

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

**Lecture - 30**  
**Lists**

Hello friend. Welcome to the course Foundations of R Software and you can recall that in the last couple of lectures we had talked about different topics related to the data handling and data management. So, in this lecture also we are going to talk about a new topic, that is about List.

So, now you know that in R software, we have different types of observations, for example, they are numerical values, they are some characters etc. and when you want to combine them together, combine them means you want to create a list of all such aspects all such objects, then how to get it done.

So, list is an option by which you can create a list of different types of objects in the R software and then you can handle it. So, what is this list first we try to understand once you understand what is the list and how it is created then after that we will try to have different types of operation. For example, if you have created a list then how are you going to extract a particular element in the list, how are you going to extract a sub list from the list and how are you going to make different types of data manipulations in the list, these are the different topics which we are trying to learn here, ok.

Before we try to understand what is here a list, let me try to ask you a very simple question. Try to consider a situation that happens in our home, suppose your mom ask you that, ok, please go to the market and bring something and she tells you ok, bring some milk, bring some cloth, bring some vegetables etc. etc. what is your first reply? You simply try to say mom please try to prepare a list and then give it to me, what is this list? That is what you have to understand, whatever is this list, this list is the same as in the R software what we are going to consider today.

What do you do? You simply try to write whatever you want to buy in that list, you will write number 1 milk, number 2 say vegetable, number 3 medicines, number 4 clothing etc., all these things are different. Clothing is a clothing, medicine is a medicine and

above of all they are not really going to be sold at the same shop, you have to go to different places.

So, now if you try to see what is a list; list is an object in which you are trying to club objects of various types. And after this once we have prepared the list then you want to do different types of operation which are helpful. For example, suppose you go to the market and then you want to understand that what your mom has written in the vegetable what you really want to have.

So, what you will say? You will try to pick up a particular number or a value of that variable and you will call your mom and ask ok you have written here vegetable what do you really want me to bring. So; that means, by the address of that object you are trying to get some more information on it. So, these are various types of information which we are going to handle here.

So, remember whenever you are trying to work on the list, we are going to have different types of objects and based on that we have to learn how we can call an element and how we can do different types of manipulations on the list. So, let us begin our lecture and try to understand first this basic concept ok. So, now we try to understand here list.

(Refer Slide Time: 00:32)

**Lists**

- Vectors, matrices, and arrays is that each of these types of objects may only contain one type of data.
- For example, a vector may contain all numeric data or all character data.
- A list is a special type of object that can contain data of multiple types.
- Lists are characterized by the fact that their elements do not need to be of the same object type.

So, list as you see here, list is combination of different object like as vector matrices array etc. right, and they contain a one type of data right, that is numerical or say

character etc. For example, a vector may contain all numeric data or all the character data. So, this list is a special type of object which contain data of multiple type and lists are characterized by the fact that their element do not need to be of the same object type.

For example, in the list if you try to see here you can write down here at place number 1 clothings, then medicine, then vegetable etc. So, similarly you have different types of data objects which is character, numeric, data frame, etc. right.

(Refer Slide Time: 05:24)

**Lists**

- ❖ Lists can contain elements of different types so that the list elements may have different modes.
- ❖ Lists can even contain other structured objects, such as lists and data frames which allows to create recursive data structures.
- ❖ Lists can be indexed by position.  
So `x[[5]]` refers to the fifth element of `x`.

So, now, when you are trying to obtain here a list, what you have to understand that list can have different elements and these elements may have different types of elements, which may have different modes like as one mode may be character, one mode may be numeric one mode may be something else.

So, this list is a platform that help us in collecting all such type of data on which we can do different type of observations number 1 and list can even contain some other type of structure objects such as lists data frame etc., which allow us to create the recursive data structure. So, this statement you will understand as soon as we begin our lecture and we try to understand the data frame that from the list also you can create here sub list, which is again going to be a list and which has the structure data.

Means you know where is what data type of thing and the list can be indexed by its position. For example, you are writing point number 1 vegetable, point number 2

clothing, point number 3 say this medicine and these elements can be accessed by its position. For example, if I have a list suppose I call it here as say x. So, and I want to call the fifth element of this list.

So, for example, I can write down here x and then I have to write here say double square brackets and then I have to write down here the location or the index or the position of that value which I want to call. So, what I am trying to tell you that ok, means different values in the list can also be called by their indexes.

(Refer Slide Time: 07:10)

**Lists**

- ❖ Lists can extract sublists.  
So `x[c(2,5)]` is a sublist of `x` that consists of the second and fifth elements.
- ❖ List elements can have names.  
Both `x[['Students']]` and `x$Students` refer to the element named "Students".  
*c(5, "apple")*
- ❖ Difference between a vector and a list :
  - In a vector, all elements must have the same mode.
  - In a list, the elements can have different modes.

And these indexes can be a single value or that can be a means a data vector also. So, when you are trying to give more than one indexes together, then it is going to extract a sub list right. So, for example, if I try to write down here suppose x is here a list and you are trying to extract the data at the second and fifth element.

So, it is going to be like here c and inside the parenthesis you write 2 comma 5, but this is also a going to be a list. So, this is the sub list, but its modes is again going to be a list and the elements of the list also can have names and you can call those elements by their name also.

For example, if I try to write down here x and inside the double brackets if I write within the double quote say Students, right, then it is going to refer to an element whose name is the Students and then even I write this way or I can write down here x dollar Students

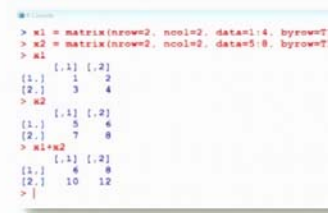
like this. So, they both of them are going to refer to the element whose name is Students. Well I am not taking here to show you that these are the possibilities when you are trying to do it.

Now, one basic questions comes here, then what is the difference between the vector and the list. Because in vector also you can write number like this c here 5 and then you can also write down here apple. So, this c is a numeric and apple is a character. So, what is the difference between the list and the data vector?

So, in a data vector all the elements must have the same mode, right, unless and until you give here all the numbers the mode is not going to be numeric. And so, that is the thing and in the case of a list the elements can have different mode for example, one value can be numeric another value can be character and so on.

(Refer Slide Time: 09:10)

```
Lists  
Example  
> x1 = matrix(nrow=2, ncol=2, data=1:4, byrow=T)  
> x2 = matrix(nrow=2, ncol=2, data=5:8, byrow=T)  
> x1  
      [,1] [,2]  
[1,]  1   2  
[2,]  3   4  
  
> x2  
      [,1] [,2]  
[1,]  5   6  
[2,]  7   8  
  
> x1+x2  
      [,1] [,2]  
[1,]  6   8  
[2,] 10  12
```

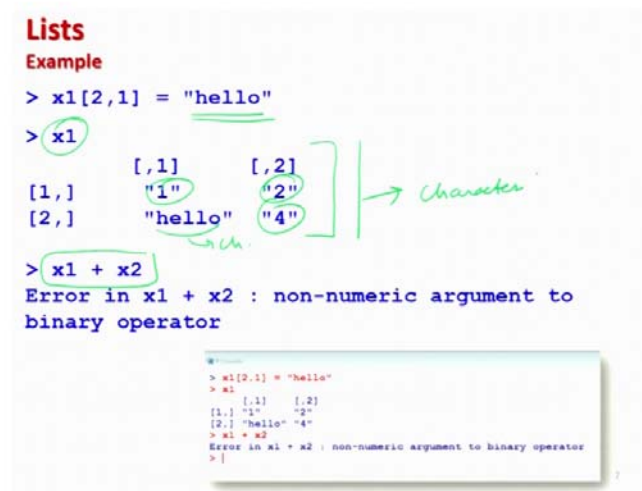


So, that is the main difference between the two. So, let me try to give you here some examples through which I can explain you the concept law of list in a better way, right. So, let me try to create here two matrices. Now, you know how to create a matrix. So, I try to create here two matrices x1 and x2. So, x1 is a matrix of order 2 by 2 in which the data is going to be like 1, 2, 3, 4 and it is arranged by rho is my another 2 by 2 matrix in which the data is 5, 6, 7, 8 and it is again going to be arranged by rho right. So, this is my here x1 and this is my here x2.

Now, in case if you try to add here x1 and x2 then their sum is going to be obtained here like this, you know how to get the matrix operation. But you can see here that when you are trying to add x1 and x2, that is going to give you a result and you can see here all the values here in x1 and x2 they are numbers, they are numeric.

(Refer Slide Time: 10:08)

```
Lists  
Example  
> x1[2,1] = "hello"  
> x1  
      [,1] [,2]  
[1,]  "1"  "2"  
[2,] "hello" "4"  
> x1 + x2  
Error in x1 + x2 : non-numeric argument to  
binary operator
```



```
## R console output  
> x1[2,1] = "hello"  
> x1  
      [,1] [,2]  
[1,]  "1"  "2"  
[2,] "hello" "4"  
> x1 + x2  
Error in x1 + x2 : non-numeric argument to binary operator  
> |
```

Now, I try to do one thing, I try to replace one element by some character. So, in the x1 matrix I try to replace the value in the second row and first column as a “hello”. So, now this x1 becomes here like this and you know how to address a particular element in a matrix, but now after this if you try to see here that “1” “2” and “4” they are the numbers, but “hello” is the character.

So, if you try to now obtain here x1 plus x2. So, you can see here that it is not going to give you an output because these two matrices cannot be added because their modes are different. One is numeric and the mode of this matrix which is now obtained after replacing the element by “hello” that will become a character, right. So, this is the difference.

(Refer Slide Time: 10:59)

### Lists

Lists can contain any kind of objects as well as objects of different types. For example, lists can contain matrices as objects:

**Example**

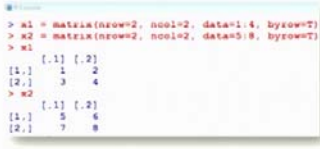
```
> x1 = matrix(nrow=2, ncol=2, data=1:4, byrow=T)
> x2 = matrix(nrow=2, ncol=2, data=5:8, byrow=T)
```

> x1

```
      [,1] [,2]
[1,]    1    2
[2,]    3    4
```

> x2

```
      [,1] [,2]
[1,]    5    6
[2,]    7    8
```



The screenshot shows the R console output for the example. It displays the commands to create matrices x1 and x2, and the resulting matrices. x1 is a 2x2 matrix with values 1, 2, 3, 4. x2 is a 2x2 matrix with values 5, 6, 7, 8.

So, now I try to just consider the same matrices x1 and x2 here, just like this what I just shown you and which are here this matrix.

(Refer Slide Time: 11:09)

### Lists

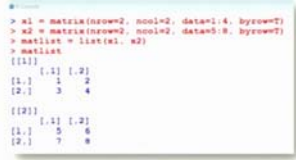
**Example**

```
> matlist = list(x1, x2)
```

> matlist

```
[[1]]
      [,1] [,2]
[1,]    1    2
[2,]    3    4

[[2]]
      [,1] [,2]
[1,]    5    6
[2,]    7    8
```



The screenshot shows the R console output for creating a list. The command `matlist = list(x1, x2)` is executed. The resulting list is displayed as a list of two matrices. The first element is a 2x2 matrix with values 1, 2, 3, 4. The second element is a 2x2 matrix with values 5, 6, 7, 8. Handwritten notes in green indicate that the list contains two elements, x1 and x2, and that the list is a collection of objects.

*list (Vegetable, Clothing, medicine)*  
[[1]] [[2]] [[3]]

And I try to show you that how you can create a list from this matrices and then how we can operate it. For example, the basic command to create a list is very simple, just try to write down here the list all in lowercase alphabets and after that whatever elements you want to join in the list or you whatever you elements you want to incorporate in the list just try to write them inside the parentheses separated by a comma.

So, just write suppose if I have these two matrices x1 and x2. So, I write down here list and inside the parenthesis x1 comma x and let us try to store this list in the name as matlist. So, this is a short form say matrix list so that you can recall what we are trying to do. So, now if you try to see the structure of this matlist what happens here, right.

So, this is here like this, you can see here your x1 matrix comes here and your x2 matrix comes here right, and this is here something like this you can see here this is written here in the double bracket, double square brackets 1 and double square bracket 2. So, this is how the this now list will look like, right.

So, this list do not you think it is like that your mom asked you that you bring some vegetables and then bring some clothings and bring some medicine and then you try to create here a list like this. So, this vegetable is at the location number 1. So, this will be here like this, clothing is at the location number 2 and medicines they are at location number 3. So, that is the same thing what we are trying to do here under this operation.

(Refer Slide Time: 12:54)

```
Lists
Example

> matlist[1] ✓
[[1]]
 [1,] [,1] [,2]
 [1,]  1  2
 [2,]  3  4 x1

> matlist[2]
[[1]]
 [1,] [,1] [,2]
 [1,]  5  6
 [2,]  7  8 x2

> matlist[1]
[[1]]
 [1,] [,1] [,2]
 [1,]  1  2
 [2,]  3  4

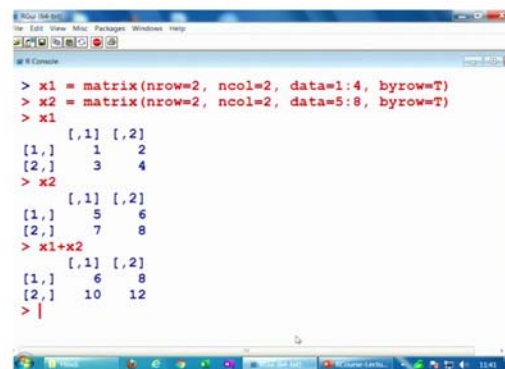
> matlist[2]
[[1]]
 [1,] [,1] [,2]
 [1,]  5  6
 [2,]  7  8
```

And now in case if you want to extract a particular element, as I told you that it is possible to extract a particular element in this list. So, how to get it done? We simply have to write down the name of the list what we have given like as here matlist and then you have to write down the square brackets and then you have to give the index, that which element you want to have first or second. So, if I try to write down here say matlist and inside the square bracket 1. So, you will get here the first value the matrix here x1.



And if you try to write down here matlist 2, then you are going to get here the second matrix here x2, right. But you can see here one thing that now here this 1 will remain as 1 in both the cases because that is now going to indicate you that what is the position of this element in the new sub list, right. So, that is what you have to keep in mind. So, let us try to first understand these operations on the R console here and then I will try to show you that how are you going to handle it, right. So, let me try to first copy here this matrix.

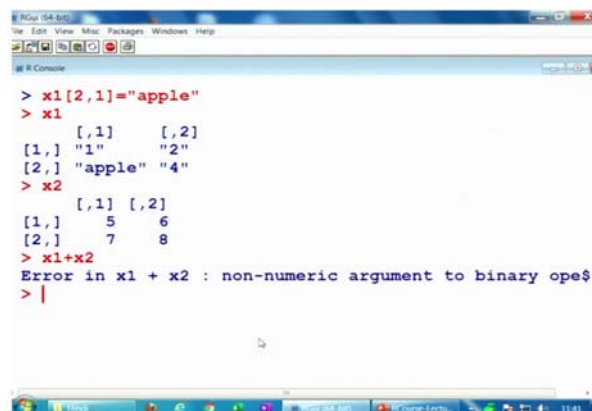
(Refer Slide Time: 14:13)



```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
> x1 = matrix(nrow=2, ncol=2, data=1:4, byrow=T)
> x2 = matrix(nrow=2, ncol=2, data=5:8, byrow=T)
> x1
      [,1] [,2]
[1,]    1    2
[2,]    3    4
> x2
      [,1] [,2]
[1,]    5    6
[2,]    7    8
> x1+x2
      [,1] [,2]
[1,]    6    8
[2,]   10   12
> |
```

See here x1. So, this is here my x1 matrices and then I try to create here my x2 matrix and you can see here this is my here x2. So, x1 is here like this and x2 here is like this. So, you can see here x1 plus x2 this is here like this right.

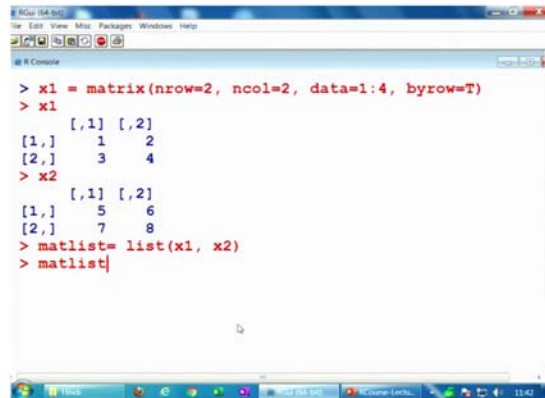
(Refer Slide Time: 14:37)



```
RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
> x1[2,1]="apple"
> x1
      [,1] [,2]
[1,]  "1"  "2"
[2,] "apple" "4"
> x2
      [,1] [,2]
[1,]    5    6
[2,]    7    8
> x1+x2
Error in x1 + x2 : non-numeric argument to binary operator
> |
```

And now in case if you try to suppose replace say x1 inside this x1 matrix if you try to replace the element at the second row and first column, say here as say apple, right. So, your x1 becomes here like this and x2 will remain here like this, but if you try to see here this x1 and x2, this will give you an error because these things cannot be added, right.

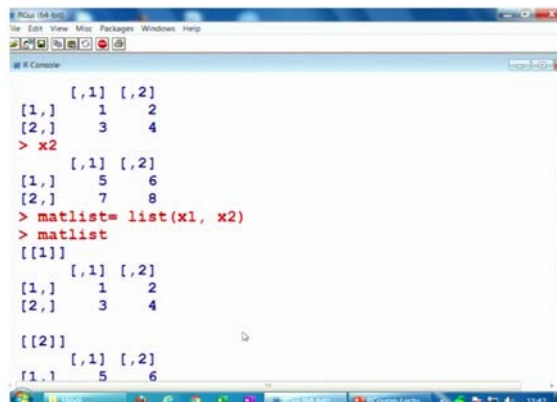
(Refer Slide Time: 15:05)



```
> x1 = matrix(nrow=2, ncol=2, data=1:4, byrow=T)
> x1
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> x2
     [,1] [,2]
[1,]    5    6
[2,]    7    8
> matlist= list(x1, x2)
> matlist
```

So, now let us come back to our I mean this original matrices. So, that I can show you what you really want to do. So, let me try to x1 create x1 as earlier and x2 is here like this. So, now, if I try to create here a list so matlist. So, if you try to see here, I am simply trying to write down here the name mat list and then I will type here list and inside the parenthesis I will simply try to give the names of these two elements x1 and x2 and it is created here, right.

(Refer Slide Time: 15:40)

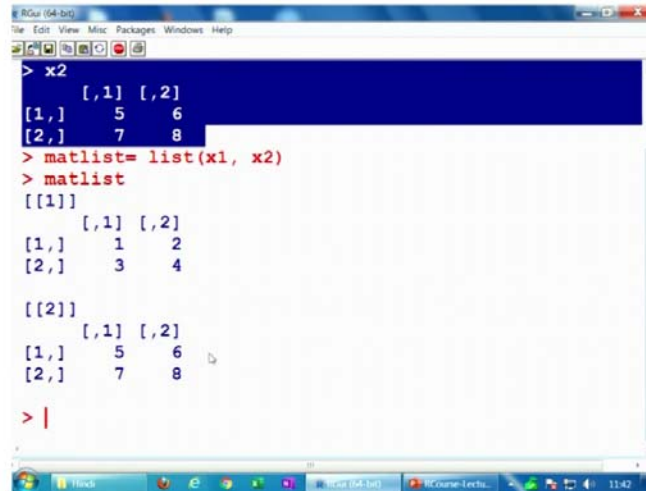


```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
> x2
     [,1] [,2]
[1,]    5    6
[2,]    7    8
> matlist= list(x1, x2)
> matlist
[[1]]
     [,1] [,2]
[1,]    1    2
[2,]    3    4

[[2]]
     [,1] [,2]
[1,]    5    6
```

So, in case if you try to see what is happening here at this mat list you can see here like this right. So, this is here the 1st matrix which is here in the x1 and 2nd one here x2 here is here like this.

(Refer Slide Time: 15:51)



```
> x2
  [,1] [,2]
[1,]  5   6
[2,]  7   8

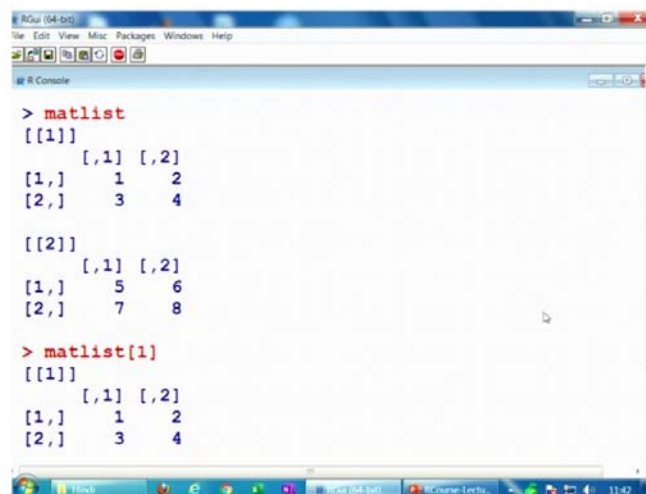
> matlist= list(x1, x2)
> matlist
[[1]]
  [,1] [,2]
[1,]  1   2
[2,]  3   4

[[2]]
  [,1] [,2]
[1,]  5   6
[2,]  7   8

> |
```

So, you can see here that this is your here x2 right.

(Refer Slide Time: 15:57)



```
> matlist
[[1]]
  [,1] [,2]
[1,]  1   2
[2,]  3   4

[[2]]
  [,1] [,2]
[1,]  5   6
[2,]  7   8

> matlist[1]
[[1]]
  [,1] [,2]
[1,]  1   2
[2,]  3   4
```

So, let me try to show you here more clearly it is here like this. So, now if you try to extract here the first element or the element at the index number 1. So, you have to

simply write down the index number inside the square brackets and you will get here like this one.

(Refer Slide Time: 16:15)

```

RStudio [64-bit]
File Edit View Misc Packages Windows Help
[1,] [,1] [,2]
[1,] 5 6
[2,] 7 8

> matlist[1]
[[1]]
[1,] [,1] [,2]
[1,] 1 2
[2,] 3 4

> matlist[2]
[[1]]
[1,] [,1] [,2]
[1,] 5 6
[2,] 7 8

> |

```

And similarly, if you want to have the value at the index number 2 in this list you have to simply write down here the name of the list and inside the square bracket you have to write down the index number. So, you can see here this is now your second matrix which has element 5, 6, 7, 8. And if you try to see here this element x1 is in the first position in the mat list this one and this mat list two, it is x2 is at the first position that is why this double square bracket one is appearing here right.

So, now, we come back to our slide and try to create here one more example. So, that you can understand that what is this matrix doing.

(Refer Slide Time: 16:56)

**Lists**

An example of a list that contains different object types:

```

> z1 = list( c("water", "juice", "lemonade"),
  rep(1:4, each=2), matrix(data=5:8, nrow=2,
  ncol=2, byrow=T) )

```

*list (ch. data rectly, numeric, matrix)*

```

> z1
[[1]]
[1] "water" "juice" "lemonade"

[[2]]
[1] 1 1 2 2 3 3 4 4

[[3]]
[1,] [,1] [,2]
[1,] 5 6
[2,] 7 8

```

So, now, let me try to give you a more realistic example in the sense that earlier I had considered only two elements and both were numeric. So, now I try to create here a list with 3 different types of values. So, I try to take here my first data vector here as a character, which has 3 values water, juice and lemonade. And then I try to consider here the second value which is here rep 1 colon 4 each is equal to 2. So, the sequence 1, 2, 3, 4 is going to be repeated two times at the option each is equal to 2, so that is also going to be a numeric.

So, the first value here is character second value here is numeric and third value here is now matrix. So, that is the same matrix which we have considered just now your x2 matrix which has data 5, 6, 7, 8, it is the 2 by 2, matrix with number of rows equal to 2 number of column equal to 2 and the data is arranged by row.

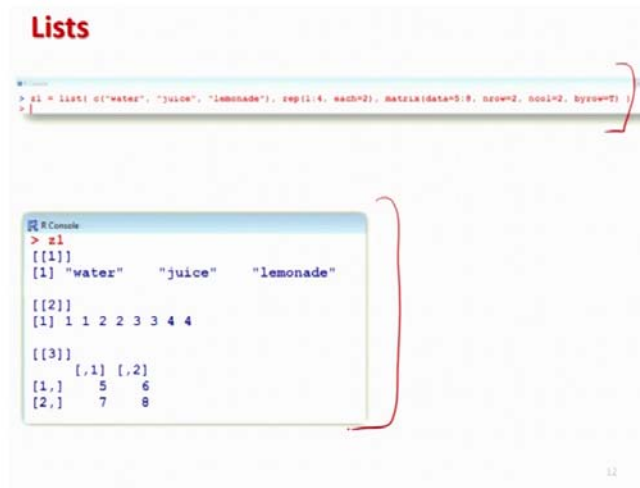
So, now you can see here now I have created here a list of character data vector and then here numeric vector and then here once again I am trying to take here matrix. So, now, I am trying to create here a list of these 3 things together. So, you can see here that the modes of these individual elements they are actually different right.

And now I am trying to combine them together with the command here list ok and I try to give it here are names say z1. So, now, if you try to see the structure of here z1 this will be here like this at the position number 1 you have here water, juice and lemonade and this was located in the at the first position in the list you can see here like this.

Now, the second element here is 1, 1, 2, 2, 3, 3, 4, 4 what is this is the second value or the value at the second position in the list, this is obtained from the command rep, right. And after this what is your here third value? Third value here is a matrix and this matrix was present at index number 3 in the list as say here matrix.

So, now you can see here you have created here a list which has got different types of objects which may have different modes. So, this is the role of the list and here you can see that the good part what you see here the matrix looks like only as a matrix, in the order of the values inside the matrix is not changed, right.

(Refer Slide Time: 19:28)



**Lists**

```
> z1 = list( c("water", "juice", "lemonade"), rep(1:4, each=2), matrix(data=5:8, nrow=2, ncol=2, byrow=T) )
> z1
```

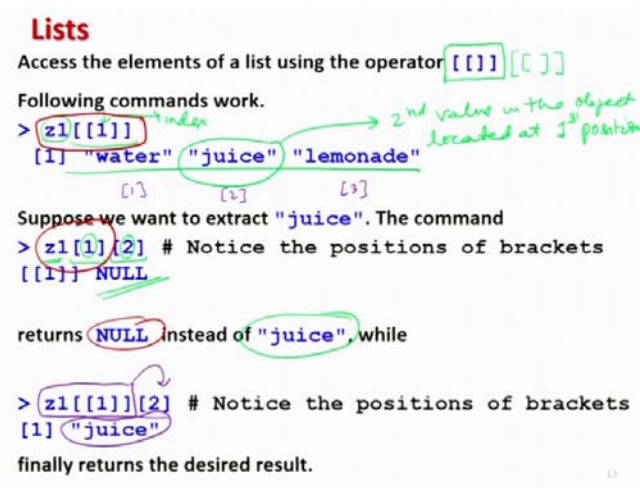
```
R Console
> z1
[[1]]
[1] "water" "juice" "lemonade"

[[2]]
[1] 1 1 2 2 3 3 4 4

[[3]]
[,1] [,2]
[1,] 5 6
[2,] 7 8
```

So, this is here the screenshot of the same operation, I will try to show you it on the R console also.

(Refer Slide Time: 19:35)



**Lists**

Access the elements of a list using the operator `[[ ]]` `[ [ ]]`

Following commands work.

```
> z1[[1]]
[1] "water" "juice" "lemonade"
```

Suppose we want to extract "juice". The command

```
> z1[1][2] # Notice the positions of brackets
[[1]] NULL
```

returns **NULL** instead of "juice", while

```
> z1[[1]][2] # Notice the positions of brackets
[1] "juice"
```

finally returns the desired result.

But before that I try to show you here very simple some operation that how are you going to execute them. Suppose I want to access a particular element in the list then how to do it? So, for that I have to use here an operator which is like double square brackets, you have to write bracket square bracket and then write another square bracket. For example, if you want to access the contents of the elements at the first place you have to

write down here the list name which is here z1 and then inside the double square brackets you have to write down here the index of the or the location.

That means, you want to access the element in the list which is at the position number 1 or the index number 1. So, you can see here what is your here in the position number 1? This is here water, juice and lemonade. So, this is coming here as a water, juice and lemonade and now I have another job. I want to access here this second element of the first object in the list, which is here juice. So, in order to obtain the juice value here you have to see; what is this?

This is the second value in the object located at first position. So, now, I will try to take here two option and I will try to show you that you should not make this type of mistake. So, suppose I write here say name of the list z1 and after this in single brackets I write down here 1 and 2 like this and I believe that ok this is the index number 1 of the list and this is here 2 which is the index number of the element in the first position of the list.

So, now it will give you here a value here null and we wanted here juice. So, this is not the correct command, what you have to inform here that, you want here the 2nd element of what? 2nd element of the object which is located at the first position. Now, you have to understand how you are going to address the object which is located at the index number 1 in the list z1, it is here like this z1 and then you have to give two square brackets and then you have to write one inside it. What mistake you have done here? You have written only here a single square bracket.

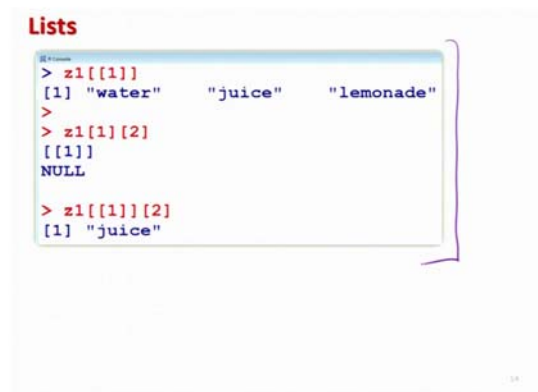
So, that is trying to give you a value here NULL and if you recall that earlier in the lecture we had understood the meaning of say NULL, when we consider the NA and NULL. So, NULL is the value which never existed, that goes back to the same example which I took that there is a student who has not got admission in the school. But when we are trying to take the attendance of the student, then what will happen this is NULL because that student was not admitted in the school.

So, that is why and if the student is absent today his position is going to be NULL. But on the other hand in case if a student is admitted in the school and the and if the student is absent today, then we are going to mark absent that will be your NA. So, this that is why you are getting here NULL.

So, now the correct option here will be, if you want to access the second element of the first object of the list, then you have to write the correct address here first, see z1 and inside the double square brackets you write 1 and after that you write here inside the square bracket 2. So, once you write this thing so this position is like here 1 as water, 2 here as juice and 3 here as lemonade.

So, now the control will come to the second position and it will give you the answer here juice. So, that is how you have to understand that the things are working in this R software.

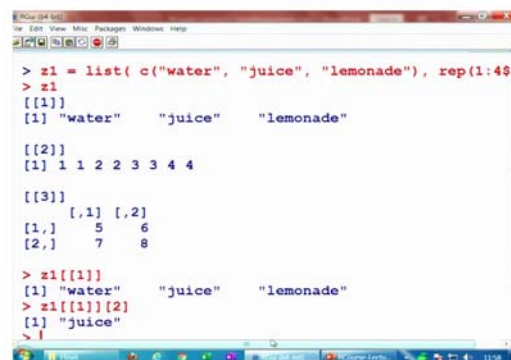
(Refer Slide Time: 23:39)



```
Lists
> z1[[1]]
[1] "water" "juice" "lemonade"
> z1[[1]][2]
[[1]]
NULL
> z1[[1]][2]
[1] "juice"
```

And you can see here this is here the screenshot of the same operation which I shown you here. But let me try to first show you these operations on the R console. So, let me try to first create this here list.

(Refer Slide Time: 24:00)



```
RStudio [64-bit]
File Edit View Misc Packages Windows Help
> z1 = list( c("water", "juice", "lemonade"), rep(1:4)
> z1
[[1]]
[1] "water" "juice" "lemonade"

[[2]]
[1] 1 1 2 2 3 3 4 4

[[3]]
      [,1] [,2]
[1,]    5    6
[2,]    7    8

> z1[[1]]
[1] "water" "juice" "lemonade"
> z1[[1]][2]
[1] "juice"
> !
```



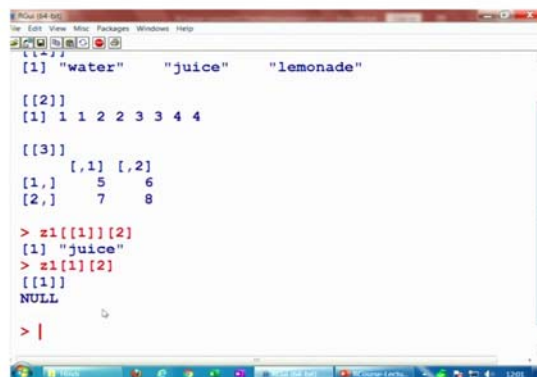
So, I try to create here the and then I come to R console and I try to copy and paste my command. So, this will give you here like this. So, this is your here actual list you can see here, right. Now, in case if you want to access means any particular element here, I can explain you here a very simple approach, try to see what is this here. This is the address of the first element of the list, then this is the address of the second element of the list and this is here the third value or the address of the third value of the list.

So, you can see here this is given here as say inside the double brackets, after this you have to suppose you want to extract here the juice. So, you have to write the correct address. So, there can be a confusion whether you have to write single bracket or double bracket as I showed you in the example, if you try to see after that it is written here only the single bracket.

So, you have to write simply z1, then double brackets whatever index you want and then try to write down here single brackets and then try to give the location 1, 2 or 3 in matrix you have to see that how these locations are going to work. So, now let me try to show you here this operation which I shown you on the say here this one, this is the same where I want to access a particular address.

So, you can see here I can write down here see here 1. So, this will give me here like as water, juice and if I try to write down here say here suppose here z double square bracket 1 and then single bracket 2, this will give you here the juice, right, you can see here.

(Refer Slide Time: 25:45)



```
RStudio [64-bit]
File Edit View Misc Packages Windows Help

[[1]]
[1] "water" "juice" "lemonade"

[[2]]
[1] 1 1 2 2 3 3 4 4

[[3]]
      [,1] [,2]
[1,]    5    6
[2,]    7    8

> z1[[1]][2]
[1] "juice"
> z1[1][2]
[[1]]
NULL
> |
```

Now, in case if you try to simply make a single bracket. So, that is the mistake what I am going to do here, because I want to x yeah here like here this then you can see here this is going to give you the answer here NULL, you can see here because this value does not exist here. So, now we come to an end to this lecture and you can see here that in this lecture I simply have introduced the concept of list and I have explained you how you can access a particular element in the list. And after this there are many more operations which are possible on the list I will try to take them in the next lecture.

So, but more important part today is that you please try to understand this concept and try to consider different types of objects which are numeric, character etc. and then you try to create a list. Yes, I agree you cannot use here at this moment data frame and factors. But whatever you have learnt try to consider them and try to see what happens and one exercise for you, I have not shown you here on the screen, that in this case the third element was your matrix.

Try to see how you can access different elements of the matrix in this list you know that how you can access the particular element of a matrix, but in this case if you try to write down here z1, with double brackets here 3 and after that you have to write 1, 2, 3, 4 what do you get. Try to observe it. Because unless and until you experiment with see these things you will not understand how R is going to work, how R is going to think.

So, you try to practice it and I will see you in the next lecture with more commands on the list. Till then goodbye.