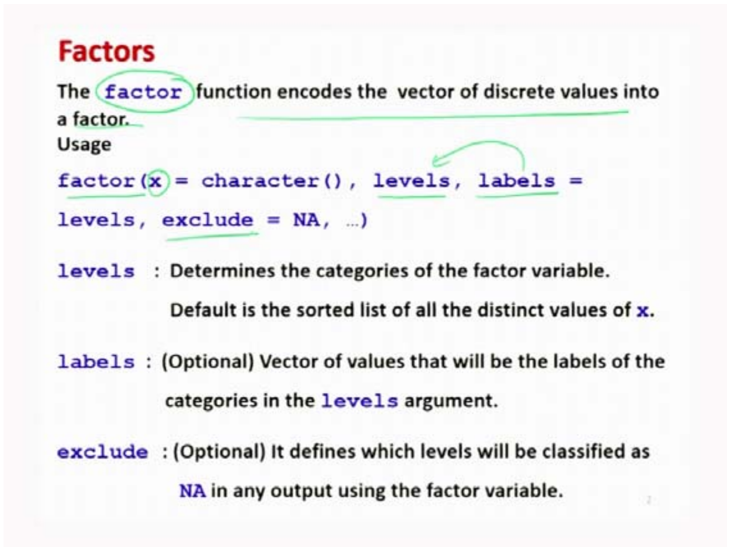
Foundations of R Software
Prof. Shalabh
Department of Mathematics and Statistics
Indian Institute of Technology, Kanpur

Lecture - 34
Factors - Class and Unclass

Hello friends. Welcome to the course Foundations of R Software. You may recall that in the last lecture, we initiated a discussion on the Factors, and we had understood that how factors work and how do they help us. So, now, in this lecture, I will try to give you some more applications and commands for the factors.

For example, if you have got a data vector how can you convert it into a factor and beside that thing there is a very important application of the factors when we are trying to use factors with the command Class and Unclass, right. For the class, in one of the earlier lecture we had briefly talked about it, but now today I am going to give you an application of the class and unclass function using the factors. So, let us try to begin our lecture and try to understand the further topics in this factors.

(Refer Slide Time: 01:08)



Factors

The `factor` function encodes the vector of discrete values into a factor.

Usage

```
factor(x = character(), levels, labels =
levels, exclude = NA, …)
```

`levels` : Determines the categories of the factor variable. Default is the sorted list of all the distinct values of `x`.

`labels` : (Optional) Vector of values that will be the labels of the categories in the `levels` argument.

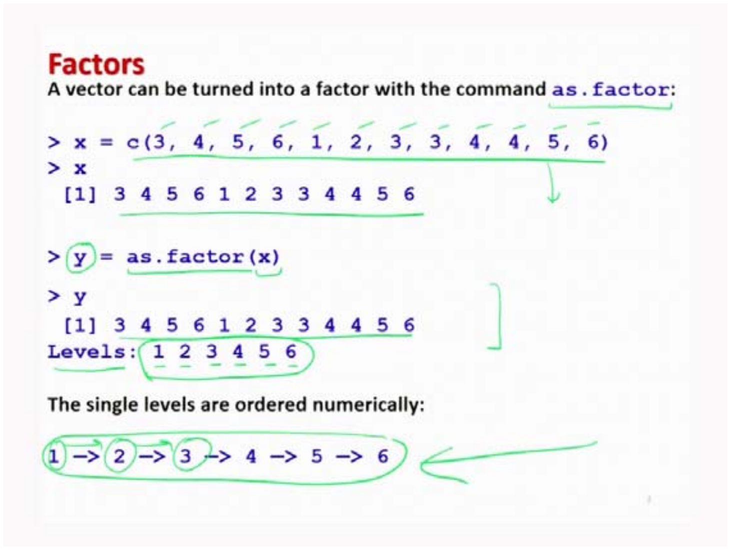
`exclude` : (Optional) It defines which levels will be classified as `NA` in any output using the factor variable.

So, you see you can recall that in the last lecture we had learnt about the command factor, f a c t o r, all in lower case alphabet and this factor function actually encodes the vector of discrete values into a factor. Factor means we had understood in very simple language that it is a categorical variable, right. For example, if I take an example that a student has to be classified as male or female, then suppose male takes value 1 and female takes value 0. So, then the data will look like 1 0 0 1 etc., where 1 means male and 0 means female.

And similarly, if we have some competition in which people are trying to perform and we are trying to give the grades they are excellent, good, better, worse etc. So, they can also be classified as a categories 1, 2, 3, 4. For example, 1 is indicating excellent, 2 is indicating good and so on, right.

So, these are our factors and if you want to use this factor, you have to give the data vector here as x and then you have to specify the levels and then you have to specify the labels. So, levels are going to contain the name that you want to assign to the levels, right. And then you also can handle the NA values using the option exclude.

(Refer Slide Time: 02:27)



Factors

A vector can be turned into a factor with the command `as.factor`:

```
> x = c(3, 4, 5, 6, 1, 2, 3, 3, 4, 4, 5, 6)
> x
 [1] 3 4 5 6 1 2 3 3 4 4 5 6

> y = as.factor(x)
> y
 [1] 3 4 5 6 1 2 3 3 4 4 5 6
Levels: 1 2 3 4 5 6
```

The single levels are ordered numerically:

```
1 -> 2 -> 3 -> 4 -> 5 -> 6
```

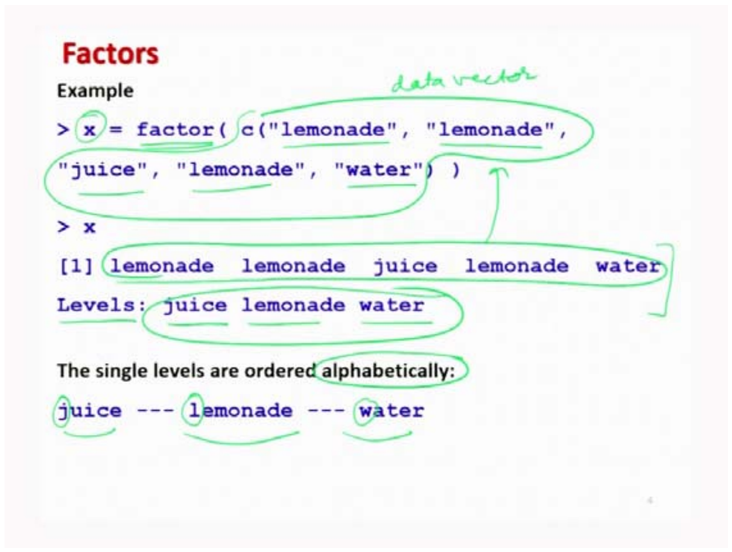
So, after this, I would like to show you couple of things through the example. So, suppose you have got a data vector, numerical data vector and you want to convert it into a factor. So, how to get it done? So, for that the command is as dot factor, a s dot f a c t o r. So, you can see that as dot factor is a similar command as you did in the case of as dot numeric, as dot character etc.

So, let us try to understand that how it works. So, we have considered here a data vector which is consisting of these values which are 3, 4, 5, 6, 1, 2, 3, 3, 4, 4, 5, 6, right. And I want to convert these numerical values as factor. So, factor means we know that it will try to identify the unique numbers out of this data vector x and then it will try to classify them as factors. And it will give the name them as say those unique numbers.

So, if you try to see here I write down here as dot factor and inside the parenthesis I write down the data vector, and whatever is the outcome I am trying to store in a variable y. So, now the outcome looks here like this. So, if you try to see here, it is trying to give here this data which is now in the form of a factor and after that it is trying to give you the levels and this levels had been given the labels as 1, 2, 3, 4, 5, and 6.

So, one point which you have to observe here is that when you are trying to consider a numerical data, then these single levels are ordered numerically. For example, you can see here this. If you try to see here they are arranged like this one, means first and then in an increasing order, then 2, then an increasing order 3, and 4, and 5 and then 6 finally. So, this is how the factor a command work when you are trying to deal with the numeric data and you want to convert it into a factor.

(Refer Slide Time: 04:38)



Factors

Example

```
> x = factor( c("lemonade", "lemonade",
"juice", "lemonade", "water") )
> x
[1] lemonade  lemonade  juice  lemonade  water
Levels: juice lemonade water
```

The single levels are ordered alphabetically:

```
juice --- lemonade --- water
```

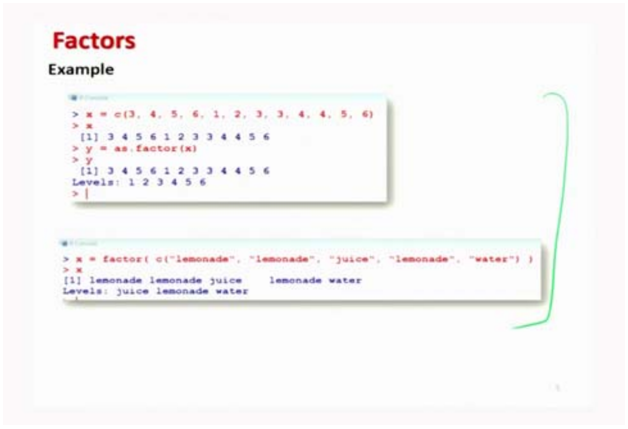
Similarly, if I try to take here one more example where I am trying to consider the data which is in the form of a strings; so, suppose I try to take here a data vector, say here which has values like lemonade, lemonade, juice, lemonade and water. So, you can see here there are 3 possible categories. There are exactly 3 categories actually juice, water and lemonade, right. So, now if you try to factor it, then you can see here what happens. So, this is your here data, data vector, right.

So, if you try to see here I am trying to store its outcome as factor in the value here x. So, this comes out to be here like this. So, we can see here first this data vector is reproduced here just like here this, and after this it has automatically assigned the levels. So, if you try to see here there are three unique values in the data vector which are juice, lemonade, and water, and these values have been assigned here as say levels, juice, lemonade and water.

And if you try to recall that when we had considered this numerical value, then these values were arranged increasing order mean the values of levels arranged in the increasing order. But when we are trying to handle the strings then these single levels are ordered alphabetically, right.

Alphabetically means first the word whose alphabet comes earlier that will appear, and after this the next alphabet that appears in the list of alphabets that will appear and so on. So, you can see here you have here three words juice, lemonade, and water. So, j comes first, then comes l then comes w, right means in the sequence of a, b, c, d up to z. So, juice comes here at the first position, then lemonade, and then water. So, you can see here this levels have been arranged in an alphabetical order, right. So, this is an application of this here factor.

(Refer Slide Time: 06:48)



Factors

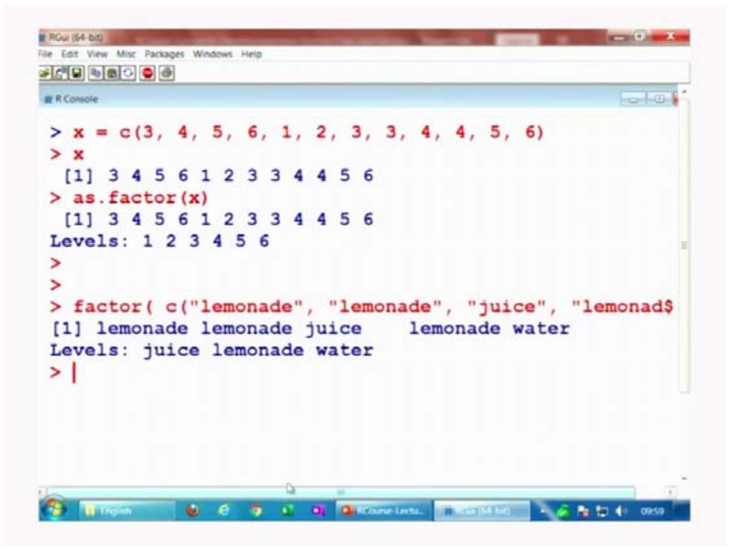
Example

```
> x = c(3, 4, 5, 6, 1, 2, 3, 3, 4, 4, 5, 6)
> x
 [1] 3 4 5 6 1 2 3 3 4 4 5 6
> y = as.factor(x)
> y
 [1] 3 4 5 6 1 2 3 3 4 4 5 6
Levels: 1 2 3 4 5 6
> |
```

```
> x = factor( c("lemonade", "lemonade", "juice", "lemonade", "water") )
> x
[1] lemonade lemonade juice    lemonade water
Levels: juice lemonade water
```

And this is the screenshot of the same operation which I shown you. So, look let me, try to show you these operations on the R console also, so that you become here more confident.

(Refer Slide Time: 07:01)

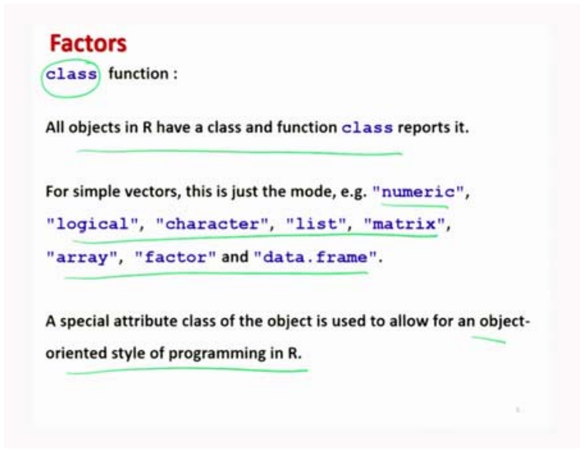


```
> x = c(3, 4, 5, 6, 1, 2, 3, 3, 4, 4, 5, 6)
> x
 [1] 3 4 5 6 1 2 3 3 4 4 5 6
> as.factor(x)
 [1] 3 4 5 6 1 2 3 3 4 4 5 6
Levels: 1 2 3 4 5 6
>
>
> factor( c("lemonade", "lemonade", "juice", "lemonad$
[1] lemonade lemonade juice    lemonade water
Levels: juice lemonade water
> |
```

So, if you try to see here x here, like here, this we can see here this is your here x, here like this. But if you try to convert this x into a factor then you have to write down here as dot factor x, and you can see here that these values have been arranged like this, the different levels have been assigned to this data vector and you can see here this is happening.

Now, similarly if you try to take here this is the string variable means all the values are in the form of a string, then we can see here that this comes out to be here like this, all the values are here and then levels are assigned in an alphabetical order, juice, lemonade, and water, right. So, you can see here, this is a very simple operation which is not difficult, but it gives you good way that how you can convert the numerical and characters into factors.

(Refer Slide Time: 07:40)



Factors

class function :

All objects in R have a class and function class reports it.

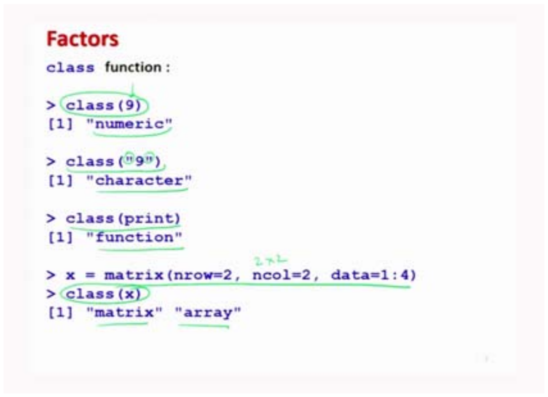
For simple vectors, this is just the mode, e.g. "numeric", "logical", "character", "list", "matrix", "array", "factor" and "data.frame".

A special attribute class of the object is used to allow for an object-oriented style of programming in R.

After this I would like to give you one more application using the factor command. This is about the class and the opposite of class will be unclass. So, you can recall that in one of the earlier lecture I had briefly explained you about the this function class, c l a s s, all written in the lower class alphabets. And we had understood that this class actually gives us the information that what is the class of the object in the R Software. Means every object in the R Software has got a class, and this function reports the class of that object.

And we have different classes like as here numeric, logical, character, list, matrix, array, factor and data frame, right. And these type of this class objects they are used essentially in the object oriented programming in R. Well, we are not considering here that object oriented programming, but this is for those who are quite familiar and expert in the programming.

(Refer Slide Time: 08:50)



Factors

class function :

```
> class(9)
[1] "numeric"

> class("9")
[1] "character"

> class(print)
[1] "function"

> x = matrix(nrow=2, ncol=2, data=1:4)
> class(x)
[1] "matrix" "array"
```

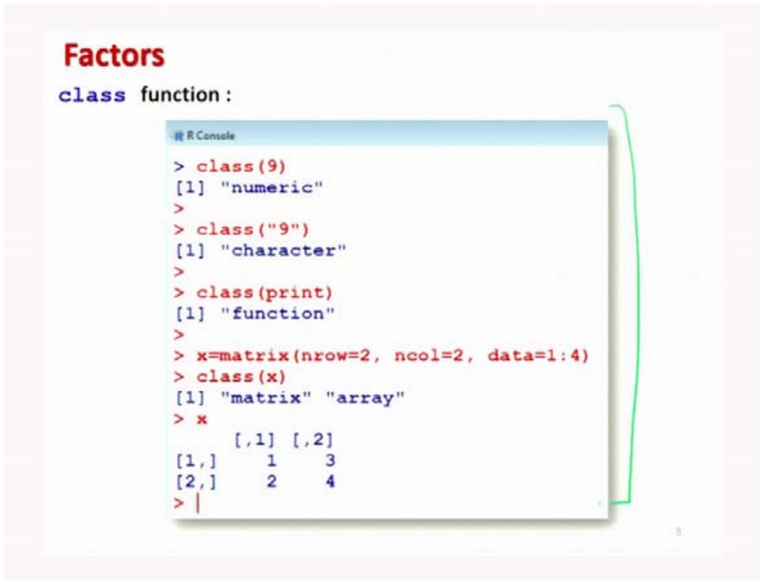
2×2

Well, firstly, let us try to understand that what is the meaning of this class and how it gives us the information. So, if you try to take here a number say 9 and if you try to find out its class then I have to write down here class, and inside the parenthesis 9 and it will give you the class as "numeric". Now, I try to convert this "9" into a character. So, you can see here I have written here double quotes, and then I write the value 9 inside the double quote. So, that it now becomes a character.

So, it comes the class of this correct comes out to be here a character. And similarly if you try to take any function like as print, then the class of the print is like here "function", right. You just write class and within parentheses write down the name of the function.

And similarly if you try to take here a matrix so, if you try to see here that I have taken a matrix here of order 2 by 2 in which the data is 1, 2, 3, 4. And now if you try to see what is the class of this x? This is "matrix" as well as "array". You know that matrix is a special type of array, right in which the values are arranged in rows and columns in a particular way and the that way is actually according to the theory of matrix, ok.

(Refer Slide Time: 10:03)



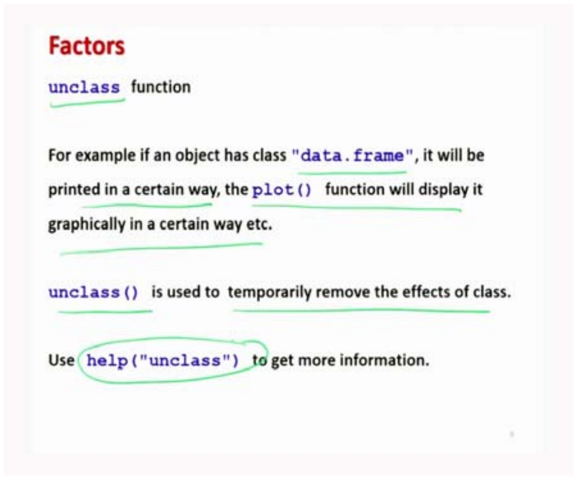
Factors

class function :

```
> class(9)
[1] "numeric"
>
> class("9")
[1] "character"
>
> class(print)
[1] "function"
>
> x=matrix(nrow=2, ncol=2, data=1:4)
> class(x)
[1] "matrix" "array"
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> |
```

So, you can see here this is the outcome here, right.

(Refer Slide Time: 10:07)



Factors

unclass function

For example if an object has class "data.frame", it will be printed in a certain way, the plot() function will display it graphically in a certain way etc.

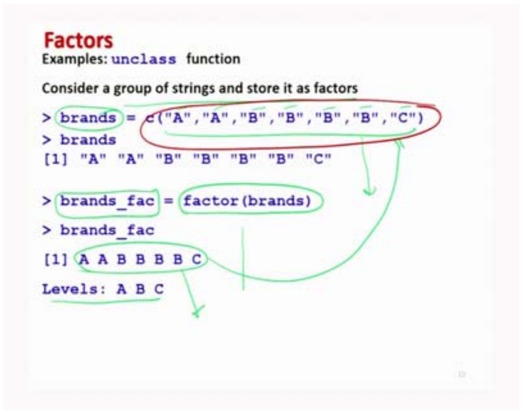
unclass() is used to temporarily remove the effects of class.

Use help("unclass") to get more information.

After this I try to give you here a command and its utility using the factors. So, first we try to understand this command and this command is unclass, u n c l a s s. So, this is a function, right which who can say in the simple words that is just opposite to the class. When you try to something class, then unclass will remove the effect of the class temporarily. So, for example, if you have an object like data frame so, this data frame is going to be printed in a certain way and the plot function will display it graphically in a certain way, right.

So, what is this data frame? We are going to consider it in a forthcoming lectures. But when you try to use this unclass function then it will temporarily remove the class of the effect of the class, right. If you want to have more information on the unclass function I would request to look into the help and try to have a look, right.

(Refer Slide Time: 11:07)



Factors

Examples: unclass function

Consider a group of strings and store it as factors

```
> brands = c("A","A","B","B","B","B","C")
> brands
[1] "A" "A" "B" "B" "B" "B" "C"

> brands_fac = factor(brands)
> brands_fac
[1] A A B B B B C
Levels: A B C
```

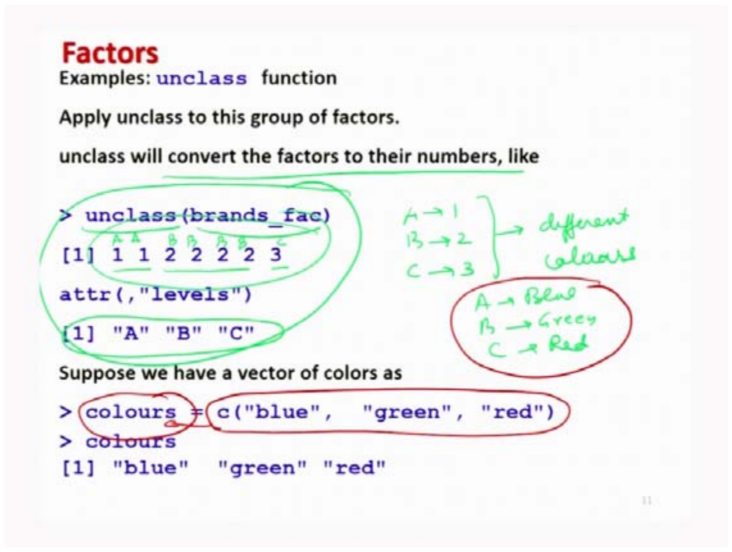
But anyway I will try to take here an example, and through that example I will try to explain you what is the use of this unclass function, right. Suppose, ok I am going to continue with this example in the next three slides so, I will try to explain you, but you please do not try to lose the track, that is my request.

So, if you see first of all I am trying to consider here a group of strings and I try to store it as a factor. So, let us try to suppose there are three brands of some clothing’s and I try to categorize them as a A, B, C and the data on those brand is a character which is stored here in the data vector whose name I have given here as a brands.

So, this brands are stored like as a A, A, B, B, B, B and C, right. So, now, I try to convert this brands into factor. So, I try to use here the factor command and then inside the parenthesis I write here brands, and whatever is the outcome I try to store it in a new variable brands underscore fac. So, that means, brands are factored, right.

And this outcome will look like this. You know that there are three unique strings in this A, B, C, just like we had considered the example of juice, water, and lemonade. So, now this A, B, C will be assigned as levels and the data on this brands which is reproduced here also from here, right. So, now, you see in this slide we have simply considered the brands and we have converted them into factors, right.

(Refer Slide Time: 12:45)



Factors

Examples: unclass function

Apply unclass to this group of factors.

unclass will convert the factors to their numbers, like

```
> unclass(brands_fac)
[1] 1 1 2 2 2 2 3
attr(,"levels")
[1] "A" "B" "C"
```

Suppose we have a vector of colors as

```
> colours = c("blue",  "green", "red")
> colours
[1] "blue"  "green" "red"
```

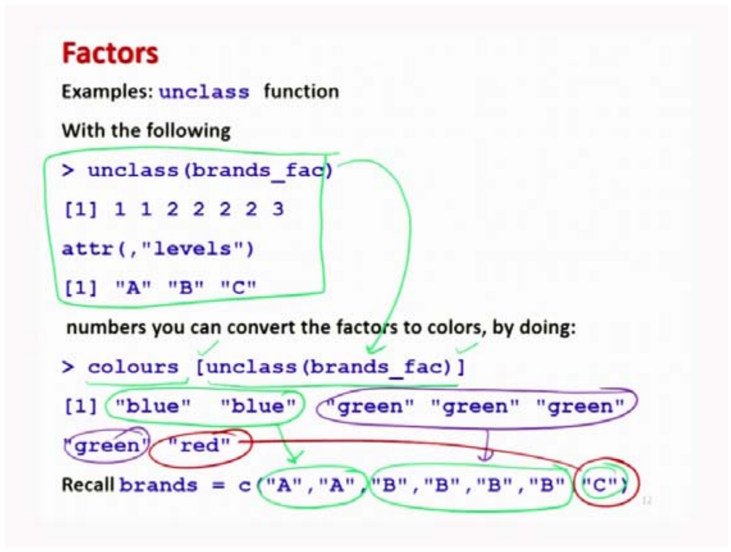
Now, in case if you try to use the unclass function, then unclass will convert the factors to their numbers, right. For example, you can see here your this value here is A, A, B, B, B, B and C, and if you try to unclass it, on this outcome brands underscore fac, then A is converted into 1, B is converted into 2 and C is converted into 3. You can see here is a 1, 1, 2, 2, 2, 2 and 3. So, this was actually here A, A, B, B, B, B and here C.

So, you can see here that these levels which there earlier that levels are the same, you can see. The levels are still the same as A, B, C, but those values are converted into numbers, ok. Now, what is the use? So, suppose I want to give these brands A, B, C I want to assign different colors, right and suppose I try to assign say A blue colour, B green colour, and C as red colour. Now, how to get it done?

So, if you try to understand it in very simple words now, I can explain you that whatever is your data here, you want to replace A, B and C by their colours like as this that A, B, C are replaced by these colors. A is replaced by blue, B is replaced by green, and C is replaced by red.

So, what I try to do here is the following. That I try to give here, I try to create here a data vector of this colors like as here blue, green and red which are here like this, right and I try to store them in a variable colours, c o l o u r s.

(Refer Slide Time: 14:38)



Factors

Examples: `unclass` function

With the following

```
> unclass(brands_fac)
[1] 1 1 2 2 2 2 3
attr(,"levels")
[1] "A" "B" "C"
```

numbers you can convert the factors to colors, by doing:

```
> colours [unclass(brands_fac)]
[1] "blue"  "blue"  "green" "green" "green"
"green" "red"
```

Recall `brands = c("A","A","B","B","B","B","C")`

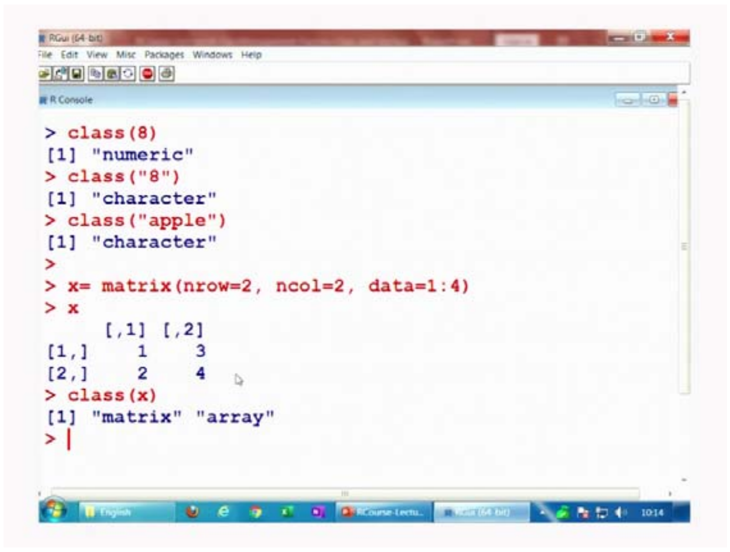
Now, if you try to see here what is happening here now? You can recall this thing, right. This is the same output which you had got here. So, I have just copied it here, so that you can see them on the same slides. Now, I try to use here a command colors, and inside the square bracket I write down here unclass brands underscore fac that this variable. And now if you try to see what happens.

Now, your brands were earlier like as first two were A, next 4 were B and the last was C. But now they are converted into colours. A is now converted to blue. So, you can see here there are two blues which is corresponding to these two A's. Then, now there are four green and these four greens are corresponding to this category B or the brand B. And finally, the last value here C, this is replaced by here red.

So, that is the advantage that if you want to change something here you can do these types of operation using the this class unclass functions and then you try to extend it more. So, if you try to see here the main utility of class and unclass function is essentially that if you are trying to consider here a data which is consisting of strings like this one.

Then this will have some string values and then unclass function is trying to convert it into some numbers, right. So, if you try to show you all these operations on the R console here, then it will be then you will understand it very easily, right.

(Refer Slide Time: 16:25)

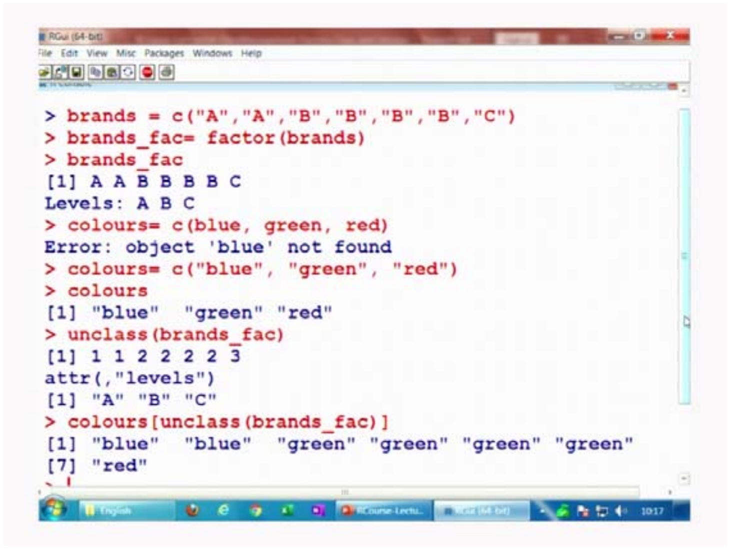


```
> class(8)
[1] "numeric"
> class("8")
[1] "character"
> class("apple")
[1] "character"
>
> x= matrix(nrow=2, ncol=2, data=1:4)
> x
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> class(x)
[1] "matrix" "array"
> |
```

So, if I try to find out here the class of the value, numerical value here say 8, this will come out to be here “numeric”. And if I try to convert this “8” into a character by writing the 8 into the double quotes, the class will come out to be the “character”. And similarly if I try to find out here the class of a string like as here “apple” which we have used many times in the earlier lecture, this will also come out to be here see here “character”.

And similarly if I try to create here a matrix with nrow equal to 2, ncol equal to 2, and data is equal to 1 to 4, you can see here this matrix will come out to be here like this. And if you try to find out here the class of this matrix here x, this will come out to be here matrix and array like this, right, ok.

(Refer Slide Time: 17:16)



```
> brands = c("A","A","B","B","B","B","C")
> brands_fac= factor(brands)
> brands_fac
[1] A A B B B B C
Levels: A B C
> colours= c(blue, green, red)
Error: object 'blue' not found
> colours= c("blue", "green", "red")
> colours
[1] "blue"  "green" "red"
> unclass(brands_fac)
[1] 1 1 2 2 2 2 3
attr(,"levels")
[1] "A" "B" "C"
> colours[unclass(brands_fac)]
[1] "blue"  "blue"  "green" "green" "green" "green"
[7] "red"
```

Now, let us try to consider the example of this class unclass. So, I have given here the brands in three values A, B and C which are a strings. And if I try to convert them into factors, so this will be here brands underscore factor and it will here factor of here say here factor brands which I have used here a variable. So, you can see here this brands underscore fac, this will come out to here like this, right.

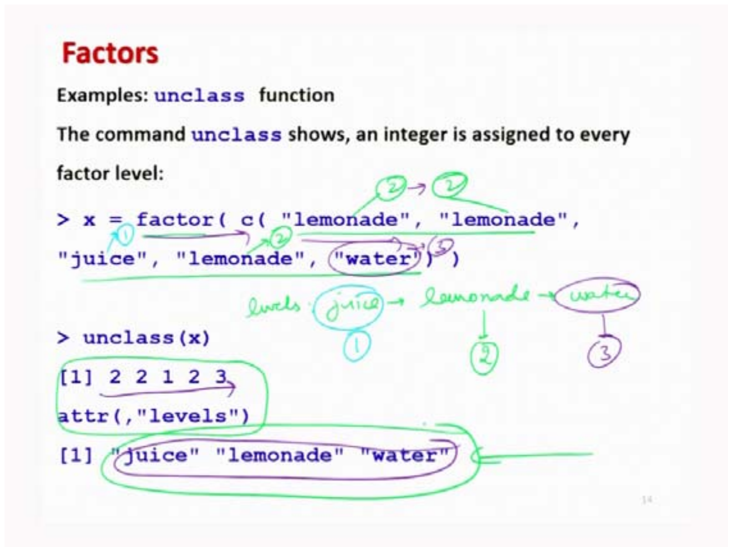
Now, I want to give it here colour. So, let me try to write down here, colours is equal to say here c, blue, green and say red. And so this is now my here colour. And you can see here why it is trying to give you an error here because you have not given it in the double quotes. So, let me try to give you give this value inside the double quotes because these are the strings, and then you will see that these mistakes will not happen. And surely, when you are trying to do such programming these things are going to happen, right

So, now you can see here this colours is like here like this. And after this I try to, first I try to use here the function on class and then I will try to use here this colours on it. So, if I try to write down here this brands underscore fac and I try to unclass it, you will see here it will come out to be here like this, right.

And in case if I try to operate the colours on this unclass brands underscore fac, this will come out to be here like this, right. So, you can see here I am just trying to give put it on the same screen. So, you can see here whatever here is A, this is now become here blue because blue is at the first place. So, this is here at blue. Then in the data vector you have here 4 Bs. Please try to have a look where I am trying to highlight it.

And then we have given it the second value in the colours vector here is green. So, next four values are here green. And then the last value here is C, and the last value in the colours vector is here red. So, the last value here 3, this becomes here red. So, now, you can see here that this is a very important function which is going to be useful when you are trying to deal with the real life data, right.

(Refer Slide Time: 19:34)



Factors

Examples: unclass function

The command unclass shows, an integer is assigned to every factor level:

```
> x = factor( c( "lemonade", "lemonade",
"juice", "lemonade", "water") )

> unclass(x)
[1] 2 2 1 2 3
attr(,"levels")
[1] "juice" "lemonade" "water"
```

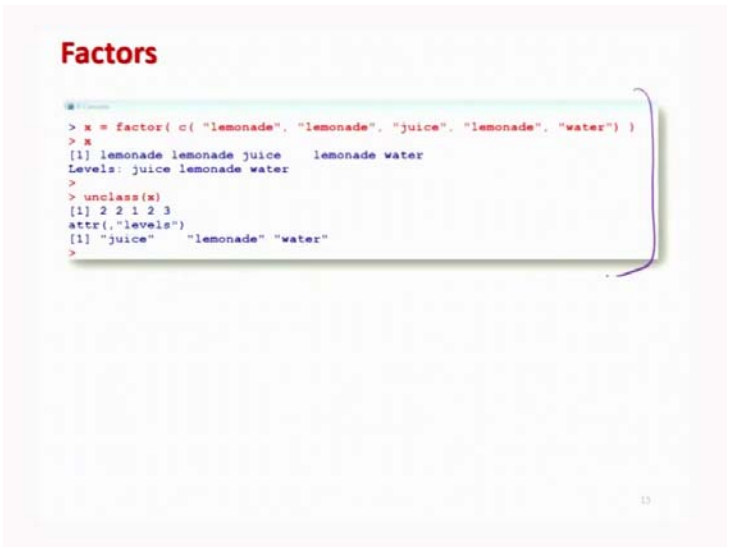
Similarly, just to make you more confident here, let me try to take here some more example. And I will try to simply show you that how you can use the class and unclass function to convert a data vector which is obtained by factor on a string variable and then how it is converted into the numbers.

So, let me try to take here this example where I am trying to take here the factor. Say which is have the data lemonade, lemonade, juice, lemonade and water. This is the same example that we did in the beginning, and this, and in case if you try to see the outcome the outcome will look like here this data, and then this juice, and then lemonade, and then here water they will be the levels which are arranged in the alphabetical order.

So, now if you try to unclass it, then you can see here this outcome will become here like this. The data will become here 2, 2, 1, 2, 3. So, this lemonade is at place number 2. So, this lemonade, this lemonade they will be converted as number 2, and this lemonade we will be converted into 2.

And after this here juice, juice is at the value 1. So, wherever is the juice here in the data vector x that will be converted into 1. And then finally, you have here the see here water, water is at place number 3. So, this water will be replaced by number here 3. And then whatever is the outcomes 2, 2, 1, 2 and 3, this is obtained here 2, 2, 1, 2, 3. And the levels are the same, juice, lemonade, and water. The only thing is this now this data has been converted into numbers, right.

(Refer Slide Time: 21:21)

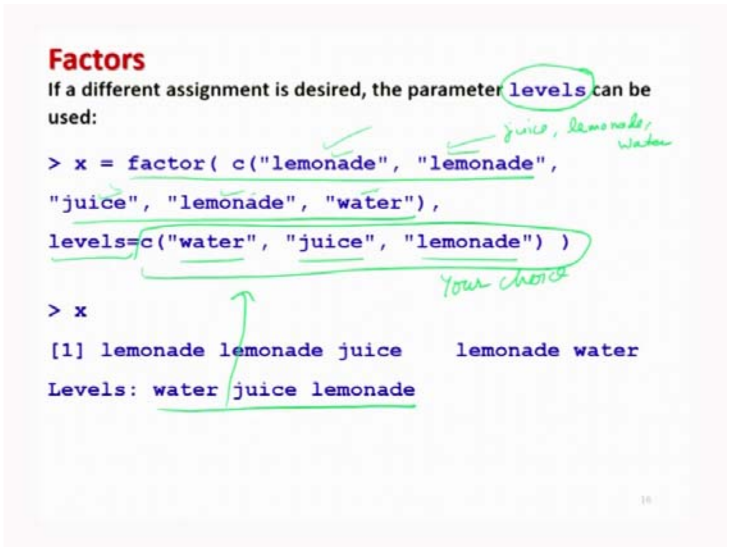


Factors

```
> x = factor( c( "lemonade", "lemonade", "juice", "lemonade", "water") )
> x
[1] lemonade lemonade juice    lemonade water
Levels: juice lemonade water
>
> unclass(x)
[1] 2 2 1 2 3
attr(,"levels")
[1] "juice"    "lemonade" "water"
>
```

And this is here the screenshot of the same operation which I shown you. I will try to show you to on the R console also.

(Refer Slide Time: 21:28)



Factors

If a different assignment is desired, the parameter `levels` can be used:

```
> x = factor( c("lemonade", "lemonade",
"juice", "lemonade", "water"),
levels=c("water", "juice", "lemonade") )

> x
[1] lemonade lemonade juice    lemonade water
Levels: water juice lemonade
```

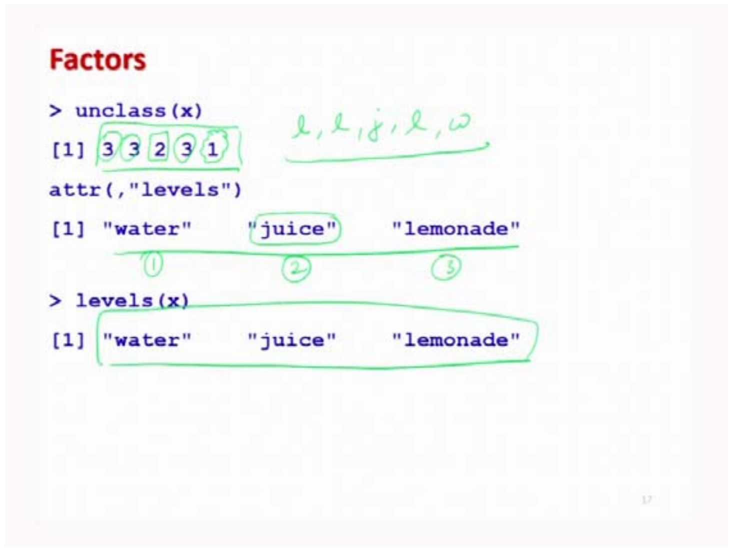
And now if you try to see in all the examples whatever we have considered, the name of the levels is assigned automatically, right. For example, you can see here in this case the levels are automatically assigned as juice, lemonade, and water in an alphabetical order. But suppose you want to give the levels of your choice, right.

So, remember earlier we had here juice, and then lemonade, and then water in an alphabetical order, now you want to give it new combination. This combination can be see here levels equal to here water, juice, and lemonade, right. So, this is not in the alphabetical order, but this is your choice.

So, now how to give the levels of your choice or how to change the levels which are obtained by the default operation? So, in order to do this thing you simply try to do the same operation that you try to factor the data say lemonade, lemonade, juice, lemonade, and water. And after that you try to give here one more command here levels equal to whatever levels you want to give, give here, right.

So, if you want to have a different assignment for the levels, try to use the parameter levels, l e v e l s, and try to give the name of the new variables in the order you want under this levels. And after this you will see here this levels will become here water, juice, lemonade, right. So, this is here like this. And it does not make any difference because you are simply trying to give it here a levels, that is all, right.

(Refer Slide Time: 23:14)



Factors

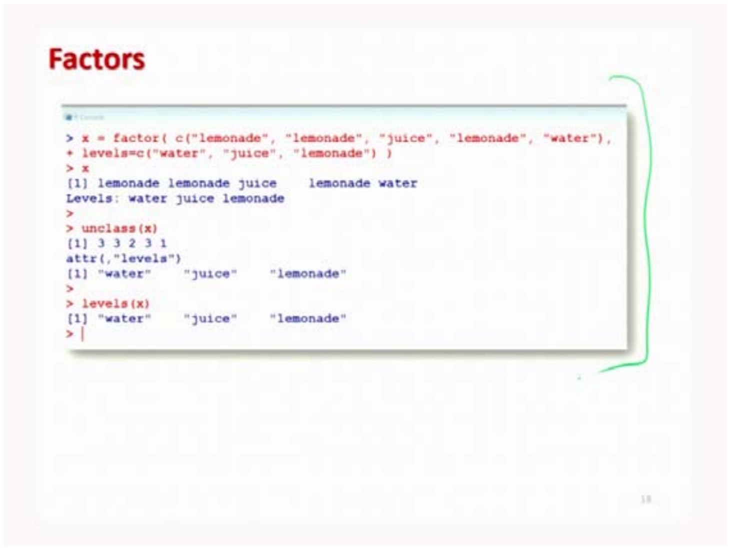
```
> unclass(x)
[1] 3 3 2 3 1
attr(,"levels")
[1] "water"    "juice"    "lemonade"
> levels(x)
[1] "water"    "juice"    "lemonade"
```

So, and now in case, if you try to unclass it, then you will see here. Now, the new levels are going to be water, juice, and lemonade, and the data will remain as here 3, 3, 2, 3, 1. Now, if you try to see here this is exactly in the same way. Means now in case if you try to look at this outcome, and suppose you want to get your data back.

So, now, this is here 3, right, 3, 3; so, 3. So, water is at 1 place, juice here is at 2 place and lemonade is at place number 3. So, now, this is here 3, 3, so that means, this is here lemonade, lemonade. And then you have here 2, so 2 here is juice, so then you have here juice. And then you have here once again 3, so 3 here is lemonade. And then you have here 1, so 1 here is water.

So, now, can you verify whether your original data was like this lemonade, lemonade, juice, lemonade and water, try to see, lemonade, lemonade, juice, lemonade and water. So, data is not changing, right only the levels are going to be change. And when you and once you try to unclass it, the new levels will appear in the data. And in that case if you try to find out the levels of this x, they will be coming out here water, juice and lemonade.

(Refer Slide Time: 24:37)

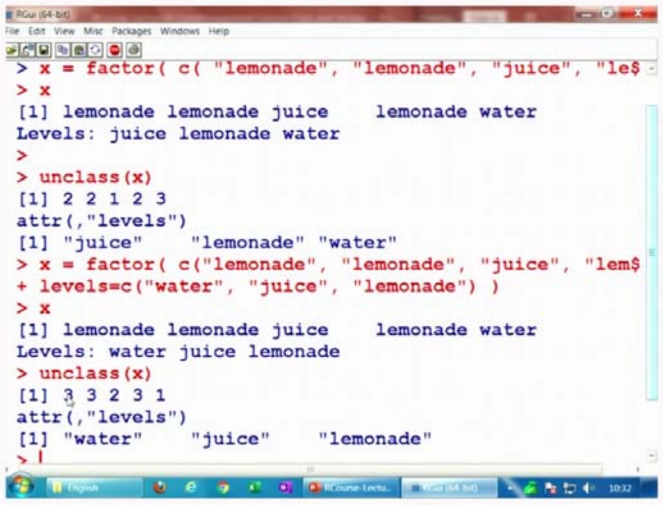


Factors

```
> x = factor( c("lemonade", "lemonade", "juice", "lemonade", "water"),
+ levels=c("water", "juice", "lemonade") )
> x
[1] lemonade lemonade juice    lemonade water
Levels: water juice lemonade
>
> unclass(x)
[1] 3 3 2 3 1
attr(,"levels")
[1] "water"    "juice"    "lemonade"
>
> levels(x)
[1] "water"    "juice"    "lemonade"
> |
```

So, this is how you can make different type of operations here. And this is here the screenshot of the same operation. And first let me try to show you this outcome on the R console, and then I will try to show you one more operation, right.

(Refer Slide Time: 24:52)



```
> x = factor( c( "lemonade", "lemonade", "juice", "le$
> x
[1] lemonade lemonade juice    lemonade water
Levels: juice lemonade water
>
> unclass(x)
[1] 2 2 1 2 3
attr(,"levels")
[1] "juice"    "lemonade" "water"
> x = factor( c("lemonade", "lemonade", "juice", "lem$
+ levels=c("water", "juice", "lemonade") )
> x
[1] lemonade lemonade juice    lemonade water
Levels: water juice lemonade
> unclass(x)
[1] 3 3 2 3 1
attr(,"levels")
[1] "water"    "juice"    "lemonade"
> |
```

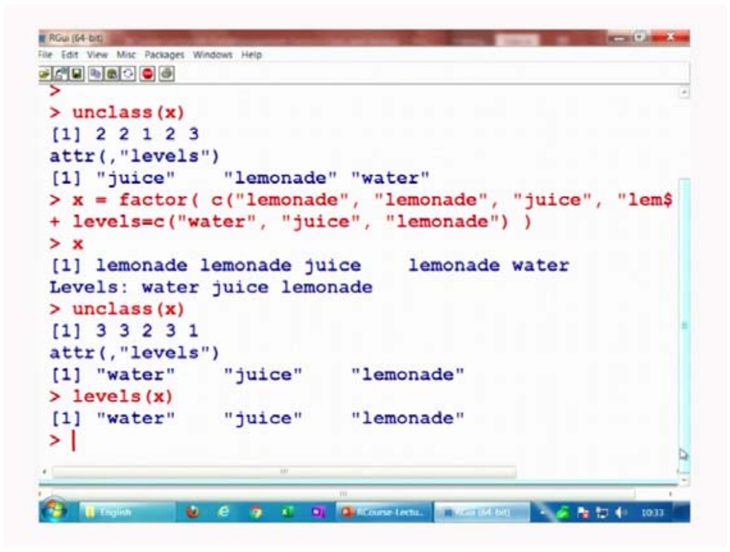
So, if I try to take here this data here x as factor here like this. So, you can see here now I try to unclass it. So, at this output I have stored in here x. So, you can see here. Now, this levels are changed and accordingly these values are now converted into some numerical values, right. So, after that I try to change the levels here. So, I try to write down here this a factor of the same data, but I also try to add here this option here levels. So, if you try to see here, I have added here like this and yeah. So, I write down here factor and earlier I had taken only the data, but now I try to add here levels equal to c, and then if you try to see here what type of data I get here, right.

So, now you can see here in earlier x levels were juice, lemonade, and water, and now I am trying to replace it by water, juice, and lemonade. And if you try to see the new outcome of here x here, the levels is now water, juice and lemonade, right. So, earlier it was juice, lemonade and water, right.

And now in case if you try to unclass now once again this outcome x, then you can see here now this is remember you are observing here 2, 2, 1, 2, 3, right. But now once you try to unclass it, you are getting here 3, 3, 2, 3, 1, right. Please try to observe here I am trying to highlight, right.

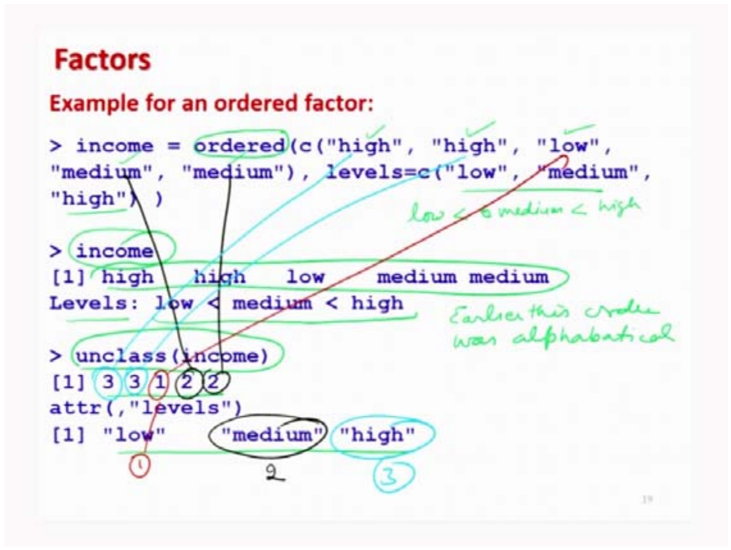
So, you can see here the earlier which was lemonade, lemonade, juice, lemonade and water that was coded as 2, 2, 1, 2, 3, but now you have given the new levels as water juice and lemonade. So, your this data lemonade, lemonade, juice, lemonade and water this is now coded as 3, 3, 2, 3, 1, right. So, you can see here these are not very different options, very difficult operation.

(Refer Slide Time: 26:46)


```
>
> unclass(x)
[1] 2 2 1 2 3
attr(,"levels")
[1] "juice"    "lemonade" "water"
> x = factor( c("lemonade", "lemonade", "juice", "lem$
+ levels=c("water", "juice", "lemonade") )
> x
[1] lemonade lemonade juice    lemonade water
Levels: water juice lemonade
> unclass(x)
[1] 3 3 2 3 1
attr(,"levels")
[1] "water"    "juice"    "lemonade"
> levels(x)
[1] "water"    "juice"    "lemonade"
> |
```

And if you once again if you try to obtain here the levels of this is a new x you can obtain here, this is the water, juice and lemonade. And the levels is also given you here the same option water, juice and lemonade, right.

(Refer Slide Time: 27:00)



Factors

Example for an ordered factor:

```
> income = ordered(c("high", "high", "low",
"medium", "medium"), levels=c("low", "medium",
"high") )

> income
[1] high   high   low    medium medium
Levels: low < medium < high

> unclass(income)
[1] 3 3 1 2 2
attr(,"levels")
[1] "low"    "medium" "high"
```

So, after this I try to give you here the last example where we I am trying to use here the command ordered. So, whenever you see you have some data which is arranged in some order, suppose if I try to see here the temperature, temperature is say low, medium and high, right. And suppose you are trying to get the data like this, you of temperature like this high, high, low, medium and medium, right, and you try to define the levels here as say low medium and high.

So, they are actually ordered. They are ordered in the sense that low temperature is indicated by low, which is less than the medium temperature, which is indicated by medium and this is smaller than the high temperature which is indicated by the wood or string high.

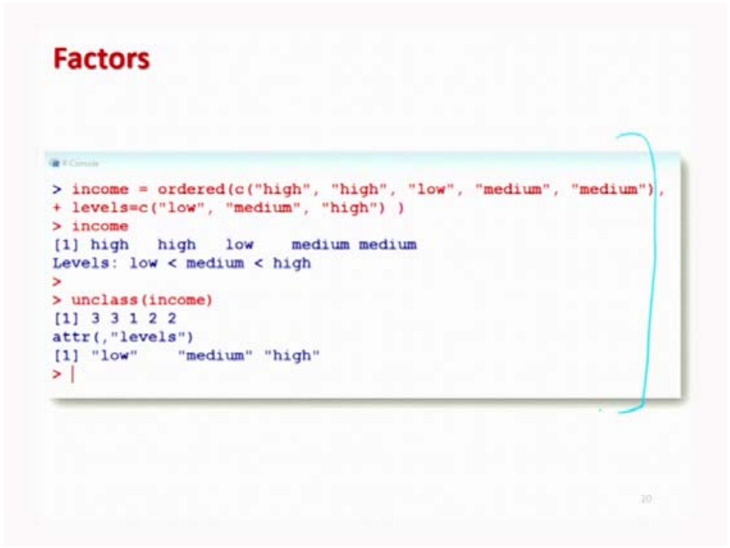
So, now what you will happen here? That you want to factorize it. So, you use here the command ordered, o r d e r e d, right. So, in this case, the same process what happened under the factor command will also happen here, but your levels are going to be ordered, right. So, you can see here, the data comes out to be here the same as high, high, low and medium, medium, but the levels are here like this low, less than medium, less than high.

So, remember, means earlier your these levels which were in the form of some string, they were arranged in an alphabetical order, but now they are arranged in some increasing order or decreasing order, whatever you are trying to say. So, this is like as a low less, than medium less, than high. And earlier this order was alphabetical. So, this is what you have to keep in mind, right.

And now in case if you try to use the command here unclass over the same output income, then you get here 3, 3, 1, 2, 2 and you can see here this is again the levels are the same low, medium, and high, right. If you try to see here this low is indicated by 1. So, this is here, and this is the same data here low.

And then there is here medium, medium is indicated by 2. So, these two values they are the medium. They were here, this was here, they were here, like this medium and here this medium and then finally, you have here say high, high is indicate by here 3, and this is here 3 and 3, and this is here this high and this high, right. So, this is how you can do all such operation without any problem.

(Refer Slide Time: 29:45)

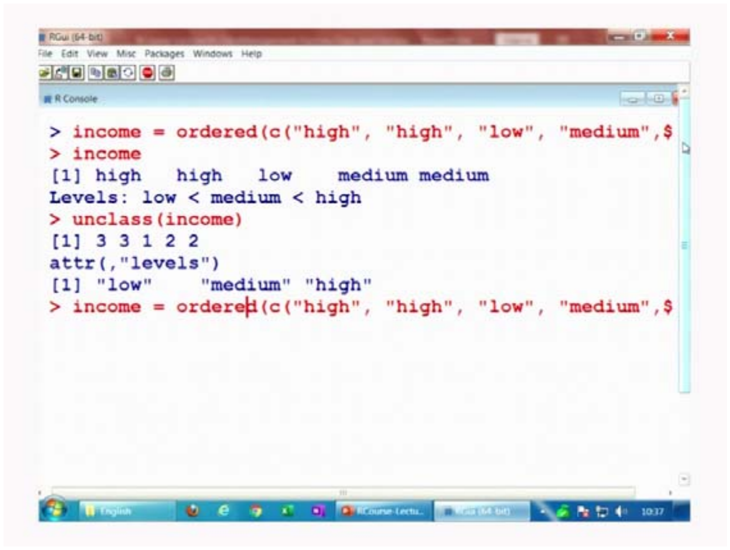


Factors

```
> income = ordered(c("high", "high", "low", "medium", "medium"),
+ levels=c("low", "medium", "high") )
> income
[1] high   high   low    medium medium
Levels: low < medium < high
>
> unclass(income)
[1] 3 3 1 2 2
attr(,"levels")
[1] "low"    "medium" "high"
> |
```

And you can see here this is the screenshot of the same operation, but let me try to show you these things on the R console, so that you become more confident, right. So, you can see here this is my here the data see.

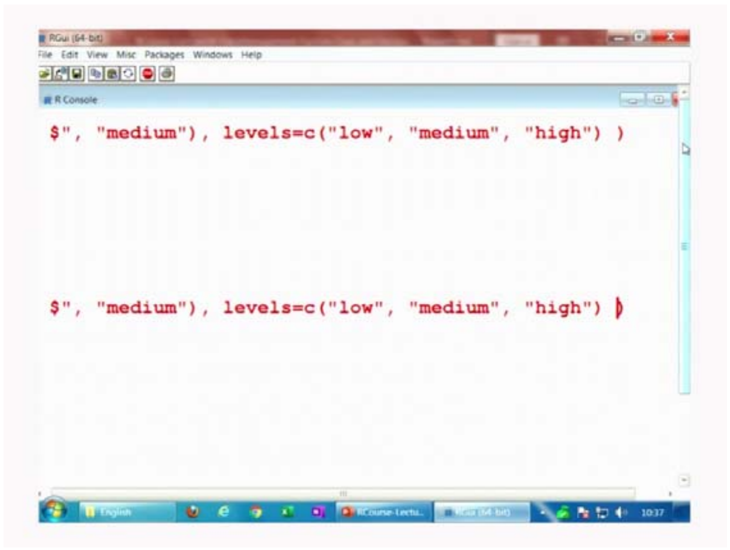
(Refer Slide Time: 30:00)



```
> income = ordered(c("high", "high", "low", "medium",$
> income
[1] high   high   low    medium medium
Levels: low < medium < high
> unclass(income)
[1] 3 3 1 2 2
attr(,"levels")
[1] "low"    "medium" "high"
> income = ordered(c("high", "high", "low", "medium",$
```

And this has been ordered, so the outcome looks here like this income. And if you try to unclass it, like this, it will come here like this income.

(Refer Slide Time: 30:13)



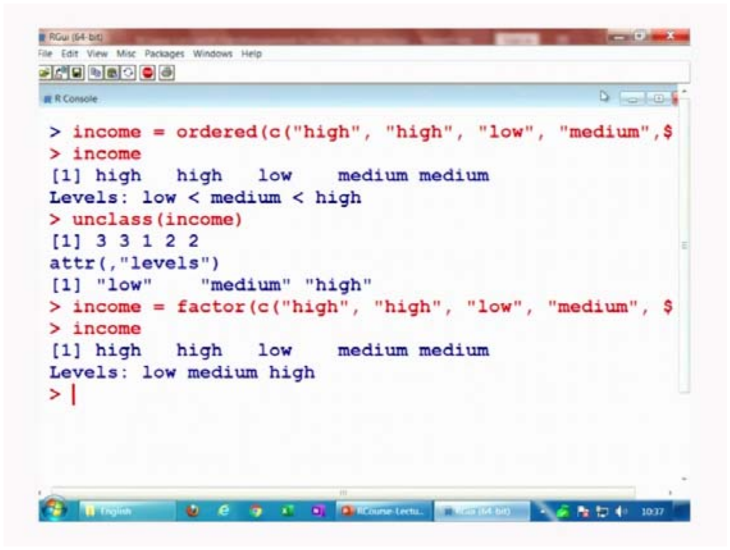
```
$", "medium"), levels=c("low", "medium", "high") )




$", "medium"), levels=c("low", "medium", "high") )
```

And if you want to see what will really happen if you simply try to use here the factor command, so, we try to see just before you I am trying to replace the same order command by factor and if you try to see what happens here income.

(Refer Slide Time: 30:21)



```
> income = ordered(c("high", "high", "low", "medium",$
> income
[1] high   high   low    medium medium
Levels: low < medium < high
> unclass(income)
[1] 3 3 1 2 2
attr(,"levels")
[1] "low"    "medium" "high"
> income = factor(c("high", "high", "low", "medium", $
> income
[1] high   high   low    medium medium
Levels: low medium high
> |
```

You can see here, means earlier it was like as here low less than medium, less than high, but now there is no like ordering, but by chance it is in the like is here simple simply like here low, medium and high, right. But there is no ordering here, ok. So, now we come to an end to this lecture. And you can see here that we had learned very interesting application of this factor when it is combined with the class and unclass functions.

And earlier when we had done the class command, I had explained you very briefly that what it is trying to represent, but then how it is going to be useful I wanted to club it with the factor application. So, that is why I had not explained you earlier, but now I explained here in this lecture.

So, I hope you have understood it. And yeah, in order to understand it in a better way please try to have a revision of this lecture. And try to note down this data value with your own hand, with your own pen, on your own paper. Because when I am trying to speak here low, medium, high or say juice, lemonade, water, this looks very trivial unless and until you understand that what is the difference between juice, lemonade, and water; and water, lemonade and juice.

So, once you try to type the command yourself and then try to execute it, and then always try to compare the outcomes that when you are trying to change the levels and when the default levels are coming, then what is happening in the nature of levels that how they are reported, in the nature of the data that is converted etcetera. So, unless and until you try to observe that how the outcomes are changing when you are trying to make such a small changes in the command, you cannot understand that how R is working which is very important for you to understand, which I always say.

So, why do not you try to take some example and try to create your examples yourself with factor and say ordered both. And try to operate it on the R Software and try to see how do you get the outcome, and are you getting the same outcome which you expected or there is some change if there is some change try to see why this change is coming. So, now, we stop here. And it is your turn to take some example, try to practice it. And I will see in the next lecture with some more new topic.

Till then, goodbye.