Essentials of Data Science with R Software - 2
Sampling Theory and Linear Regression Analysis

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

Introduction to R Software
Lecture - 07
Data Handling

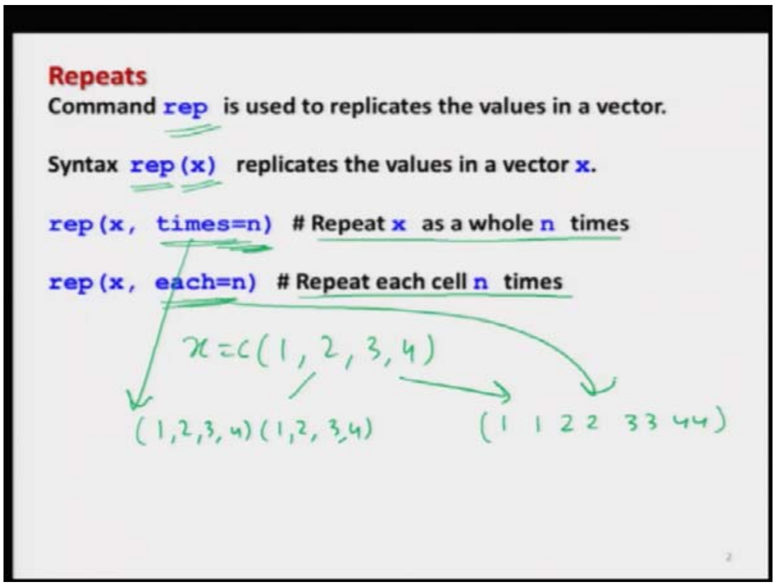
Hello, welcome to the course Essentials of Data Science with R Software 2, where we are dealing with the topics of Sampling Theory and Linear Regression Analysis. And we will be continuing with the topics on Introduction to R Software in this lecture, in this module. So, in this lecture also, we will continue with small introduction to the different commands and functions in the R software.

So, in this lecture, I am essentially going to address different aspect of Data Handling. There are small commands, very simple commands, which are very useful when you are trying to handle the data. And particularly in data sciences, these small commands will give you bigger outcome. Why? Because whenever you are getting any inference, you have to compile it; just by looking at the numbers, nobody will be able to understand what you have done.

For example, if I take simple situation, where you have suppose one million data set of say male and female candidates and you simply want to know that, how many males are there how many females are there or you want to arrange the data in a particular way that, certain number of males and certain number of females are arranged in a required format.

Then how to achieve those things? These small commands are going to help you. So, I will try to take up one command at a time and I will try to show you some example and context. So, let us start.

(Refer Slide Time: 02:03)



Repeats

Command `rep` is used to replicates the values in a vector.

Syntax `rep(x)` replicates the values in a vector `x`.

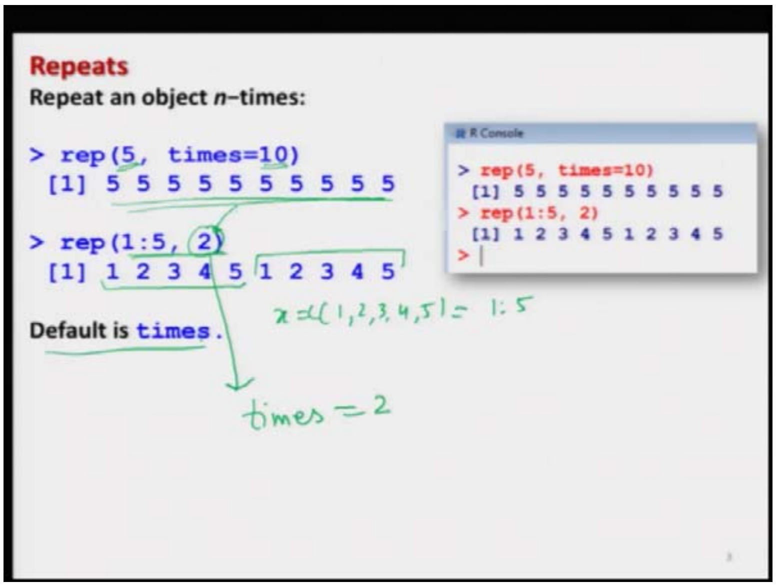
```
rep(x, times=n)  # Repeat x as a whole n times
rep(x, each=n)  # Repeat each cell n times
```

The first command which I am going to explain here is the rep command. As a rep itself indicates the meaning, this means replicate; that means if you want to copy the same value again and again. So, for that, the command here is rep and inside the parenthesis, you have to write down the data vector; say here x. Now, there are two options that, when you are trying to write down the data vector, what do you want?

Whether you want to repeat the entire data vector or you want to repeat the values? For example, if I say here x is equal to 1, 2, 3 and 4 with c command. So, one option is this; suppose if I want to repeat this values two times. So, one option is this I can repeat it 1, 2, 3, 4 and once again 1, 2, 3, 4 or second option is this I can repeat the values 1 1 2 2 3 3 4 4. So, these are two different options.

So, in order to achieve them, there is an option that I can include in my rep command; this is a times equal to n or second option is each is equal to n each is equal to small n. So, when I say times equal to n, this means the entire data vector is repeated n times. And when I say each is equal to 9; that means every data value is repeated n times. For example, this corresponds to here times and this correspond to here each, right.

(Refer Slide Time: 03:57)



Repeats

Repeat an object *n*–times:

```
> rep(5, times=10)
 [1] 5 5 5 5 5 5 5 5 5 5

> rep(1:5, 2)
 [1] 1 2 3 4 5 1 2 3 4 5
```

Default is `times`.

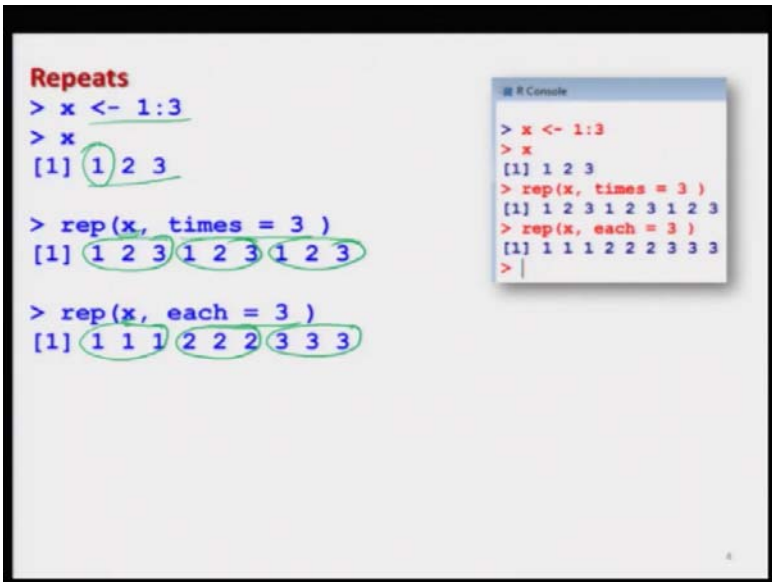
```
> rep(5, times=10)
 [1] 5 5 5 5 5 5 5 5 5 5
> rep(1:5, 2)
 [1] 1 2 3 4 5 1 2 3 4 5
> |
```

So, now, let me take here some example and try to illustrate it. For example, if I say just repeat a value 5, suppose I take a data value 5 and I want to repeat it 10 times. So, I can write down here rep 5 and comma times equal to 10 and you will get here this type of outcome.

And then I try to create a data vector here x equal to here 1, 2, 3, 4 and 5, which I can denote by here simply here 1 colon 5; this data vector has to be repeated two times. So, you can see here that this 1, 2, 3, 4, 5 and 1, 2, 3, 4, 5 they are repeated; the entire data vector is repeated.

But you have to observe that, here I have not written anything; but you can see here that it is considering this 2 as times equal to 2; it is not each is equal to 2. So, the moral of the story is this, when you are not writing clearly each or time; then default is times, ok.

(Refer Slide Time: 05:15)



Repeats

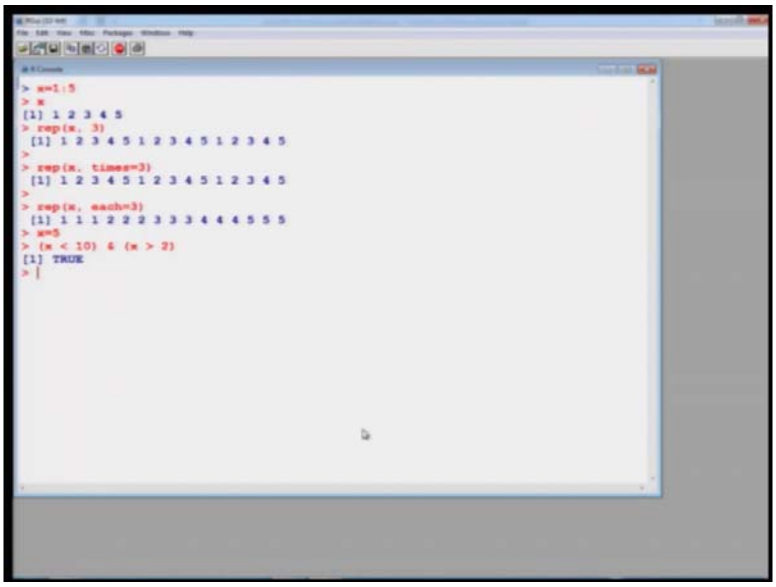
```
> x <- 1:3
> x
[1] 1 2 3

> rep(x, times = 3 )
[1] 1 2 3 1 2 3 1 2 3

> rep(x, each = 3 )
[1] 1 1 1 2 2 2 3 3 3
```

```
> x <- 1:3
> x
[1] 1 2 3
> rep(x, times = 3 )
[1] 1 2 3 1 2 3 1 2 3
> rep(x, each = 3 )
[1] 1 1 1 2 2 2 3 3 3
> |
```

Now, let me try to take here one more example to illustrate this thing. Suppose I try to take a data vector x equal to 1, 2 and 3 like this and I try to repeat this data vector in x 1, 2, 3, say times equal to 3 and each is equal to 3. So, this will clearly explain you the difference between the two. So, times mean the entire data vector is going to be repeated. So, this entire data vector is 1, 2, 3; 1, 2, 3 and 1, 2, 3 which is repeated 3 times

And now when I say that each is equal to 3; that means each value in the data vector is repeated three times. For example, there is a value here 1 this is repeated three times, value 2 this is repeated three times and value 3 that is repeated three times. So, first let me try to show you all these things on the R console and then I will try to move further, right.

(Refer Slide Time: 06:12)

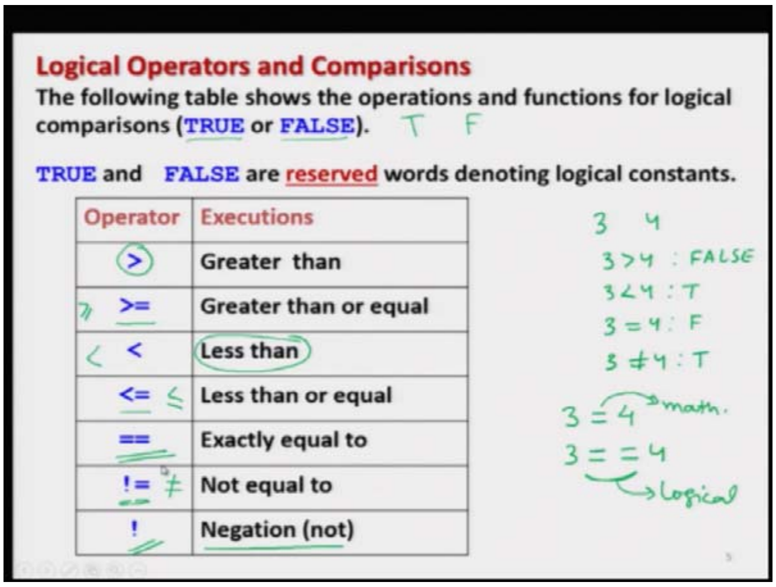


```
> x=1:5
> x
[1] 1 2 3 4 5
> rep(x, 3)
 [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
>
> rep(x, times=3)
 [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
>
> rep(x, each=3)
 [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
> x=5
> (x < 10) & (x > 2)
[1] TRUE
> |
```

So, now I try to create here a data vector suppose here 1 to 5. So, you can see here this is x equal to 1, 2, 3, 4, 5 and now I try to take here rep of x and I do not write anything, but I simply write here 3. So, you can see here the entire data vector here 1, 2, 3, 4, 5 this has been repeated here 1 2 3 4 5, then again 1 2 3 4 5 and then again 1 2 3 4 5. But now just to make it more precise; I will write here say times equal to 3 and then you can see here that, you are getting the same command.

So, this indicates that, the default is here times. But if I try to write down here each, then you can see here that each value in the data vector is repeated three times 1 1 1, 2 2 2, 3 3 3 and so on 4 4 4 and 5 5 5. So, this is what I mean, now these things can be used in various type of situation when you are trying to deal with the data.

(Refer Slide Time: 07:20)



Logical Operators and Comparisons

The following table shows the operations and functions for logical comparisons (TRUE or FALSE). T F

TRUE and FALSE are reserved words denoting logical constants.

Operator	Executions
>	Greater than
≥ >=	Greater than or equal
< <	Less than
<= ≤	Less than or equal
==	Exactly equal to
!= ≠	Not equal to
!	Negation (not)

3 4

3>4 : FALSE

3<4 : T

3=4 : F

3≠4 : T

$3 = 4 \rightarrow$ math.

$3 == 4 \rightarrow$ logical

Now, after this I come on another aspect, which is the logical operators which are used for comparison. So, two logical operators I already have explained you, which are TRUE and FALSE and they are stated by the capital letters T R U E and capital letters F A L S E or you can also use T or F; because the first letter in capital alphabet. So, that will also work.

So, they are also the reserved words. So, they cannot be used to denote any variable name. Beside those things, we have couple of more logical operators which are used for comparison. Now, you see, what is the meaning between the two. For example, if I say I have got here two numbers say 3 and here 4. So, one thing is this, I want to know that whether any number is smaller or greater than to any number or equal to or not equal to.

So, I can say here whether 3 is greater than 4; 3 is smaller than 4; 3 is equal to 4; 3 is not equal to 4. So, what are these things? So, the outcome of these computation will be some answer in terms of true or false; whether 3 is greater than 4, no this is FALSE; 3 is smaller than 4, yes TRUE; 3 is equal to 4, FALSE; 3 is not equal to 4, TRUE.

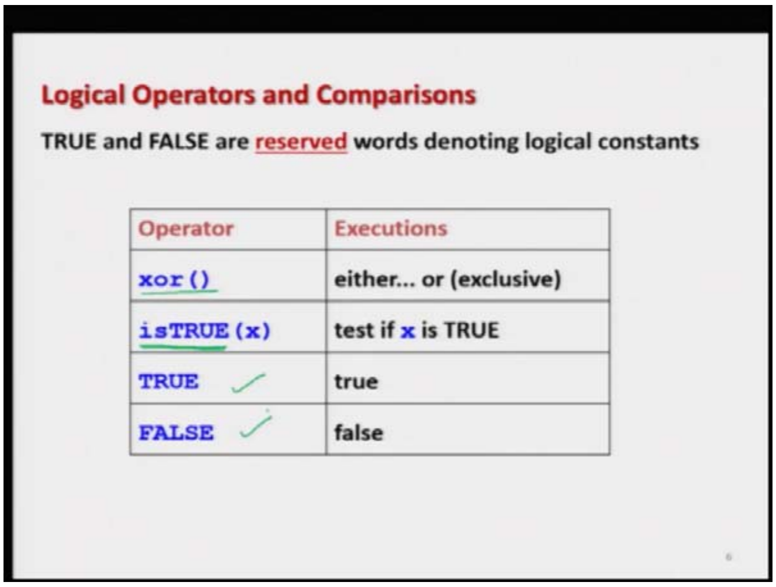
So, these are the different operators which are the logical operators. If you want to make a comparison with the greater than sign, suppose you want to know whether some number is greater than some number or not; then we have to use the sign greater than.

And if you want to use the sign here greater than or equal to; means the number can be greater or number can be equal, both the possibilities are included, then you have to use the symbol greater than and equal to. And similarly if you want to compare with the smaller than, less than; then you simply have to use the usual command say less than. And if you want to go with less than or equal to like this one; then you have to use the symbol less than and equal. And now you have to be careful with one thing.

Suppose if I want to compare two number 3 and 4 whether they are equal or not. So, there are two ways to write it; 3 equal to 4 or 3 and I write the equality sign two times. This is a mathematical operator and this is a logical operator; that means if I write single equality sign that is mathematical operator and if I write double equality signs, then this is the logical operator. So, if I want to compare two numbers, then I have to use the logical equality sign and then I have to write down two equality signs side by side.

Similarly, if I want to use the logical operator not equal to, which we used to denote by this thing in mathematics; then I have to write down exclamation sign and then equality sign. And if I simply want to use say negation; that means not, then I simply have to use here the exclamation sign. So, these things will be very useful when you are trying to deal with data sciences at later stage.

(Refer Slide Time: 11:18)



Logical Operators and Comparisons

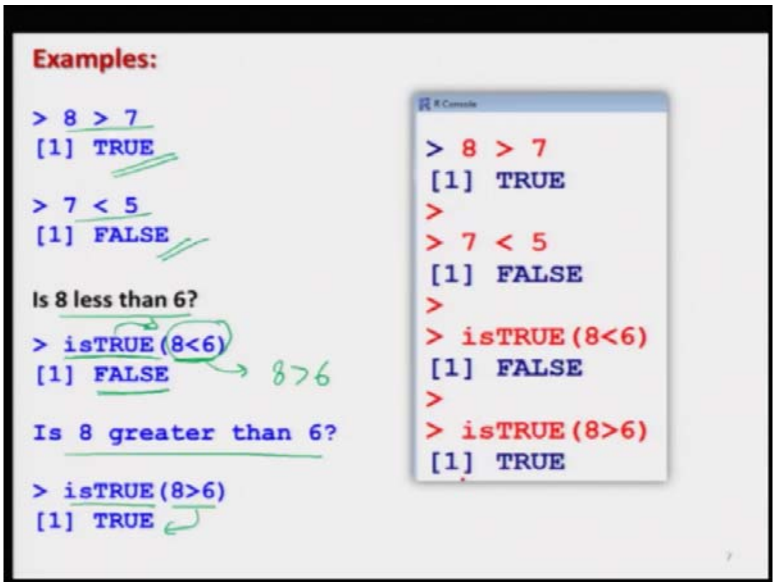
TRUE and FALSE are reserved words denoting logical constants

Operator	Executions
`xor()`	either... or (exclusive)
`isTRUE(x)`	test if **x** is TRUE
`TRUE` ✓	true
`FALSE` ✓	false

And so, I will try to take some example to explain you. And similarly now means, means another application of logical operator is that, you want to compare two expressions. So, those two expressions can be compared with or say and. So, we have a command here xor, which explain us that means either or type of logical operator.

And if I really want to know whether some expression is true or not; for that we have a command here is true, remember and observe this thing i s is in a small letter and TRUE is in capital letter and inside the parenthesis, you have to write the expression for x. And it will actually test whether x is true or not for a given value of data. And similarly as we have discussed capital TRUE and capital FALSE, they are indicating the TRUE and FALSE status of the command.

(Refer Slide Time: 12:19)



Examples:

```
> 8 > 7
[1] TRUE

> 7 < 5
[1] FALSE
```

Is 8 less than 6?

```
> isTRUE(8<6)
[1] FALSE
```

$8>6$

```
Is 8 greater than 6?

> isTRUE(8>6)
[1] TRUE
```

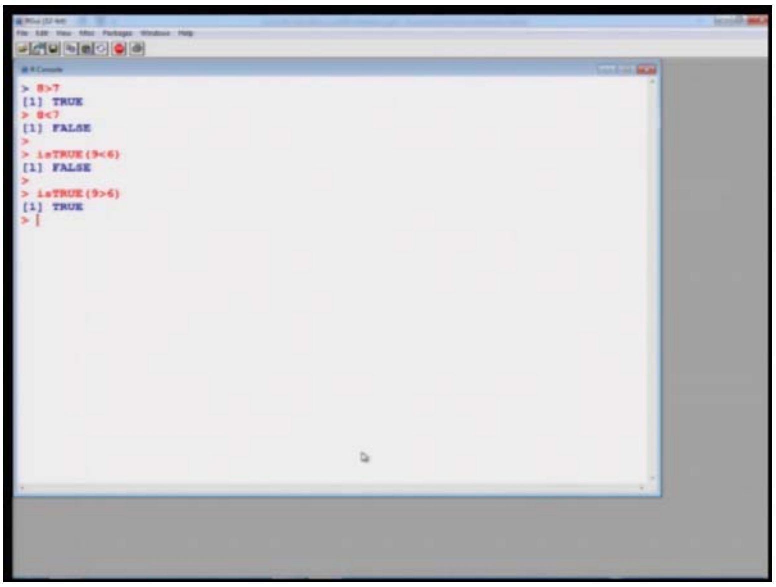
```
> 8 > 7
[1] TRUE
>
> 7 < 5
[1] FALSE
>
> isTRUE(8<6)
[1] FALSE
>
> isTRUE(8>6)
[1] TRUE
```

Now, let me try to take some example to explain you. Suppose I want to know whether 8 is greater than 7 or not. So, if I simply type here 8 greater than 7, you will get here an answer TRUE and it is correct, yeah 8 is greater than 2 says 7. And similarly if you want to test 7 is smaller than 5; I will simply say 7 less than 5, which is FALSE and yeah, that is correct.

Now, suppose I want to test the statement; is 8 less than 6? So, I can write down here; is true 8 less than 6? Means I want to test that, I have a statement which I am believing that this isTRUE and now please test and let me know whether my assumption is true or false. So, I had assume here that 8 is less than 6; but now when I ask R to check it, R says it is FALSE, that means my this assumption is wrong, this is FALSE.

So, the opposite is TRUE, which will hold true; that means 8 will be greater than 6 which is obvious, right. And similarly if I try to tests; is 8 greater than 6? So, I have to use the command is TRUE and inside the parenthesis 8 greater than 6 and you can see here that this is coming out to be TRUE, right ok. Now, let us try to come to R console and I will try to show you that whether these things are happening or not.

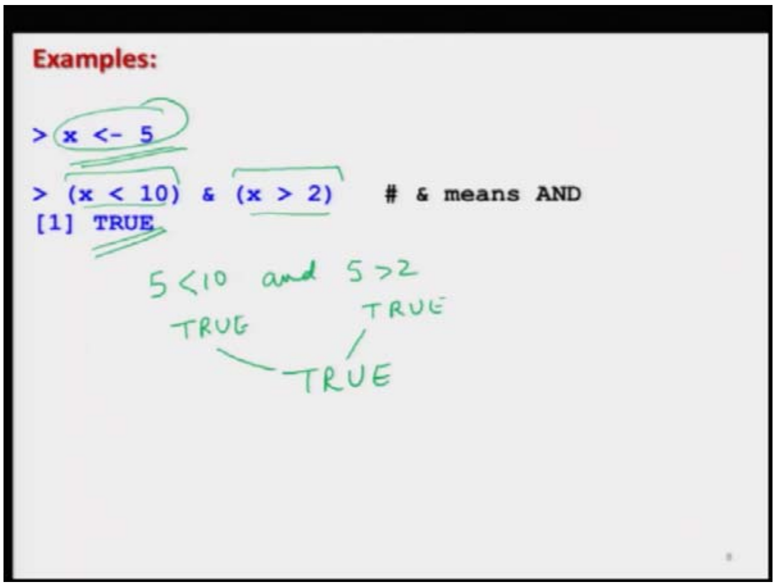
(Refer Slide Time: 13:59)



```
> 8>7
[1] TRUE
> 8<7
[1] FALSE
>
> isTRUE(9<6)
[1] FALSE
>
> isTRUE(9>6)
[1] TRUE
> 
```

So, you can see here if I try to write down here a command 8 greater than 7, it will give us answer TRUE. And similarly if I try to write down the command 8 is less than 7, it will give me an answer FALSE. Similarly if I want to find out here; isTRUE say 9 is less than 6, so and you can see here, this will give me here answer FALSE, right. And similarly if I try to say here n say 9 is greater than 6 and isTRUE, it will give me answer TRUE. So, you can see that these things are working on R console also.

(Refer Slide Time: 14:44)



Examples:

```
> x <- 5
> (x < 10) & (x > 2)     # & means AND
[1] TRUE
```

$5 < 10$ and $5 > 2$

TRUE TRUE

TRUE

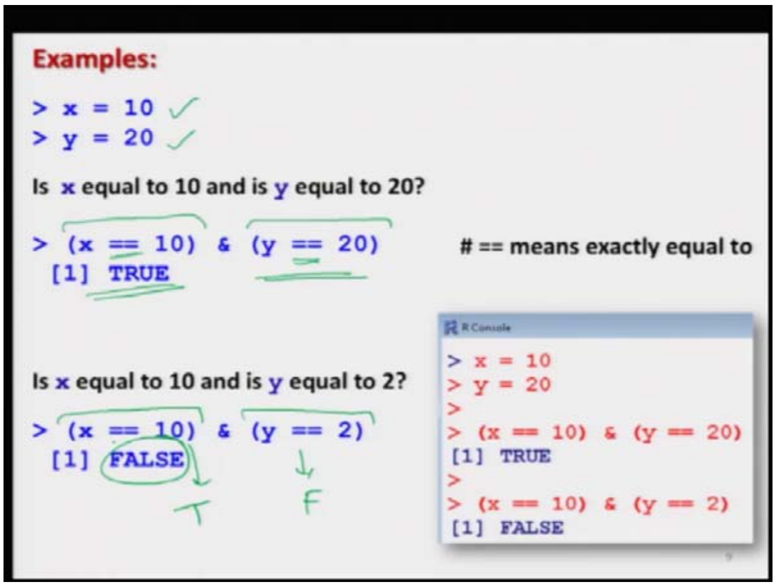
And now I try to give you some more detail that for this logical comparison, I will take couple of example. Suppose if I take x equal to 5 and suppose I have an expression x less than 10 and x greater than 2.

Well, I am not going here to discuss the operator for and or in more detail; but as I said earlier, you can refer to my earlier lectures on the course Introduction to R software, where I have given these commands in more detail, there can be single & double & all those things. So, well anyway, my objective here is to give you a brief overview, so anyway.

So, this means that, I want to check for the given value of x equal to 5; whether this statement x less than 10 and x greater than 2, are they correct? So, let me try to write down here 5 less than 10 and 5 greater than 2, is this TRUE? So, 5 is smaller than TRUE, 10 it is TRUE; 5 is greater than 2, it is TRUE.

So, now TRUE and TRUE is TRUE yeah; there is a mathematics behind the logical operator that TRUE and TRUE is TRUE, TRUE and FALSE is FALSE and FALSE is FALSE and so on for depending upon and or so anyway. So, this because the same answer that you are getting over here.

(Refer Slide Time: 16:20)



Examples:

```
> x = 10 ✓
> y = 20 ✓
```

Is x equal to 10 and is y equal to 20?

```
> (x == 10) & (y == 20)
  [1] TRUE
```

== means exactly equal to

Is x equal to 10 and is y equal to 2?

```
> (x == 10) & (y == 2)
  [1] FALSE
```

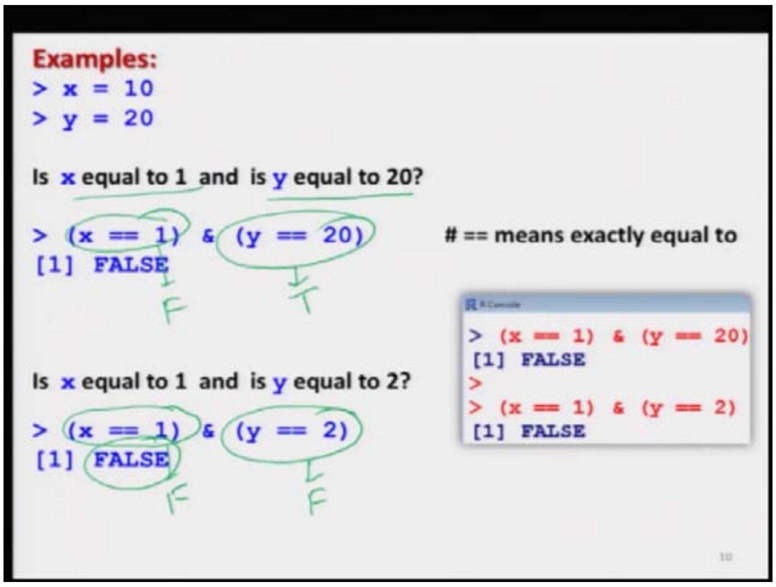
T F

```
> x = 10
> y = 20
>
> (x == 10) & (y == 20)
[1] TRUE
>
> (x == 10) & (y == 2)
[1] FALSE
```

Now, I try to take here say some more example; suppose I take two values x equal to 10 and y equal to 20 and I want to check whether x is equal to 10 or and y equal to 20 or not. So, you can see here, I am using here double equality sign which are the logical operators and you can see here once I am taking here the value here x equal to 10 and y equal to 20.

So, here in this case the x is 10 here the y is 20; so that means both are TRUE, the TRUE and TRUE is TRUE. Now, if I try to change my sentence, change my this context and if I try to test whether x is equal to 10 or not and y is equal to 2 or not. So obviously, here x is equal to 10. So, that is here TRUE; but y is equal to here 20, y is not equal to 2. So, y y double equality sign 2 will come out to be here FALSE and TRUE and FALSE altogether will give me FALSE, right.

(Refer Slide Time: 17:22)



Examples:

```
> x = 10
> y = 20
```

Is x equal to 1 and is y equal to 20?

```
> (x == 1) & (y == 20)
[1] FALSE
```

F T

== means exactly equal to

```
> (x == 1) & (y == 20)
[1] FALSE
>
> (x == 1) & (y == 2)
[1] FALSE
```

Is x equal to 1 and is y equal to 2?

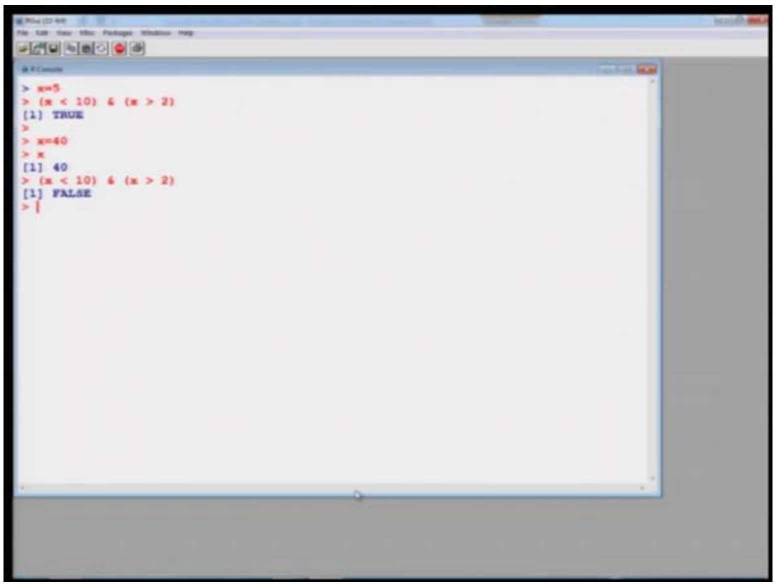
```
> (x == 1) & (y == 2)
[1] FALSE
```

F F

So, this is what I meant. And similarly if I means you can take some more example; if I try to take it here x equal to 1 and y equal to 20. So, again x is equal to here 10. So, this is going to be here FALSE; y equal to 20, this is going to be here TRUE. So, FALSE and TREUE is again FALSE. And if I try to take it here x equal to 1 and y equal to 2; then obviously when I say x equal to 1 which is FALSE. Why? Because x equal to 10 and if I try to take y equal to 2, this is also FALSE; because y is equal to 20.

So, FALSE and FALSE, they are going to give me here FALSE, right. So, these are different types of things which you can do; but let me try to first show you these things on the R console over here.

(Refer Slide Time: 18:15)

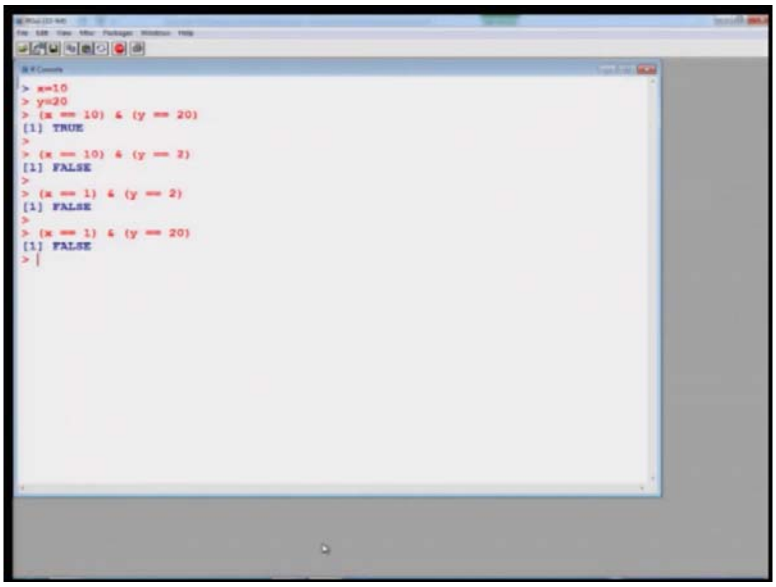


```
> x=5
> (x < 10) & (x > 2)
[1] TRUE
>
> x=40
> x
[1] 40
> (x < 10) & (x > 2)
[1] FALSE
> |
```

So, let me copy this command over here and you can see here that, I can say here x is equal to suppose 5 and if I say here this is here TRUE. And similarly if I try to take it here x equal to here 40; then you can see here x equal to 40, but the same command, this will become here FALSE.

Why? Because 40 is smaller than 10, but 40 is and 40 is greater than 2; this is the statement we are going to check, right. So, 40 is smaller than 10; no, this is FALSE, right. So, now, let me take here this example also with x equal to 10 and y equal to 20.

(Refer Slide Time: 19:02)

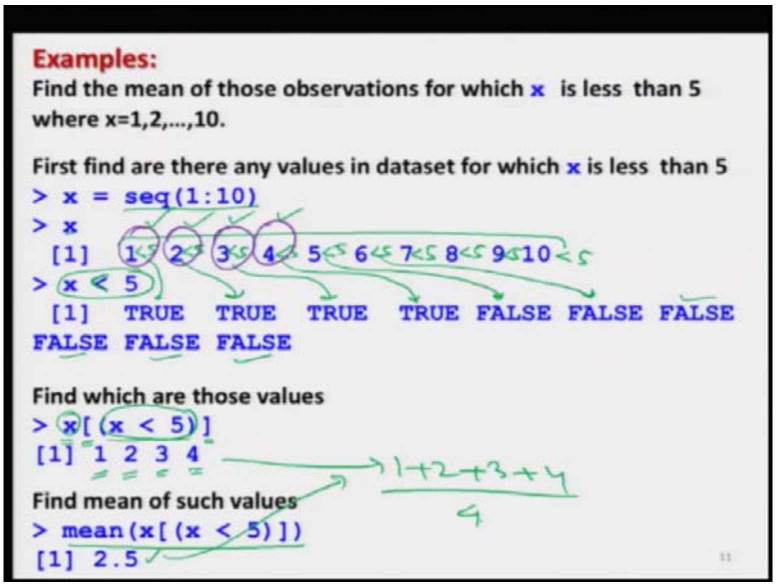


```
> x=10
> y=20
> (x == 10) & (y == 20)
[1] TRUE
>
> (x == 10) & (y == 2)
[1] FALSE
>
> (x == 1) & (y == 2)
[1] FALSE
>
> (x == 1) & (y == 20)
[1] FALSE
> |
```

So, I can say x equal to here 10, y equal to here 20 and if I try to write down this command over here; this will come out to be here TRUE. But on the other end if I try to see here y is equal to 2; if I try to change my syntax, this will come out to be FALSE.

And if I try to change the command once again that x equal to 1 and y equal to 2; this will again come out to be FALSE. And if I try to say here x equal to here 1 and say y equal to here 20; then even then it will come out to be FALSE, right. So, you can see here all these commands are working nicely on the R console also, ok.

(Refer Slide Time: 19:51)



Examples:

Find the mean of those observations for which x is less than 5 where x=1,2,...,10.

First find are there any values in dataset for which x is less than 5

```
> x = seq(1:10)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x < 5
 [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
FALSE FALSE FALSE
```

Find which are those values

```
> x[(x < 5)]
[1] 1 2 3 4
```

Find mean of such values

```
> mean(x[(x < 5)])
[1] 2.5
```

$$\frac{1+2+3+4}{4}$$

Now, let me try to address one thing more. Suppose I consider a data vector which has got values 1 to 10; 1, 2, 3, 4 up to 10, just like this one. So, this can be created by the command sequence seq 1 colon 10. Now, I have two objectives in my consideration; first objective is that I want to know which are the values out of this data set which are smaller than 5.

So, note that x contains the value 1 to 10. Now, when I try to say x less than 5, this is a logical operator. So, what it will do? It will try to operate this sign less than sign on each of the element; 1 is less than 5, 2 is less than 5, 3 is less than 5, 4 is less than 5, 5 is less than 5, 6 is less than 5, 7 is less than 5, 8 is less than 5, 9 is less than 5, and 10 is less than 5.

And now it will give me an output 1 is less than 5, answer is yes, it is TRUE; 2 less than 5 TRUE; 3 less than 5 TRUE; 4 less than 5 TRUE; 5 less than 5, no, this is FALSE. Similarly here 6 less than 5 FALSE; 7 less than 5 FALSE; 8 less than 5; 9 less than 5, and 10 less than 5 FALSE. So, this is the first objective.

So, now, I have got an answer that that, first four values are smaller than 5 and all other values are not smaller than 5. But now I want to know that, how many values are there which are smaller than 5. Means, I am not interested in the individual value; but I am interested in the cumulative number.

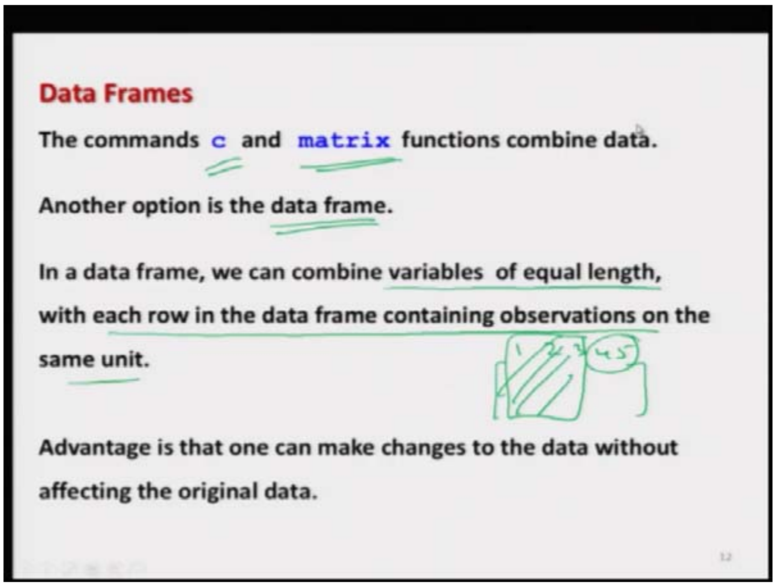
For example, in this case, if you try to see here that, there are four values 1, 2, 3 and here 4 which are smaller than 5. So, I want to know that how many values are there which are smaller than 5 and the answer should be 4 and what are those values, right. So, now I try to write down here like this. So, x a square bracket and then inside the square bracket I have to give the logical expression, x less than 5.

So, up to now you have handle only this part; but now I am adding here this x with the brackets is bracket sign a square bracket sign rather. So, now, it is giving me there are four values which are smaller than 5; which are these values 1, 2, 3, and 4. So, I have got those four values which are smaller than 5 and this confirms to my finding that there are four values 1, 2, 3 and 4 which are smaller than 5.

Now, means if you want to find out; suppose my question is this, I want to find out the arithmetic mean of those values which are smaller than 5. So, now I can operate here a command like this one; that mean of those values in the x vector which are smaller than 5. So, this will come out to be 2.5. Why 2.5? This will be 1 plus 2 plus 3 plus 4 divided by 4, right.

So, now you can imagine that when you are trying to deal with data and data sciences; these type of questions are very common. Whenever you are trying to count the number, whenever you are dealing with the frequency table; these numbers will be there and you need to handle and usually these numbers are going to be extremely large, you cannot count them with their your hand with manually. So, you have to depend on these tools, right.

(Refer Slide Time: 23:59)



Data Frames

The commands `c` and `matrix` functions combine data.

Another option is the data frame.

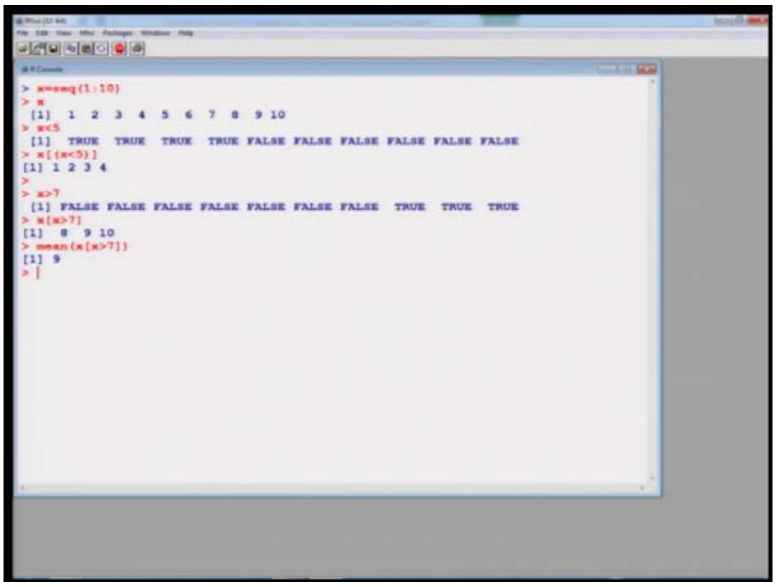
In a data frame, we can combine variables of equal length, with each row in the data frame containing observations on the same unit.

Advantage is that one can make changes to the data without affecting the original data.

12

So, before I go further, let me try to show you this thing on the R console also.

(Refer Slide Time: 24:08)



```
> x=seq(1:10)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> x<5
 [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
> x[(x<5)]
[1] 1 2 3 4
>
> x>7
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
> x[x>7]
[1]  8  9 10
> mean(x[x>7])
[1] 9
> |
```

So, you can see here, I will try to first create here a data vector seq 1 to 10; well you can also use, so 1 colon 10 that is not a big deal. So, now, you can see here x is the data vector from 1 to 10; if I say here, suppose if I say here x less than 5. So, now, you can see here, you are getting first four consecutive values are TRUE and remaining values are FALSE. Now, in case if I want that, how many values are here which are less than 5?

So, x less than 5, yeah, I am using the parenthesis only for the sake of clarity, right. So, you can see here these are the 1, 2, 3, 4. Similarly, if I try to say here x suppose greater than 7. So, you can see here these are the values which are; so you can see here only 8, 9, 10, these are the three values which are here TRUE. So, this is giving me those value and if I want to know, which are those values which are greater than 7. So, I can say write the expression inside the square bracket and this is here 8, 9, 10.

And yeah means if you want to find out the values; the mean of those values which are greater than 7. So, I can write down here mean and inside the expression I have to give you the logical expression. So, it is giving me the mean of 8 plus 9 plus 10 which is 9. Well, so you can see that it is not difficult; the only thing is this you have to just keep in your mind that what operation is going to give you what type of result.

Now, I try to address another aspect. Most of us have worked in data sheets and one of the popular software which we commonly use to handle the data sheets is the MS excel, Microsoft excel. I am using the name Microsoft excel because this is one of the most popular, there are several equivalent software. And when I say excel; that means the data has been arranged in rows and columns. If you try to recall in the excel sheet, you try to arrange the data in rows and in columns. So, there are row number 1, 2, 3, 4 and so on and the column numbers are a, b, c, d and you try to do different types of mathematical manipulation, statistical manipulation in that sheet.

So, similarly in R, if you want to handle that type of framework that can be done through the concept of data frame. So, you can imagine that whatever are those data sheets, they are equivalent to the concept data frame in R.

And data frame has one more capability that, it can handle the data sheet from different software also; it is not only excel, there are different software which will give you the outcome or the data which is arranged in rows and columns. So, data frame will help us in handling the, such data in the R software.

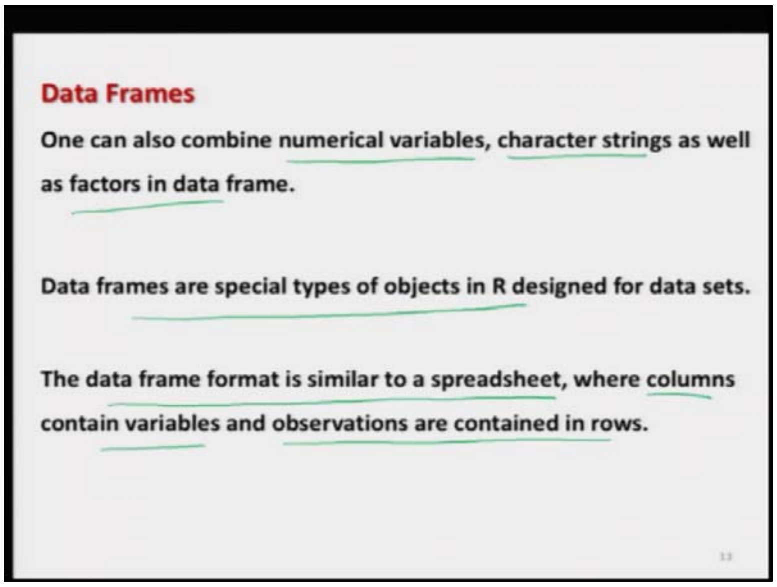
So, I will try to give you here a brief introduction to this data frame. Well, that is a very detailed topic and it has enormous capabilities; but I am giving you only a couple of them, which I expect that I will be using in the forthcoming lecture. So, now, you see one of the basic objectives in our R program was that, we always wanted to combine the data; for that we have used different types of command, c command, matrix command and another option is data frame.

The data frame can also combine the variables, combine the data; but there is a condition that, all the variable should have the data of equal length. And every row in the data frame contains the observation on the same unit, right.

And the advantage of converting the data into data frame is that, one can make different types of changes without affecting the original data. What does this mean? Suppose I have data here which has got here suppose 1, 2, 3, 4, 5 columns, right and I want to extract here a data only for first three columns.

So, it is possible mean; that means I do not have to it to remove or delete the 4rth and 5th column from the original file or the original data set, but I simply have to extract the information on 1, 2 and 3 and that will create another file for me, right.

(Refer Slide Time: 29:07)



Data Frames

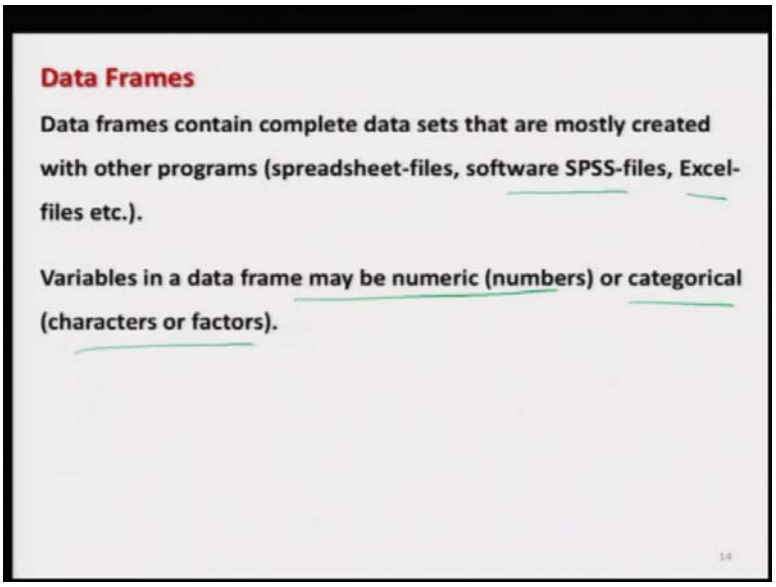
One can also combine numerical variables, character strings as well as factors in data frame.

Data frames are special types of objects in R designed for data sets.

The data frame format is similar to a spreadsheet, where columns contain variables and observations are contained in rows.

So, another advantage of data frame is that, this data frame can combine the numerical values, characteristic as well as the factors. So, different types of data can be combined under the same (Refer Time: 29:23). And these are actually, data frames are special types of objects in R, which are designed especially for handling the data set. And as I said, the data frame can be compared or it is equivalent to a spreadsheet, where the column contains the variables and observations are contained in row, right.

(Refer Slide Time: 29:45)



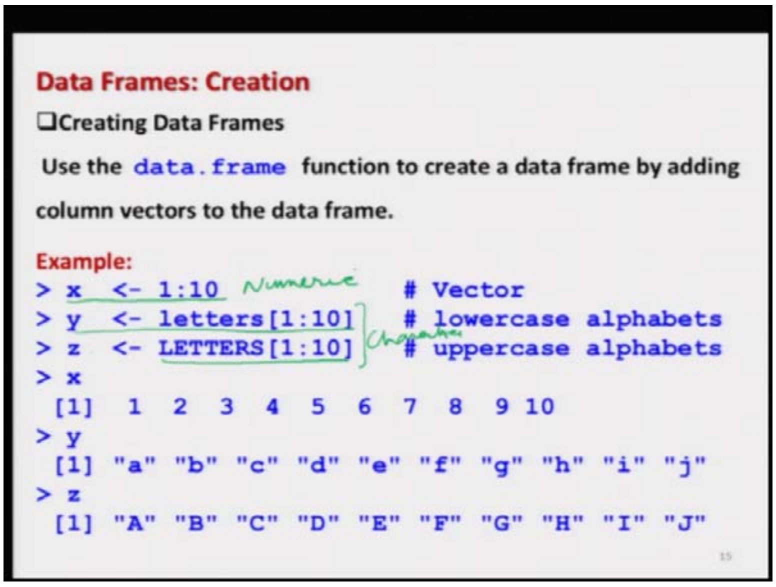
Data Frames

Data frames contain complete data sets that are mostly created with other programs (spreadsheet-files, software SPSS-files, Excel-files etc.).

Variables in a data frame may be numeric (numbers) or categorical (characters or factors).

So, these are the basic like concept and means another advantage is that, that data frames can also handle the spreadsheets type of files which are created by different software like SPSS, Excel etc. etc. And the and the variables in the data frame may be numeric, in some numbers or there can be some categorical variables, some characters, some factors. So, this is a very general framework in which one can handle that data.

(Refer Slide Time: 30:16)



Data Frames: Creation

❑Creating Data Frames

Use the `data.frame` function to create a data frame by adding column vectors to the data frame.

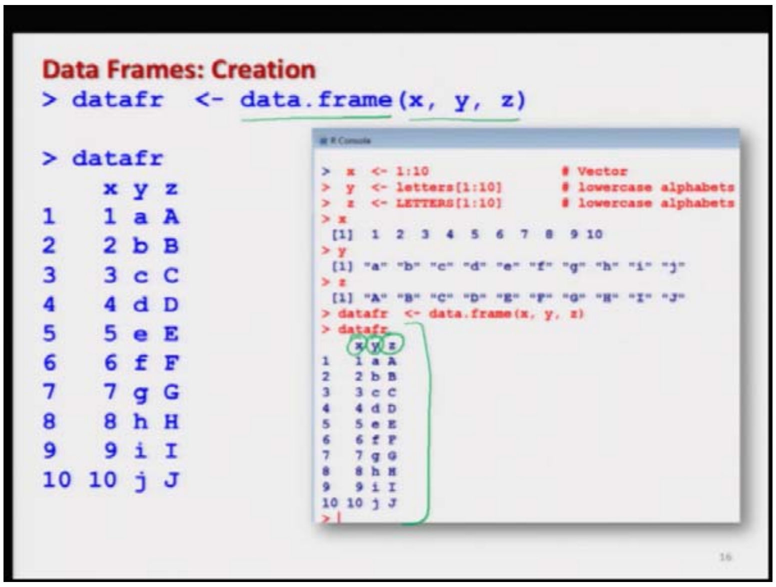
Example:

```
> x  <- 1:10  Numeric     # Vector
> y  <- letters[1:10]     # lowercase alphabets
> z  <- LETTERS[1:10]  Character  # uppercase alphabets
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> y
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
> z
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
```

So, now, let me take a simple example and if and I try to show you these various aspects of data frame. So, first I try to address; how can you create a data frame, if you have individual variables, right. So, I have taken here three variables x, y and z and you can see here x I am taking to be numeric, which is sequence of number from 1 to 10. And why I am trying to take first ten alphabets in say, small letters, lower case a, b, c, d up to j, and z I am trying to take another variable where first ten alphabets in capital letter that is upper case alphabets are considered.

So, you can see here that first variable is numeric whether the and remaining two variables are the characters. And one thing you can see here, I have intentionally taken the number of observations to be the same in every variable, because that is the need. If you do not do it; do not do it, then this creation of data frame will do something which is not desirable. I am not addressing that issue here, but you can try by choosing different types of sizes of the of data vectors, ok.

(Refer Slide Time: 31:47)



Data Frames: Creation

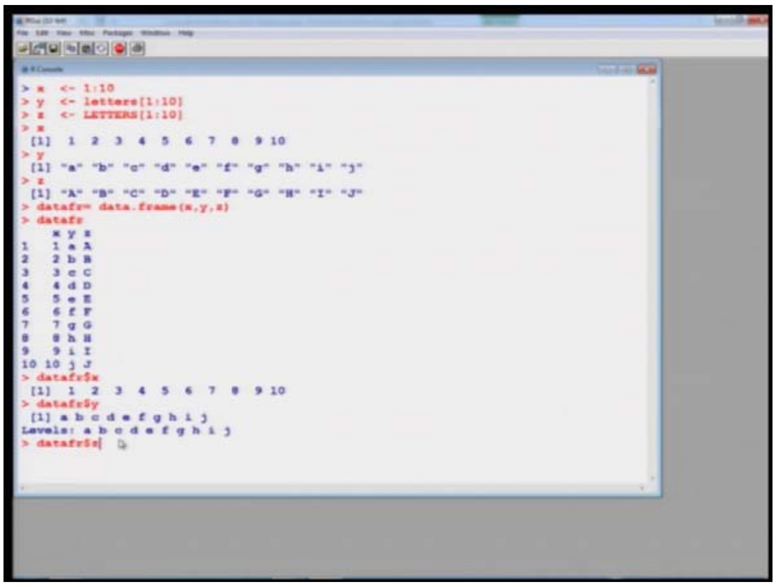
```
> datafr  <- data.frame(x, y, z)

> datafr
    x y z
1   1 a A
2   2 b B
3   3 c C
4   4 d D
5   5 e E
6   6 f F
7   7 g G
8   8 h H
9   9 i I
10 10 j J
```

```
>  x  <- 1:10              # Vector
>  y  <- letters[1:10]     # lowercase alphabets
>  z  <- LETTERS[1:10]     # lowercase alphabets
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> y
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
> z
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
> datafr  <- data.frame(x, y, z)
> datafr
    x y z
1   1 a A
2   2 b B
3   3 c C
4   4 d D
5   5 e E
6   6 f F
7   7 g G
8   8 h H
9   9 i I
10 10 j J
> |
```

16

So, now, I have these three variables and now I try to create data frame. So, the data frame can be created by the command data dot frame d a t a dot f r a m e and inside the parenthesis, you have to write down the names of the variable which are to be combined. So, you can see here, once I try to do it here; I get here a data frame like this, right. So, first let me try to show you this command on R console and then I will try to show you. So, I will try to copy here. So, I will try to copy here this thing, control c.

(Refer Slide Time: 32:35)

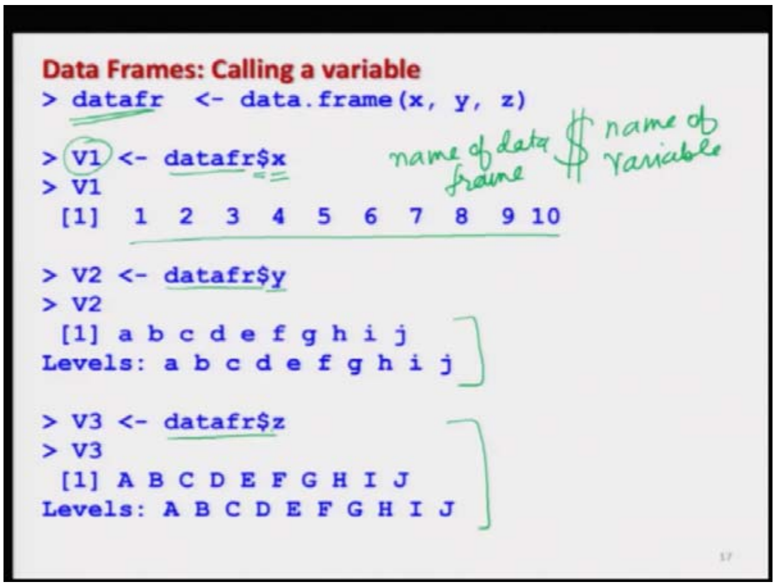


```
> x  <- 1:10
> y  <- letters[1:10]
> z  <- LETTERS[1:10]
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> y
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
> z
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
> datafr= data.frame(x,y,z)
> datafr
    x y z
1   1 a A
2   2 b B
3   3 c C
4   4 d D
5   5 e E
6   6 f F
7   7 g G
8   8 h H
9   9 i I
10 10 j J
> datafr$x
 [1]  1  2  3  4  5  6  7  8  9 10
> datafr$y
 [1] a b c d e f g h i j
Levels: a b c d e f g h i j
> datafr$z
```

So, this is my here x and then I am taking here y to be here like this, and here z is going to be the L E T T E RS. So, you can see here x is here like this, y here is like this and z here is like this. So, I can create here a data frame and let me store that data frame in the name datafr; just for the sake of remembrance and we can use it later on. So, the command is data dot frame and you can see here x comma y comma z.

And the data frame is created and if you want to see the structure of data frame, you can see here; the first variable all the values in the first variable are in the first column, all the values in the second variable are in the second column and all the values in the third variable are in the third column, ok.

(Refer Slide Time: 33:49)



Data Frames: Calling a variable

```
> datafr  <- data.frame(x, y, z)

> V1 <- datafr$x
> V1
 [1]  1  2  3  4  5  6  7  8  9 10

> V2 <- datafr$y
> V2
 [1] a b c d e f g h i j
Levels: a b c d e f g h i j

> V3 <- datafr$z
> V3
 [1] A B C D E F G H I J
Levels: A B C D E F G H I J
```

name of data frame
$
name of variable

So, now the next is aspect is that, suppose you are given a data frame which has got certain number of variable and you want to extract the information only on a particular variable. Suppose I take the same data set which I have just created and I move in the opposite direction; means first I took the variable and then I created the data frame.

Now, I am considering the data frame and I am trying to extract the variables. In order to extract the variables from a data set; first thing what you have to remember, you have to use the same name of the variable which is given in the data frame.

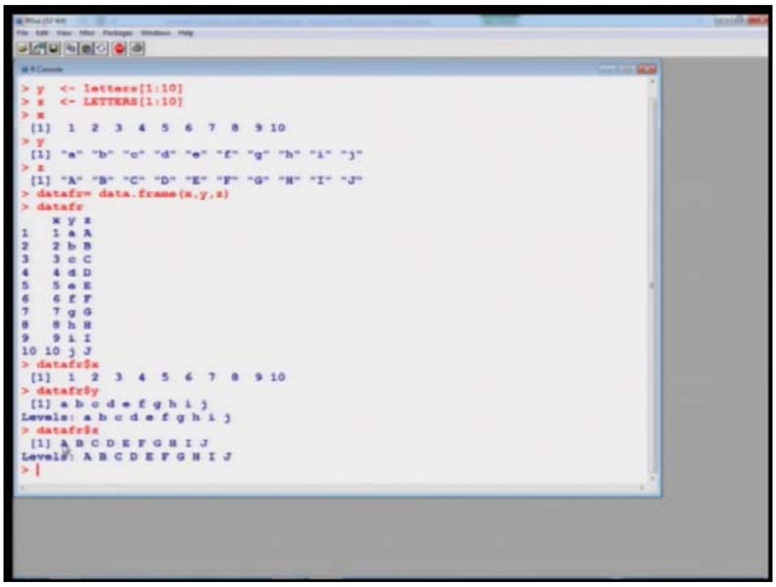
My three variables are here x, y and z that I have saved in my data frame, where you can see here; that these are my three variables here x y and here z. So, you have to look from the outcome; do not try to make a mistake by giving some other name, that will create some confusion, I am not discussing it here. But that is my sincere advice to you.

Then the rule is, you have to write down the name of data frame and you have to write down the $ and then you have to write down the name of variable. For example, here in this case my data frame name is datafr. So, this is datafr $ and here suppose I want to extract the information on the first variable x. So, this is the name of the variable x and you will see here that, this is my outcome which I have stored in a new variable name say V1.

Similarly, when I try to do the same thing for variable y; then datafr $ y. So, this will give me the information on the second variable y like this. And similarly for the third variable, I can write down datafr dot $ z and it will give me the same information. Now, let me try to show you this on the R console. So, you can see here, if I try to write down here data say here fr $ here x you can see here; you are getting the same information which you had entered here.

And similarly if you try to give here the information on y, you are getting the same data set a to j over here. Well these are the different levels; well I am not discussing here this concept. And if you want to have the information on the third variable z; from this data frame you can see here, this is giving you this to be data set which is same as this data set over here, right ok.

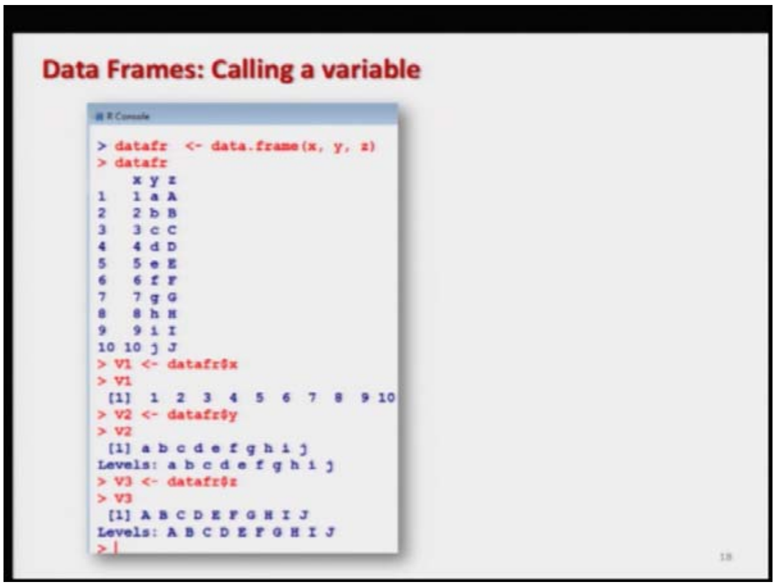
(Refer Slide Time: 36:37)



```
> y  <- letters[1:10]
> z  <- LETTERS[1:10]
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> y
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
> z
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"
> datafr= data.frame(x,y,z)
> datafr
    x y z
1   1 a A
2   2 b B
3   3 c C
4   4 d D
5   5 e E
6   6 f F
7   7 g G
8   8 h H
9   9 i I
10 10 j J
> datafr$x
 [1]  1  2  3  4  5  6  7  8  9 10
> datafr$y
 [1] a b c d e f g h i j
Levels: a b c d e f g h i j
> datafr$z
 [1] A B C D E F G H I J
Levels: A B C D E F G H I J
> |
```

So, well this is the screenshot whatever I shown you here, so that you be confident that this is working.

(Refer Slide Time: 36:47)



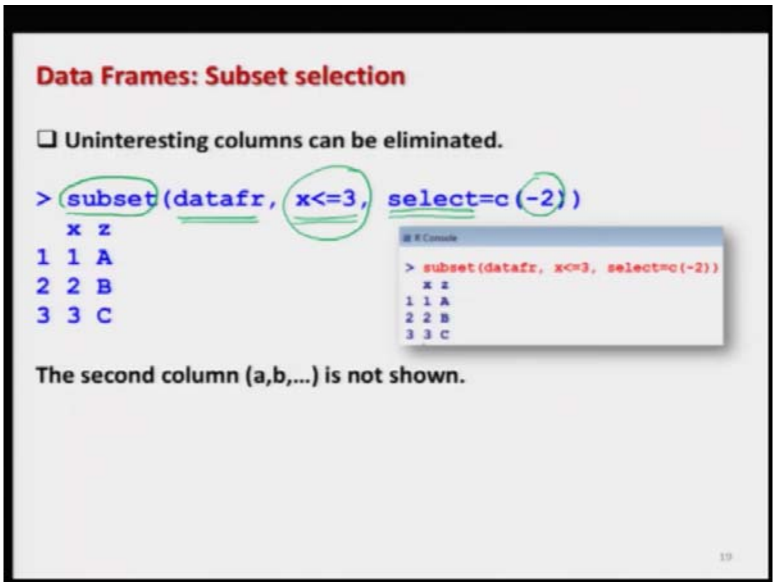
Data Frames: Calling a variable

```
> datafr  <- data.frame(x, y, z)
> datafr
    x y z
1   1 a A
2   2 b B
3   3 c C
4   4 d D
5   5 e E
6   6 f F
7   7 g G
8   8 h H
9   9 i I
10 10 j J
> V1 <- datafr$x
> V1
 [1]  1  2  3  4  5  6  7  8  9 10
> V2 <- datafr$y
> V2
 [1] a b c d e f g h i j
Levels: a b c d e f g h i j
> V3 <- datafr$z
> V3
 [1] A B C D E F G H I J
Levels: A B C D E F G H I J
> |
```

18

Now, I try to give you just two more simple operation.

(Refer Slide Time: 36:54)



Data Frames: Subset selection

- Uninteresting columns can be eliminated.

```
> subset(datafr, x<=3, select=c(-2))
  x z
1 1 A
2 2 B
3 3 C
```

```
> subset(datafr, x<=3, select=c(-2))
  x z
1 1 A
2 2 B
3 3 C
```

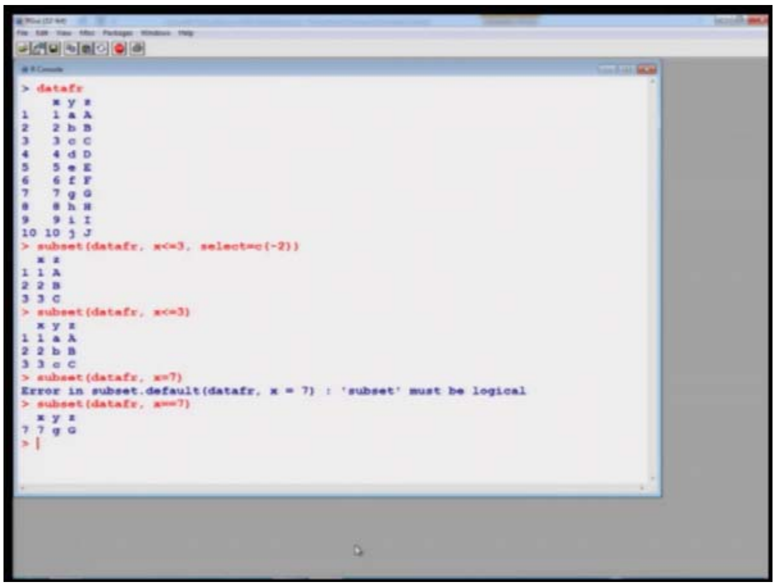
The second column (a,b,...) is not shown.

Suppose I want to select a subset of a data set; means I have a bigger data set and I want to choose a subset. And this is possible suppose you are getting a data set on the age of the people and there are one million or one billion observations and suppose you want to extract the information on those candidates, on those person whose ages are suppose less than say 5 years.

So, how to get it done? And you want to do your statistical computation on those people only. So, in order to do this thing without changing the original data set, we have a command here subset which can provide the subset of the data frame. So, the command here is s u b s e t subset and inside the parenthesis; you have to write down the name of the data frame. And you have to give your condition that, what is your criteria by which you want to select the values.

And suppose you want to say that, from the data set you also remove the second column; so for that you have a command here select and you and if you want to remove it, just use the negative sign. So, c minus 2 inside the parenthesis that will remove the second column and it will give you the data set corresponding to which x is less than or equal to 3. So, let me try to show you here, show this thing on the R console also. So, I try to copy this command over here and I try to use here the same data frame.

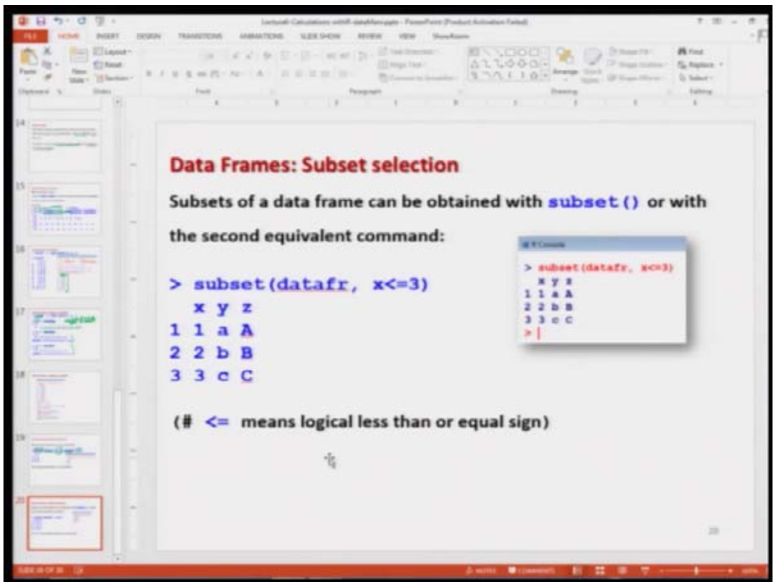
(Refer Slide Time: 38:47)



```
> datafr
    x y z
1   1 a A
2   2 b B
3   3 c C
4   4 d D
5   5 e E
6   6 f F
7   7 g G
8   8 h H
9   9 i I
10 10 j J
> subset(datafr, x<=3, select=c(-2))
  x z
1 1 A
2 2 B
3 3 C
> subset(datafr, x<=3)
  x y z
1 1 a A
2 2 b B
3 3 c C
> subset(datafr, x=7)
Error in subset.default(datafr, x = 7) : 'subset' must be logical
> subset(datafr, x==7)
  x y z
7 7 g G
> |
```

So, you can see here, I can clear; but I can call the data frame, so that you can see here. Now, I tried, what I am asking? I am asking that give me a data set for which x is less than or equal to 3 and please remove the second column, right. So, you can now see here the outcome. So, this is only those values which are x less than 3 and the second column is removed, right.

(Refer Slide Time: 39:17)



Data Frames: Subset selection

Subsets of a data frame can be obtained with subset() or with the second equivalent command:

```
> subset(datafr, x<=3)
  x y z
1 1 a A
2 2 b B
3 3 c C
```

(# <= means logical less than or equal sign)

```
> subset(datafr, x<=3)
  x y z
1 1 a A
2 2 b B
3 3 c C
> |
```

So, all these operations can be done without much trouble in R sub and similarly if you do not want to remove anything and if you simply want to have a data set where x is less than 3.

So, you just remove this select command and you can see here this will give you the entire data set all the three variables corresponding to which x is smaller than or equal to 3. Similarly, you can take here say any other thing also; say here see if I say x equal to here 7, you can see here, why this is not happening? Do you know why this is not happening?

Because you have to give here two equality sign; because this is the logical operator, this is not a mathematical operator. So, you can see here this has given you the 7th row, right. So, now, I stop here, I have given you a brief quick review of the data frame that is a very detailed topic in R, right. But still I would say that, I am here only for some time, 30 minute, 35 minutes, 40 minutes; but you have more time, so you try to look into the books, try to take some examples and try to create the problems yourself.

I am sure that you must have heard some situations where you need to use such operations. So, unless and until you become a data scientist, try to consider those hypothetical situation; try to assume that yes, you have become a data scientist and you have a problem before you, how you are going to handle it.

Remember one thing, all this operation they are heavily dependent on your logic; then next question come how will you get the logic? Believe me on this platform I would like to acknowledge my teacher, who explained me how to get the logic; I mean that story goes to 1992 when they started, when we started computers in India.

So, there was a teacher who used to help me out and he was teaching us how to do programming that was something like a fairy tale for us. And one day I remember that, I was practically crying; that sir is always saying that get a logic, get a logic, sir from where I will get the logic. So, I went to my teacher and he said Shalabh you do not worry, you just practice is, logic will come and is and the name of the teacher which I can recall was Shabir sir.

So, thank you Shabir sir and I would acknowledge the same thing to you all; just practice, logic will come, more you practice, more logic will come. And once you get the good logic, your programming will become very fast; the time will become less and believe me the same thing can be solved by different people in different ways, some programs may take a very long time to get the same outcome which can be obtained in a much less time in a different outcome.

So, you develop your logic, you practice and I will see you in the next lecture. Till then good bye.