Foundations of R Software

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

Data Handling
Lecture - 47
Data Handling - Introduction, Frequency and Partition Values Saving and Writing Data Files

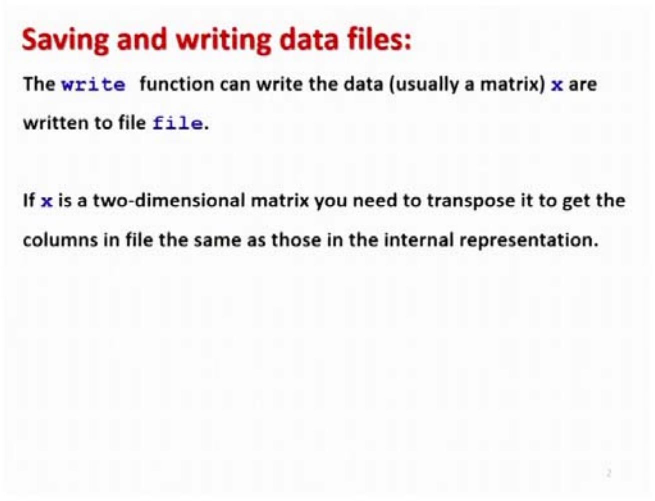
Hello friends, welcome to the course Foundations of R Software and in this lecture we are going to take up the last topic of the Data Handling. Well, I am not saying that this is the last topic in the sense that there are no more topics which are related to the data handling, but possibly in this course today we are going to take up the last topic of this broader topic data handling.

Now, up to now what you have done you have created different types of files, you have learned how to read the file and how to do different types of operation. Now at the end the last question come, how are you going to save the file? For example, whenever you are trying to do some programming there is going to be an outcome there is going to be output. So, the question is up to now, what you have seen you are trying to see the outcome on the screen, right using print, cat, etcetera.

Now, my objective is that I want to save these files to an external file or in simple word I want to save the output in some external files. So, how to get it done that is a very simple topic and in R there are couple of ways in which you can do it, but to make the life simple and to make this presentation simple and because we are learning at an very elementary course. He will try to consider here some elementary operation through which you can save the outcome of your program in a file.

So, whenever I am trying to do it yeah you have to first specify the location of the folder in which you want to save the file, otherwise that will be saved in the default folder of the R software. So, that is up to you where you want to save your file, right. So, let us try to begin this lecture and try to see how we can save the output of a command into a file ok.

(Refer Slide Time: 02:20)



Saving and writing data files:

The `write` function can write the data (usually a matrix) `x` are written to file `file`.

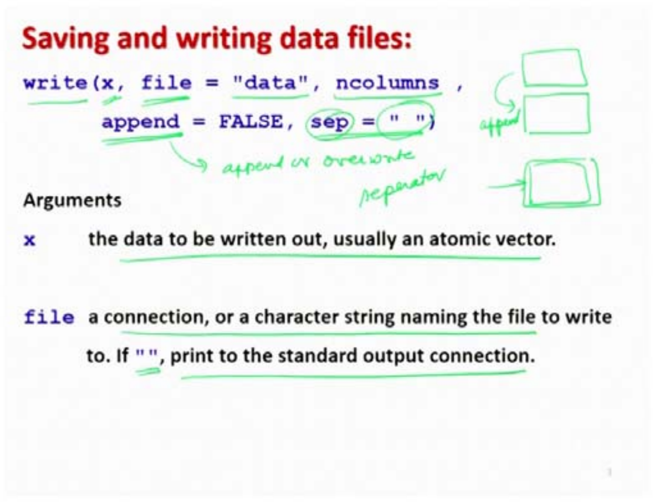
If `x` is a two-dimensional matrix you need to transpose it to get the columns in file the same as those in the internal representation.

So, we have a function here write w r i t e. So, you see as the name suggest write the job of this function is to write the data to a file, right. So, if I say if I want to write a data which is here x and I want to write it inside a file. Then usually you will see that with this write function the data in x is usually in the form of a matrix, right.

And in case if you want to make some changes you have to do it accordingly, for example, if x is a two-dimensional matrix and if you need to transpose it. So, that columns in the file are matching with the columns in the outcome then you need to transpose it, right. So, I will say very simply that you have to look that how your data is arranged in the outcome and how it is saved that you have to look into the behavior of the R software that how it is trying to save and then if you want to do it in other way.

You have to use a proper logic and then you have to convert it in the way you want, it is as simple as that, right.

(Refer Slide Time: 03:34)



Saving and writing data files:

```
write(x, file = "data", ncolumns ,
      append = FALSE, sep = " ")
```

append or overwrite
separator
append

Arguments

x	the data to be written out, usually an atomic vector.
file	a connection, or a character string naming the file to write to. If " ", print to the standard output connection.

And when you are trying to use the function here, right then there are many many options which can help you in saving the file in the format you want.

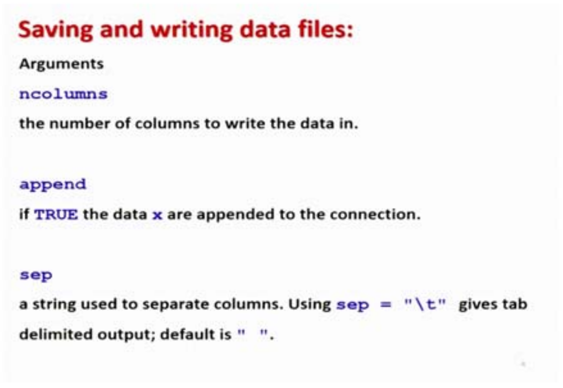
So, for example, if I say here x; so, x is the data which is to be written usually an atomic vector and after this I have to give here the name of the file. So, I write down here f i l e all in lower case alphabet and is equal to within double quotes you have to give the name of the file, right.

So, this file is indicating a connection or a character string naming the file to write to it, right and if you try to use that double quotes the print to the standard output connection is done. And similarly if you want to give that how many columns should be there in the outcome that you can control by the option n c o l u m n s then after that there is an option of here, append; means append means for example, you are trying to execute a program and some program comes here now you try to re execute the program.

So, now there will be next outcome. So, what do you want do you want to write it after this or means do you want to add it or do you want to append it or this is your earlier outcome and if you are coming with the newer outcome this is going to be just overwritten on it. So, append is trying to control the option whether you want to append or overwrite, right.

So, it takes value in terms of TRUE and FALSE, so if you say this append is equal to false; that means, the data is going to be overwritten, right. Now, similarly you have here another option here sep; that means, separator means how the values are going to be separated and within the double quotes you have to give the symbol or blank space whatever you want here, right.

(Refer Slide Time: 05:32)



Saving and writing data files:

Arguments

`ncolumns`

the number of columns to write the data in.

`append`

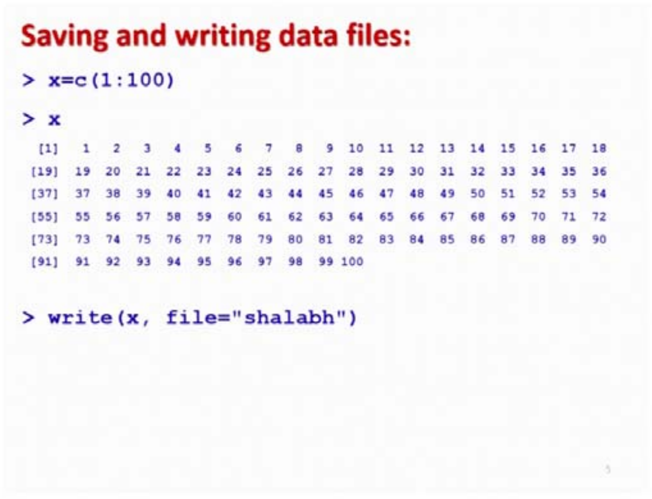
if `TRUE` the data `x` are appended to the connection.

`sep`

a string used to separate columns. Using `sep = "\t"` gives tab delimited output; default is " ".

So, and in case if you try to use here is equal to backslash t then it will give you a tab delimited output; that means, the values are separated by a tab, right. So, similarly you can see here these are here n columns and append, right.

(Refer Slide Time: 05:51)

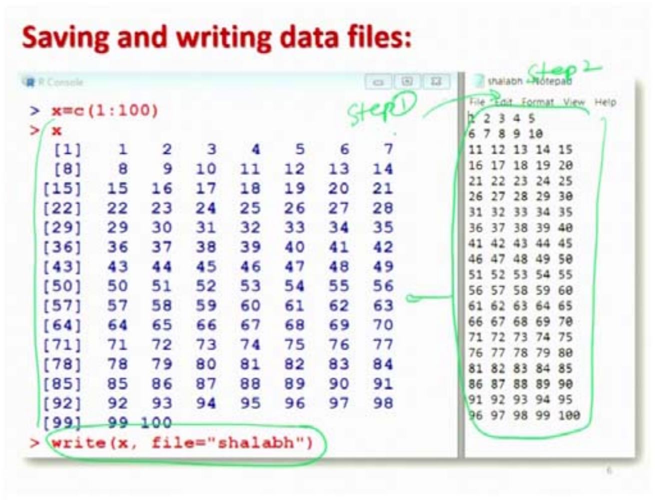


Saving and writing data files:

```
> x=c(1:100)
> x
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100

> write(x, file="shalabh")
```

So, let me try to show you here an output an example and I will try to show you that the outcome how the outcome is going to be diverted to a file, right. So, just for the sake of simplicity I try to take here the numbers from 1 to 100 and which are stored in the variable here x. Now, I want to write this x in a file whose name is shalabh, right. So, what I try to do here that I try to write down here write and then I try to write down here x and then file name and then here s h a l a b h, right.

(Refer Slide Time: 06:30)



Saving and writing data files:

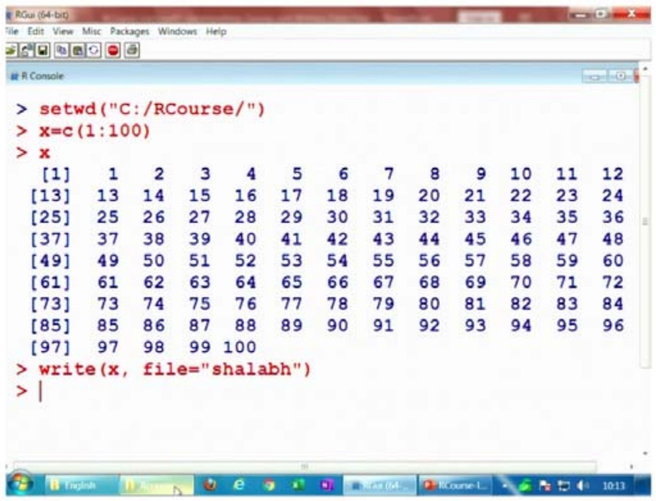
```
> x=c(1:100)
> x
  [1]   1   2   3   4   5   6   7
  [8]   8   9  10  11  12  13  14
 [15]  15  16  17  18  19  20  21
 [22]  22  23  24  25  26  27  28
 [29]  29  30  31  32  33  34  35
 [36]  36  37  38  39  40  41  42
 [43]  43  44  45  46  47  48  49
 [50]  50  51  52  53  54  55  56
 [57]  57  58  59  60  61  62  63
 [64]  64  65  66  67  68  69  70
 [71]  71  72  73  74  75  76  77
 [78]  78  79  80  81  82  83  84
 [85]  85  86  87  88  89  90  91
 [92]  92  93  94  95  96  97  98
 [99]  99 100
> write(x, file="shalabh")
```

```
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
26 27 28 29 30
31 32 33 34 35
36 37 38 39 40
41 42 43 44 45
46 47 48 49 50
51 52 53 54 55
56 57 58 59 60
61 62 63 64 65
66 67 68 69 70
71 72 73 74 75
76 77 78 79 80
81 82 83 84 85
86 87 88 89 90
91 92 93 94 95
96 97 98 99 100
```

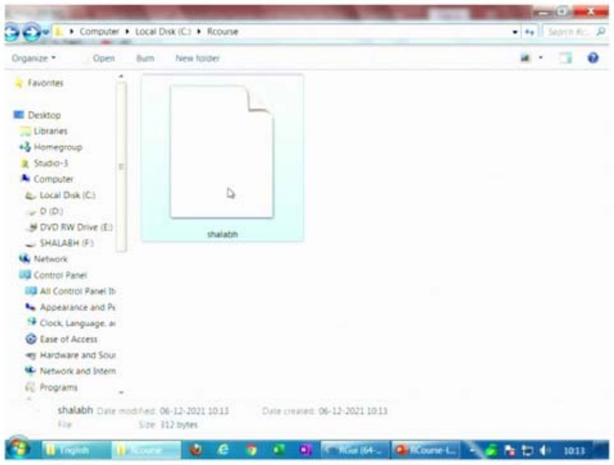
So, if you try to see what will happen here on the R console what I am going to do yeah first I have to change my directory. So, that I can show you where the file is going to be then I will write down here this data value here x. And then I execute the command here, right and then you will see here that in the notepad I can open a file here this will be opened after this.

So, I will try to show you, so this is going to be your here is step 1 and this is going to be a step 2, where you will see that all these 100 values which you have generated in the x they are stored here, right. So, let me try to show you this operation on the R console. So, that you become me confident ok.

(Refer Slide Time: 07:12)


```
> setwd("C:/RCourse/")
> x=c(1:100)
> x
  [1]   1   2   3   4   5   6   7   8   9  10  11  12
 [13]  13  14  15  16  17  18  19  20  21  22  23  24
 [25]  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48
 [49]  49  50  51  52  53  54  55  56  57  58  59  60
 [61]  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84
 [85]  85  86  87  88  89  90  91  92  93  94  95  96
 [97]  97  98  99 100
> write(x, file="shalabh")
> |
```

(Refer Slide Time: 07:33)


Computer ▸ Local Disk (C:) ▸ Rcourse

shalabh

shalabh Date modified: 06-12-2021 10:13 Date created: 06-12-2021 10:13
File Size: 312 bytes

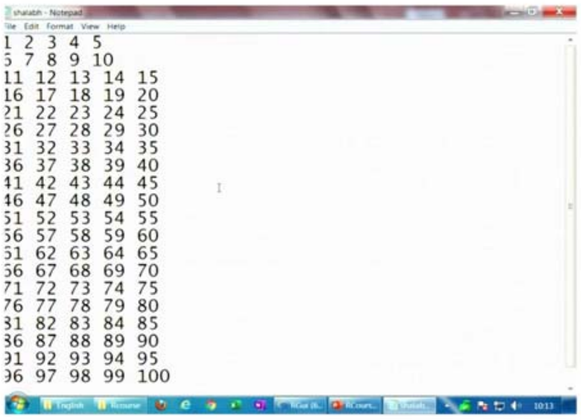
So, first you could see as usual I will try to change my working directory as to C colon R course. So, I have created a folder R course on the free drive of my computer. So, I set it here and yeah you can see here that this is here the folder R course and you can see here very clearly that this folder is completely empty.

You can see here it is written the folder is empty, right, so that you can believe on me that ok whatever is the outcome that is going to be generated and that is going to be the same.

So, now I try to come here on the R console and I try to generate here say x is equal to here 1 to 100 like this, so you can see here x here looks like this, right. And then I try to write down here write x and inside the file name here shalabh. So, if you try to see here I am trying to execute it here and there is no outcome here.

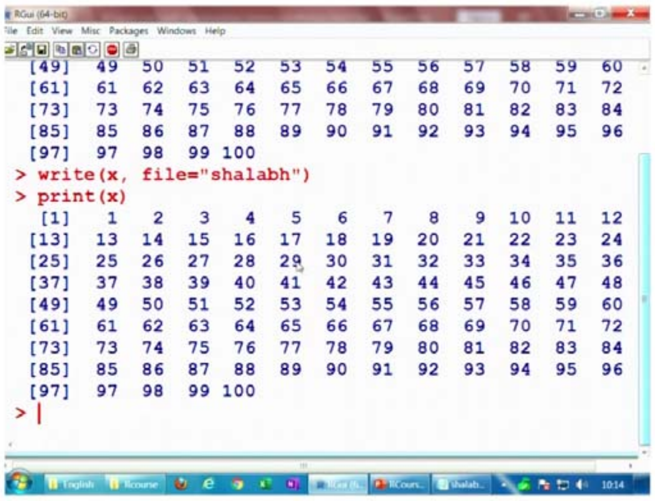
Because the outcome has gone to this here R course if you come here you can see here means just before that there was no file and now there is a file, and if you try to open it here you can see here this is here the file, right.

(Refer Slide Time: 08:21)



```
1 2 3 4 5
5 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
26 27 28 29 30
31 32 33 34 35
36 37 38 39 40
41 42 43 44 45
46 47 48 49 50
51 52 53 54 55
56 57 58 59 60
61 62 63 64 65
66 67 68 69 70
71 72 73 74 75
76 77 78 79 80
81 82 83 84 85
86 87 88 89 90
91 92 93 94 95
96 97 98 99 100
```

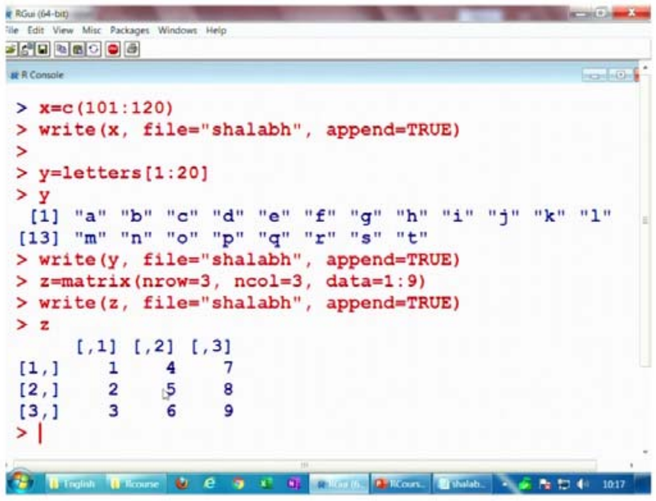
(Refer Slide Time: 08:39)



```
 [49]  49  50  51  52  53  54  55  56  57  58  59  60
 [61]  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84
 [85]  85  86  87  88  89  90  91  92  93  94  95  96
 [97]  97  98  99 100
> write(x, file="shalabh")
> print(x)
  [1]   1   2   3   4   5   6   7   8   9  10  11  12
 [13]  13  14  15  16  17  18  19  20  21  22  23  24
 [25]  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48
 [49]  49  50  51  52  53  54  55  56  57  58  59  60
 [61]  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84
 [85]  85  86  87  88  89  90  91  92  93  94  95  96
 [97]  97  98  99 100
> 
```

So, you can see here that when I am trying to execute it here. Now if you try to see what is the difference between this, right and x, if you try to say here use here print x. So, what this print x is going to do? Print x is simply going to show you the output on the R console only, right, but when you are trying to do here, right then write is going to save the output in the file ok. Now, let me try to show you here one thing here more.

(Refer Slide Time: 09:10)

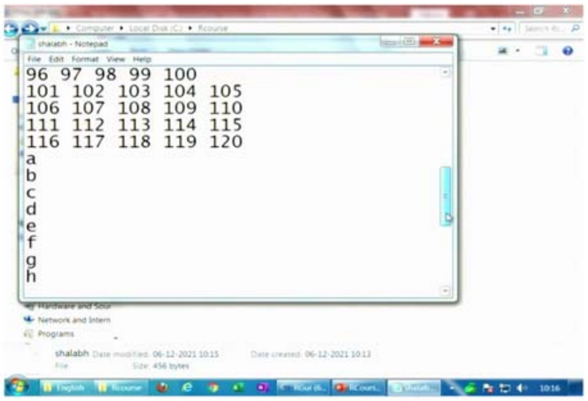


```
> x=c(101:120)
> write(x, file="shalabh", append=TRUE)
>
> y=letters[1:20]
> y
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
[13] "m" "n" "o" "p" "q" "r" "s" "t"
> write(y, file="shalabh", append=TRUE)
> z=matrix(nrow=3, ncol=3, data=1:9)
> write(z, file="shalabh", append=TRUE)
> z
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
```

Now, suppose I try to generate here one more here x say x is equal to suppose c say 101 to say 120. So, this is here like this, right now here x and now I want to show you that when I try to write down this outcome to this here file and if I give here append is equal to here TRUE then what happens, right.

So, let me try to execute it here and you will yeah let me try to close the file first yeah and then if you try to see here I have executed it and then now if you try to come to this file here.

(Refer Slide Time: 09:50)



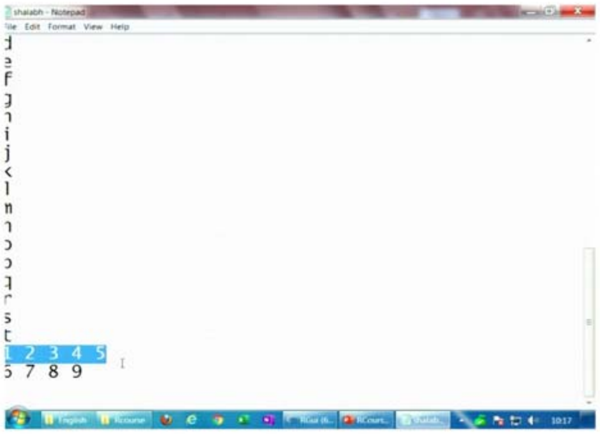
```
96 97 98 99 100
101 102 103 104 105
106 107 108 109 110
111 112 113 114 115
116 117 118 119 120
a
b
c
d
e
f
g
h
```

And if you try to see here this file here you can see here, now there are values here 101 to 120 which are added here, right.

And similarly if you want to get here more confidence then what I can do here that I try to take here say here y is equal to l e t t e r s say letters 1 to say here 20, right. And if you try to see here y now here is like this alphabet and I try to write down the same here y in the same file whose name was shalabh and I am saying here append is equal to TRUE, right.

So, now after I execute till it look let us try to see what happens to the file shalabh. So, you could see here now if you try to see at the end what is happening here.

(Refer Slide Time: 10:41)

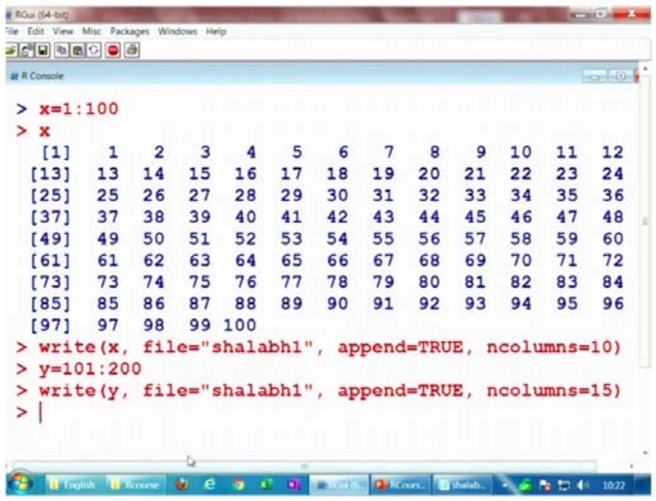


```
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
1 2 3 4 5
6 7 8 9
```

You can see here that all the alphabets they have come here, right and similarly if you try to see here, for one more example here say z is equal to here say here matrix now let me try to say here nrow is equal to support 3, ncol is equal to 3 and data is equal to 1 to 9, right.

So, now if you try to write down the same z in this file here what happens if you try to see now here? Now what will happen I try to open here this here file here shalabh and then you can see here after this the 20th letter all these values here they have come like this, right. So, and if you try to see here how this z looks like it is like here 1 to 9 number, but they are arranged column wise. And in this file you can see here all the 1 2 3 4 5 6 7 8 9 they have just arranged in this way, right.

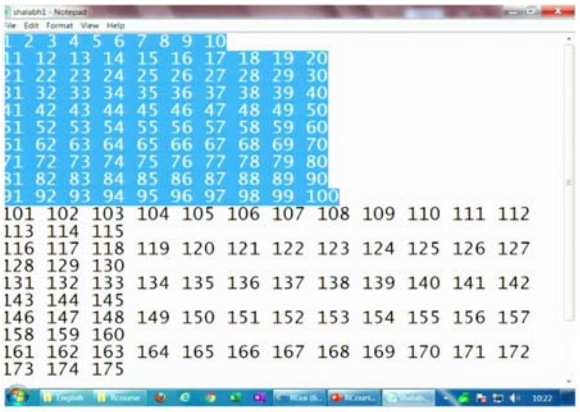
(Refer Slide Time: 11:58)



```
> x=1:100
> x
  [1]   1   2   3   4   5   6   7   8   9  10  11  12
 [13]  13  14  15  16  17  18  19  20  21  22  23  24
 [25]  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48
 [49]  49  50  51  52  53  54  55  56  57  58  59  60
 [61]  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84
 [85]  85  86  87  88  89  90  91  92  93  94  95  96
 [97]  97  98  99 100
> write(x, file="shalabh1", append=TRUE, ncolumns=10)
> y=101:200
> write(y, file="shalabh1", append=TRUE, ncolumns=15)
> |
```

And similarly in case if you try to say here say if I try to take here x is equal to here suppose 1 to here 100 now once again and then if you try to see this is your here x. And now what I try to do here that I try to create here say this n columns is equal to 10; that means, now I want to store this data into 10 columns and then I try to create here another file here shalabh 1, right and then I try to write down the this here x. So, if you try to see here what happened?

(Refer Slide Time: 12:28)

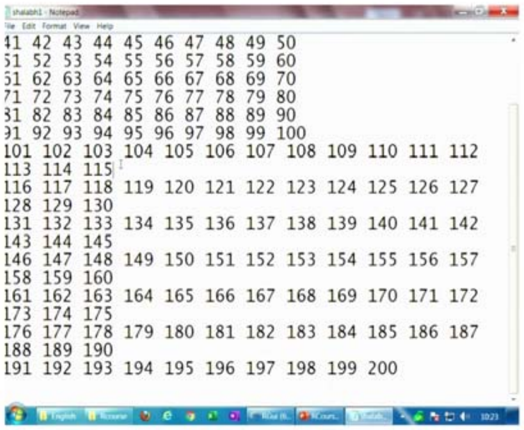


```
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100
101 102 103 104 105 106 107 108 109 110 111 112
113 114 115
116 117 118 119 120 121 122 123 124 125 126 127
128 129 130
131 132 133 134 135 136 137 138 139 140 141 142
143 144 145
146 147 148 149 150 151 152 153 154 155 156 157
158 159 160
161 162 163 164 165 166 167 168 169 170 171 172
173 174 175
```

Now, there is a new file shalabh 1 which is created here and if you try to see here now the data is arranged here from 1 to 100, right. Now similarly if you come here and try to generate here one more y and then you try to see here that I try to generate the data from 101 to 200. And then you try to write down this y in the same file here shalabh 1, but now suppose I in try to increase my number of columns let us see what happens and the append is equal to here true.

So, right if you try to now see here what will happen here that I try to open this file once again you can see here that 101 to here these values here you can see here up to 1 to 100 they were stored in 10 columns, but 101 onwards they have been stored in a data file which has 15 columns, right.

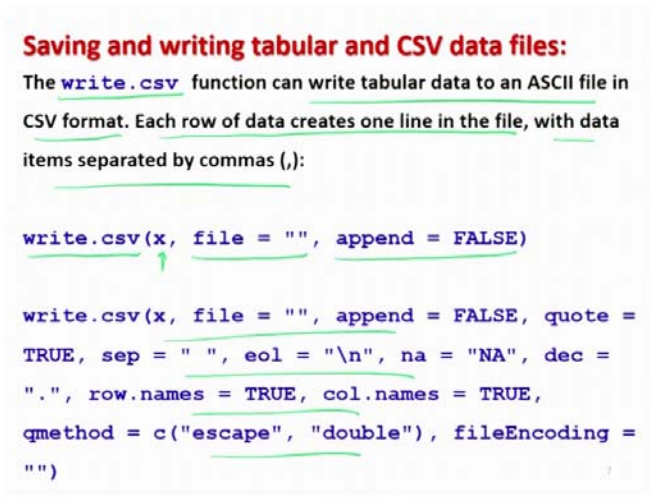
(Refer Slide Time: 13:23)



```
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100
101 102 103 104 105 106 107 108 109 110 111 112
113 114 115
116 117 118 119 120 121 122 123 124 125 126 127
128 129 130
131 132 133 134 135 136 137 138 139 140 141 142
143 144 145
146 147 148 149 150 151 152 153 154 155 156 157
158 159 160
161 162 163 164 165 166 167 168 169 170 171 172
173 174 175
176 177 178 179 180 181 182 183 184 185 186 187
188 189 190
191 192 193 194 195 196 197 198 199 200
```

You can see here this is the actually first row which has 15 values this is here the second row which has the 15 values and so on. So, this is how you can see that we can proceed and we can save these files in different alphabets, right.

(Refer Slide Time: 13:36)



Saving and writing tabular and CSV data files:

The `write.csv` function can write tabular data to an ASCII file in CSV format. Each row of data creates one line in the file, with data items separated by commas (,):

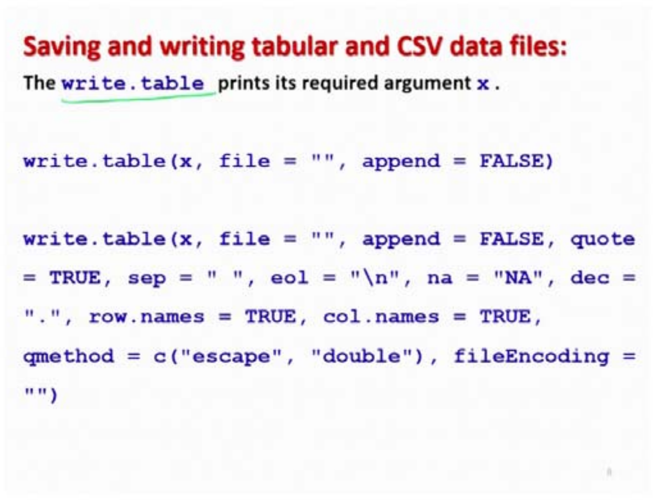
```
write.csv(x, file = "", append = FALSE)

write.csv(x, file = "", append = FALSE, quote =
TRUE, sep = " ", eol = "\n", na = "NA", dec =
".", row.names = TRUE, col.names = TRUE,
qmethod = c("escape", "double"), fileEncoding =
"")
```

And now similarly I have given you one example in detail and after that I believe that otherwise if you want to save other types of file then you simply have to look that what is the correct command, right, and after this the functioning will remain the same, right ok. So, if you try to suppose if I want to know that how I can save a tabular or CSV data files.

So, now what you have to look into the R software and then try to find what is this function. So, the function will you will find is write dot csv; w r i t e dot c s v. So, this function can write the tabular data to an ASCII file in the CSV format and the way it writes the data is like each row of the data creates one line in the file and the data items are separated by commas.

So, for that also you have a simple command here write dot csv and then here x you try to give here the file name and then you can use here append is equal to TRUE or FALSE exactly in the same way as you have done in the earlier case. And after that you will see here there are many more options and I have just I given you here this idea that how you can use them.

(Refer Slide Time: 14:40)



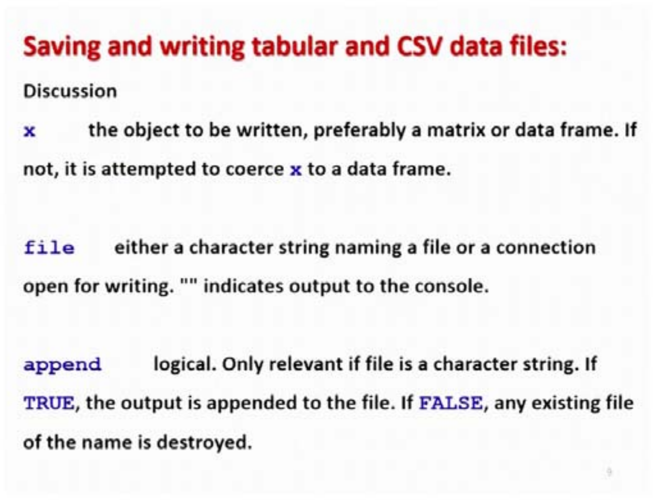
Saving and writing tabular and CSV data files:

The `write.table` prints its required argument `x` .

```
write.table(x, file = "", append = FALSE)

write.table(x, file = "", append = FALSE, quote
= TRUE, sep = " ", eol = "\n", na = "NA", dec =
".", row.names = TRUE, col.names = TRUE,
qmethod = c("escape", "double"), fileEncoding =
"")
```

And similarly if you want to write the tabular data file. So, similarly you have a command here, right dot table w r i t e dot t a b l e and the format is the same write dot table and within the parenthesis you try to write down the content the outcome and then the name of the file and then you have to use append is equal to TRUE or FALSE.

(Refer Slide Time: 15:08)



Saving and writing tabular and CSV data files:

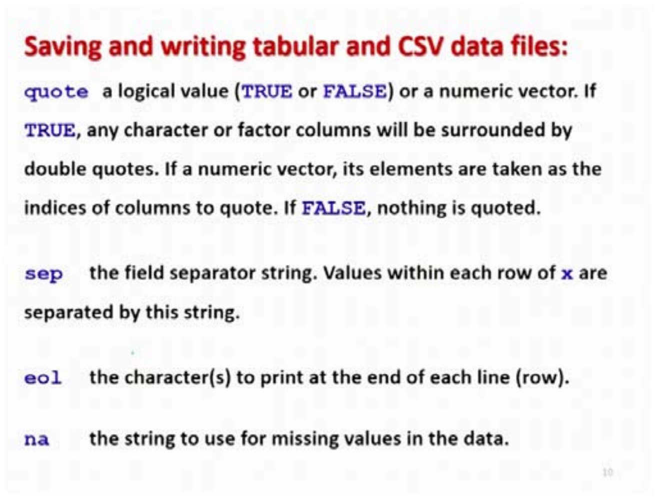
Discussion

`x` the object to be written, preferably a matrix or data frame. If not, it is attempted to coerce `x` to a data frame.

`file` either a character string naming a file or a connection open for writing. "" indicates output to the console.

`append` logical. Only relevant if file is a character string. If TRUE, the output is appended to the file. If FALSE, any existing file of the name is destroyed.

(Refer Slide Time: 15:15)



Saving and writing tabular and CSV data files:

`quote` a logical value (`TRUE` or `FALSE`) or a numeric vector. If `TRUE`, any character or factor columns will be surrounded by double quotes. If a numeric vector, its elements are taken as the indices of columns to quote. If `FALSE`, nothing is quoted.

`sep` the field separator string. Values within each row of `x` are separated by this string.

`eol` the character(s) to print at the end of each line (row).

`na` the string to use for missing values in the data.

10

And after that similar to write dot csv we have here many more options which well I am trying to give you here x. I already have explained you, file I already have explained you, append I already have explained you and after that yeah you can see here code separator eol, na etc., right they can be used here. For example separator you know what is this, right.

And if you want to print character at the end of each line that is a row then you have to use the here the option eol if you want to handle the missing values then you have an option here na, right and so on. And then there are complete details and examples if you try to look into the help of this function, right.

So, now I come to an end to this lecture and you can see that it was a pretty simple lecture and what I have done I have shown you the examples related to the write and then I have taken couple of examples. So, that I can convince you that write function can work in different ways and you can save the file in different ways you want, right.

Similar is the command to write the CSV file tabular data files and there are many more say such structures, but I will leave them up to you I have given you the basic idea, now you simply have to look into the correct function or correct command to execute those actions.

And after that yeah please do not forget to look into the help menu and try to read that what are the different options, because whenever you are trying to generate something you always need it in a format; so, that this file can be used for some further execution, right. It is possible that the outcome file which you have created here that is some data file and that is going to be used as an input in some other program.

So, try to be careful that what type of format do you want and this format should be compatible for the use for the next job. So, now it is your turn once again that try to take some example, try to experiment it, try to write down the names of the file, try to change them, try to use different option and after that you try to look into the file for sure that the type of changes what you wanted are they being executed by the R in that file or not and whether those effects are present in the file or not.

And I will see you in the next lecture with a new topic, till then good bye.