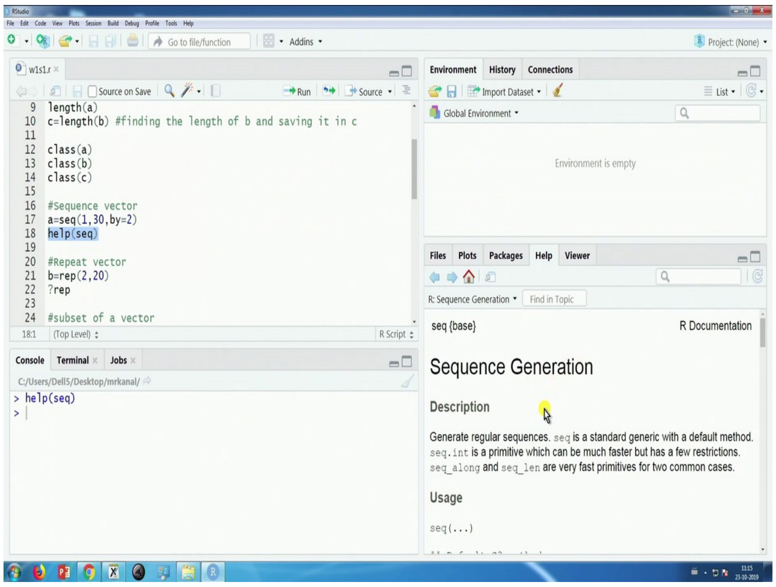
Marketing Analytics
Professor Swagato Chatterjee
Vinod Gupta School of Management
Indian Institute of Technology, Kharagpur
Lecture 02
Introduction to R Programming (Contd.)

Hello, everybody, welcome to session 2 of marketing analytics this is Dr. Swagato Chatterjee from VGSOM IIT Kharagpur will be taking your class.

(Refer Slide Time: 00:27)



```
length(a)
c=length(b) #finding the length of b and saving it in c

class(a)
class(b)
class(c)

#Sequence vector
a=seq(1,30,by=2)
help(seq)

#Repeat vector
b=rep(2,20)
?rep

#subset of a vector
```

```
> help(seq)
>
```

seq {base} R Documentation

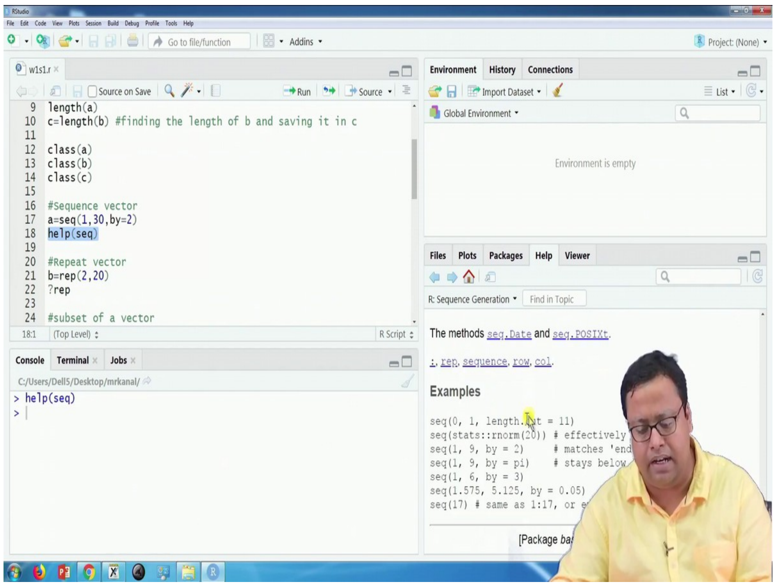
Sequence Generation

Description

Generate regular sequences. `seq` is a standard generic with a default method. `seq.int` is a primitive which can be much faster but has a few restrictions. `seq_along` and `seq_len` are very fast primitives for two common cases.

Usage

```
seq(...)
```



```
length(a)
c=length(b) #finding the length of b and saving it in c

class(a)
class(b)
class(c)

#Sequence vector
a=seq(1,30,by=2)
help(seq)

#Repeat vector
b=rep(2,20)
?rep

#subset of a vector
```

```
> help(seq)
>
```

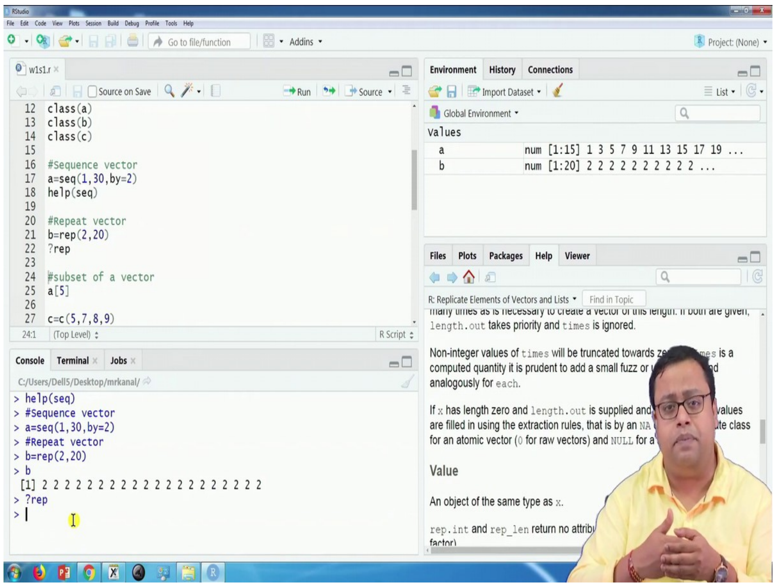
The methods `seq.Date` and `seq.POSIXt`.

`:`, `rep`, `sequence`, `row`, `col`.

Examples

```
seq(0, 1, length.out = 11)
seq(stats::rnorm(20)) # effectively
seq(1, 9, by = 2)     # matches 'end
seq(1, 9, by = pi)    # stays below
seq(1, 6, by = 3)
seq(1.575, 5.125, by = 0.05)
seq(17) # same as 1:17, or e
```

[Package *ba*



```
12  class(a)
13  class(b)
14  class(c)
15
16  #Sequence vector
17  a=seq(1,30,by=2)
18  help(seq)
19
20  #Repeat vector
21  b=rep(2,20)
22  ?rep
23
24  #subset of a vector
25  a[5]
26
27  c=c(5,7,8,9)
```

```
C:/Users/Dell5/Desktop/mrkanal/
> help(seq)
> #Sequence vector
> a=seq(1,30,by=2)
> #Repeat vector
> b=rep(2,20)
> b
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
> ?rep
> |
```

```
Values
  a        num [1:15] 1 3 5 7 9 11 13 15 17 19 ...
  b        num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
```

R: Replicate Elements of Vectors and Lists

many times as is necessary to create a vector of this length. If both are given, `length.out` takes priority and `times` is ignored.

Non-integer values of `times` will be truncated towards ze[illegible]mes is a computed quantity it is prudent to add a small fuzz or [illegible]d analogously for `each`.

If `x` has length zero and `length.out` is supplied and [illegible] values are filled in using the extraction rules, that is by an `NA` [illegible]te class for an atomic vector (`0` for raw vectors) and `NULL` for a

Value

An object of the same type as `x`.

`rep.int` and `rep_len` return no attrib[illegible] factor)

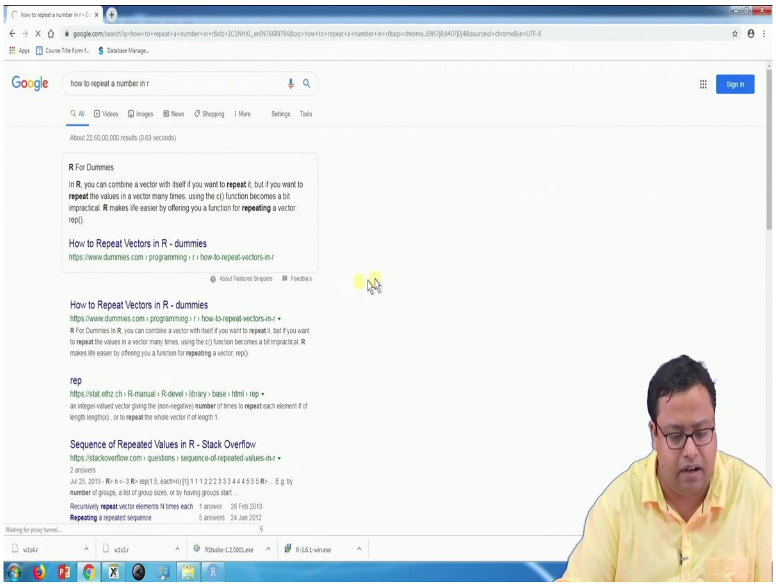
In the last session we have done till this part where we have just started with these two lines and in this particular class we will go ahead from here. So sequence is the function that we have used “seq” and I have shown you that if you know the function name, you can run this help line, help of sequence you can run this line and in here you will get all the overall help of that particular function.

And if you come down there are also arguments which explains that what are the various syntax you have to write, what that mean and if you further come down and read all of these things and further come down you can find out examples as well. And from the examples you get an idea that what these things are and I have run these 2 line and probably the line number 17 and that created a which is a sequence of 1, 3, 5, 7, 9 and so on till 29.

Similarly, there is another function called repeat function and the function is written as rep. So, here if I run this function and if I just print b, as I have saved it, it actually repeats 2, 20 times, so repeat whatever you want to repeat, then comma how many times you want to repeat. And if you want to know about the syntax, you can also write question mark. So, instead of writing helps within bracket rep, you can also write question mark rep, it will do the same thing in this area it will give you I would say the help of that particular function.

Now all of these thing is valid as long as you know that okay that there is a seq function available or there is a rep function available. Now, if you do not know the function name actually as I told in the last class there are if not millions, probably lakhs of functions that are there inbuilt, developed by somebody and stored in a package in R, so it is not possible for anybody to actually go ahead and know all those functions, it is absolutely impossible. But as I again told that R is an open software so there are lots of helps available online. So the best choice in this particular case is to go online.

(Refer Slide Time: 3:02)


how to repeat a number in r

About 22,60,00,000 results (0.63 seconds)

R For Dummies

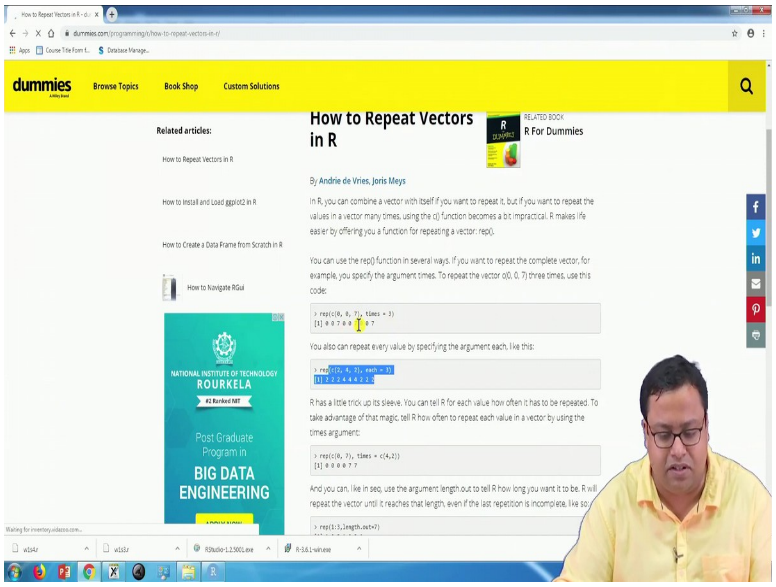
In **R**, you can combine a vector with itself if you want to **repeat** it, but if you want to **repeat** the values in a vector many times, using the c() function becomes a bit impractical. **R** makes life easier by offering you a function for **repeating** a vector: rep().

How to Repeat Vectors in R - dummies
https://www.dummies.com › programming › r › how-to-repeat-vectors-in-r

How to Repeat Vectors in R - dummies
https://www.dummies.com › programming › r › how-to-repeat-vectors-in-r
R For Dummies In **R**, you can combine a vector with itself if you want to **repeat** it, but if you want to **repeat** the values in a vector many times, using the c() function becomes a bit impractical. **R** makes life easier by offering you a function for **repeating** a vector: rep().

rep
https://stat.ethz.ch › R-manual › R-devel › library › base › html › rep
an integer-valued vector giving the (non-negative) **number** of times to **repeat** each element if of length length(x) , or to **repeat** the whole vector if of length 1.

Sequence of Repeated Values in R - Stack Overflow
https://stackoverflow.com › questions › sequence-of-repeated-values-in-r
2 answers
Jul 25, 2019 - **R**> n <- 3 **R**> rep(1:5, each=n) [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5 **R**> ... E.g. by **number** of groups, a list of group sizes, or by having groups start ...
Recursively **repeat** vector elements N times each 1 answer 28 Feb 2013
Repeating a repeated sequence 5 answers 24 Jun 2012


How to Repeat Vectors in R

By Andrie de Vries, Joris Meys

In R, you can combine a vector with itself if you want to repeat it, but if you want to repeat the values in a vector many times, using the c() function becomes a bit impractical. R makes life easier by offering you a function for repeating a vector: rep().

You can use the rep() function in several ways. If you want to repeat the complete vector, for example, you specify the argument times. To repeat the vector c(0, 0, 7) three times, use this code:

```
> rep(c(0, 0, 7), times = 3)
[1] 0 0 7 0 0 7 0 7
```

You also can repeat every value by specifying the argument each, like this:

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

R has a little trick up its sleeve. You can tell R for each value how often it has to be repeated. To take advantage of that magic, tell R how often to repeat each value in a vector by using the times argument:

```
> rep(c(0, 7), times = c(4,2))
[1] 0 0 0 0 7 7
```

And you can, like in seq, use the argument length.out to tell R how long you want it to be. R will repeat the vector until it reaches that length, even if the last repetition is incomplete, like so:

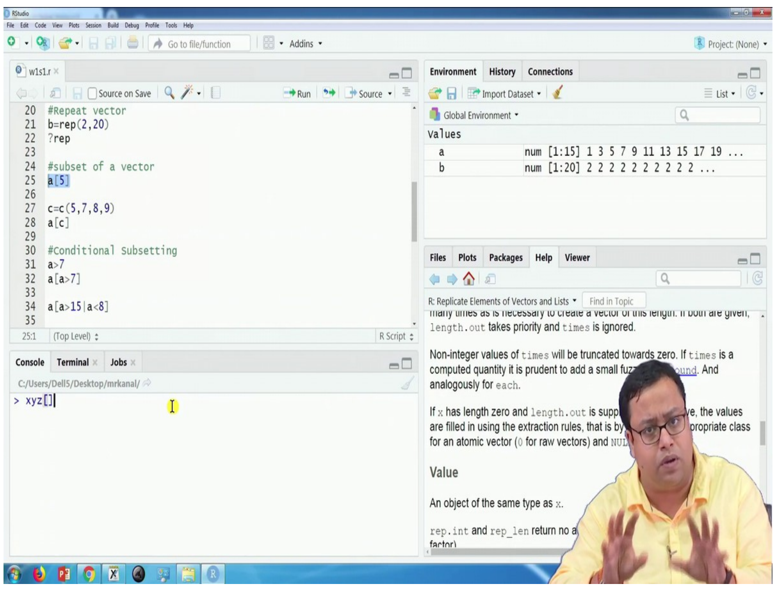
```
> rep(1:3,length.out=7)
```

So how will you go online? You will just open your browser and search in the browser that I am just writing it down that how to repeat a number in R something like this, so whatever means something like this, and then there will be lots of results coming in and some of the results will help you.

So for example, if you just click on the first one, how to repeat Vectors in R and there is something called R dummies and there are some other pages as well, I am not saying that this is the only page where you can go and we will see that somebody has written. So there are lots of helps available online.

That is what I am trying to say that there are lots of helps available online and somebody has written that this is how you have to write. Or this is how you have to write and that is the result that comes so when you write, repeat 007 times 3 that 007, 007, 007 comes up. So similarly you can get lots of other helps and those helps or something that you can use. So it is very easy in that way in R that you can get lots of online resources available and you can actually take help of that.

(Refer Slide Time: 4:12)



```
#Repeat vector
b=rep(2,20)
?rep

#subset of a vector
a[5]

c=c(5,7,8,9)
a[c]

#Conditional subsetting
a>7
a[a>7]

a[a>15|a<8]
```

```
C:/Users/Dell5/Desktop/mrkanal/
> xyz[]
```

```
Values
  a        num [1:15] 1 3 5 7 9 11 13 15 17 19 ...
  b        num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
```

R: Replicate Elements of Vectors and Lists

many times as is necessary to create a vector of this length. If both are given, `length.out` takes priority and `times` is ignored.

Non-integer values of `times` will be truncated towards zero. If `times` is a computed quantity it is prudent to add a small fuz ound. And analogously for `each`.

If `x` has length zero and `length.out` is supp e, the values are filled in using the extraction rules, that is by propriate class for an atomic vector (`0` for raw vectors) and NUL

Value

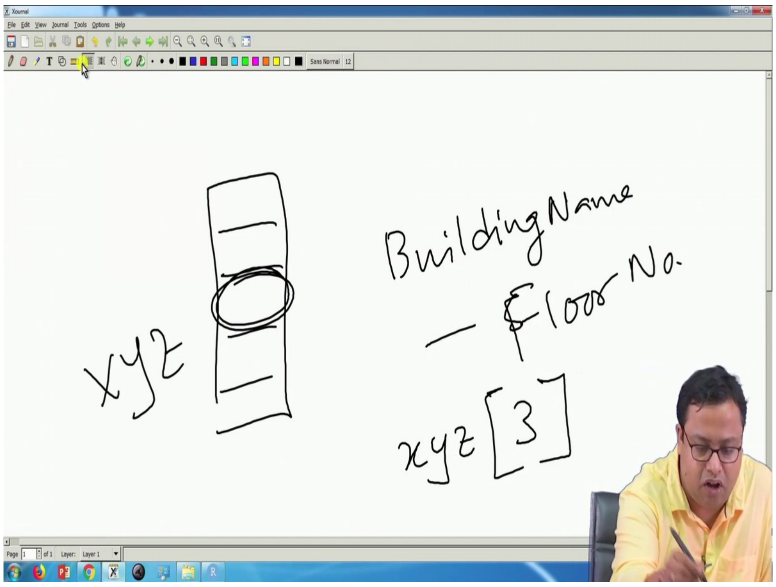
An object of the same type as `x`.

`rep.int` and `rep_len` return no a factor)

Now, as I go ahead, I will try to see something else. So I will clean my console CRTL+L and then we will try to create a subset of a vector. That means we will try to find out a smaller part of the vector. So if I just run ‘a’, and I told the third bracket, whenever you write something xyz let us say xyz and then a third bracket, that third bracket actually stands for a location within xyz. So whatever you have saved under xyz, within that, you are giving a third bracket you are trying to actually give a location.

And now that location is nothing but an address so imagine that, so there is a multi-storey building, it is a long multi story building and something like this I will just try to show you.

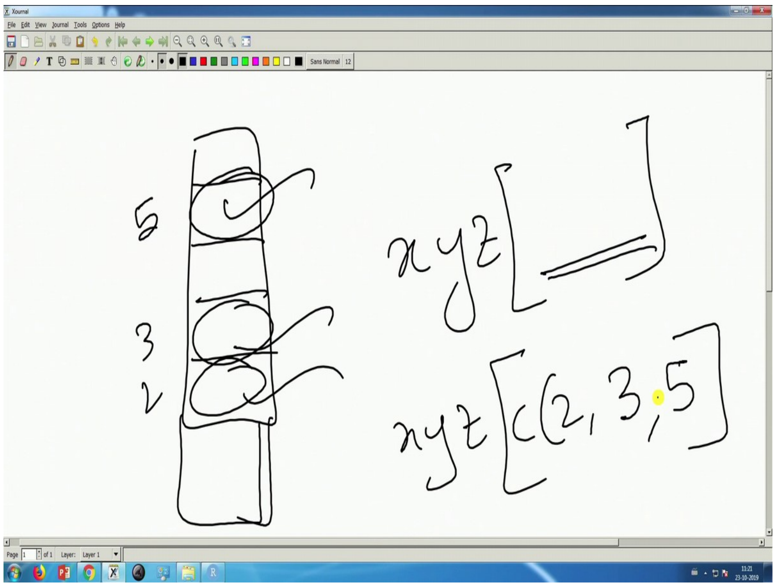
(Refer Slide Time: 5:03)



Let us say there is a multi-storey building like this. So, this is the multi storey building and if many people stay in this building and you want to know that who stays in this particular floor, in this particular building, so what do you have to give that okay, I have to give the building name and then within that probably the floor name, floor number.

So similarly here, if the building name in this case is xyz, you write xyz and if the floor number is 3, you write within bracket 3, so that will give whoever is present in the third floor of xyz.

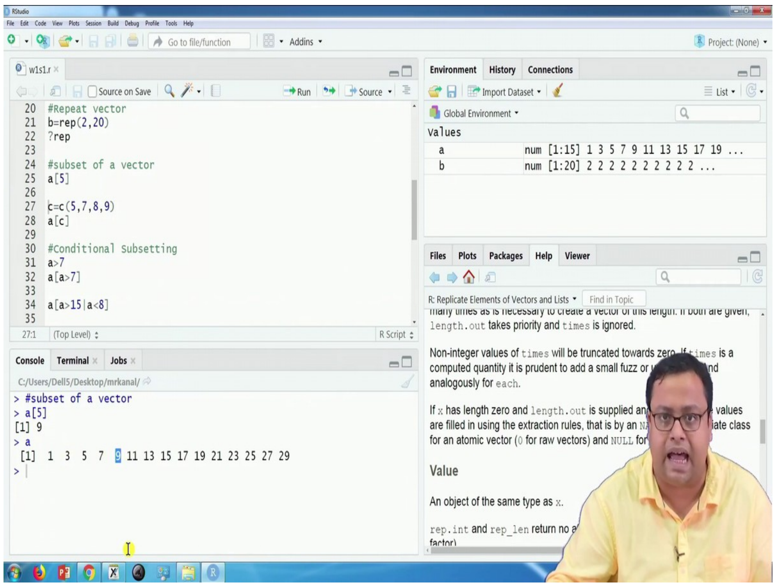
(Refer Slide Time: 5:53)



On the other hand if you want to do let us say a further multi-story building, another multi-story building which looked like this. Now, you want to know whoever stays in this floor, this floor and let us say another probably this floor also, so this is the whole building. So, you want to know this, this and this, these 3 things you want to know.

So, then you have to actually again give the building name and within that you have to give somehow 3 values. Now if you have to give 3 values within there, you have to write if it is let us say 2nd floor, 3rd floor, 4th and 5th floor, you have to write xyz within that somehow you have to write 2, 3, 5 together so you write C (2, 3, 5) something like this. So, that is where we will do in our R also.

(Refer Slide Time: 6:53)



```
#Repeat vector
b=rep(2,20)
?rep

#subset of a vector
a[5]

c=c(5,7,8,9)
a[c]

#Conditional subsetting
a>7
a[a>7]

a[a>15|a<8]
```

```
> #subset of a vector
> a[5]
[1] 9
> a
 [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29
> |
```

```
Values
a     num [1:15] 1 3 5 7 9 11 13 15 17 19 ...
b     num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
```

R: Replicate Elements of Vectors and Lists

many times as is necessary to create a vector of this length. If both are given, length.out takes priority and times is ignored.

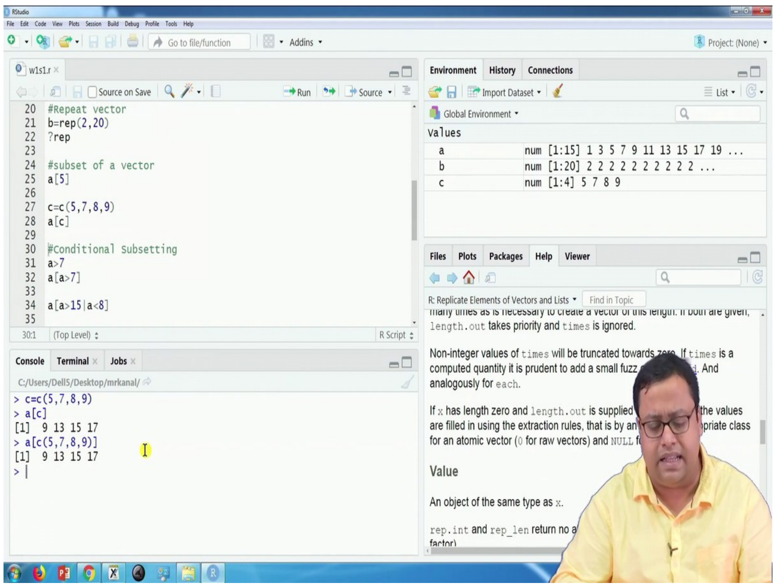
Non-integer values of times will be truncated towards zero. If times is a computed quantity it is prudent to add a small fuzz or ... and analogously for each.

If x has length zero and length.out is supplied an... values are filled in using the extraction rules, that is by an NA ... iate class for an atomic vector (0 for raw vectors) and NULL for

Value

An object of the same type as x.

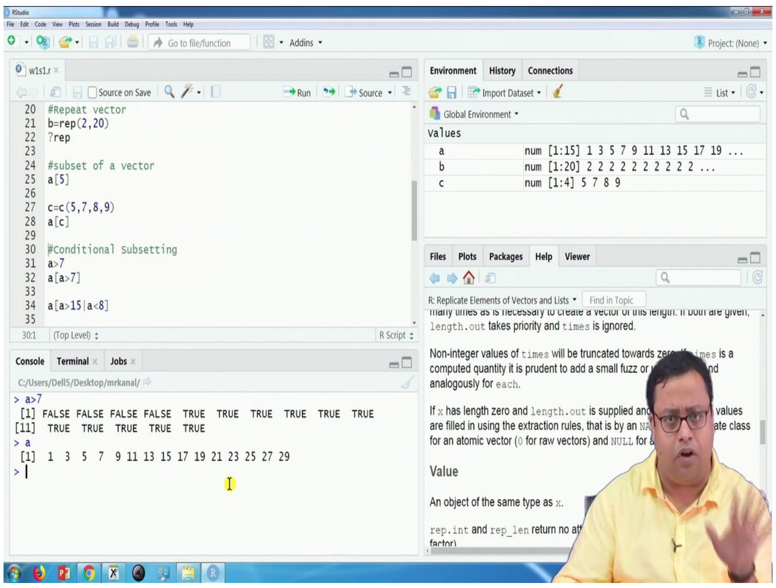
rep.int and rep_len return no a... factor).



```
#Repeat vector
b=rep(2,20)
?rep

#subset of a vector
a[5]

c=c(5,7,8,9)
a[c]

#Conditional subsetting
a>7
a[a>7]

a[a>15|a<8]
```

```
> c=c(5,7,8,9)
> a[c]
[1]  9 13 15 17
> a[c(5,7,8,9)]
[1]  9 13 15 17
> |
```

```
Values
a     num [1:15] 1 3 5 7 9 11 13 15 17 19 ...
b     num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
c     num [1:4] 5 7 8 9
```



```
20  #Repeat vector
21  b=rep(2,20)
22  ?rep
23
24  #subset of a vector
25  a[5]
26
27  c=c(5,7,8,9)
28  a[c]
29
30  #Conditional Subsetting
31  a>7
32  a[a>7]
33
34  a[a>15|a<8]
35
```

```
C:/Users/Dell5/Desktop/mrkanal/
> a>7
 [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[11]  TRUE  TRUE  TRUE  TRUE  TRUE
> a
 [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29
> |
```

```
Values
  a          num [1:15] 1 3 5 7 9 11 13 15 17 19 ...
  b          num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
  c          num [1:4] 5 7 8 9
```

length.out takes priority and times is ignored.

Non-integer values of times will be truncated towards ze… times is a computed quantity it is prudent to add a small fuzz or … nd analogously for each.

If x has length zero and length.out is supplied an… values are filled in using the extraction rules, that is by an NA… ate class for an atomic vector (0 for raw vectors) and NULL for …

Value

An object of the same type as x.

rep.int and rep_len return no at…

So in R also we will do the same thing. So, we write a[5], if I run this it gives me whatever is there in the 5th entry of 'a', so if you see that the 5th entry of a is is 9. So that is why a[5] is giving 9. Similarly, if I want to give what are the 5th, 7th, 8th or 9th entry of a, I can store this 5, 7, 8 and 9 in a vector called 'c' and then call whatever is within a by this name 'c'.

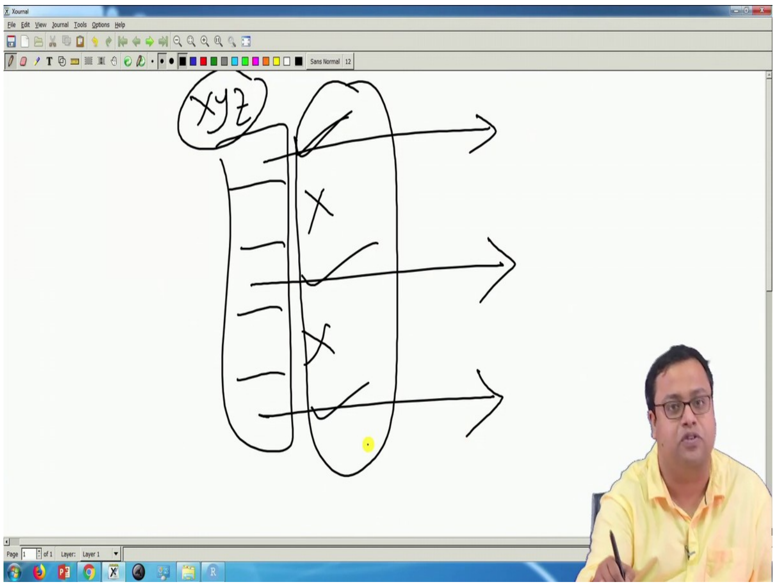
So you give this particular Vector 5, 7, 8, 9, also a name 'c' and call 'c' within 'a', so if I call that the 5th entry is 9, which we have shown it right now and the 7th entry is 13, the 8th entry is 15 and a 9th entry is 17. That is getting printed. You can directly write it like this as well. You can also write a[c(5, 7, 8, 9)] something like this. You can directly write that as well instead of creating a 'c' vector separately, and you will get the same result. So that is how you create a subset of the vector.

Now here I am giving you the location not always I will be able to give you the location sometimes, we try to find out that which of the guys who has let us say, profitability higher than 10 percent, margin higher than 10 percent. So, you sometimes the sub-setting is conditional. So, when the conditional sub-setting you have to give a condition for the sub-setting we use certain conditional notations as well.

So for example, logical operators we call them for example let us say if I just write a>7 as written in line number 31, if I enter here, it is giving lots of false and lots of trues, the false are what? So 'a' is like this 1, 3, 5, 7, 9, 11 and I have written a >7. So these 4 guys are not greater than 7. So that is why the first 4 entries are false. And then 9 to 29, all are greater than 7.

So the rest of the entries are true. So wherever a is higher than 7, it is giving me false, wherever a is sorry, it is giving me true and wherever a is smaller than 7, it is giving me false, so a greater than 7 gives me a true-false kind of output, output that gives me true and false.

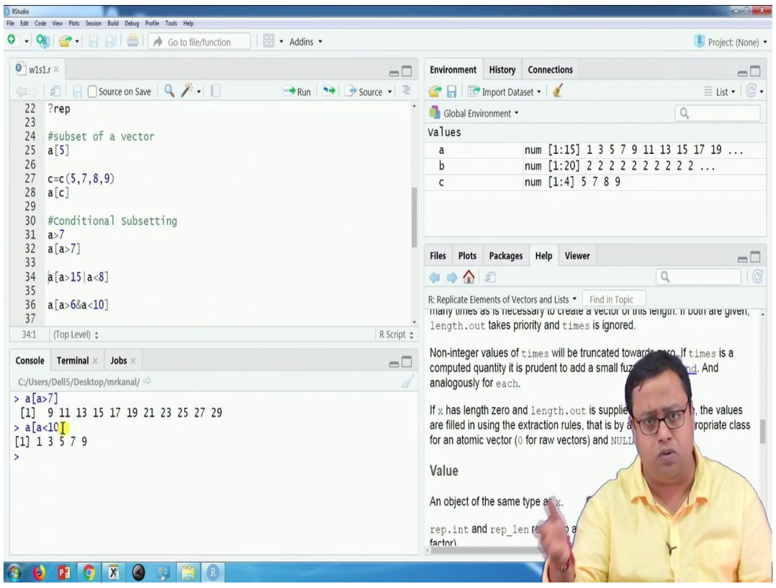
(Refer Slide Time: 9:41)



Now, if I use this true-false within ‘a’, so if I just write something like this, where let us say this is my house, sorry, this is my multi story building. And I say that okay, you give me the person who is living here, living here, living here, and not living here, not living here. So this part is the true false part.

So this is the true false thing that I am fitting on this xyz building. So, then this is tick, so this result comes up this is tick so, this result comes up and this is tick, so this result comes up. So, all these 3 results come up when within xyz I give this true false function.

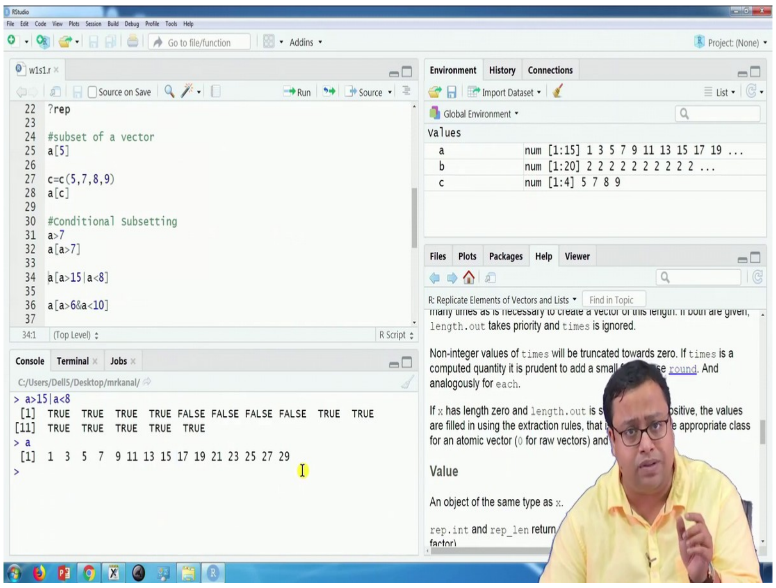
(Refer Slide Time: 10:16)



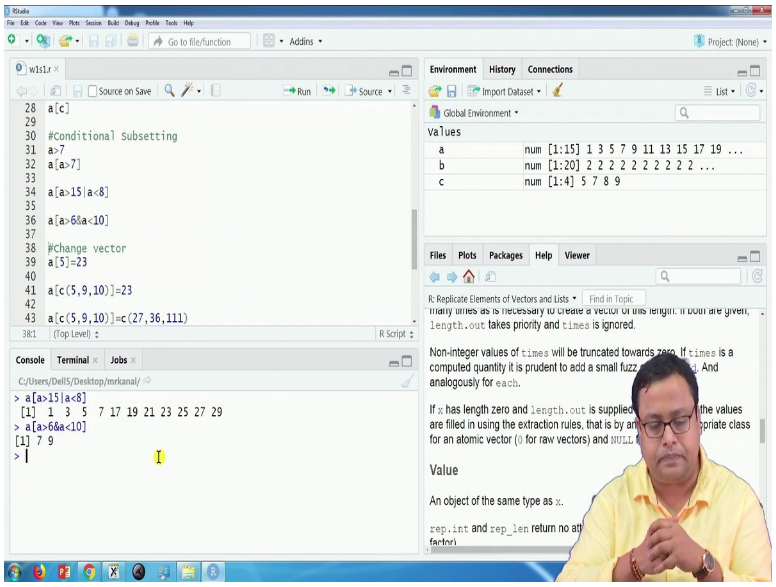
```
?rep

#subset of a vector
a[5]

c=c(5,7,8,9)
a[c]

#Conditional Subsetting
a>7
a[a>7]

a[a>15|a<8]

a[a>6&a<10]
```

```
Values
a        num [1:15] 1 3 5 7 9 11 13 15 17 19 ...
b        num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
c        num [1:4] 5 7 8 9
```

```
> a[a>7]
 [1]  9 11 13 15 17 19 21 23 25 27 29
> a[a<10]
[1] 1 3 5 7 9
>
```



```
> a>15|a<8
 [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE
[11]  TRUE  TRUE  TRUE  TRUE  TRUE
> a
 [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29
>
```



```
a[c]

#Conditional Subsetting
a>7
a[a>7]

a[a>15|a<8]

a[a>6&a<10]

#Change vector
a[5]=23

a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)
```

```
> a[a>15|a<8]
 [1]  1  3  5  7 17 19 21 23 25 27 29
> a[a>6&a<10]
[1] 7 9
> 
```

```
Values
  a        num [1:15] 1 3 5 7 9 11 13 15 17 19 ...
  b        num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
  c        num [1:4] 5 7 8 9
```

R: Replicate Elements of Vectors and Lists

length.out takes priority and times is ignored.

Non-integer values of times will be truncated towards zero. If times is a computed quantity it is prudent to add a small fuzz [illegible]. And analogously for each.

If x has length zero and length.out is supplied [illegible] the values are filled in using the extraction rules, that is by an [illegible] priate class for an atomic vector (0 for raw vectors) and NULL f [illegible]

Value

An object of the same type as x.

rep.int and rep_len return no att [illegible]

This is what I will do it here. So, I will write let us say a[a > 7], so if I run this what it gives is a > 7 ,whoever is true in that vector whatever it was coming true corresponding values of ‘a’ is being given to me and all the values who are false it is not been given to me.

Now, I can do it for the other one also so, let us say if I ask you that how to get the same thing, but a is smaller than 10 print ‘a’, but a is smaller than 10. So, all you have to write is a[a<10] and all the values who are smaller than 10 is being shown to you. So, at this moment I would probably sometimes it is my suggestion that you probably pause the video and try to do a little bit of playing on your own let us say if you want to find out which is higher than 15, how to do that try to do it on your own.

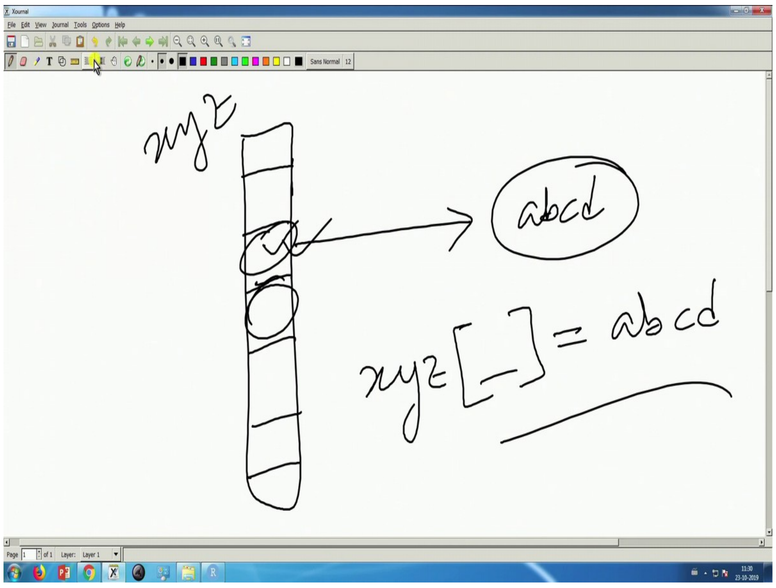
So, next is sometimes these operators are there are more than one conditions and you have to, we all know that we have more than one conditions we have to join them by AND operator or OR operator or NOR operator and so on. So I will show you the easiest one and you can search as I told that if you do not know how to find out this thing you can go to Google and close your eyes and just search that how to do OR operator in R or how to do AND operator in R?

And you will find out the results, but I will show you some basic ones. So for example, let us say if I just write a > 15 or OR is like this sign which is shift and the key which is just above enter and then a < 8. So a > 15 | a < 8 and if I just run this, it will give me trues and false such that, see I have got first 4 as trues because a was like this 1, 3, 5, 7. So these are smaller than 8. That is why they are true.

Again, I got last few values are true because 19 to 28 are higher than 15. So these guys are also true. In between so 7 to 17 they are neither 7 to 15 they are neither higher than 15 nor smaller 9 to 15 actually, they are neither higher than 15 nor smaller than 8. So these 4 guys are coming false. So that is why there are 4 falses. So now if I use these true false in the sub-setting, if I just run the line which has been written in line number 34, I get all the values who are in bit.

So all I am trying to show in this particular line is how to write OR function. Similarly, you change it to AND function, it is just like this. So, a > 6 & a < 10 ; so, that means between 6 and 10 if I just print that, I get the value 7 and 9 because these are the only 2 values which are between 6 and 10. So, this is how I can subset a particular vector. Now, if I can subset a particular vector, I can change the values also.

(Refer Slide Time: 13:38)

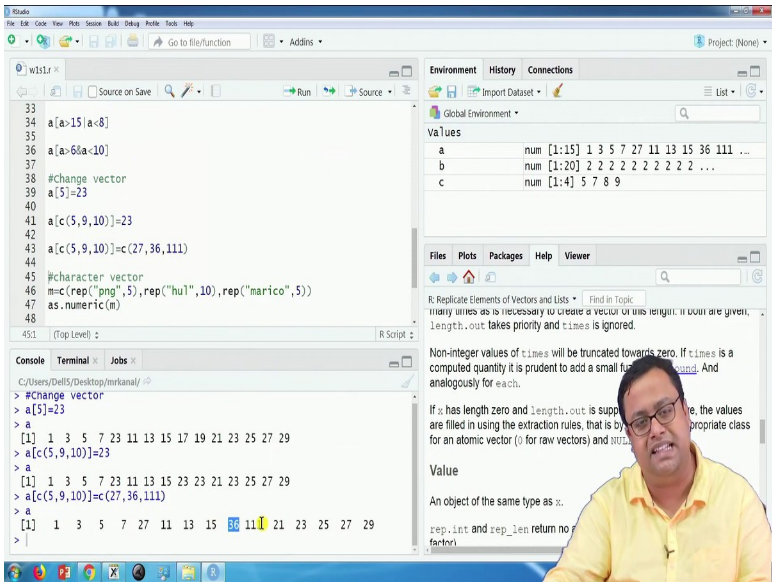


So now if you want to see that , so I will be using this particular building, and I am saying that the person who was living in this particular building has changed. And in this building, whoever was living in the 4th floor, whatever was the name, you have to change the name to something else and I have given you the new name whoever stays in the building.

So 1, 2, 3, 4 this is the 4th floor or actually this is the 4th floor, the ground floor and then 1, 2, 3, 4 this one is the 4th floor. So in the 4th floor I will say that okay, this name, whatever resides there, I do not care. The name whatever nameplate is written here, you just remove that nameplate and put a new name, the new name is let us say abcd whatever you want to put here.

So, what you have to do, if the buildings name is xyz, you have to say xyz and there is a 4th floor, whatever is a floor number, and this value has to be whatever was the value I do not care This value has to be abcd something like this. So the same thing we try to do it here also.

(Refer Slide Time: 14:59)



```
a[a>15|a<8]

a[a>6&a<10]

#Change vector
a[5]=23

a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)

#character vector
m=c(rep("png",5),rep("hul",10),rep("marico",5))
as.numeric(m)
```

```
> #Change vector
> a[5]=23
> a
 [1]  1  3  5  7 23 11 13 15 17 19 21 23 25 27 29
> a[c(5,9,10)]=23
> a
 [1]  1  3  5  7 23 11 13 15 23 23 21 23 25 27 29
> a[c(5,9,10)]=c(27,36,111)
> a
 [1]   1   3   5   7  27  11  13  15  36 111  21  23  25  27  29
> 
```

```
Values
a    num [1:15] 1 3 5 7 27 11 13 15 36 111 ...
b    num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
c    num [1:4] 5 7 8 9
```

R: Replicate Elements of Vectors and Lists

length.out takes priority and times is ignored.

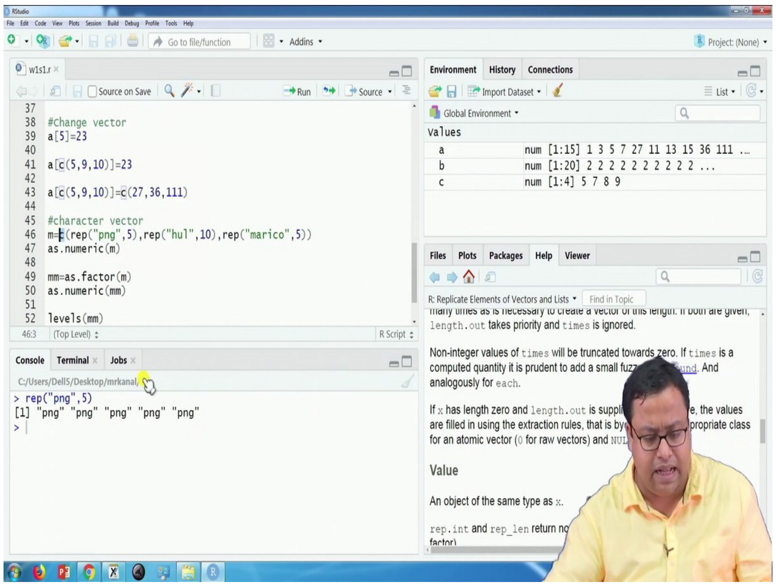
Non-integer values of times will be truncated towards zero. If times is a computed quantity it is prudent to add a small fuz ound. And analogously for each.

If x has length zero and length.out is supp e, the values are filled in using the extraction rules, that is by propriate class for an atomic vector (0 for raw vectors) and NUL

Value

An object of the same type as x.

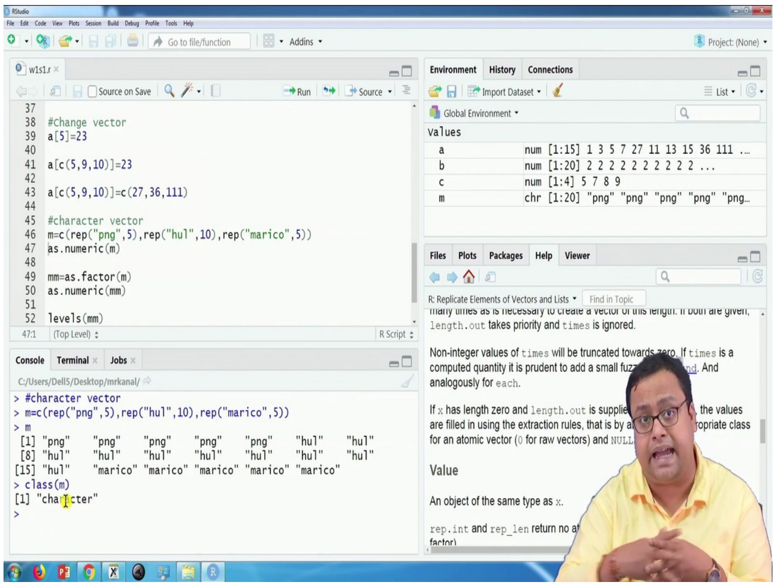
rep.int and rep_len return no



```
#Change vector
a[5]=23

a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)

#character vector
m=c(rep("png",5),rep("hul",10),rep("marico",5))
as.numeric(m)

mm=as.factor(m)
as.numeric(mm)

levels(mm)
```

```
> rep("png",5)
[1] "png" "png" "png" "png" "png"
> 
```

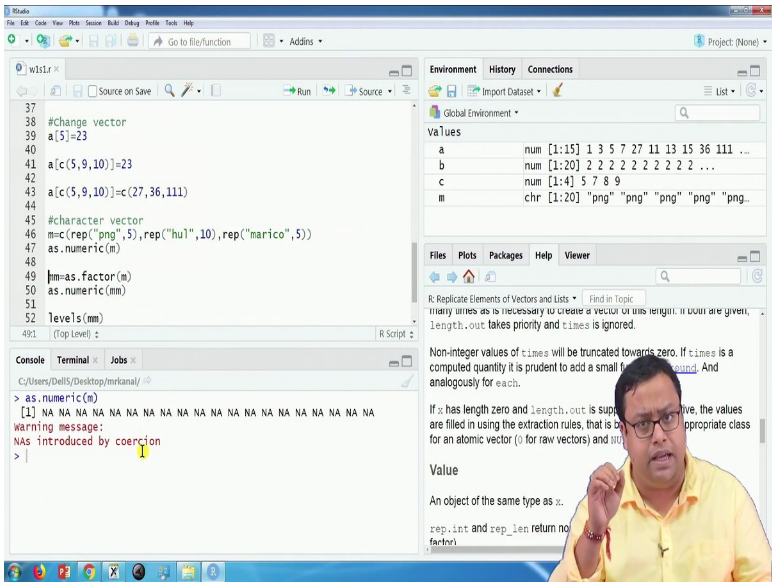
```
Values
a    num [1:15] 1 3 5 7 27 11 13 15 36 111 ...
b    num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
c    num [1:4] 5 7 8 9
```



```
#Change vector
a[5]=23

a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)

#character vector
m=c(rep("png",5),rep("hul",10),rep("marico",5))
as.numeric(m)

mm=as.factor(m)
as.numeric(mm)

levels(mm)
```

```
> #character vector
> m=c(rep("png",5),rep("hul",10),rep("marico",5))
> m
 [1] "png"    "png"    "png"    "png"    "png"    "hul"    "hul"   
 [8] "hul"    "hul"    "hul"    "hul"    "hul"    "hul"    "hul"   
[15] "hul"    "marico" "marico" "marico" "marico" "marico"
> class(m)
[1] "character"
> 
```

```
Values
a    num [1:15] 1 3 5 7 27 11 13 15 36 111 ...
b    num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
c    num [1:4] 5 7 8 9
m    chr [1:20] "png" "png" "png" "png" "png...
```



```
37
38 #Change vector
39 a[5]=23
40
41 a[c(5,9,10)]=23
42
43 a[c(5,9,10)]=c(27,36,111)
44
45 #character vector
46 m=c(rep("png",5),rep("hul",10),rep("marico",5))
47 as.numeric(m)
48
49 mm=as.factor(m)
50 as.numeric(mm)
51
52 levels(mm)
```

```
a    num [1:15] 1 3 5 7 27 11 13 15 36 111 ...
b    num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
c    num [1:4] 5 7 8 9
m    chr [1:20] "png" "png" "png" "png" "png...
```

```
> as.numeric(m)
 [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
Warning message:
NAs introduced by coercion
> 
```

So we write, you will see that a5 is equal to 23 means whatever is the 5th guy in the vector ‘a’ change it to 23. So, if I now see ‘a’, ‘a’ gets changed here, there automatically 23 comes up. Similarly, if I want to change more than one value more than one places where I want to change I will write 5, 9, 10 that means 5th, 9th and 10th entry all of them changes to 23.

Now, if I write ‘a’ it will say that here 5th value is 23, the 9th value is 23 and 10th value is 23. Instead of 23, I can have multiple values as well different values as well. So, what I will do, I will write 5, 9, 10 is there are 3 entries we are changing, so, 27, 36, 111 something like that some values. So, the moment I put that and again I print a I will see that okay, the 5th entry is 27. The 9th entry is 36 and the 10th entry is 111.

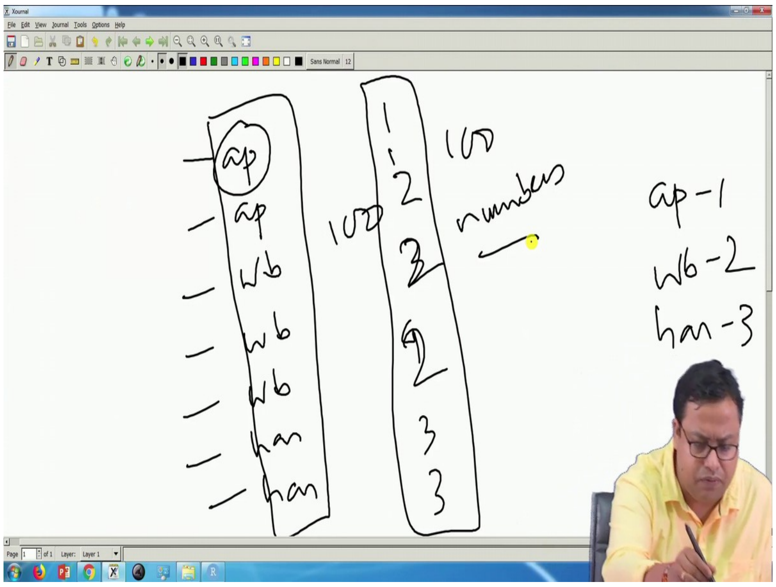
So, this is how I can actually change the values in a particular vector as well. We have dealt quite a lot of time with a numeric vector not all vector will be numeric. So let us say I will be creating a character vector. So, just check line number 46 carefully. So line number 46 what this part does, only this part? it will create repeat function, so it will create 5 pngs, png, png, png, png, png. So if I just copy that, copy that part and paste it here and run it, it will create something like this.

So what I am doing, I am creating 5 pngs and then I am creating 10 huls and then I am creating 5 marico. And then this ‘c’ what this guy doing, it is joining all of them 5 pngs, then 10 hul, and then 5 marico joining all of them and creating a vector and then it is putting that vector in a name called ‘m’, some ‘m’ name I have given. So in that name it is getting saved. So ‘m’ is equal to this, so, if I just run ‘m’, it will be like this first 5 pngs then 10 huls and then 5 mericos.

So, if what is the type of ‘m’? So what is the class of ‘m’? Class of ‘m’ is a character variable it is a character variable. Now, what does that mean? If I try to forcefully change a character variable to its numeric form, it will give me NA values, it will give me not available values error, some form of error, so if I write as dot numeric within bracket m and run this, you will see lots of NAs are coming.

That means there is a character vector you cannot change it to numeric it cannot be changed to numeric. But what happens is that sometimes R actually tries to compress this character vector is some form, which can be remembered and which will take less space.

(Refer Slide Time: 18.00)

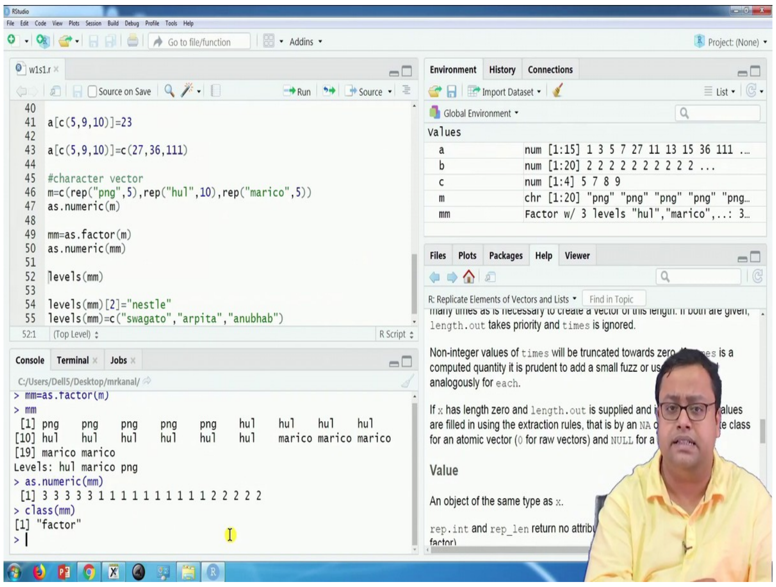


For example, let us say that there are lots of states people are coming in my class, let us say people are coming from lots of different states and some people are coming from let us say Andhra Pradesh, Andhra Pradesh, West Bengal, West Bengal, West Bengal, Haryana, Haryana somewhat it is a huge long list is there, so this long list is there each of them is a text and text you know take more space than numbers.

So, if there is a long list and this list of 100 students, then it takes a lot of space. Instead of that if I create a small list that ok ap is equal to1, wb is equal to 2 and Haryana is equal to 3 and some 4, 5 code numbers and if I write 1, 2, 3, 4 so 11 222 33 and so on, if I just create this particular thing as code number with hundred numbers then my problem is solved, then I have, I can actually put the same information in less space.

Because this number column will take much less space than a character column. So, this kind of formation will actually help us in further analysis as well in lots of places. So, this is called a factor form. So, what we do in a factor form is try to change so, we generally try to change it in a code word kind of a form.

(Refer Slide Time: 19.36)



```
a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)

#character vector
m=c(rep("png",5),rep("hul",10),rep("marico",5))
as.numeric(m)

mm=as.factor(m)
as.numeric(mm)

levels(mm)

levels(mm)[2]="nestle"
levels(mm)=c("swagato","arpita","anubhab")
```

```
> mm=as.factor(m)
> mm
 [1] png    png    png    png    png    hul    hul    hul    hul   
[10] hul    hul    hul    hul    hul    hul    marico marico marico
[19] marico marico
Levels: hul marico png
> as.numeric(mm)
 [1] 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
> class(mm)
[1] "factor"
> 
```

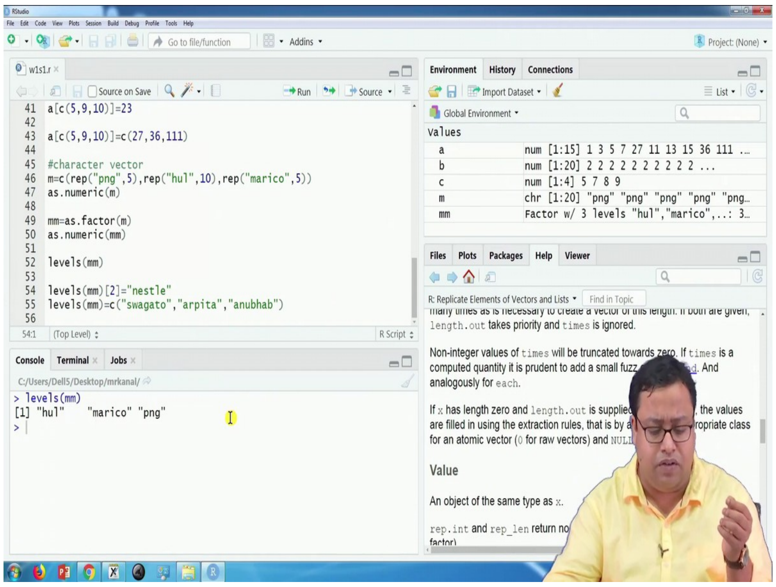
```
Values
  a      num [1:15] 1 3 5 7 27 11 13 15 36 111 ...
  b      num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
  c      num [1:4] 5 7 8 9
  m      chr [1:20] "png" "png" "png" "png" "png...
  mm     Factor w/ 3 levels "hul","marico",..: 3...
```

R: Replicate Elements of Vectors and Lists

length.out takes priority and times is ignored.

Value

An object of the same type as x.



```
a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)

#character vector
m=c(rep("png",5),rep("hul",10),rep("marico",5))
as.numeric(m)

mm=as.factor(m)
as.numeric(mm)

levels(mm)

levels(mm)[2]="nestle"
levels(mm)=c("swagato","arpita","anubhab")
```

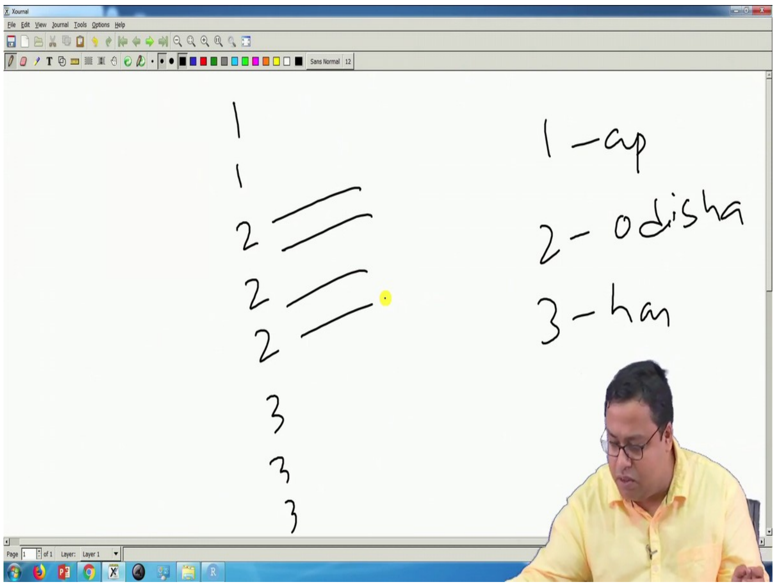
```
> levels(mm)
[1] "hul"    "marico" "png"   
> 
```

So what we do here is, if I write mm=as.factor m, mm still looks like this, it is still the names are coming up, but internally inherently R has stored these as code numbers. What kind of code numbers? R knows that there are 3 levels hul, marico and png, hul is code 1, marico is code 2 png is code 3, so in his mind it is numerical it is actually alphabetical order and it knows that it is 33333 then 1111 10 times then 2, 5 times that is how internally in his memory it has stored it like this.

And it is showing it to you with the corresponding values. How will I know that whether I am saying it is correct or not? Now if I change this mm to its numeric form as dot numeric mm, you will see that this internal code numbers are coming up. And this will have lots of usage at a later point of time we will talk about that. But I am just asking you to understand that class of mm is factor not character. And factor and character are 2 different things in R, character is not something that is numeric. Factor is internally saved as numeric variable.

What are the levels? What are the groups, various groups in a factor variable? As it can be shown using the levels function, hul, marico and png. Now the advantage is I can change this thing.

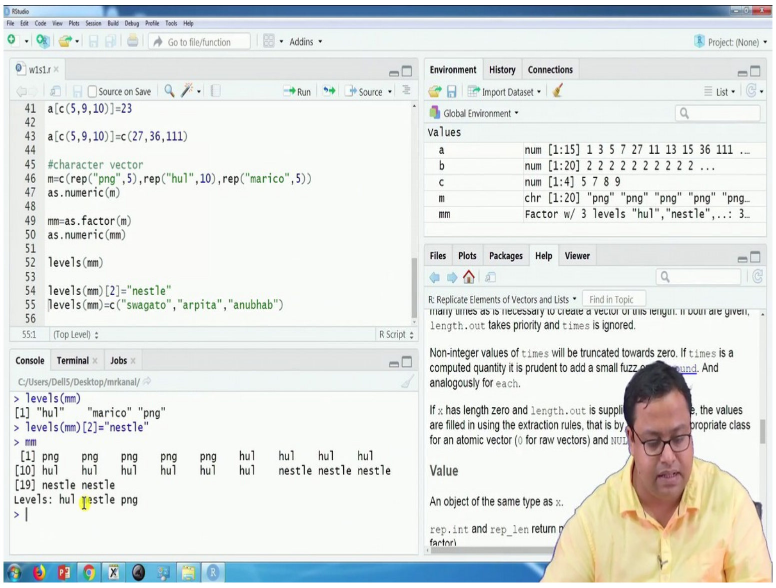
(Refer Slide Time: 21:11)



So if you remember here if there was let us say 11 222 333 and so on and 1 is ap, 2 is West Bengal, 3 is let us say Haryana and so on, I can change West Bengal I can say that okay I have written wrong name. This is not actually West Bengal these should be, let us say Odisha and all of a sudden, all that places where West Bengal was there changes to Odisha I do not have to change each of them individually.

So I can change the label name, the code number and corresponding value for that code number and the whole column gets changed. So that is what we can do here as well.

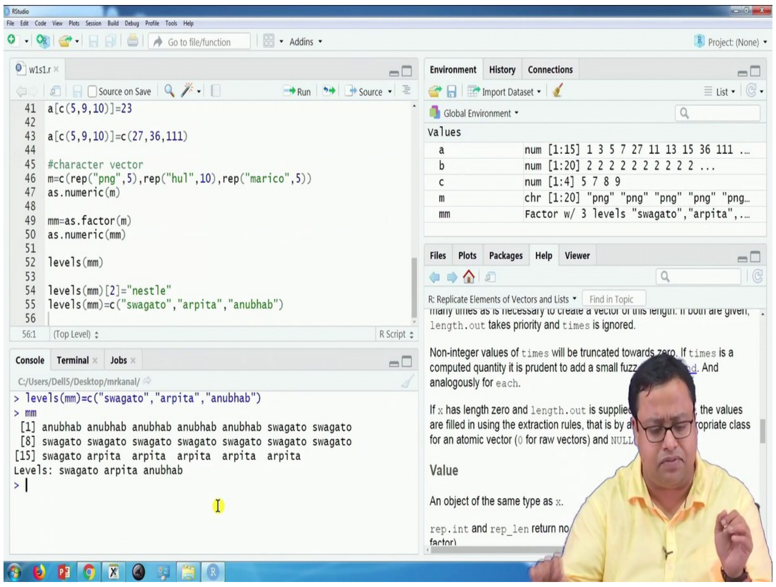
(Refer Slide Time: 21:55)



```
a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)

#character vector
m=c(rep("png",5),rep("hul",10),rep("marico",5))
as.numeric(m)

mm=as.factor(m)
as.numeric(mm)

levels(mm)

levels(mm)[2]="nestle"
levels(mm)=c("swagato","arpita","anubhab")
```

```
> levels(mm)
[1] "hul"    "marico" "png"
> levels(mm)[2]="nestle"
> mm
 [1] png    png    png    png    png    hul    hul    hul    hul
[10] hul    hul    hul    hul    hul    hul    nestle nestle nestle
[19] nestle nestle
Levels: hul nestle png
> |
```

```
Values
a     num [1:15] 1 3 5 7 27 11 13 15 36 111 ...
b     num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
c     num [1:4] 5 7 8 9
m     chr [1:20] "png" "png" "png" "png" "png...
mm    Factor w/ 3 levels "hul","nestle",..: 3...
```



```
a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)

#character vector
m=c(rep("png",5),rep("hul",10),rep("marico",5))
as.numeric(m)

mm=as.factor(m)
as.numeric(mm)

levels(mm)

levels(mm)[2]="nestle"
levels(mm)=c("swagato","arpita","anubhab")
```

```
> levels(mm)=c("swagato","arpita","anubhab")
> mm
 [1] anubhab anubhab anubhab anubhab anubhab swagato swagato
 [8] swagato swagato swagato swagato swagato swagato swagato
[15] swagato arpita  arpita  arpita  arpita  arpita
Levels: swagato arpita anubhab
> |
```

```
Values
a     num [1:15] 1 3 5 7 27 11 13 15 36 111 ...
b     num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
c     num [1:4] 5 7 8 9
m     chr [1:20] "png" "png" "png" "png" "png...
mm    Factor w/ 3 levels "swagato","arpita",...
```

So we can just say that levels of mm, what is levels of mm? These values, what is the second entry of that, second entry of levels of mm is marico this value, so whatever is the second entry of levels of mm, change it to Nestle. So if I change it to Nestle what happens, mm becomes like this, see, all of a sudden the second entry was marico that will be changed to Nestle. And wherever merico was coming up, Nestle has coming up now. So I just changed the level name, the underlying coding is still the same threes and then ones and then twos.

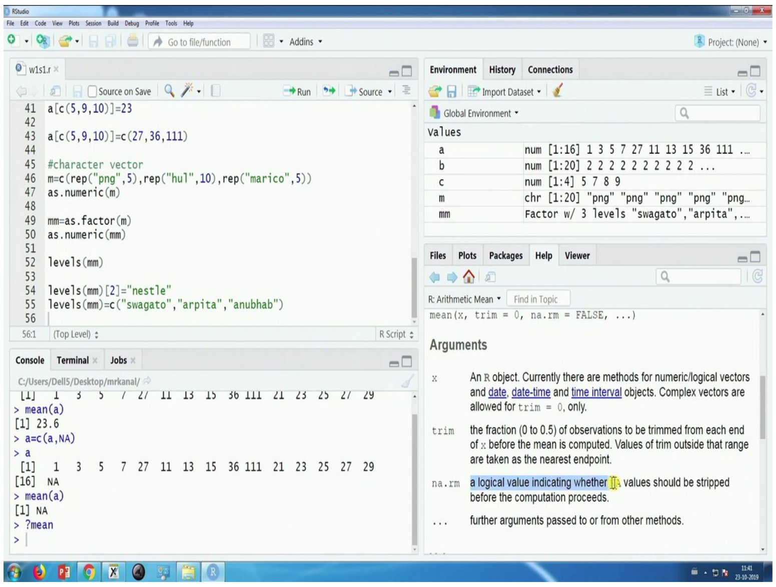
If I can change one level, I can change multiple levels as well. So I can change the whole thing as well. I can change set levels of mm instead of it is hul, marico, Nestle, blah, blah, blah, it will be Swagato, Arpita and Anubhav. So if I click it, and now I enter mm, see the 5

Anubhab and then 10 Swagato and then 5 Arpita comes up so it will change it like anything. So that is what I wanted to just show that you can change it like anything.

So, this is how you can deal with a factor variable. Now, if we can do little bit more on this thing. So, we have shown you till now the repeat function, we have shown you the sequence function and I will show you a few more things on Vector. So, let us say that I want to find out that how I will find out if very basic statistical analysis on the numerical variables that I have till now. So, for example, I have let us say ‘a’ and ‘a’ looks like this 1, 3, 5, 7, 27.

So, there are certain inbuilt functions available also, which can be used on this particular vectors or variables which can help you to find out something basic statistical analysis. For example, I want to find out the mean the basic statistical analysis of a vector is called let us say means, standard deviation if you know, so mean, mean is the average, the arithmetic average of a numerical vector.

(Refer Slide Time: 24:16)



```
a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)

#character vector
m=c(rep("png",5),rep("hul",10),rep("marico",5))
as.numeric(m)

mm=as.factor(m)
as.numeric(mm)

levels(mm)

levels(mm)[2]="nestle"
levels(mm)=c("swagato","arpita","anubhab")
```

```
 [1]   1   3   5   7  27  11  13  15  36 111  21  23  25  27  29
> mean(a)
[1] 23.6
> a=c(a,NA)
> a
 [1]   1   3   5   7  27  11  13  15  36 111  21  23  25  27  29
[16]  NA
> mean(a)
[1] NA
> ?mean
>
```

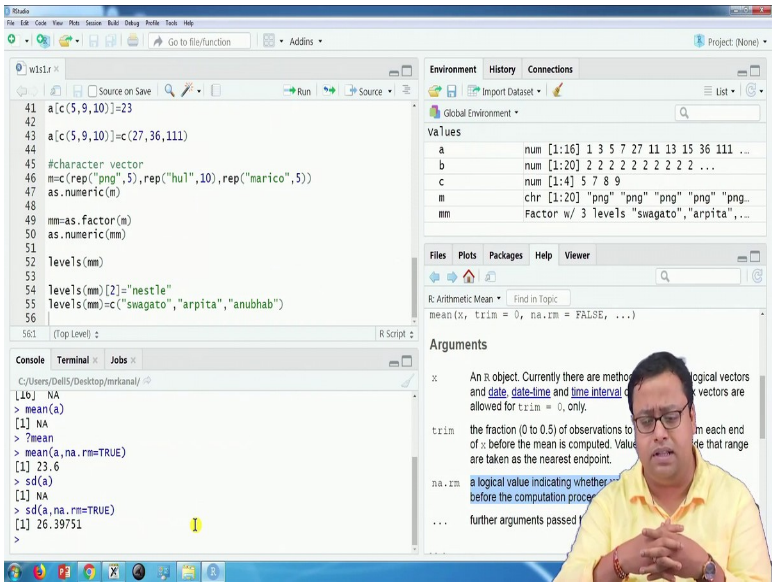
Values	
a	num [1:16] 1 3 5 7 27 11 13 15 36 111 ...
b	num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
c	num [1:4] 5 7 8 9
m	chr [1:20] "png" "png" "png" "png" "png...
mm	Factor w/ 3 levels "swagato","arpita",...

R: Arithmetic Mean

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

x	An R object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for trim = 0, only.
trim	the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.
...	further arguments passed to or from other methods.



```
a[c(5,9,10)]=23

a[c(5,9,10)]=c(27,36,111)

#character vector
m=c(rep("png",5),rep("hul",10),rep("marico",5))
as.numeric(m)

mm=as.factor(m)
as.numeric(mm)

levels(mm)

levels(mm)[2]="nestle"
levels(mm)=c("swagato","arpita","anubhab")
```

```
C:/Users/Dell5/Desktop/mrkanal/
[16]  NA
> mean(a)
[1] NA
> ?mean
> mean(a,na.rm=TRUE)
[1] 23.6
> sd(a)
[1] NA
> sd(a,na.rm=TRUE)
[1] 26.39751
>
```

```
Values
  a      num [1:16] 1 3 5 7 27 11 13 15 36 111 ...
  b      num [1:20] 2 2 2 2 2 2 2 2 2 2 ...
  c      num [1:4] 5 7 8 9
  m      chr [1:20] "png" "png" "png" "png" "png...
  mm     Factor w/ 3 levels "swagato","arpita",...
```

R: Arithmetic Mean

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

x — An R object. Currently there are metho logical vectors and date, date-time and time interval vectors are allowed for trim = 0, only.

trim — the fraction (0 to 0.5) of observations to m each end of x before the mean is computed. Value de that range are taken as the nearest endpoint.

na.rm — a logical value indicating whether before the computation proce

... — further arguments passed

So, you write mean of 'a', that will give you the basic arithmetical average, but often times in a particular vector, there will be missing value. So, you will see that in the data set when we create there are missing values available also. So, let us say that I actually add a missing value in my 'a', I say a=(c,a, NA) that means whatever 'a' values were there, at the end of that NA one missing value, that means not available value will get added. So now if I press a, see wherever a values were there addition of that at the end, there is one single not available value.

So, if there is one single not available value, how the problem comes is if I write mean of a, it will give me NA, so for that particular one single value it is creating a problem, it is a very common mistake that we generally do. It is a very common mistake that when we deal with a vector, we write mean(a) and it does not come. So, if it does not convert, will you do again you can either go to Google or you can search mean.

So, if I just search help of mean it will say that, so there is something called na.rm=false that means this is default. By default it is saying that na.rm is equal to false means na.rm stands for whether I want to remove any illogical value indicating whether NA value should be stripped before the competition process. So, should I remove the NA values before the competition process? And the answer is yes.

In this particular case, yes, default is no, default is false, but in this particular case I want to do that. So, I will write na.rm=true and if I press an enter I will get the mean value. Similarly, I can do it the similar thing for standard deviation as well, I can find out the standard deviation of this particular value as well. So, what will I do? I will do sd, sd stands for standard deviation, sd of a and sd of a is still given me na. I can still do na.rm=true and I will get certain amount of some standard deviation. Similarly, there are other functions which is called median, median is the middle value of any variable when it is ordered in either ascending order or descending order you can find medians you can find on quantiles, 1st quantiles, 3rd quantiles, 2nd quantiles.

And you can also find out probably the percentile values. So some of these particular things you should learn on your own, I would strongly suggest that is something that we will not cover separately, I will strongly suggest you go to Google and search for them and search for the particular function which helps in doing this. Median is easy, mean is easy, sometimes percentile is also something that you can do.

And that is all for vector. In the next session, we will talk about matrix and data frame and we will do a little bit of manipulation also. So, thank you for this particular session. Thank you for attending this particular session I will see you in the next session.