Introduction to R Software

Swayam Prabha

Lecture 31

Importing, Reading and Saving Data Files

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

Slides can be downloaded from

http://home.iitk.ac.in/~shalab/sp

Setting up directories

❑ We can change the current working directory as follows:

```
> setwd("<location of the dataset>")
```

Example:

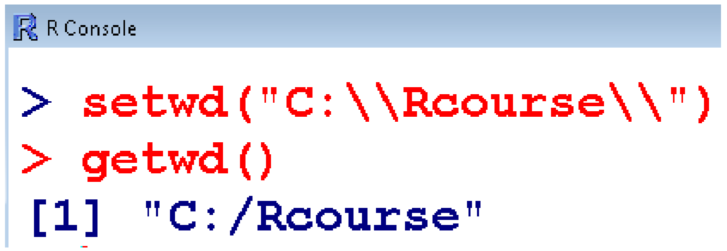
```
> setwd("C:/RCourse/")
```

or

```
> setwd("C:\\Rcourse\\")
```

❑ The following command returns the current working directory:

```
> getwd()
[1] "C:/RCourse/"
```



```
> setwd("C:\\Rcourse\\")
> getwd()
[1] "C:/Rcourse"
```

Importing Data Files

Suppose we have some data on our computer and we want to import it in R.

Different formats of files can be read in R

- **comma-separated values (CSV) data file,**
- **table file (TXT),**
- **Spreadsheet (e.g., MS Excel) file,**
- **HTML table files**
- **files from other software like SPSS, Minitab etc.**

Importing Data Files

One can also read or upload the file from Internet site.

We can read the file containing rent index data from website:

http://home.iitk.ac.in/~shalab/Rcourse/munichdata.asc

as follows

```
> datamunich <- read.table(file=
"http://home.iitk.ac.in/~shalab/Rcourse/munichd
ata.asc", header=TRUE)
```

File name is `munichdata.asc`

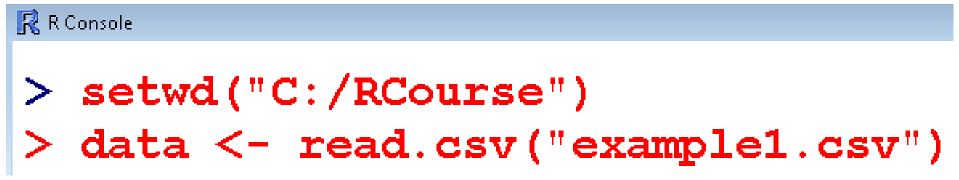
Importing Data Files

Comma-separated values (CSV) files

First set the working directory where the CSV file is located.

```
setwd("<location of your dataset>")
```

```
> setwd("C:/RCourse/")
```



```
> setwd("C:/RCourse")
> data <- read.csv("example1.csv")
```

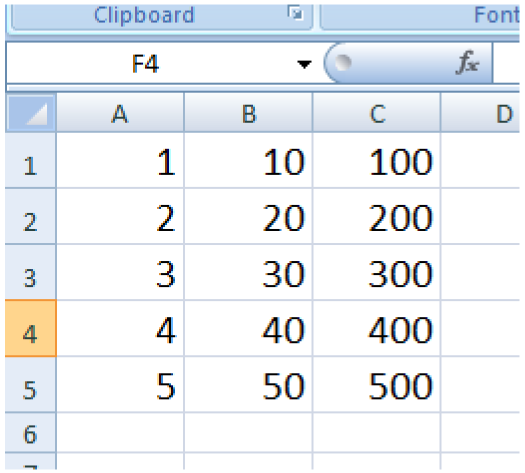
To read a CSV file
Syntax: `read.csv("filename.csv")`

Example:

```
> data <- read.csv("example1.csv")
```

Importing Data Files

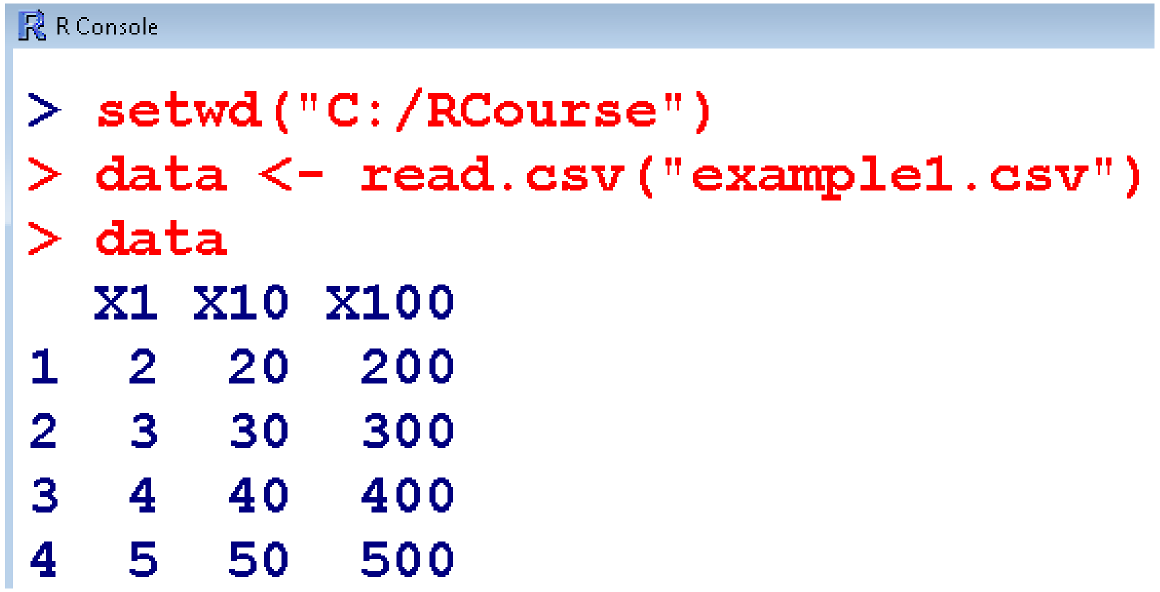
Comma-separated values (CSV) files



	A	B	C	D
1	1	10	100	
2	2	20	200	
3	3	30	300	
4	4	40	400	
5	5	50	500	
6				

Example:

```
> data <- read.csv("example1.csv")
> data
  X1 X10 X100
1  2  20  200
2  3  30  300
3  4  40  400
4  5  50  500
```



```
> setwd("C:/RCourse")
> data <- read.csv("example1.csv")
> data
  X1 X10 X100
1  2  20  200
2  3  30  300
3  4  40  400
4  5  50  500
```

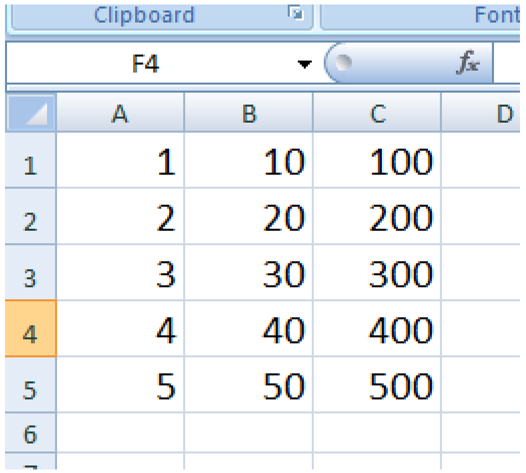
Notice the difference in the first rows of excel file and output

Importing Data Files

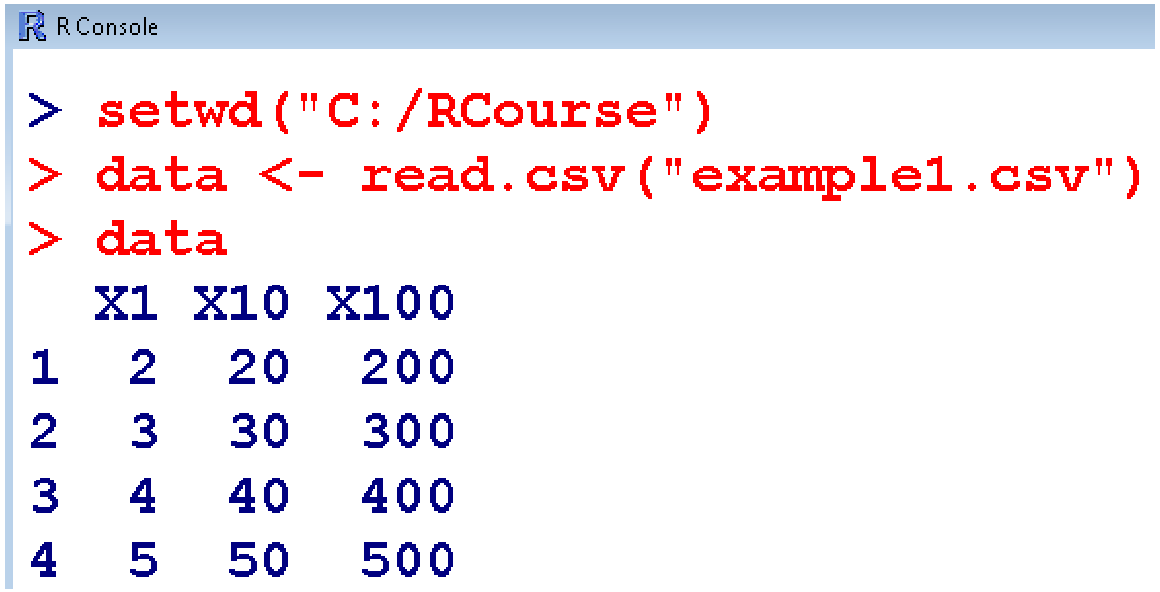
Comma-separated values (CSV) files

Notice the difference in the first rows of excel file and output

Example:



	A	B	C	D
1	1	10	100	
2	2	20	200	
3	3	30	300	
4	4	40	400	
5	5	50	500	
6				



```
> setwd("C:/RCourse")
> data <- read.csv("example1.csv")
> data
  X1 X10 X100
1  2  20  200
2  3  30  300
3  4  40  400
4  5  50  500
```

Importing Data Files

Comma-separated values (CSV) files

Data files have many formats and accordingly we have options for loading them.

If the data file does not have headers in the first row, then use

```
data <- read.csv("datafile.csv", header=FALSE)
```

Importing Data Files

Comma-separated values (CSV) data

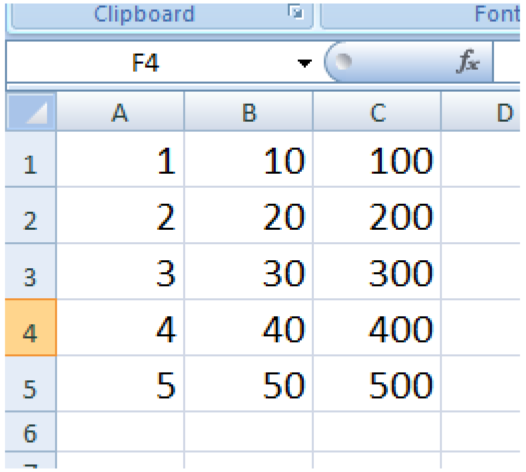
Example:

```
> data <- read.csv("example1.csv", header=FALSE)

> data
  V1 V2  V3
1  1 10 100
2  2 20 200
3  3 30 300
4  4 40 400
5  5 50 500
```

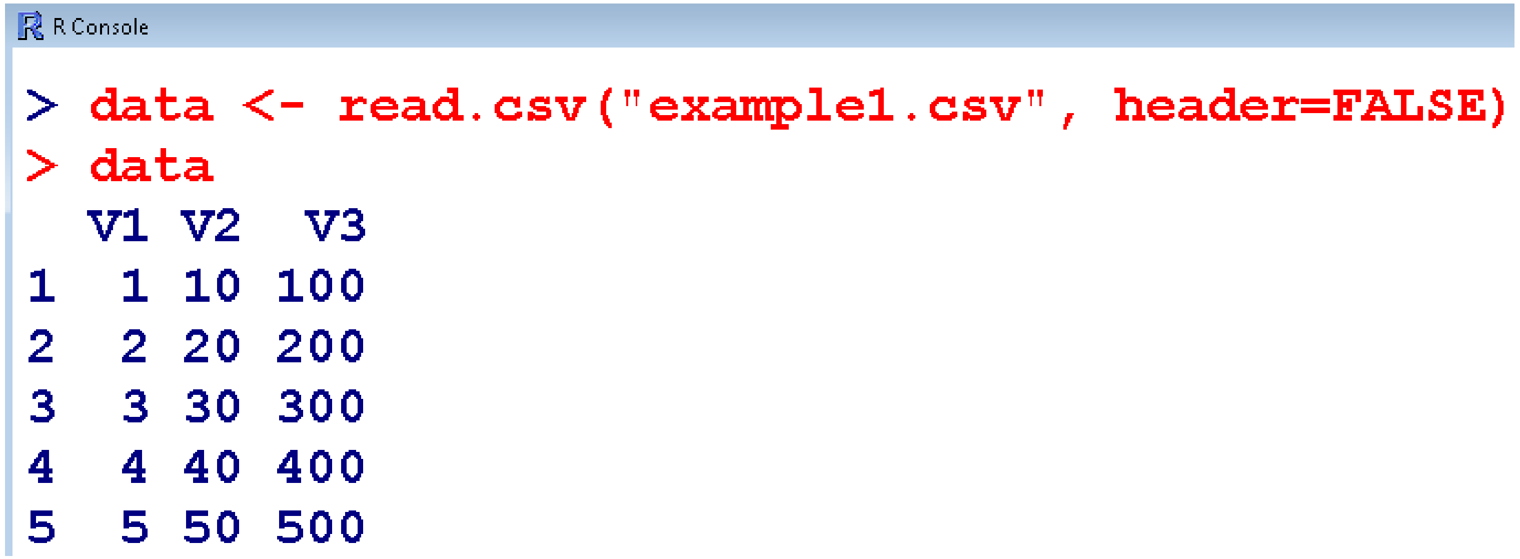
Importing Data Files

Comma-separated values (CSV) data

Example:



	A	B	C	D
1	1	10	100	
2	2	20	200	
3	3	30	300	
4	4	40	400	
5	5	50	500	
6				



```
> data <- read.csv("example1.csv", header=FALSE)
> data
  V1 V2  V3
1  1 10 100
2  2 20 200
3  3 30 300
4  4 40 400
5  5 50 500
```

Importing Data Files

Comma-separated values (CSV) files

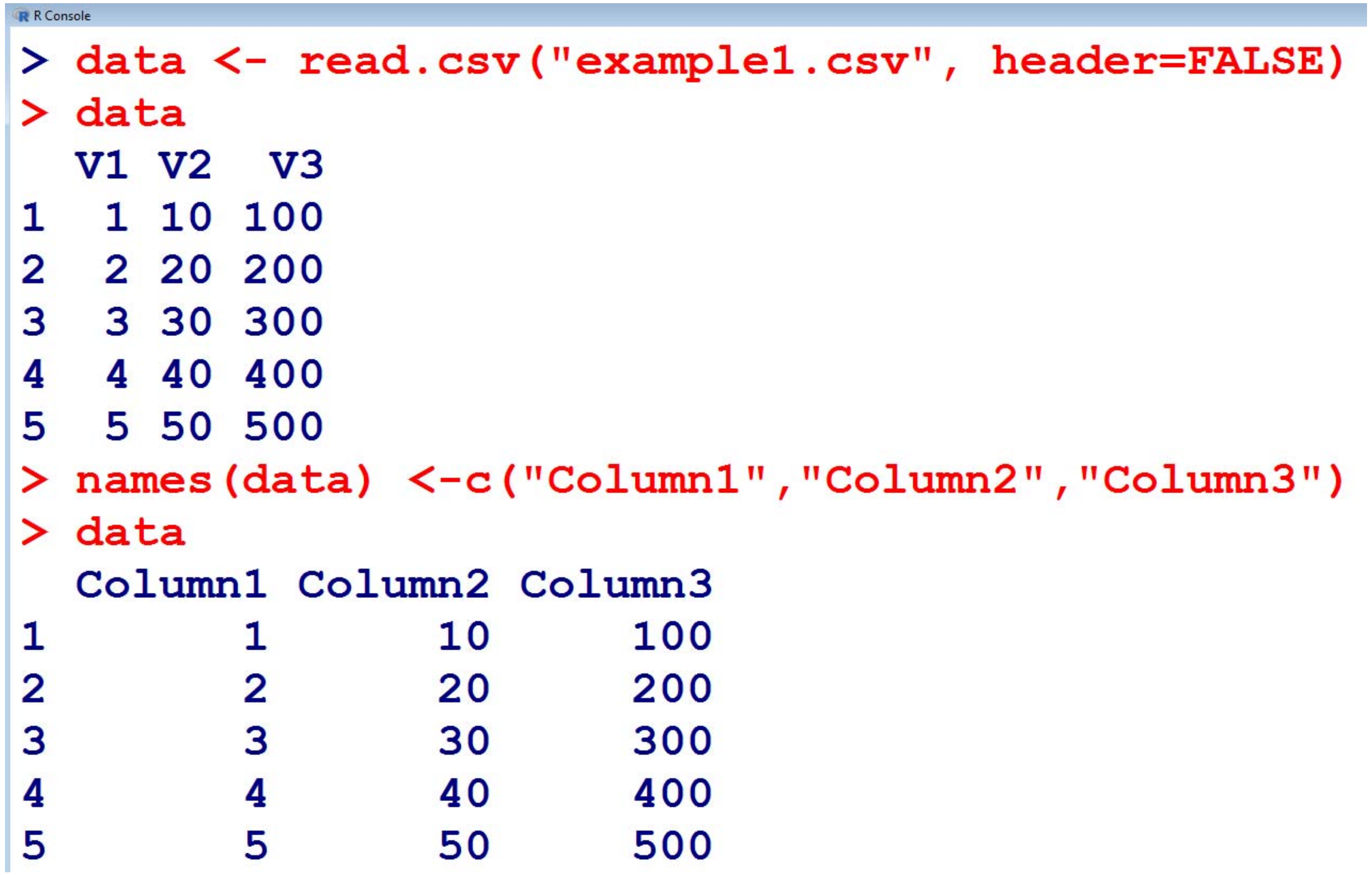
The resulting data frame will have columns named V1, V2, …

We can rename the header names manually:

```
> names(data) <-
c("Column1","Column2","Column3")


> data
  Column1 Column2 Column3
1       1      10     100
2       2      20     200
3       3      30     300
4       4      40     400
5       5      50     500
```

Importing Data Files

Comma-separated values (CSV) files



```
> data <- read.csv("example1.csv", header=FALSE)
> data
  V1 V2  V3
1  1 10 100
2  2 20 200
3  3 30 300
4  4 40 400
5  5 50 500
> names(data) <-c("Column1","Column2","Column3")
> data
  Column1 Column2 Column3
1       1      10     100
2       2      20     200
3       3      30     300
4       4      40     400
5       5      50     500
```

Importing Data Files

Comma-separated values (CSV) files

We can set the delimiter with `sep`.

If it is tab delimited, use `sep="\t"`.

```
data <- read.csv("datafile.csv", sep="\t")
```

If it is space-delimited, use `sep=" "`.

```
data <- read.csv("datafile.csv", sep=" ")
```

Importing Data Files

Reading Tabular Data Files

Tabular data files are text files with a simple format:

- Each line contains one record.
- Within each record, fields (items) are separated by a one-character delimiter, such as a space, tab, colon, or comma.
- Each record contains the same number of fields.

We want to read a text file that contains a table of data.

`read.table` function is used and it returns a data frame.

```
read.table("FileName")
```

Importing Data Files

Reading Tabular Data Files

Data:

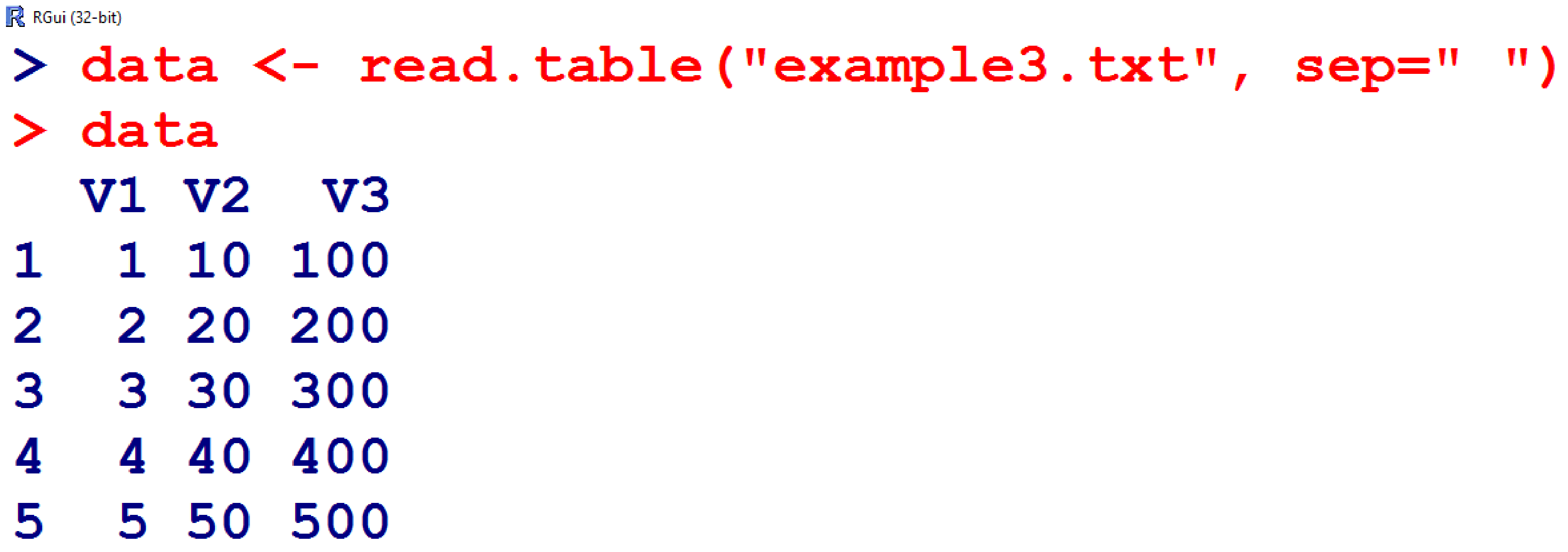
```
1 10 100
2 20 200
3 30 300
4 40 400
5 50 500
```

Saved in example3.txt

```
> data <- read.table("example3.txt", sep=" ")
> data
  V1 V2  V3
1  1 10 100
2  2 20 200
3  3 30 300
4  4 40 400
5  5 50 500
```

Importing Data Files

Reading Tabular Data Files



```
> data <- read.table("example3.txt", sep=" ")
> data
  V1 V2  V3
1  1 10 100
2  2 20 200
3  3 30 300
4  4 40 400
5  5 50 500
```

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.

```
write(x, file = "data",
      ncolumns = if(is.character(x)) 1 else 5,
      append = FALSE, sep = " ")
```

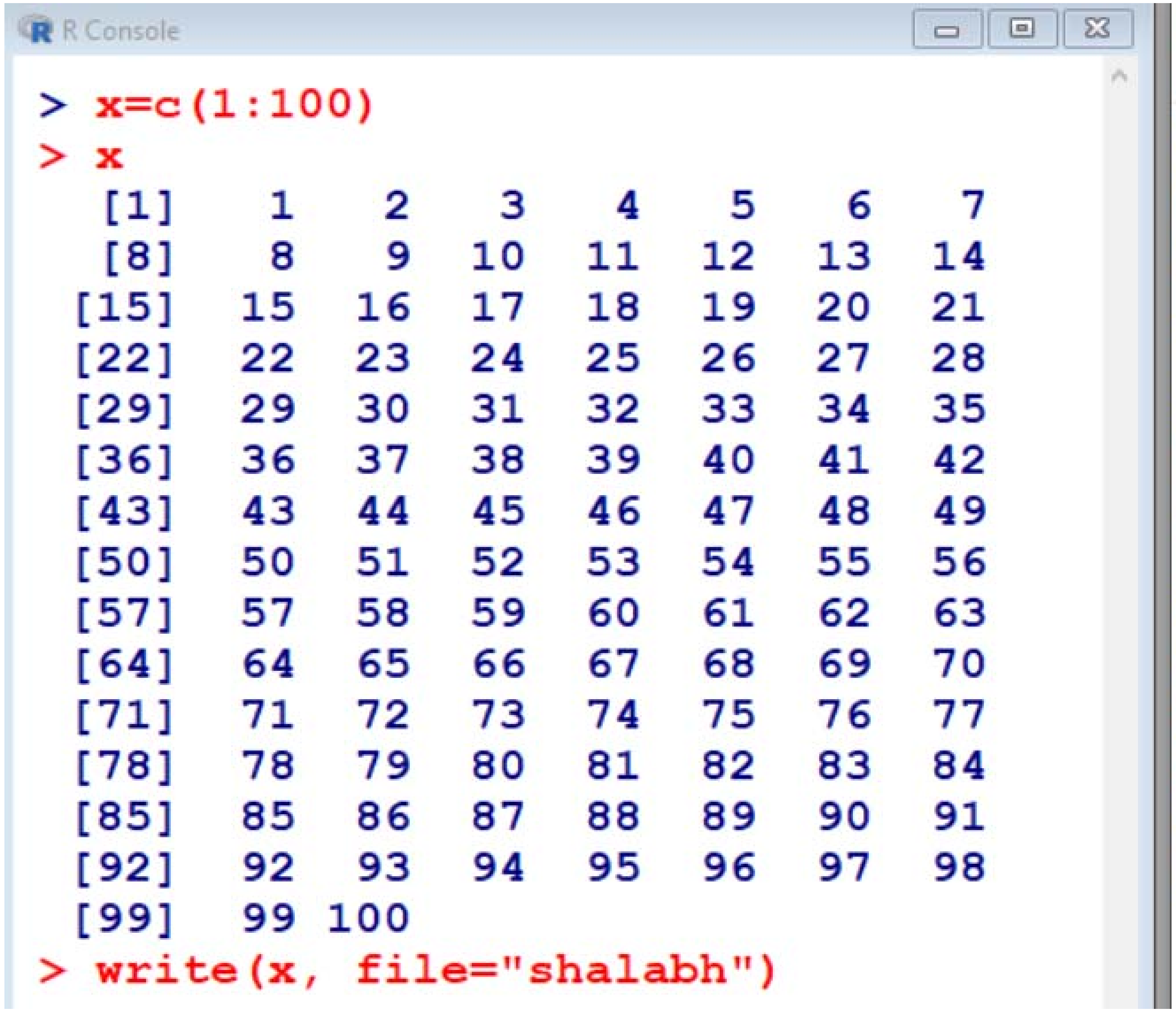
Arguments

`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.

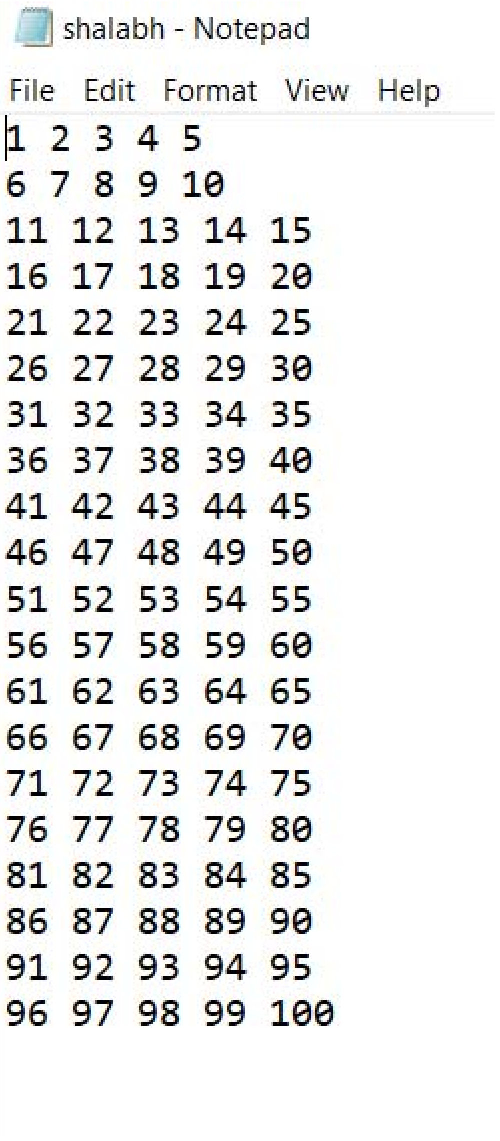
Saving and Writing Data Files

```
> 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")
```

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
```

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 `" "`.

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 =
"")
```

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 =
"")
```

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.**

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.