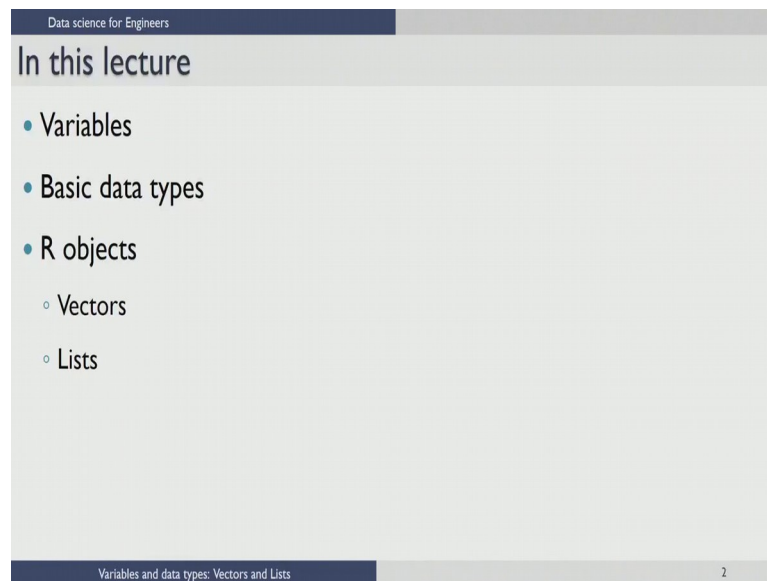


Data Science for Engineers
Prof. Raghunathan Rengaswamy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 04
Variables and datatypes in R

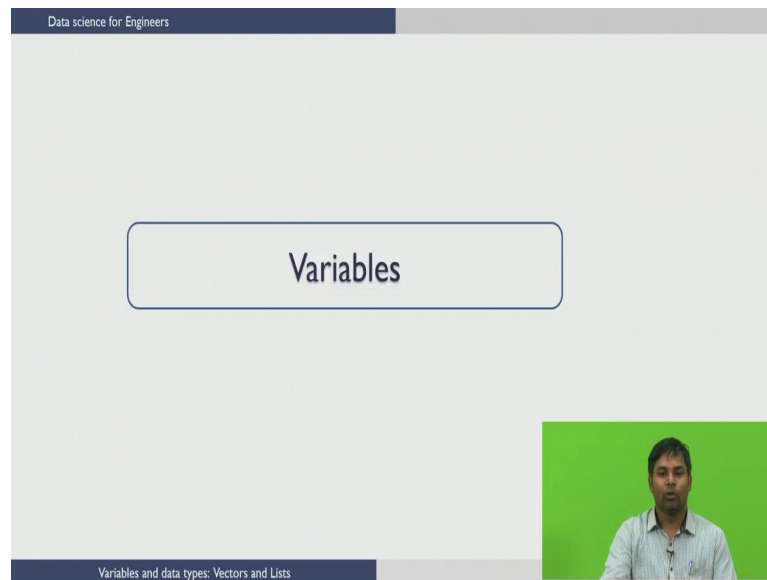
Welcome, to the lecture – 3 of R module in the course Data Science for Engineers.

(Refer Slide Time: 00:21)



In this lecture we are going to see the rules for naming the variables in R and what are the basic data types that are available in R and we are also going to see two basic R objects; vectors and lists, in detail.

(Refer Slide Time: 00:36)



Let us see the rules for naming the variables in R first.

(Refer Slide Time: 00:41)

A presentation slide titled "Variables" from a course "Data science for Engineers". The slide is divided into two main sections: "Variable names - Rules" and "Examples". The "Rules" section lists three bullet points: "Allowed characters are Alphanumeric, \\' and \'.", "Always start with alphabets", and "No special characters like !, @, #, \$,". The "Examples" section shows "Correct naming" with two R commands: "> b2 = 7" and "> Manoj_GDPL = \"Scientist\"", and "Wrong naming" with the command "> 2b = 7" followed by an error message: "Error: unexpected input in \"2b \"". In the bottom right corner, there is a small video inset showing the same man as in the previous slide. The footer of the slide reads "Variables and data types: Vectors and Lists".

The variable name in R has to be alphanumeric characters with an exception of underscore and period, the special characters which can be used in the variable names. The variable name has to be started always with an alphabet and no other special characters except the underscore and period are allowed in the variable names. This shows some examples of the correct variable names that can be used in R. The first one,

$b2 = 7$, assigns the value of 7 to the variable `b2`. This is a valid variable name because it started with an alphabet and it has only alphanumeric characters.

Similarly, the second variable `Manoj_GDPL = scientist` this is also valid variable name because it has a special character, but it is underscore which is allowed special character for the variable names. Now, let us see some examples where the variable names are not correct the variable `2b = 7`, gives an error because that variable name has started with the numeric character which is not following the rules for the names of the variables in R.

(Refer Slide Time: 02:00)

Data science for Engineers

Predefined constants

Constant	Symbol in R
<i>Pi</i>	<code>pi</code>
<i>letters</i>	<code>a,b,c,.....x,y,z</code>
<i>LETTERS</i>	<code>A,B,.....X,Y,Z</code>
<i>Months in a year</i>	<code>month.name,</code> <code>month.abb</code>

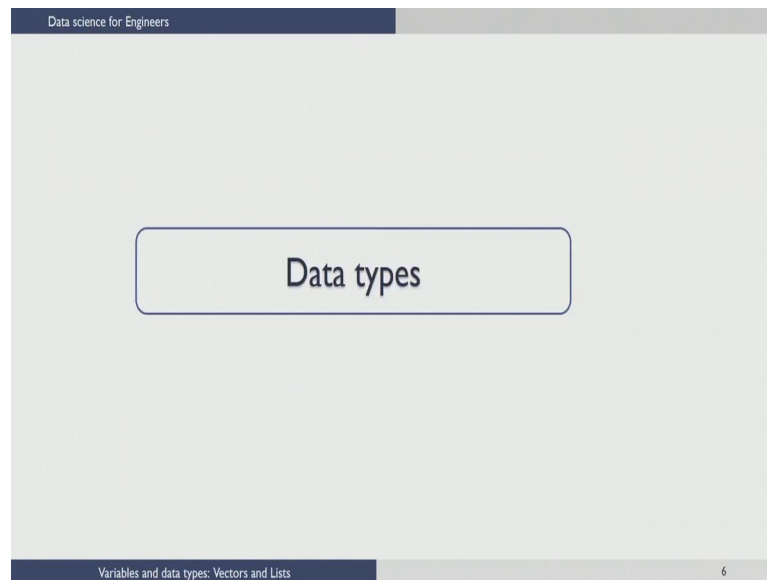
```
> pi
[1] 3.141593
> letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i"
[10] "j" "k" "l" "m" "n" "o" "p" "q" "r"
[19] "s" "t" "u" "v" "w" "x" "y" "z"
> LETTERS
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I"
[10] "J" "K" "L" "M" "N" "O" "P" "Q" "R"
[19] "S" "T" "U" "V" "W" "X" "Y" "Z"
> month.name
[1] "January" "February" "March"
[4] "April" "May" "June"
[7] "July" "August" "September"
[10] "October" "November" "December"
> month.abb
[1] "Jan" "Feb" "Mar" "Apr" "May" "Jun"
[7] "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
>
```

Variables and data types: Vectors and Lists

5

R also contains some predefined constants that are available such as `pi`, `letters`, the lowercase `a` to `z` and `LETTERS` in the uppercase which are uppercase letters from `A` to `Z` and months in a year, you can have full month name by `month.name` and you can have abbreviated month names by typing `month.abb`.

(Refer Slide Time: 02:25)



Let us now look at the data types that are available in the R.

(Refer Slide Time: 02:31)

The slide is titled 'Basic data types' in a large, bold font. Below the title is a table with two columns: 'Basic data types' and 'Values'. The table lists five data types: Logical, Integer, Numeric, Complex, and Character, each with its corresponding values. In the bottom right corner, there is a small video inset showing a man with a beard and short dark hair, wearing a light blue shirt, speaking against a green background.

Basic data types	Values
<i>Logical</i>	TRUE and FALSE
<i>Integer</i>	Set of all integers, \mathbb{Z}
<i>Numeric</i>	Set of all real numbers
<i>Complex</i>	Set of complex numbers
<i>Character</i>	"a", "b", "c", ..., "x", "y", "z", "@", "#", "\$", "%", "&", "1", "2", ... etc..

R has the following basic data types and this table shows the data type and the values that each data type can take. So, R has logical data types which take either a value of true or false, it supports integer data types which is the set of all integers and numeric which is set of all real numbers. We can also define complex variables, R supports set of all the complex numbers. Also, we can have a character data type where you have all the

alphabets and special characters which are under the window of basic data types of characters. There are several task that can be done using data types.

(Refer Slide Time: 03:14)

Data science for Engineers

Basic data types

TASK	ACTION	SYNTAX/EXAMPLE
Find data type of object	use command "typeof()"	Syntax: typeof(object)
Verify if object is of a certain datatype	use prefix "is." before datatype as command.	Syntax: is.data_type(object) Example : is.integer()
Coerce or convert data type of object to another	use prefix "as." before datatype as command.	Syntax: as.data_type(object) Example :as.logical()

Note : Not all coercions are possible and if attempted will return "NA" as output

Sample Codes

```

Console - R
> typeof(1)
[1] "double"
> typeof("22-01-2001")
[1] "character"

```

```

Console - R
> is.character("21-11-2001")
[1] TRUE
> is.character(as.Date("21-11-2001"))
[1] FALSE


```

```

Console - R
> as.complex(2)
[1] 2+0i
> as.numeric("a")
[1] NA

```

Variables and data types: Vectors and Lists



The following table gives you the task action and the syntax for doing the task. For example, the first task is to find the data type of the object. To find the data type object you have to use type of function. The syntax for doing that is you need to pass the object as an argument to the function type of to find the data type of an object

The second task is, to verify if object is of certain data type. To do that you need to prefix is dot before the data type as a command. The syntax for that is, is dot, data type of the object you have to verify. For example, if you have variable a, which is defined as an integer and if you use this command is dot integer of a, it will show true originally.

The variable a is not defined as integer this will show false and the third task is interesting task where you can change the or convert the data type of one object to another to perform this action you have to use, as dot, before the data type as the command; the syntax for doing that is as dot data type of the object which you want to coerce. Note that all the coercions are not possible and it attempted will be returning a null value.

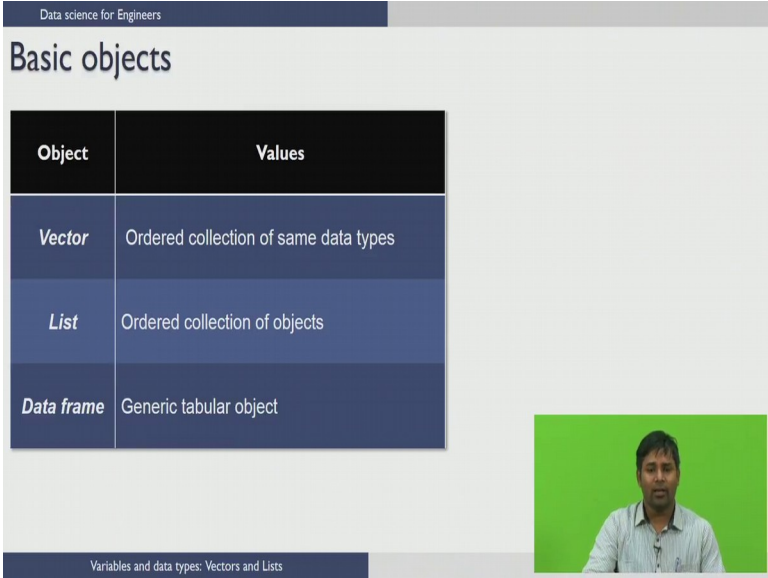
There are sample codes for doing this as which are in the bottom. The first one is type of 1, so, 1 is a numeric variable. So, if you say, type of, you will get double which is

numerical variable and if you say, type of a string, that is printed 22-1-2001, it will give value as character. Now, if you going to ask whether this; the character variable which have created 21-11-2001 is this character type variable, you can use this command is dot character the result is true because you have defined it as a character variable.

The next example, here what you are going to do here is we are coercing the character variable which is defined earlier that is 22-11-2001 as date and then you are checking whether that date is a character variable. The result is false because when you coerce this character variable as a date it will be a numeric variable, when you want to ask whether this variable is a character the result will be false.

You can also coerce numeric variable into complex variable by using as dot complex of, so, for example, we have as dot complex of 2, will convert this numeric variable 2 into the complex variable $2 + 0i$. Now, let us try coercing a character into a numeric variable using this command as dot numeric of which has given us not available or NA. This means the coercion from the characters to numeric numerical variables is not possible.

(Refer Slide Time: 06:48)



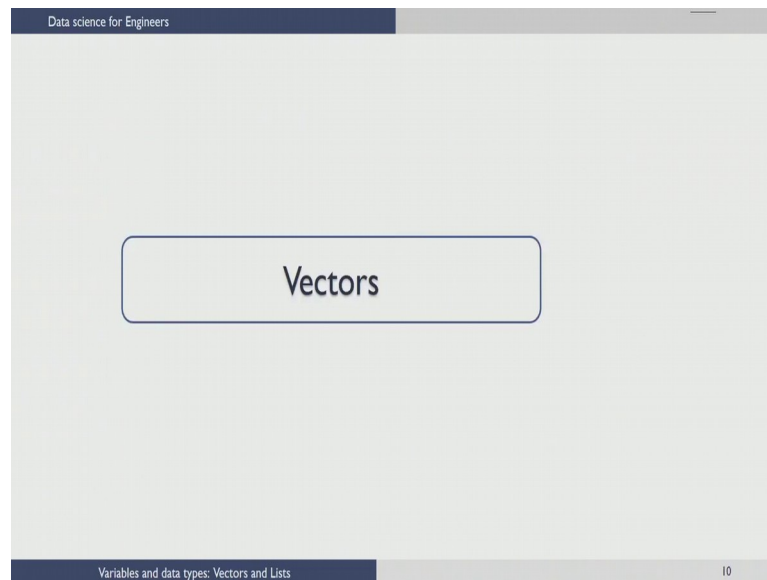
The slide is titled "Basic objects" and is part of a presentation on "Data science for Engineers". It contains a table with two columns: "Object" and "Values". The table lists three basic R objects: Vector, List, and Data frame, along with their descriptions. A video inset in the bottom right corner shows a man speaking.

Object	Values
Vector	Ordered collection of same data types
List	Ordered collection of objects
Data frame	Generic tabular object

Variables and data types: Vectors and Lists

We have several basic objects of R, in this the most important ones are; vectors, lists and data frames. A vector is an ordered collection of same data types, list is ordered collection of object themselves and data frame is a generic tabular object which is very important and the most widely used objects of R programming language. We will see in detail about each of this in the coming parts of the lecture and the other lectures also.

(Refer Slide Time: 07:22)



Let us now first see, what is a vector?

(Refer Slide Time: 07:25)

The slide is titled "Data science for Engineers" at the top and "Vectors" as the main heading. It contains two bullet points in a list box: "Vector : an ordered collection of basic data types of given length" and "All the elements of a vector must be of same data type". Below this, there are two panels. The "Code" panel shows the following R code:

```
# Vectors Example  
X = c(2.3,4.5,6.7,8.9)  
  
print(X)
```

 The "Console Output" panel shows the execution results:

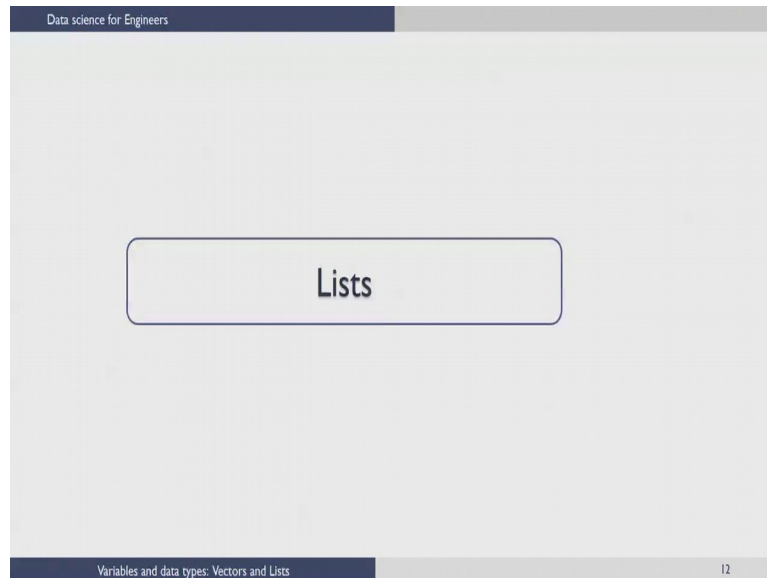
```
> # Vectors Example  
> X = c(2.3,4.5,6.7,8.9)  
> print(X)  
[1] 2.3 4.5 6.7 8.9  
>  
>
```

 The footer bar at the bottom contains "Variables and data types: Vectors and Lists" on the left and "11" on the right.

Vector is an ordered collection of basic data types of a given length. So, only key thing here is all the elements of a vector must be of a same data type. If you want to see an example the way you creating vector in R is using the concatenation command that is C. So, now I am going to define a vector which is containing four numeric variables and I am assigning it to a variable X. This is what the code here X = concatenation of these numbers and then I am printing X. So, if you execute this piece of code, this is how the

output in the console looks. It creates a vector X with the variables 2.3, 4.5, 6.7, 8.9 and prints them in the console.

(Refer Slide Time: 08:21)



Next, we move onto list.

(Refer Slide Time: 08:23)

The slide is titled "Lists in R: create a list". It contains a list of bullet points, a code block, and a console output block.

- List : a generic object consisting of an ordered collection of objects
- A list could consist of a numeric vector, a logical value, a matrix, a complex vector, a character array, a function, and so on

Code

```
# List Example : Employee details
ID = c(1,2,3,4)
emp.name = c("Man", "Rag", "Sha", "Din")
num.emp = 4
emp.list = list(ID, emp.name, num.emp)
print(emp.list)
```

Console Output

```
> ID = c(1,2,3,4)
> emp.name = c("Man", "Rag", "Sha", "Din")
> num.emp = 4
> emp.list = list(ID, emp.name, num.e
emp)
> print(emp.list)
[[1]]
[1] 1 2 3 4

[[2]]
[1] "Man" "Rag" "Sha" "Din"

[[3]]
[1] 4
```

The slide has a dark blue header bar with "Data science for Engineers" and a dark blue footer bar with "Variables and data types: Vectors and Lists" and the number "13".

List is a generic object consisting of ordered collection of objects. List can be a list of vectors, list of matrices, list of characters and list of functions and so on. To illustrate how a list looks, we take an example here. We want to build a list of employees with the

details for this I want the attributes such as ID, employee name and number of employees. So, I am creating each vector for those attributes.

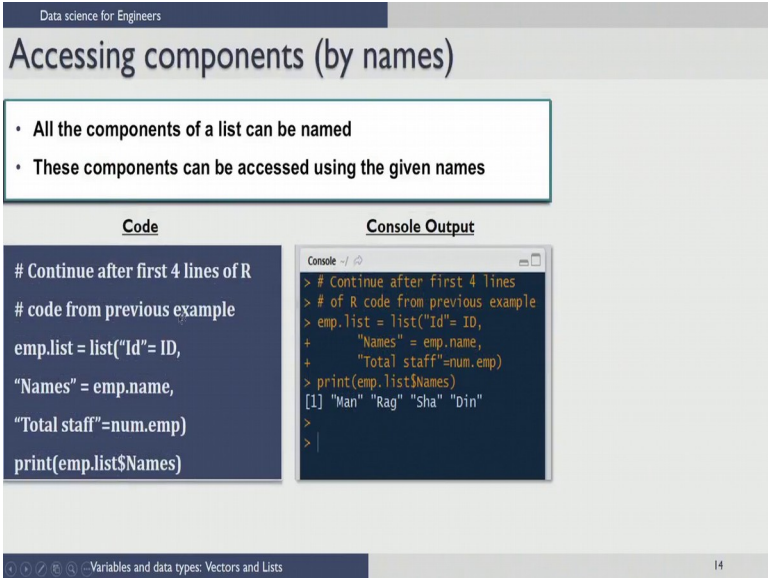
The first attributes is a numeric variable containing the employee IDs which is created using the command here, which is a numeric vector and the second attribute is employee name which is created using this line of code here, which is the character vector and the third attribute is number of employees which is a single numeric variable.

Now, I can combine all these three different data types into a list containing the details of employees which can be done using a list command. So, this command here creates a list variable which is a list of the ID, emp dot name and num dot employees that are defined above.

Once you create a list you can print the list and see how the output looks. So, when you execute this course, you can see in the console the list is printed this is the first one IDs 1, 2, 3, 4; this is the second element of the list which are contain the names of employees and the third element of the list which are saying how many number of employees are available. So, we have created a list.

Now, we can see how to access the components of the list.

(Refer Slide Time: 10:12)



The slide is titled "Accessing components (by names)" and is part of a presentation on "Data science for Engineers". It contains two main sections: "Code" and "Console Output".

Code:

```
# Continue after first 4 lines of R
# code from previous example
emp.list = list("Id"= ID,
               "Names" = emp.name,
               "Total staff"=num.emp)
print(emp.list$Names)
```

Console Output:

```
> # Continue after first 4 lines
> # of R code from previous example
> emp.list = list("Id"= ID,
+               "Names" = emp.name,
+               "Total staff"=num.emp)
> print(emp.list$Names)
[1] "Man" "Rag" "Sha" "Din"
>
> |
```

At the bottom of the slide, there is a navigation bar with icons and the text "Variables and data types: Vectors and Lists". The slide number "14" is visible in the bottom right corner.

All the components of a list can be named and you can use that names to access the components of the list. For example, this is the same list we have created you can use the

same ID, emp dot name and emp dot employee. Instead of directly creating a list you can also give the names for this attributes as ID, names of employees and the total staff as shown in the code here. Once you execute this code we can see now that list is created and if you want to access this element of the list you can do that by using the dollar command m dot list is the list and you want access the component with the name, names.

So, when you use this command and print the result you can see the names of the employees that are printed.

(Refer Slide Time: 11:14)

Data science for Engineers

Accessing components (indices)

To access top level components, use double slicing operator "[[]]" or "[]" and for lower/inner level components use "[]" along with "[[]]"

Code	Console Output
<pre># continuing from previous # code print(emp.list[[1]]) print(emp.list[[2]]) print(emp.list[[1]][1]) print(emp.list[[2]][1])</pre>	<pre>> print(emp.list[1]) \$id [1] 1 2 3 4 > print(emp.list[2]) \$names [1] "Man" "Rag" "Sha" "Din" > print(emp.list[[1]][1]) [1] 1 > print(emp.list[[2]][1]) [1] "Man"</pre>

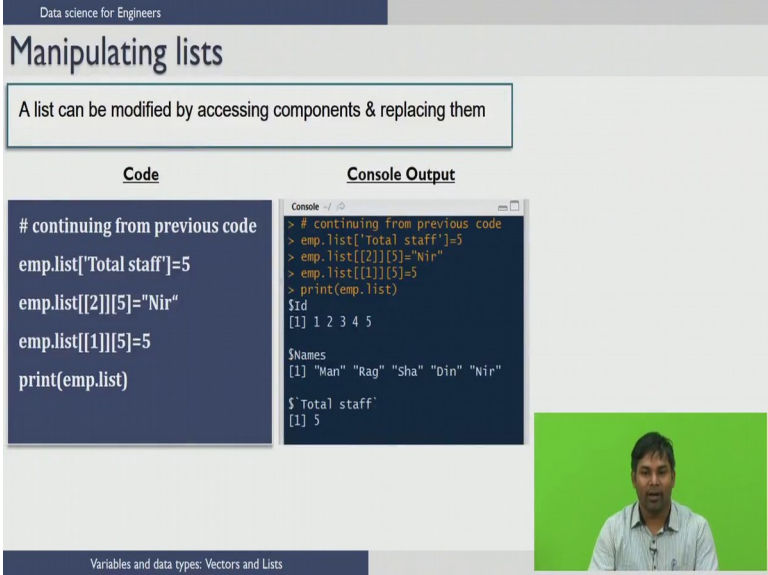
Variables and data types: Vectors and Lists 15

You can also access the components of the list using indices. To access the top level components of a list you have to use double slicing operator which is two square brackets and if you want access the lower or inner level components of a list you have to use another square bracket along with the double slicing operator. The course here illustrates how to access the top level components; for example, I want access the IDs, I can use print emp dot list and this is a double slicing operator which will give me the first level which is ID.

The second component can also be similarly accessed that is the result is shown here and if you want access, for example, the first sub element or the inner component of the component ID you have to use emp dot list the double slicing operator and the first element in the another square bracket.

Similarly, you can access the first employee name using double slicing operator to be followed by the element one which prints the value man from the employee list.

(Refer Slide Time: 12:35)



The slide is titled "Manipulating lists" and includes a subtitle "A list can be modified by accessing components & replacing them". It features a code editor on the left and a console output window on the right. The code editor shows the following R code:

```
# continuing from previous code
emp.list["Total staff"]=5
emp.list[[2]][5]="Nir"
emp.list[[1]][5]=5
print(emp.list)
```

The console output shows the following results:

```
> # continuing from previous code
> emp.list["Total staff"]=5
> emp.list[[2]][5]="Nir"
> emp.list[[1]][5]=5
> print(emp.list)
$Id
[1] 1 2 3 4 5

$Names
[1] "Man" "Rag" "Sha" "Din" "Nir"

$`Total staff`
[1] 5
```

A small video inset in the bottom right corner shows a man speaking.

A list can also be modified by access in the components and replacing them with the ones which you want. For example, I want to change the total number of staff into 5, that can be done easily by assigning a value 5 to the total staff and I want to add a new employee name to the list the component of the list which has the employee names is 2 and I want to add this new name Nir as a new employee to that sub component.

So, I can directly assign this character variable Nir to the second component and fifth sub component of the list. Now, we need to also increase the employee ID and you have to give this employee and new ID which is 5, what we are doing now, in this command is your accessing the fifth sub element of the level one component and then assigning data value of 5.

Now, once you print the list you can see that the IDs, number of employees are 5 and total staff is 5 and the name Nirav is getting added to the list.

(Refer Slide Time: 13:50)

The slide is titled "Concatenation of lists" and is part of a presentation on "Data science for Engineers". It explains that two lists can be concatenated using the `c(list1, list2)` function. The slide is divided into two main sections: "Code" and "Console Output".

Code:

```
# continuing from previous code
emp.ages = list("ages" =
c(23,48,54,30,32))
emp.list= c(emp.list , emp.ages)
print(emp.list)
```

Console Output:

```
> emp.ages = list("ages" = c(23,48,54,30,32))
> emp.list= c(emp.list , emp.ages)
> print(emp.list)
$Id
[1] 1 2 3 4 5

$Names
[1] "Man" "Rag" "Sha" "Din" "Nir"

$`Total staff`
[1] 5

$ages
[1] 23 48 54 30 32
```

The slide footer indicates the topic is "Variables and data types: Vectors and Lists" and the slide number is 17.

Next, we will see how to concatenate the list. Two lists can be concatenated using the concatenation function. The syntax for that is concatenation of list 1 and list 2. We have a list which already contains three attributes, you want add another attribute which is employee dot ages; for that I am creating a new list which contains the ages of the employees five employees.

Now, I want to concatenate this new list that is m dot ages with the original list which is emp dot list. So, when you want to concatenate these two lists you have to use this concatenation operator, the original list and then the new list. So, this command concatenates these two lists, you are now assigning it to the employee dot list. When you print this new employee dot list you will see that you have added another attribute ages to the original list.

To summarize, we have seen the data types that are supported an R and two objects of R vectors and list in detail. In the next lecture, we are going to look at the important data object of R which is data frames.

Thank you.