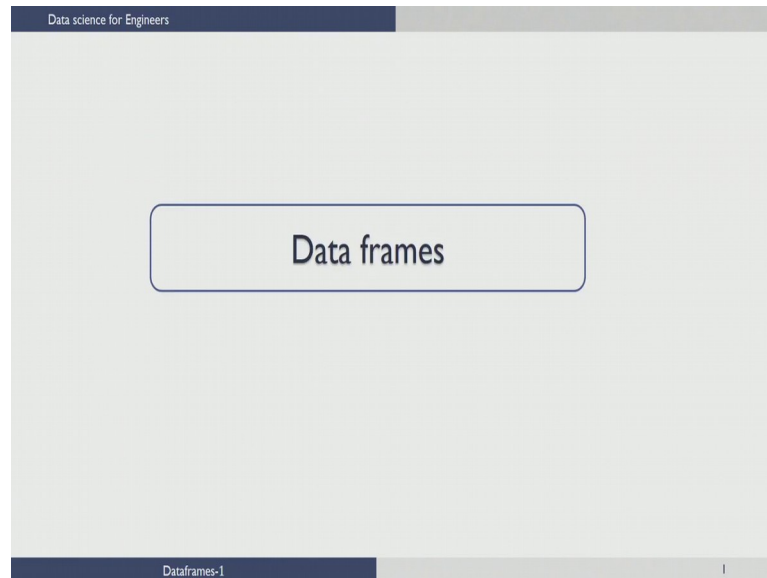


Data Science for Engineers
Prof. Raghunathan Rengaswamy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

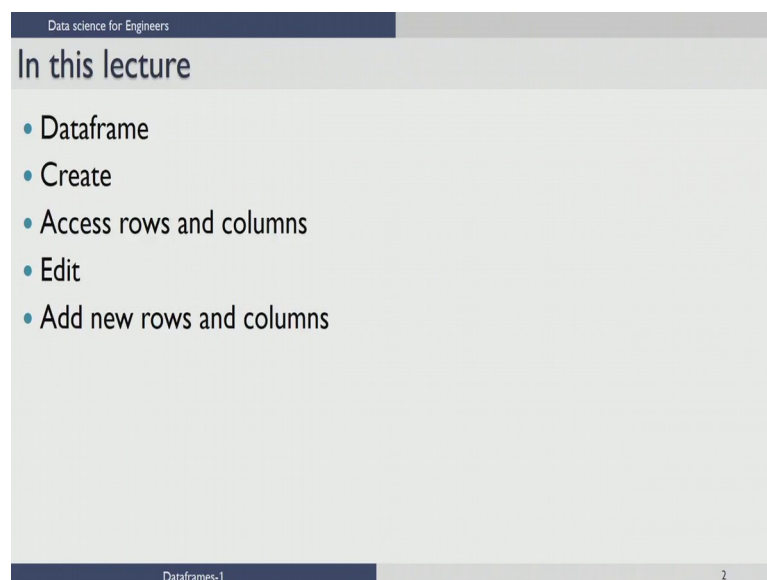
Lecture – 05
Data frames

(Refer Slide Time: 00:12)



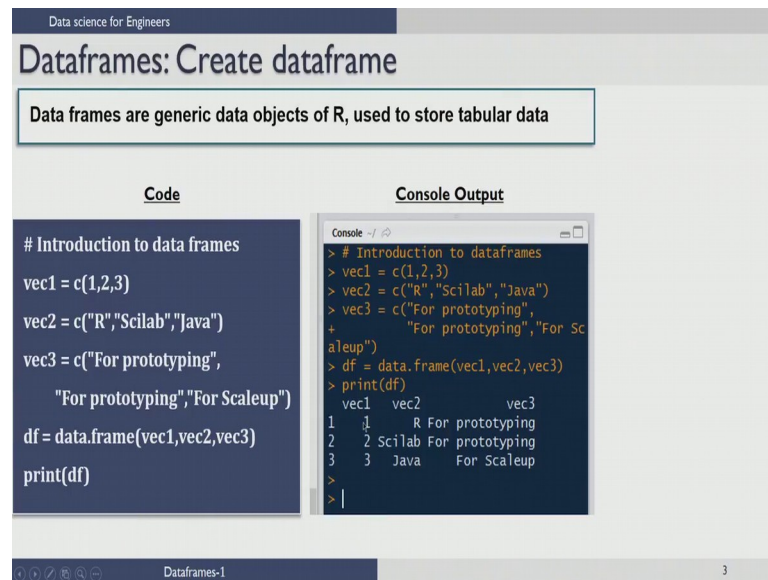
Welcome to the lecture 4 in the module r of the course data science for engineers. In this lecture we are going to introduce you to the data frame objects of R.

(Refer Slide Time: 00:23)



How to create data frames, how to access rows and columns of data frame, how to edit data frames and how to add new rows and columns of the data frame. Let us first look at what data frame is?

(Refer Slide Time: 00:40)



```
# Introduction to data frames
vec1 = c(1,2,3)
vec2 = c("R","Scilab","Java")
vec3 = c("For prototyping",
        "For prototyping", "For Sc
aleup")
df = data.frame(vec1,vec2,vec3)
print(df)
```

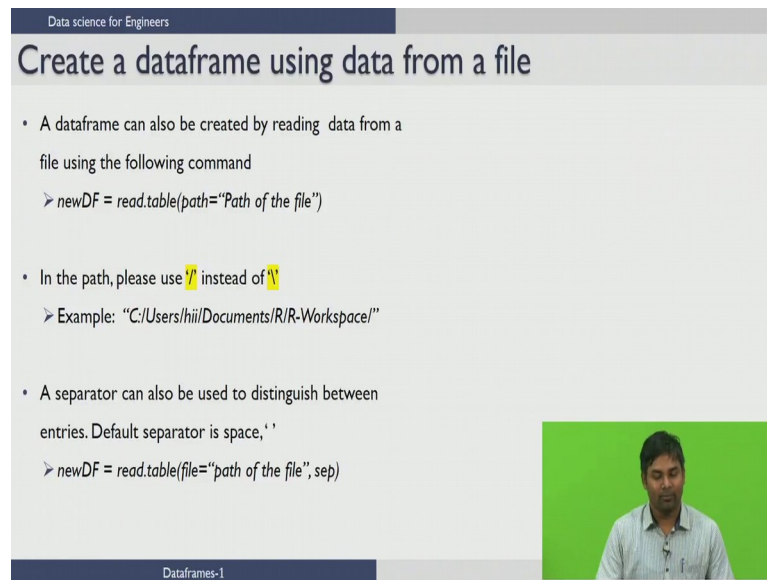
```
> # Introduction to dataframes
> vec1 = c(1,2,3)
> vec2 = c("R","Scilab","Java")
> vec3 = c("For prototyping",
+         "For prototyping", "For Sc
+         aleup")
> df = data.frame(vec1,vec2,vec3)
> print(df)
  vec1 vec2      vec3
1    1   R For prototyping
2    2 Scilab For prototyping
3    3   Java   For Scaleup
```

Data frame are generic data objects of R which you are used to store the tabular data. Data frames are the most popular data objects in R programming because we are comfortable in seeing the data in the tabular form. Data frames can also be taught as mattresses where each column of a matrix can be of different data type. Let us see how to create a data frame in R.

The code here shows how to create a data frame; the command here where the mouse is pointed creates a vector which is a numeric vector, which is containing 1, 2 and 3. The second command creates a character vector which contains 3 strings are Scilab and java; and the third command here is creating another character vector which is having entries for prototyping and first scale up.

The way you create the data frame is use the data dot frame command and then pass each of the elements you have created as arguments to the function data dot frame. This command will create a data frame d f when you print the data frame df, this is how the output looks. So, we can see that the names of the variables you have created are taken as columns and the entries in the each column are of the same data type; this is the condition which you need to be satisfying while creating a data frame.

(Refer Slide Time: 02:27)



The slide is titled "Create a dataframe using data from a file" and is part of a presentation on "Data science for Engineers". It contains three bullet points:

- A dataframe can also be created by reading data from a file using the following command
➤ `newDF = read.table(path="Path of the file")`
- In the path, please use `/` instead of `\`
➤ Example: `"C:/Users/hiii/Documents/R/R-Workspace/"`
- A separator can also be used to distinguish between entries. Default separator is space, `' '`
➤ `newDF = read.table(file="path of the file", sep)`

A small video inset in the bottom right corner shows a man speaking. The slide footer includes "Dataframes-1".

Data frames can also be created by importing the data from text file to the way you have to do it is you have to use the function called read dot table and the syntax for that is you assign the data which your reading to a new data frame you have name that data from which you want to create and then read dot table takes in the argument the path of the file.

Let say you have data in a sum text file where the data is separated by spaces, you have to use this command read dot table and path = path of the file which from which you want import the data. This path specification is the os dependent you have to take care of whether you need to use backslash are slash operator depending upon your OS. A separator can also be specified to distinguish between entries the default separated between the entries of data is space, when you want to see the syntax for importing the data and creating a data frame this is how it looks; new data frame is read dot table you have to specify the path of the file and then you can also specify the separator that is being used to separate the entries of data.

The separator can also be a comma or a tab etcetera. So, what we have seen here is you can either create data frames on the go or you can use the data that is already existing in some format and use that to create data frames. Now that we have created a data frame,

(Refer Slide Time: 04:13)

Data science for Engineers

Accessing rows and columns

- `df[val1,val2]` refers to row "val1", column "val2". Can be number or string
- "val1" or "val2" can also be array of values like "1:2" or "c(1,3)"
- `df[val2]` (no commas) - just refers to column "val2" only

Code

Console Output

```
# accessing first & second row:
print(df[1:2,])
# accessing first & second column:
print(df[:,1:2])
# accessing 1st & 2nd column -
# alternate:
print(df[1:2])
```

```
> print(df[1:2,])
  vec1  vec2  vec3
1     1     R  For prototyping
2     2  Scilab  For prototyping
> # accessing first & second column:
> print(df[:,1:2])
  vec1  vec2
1     1     R
2     2  Scilab
3     3   Java
> # accessing 1st & 2nd column - alternate:
> print(df[1:2])
  vec1  vec2
1     1     R
2     2  Scilab
3     3   Java
```

Dataframes-1

5

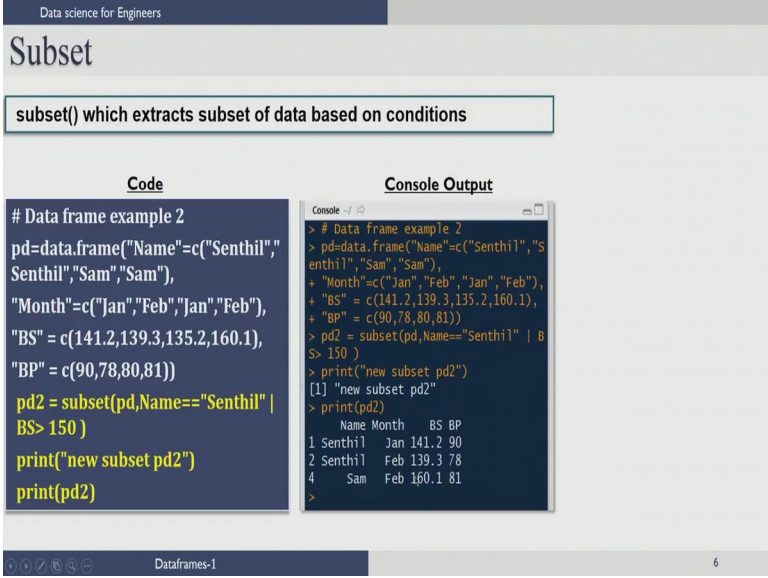
We need to see how we can access rows and columns of a data frame; the syntax for that is the data frame and 2 arguments has to be passed, the first argument val1 refers to the rows of a data frame and the second argument val2 refers to the columns of a data frame. So, this val1 and val2 can be array of values such as one to 2 or c of one comma, etcetera.

If you specify only val2 which is the syntax given here `df[,1:2]` this refers to the set of columns, that you need to access from the data frame. In this code we can see that if you want to access first and second row of the data frame that is created. You can do so by accessing the rows we have to put 1 to 2 comma, nothing in the column specifies all the columns has to be accessed.

You can see in the result from the data frame what you have created in the previous slide you are able to access the first 2 rows. If you want to access the first 2 columns; instead of rows what you need to do is you need to leave a space first and then comma, and you need to specify the list of columns you need to access that is one to 2 that is shown here in this command and we can see the result on the console output, you are able to access the first 2 columns of the data from which you have created. If you want to access the first and second columns using just the column names you can do.

So, by just specifying `df[,1:2]` this is another way of accessing the columns of a data frame..

(Refer Slide Time: 05:58)



Subset

subset() which extracts subset of data based on conditions

Code	Console Output
<pre># Data frame example 2 pd=data.frame("Name"=c("Senthil","Senthil","Sam","Sam"), "Month"=c("Jan","Feb","Jan","Feb"), "BS" = c(141.2,139.3,135.2,160.1), "BP" = c(90,78,80,81)) pd2 = subset(pd,Name=="Senthil" BS> 150) print("new subset pd2") print(pd2)</pre>	<pre>> # Data frame example 2 > pd=data.frame("Name"=c("Senthil","Senthil","Sam","Sam"), + "Month"=c("Jan","Feb","Jan","Feb"), + "BS" = c(141.2,139.3,135.2,160.1), + "BP" = c(90,78,80,81)) > pd2 = subset(pd,Name=="Senthil" BS> 150) > print("new subset pd2") [1] "new subset pd2" > print(pd2) Name Month BS BP 1 Senthil Jan 141.2 90 2 Senthil Feb 139.3 78 4 Sam Feb 160.1 81</pre>

Dataframes-1 6

Sometimes you will be interested in selecting the subset of the data frame; based on certain conditions. So, the way you do is you should have the conditions based on which you have to select the data frame and you should also have a data frame, once you have you can use the command subset to get the subset of data frame.

Let us illustrate by an example. Now we are going to create a data frame by name pd; using the first line which has name month blood sugar and blood pressure as the columns in the name we have Senthil and Sam in the month we have Jan and February, in the blood sugar we have a vector of blood sugar values and in the blood pressure you have a vector of blood pressure values you can print the data frame and see how this it looks.

But in that data frame what I want to extract is a subset where the name has to be Senthil or the blood sugar value has to be greater than 150. Now I can print the new data frame with this p d 2 the result is as shown in the console output here. The original data frame contains all the entries, but the new data from pd2 selects these entries because the first entry is selected because the name is Senthil, the second entry is selected because the name is Senthil the third entry is selected because the blood sugar value is greater than 150.

(Refer Slide Time: 07:30)

The slide is titled "Editing dataframes" and has a subtitle "Dataframes can be edited by direct assignment". It displays R code in a dark-themed editor and a console window. The code creates three vectors: `vec1` (values 1, 2, 3), `vec2` (values "R", "Scilab", "Java"), and `vec3` (values "For prototyping", "For Scaleup", "For prototyping"). These are combined into a dataframe `df`. The console shows the output of `print(df)`, which is a table with 3 rows and 3 columns. The second row, second column value is "R". A final line of code `df[[2]][2] = "R"` is shown, indicating the edit operation.

```
# Introduction to dataframes
vec1 = c(1,2,3)
vec2 = c("R","Scilab","Java")
vec3 = c("For prototyping","For
prototyping","For Scaleup")
df = data.frame(vec1,vec2,vec3)
print(df)
df[[2]][2] = "R"
```

	vec1	vec2	vec3
1	1	R	For prototyping
2	2	R	For prototyping
3	3	Java	For Scaleup

Now we will see how to edit data frames; much like list you can edit the data frames by direct assignment. We have seen this data frame earlier we have vector 1, vector 2, vector 3 containing the elements in them we have created a data frame using this command. We can print that data frame also; now if I want to change the second entry in vector 2 as an R instead of Scilab I can achieve that by using this command I am accessing and I want to replace the element in the second row second column with the string R, when you execute this command `df[2,2] = "R"` what it does is it replaces the entry Scilab with R as shown in the results.

You can see that the Scilab has been replaced with the R, this is how you can edit the data frame by direct assignment.

(Refer Slide Time: 08:38)

Data science for Engineers

Editing dataframes

- A dataframe can also be edited using the `edit()` command
- Create an instance of data frame and use `edit` command to open a table editor, changes can be manually made

Code

```
# Editing a data frame  
myTable = data.frame()  
myTable = edit(myTable)
```

	English	Maths	Science
1	85	99	88
2	80	100	81
3	92	98	92
4	67	90	70
5	76	85	87
6	87	100	92
7	77	78	95

Enter these values in the table
And close the editor

Data Editor

	English	Maths	Science
1	85	99	88
2	80	100	81
3	92	98	92
4	67	90	70
5	76	85	87
6	87	100	92
7	77	78	95
8			

Dataframes-1 8

Next, we see anything a data frame using `edit` command. So, what you need to do for this is you have to create an instance of data frame for example, you can see that I am creating an instance of data frame and naming it as my table by using the command `data dot frame`, this creates an empty data frame and I can use this `edit` command to edit the entries in my data frame. To do that what I have done is I am assigning whatever that is being edited into create a frame my table. So, when I execute this command it will pop up a window, where I can fill in the details what I want to fill in and then when I close this will save the data as a data frame by name my table.

Next, we will see how to add extra rows and columns to the data frame. We will continue the same example which we have used and now let us say we want to add another row to the data frame which you have created earlier.

(Refer Slide Time: 09:45)

Data science for Engineers

Adding extra rows and columns

Extra row can be added with "rbind" function and extra column with "cbind"

Code

continuing from previous example
adding extra row and column:
df = rbind(df,data.frame(vec1=4,
vec2="C", vec3="For Scaleup"))
print("adding extra row")
print(df)
df = cbind(df,vec4=c(10,20,30,40))
print("adding extra col")
print(df)

Console Output

> # continuing from previous example
> # adding extra row and column:
> df = rbind(df,data.frame(vec1=4,
+ vec2="C",
+ vec3="For Scaleup"))
> print("adding extra row")
[1] "adding extra row"
> print(df)
+ vec1 vec2 vec3
1 1 R For prototyping
2 2 Scilab For prototyping
3 3 Java For Scaleup
4 4 C For Scaleup
> df = cbind(df,vec4=c(10,20,30,40))
> print("adding extra col")
[1] "adding extra col"
> print(df)
+ vec1 vec2 vec3 vec4
1 1 R For prototyping 10
2 2 Scilab For prototyping 20
3 3 Java For Scaleup 30
4 4 C For Scaleup 40

Dataframes-1

9

So, we can add extra row using the command r bind and to add an extra column we use the command c bind. Let us see how we can add an extra row using r bind command the syntax for that is r bind, the data frame to which you want add and the entries for the new row you have to add you have to be careful when using r bind because the data types in each column entry should be = the data types that are already existing rows..

So, we are creating another row entry in which we have in the column 1 that is vec1 we have a numeric data type 4, in the column 2 we have a character variable c, in the column 3 we have a character variable for scale up. So, this command adds the row to the data frame and when you print the data frame you can see that row has been added to the original data frame.

Now, let us see how to add a column; adding a column is simple this can be done using a c bind command the syntax for that is c bind the original data frame and the entries for the new column. Now I am going to add a new column call vec4 which contains the entries 10 20 30 40 and when you add a new column to this and print the new data frame, you can see that this vec4 is added to the existing data frame.

(Refer Slide Time: 11:18)

Data science for Engineers

Deleting rows and columns

There are several ways to delete a row/column, some cases are shown below

Code

```
# continuing from previous example
# Deleting rows and columns:
df2 = df[-3:-1]
print(df2)
# conditional deletion:
df3 = df[!names(df) %in% c("vec3")]
print(df3)
df4 = df[df$vec1==3,]
print(df4)
```

A '-' sign before value and before ';' for rows & after ';' for columns

!' means no to those rows /columns which satisfy the condition

```
> print(df2)
  vec2 vec3 vec4
1  vec2 For prototyping 10
2 Scilab For prototyping 20
4    C For Scaleup 40
# conditional deletion:
> df3 = df[!names(df) %in% c("vec3")]
> print(df3)
  vec1 vec2 vec4
1    1    R    10
2    2 Scilab  20
3    3  Java   30
4    4    C   40
> df4 = df[df$vec1==3,]
> print(df4)
  vec1 vec2 vec3 vec4
1    1    R    10
2    2 Scilab For prototyping 20
4    4    C For Scaleup 40
```

Dataframes-1 10

Now we will see how to delete rows and columns in a data frame there are several ways to delete rows and columns; we will see some of them to delete a row or a column, you need to access that row first and then insert a negative sign before that close, it indicates that you had to delete that rows. So, let us see the example here now from the data frame we have if you want to delete the third row and the first column that can be done using this command. So, I want to delete third row so I chose the third row and insert a negative symbol before it.

Similarly, I want to delete a column one I chose that column and then insert a negative symbol before that column. And I am assigning that to new data frame df2 now when I print the df2. You can see in the results we do not have the column vector one and we do not have vector 3 which is what we expected to happen. We can also do conditional deletion of rows and columns as we have seen this command will delete column 3 from the data frame what we have created.

So, the explanation goes as follows; we have a data frame we want to access all the rows in the columns we want to access those columns where there is no vector 3; that means, we want to access vector 2 vector 4 and vector one. So, this exclamatory symbol says no to the columns that are having column name vector 3 and I am assigning that to data from 3 when I print data from 3 you can see that there is no column vec3 in the data

frame which we are looking for, you can also delete the rows where we have an entry 3 by using this command.

So, what you are saying here is access those rows where the element in the vector 1 is not = 3 and we need to access all the columns. So, and we are assigning that 2 data from df df4 and when you print the df4 we can see that the row which is having the entry 3 is deleted from the data frame.

(Refer Slide Time: 13:35)

Data science for Engineers

Manipulating rows – the factor issue

- When character columns are created in a data.frame, they become factors
- Factor variables are those where the character column is split into categories or factor levels

Code	Console Output
<pre># Manipulating rows in data frame # continued from previous page df[3,1]= 3.1 df[3,3]= "Others" print(df)</pre>	<pre>> # Manipulating rows in dataframe > # continued from previous page > df[3,1]= 3.1 > df[3,3]= "Others" Warning message: In `[<-factor` (*tmp*, iseq, value = "Others") : invalid factor level, NA generated > print(df) vec1 vec2 vec3 1 1.0 R For prototyping 2 2.0 Scilab For prototyping 3 3.1 Java NA</pre>

Notice the NA values displayed instead of the string "Others".
Also see the use of the word "factor" in the warning above

Dataframes-1 11

now we will see how to manipulate the rows in the data frame and what is called as a factory issue. R has inbuilt characteristic to assign the data types to the data you enter. When you enter numeric variables, it knows all the numeric variables that are available when you enter character variables it takes whatever the character variables you are giving as categories or factors levels.

And it assumes that these are the only factors that are available for now; when you want to change the element in the third row third column to others; what happens is it will display warning message saying that, this others categorical variable is not available and it replaces that with the NA you can notice that the place where we want others to be there we are having a NA and we can also see the use of word factor in the warning message, how to get rid of the factor issue is the question now.

(Refer Slide Time: 14:43)

Data science for Engineers

Resolving factor issue

New entries need to be consistent with factor levels which are fixed when the dataframe is first created

Code	Console Output
<pre>vec1 = c(1,2,3) vec2 = c("R","Scilab","Java") vec3 = c("For prototyping", "For prototyping","For Scaleup") df = data.frame(vec1,vec2,vec3, stringsAsFactors = F) # Now trying the same manipulation df[3,3]= "Others" print(df)</pre>	<pre>> vec1 = c(1,2,3) > vec2 = c("R","Scilab","Java") > vec3 = c("For prototyping", + "For prototyping","For Scaleup") > df = data.frame(vec1,vec2,vec3, + stringsAsFactors = F) > # Now trying the same manipulation > df[3,3]= "Others" Warning: replacing with factor levels not consistent with current levels > print(df) vec1 vec2 vec3 1 1 R For prototyping 2 2 Scilab For prototyping 3 3 Java Others > </pre>

Dataframes-1 12

New entries in R when you are entering should be consistent with the factor levels that are already defined if not those error message will be printed out. If you do not want this issue to happen what you have to do is while defining the data from itself you need to pass another argument, which says strings as factors is false by default this argument is true that is the reason why you get this warning message when you want to change the string characters into new string characters as an element..

Now try doing the same manipulation you want to change the third row third element to others and print the data frame you can see that there is no NA anymore and we achieved what we want. In this lecture we have seen how to create data, frames how to access rows and columns of data frame and how to delete rows and columns of a data frame and so on.

In the next lecture we are going to see some other operations that can be done on data frames.

Thank you.