

Essentials on Data Science with R Software - 1
Professor Shalabh
Department of Mathematics & Statistics
Indian Institute of Technology Kanpur
Lecture No. 07
Logical Operators and Selection of Sample

Hello friends, welcome to the course Essentials of Data Sciences with R software 1, in which we are trying to understand the topics of probability theory and statistical inference. So, now you can see that in the last couple of lectures we have understood some basic commands related to the R software. So, in this lecture we will continue with the same topic but after this we will come to our statistics part.

In this lecture I am going to take up two topics, one is the logical operators and second is the sample collection. What you mean by samples collection? Means when we have got a finite population from where we have to draw a sample then how to get it done. But remember one thing when we are trying to talk of the choice of sample or the samples collection we have two options, that we are trying to draw a sample from an infinite size of population or a finite size of population.

So, here in this lecture we are trying to concentrate on the selection of the sample when the sample is drawn from a finite population, the population is known to us. Well, so let us begin our lecture and try to understand these topics. And in case if you ask me that how this logical operators are going to be used, this point I would like to clarify here before I move forward.

Well, we are going to use this conditional operations, we are going to compute different types of quantities which are condition on something that if this happened only then this happens. So, in those types of operation these logical operators are going to help us and that is why I have chosen this topic so that I can give you a brief idea about these operators, so that when we are trying to use them, then it is not difficult for you. So, let us begin our lecture. So, we know that we have two types of operators, 1 are mathematical operators and say another are logical operators.

(Refer Slide Time: 2:36)

Logical Operators and Comparisons:

- The cities and households are categorized in the data as 1, 2,...
- We want to find the mean of income those households which are in the cities coded as 1.
- We want to find the mean of income of the households in cities coded as 1 having household size more than 3. *→ Right/Wrong*
- Less than, more than and not equal to are the logical operations, not mathematical operations. *5-100 } less than
5-50 }*

And what is the difference between the two? The mathematical operators are like addition, subtraction, multiplication, divisions, etc. And logical operators are like more than, not equal to, smaller than, less than, etc. So, the logical operators makes only the comparison. For example, if I say 5 is smaller than 6 or not, so we know that 5 is smaller than 6. So, I am simply trying to compare whether it is smaller than or more than. Means I am not interested in the magnitude.

For example, if I say 5 is smaller than 10 or say 5 is smaller than 100, means if you try to use here a mathematical operator this will be like as 5 minus 100 but if you or 5 minus 50. So, their magnitudes are going to be different but when we are trying to use the logical operator that is only going to be, there is only going to be only one answer that they are less than that is all that 5 is less than 50 or say 100.

So, now in order to explain you this logical operators, let me try to take here a very simple example. Suppose, we have got a data set in which the cities and the household, household means you can take it as a families, those cities and households are categorized. That means, for example, if I take cities like say Kanpur, Kolkata, Delhi, Mumbai, etc., they are suppose coded as 1, 2, 3, 4. And after that in every city there are people, there are families, there are household and they have got different types of income and we categorize them as lower income, middle income or say higher income groups.

So, all those families which are coming under the lower income group they will be indicated by the number 1, all those families which are under the middle income group they will be indicated by number 2, and similarly all the families which are coming under the higher income group they will be indicated by number 3. Well, this 1, 2 and 3 these are only the indicator values, they are simply indicate the absence and presence of the characteristic, they are not trying to compare the magnitude.

So, now suppose we want to find out the average income or mean of the income of those households which are in the cities which are coded as 1. So, now what I have to do? First I have to find out from the household or the families which have been coded as 1 and then I have to take out those values of the data sets. Because data set will be having the codings 1, 2, as well as 3, so I want to take them out. And then we try to find out the values or the arithmetic mean of only those values which are corresponding to the coding 1.

And similarly, if I want to find out the mean of the income of those households in cities which are coded as 1 and having household size more than 3. Suppose, there is another variable in which you are also trying to give the family size, so I want to take that variable also under consideration. But here in this case, I am just interested in knowing that whether household size is more than 3 is right or wrong, that is all, I am not interested whether the size is 4, 5, 6 or 7. So, under these types of cases, this logical operators are going to help us.

So, all this operation like less than, more than or not equal to, they are the logical operators and they are not the mathematical operators. They will conduct only the logical operations not the mathematical operations.

(Refer Slide Time: 6:36)

Logical Operators and Comparisons
The following table shows the operations and functions for logical comparisons (True or False).

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)

TRUE
True

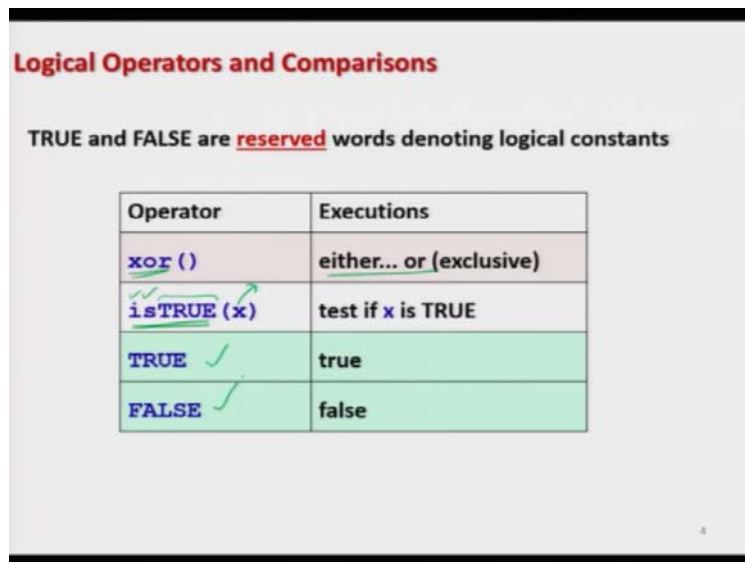
= math.
== logical.

So, when we come to the list of logical operators then I have given here the list but before that you have to understand there are two words which are TRUE and FALSE both are written in the uppercase alphabets that is the capital English letters, they are the reserved words. TRUE mean true and FALSE means false, they cannot be used for anything else. But, definitely remember 1 thing that this capital 'TRUE' and 'True' like this 'T' but all 'r u and e' are in small letter or any combination that is not the logical operator, only this capital letters TRUE and capital letters FALSE, they are the reserved word. So, they have to be used only as a logical operators, they cannot be assigned to any other variable.

And similarly, we have a operator here greater than which is indicated by the greater than sign that we know. Similarly, you have another symbol what you know that 1 is greater and another is greater than or equal to which you write like this. So, this symbol greater than or equal to is written here as greater than and equal to sign. And similarly, the operator less than is written here as say only less than and less than or equal to that is written as less than and equal to. And similarly, if you want to write down the two equality sign that is going to give you exactly equal to. So, this equal to sign is a mathematical operator but double equal to sign this is a logical operator that is what you have to keep in mind.

And similarly, when I say that the symbol like not equal to this is indicated by exclamation sign and equal to sign, so exclamation sign means not. And in case if you simply want to have the negation, then it is indicated only by the exclamation sign.

(Refer Slide Time: 8:37)



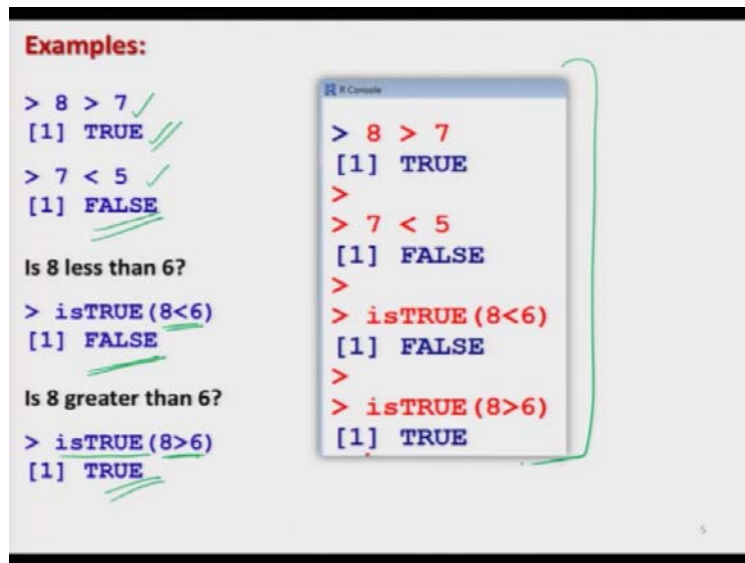
The slide is titled "Logical Operators and Comparisons" in red. Below the title, it states "TRUE and FALSE are reserved words denoting logical constants". A table with two columns, "Operator" and "Executions", lists the following:

Operator	Executions
<code>xor ()</code>	either... or (exclusive)
<code>isTRUE (x)</code>	test if x is TRUE
<code>TRUE</code> ✓	true
<code>FALSE</code> ✓	false

So, let me try to give you some simple example so that I can show you, but beside those things if you want to make sort of a combination of the expression then we have 1 more here operator xor which is used as either or. And similarly, if you want to know about some expression that if the expression is TRUE or say FALSE, then we have here a statement or a command 'isTRUE' is true but you have to be careful that in here 'i and s' they are in lowercase alphabets and 'TRUE' they are in upper case alphabet.

So, and if you try to write down here something inside the parenthesis it will check whether this statement whatever is written inside the parenthesis is true or not. And similarly, and this TRUE and FALSE they are the reserved words that we know.

(Refer Slide Time: 9:23)



Now, I try to take here some examples and try to show you that how these things are working. So, suppose if I write down here statement like 8 greater than 7, do you think that is it correct? The answer is yes, 8 is bigger than 7, so answer will come out to be TRUE. Similarly, if you try to write down here 7 less than 5, what do you think? Whether 7 is smaller than 5 or 7 is greater than 5? So, this is FALSE. So, you will get here an answer like FALSE. Similarly, if you want to check a statement whether 8 less than 6, is this TRUE or say FALSE? The answer is FALSE. So, if you write down here this statement it will give you an answer FALSE.

And similarly, if you want to check whether 8 is greater than 6 or not, so try to write down here `isTRUE(8 > 6)` and it will give you the answer TRUE. And this is the screenshot here what is going to happen. So, now if you try to understand the utility of these things, when you are working with data sciences you are working with huge data sets. And suppose, I take a very simple example, suppose there is a shopping website which has got the items which are cheap, which are expensive, which are costly and all sorts of things are there.

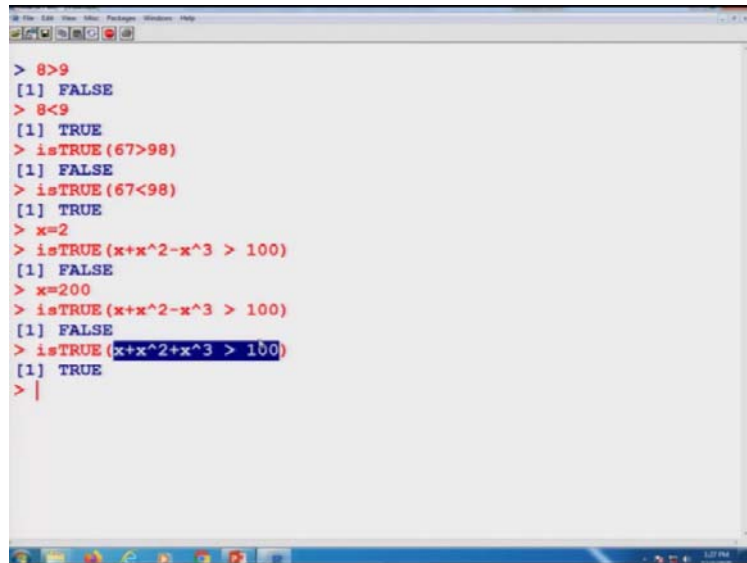
Now, definitely if you remember or if you observe whenever people are trying to create a login ID for the shopping, usually they will ask, what is your annual income? Now, suppose some customer wants to buy speaker, simply audio speakers. Now, the cost of a speaker may be extremely low, average, high or it can be very high. Now, the person is looking for speakers and

the person does not buy but he or she leaves the website. Now, what the data science is going to do there to help the shopping website, do not you think that in case if somebody is trying to search for the speaker on a website that is, that clearly indicates that the person is interested in buying a speaker.

Well, if the person is coming there for more than one time or a couple of times then that will strengthen our belief that the person really wants to buy the speaker. And suppose, person is trying to look for some speakers and possibly the person could not find the speaker which can fit into the pocket, whether they are very cheap or they are very expensive suppose. So now, as a data scientist you can go to the profile of that customer, try to look into the income bracket and according to that income bracket, can you write a program which can search for those speakers which can fit with the pocket of the customer and then after that an offer or email or a reminder or a request can be sent to the customer, that please come to our this site and try to look at we have some more options.

In case if the person is coming there but still the person is not buying it possibly you can provide some coupon code that can give the person some discount possibly the person may buy. Under those types of operation how will you sort out that which customer has to be given what type of offer depending on its income bracket. These logical operators are going to help there. You can simply check whether the income is greater than the price of the speaker or similar type of comparison you can make. And if that comes out to be true, then you try to give this offer and false try to give that offer. That is how these logical operators are going to help you. So, now let us try to take some more example and try to show you, but before that let me try to show it on the R console.

(Refer Slide Time: 14:25)



```
> 8>9
[1] FALSE
> 8<9
[1] TRUE
> isTRUE(67>98)
[1] FALSE
> isTRUE(67<98)
[1] TRUE
> x=2
> isTRUE(x+x^2-x^3 > 100)
[1] FALSE
> x=200
> isTRUE(x+x^2-x^3 > 100)
[1] FALSE
> isTRUE(x+x^2+x^3 > 100)
[1] TRUE
> |
```

Suppose, if I write 8 greater than 9 this will come out to be FALSE. But in case if I write 8 less than 9 that will come out to be TRUE. And similarly, if you want to find out isTRUE say 67 is greater than 98 this will come out to be FALSE. But, in case if you want to find out whether this 67 is smaller than 98. Is this a statement TRUE? It is coming out to be TRUE. And similarly, means if you want to write down here suppose I, if I try to take here x equal to here 2 and now I try to write down this statement is true x plus x square minus x cube this is greater than suppose 100. Suppose, I want to check this condition. So, x is being taken here as x equal to 2 and you can see here 2 plus 4 minus 8 that is not greater than 100.

But, in case if you try to take care x equal to 200, then what will happen that if you try to execute this condition this is again FALSE. But, in case if you try to change this condition to be x plus x square plus x cube then you will see that this comes out to be TRUE obviously 200 plus 200 square by 200 cube that is obviously going to be greater than 100, so you can imagine that you are trying to give here some condition for the customer to determine whether the income of the customer is close to the cost of the speaker or not.

(Refer Slide Time: 16:07)

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

So now, similarly I try to take here some more example and try to show you here. Similarly, we have here say here 1 more operator say &&, that you know but I am simply trying to give you here a quick revision. So, this && means & operator means both the conditions are going to be satisfied. Suppose, if I write down here x equal to 5 and if I write there here x less than 10 and x greater than 2. So, obviously 5 is less than 10 and 5 is greater than 2, 5 which is smaller than 10, is this true? Yes, TRUE and 5 is greater than 2, yes this is TRUE.

So, TRUE and TRUE will become here TRUE and it will give you answer here TRUE. So, this is trying to combine the different statements using the AND command. And here you can see this is the screenshot of the same outcome.

(Refer Slide Time: 16:50)

Examples:

```
> x <- 5
```

Is x less than 10 or x is greater than 5 ?

```
> (x < 10) || (x > 5) # || means OR
```

[1] TRUE

Is x greater than 10 or x is greater than 5 ?

```
> (x > 10) || (x > 5)
```

[1] FALSE

Handwritten notes: ① ②, $5 < 10$ TRUE, $5 > 5$ FALSE, OR

```
R 64-bit
> (x < 10) || (x > 5)
[1] TRUE
> (x > 10) || (x > 5)
[1] FALSE
>
```

And similarly, if you try to take here some more examples, this symbol, 2 vertical lines that is indicating the OR operator 'OR' means either this is TRUE or that is TRUE. So, if you try to write down here x equal to 5 and then I am trying to check here two condition that is x less than 10 or x is greater than 5. So, I try to write down here x greater than 10 and 2 vertical lines which are OR x greater than 5. What does this mean? Even if any of the condition out of condition number 1 or condition number 2 whichever is TRUE if any of them is true, then this will give us an answer TRUE.

So, I am trying to write down here 5 less than 10 and say 5 greater than 5. So, you can see here 5 less than 10 is TRUE and 5 greater than 5 this is FALSE. But, in case if you try to write down here TRUE or FALSE then it is here TRUE. So, this double vertical line or the 'o r' or the or operator that is going to work even if any of the condition is going to be satisfied. Whereas in the case of && and or the & operator the condition is going to be satisfied only when condition 1 and condition 2 both are satisfied.

(Refer Slide Time: 17:59)

Examples:

```
> x <- 5
```

Is x less than 10 or x is greater than 5 ?

```
> (x < 10) || (x > 5) # || means OR
```

[1] TRUE

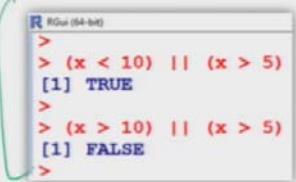
Handwritten notes: 5 < 10 TRUE OR 5 > 5 FALSE TRUE OF FALSE

Is x greater than 10 or x is greater than 5 ?

```
> (x > 10) || (x > 5)
```

[1] FALSE

Handwritten notes: 5 > 10 or 5 > 5 False False



Similarly, if I try to test whether x is greater than 10 or x is greater than 5. So, this condition I can write down here x greater than 10 and OR that means 2 vertical lines x greater than 5 this will come out to be here FALSE. Why? Because 5 is greater than 10 or 5 is greater than 5. So, this is here FALSE and 5 is greater than 10, no, this is also FALSE. So, FALSE and FALSE is FALSE. And this is here the screenshot of the same operation.

(Refer Slide Time: 18:28)

Examples:

```
> x = 10 ✓
> y = 20 ✓
```

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

```
> (x == 10) & (y == 20) # == means exactly equal to
```

[1] TRUE

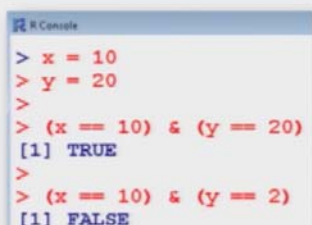
Handwritten notes: 10 == 10 T, 20 == 20 T

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

```
> (x == 10) & (y == 2)
```

[1] FALSE

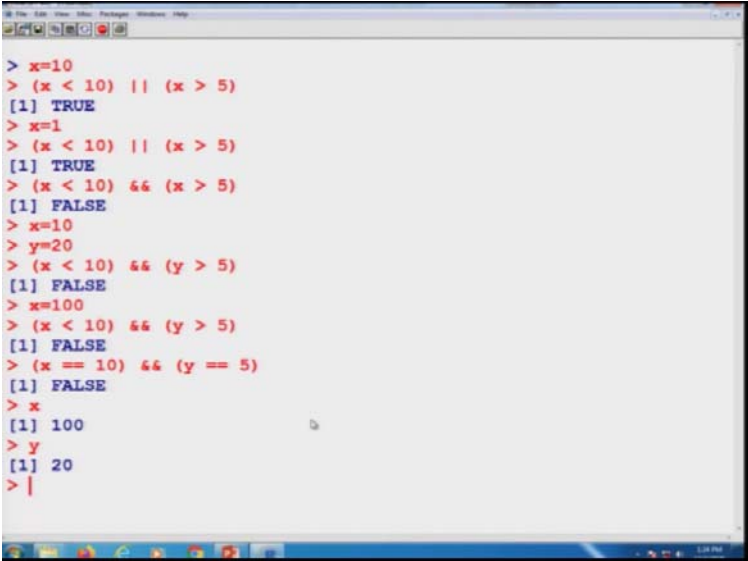
Handwritten notes: 10 == 10 T, 20 == 2 F



Suppose, if I take x equal to 10 and y equal to 20 then I want to check whether x is equal to 10 and y is equal to 20. So, I try to use here this two equality sign which is a logical equality signs. So, I want to check whether the value of here x which is given here say 10 this is exactly equal to 10 and the value of y which is given here say 20 is equal to 20. So, yes, that is TRUE. So, this is also TRUE, this is also TRUE and so the answer comes out to be here TRUE. And similarly if I want to test is x equal to 10 and y equal to 2 then in that case the value of x which you have given here, this is 10, so 10 is equal to 10, this is TRUE, but 20 is equal to 2 this is FALSE. So, this condition will come out to be here FALSE.

So, you can see here this is how you can combine different types of condition and you can do different types of operation. But before I go further let me try to show you these things on the R console also.

(Refer Slide Time: 19:39)



```
> x=10
> (x < 10) || (x > 5)
[1] TRUE
> x=1
> (x < 10) || (x > 5)
[1] TRUE
> (x < 10) && (x > 5)
[1] FALSE
> x=10
> y=20
> (x < 10) && (y > 5)
[1] FALSE
> x=100
> (x < 10) && (y > 5)
[1] FALSE
> (x == 10) && (y == 5)
[1] FALSE
> x
[1] 100
> y
[1] 20
> |
```

So, let me try to just copy this thing so that some time is saved, I try to clear the screen by pressing ctrl l and suppose if I try to take here x equal to 10 and if I try to write down here the condition x is less than 10 and x greater than 5, this condition comes out to be here TRUE, because 10 is greater than 5. But in case if I try to take x equal to here 1, then what will happen? You can see here still the, this condition is 1 because you are trying to take here or. So, if this

condition is TRUE that 1 is smaller than 10, this number will or this condition will always be satisfied.

But in case if you try to take here this & operator here, you can see here this condition comes out to be here FALSE, because x equal to here 1, so 1 is less than 10. Yes, but 1 is greater than 5, no. And similarly, if I try to take here x equal to 10 and suppose y equal to 20, this condition also can be checked with the those thing that if I try to write down here x is less than 10 and y is greater than 5, you can see here. For x equal to 10, 10 is not greater than 10 and 20 is greater than 5. So, this condition is FALSE.

So, this is how you can check different types of thing and also in case if I try to take here one more condition, suppose, if I make it here x equal to 100 and if I try to tell the same condition that 100 is less than 10 and twenty is greater than 5. Yes, so what do we expect? The condition is going to be FALSE. But in case if I try to choose the double equal to sign, that means I want to check whether x is exactly equal to 10 and y is exactly equal to 5, the answer is coming out to be FALSE. Why? Because the value of x here is 100 and the value of y here is 20. So, none of them is equal to 10 over 5 respectively. So, you can see here that this logical operators are really going to help you in your data science when you are trying to make different types of comparisons.

(Refer Slide Time: 21:55)

Simple Random Sampling:
Simple random sampling (SRS) is a method of selection of a sample comprising of n number of sampling units from the population having N number of units such that every sampling unit has an equal chance of being chosen.

Handwritten notes:
100 balls
 $N=100$
Population size
20 balls
 $n = \text{sample size} = 20$

Now, I try to come to another topic which is about how to draw a sample. What do you mean by sample? You see, in statistics whenever we are trying to conduct an analysis, that is always conducted on the basis of a small sample. What is the sample? Sample is only a fraction of the population. For example, if I say I want to determine the heights of the student of a class. And suppose there are 500 students in the class. So, what will I do? I will try to select some of the students from there and then I will try to note down their heights and then I will try to apply different statistical tool on that data which I have selected. So, that is actually our sample.

Well, when you are trying to take the sample, there are certain condition that the sample has to be truly representative, means all the characteristics which are present in the population they have to be present in the sample also. For example, if the class has girls and boys both, then you expect that when you are trying to take the sample, the sample should also contain the boys and girls.

In case if out of 500 students you are trying to choose 20 student and suppose all are girls and all are boys, then that is not really going to reflect because if you try to look at the sample which has got only girls that will indicate that possibly there are no boys in the class and vice versa also. So, that is important that the sample should have the girls and boys in a proportionate number that will be our representative sample.

Now, the question is this, how to draw the sample? And it is a very important, why? Because your entire decision sciences, entire statistic that is going to based on this thing. Now, when we are trying to do such a sampling we have two options, the population from where we are trying to draw the sample that is finite or that is very large, very huge. When it is very huge, then we symbolically call it as infinite population that is very large. So, once it is very large you can choose any type of sample from anywhere from that population.

But here in this topic I want to give you an idea that if you want to select a sample from a finite population, then how can it be done using the R software. And basically, we have a command here sample. But the sample is actually trying to, try to draw the sample using that technique of simple random sampling. Actually there are various types of sampling techniques which help us in drawing with different types of representative sample from various types of populations.

But here my objective is not really to give you an idea of the sampling theory. But I just want to give you the command by which we can draw a simple random sample. So, the simple random sample is one of the most fundamental basic technique in which we try to draw a sample from a population such that all the sampling units are drawn independently and they have equal chance of being collected.

What does this mean? Means if you try to put suppose 100 balls in a box and if you simply try to take out the ball only from the upper surface. Do you think that the balls which are inside the, towards the bottom of the box, do they have any probability of collection? It is very low, because you will never try to put your hands so much into depth inside the box. You will simply be taking it outside from the surface of the box. So, this is not a simple random sample but what we want that each of the ball which is inside the box, each of the ball should have equal chance of being selected, equal chance of being coming out of the box.

So, for that what we can do? We can take the box, we can shake it heavily so that we have no idea absolutely that which ball is where. And then I can close my eyes and can put my hand inside the box and I can take out 1 ball. So, before the drawing of the ball I do not know which of the ball is going to come out. And once again then I will try to shake the box heavily so that all the balls are dislocated and we have no idea of where is which ball. Then again I can put my hand and I can take a ball outside. So, this is basically that technique of simple random sampling.

Now, when we are trying to do it then I have two options, when I have taken a ball outside then either I can put it back inside the box and then I try to draw the second ball or I do not keep it inside the box and I try to draw the second ball. So, that is the idea of the sample or sampling with and without replacement. So, when we are trying to draw a ball and we are trying to keep it inside the box and then trying to draw the second ball, this is sampling with replacement. And when we are trying to draw the ball and we are not keeping it back inside the box and then we draw the second unit then this is called as sampling without replacement.

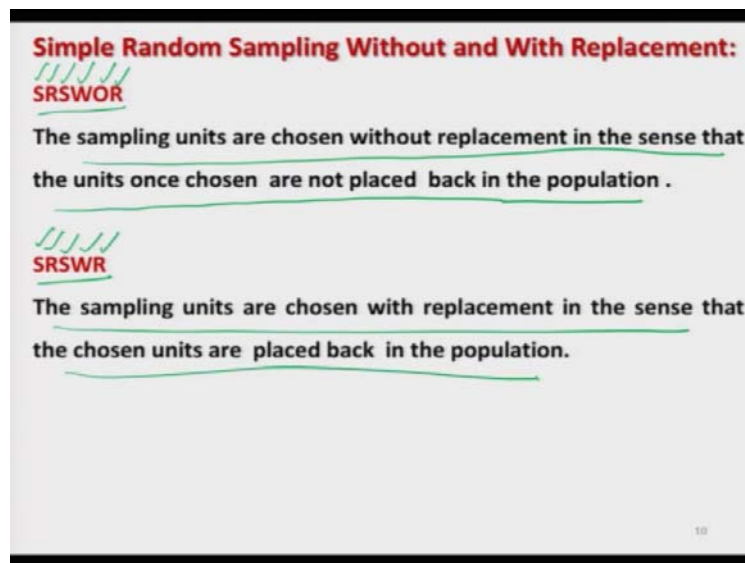
But in both the cases in the setup of simple random sampling the probability of collection of each and every unit will always be the same and at every draw that is the advantage. Well, how this happened, what is the theoretical construct, what is the background, this is out of the purview of

this course. So, I will not discuss it here, but definitely you can believe on me that means one can show theoretically and using the concept of statistic that this is going to happen.

So, my objective here is to just very quickly give you the definition of simple random sampling. What is simple random sampling with replacement, without replacement? And now the main objective is how are you going to draw such a sample? So, let us come back to our slides. So, you see, the first question comes here what is a simple random sampling. So, the simple random sampling here is a method of selection of a sample comprising of small n number of sampling units from the population having N number of the units such that every sampling unit has an equal chance of being chosen.

For example, if I try to take here a box and suppose there are 100 balls. So, my N becomes here 100. And now I want to draw here, suppose here 20 balls, so small n is the sample size and this is equal to here 20 and N will be here the population size.

(Refer Slide Time: 29:41)



And when we are trying to execute this simple random sampling we have two options, simple random sampling without replacement and simple random sampling with replacement. So, in the case of simple random sampling without replacement we try to indicate by 'SRSWOR' which is Simple Random Sampling Without Replacement. In this case the sampling units are chosen without replacement in the sense that the units once chosen are not placed back in the population.

And the second technique is 'SRSWR' Simple Random Sampling With Replacement. So, in this case the sampling units are chosen with replacement in the sense that the chosen units are placed back in the population that is the difference.

(Refer Slide Time: 30:23)

Drawing a Simple Random Sample Without Replacement (SRSWOR)

`sample` takes a sample of the specified size from the elements of `x` using either with or without replacement.

Usage

`sample(x, size, replace = FALSE)`

Arguments

- `x` Either a vector of one or more elements from which to choose, or a positive integer.
- `size` a non-negative integer giving the number of items to choose.
- `replace` Should sampling be with replacement?

So now, my objective here is that once we are given a population then how to draw such type of sample using the R software. So, here we have a command sample 'sample', this sample takes the sample of the specified of a population which is indicated by here a small x using with or without replacement. So, how it is given? Suppose, we have a population which is indicated by here x so this is here population and we try to use the command here sample and inside the parenthesis we write the population from where we want to draw the sample and then we try to write down the size, size is the small n. Means how many units you want to draw from the population and then you have an option here replace.

This replace can take two possible logical operators TRUE or FALSE and you also know that this TRUE and FALSE can also be indicated by the first letter T and F also. For example, TRUE can be represented by T and FALSE can be represented by F. So, in case if I am trying to use here replace equal to FALSE. So, do not you think that the meaning itself is indicating whether you are trying to draw the sample by without replacement or by with replacement, you are trying

to say replace is FALSE that means you do not want to replace. When you do not want to replace this is sampling without replacement. So, that is actually default.

But in case if you want to have sampling with replacement, this replace will become TRUE here in this case you have to write down here TRUE, that is all. So now, we have indicated here all the meanings of this x size and replace.

(Refer Slide Time: 32:20)

```
Drawing a Simple Random Sample Without Replacement (SRSWOR)

First we define a population units containing the numbers 1 to 20.
This can be defined by a sequence as x.

> x <- 1:20
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

# R Console
> x <- 1:20
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

So now, let us try to take some example and try to see how it is working and after that I will try to show you it on the R console also. So, suppose I try to create an artificial population. Suppose, my population is having the numbers 1 to 20, I mean 1, 2, 3, 4, up to 20 numbers. So now, I create here, this here x, this is my here population. So, you can see here this I have created like this.

(Refer Slide Time: 32:48)

Drawing a Simple Random Sample Without Replacement (SRSWOR)

Let us draw the sample of size 5 from population x by SRSWOR .

This is controlled by the statement `replace = FALSE` inside the argument.

```
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20
```

SRSWOR command

```
sample(x, size=5, replace = FALSE)
```

13

Now, I would like to draw the sample from this population by using simple random sampling with and without replacement. So, first I go for simple random sampling without a replacement. Suppose, I want to draw here a sample of size 5. So, I want to draw this sample from the population which is indicated by here x so I will write down here sample, then inside the parenthesis x which is the population and then size is equal to 5. That means I want to draw 5 units from these numbers 1 to 20 which are mentioned inside the data vector x . And then I am using here the option `replace`, `replace` equal to `FALSE` that means I need simple random sampling without replacement.

(Refer Slide Time: 33:27)

```
Drawing a Simple Random Sample Without Replacement (SRSWOR)  
  
> sample(x, size=5, replace = FALSE) ||  
[1] 15 1 10 11 5  
  
> sample(x, size=5, replace = FALSE)  
[1] 13 9 10 17 20  
  
> sample(x, size=5, replace = FALSE)  
[1] 11 8 5 12 13
```

So, if you try to execute it on the R console this will give you this type of outcome. For example, you can see here once you operate it, it will give you the values 15, 1, 10, 11, 5. That means these units 1, 5, 10, 11, 15 you can see here 1, 5, 11, 10 and 15, these five units are going to create your sample. Now, you choose this sample and try to collect observation on these 5 sampling units and similarly if you try to repeat this command possibly you will get a different values because the units have been drawn at random, so these units or the sampling units are expected to change as soon as you draw a new sample.

So, you can see here I have taken here three possible sample of size 5 and each of the sample is different, I will try to show you on the R console also. For example, here you can see my sampling units are 15, 1, 10, 11, 5 in the second sample the units are 13, 9, 10, 17 and 20 and in the third case the sampling units are 11, 8, 5, 12 and 13. But the advantage is that the probability of drawing any of the sampling unit from this population of size 20 is the same and it remain the same at each every, at every step.

(Refer Slide Time: 34:47)

Drawing a Simple Random Sample Without Replacement (SRSWOR)

```
R Console
> sample(x, size=5, replace = FALSE)
[1] 15 1 10 11 5
>
> sample(x, size=5, replace = FALSE)
[1] 13 9 10 17 20
>
> sample(x, size=5, replace = FALSE)
[1] 11 8 5 12 13
>
```

And this is the screenshot of the same operation.

(Refer Slide Time: 34:51)

Drawing a Simple Random Sample With Replacement (SRSWR)

Let us draw the sample of size 10 from population **x** by SRSWR .
This is controlled by the statement `replace = TRUE` inside the argument.

```
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20
```

SRSWR Command

```
sample(x, size=10, replace = TRUE)
```

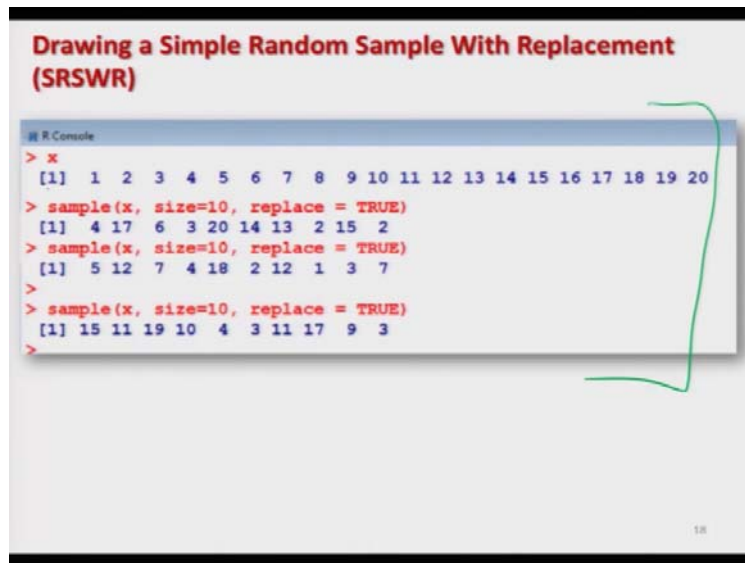
Similarly, if you want to draw here a sample by simple random sampling with replacement then you have to use the same command `sample x size` and then but `replace` will become here `TRUE` now. So, suppose I want to draw here a sample of size 10 from this population of size 20, then I have to write down here the sample `x` is the population and `size` is equal to here 10 and `replace` equal to `TRUE`.

(Refer Slide Time: 35:19)

```
Drawing a Simple Random Sample With Replacement (SRSWR)  
SRSWR  
> sample(x, size=10, replace = TRUE)  
[1] 4 17 6 3 20 14 13 2 15 2  
Value 2 is repeated.  
  
> sample(x, size=10, replace = TRUE)  
[1] 5 12 7 4 18 2 12 1 3 7  
Values 12 and 7 are repeated.  
  
> sample(x, size=10, replace = TRUE)  
[1] 15 11 19 10 4 3 11 17 9 3  
Value 11 is repeated.
```

And when I try to execute this command on the R console I get here this type of outcome, if I try to write down here sample x size 10 replace equal to TRUE I get here this value and you can see here this 2 value is getting repeated here. And similarly, if I try to repeat it you can see here in the second sample 12 and 7 they are repeated. And similarly, if you try to take here 1 more sample means say this value 11 is going to be repeated.

(Refer Slide Time: 35:47)

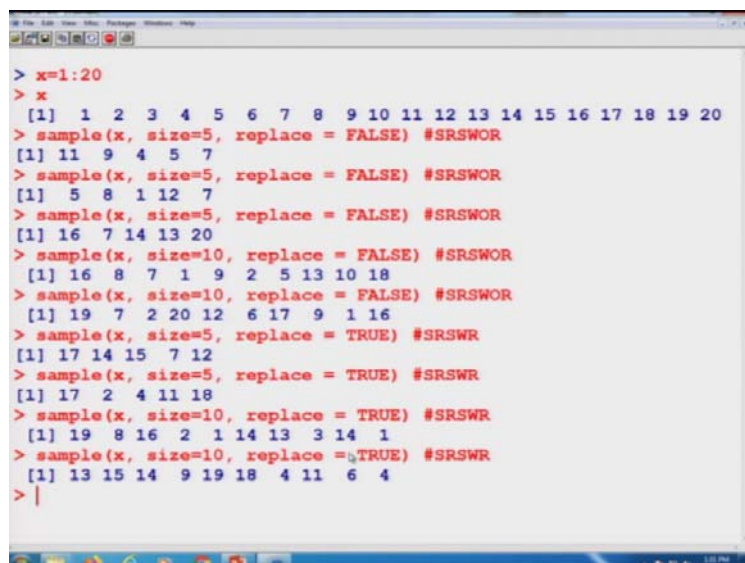


Drawing a Simple Random Sample With Replacement (SRSWR)

```
R Console
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
> sample(x, size=10, replace = TRUE)
[1] 4 17 6 3 20 14 13 2 15 2
> sample(x, size=10, replace = TRUE)
[1] 5 12 7 4 18 2 12 1 3 7
>
> sample(x, size=10, replace = TRUE)
[1] 15 11 19 10 4 3 11 17 9 3
>
```

Now, tell me one thing that when I am trying to do the same operation on the R console, do you think that I am going to get the same values as in this screenshot? Certainly not, this probability is very less because all the units are going to be drawn randomly and that is the advantage that you are trying to choose a sample without any bias, without any hidden feeling or intentions and that is how R helps you in having this type of sample. So, let us try to do these operations on the R console and try to see how these things are happening.

(Refer Slide Time: 36:27)



```
> x=1:20
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
> sample(x, size=5, replace = FALSE) #SRSWOR
[1] 11 9 4 5 7
> sample(x, size=5, replace = FALSE) #SRSWOR
[1] 5 8 1 12 7
> sample(x, size=5, replace = FALSE) #SRSWOR
[1] 16 7 14 13 20
> sample(x, size=10, replace = FALSE) #SRSWOR
[1] 16 8 7 1 9 2 5 13 10 18
> sample(x, size=10, replace = FALSE) #SRSWOR
[1] 19 7 2 20 12 6 17 9 1 16
> sample(x, size=5, replace = TRUE) #SRSWR
[1] 17 14 15 7 12
> sample(x, size=5, replace = TRUE) #SRSWR
[1] 17 2 4 11 18
> sample(x, size=10, replace = TRUE) #SRSWR
[1] 19 8 16 2 1 14 13 3 14 1
> sample(x, size=10, replace = TRUE) #SRSWR
[1] 13 15 14 9 19 18 4 11 6 4
> |
```

```

> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
> sample(x, size=5, replace = FALSE) #SRSWOR
[1] 11 9 4 5 7
> sample(x, size=5, replace = FALSE) #SRSWOR
[1] 5 8 1 12 7
> sample(x, size=5, replace = FALSE) #SRSWOR
[1] 16 7 14 13 20
> sample(x, size=10, replace = FALSE) #SRSWOR
[1] 16 8 7 1 9 2 5 13 10 18
> sample(x, size=10, replace = FALSE) #SRSWOR
[1] 19 7 2 20 12 6 17 9 1 16
> sample(x, size=5, replace = TRUE) #SRSWR
[1] 17 14 15 7 12
> sample(x, size=5, replace = TRUE) #SRSWR
[1] 17 2 4 11 18
> sample(x, size=10, replace = TRUE) #SRSWR
[1] 19 8 16 2 1 14 13 3 14 1
> sample(x, size=10, replace = TRUE) #SRSWR
[1] 13 15 14 9 19 18 4 11 6 4
> sample(x, size=10, replace = TRUE) #SRSWR
[1] 18 6 18 4 8 6 6 1 7 15
> |

```

So, first let me try to create here my population, I clear the screen and I try to say x is equal to 1 to 20. So, this is my head, data vector x, from where I want to draw here a sample of size here 5. So, you can see here this is your here 'SRSWOR' you can see here you are getting here the value 11, 9, 4, 5, 7 but if you try to repeat it, you are going to get here the values 5, 8, 1, 12, 7. And if you try to repeat it you are going to get here the values 16, 7, 14, 13, 20.

And similarly, if you try to here increase the sample size suppose I make it here 10, you are getting here instead of 5 values you are getting here 10 values. And if you try to repeat it you get here another set of 10 values here which are very different from the sample that you drawn earlier. Now, under the same setup if you want to draw here a sample by simple random sampling with the replacement, suppose if I want to have a sample of size 5, I simply have to change the command that replace is equal to here TRUE that is all.

Now, this will give us a simple random sample with replacement. So, you can see here that in this case 17, 14, 15, 7 and 12 this is here but no value is repeated if you try to repeat it then again no value is repeated. Why? Because your sample size is just 5 values so the probability of repetition is less but if you try to make this sample to be here 10 you can see here whether the values are getting repeated or not, you can see here this 1 is getting repeated. And if you try to re-execute this command here you can see here that here no value is getting repeated. But, that is possible but if you try to repeat it again you can see here this 6 value is repeated 3 times.

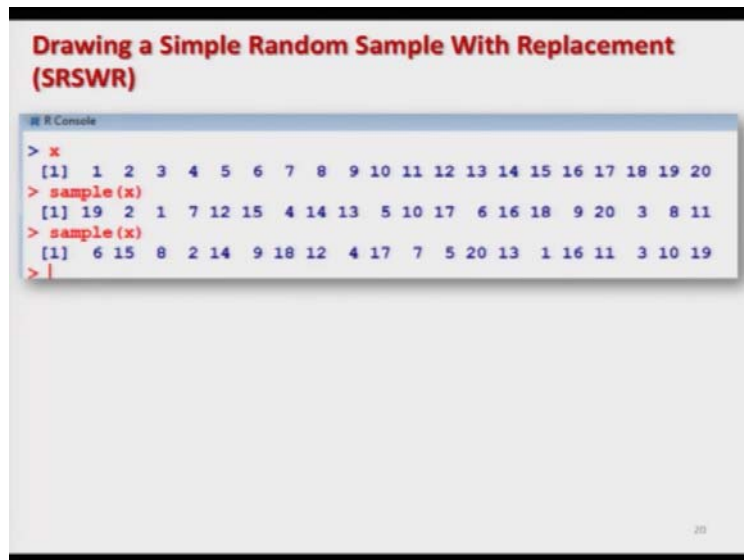
(Refer Slide Time: 38:29)

```
Drawing a Simple Random Sample With Replacement (SRSWR)  
> x  
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
18 19 20  
For sample the default for size is the number of items inferred  
from the first argument, so that sample(x) generates a  
random permutation of the elements of x (or 1:x).  
> sample(x) N sample(x) → n=N  
[1] 19 2 1 7 12 15 4 14 13 5 10 17 6 16  
18 9 20 3 8 11  
> sample(x)  
[1] 6 15 8 2 14 9 18 12 4 17 7 5 20 13  
1 16 11 3 10 19
```

So, you can see here this is what I have shown you here and this is what I have drawn here. Now, I just want to show you here the last thing that if you try to take here the same population and if you simply try to use here the command here `sample x`, `sample x` means you are trying to choose the same number of units in the sample from the population, whatever is the size of the population. So, in case if the population is of size N , then once you try to write down here `sample x` that means the size of the sample is also going to be N , so in, so from the mathematical point of view you are simply trying to get a permutation of the values in x .

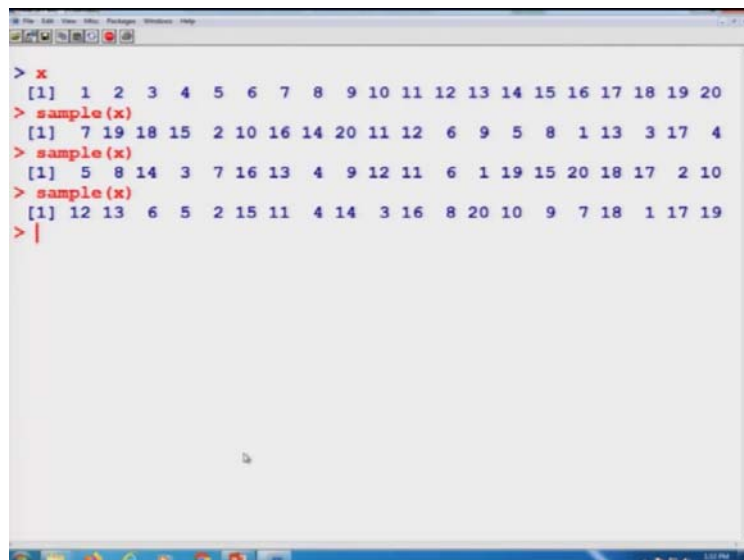
So, in case if you want to have a rearrangement of the values of x , then you can directly use here the command here `sample x`, so without using any command like `replace` or say `n`. So, you can see here once you try to take the same say here x from 1 to 20 but if you try to use here the command here `sample` you are getting here 20 values but they are arranged in a different way.

(Refer Slide Time: 39:35)



Drawing a Simple Random Sample With Replacement (SRSWR)

```
R Console
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
> sample(x)
[1] 19 2 1 7 12 15 4 14 13 5 10 17 6 16 18 9 20 3 8 11
> sample(x)
[1] 6 15 8 2 14 9 18 12 4 17 7 5 20 13 1 16 11 3 10 19
> |
```



```
R Console
> x
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
> sample(x)
[1] 7 19 18 15 2 10 16 14 20 11 12 6 9 5 8 1 13 3 17 4
> sample(x)
[1] 5 8 14 3 7 16 13 4 9 12 11 6 1 19 15 20 18 17 2 10
> sample(x)
[1] 12 13 6 5 2 15 11 4 14 3 16 8 20 10 9 7 18 1 17 19
> |
```

So, let me try to show you this thing on the R console also and that you can see this is the screenshot of the same outcome over here. So, you can see here that x here is your like this 1 to 20 but if you want to execute here sample of x, this is coming out to be like this, there are 20 values but their arrangement is different if you try to repeat it their arrangements are coming out to be different. So, this is only a random permutation of the x. So, in all those cases where you want to have a rearrangement of the values or if you want to draw a random permutation of the values you can use this sample command.

So, now we come to an end to this lecture and we are also now ending these lectures on R software. Now, I have given you sufficient background which is needed for us to understand the topics in probability theory and statistical inference, wherever I am going to use the R software. Well, I am not going to use here very high computation, I am not going to give you here very difficult mathematical proofs. In fact, I will try my best to avoid the proofs of those results as far as possible but it does not mean that there are no proofs, there are very strong mathematical proofs which gives us a confidence that yes, the things are going to work, but possibly that is beyond the scope of this course.

For that means you can continue in learning the statistical topic and you can do it. But my objective of using R in those topic is very simple that whatever we are trying to do in theory that I want to show you with computation that if you try to do it the same thing will work, whatever assumptions in theory I am going to make they will work when you are trying to use them on a real data.

So, we are not going to use here any real data but we will try to generate the data from the software itself, that is my objective. So, I would request you that you try to have a quick look on the concept of this R software and I will see you in the next lecture with the topics on probability theory. Till then, good bye.