division:** modulo operation finds the remainder after division of one number by another.

Operator: `%%`     $\% \%$

```
> 2 %% 2
[1] 0
```

```
> 3 %% 2
[1] 1
```

```
> 7 %% 3
[1] 1
```

```
> 7 %% 4
[1] 3
```

```
R Console
> 2 %% 2
[1] 0
> 3 %% 2
[1] 1
> 7 %% 3
[1] 1
> 7 %% 4
[1] 3
> |
```

So, after this I try to give you an idea about one more topic which is the last topic of this lecture this is about modulo division. Yeah a popular name is like  $x \bmod y$  this is how we call it. So, what is this modulo division? If you remember we have just done this exercise that I try to divide 2 by 2. So, you will hear this will be here 2 1s are 2 and the remainder here is 0.

And similarly, if you try to divide 3 by 2, this will be 2 1s are 2 and the remainder here is 1. And then similarly, if you try to divide 7 by 3, this will be here 3 2s are 6 and the remainder here will be 1. And similarly, if you try to divide 7 by say here 4 then will be 4 1s are 4 and the remainder here will be 3. So, this modulo division, this gives us the remainder after division of one number by another.

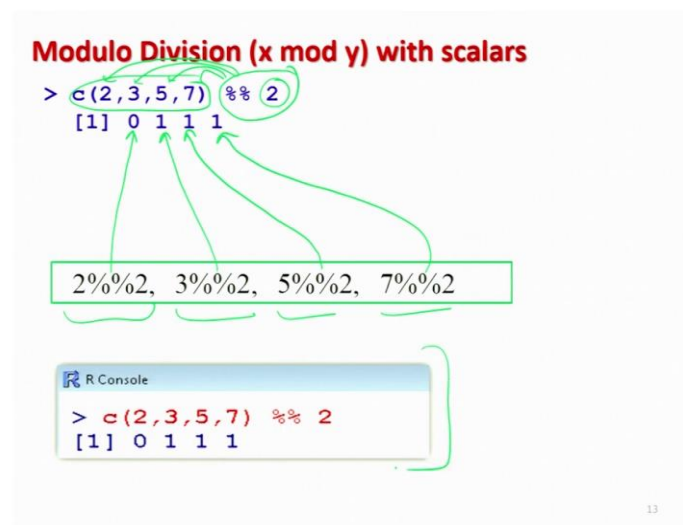
What does this mean? That when we are trying to divide 2 by 2 then the remainder is 0 when we are trying to divide 3 by 2, then the remainder is 1 and when we try to divide 7 by 3, then the remainder here is 1 and when we try to divide 7 by 4 then the remainder here 3 integer value.

And this modulo division will provide this remainder values. So, for example, the symbol for operating the modulo division in R software is like this one two signs of percentage like this. So, if you try to have the 2 mod 2 which is the 2 modulo division by 2, then its answer is going to be here 0 from here.

And similarly, in case if you try to have  $3 \bmod 2$  which is 3 modulo division by 2 which is here like this and its remainder here is 1 which is the answer here. Then the next operation is 7 modulo division by 3,  $7 \bmod 3$  and this is done here and its remainder here is one which is here like this. And then the last one 7 modulo division by 4 that is  $7 \bmod 4$ , which is here done here and its remainder here is 3 which is here like this 3.

So, this is the modulo division it will give you the remainder. So, now, this I have illustrated when I am trying to take the scalars only.

(Refer Slide Time: 21:48)



Now, in case if you try to take the first data vector as like this `c(2, 3, 5, 7)` and second value is the scalar. So now, in this case if you try to operate the modulo division what will happen, now this modulo division is going to be operated on each of this element 2, 3, 5 and 7 like this one 2, 3, 5 and 7.

So obviously, when you try to divide 2 by 2 then the remainder is going to be 0, when you try to divide 3 by 2 the remainder is going to be 1, when you try to divide 5 by 2 the remainder is going to be 1 and when you try to divide 7 by 2 the remainder is going to be 1 and this is here the screenshot.

So, you can see here this modulo division is working exactly on the same line as the other operations.



(Refer Slide Time: 22:31)

**Modulo Division (x mod y) with data vectors**

```
> c(2,3,5,7) %% c(2,3)
[1] 0 0 1 1
```

$2 \% 2$ ,  $3 \% 3$ ,  $5 \% 2$ ,  $7 \% 3$

```
R Console
> c(2,3,5,7) %% c(2,3)
[1] 0 0 1 1
```

14

Now in case if I try to take both the parts as data vector. So, the first data vector here is c 2, 3, 5, 7 and the second data vector is c 2, 3. I can also take the second data vector of the same length, but now you know that how it is going to be operated. There is no issue in understanding.

So now, what will happen to this that this c 2, 3 will come over here and this will be operated for the modulo division like this. 2 will be operated over 2 and 3 will be operated over 3 and this will give you an answer 2 divided by 2 remainder is 0, 3 divided by 3 remainder is 0.

Now, once again this 2 and 3 will come over here and they will be operated over 5 and 7. So now, this will be your here 5 modulo division 2 the remainder will be here 1 and then 7 modulo division 3 and the remainder will be here 1. So, once again you will get here the values like the 0 0 1 1 and you can see here this is here the screenshot.

So, you can see here that in this case also the modulo division works exactly in the same way as other operations.

(Refer Slide Time: 23:32)

**Modulo Division (x mod y) with data vectors**

```
> c(2,3,5) %% c(2,3)
[1] 0 0 1
Warning message:
In c(2, 3, 5) %% c(2, 3) :
longer object length is not a multiple of
shorter object length
```

2%%2, 3%%3, 5%%2

```
@ Console
> c(2,3,5) %% c(2,3)
[1] 0 0 1
Warning message:
In c(2, 3, 5) %% c(2, 3) :
longer object length is not a multiple of shorter object length
>
```

15

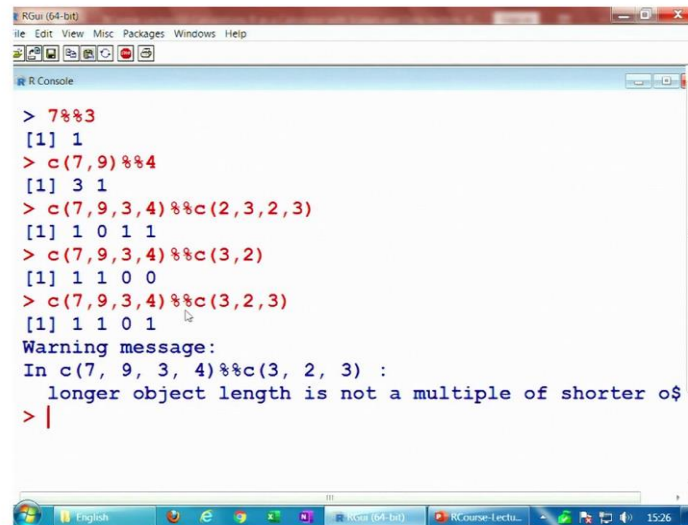
Now, I try to take the last example where I try to take the two data vectors for modulo division, but I try to take the first data vector which has got three elements and the second data vector which has got two elements.

So, the number of elements in the second data vector this is not the exact multiple length of the first data vector. For example, if you try to see in this case the length of the first data vector was 4 and the length of second data vector was two. So, the length of the second data vector was the exact multiple of the length of the first data vector.

So now, in this case the similar operation will happen that this c 2, 3 will come over here and it will try to operate over this c 2, 3. So, this will become here 2 modulo division by 2 and 3 modulo division by 3 and after that 1 value is left. So, this c 2, 3 will come here again and it will try to do this modulo division 5 irrespective 2. So, this will be here like this, but after that where to operate with this 3 in the first data vector it is not given it is not known to R.

So, it will not operate anything but it will stop and it will give you here a warning message that the longer object length is not a multiple of shorter object length. So, this is how this operation will be done. So, this is here the screenshot of the same operation and I try to show you these operations first on the R console and then you will get more confidence.

(Refer Slide Time: 24:50)



```
RGui (64-bit)
file Edit View Misc Packages Windows Help

R Console

> 7%%3
[1] 1
> c(7,9)%%4
[1] 3 1
> c(7,9,3,4)%%c(2,3,2,3)
[1] 1 0 1 1
> c(7,9,3,4)%%c(3,2)
[1] 1 1 0 0
> c(7,9,3,4)%%c(3,2,3)
[1] 1 1 0 1
Warning message:
In c(7, 9, 3, 4)%%c(3, 2, 3) :
  longer object length is not a multiple of shorter o$
> |
```

So, if I try to say here 7 mod 3. So, you can see here 3 2s are 6 and the remainder is here 1. Similarly, if you try to take here the first value here as say 7 and say 9 and then the second value is going to be here suppose here 4. So, you can see here now, if you try to divide 7 by 4 the remainder is going to be 3 and if you divide 9 by 4 the remainder is going to be 1.

Now if you try to take the second data vector also as a vector and for that we try to increase the length of the first data vector say here 7, 9, 3, 4 and say other data vector I try to say here 2, 3 and say here 2 and here 3. So, you can see here now in this case this 2 and 3 they are going to be operated over this 7 and 9 and this 2 and 3 they are going to be operated over this 3 and 4 and you get here and answer 1 0 1 1.

Similarly, if you try to change here the length of the data vector and suppose if I try to take it here only 3 and 2. So, what will happen here? That the 3 and 2 will be operated first over 7 and 9 and the 3 and 2 will be operated over 3 and 4. So, if you try to see here this modulo division will give you an answer 1 1 0 0. Why? Because if you try to divide 7 divided by 3, 3 2s are 6 remainder is 1, 9 divided by 2, 2 4s are 8 remainder is 1, 3 divided by 3 remainder is 0, 4 divided by 2 answer is 2 remainder is 0. So, that is how the things are going to work and if you try to take care the number of element which are not exact multiple length you can see here like this that this 3, 2, 3 is going to be operated with 7, 9, 3.

So, 7 divided by 3, the remainder is 1, 9 divided by 2, remainder is 1, 3 divided by this 3, remainder is 0 and 4 divided by 3 this remainder here is 1. But it will give you warning message, because there is no place where 2 and 3 can be operated over here, right. So, that is the way we are going to operate on it. So now, you can see here with this modulo of division also you have learnt one more topic that how these different types of power operation integer division modulo division they are operated in the R software.

So, now as you can see now that as we are moving forward more operations are coming and you need to remember more things. But my advice to you all is that whatever we are doing now today; please try to combine it with the earlier lectures also. And try to see how you can do better. There are some more aspects for these calculations, which we have to learn, but after the lecture today once again you have a good amount of homework to do.

Please try to take this these operations, try to take some values yourself. And try to do those calculations manually using your own hand, own pen, own paper and try to verify it with the outcome of the R software that will give you more confidence that is the way R is thinking you are also thinking. So now, whatever you want to get it done through the R you can write an appropriate syntax command function for those things.

So, you try to practice it and I will see you in the next lecture till then goodbye.