Foundations of R Software

Prof. Shalabh
Department of Mathematics and Statistics
Indian Institute of Technology, Kanpur

Basics of Calculations
Lecture - 11
Built in Functions and Assignments

Hello friend welcome to the course Foundations of R Software and you can recall that in the last couple of lectures. We had talked about the different type of mathematical operations which R does and we have understood how R does it. And you have seen that there is a special way in which the R is doing these type of mathematical computations and calculations. So, now continuing on this line today we are going to talk about some more aspect of this computation.

In R as we have discussed many time that there is a functionality of built in function. What is this mean built in function? It is something like suppose if you want to write a program to find out the sum or the mean. Then you have two options it is like that you try to write down programming structure like a sum equal to 0, num equal to 0, sum is equal to sum plus num etc. that we have done in your possibly in your childhood I would say now. So, that is first option.

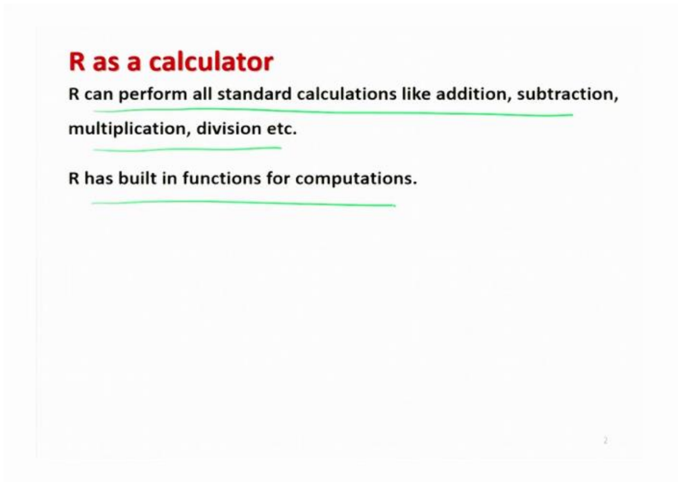
Second option is that somebody already has written a program for finding the sum or the mean. So, now what you have to do? You simply have to input the data and you have to use that program suppose the name of the program is sum. So, you have to simply write s u m sum and within the parenthesis you have to give the input data. And then as soon as you execute it this function is going to find out the value of sum.

So, the difference is that in this case you do not have to do any programming. But you can use these programs directly and in this R that is the beauty that you have an option to write your own programs and you can also use the built in program. So, in the lecture today we are going to discuss about some popular built in function there are there is a long list of functions that are available in the R software.

But surely you will understand that discussing all those is very difficult in a lecture. So, I will try to take here some popular representative functions and my objective is not really to tell you about the functions. But the main objective is, how are you going to use them.

And after that I will try to show you that how you can use these built in functions along with the simple computation that you have learnt in the last couple of lectures. So, we begin our lecture and try to understand what it does.

(Refer Slide Time: 03:01)



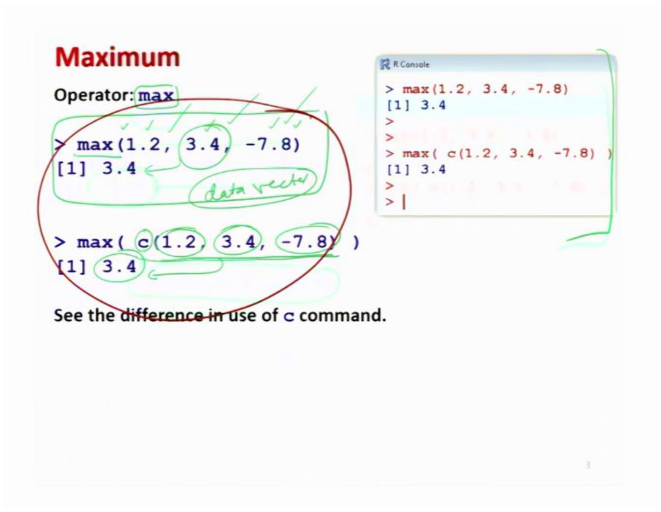
R as a calculator

R can perform all standard calculations like addition, subtraction, multiplication, division etc.

R has built in functions for computations.

So, as we have learnt now that R can perform all type of standard calculations like addition, subtraction, multiplication, division, power operation, modulo, division, integer, division etc. and R also has some built in functions for computation.

(Refer Slide Time: 03:19)



Maximum

Operator: max

```
> max(1.2, 3.4, -7.8)
[1] 3.4
```

data vector

```
> max( c(1.2, 3.4, -7.8) )
[1] 3.4
```

See the difference in use of c command.

```
> max(1.2, 3.4, -7.8)
[1] 3.4
>
>
> max( c(1.2, 3.4, -7.8) )
[1] 3.4
>
> |
```

So, what are those things? So, let me try to take here couple of examples and try to explain you. Suppose you want to find out the maximum value among some given values for that your first option is that you try to write down a program yourself or the alternative is that you can use a built in program and this built in program have been programs have been given a name.

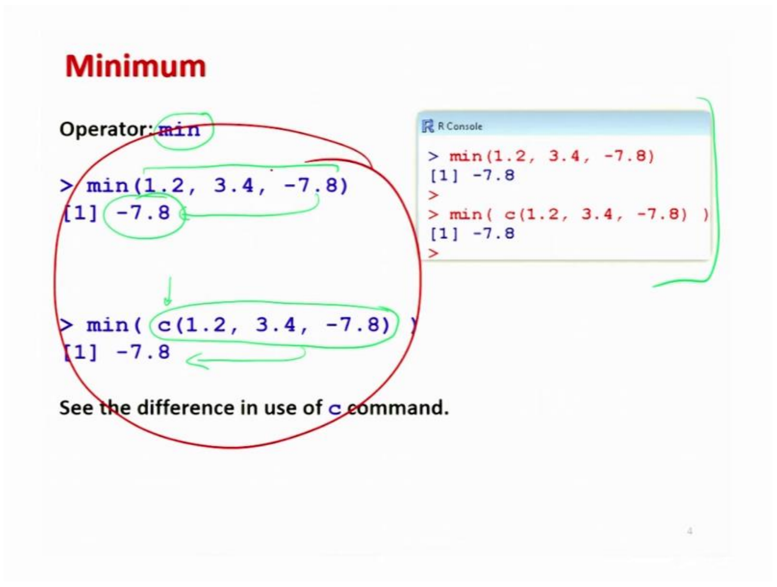
For example, if you want to find out the maximum of some numbers. So, the name of the program here is m a x. So, you can call this program by this name and the job of this program is to provide you the maximum value among the given numbers. So, you can see here I try to write down here max and within the parenthesis I try to write down the numbers among them I want to find out the maximum value.

So, you can see here I have taken three numbers; 1.2, 3.4, -7.8. So, you know the maximum value out of these 3 number is 3.4. So, the output will come out to be here 3.4. Now, I try to address you one more issue as you have learnt in the earlier lectures that whenever you are trying to give more than one numerical values in the R software as input.

Then you always use the concept of data vector and in order to define a data vector you wrote all the numbers inside a parenthesis and you use the command lowercase c. So, now, in this case if you try to see in the first option here you have not used the aspect of data vector to input the values you have simply given the values here 1.2, 3.4, 7, -7.8 without using the command c and now you are trying to use here the command c.

So, you can see here what is the difference there is no difference still if you try to see the input is the same 1.2, 3.4 and -7.8 and their output is the same as 3.4. So, now, in this case you can see here that in case if you are using the command c or you are using the concept of data vector or you are not using the option of data vector the outcomes are going to be the same. But now I will try to show you here some more example and then I will try to address what should we do, right.

(Refer Slide Time: 06:14)



Minimum

Operator: min

```
> min(1.2, 3.4, -7.8)
[1] -7.8


> min( c(1.2, 3.4, -7.8) )
[1] -7.8
```

See the difference in use of c command.

```
> min(1.2, 3.4, -7.8)
[1] -7.8
>
> min( c(1.2, 3.4, -7.8) )
[1] -7.8
>
```

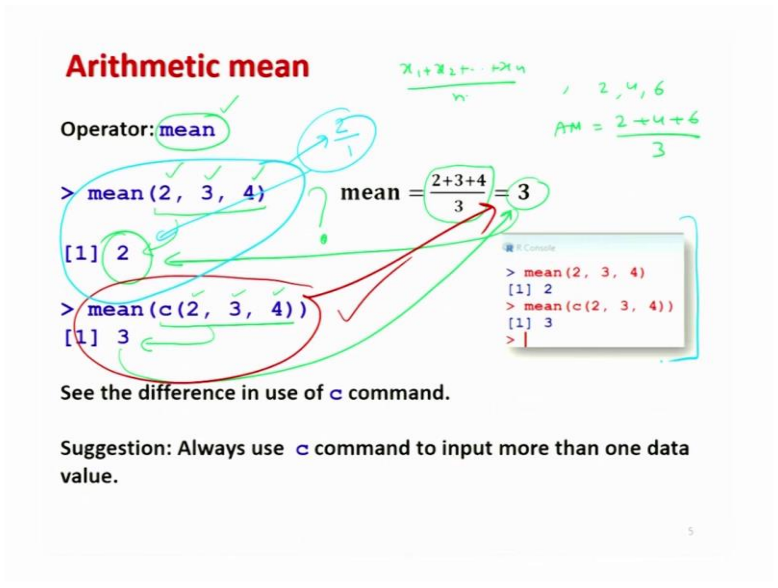
So, in the beginning I am saying again that I am going to take here sum commands where I will be demonstrating the application by using the data vector concept and without using the concept of data vector. So, similarly, if you want to use here the command m i n.

Then this m i n is used to find out the minimum value among the given values. So, once again exactly on the same way you can see here that if you are trying to write down here 3 numbers 1.2, 3.4 -7.8. Then the minimum value among all of them is the -7.8.

And in this case you are not using the concept of data vector you are simply giving the values inside the parenthesis. Now if you try to do the same thing, but you try to use here the concept of data vector and you input your data using the command c then once again the answer comes out to be the same -7.8, right.

So, in both the cases in maximum and minimum you can see here that either you are trying to use the concept of data vector or not in giving the input values the output is the same and it is not affected by the use of c command. And you can see here this is the outcome for the minimum and this was here the output for the maximum.

(Refer Slide Time: 07:40)



Arithmetic mean

Operator: `mean`

```
> mean(2, 3, 4)
[1] 2
```

$\text{mean} = \frac{2+3+4}{3} = 3$

```
> mean(c(2, 3, 4))
[1] 3
```

See the difference in use of c command.

Suggestion: Always use c command to input more than one data value.

```
> mean(2, 3, 4)
[1] 2
> mean(c(2, 3, 4))
[1] 3
>
```

But now I try to show you here something else and now you have to be very careful what I am trying to show you here. Similarly, in case if you want to find out the arithmetic mean then the command here is mean m e a n you know that arithmetic mean is defined that if you have some values x 1, x 2, x n. So, you try to define the arithmetic mean by their sum divided by the number of observations.

For example, if you have suppose some number 2, 4, 6 and if you want to find out the arithmetic mean of these numbers this will be 2 + 4 + 6 divided by 3, right. So, in R software there is a command here mean all in lower case and if you want to find out the mean of suppose 3 numbers 2, 3 and 4.

So, now, I first try to use my earlier command to give the input data using the simple concept. That means, without using the concept of data vector without using the c command and then I will try to do the same exercise after giving the input using the concept of data vector. So, let us try to see what happens.

When I try to operate here mean of 2, 3, 4 it gives us the answer 2, but in case if you try to find out this arithmetic mean this is going to be 2 + 3 + 4 divided by 3 which is equal to here 3. So, this mean here is coming out to be 3, but then R is reporting you here 2.

So, there is a big question mark what is happening? You should get here the answer 3 and that is what I always have suggested you in the past that whenever you are trying to do any calculation. First you try to do your manual calculation and then you try to cross check and verify with the output of the R software.

So, now I change my approach and I use here the data vector approach and I give the input data using the command c and I give here the command as c 2, 3, 4. And now if you try to see, this mean is coming out to be 3 which is matching with the arithmetic mean that you have found manually.

Now in case if you try to see here in these commands, if you try to see here in this maximum there was no difference whether you are using the command c or not, same was true with the minimum command also. But now in the case of mean this is changing and when you are using the command c here, only then it is trying to give you the correct answer.

So, now what should we do and why this is happening? So, one of the reason that why this is happening, I do not know, but that is my guess that when this R started then different people across the world were participating in the development and it is possible or it might be possible that people developed. actually these different functions were developed by different people. So, some people might have used the command c to execute the program and some people might have considered the function to get the input data without using the c command. So, some people used the c command to input the data and some people used the simple; that means, inputting the data without using the c command and gradually all these functions were incorporated in the R software.

So, that is why some functions gives you correct answer with c and some function give you correct answer with or without c, but ok that is ok that I understand that is happening, but what should I do as a user. Because if you try to think, you are writing a program in which you are trying to find out the value of the mean and you have not used the c command what will happen.

You will be inputting some data value they will be coming to the mean function and mean function will not be giving you the correct value. But the program is long you will not be able to understand and or it will be very difficult for you to find whether the final outcome is correct or not.

So, that can be one problem which may happen, but anyway our objective is to find out a solution there are many problems in life, but more important part is to find out the solutions. So, one solution is very clear that when you are trying to use the c command then you are getting the right answer correct answer in all the three cases - minimum, maximum as well as mean.

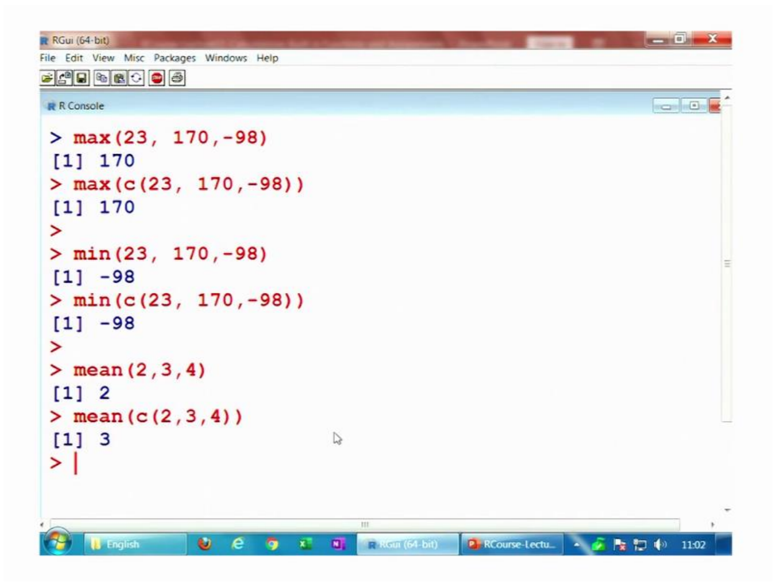
So, why cannot we make here a rule that whenever we are trying to input the data, we are always going to use the c command. At least this will ensure that the output is going to be correct and this will guarantee that there is no mistake at least in giving the data as an input for any program.

So, that is my very sincere advice to you all that when you whenever you input the data, use the c command, use the data vector concept, ok. So, let us now begin our lecture and we continue with our lecture and we try to now I will be using always the data vector command to give the input of my values that is clear to me. And I hope that is clear to you also and it was a convincing argument also, ok.

If I ask you a very simple question that you should find out what is happening in this first case where you are trying to find out the mean of 2, 3, 4 and it is giving you value the 2. Actually in this case it is simply trying to read the first value and after that it is not reading the remaining values 3 and 4.

So, that is why it is giving you 2 divided by the total number of values which is here 1 and so it is giving you here the value 2 that is what, that is happening. So, you have to be now careful, ok and then you can see here this is the screenshot. So, now before moving further let me try to show you these operations on the R console also, right.

(Refer Slide Time: 14:01)



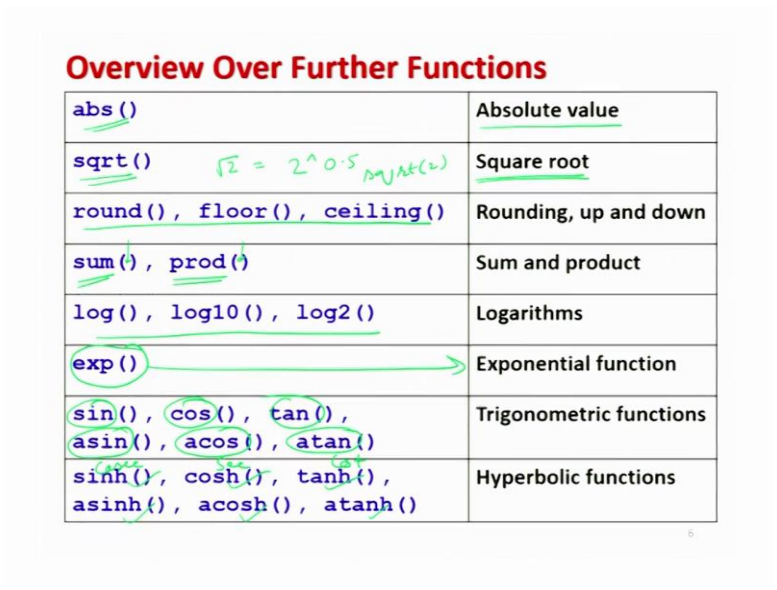
```
> max(23, 170,-98)
[1] 170
> max(c(23, 170,-98))
[1] 170
>
> min(23, 170,-98)
[1] -98
> min(c(23, 170,-98))
[1] -98
>
> mean(2,3,4)
[1] 2
> mean(c(2,3,4))
[1] 3
> |
```

So, you can see here if I try to use the command here maximum. So, let me give the data without using the c commands. We can see here 23, 170, -98 etc. and you can see here this is giving you the value 170. But in case if you try to use here the c command also it will give you the same value and in case if you try to use the minimum command.

So, let me try to use here the same value. So, you can see here the minimum of 23, 170 and minus 98 is -98. So, you can see here the answer is -98. And in case if you try to use here the c command here then also you will get the same value, but in case if you try to use here the mean say mean of 2, 3 and 4 you can you see here this is here 2.

But in case if you try to use here the c command and you input your data as data vector, you can see here it is giving you the correct value three. So, you can see here that using these built in functions is not a very difficult thing, right.

(Refer Slide Time: 15:16)



Overview Over Further Functions

`abs()`	Absolute value
`sqrt()` $\sqrt{2} = 2\hat{}0.5$ sqrt(2)	Square root
`round(), floor(), ceiling()`	Rounding, up and down
`sum(), prod()`	Sum and product
`log(), log10(), log2()`	Logarithms
`exp()` $\longrightarrow$	Exponential function
`sin(), cos(), tan(),` `asin(), acos(), atan()`	Trigonometric functions
cosec sec cot `sinh(), cosh(), tanh(),` `asinh(), acosh(), atanh()`	Hyperbolic functions

6

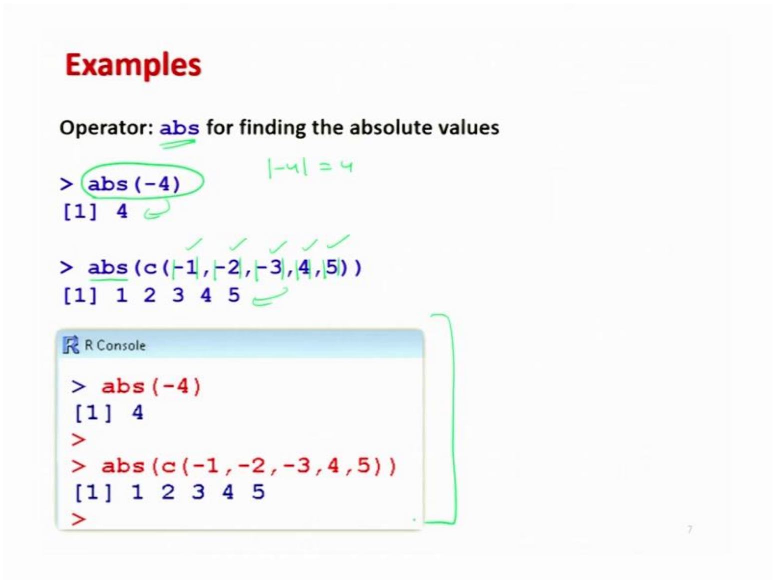
And similarly means, there are many more functions some common useful functions which are there in the R software are as like as a b s which is used to find out the absolute value, s q r t this is for finding out the square root. Remember when we were trying to learn the computation with the power operators we had considered the function like square root of 2 which we had given as 2 hat 0.5.

But now you can also give here it like that s q r t and inside the parenthesis, just write 2 and similarly we have function like here round, floor, ceiling. They are used for rounding of the number floor or say ceiling the number you know that these are the basic mathematical operations.

And similarly if you want to find out the sum or product of some numbers then we have the command here sum s u m and then p r o d product. So, if you simply try to give here some numbers inside the data vector inside the parenthesis, then it will directly give you the sum of those number and similarly different types of log functions which are also available here log, log 10, log 10 is for the log logarithmic value when the base is 10 and so on.

This exp this is used for exponential functions. Then we have the trigonometric function like sin, cos, tan and cosec is something like asin, sec is like acos and cot is like atan right. So, this is something like here cosec, sec, cot. And similarly we have here hyperbolic function also sin h, cos h, tan h, asin h, h a cos h, atan h etc. So, using these functions is straightforward the way you have used the functions like minimum maximum and mean similarly you can use all of this function exactly in the same way.

(Refer Slide Time: 17:26)



Examples

Operator: abs for finding the absolute values

```
> abs(-4)
[1] 4

> abs(c(-1,-2,-3,4,5))
[1] 1 2 3 4 5
```

$|-4| = 4$

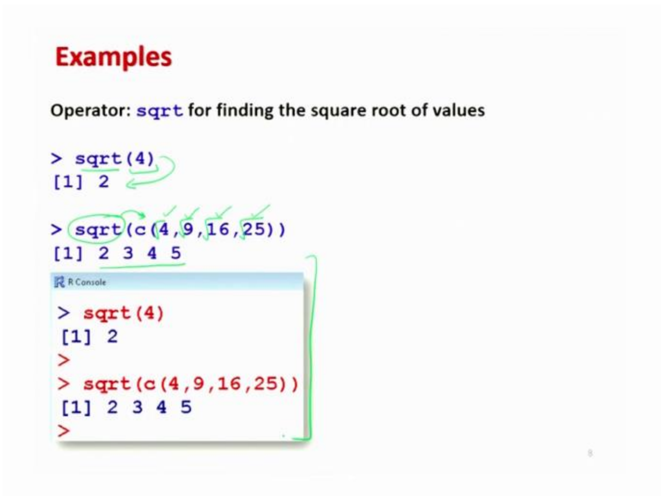
R Console

```
> abs(-4)
[1] 4
>
> abs(c(-1,-2,-3,4,5))
[1] 1 2 3 4 5
>
```

So, I try to take here some example so that I can explain you the different aspects of using these functions. So, for example, if I try to use here the operator abs so this abs is used for finding out the absolute values. So, if you try to see here the function abs -4. So, the absolute value of -4 is actually 4. So, it will give you the answer here 4 and yeah these functions wherever needed. They can also be operated over the data vectors and they will try to operate on each of the element inside the data vector.

For example, if you try to see here if you find the absolute value of say -1, -2, -3, 4 and 5 and you use the concept of data vector to input the values. So, you can see here this absolute function will go inside it and then it will try to find out the absolute value of -1, absolute value of -2, absolute value of -3, absolute value of 4 and absolute value of 5. So, this will give you the answer 1, 2, 3, 4, 5 and you can see here this is the screenshot here, ok.

(Refer Slide Time: 18:33)



Examples

Operator: `sqrt` for finding the square root of values

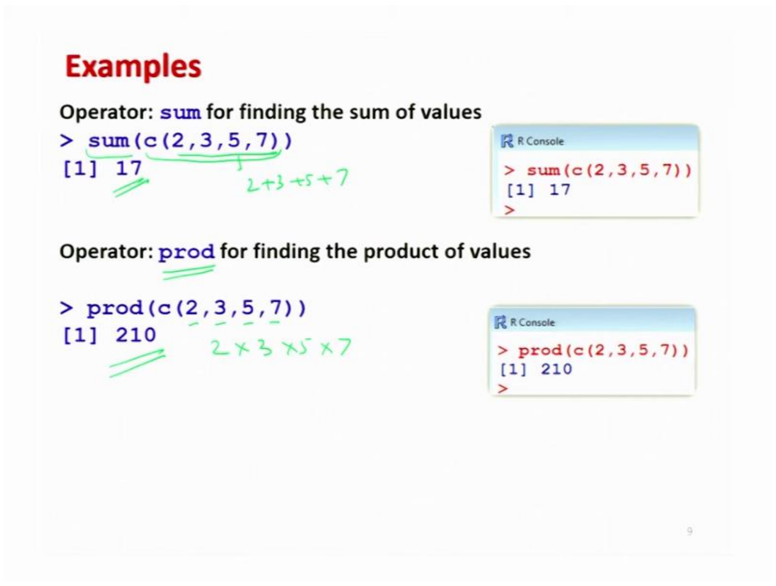
```
> sqrt(4)
[1] 2

> sqrt(c(4,9,16,25))
[1] 2 3 4 5
```

```
> sqrt(4)
[1] 2
>
> sqrt(c(4,9,16,25))
[1] 2 3 4 5
>
```

And similarly if you want to find out the square root of any number you simply have to use s q r t and within the parenthesis you have to give the number. For example, if I try to find out the square root of 4, this is here 2 and similarly if I try to use here the data vector. So, the I try to take here the value 4, 9, 16, 25 and I try to find out here the square root of all these values.

So, when I try to operate it this square root function goes inside the parenthesis and it becomes square root of 4, square root of 9, square root of 16 and square root of 25. So, this will become here 2, 3, 4 and 5 ok and this is here the screenshot.

(Refer Slide Time: 19:14)



Examples

Operator: `sum` for finding the sum of values

```
> sum(c(2,3,5,7))
[1] 17
```

$2+3+5+7$

```
> sum(c(2,3,5,7))
[1] 17
>
```

Operator: `prod` for finding the product of values

```
> prod(c(2,3,5,7))
[1] 210
```

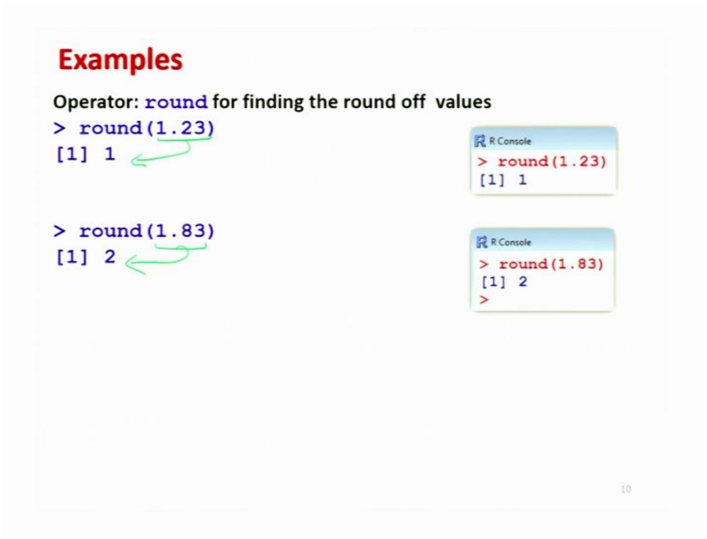
$2\times 3\times 5\times 7$

```
> prod(c(2,3,5,7))
[1] 210
>
```

Similarly, if you try to find the sum of the values for example, if I want to find the sum of 2, 3, 5 and 7; that means, I want to know the value of 2 + 3 + 5 + 7. So, then I can use here the sum and using the data vector I can try write down all these values and this value will come out to be here 17.

And similarly if I want to find out the product of some numbers for example, the product of 2, 3, 5 and 7 which is like 2 into 3 into 5 into 7. So, this will come out to be here 210. So, you can see here it is very convenient to do the mathematical computations in this R software.

(Refer Slide Time: 19:57)



Examples

Operator: round for finding the round off values

```
> round(1.23)
[1] 1
```

```
> round(1.83)
[1] 2
```

R Console

```
> round(1.23)
[1] 1
```

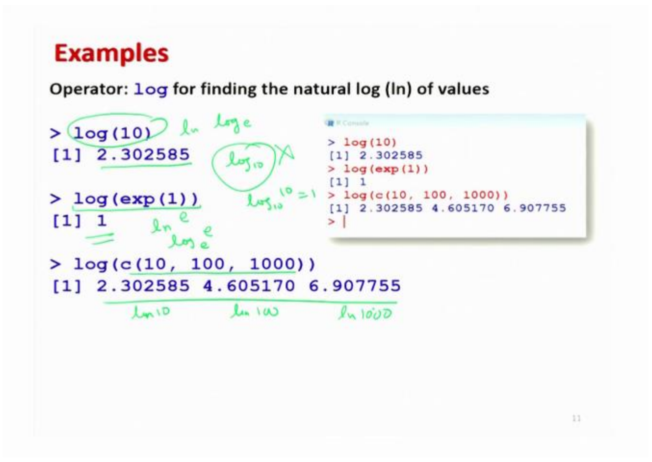
R Console

```
> round(1.83)
[1] 2
>
```

10

And similarly if you want to find out the round off value of 1.23. So, we know that in case if the number is more than 1.5, then it is rounded off to 2. Otherwise it is rounded off to 1. For example, so it is here 1.23 which is less than 1.5. So, it will be rounded off to 1, but if I try to take here 1.83 which is more than 1.5, so it will be rounded off to 2. So, you can see here these operations are very simple and very easy to use and similarly if you want to use here the log.

(Refer Slide Time: 20:31)



Examples

Operator: `log` for finding the natural log (ln) of values

```
> log(10)
[1] 2.302585

> log(exp(1))
[1] 1

> log(c(10, 100, 1000))
[1] 2.302585 4.605170 6.907755
```

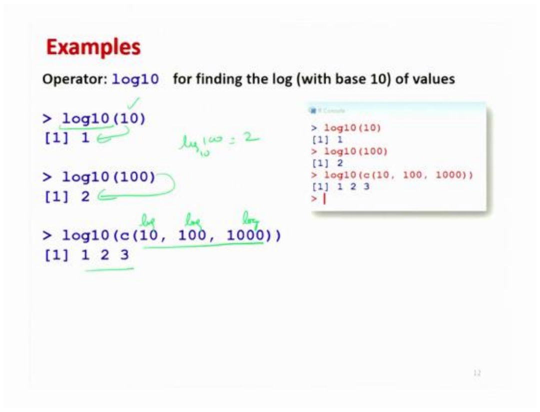
```
> log(10)
[1] 2.302585
> log(exp(1))
[1] 1
> log(c(10, 100, 1000))
[1] 2.302585 4.605170 6.907755
> |
```

So, if you try to see here if you try to find out here the log of 10. So, log of 10 here is coming out to be 2, 2.3. Actually, if you try to see here this log function you have to remember this gives you the natural log which was your log with respect to the base e and it is not the log value with respect to the base 10.

Why? Because log of 10 when the base is 10 is equal to 1 and if you want to verify it here you can see here that if you want to find out the log of exponential of 1. So, this is something like l n of e or this is here log of e base e. So, this value will come out to be here 1.

So, log is a used for finding out the natural log values and yeah this operation is valid on the data vector also. So, if you try to write down here log of c 10, 100, 1000. It will give you the values of log of 10, log of 100 and log of 1000, ok.

(Refer Slide Time: 21:45)



Examples

Operator: `log10` for finding the log (with base 10) of values

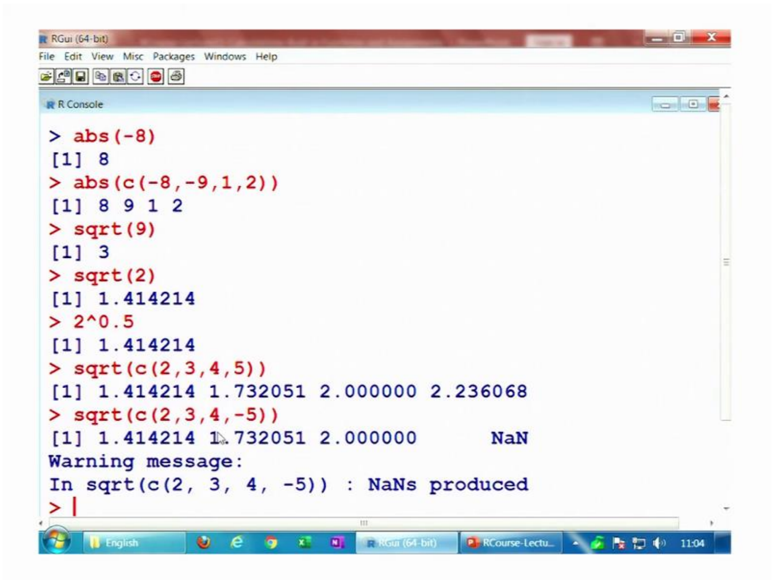
```
> log10(10)
[1] 1

> log10(100)
[1] 2

> log10(c(10, 100, 1000))
[1] 1 2 3
```

```
> log10(10)
[1] 1
> log10(100)
[1] 2
> log10(c(10, 100, 1000))
[1] 1 2 3
> |
```

And similarly if you want to find out the log the base 10 then you have to use the command log 10 right. So, if you try to see here log 10 and the of the value 10 is here 1 and similarly you know that log of 100 with the base 10 is equal to 2. So, this will come here come out to be here 2.

And similarly if you try to use here the data vector then the log of 10, 100 and 1000 will be like a log of 10, log of 100 and log of 1000 which will come be coming here as say 1, 2, and 3, right. So, let me try to first give you the example of these things. So, that you get here more confidence that how they will be working on the R software.

(Refer Slide Time: 22:37)



```
> abs(-8)
[1] 8
> abs(c(-8,-9,1,2))
[1] 8 9 1 2
> sqrt(9)
[1] 3
> sqrt(2)
[1] 1.414214
> 2^0.5
[1] 1.414214
> sqrt(c(2,3,4,5))
[1] 1.414214 1.732051 2.000000 2.236068
> sqrt(c(2,3,4,-5))
[1] 1.414214 1.732051 2.000000      NaN
Warning message:
In sqrt(c(2, 3, 4, -5)) : NaNs produced
>
```

So, if you try to see here I will try to take the example of all the numbers whatever I have considered. So, I will try to take a absolute value of here say -8. So, you can see here this is here 8 and if I try to find out here absolute value of here some data vector say -8, -9, then 1 and 2.

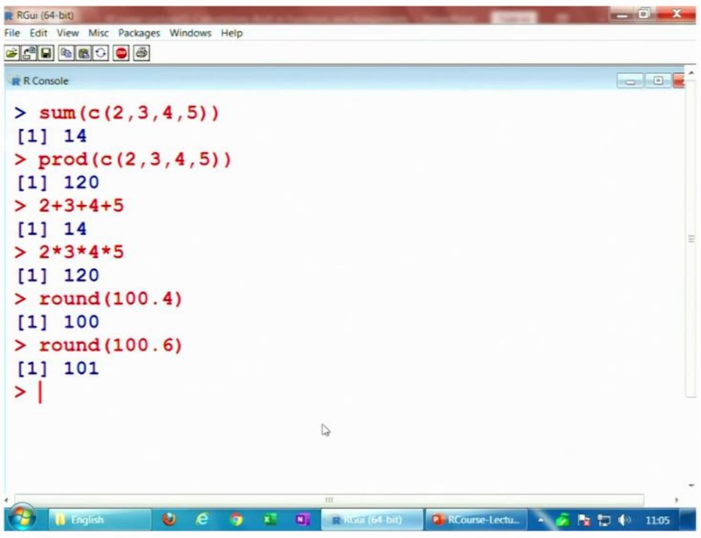
So, you can see here that it is coming out to be like this the absolute value of 8 is 8, the absolute value of -9 is 9 and absolute value of 1 and 2; they are 1 and 2. So, you can see here that the absolute values of -8, -9, 1 and 2 they are 8, 9, 1 and 2 respectively.

Similarly, if you try to find out here square root, so square root you can see here suppose I square root of here 9 this will be here 3. And earlier in the earlier lecture you had found the square root of 2. This was 1.414 which is coming here the same and yeah if you try to see here I can show you here this is square root of 2 which you try to compute this is the same value.

And similarly if you try to find out the square root of here some values like here 2, 3, 4, 5 and so on. So, it will come out to be like this. So, this is like a first value is 1.414 is the square root of 2, then the remaining these 3 values are the square root of 3, square root of 4 and square root of 5, right. And now if you try to suppose, for example, give here some negative value. What do you expect?

This will give you, there is some warning because negative number cannot have the square root. So, it is giving you here NaN, right. So, I will try to discuss about what are this NA, what is NaN, what is NULL, etc. in more detail in the forthcoming lectures. But here I wanted to show you that if you are trying to do some wrong mathematical calculations the R will give you here a warning message, right, ok. So, now look let us see that what are the other operation that we have to do which is here sum and product.

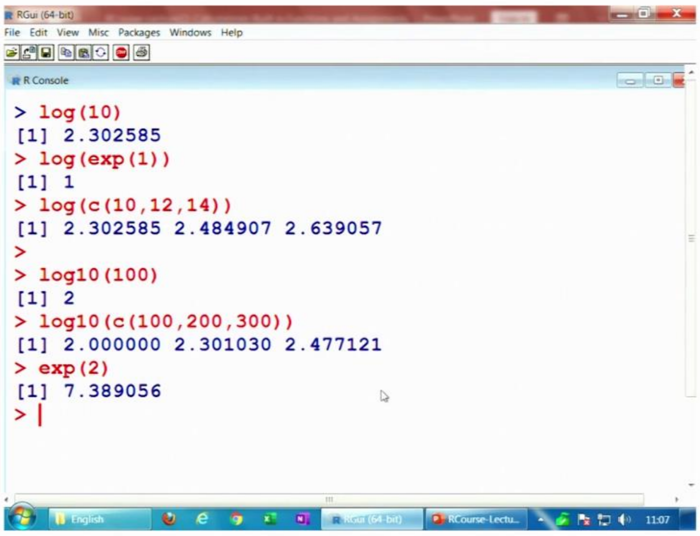
(Refer Slide Time: 20:54)


```
> sum(c(2,3,4,5))
[1] 14
> prod(c(2,3,4,5))
[1] 120
> 2+3+4+5
[1] 14
> 2*3*4*5
[1] 120
> round(100.4)
[1] 100
> round(100.6)
[1] 101
> |
```

So, you can see here the if you try to see here sum of c say 2, 3, 4, 5 etc. whatever you want it will give you here like I say this and if you want to find out here the product of these numbers. So, you simply have to give here the product and if you want to just verify it here. So, you can see here 2 + 3 + 4 + 5 this is giving you here 14 and 2 into 3 into 4 into 5, this is giving you here 120, right.

So, this is how you can do this sum and product of the operations and now if you want to use here the round. Suppose if I say 100.4, so this will be your here 100 and if I try to take a 100.6, it will become here 101. So, that is the way this rounding of the values in mathematics actually work, right.

(Refer Slide Time: 25:58)

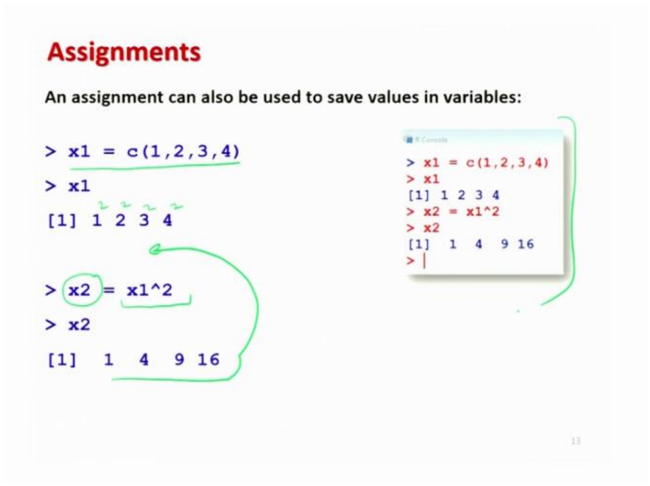


```
> log(10)
[1] 2.302585
> log(exp(1))
[1] 1
> log(c(10,12,14))
[1] 2.302585 2.484907 2.639057
>
> log10(100)
[1] 2
> log10(c(100,200,300))
[1] 2.000000 2.301030 2.477121
> exp(2)
[1] 7.389056
> |
```

So, now, in case if you try to find out here the value of log. So, you can log you can see here log of say here 10 this will give you here this value. But if you try to find out here the log of say exponential of here 1. So, you can see here this value will come out to be 1. So, that ensures that the log function is trying to find out the natural log. So, if you want to find out the log of a data vector. So, you can see here 10, 12, 17 etc. and you can see here there are three values which are representing the log of 10, log of 12 and log of 14; three values, right.

Similarly, if you want to find out here the log value with respect to the base 10, so you know that log of 100 when the base is 10 is 2. So, you can see here this is here 2 and if you want to make it here more details you can see here log of the say values like here 100, 200, 300. And if you try to enter here you will get here the value of say, log of 100 is 2, log of 200 is 2.301030 and log of 300 is 2.477121. So you can see here finding out these values is not difficult. Similarly if you want to find out the value of exponential 2 for example, this is 7.38.

(Refer Slide Time: 27:27)



Assignments

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

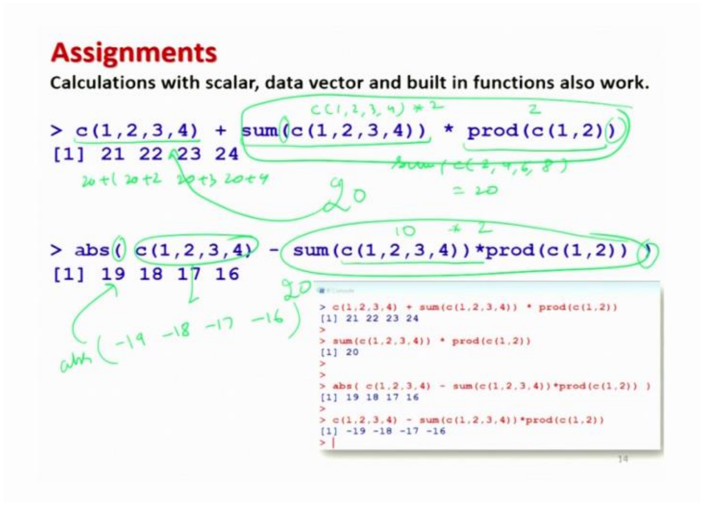
```
> x1 = c(1,2,3,4)
> x1
[1] 1 2 3 4

> x2 = x1^2
> x2
[1]  1  4  9 16
```

```
> x1 = c(1,2,3,4)
> x1
[1] 1 2 3 4
> x2 = x1^2
> x2
[1]  1  4  9 16
> |
```

So, this is how you can move further without any problem and I just want to show you here one more operation. So, that will really help you when you are trying to write your own programs. So, whatever values you try to give in a variable in the form of a data vector that can also be assigned to a variable.

For example, if you try to see here I give here an input x1 as say data vector of the values 1, 2, 3, 4 and then I define a new variable x1 square and I try to store it in a new variable x2. So, you can see here this x2 will give me here the value 1, 4, 9, 16 which are something like the if square of 1, square of 2, square of 3 and square of 4, right. So, this is also possible. So, you can see here these are the outcomes on the R console, right.

(Refer Slide Time: 28:24)



Assignments

Calculations with scalar, data vector and built in functions also work.

```
> c(1,2,3,4) + sum(c(1,2,3,4)) * prod(c(1,2))
[1] 21 22 23 24

> abs( c(1,2,3,4) - sum(c(1,2,3,4))*prod(c(1,2)) )
[1] 19 18 17 16
```

```
> c(1,2,3,4) + sum(c(1,2,3,4)) * prod(c(1,2))
[1] 21 22 23 24
>
> sum(c(1,2,3,4)) * prod(c(1,2))
[1] 20
>
>
> abs( c(1,2,3,4) - sum(c(1,2,3,4))*prod(c(1,2)) )
[1] 19 18 17 16
>
> c(1,2,3,4) - sum(c(1,2,3,4))*prod(c(1,2))
[1] -19 -18 -17 -16
> |
```

Now, I want to give you one more idea. So, now, you have learnt two types of operations one with scalar with data vector and with this built in function. So, actually you can combine any one of them together and the same mathematical rules will be followed. For example, if I try to take here data vector c 1, 2, 3, 4 and then I try to take here a function of sum of 1, 2, 3, 4 and then the product of 1 and 2, right.

So, if you try to see here the parenthesis is here. So, first of all the product of 1 and 2 this will be here 2 and then this will be here c 1, 2, 3 and 4. So, this value will be multiplied by here 2 for finally, and this will become here sum of c 2, 4, 6 and here 8 and then sum of this will become here 2 + 4 is 6; 6 + 6 is 12 and 12 + 8 is 20. So, this entire value will become here 20.

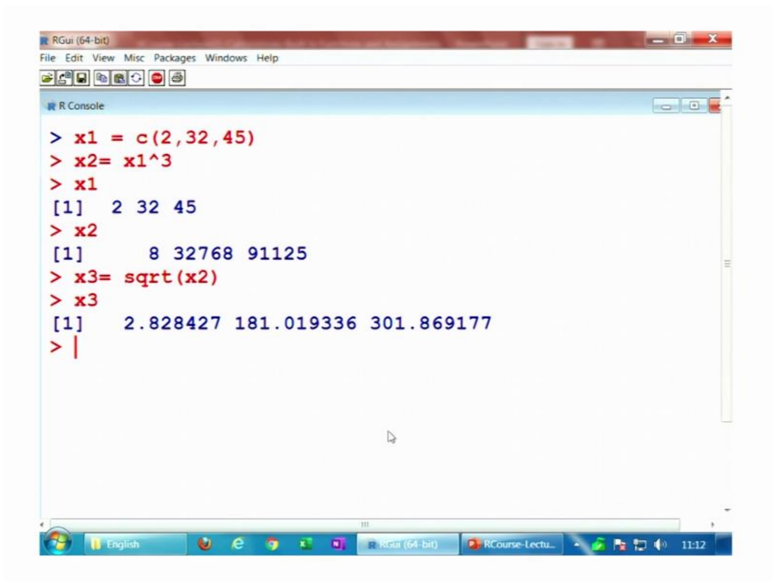
So, now this 20 is going to be added in this first data vector. So, this will become here 20 + 1, 20 + 2, 20 + 3 and 20 + 4. So, that is the same exercise which you did when you try to add a scalar with the data vector, right. Similarly if you try to find out the absolute value of some function.

So, the values are contained inside this parenthesis and if you try to see here this is once again the same thing c 1, 2, 3, 4. So, this is here 1 + 2 is 3; 3, 3 6; and 6, 4 here 10 and then product of here this is here 1 into 2 this is here 2. So, this is here 10 into 2, so whole this value is going to be 20.

And now this 20 is going to be subtracted from this data vector. So, this will become here 1 - 20 -19; then 2 - 20 - 18; 3 - 20 - 17 and 4 - 20 - 16 and then you try to operate the absolute function over this and this will give you here the values 19, 18, 17, 16.

So, you can see here that these things are not very difficult, but let us try to operate them on the R console and try to see do they really work here work or not, right. So, let me try to copy these values and then try to operate. So, this will save our some time.

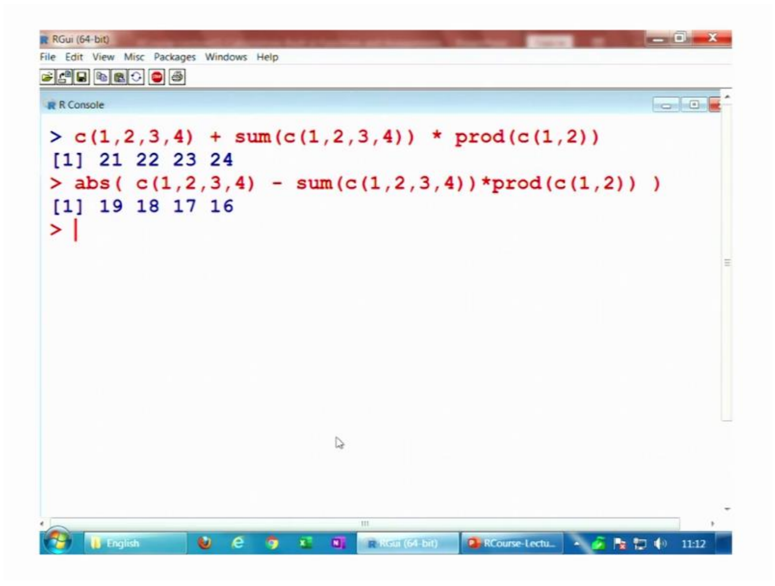
(Refer Slide Time: 31:07)



```
> x1 = c(2,32,45)
> x2= x1^3
> x1
[1]  2 32 45
> x2
[1]     8 32768 91125
> x3= sqrt(x2)
> x3
[1]   2.828427 181.019336 301.869177
> |
```

So, before going further let me try to show you this example that I try to take here c as 2, 32, 45 like this and then I try to define here x2 which is here say here x1 say cube. So, you can see here x1 here is like this and x2 here is like this. And in case if you want to define here either values here say x3 here s square root of say here x2. Then also you can do it so you can see here the x3 will comes out to be like this. So, I have not made any manual calculation, but R is doing my job.

(Refer Slide Time: 31:48)



```
> c(1,2,3,4) + sum(c(1,2,3,4)) * prod(c(1,2))
[1] 21 22 23 24
> abs( c(1,2,3,4) - sum(c(1,2,3,4))*prod(c(1,2)) )
[1] 19 18 17 16
> |
```

And similarly if you try to see here if you try to do the same operation here, you will get here 21, 22, 23, 24 and if you try to do this operation here on the R console, you once again you get the same outcome. So, now, you can see here that when we are trying to work with built in function first of all it is not difficult at all and it is not very difficult to understand how R is working on it.

And I hope you will appreciate that a built in functions helps us a lot and whenever we are trying to do some complicated calculations, lengthy calculations, then they help us a lot. And this is one of the very important feature of the R software which made the R software so popular.

So, now, I stop in this lecture, but as I always say, now once again today you have a good amount of homework. And you need to practice, try to consider these built in functions, some functions I have considered some function I have not considered. Try to consider them try to see whether they are working or not try to use some scalar values data vectors.

And try to see are they working or not and above of all whatever you have learnt in the last couple of lectures with different types of operation with the scalars data vectors etc., try to combine them with the built in functions and this will give you a more deeper knowledge. You will have a more whole of the working of the R software. So, you try to practice it and I will see you in the next lecture, till then good bye.