Foundations of R Software

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

Lecture - 02
Introduction Help Demonstration, and Examples in R

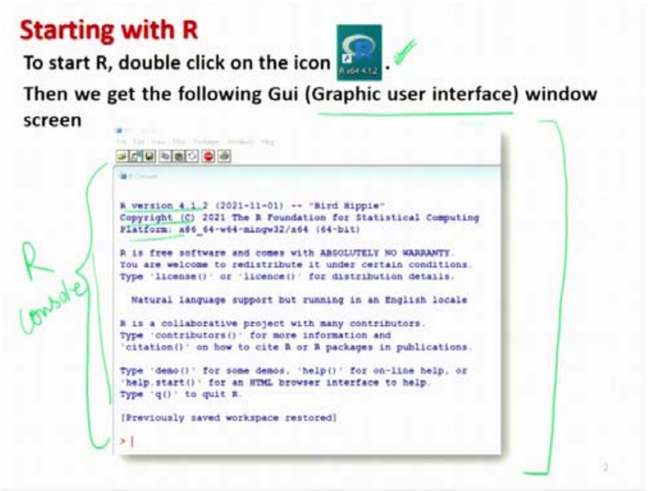
Hello friends, welcome to the course Foundations of R software and in this lecture we are going to explore that how we can take the help from the R software how we can get some demonstration and how we can get some examples so that we can understand how to execute a command and what are its interpretation. The reason why I chose this topic is as follows. Can you recall that whenever you try to study a new topic from any book what do you expect?

You expect that first some basic fundamentals should be given, then after that some examples, solved problems etc. are given through which you can explore that how we have to attempt the problems, how we have to solve them and then based on that you try to solve other problems. So, the same approach I am following in the R software also. Now at this moment, we do not know how to move forward in this process to learn the R software.

So, first of all we will try to see that how we can take the help on learning any particular topic on R and in R software, there are various ways some of the approaches are built in inside the software and some approaches are based on some external help, for example, if you go to any search engine and try to type like as tutorials in help in R or the name of the package command etc., possibly there are many forums, many websites which are offering such helps.

So, how to explore those things? So, that is what we are going to begin with in this lecture. So, let us begin our lecture and try to see what we can do ok. So, we are going to talk about here three topics - how to take help, how to get the demonstration of various commands and packages and how to look at the examples on various commands, various aspects that how to solve the problem, right.

(Refer Slide Time: 02:47)



Starting with R

To start R, double click on the icon .

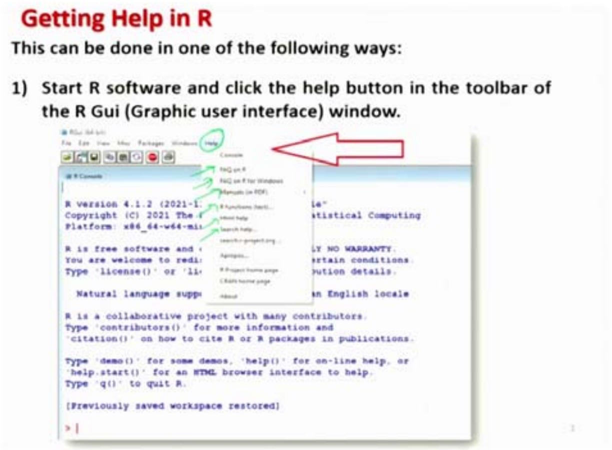
Then we get the following Gui (Graphic user interface) window screen

```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

So, now first very basic simple thing is this on your computer after you have installed the R software, you will see an icon like this one. So, you simply have to click on it. Once you click on it, then this will open a window like this one right and this window is called as GUI window that is Graphic User Interface window and this is the screen on which they have mentioned about this R, for example, what is the version of R where is the copy right what is the platform etc. etc. like this and this is the starting point from where we start working in R.

So, I will try to show you on the R console also and this whole thing, this is called as R console. So, these are some terminologies which are very basic terminologies and those people who are using R, they will be using these type of things. So, it is important that you try to understand what is the meaning of these small words and then you try to do it.

(Refer Slide Time: 04:10)



Getting Help in R

This can be done in one of the following ways:

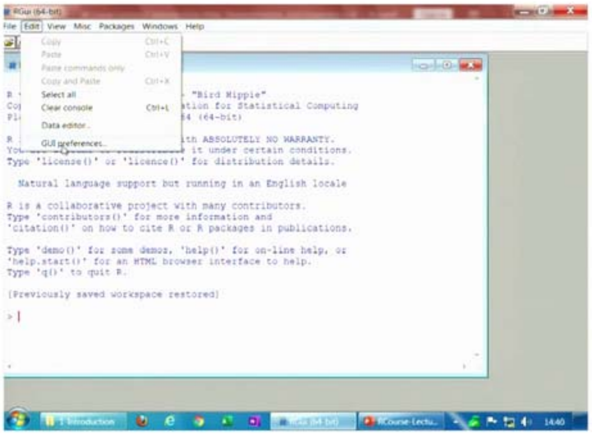
1) Start R software and click the help button in the toolbar of the R Gui (Graphic user interface) window.

```
R version 4.1.2 (2021-1:                 le"
Copyright (C) 2021 The                   atistical Computing
Platform: x86_64-w64-mi

R is free software and                   LY NO WARRANTY.
You are welcome to redi                  ertain conditions.
Type 'license()' or 'li                  bution details.

  Natural language supp                  an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

And after that, now you have to see, that how are you going to take the help. So, I will try to demonstrate here a couple of ways by which you can seek help in the R software. So, first option is this that you just go to this R GUI window and click here there is a option here Help and then you can see there are frequently asked question on R frequently asked question that is FAQ on R for windows, manuals R function etc. html search etc., they are given over there and you can click them.

And then you can explore that how they are trying to give the information on any particular aspect, right. So, before I move forward, let me try to give you this demonstration on the R console also.

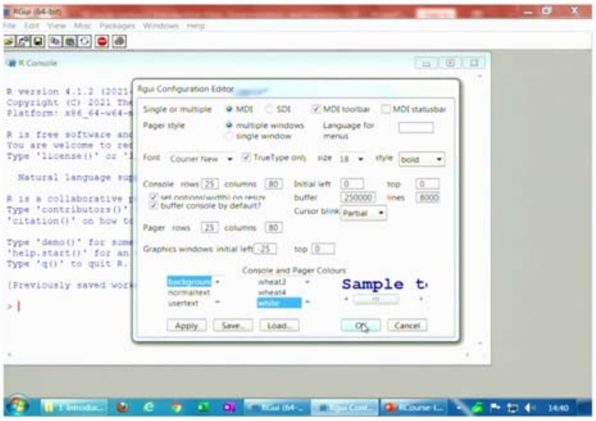
(Refer Slide Time: 05:16)



```
R                     "Bird Hippie"
Co                   ation for Statistical Computing
Pl                   64 (64-bit)

R                    ith ABSOLUTELY NO WARRANTY.
You                  it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

So, you can see here that after you click on the R icon, this will open a window like this. So, now, you can see that possibly in this case the fonts are quite light.

(Refer Slide Time: 05:34)



```
R version 4.1.2 (2021-
Copyright (C) 2021 The
Platform: x86_64-w64-m

R is free software and
You are welcome to red
Type 'license()' or 'l

  Natural language sup

R is a collaborative p
Type 'contributors()'
'citation()' on how to

Type 'demo()' for some
'help.start()' for an
Type 'q()' to quit R.

[Previously saved work

> |
```

Rgui Configuration Editor

Single or multiple ● MDI ○ SDI ☑ MDI toolbar □ MDI statusbar

Pager style ● multiple windows ○ single window Language for menus

Font Courier New ☑ TrueType only size 18 style bold

Console rows 25 columns 80 Initial left 0 top 0

☑ set options(width) on resize ☑ buffer console by default? buffer 250000 lines 8000 Cursor blink Partial

Pager rows 25 columns 80

Graphics windows: initial left -25 top 0

Console and Pager Colours

background wheat3

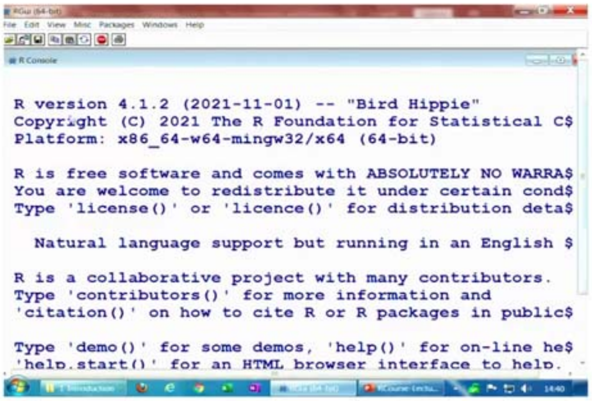
normaltext wheat4

usertext white

Sample t

So, first of all I can increase the font size so, that you can see here. So, if you go to edit then there is a option here GUI reference here I can make the font size to be, suppose here 18 and style to be here bold and you can see here different types of background etc. that you can maintain here you can control the size of the console and there are many options here. But I am not going into those details, but I will simply make it here like this, right.

(Refer Slide Time: 06:03)

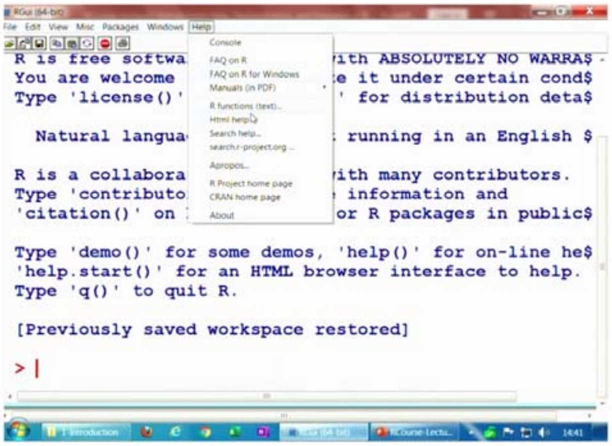


```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical C$
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRA$
You are welcome to redistribute it under certain cond$
Type 'license()' or 'licence()' for distribution deta$

  Natural language support but running in an English $

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in public$

Type 'demo()' for some demos, 'help()' for on-line he$
'help.start()' for an HTML browser interface to help.
```

So, you can see here because the font size is now more. So, you cannot see everything, but you can see here that this was like the R version 4.1.2 its name is “Bird Hippie” and then this is the copyright etc. etc. . So, whatever I shown you on the screenshot that you can see here also.

Now, you can see here, this is here are greater than sign and this is here are the vertical line. So, this greater than sign is the prompt sign and this here this vertical line here is the prompt. So, I will show you that how we are going to work.

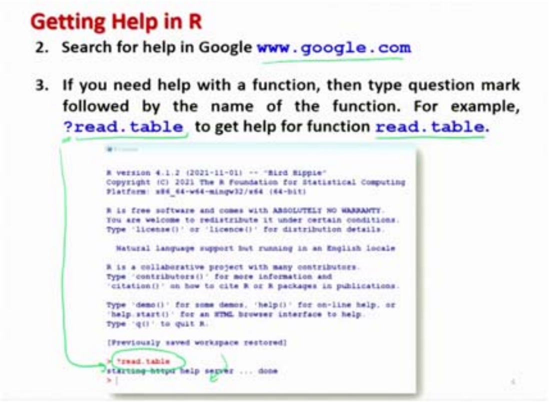
(Refer Slide Time: 06:44)



```
R is free softwa                    ith ABSOLUTELY NO WARRA$
You are welcome                    e it under certain cond$
Type 'license()'                   ' for distribution deta$

  Natural langua                   running in an English $

R is a collabora                   ith many contributors.
Type 'contributo                   information and
'citation()' on                    or R packages in public$

Type 'demo()' for some demos, 'help()' for on-line he$
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

But what I want to show you here is here Help you can see here in this Help you can see here all these things. So, if you try to explore try to spend some time on it you can see that how you can seek the help on this R software, right.

(Refer Slide Time: 07:04)



Getting Help in R

2. Search for help in Google www.google.com

3. If you need help with a function, then type question mark followed by the name of the function. For example, ?read.table to get help for function read.table.

```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> ?read.table
starting httpd help server ... done
> |
```

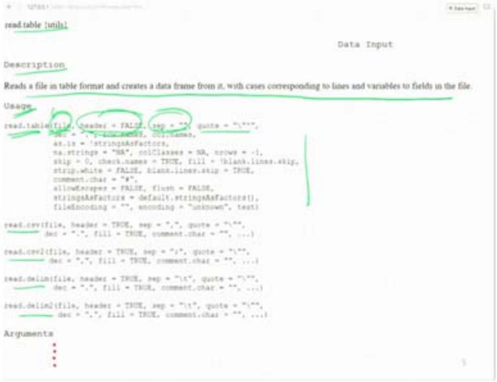
So, let me try to move forward so, that I can show you something more the second option nowadays which is very popular is that you try to take the help of any search engine like as Google just go to google dot com and try to search there for the command for the syntax or for anything related to the R software and then you will get it. But anyway I am not going into that mode, my objective is that how to seek the help from within the R software.

Now, when you want to take the help from R there are two options you have some idea about the command for which you need help or you do not know anything, you simply know ok I want to do something, what is the command I do not know and I want to know the command and then I want to have possibly more information on that. So, I will try to take up all these issues one by one.

So, first option I am going to show you here is that suppose you know that there is a command, say read dot table well we are going to learn about it, we are going to use about it this is the command to read the table in the data which is given in the tabular format. So, now, I want to know about this command. So, what I can do here I can just write a question mark and then I can write down here the read dot table command, right and this is the job which I have to do here at this prompt sign.

You can see here this is the same screen here I am trying to write out here question mark read dot table. And after that you have to simply enter and after that, I will show you what will happen this will go to the help server.

(Refer Slide Time: 09:01)



read.table {utils}

Data Input

Description

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", row.names, col.names,
           as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)

read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

Arguments

⋮

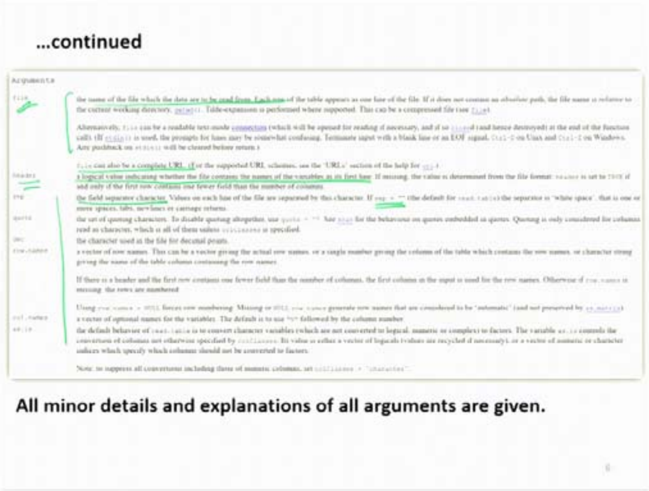
And it is like this because the help server from the R software and here you will see that it will give you all the information, complete details about this read dot table command and that is what I told you in the beginning itself that R is not a black box. Whatever information you want to know that is available. So, if you try to see here now I will try to first demonstrate here how are you going to seek the help.

You see this is very obvious that whenever you want to learn a topic on the book, the first condition is that you can find the book, but then you have to read it means without reading how can you understand the topic that how to solve the problem. So, the same rule applies here also.

So, I will show you here for example, this is the command here, then this is the description of this read dot table command that it reads a file in the table format and creates a data frame from it with cases corresponding to lines and variables to fields in the file and so on and after that, it is giving you how it is going to be used. So, you have to write down here read dot table and then inside the parenthesis like this one here, you have to write down here the file that is your file name, then header equal to FALSE or true then sep which is separator then code etc. etc., there is a long list and even after that it is also giving you the information on the read dot csv, read dot csv dot 2, etc. these are different types of files which can be read in the R software well, we are going to learn all the things gradually in the forthcoming lecture. So, you need not to be scared about these thing.

I am simply trying to show you here that these things are available and wherever you need it you do not need to go anywhere, but you can look into the R software and R software itself will give you all sorts of information and after that there will be many many arguments and every argument whatever is written here like a file, header, separator etc. etc. they are explained in detail that how you have to use it like this.

(Refer Slide Time: 11:23)



...continued

Arguments

- file: the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an absolute path, the file name is relative to the current working directory, getwd(). Tilde-expansion is performed where supported. This can be a compressed file (see file).

  Alternatively, file can be a readable text-mode connection (which will be opened for reading if necessary, and if so closed (and hence destroyed) at the end of the function call). (If stdin() is used, the prompts for lines may be somewhat confusing. Terminate input with a blank line or an EOF signal, Ctrl-D on Unix and Ctrl-Z on Windows. Any pushback on stdin() will be cleared before return.)

  file can also be a complete URL. (For the supported URL schemes, see the 'URLs' section of the help for url.)
- header: a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: header is set to TRUE if and only if the first row contains one fewer field than the number of columns.
- sep: the field separator character. Values on each line of the file are separated by this character. If sep = "" (the default for read.table) the separator is 'white space', that is one or more spaces, tabs, newlines or carriage returns.
- quote: the set of quoting characters. To disable quoting altogether, use quote = "". See scan for the behaviour on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless colClasses is specified.
- dec: the character used in the file for decimal points.
- row.names: a vector of row names. This can be a vector giving the actual row names, or a single number giving the column of the table which contains the row names, or character string giving the name of the table column containing the row names.

  If there is a header and the first row contains one fewer field than the number of columns, the first column in the input is used for the row names. Otherwise if row.names is missing, the rows are numbered.

  Using row.names = NULL forces row numbering. Missing or NULL row.names generate row names that are considered to be 'automatic' (and not preserved by as.matrix).
- col.names: a vector of optional names for the variables. The default is to use "V" followed by the column number.
- as.is: the default behavior of read.table is to convert character variables (which are not converted to logical, numeric or complex) to factors. The variable as.is controls the conversion of columns not otherwise specified by colClasses. Its value is either a vector of logicals (values are recycled if necessary), or a vector of numeric or character indices which specify which columns should not be converted to factors.

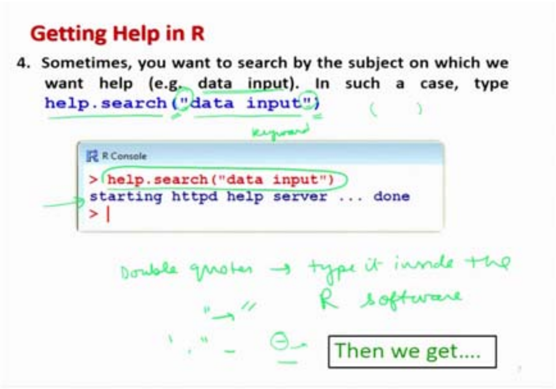
  Note: to suppress all conversions including those of numeric columns, set colClasses = "character".

All minor details and explanations of all arguments are given.

For example, you can see here file it is giving it like this is the name of the file blah blah blah blah blah. This is the header, I mean file can also have a complete URL and header is the logical value indicating whether the file contains the names of the variables as its first name etc. sep this is the field separator character and it is giving you that how it has to be given and how it has to be used and similarly, all these details are there well I have taken the screenshots because the font size is quite small here.

So, you may not be able to read it, but I request you that you try to repeat the same command on the R console and then try to see what it is trying to see. So, the bottom line is that if you are willing to read these details, you have the complete outcome that is all.

(Refer Slide Time: 12:21)



Getting Help in R

4. Sometimes, you want to search by the subject on which we want help (e.g. data input). In such a case, type `help.search("data input")`

keyword

R Console

```
> help.search("data input")
starting httpd help server ... done
> |
```

Double quotes → type it inside the R software

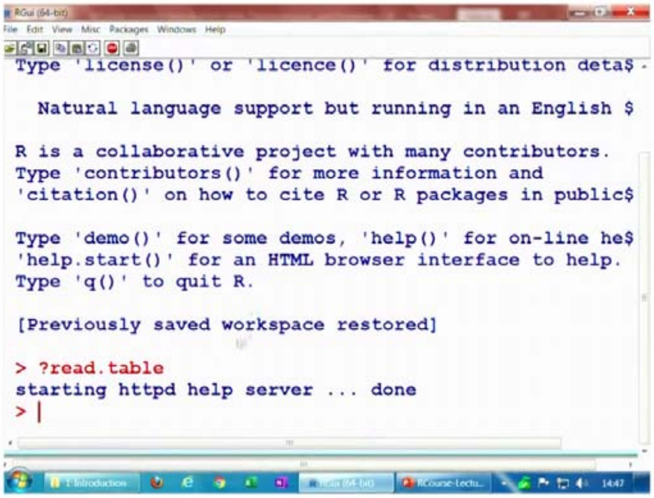
" _ "

' , " _ ⊖_

Then we get....

So, now I will try to show you it on the R console itself that how are you that how these things are going to look like this.

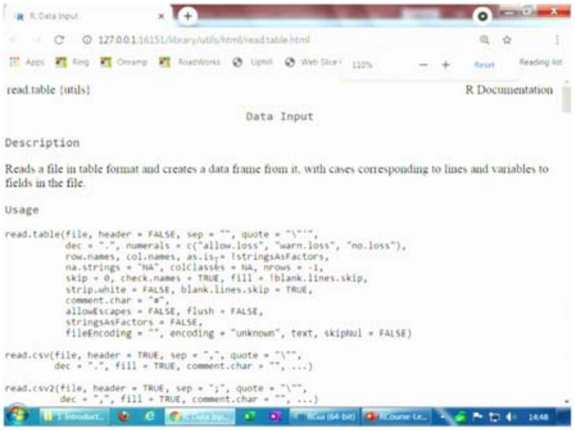
(Refer Slide Time: 12:42)



```
Type 'license()' or 'licence()' for distribution deta$

  Natural language support but running in an English $

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in public$

Type 'demo()' for some demos, 'help()' for on-line he$
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> ?read.table
starting httpd help server ... done
> |
```

So, what I can do here for example, this I come to the R console and here I try to type here question mark read dot table. So, you can see here this is the place where I have typed the command and let us try to see what happens. I enter here like this and you can see here, it is showing you here that is starting httpd, server etc. and this will open the some internet browser and you can see here this all these things are coming over here.

(Refer Slide Time: 13:05)



read.table {utils} R Documentation

Data Input

Description

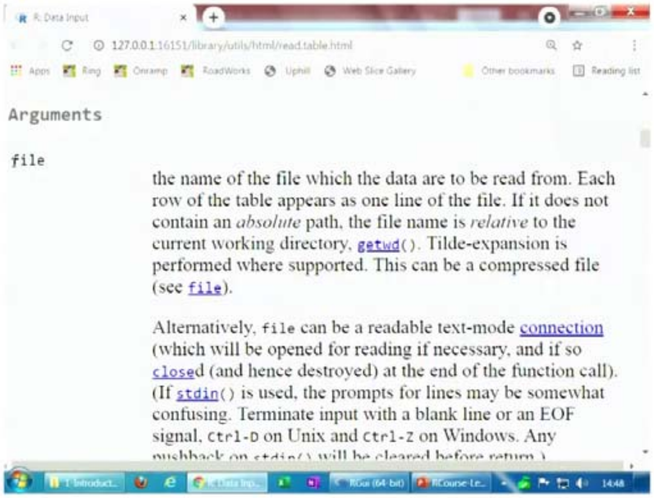
Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = FALSE,
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)
```

And if I try to increase the font size you can see very clearly here that this is giving you here the description reads a file and then this is the command here read dot table, then file, then header equal to FALSE etc. etc.

(Refer Slide Time: 13:29)



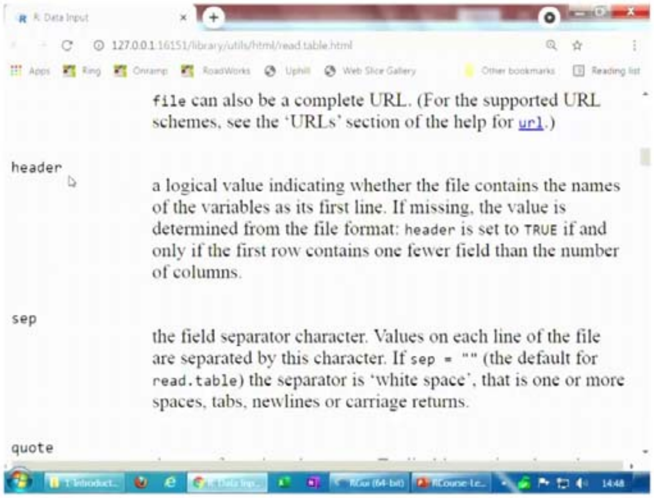
Arguments

`file`

the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an *absolute* path, the file name is *relative* to the current working directory, `getwd()`. Tilde-expansion is performed where supported. This can be a compressed file (see `file`).

Alternatively, `file` can be a readable text-mode connection (which will be opened for reading if necessary, and if so `close`d (and hence destroyed) at the end of the function call). (If `stdin()` is used, the prompts for lines may be somewhat confusing. Terminate input with a blank line or an EOF signal, `Ctrl-D` on Unix and `Ctrl-Z` on Windows. Any pushback on `stdin()` will be cleared before return.)

(Refer Slide Time: 13:33)



`file` can also be a complete URL. (For the supported URL schemes, see the 'URLs' section of the help for `url`.)

`header`

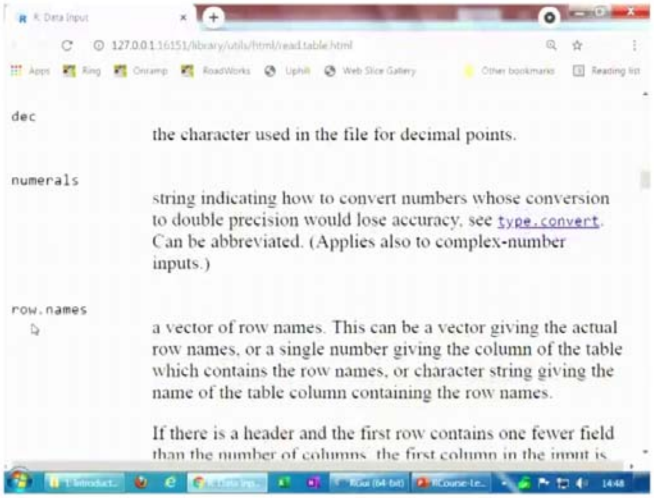
a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: `header` is set to `TRUE` if and only if the first row contains one fewer field than the number of columns.

`sep`

the field separator character. Values on each line of the file are separated by this character. If `sep = ""` (the default for `read.table`) the separator is 'white space', that is one or more spaces, tabs, newlines or carriage returns.

`quote`

(Refer Slide Time: 13:36)



`dec`

the character used in the file for decimal points.

`numerals`

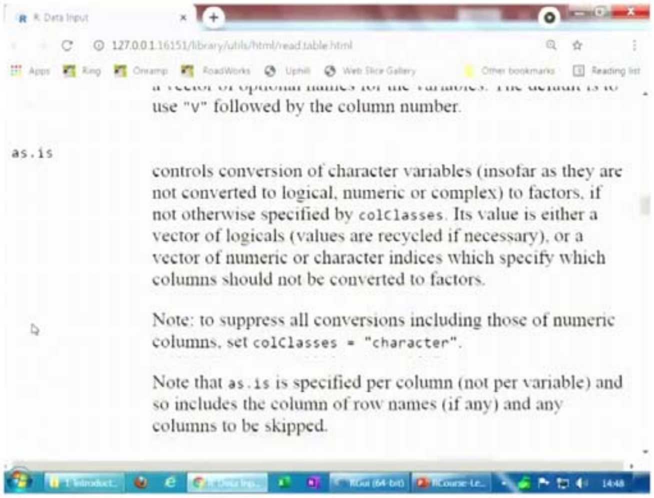
string indicating how to convert numbers whose conversion to double precision would lose accuracy, see `type.convert`. Can be abbreviated. (Applies also to complex-number inputs.)

`row.names`

a vector of row names. This can be a vector giving the actual row names, or a single number giving the column of the table which contains the row names, or character string giving the name of the table column containing the row names.

If there is a header and the first row contains one fewer field than the number of columns, the first column in the input is

(Refer Slide Time: 13:37)



a vector of optional names for the variables. The default is to use "V" followed by the column number.

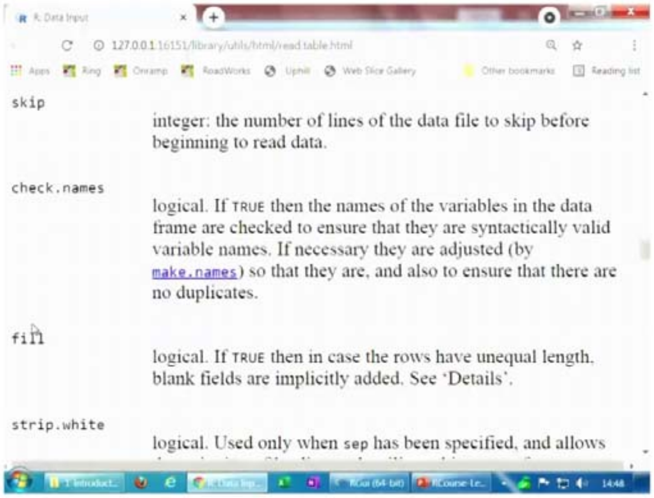
`as.is`

controls conversion of character variables (insofar as they are not converted to logical, numeric or complex) to factors, if not otherwise specified by `colClasses`. Its value is either a vector of logicals (values are recycled if necessary), or a vector of numeric or character indices which specify which columns should not be converted to factors.

Note: to suppress all conversions including those of numeric columns, set `colClasses = "character"`.

Note that `as.is` is specified per column (not per variable) and so includes the column of row names (if any) and any columns to be skipped.

(Refer Slide Time: 13:40)



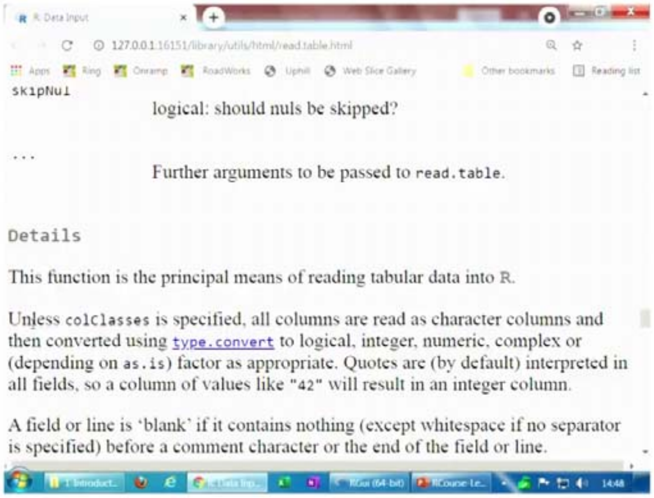
`skip`
integer: the number of lines of the data file to skip before beginning to read data.

`check.names`
logical. If `TRUE` then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names. If necessary they are adjusted (by `make.names`) so that they are, and also to ensure that there are no duplicates.

`fill`
logical. If `TRUE` then in case the rows have unequal length, blank fields are implicitly added. See 'Details'.

`strip.white`
logical. Used only when `sep` has been specified, and allows

(Refer Slide Time: 13:44)



`skipNul`
logical: should nuls be skipped?

`...`
Further arguments to be passed to `read.table`.

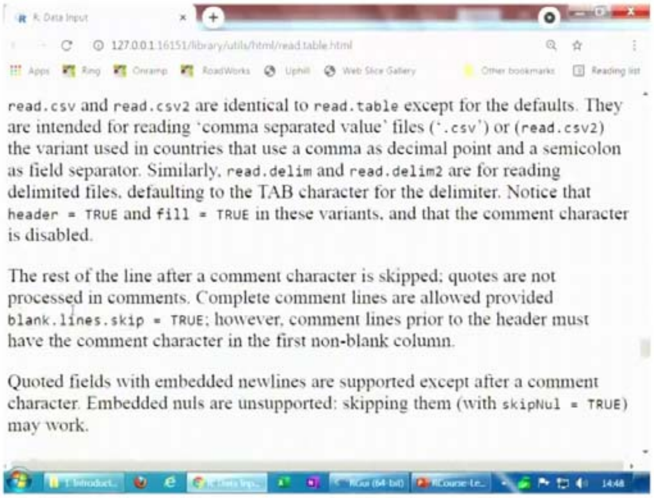
Details

This function is the principal means of reading tabular data into R.

Unless `colClasses` is specified, all columns are read as character columns and then converted using `type.convert` to logical, integer, numeric, complex or (depending on `as.is`) factor as appropriate. Quotes are (by default) interpreted in all fields, so a column of values like `"42"` will result in an integer column.

A field or line is 'blank' if it contains nothing (except whitespace if no separator is specified) before a comment character or the end of the field or line.

(Refer Slide Time: 13:50)



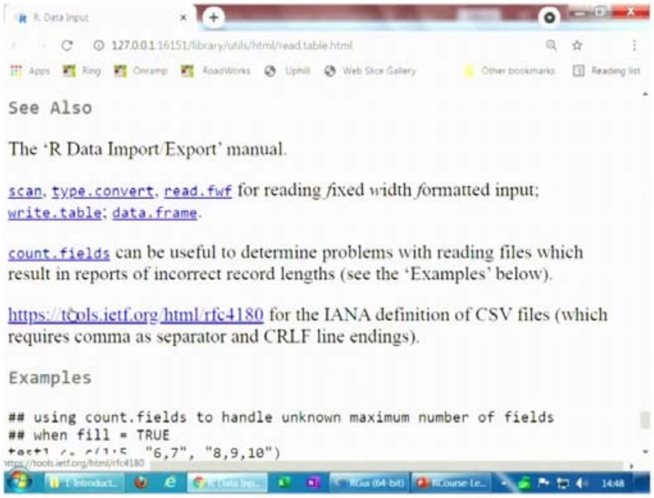
read.csv and read.csv2 are identical to read.table except for the defaults. They are intended for reading 'comma separated value' files ('.csv') or (read.csv2) the variant used in countries that use a comma as decimal point and a semicolon as field separator. Similarly, read.delim and read.delim2 are for reading delimited files, defaulting to the TAB character for the delimiter. Notice that header = TRUE and fill = TRUE in these variants, and that the comment character is disabled.

The rest of the line after a comment character is skipped; quotes are not processed in comments. Complete comment lines are allowed provided blank.lines.skip = TRUE; however, comment lines prior to the header must have the comment character in the first non-blank column.

Quoted fields with embedded newlines are supported except after a comment character. Embedded nuls are unsupported: skipping them (with skipNul = TRUE) may work.

And then it is giving you here all the details about the file, header, separator, sep etc. and you can see here it has got all the details, right and then beside this thing it is giving you that how these things have been obtained this is giving you the complete details of the differences also that how these things are interrelated to different commands.

(Refer Slide Time: 14:02)



See Also

The 'R Data Import/Export' manual.

scan, type.convert, read.fwf for reading *f*ixed *w*idth *f*ormatted input; write.table; data.frame.

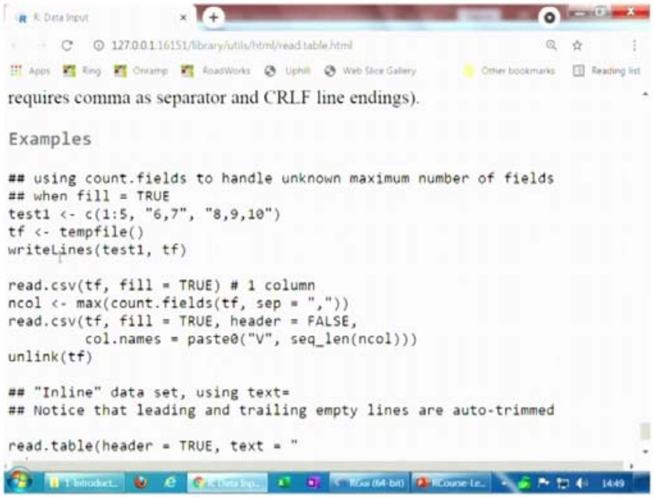
count.fields can be useful to determine problems with reading files which result in reports of incorrect record lengths (see the 'Examples' below).

https://tools.ietf.org/html/rfc4180 for the IANA definition of CSV files (which requires comma as separator and CRLF line endings).

Examples

```
## using count.fields to handle unknown maximum number of fields
## when fill = TRUE
test1 <- c(1:5, "6,7", "8,9,10")
```

(Refer Slide Time: 14:05)

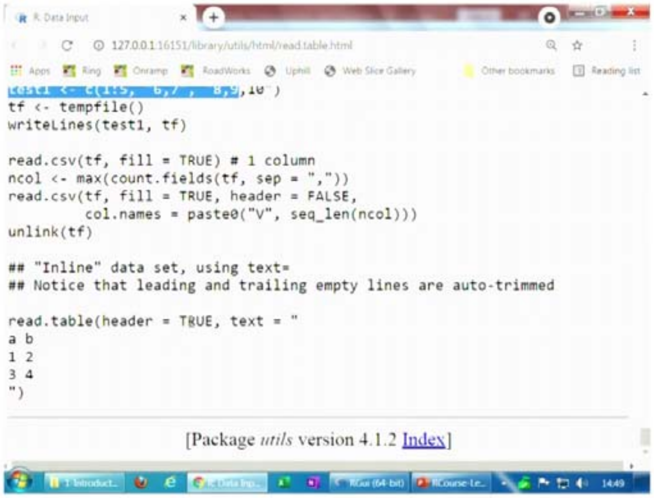


requires comma as separator and CRLF line endings).

Examples

```
## using count.fields to handle unknown maximum number of fields
## when fill = TRUE
test1 <- c(1:5, "6,7", "8,9,10")
tf <- tempfile()
writeLines(test1, tf)

read.csv(tf, fill = TRUE) # 1 column
ncol <- max(count.fields(tf, sep = ","))
read.csv(tf, fill = TRUE, header = FALSE,
         col.names = paste0("V", seq_len(ncol)))
unlink(tf)

## "Inline" data set, using text=
## Notice that leading and trailing empty lines are auto-trimmed

read.table(header = TRUE, text = "
```

(Refer Slide Time: 14:10)



```
test1 <- c(1:5, "6,7", "8,9,10")
tf <- tempfile()
writeLines(test1, tf)

read.csv(tf, fill = TRUE) # 1 column
ncol <- max(count.fields(tf, sep = ","))
read.csv(tf, fill = TRUE, header = FALSE,
         col.names = paste0("V", seq_len(ncol)))
unlink(tf)

## "Inline" data set, using text=
## Notice that leading and trailing empty lines are auto-trimmed

read.table(header = TRUE, text = "
a b
1 2
3 4
")
```

[Package *utils* version 4.1.2 Index]

How they from where they have been taken and yeah, what else do you want it is giving you a different examples also that how you can do it etc. like. So, now, if you ask me R itself is giving you the complete information and once you read it, you will know the complete information about this command read dot table. Surely when you are trying to work it may be possible that all the options which are given here they may not be useful for you.

Usually, the way it is written like that only first couple of these option they are more useful, but definitely as a programmer when you are trying to do something and if you want to get done something you would like to have different types of option which you can use at your convenience rather than the convenience of the software and that is what I said in the beginning that you will have the complete flexibility that whatever type of data you have you can use it here and read dot table is going to give you the all possible option.

So, now it depends on you how much you want to study to understand this command and I have taken here only one example, but that you can do for all possible commands which are available in the R software right ok. So, now, I come to another approach. Suppose there is a situation where I don't know what is the command, but my objective is this I want to input some data. Now what is the command I have no idea. So, what I can do that I can use here the command help dot search.

help dot search and within the parenthesis; parenthesis means these brackets within this brackets within the double quotes this is the double quote sign which is available inside the on your keyboard you have to write data input it is just like using a keyword that you know means you expect that if I try to search for data input in R on some web search engine possibly I may get some help.

So, similarly you have to think about an appropriate word and then you have to type it here and you have to type it on the R console here at this command line. So, you can see here I have typed here after that it will start in httpd help server etc. and it will try to give you something. So, we try to see here what do we get, but before that I would like to give you here a warning.

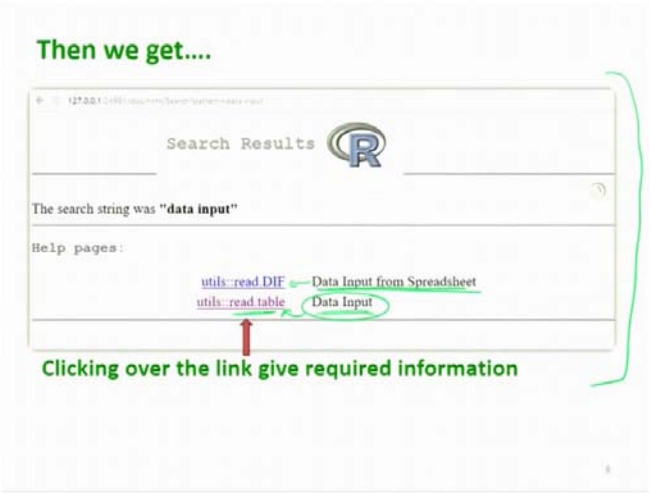
The warning is that when you are trying to use here this double quotes, this is the sign of double quotes, try to type it inside the R software ,very important thing because in case if you try to type in some editor, for example, if you try to type this in MS Word you can see here as soon as you type it and then you move forward, this symbol will become something like this and then R don't understand it and then many times we are trying to copy and paste the commands and things do not work and this creates lots of hindrances and time and basically gets waste.

So, my suggestion to you all is that whenever you are trying to use this single quote or double quote etc., try to type this command inside the R software and in case if you are doing copying and pasting and if you are copying and pasting correctly if it is creating any problem means, I would suggest you try to type the symbols with your own hand for example, symbols like single quote, double quote minus sign etc., try to type them inside the R console.

Many times when you are trying to type the minus sign inside the MS Word software as soon as you move forward, the character of this becomes change and it becomes simply little bit larger, that looks beautiful that looks just like a minus sign, but possibly it is not really the minus sign which is used as a mathematical operator.

So, that is my very simple advice and this is the way we are going to learn this R software that I will try to share all my experiences and these are smaller things and finally, after some time you will become expert in the R software.

(Refer Slide Time: 18:46)



Then we get....

Search Results

The search string was **"data input"**

Help pages:

utils::read.DIF Data Input from Spreadsheet
utils::read.table Data Input

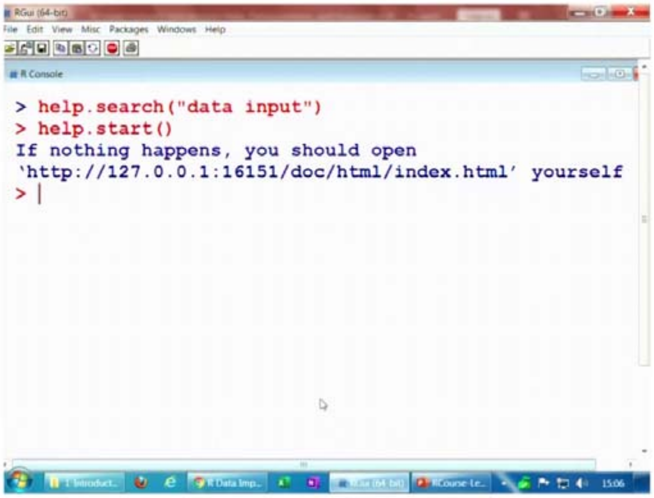
Clicking over the link give required information

So, once you use this command help dot search and inside the parentheses within double quotes if you try to type data input you get here this type of web page. And here you can see here it is showing you here something like utilities actually read dot table and this is for data input and it is giving you another option that data input from spread sheet etc.

So, you can see here now that when you have given the option data input it is giving you two options here that you can look into the read table, you can look into the read dot dif and possibly once you start looking into these things after that as soon as you go into more deeper, you will learn more things and finally, you will converge to a point where the 100 percent correct accurate help is available, right.

So, now the same thing if you try to do it in the R console also I can show you here. So, I try to take this command and then yeah if you try to see here I can come here and yeah I will show you later on, but if you try to press here control L means control plus L this will clear the screen also.

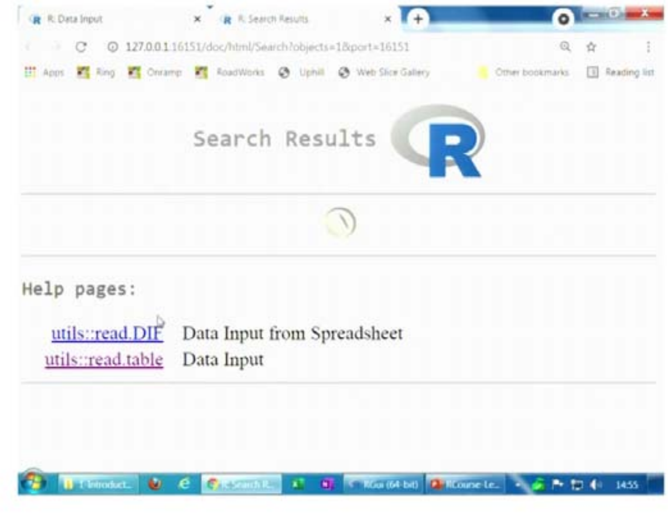
(Refer Slide Time: 20:10)



```
> help.search("data input")
> help.start()
If nothing happens, you should open
'http://127.0.0.1:16151/doc/html/index.html' yourself
> |
```

And if you try to type the same command that help dot search and within course data input you will like here help dot search and within the parentheses within the double quotes data input.

(Refer Slide Time: 20:33)

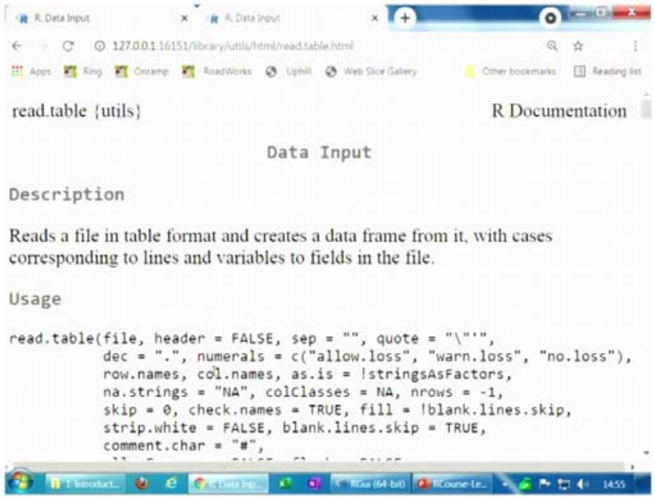


Search Results

Help pages:

utils::read.DIF Data Input from Spreadsheet
utils::read.table Data Input

(Refer Slide Time: 20:43)



read.table {utils} R Documentation

Data Input

Description

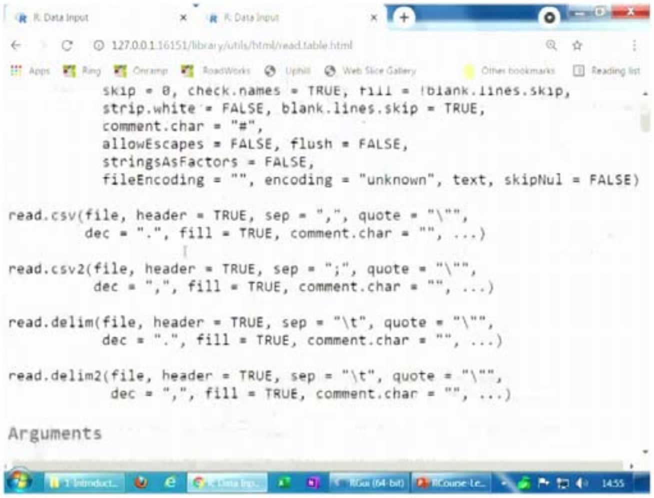
Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
```

So, you can see here as soon as you enter, this opens here a new web page like this one and if you try to go here read dot if you try to click here. So, you will see here we get here the same file what we had got earlier.

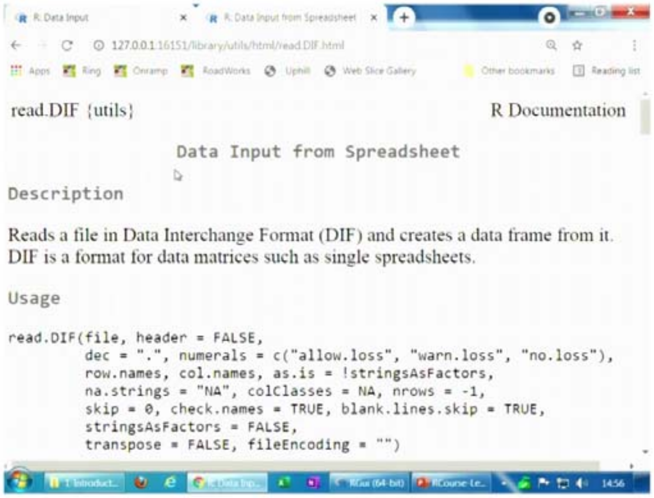
(Refer Slide Time: 20:47)



```
          skip = 0, check.names = TRUE, fill = !blank.lines.skip,
          strip.white = FALSE, blank.lines.skip = TRUE,
          comment.char = "#",
          allowEscapes = FALSE, flush = FALSE,
          stringsAsFactors = FALSE,
          fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)

read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

Arguments

(Refer Slide Time: 20:52)



read.DIF {utils} R Documentation

Data Input from Spreadsheet

Description

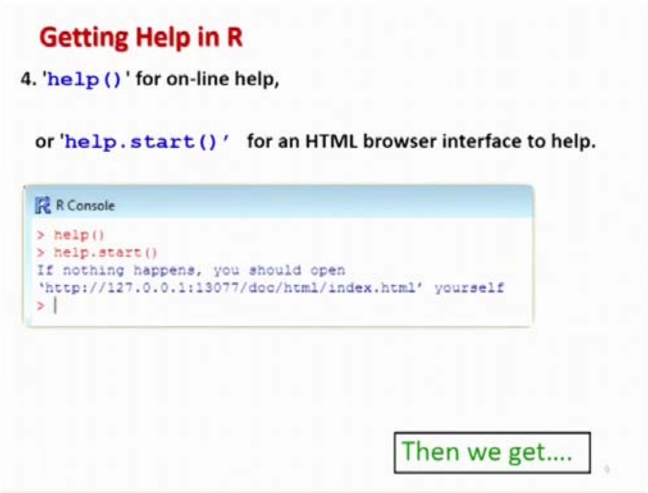
Reads a file in Data Interchange Format (DIF) and creates a data frame from it. DIF is a format for data matrices such as single spreadsheets.

Usage

```
read.DIF(file, header = FALSE,
         dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
         row.names, col.names, as.is = !stringsAsFactors,
         na.strings = "NA", colClasses = NA, nrows = -1,
         skip = 0, check.names = TRUE, blank.lines.skip = TRUE,
         stringsAsFactors = FALSE,
         transpose = FALSE, fileEncoding = "")
```

And if we give you the idea about this read dot table and if you try to click here read dot DIF it is giving you an idea that how you can input the from any spreadsheet. So, you can see here it is not so, difficult to get or to seek help in the R software the only thing is this you should have a wish and you should have a will to learn the course.

(Refer Slide Time: 21:14)



Getting Help in R

4. '`help()`' for on-line help,

or '`help.start()`' for an HTML browser interface to help.

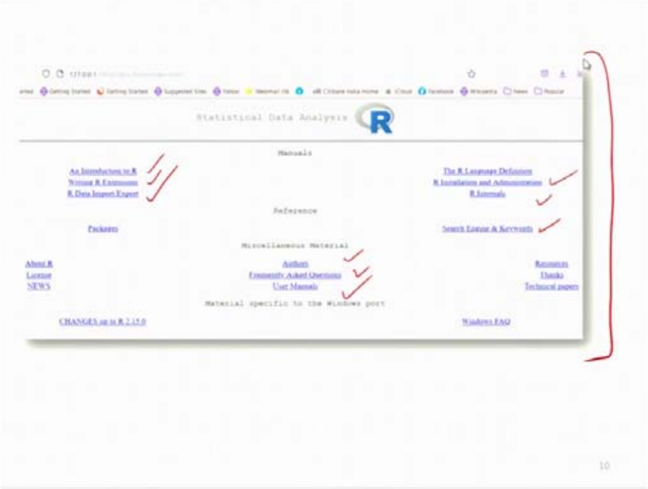
R Console

```
> help()
> help.start()
If nothing happens, you should open
'http://127.0.0.1:13077/doc/html/index.html' yourself
> |
```

Then we get....

And another option is this if you do not know anything if you want to start from the scratch, you simply can type here help or say and this parenthesis or you can dot you can type help dot start and this parenthesis only that is all, right.

(Refer Slide Time: 21:39)



Statistical Data Analysis R

Manuals

An Introduction to R
Writing R Extensions
R Data Import/Export

The R Language Definition
R Installation and Administration
R Internals

Reference

Packages

Search Engine & Keywords

Miscellaneous Material

About R
License
NEWS

Authors
Frequently Asked Questions
User Manuals

Resources
Thanks
Technical papers

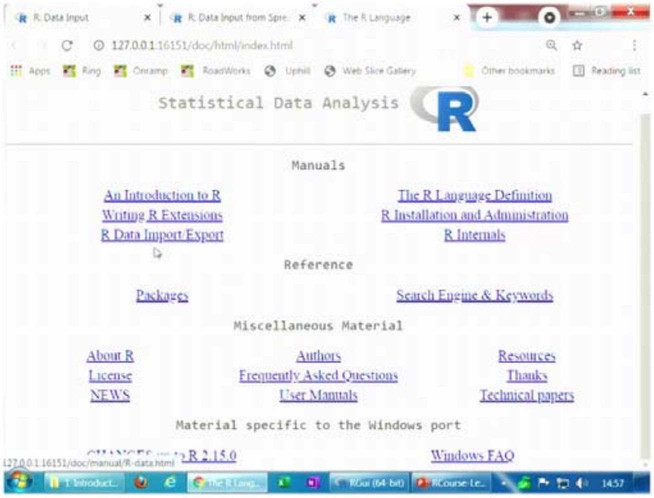
Material specific to the Windows port

CHANGES up to R 2.15.0

Windows FAQ

So, in case if you use any of this command means then this website is going to be open, I have given you here a screenshot where you can see many many things are there, now it is up to you and it is your capability that how efficient you are in finding out the correct help on the right command, right. So, anyway I try to show you this thing also on the R console here.

(Refer Slide Time: 22:09)



Statistical Data Analysis R

Manuals

An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals

Reference

Packages
Search Engine & Keywords

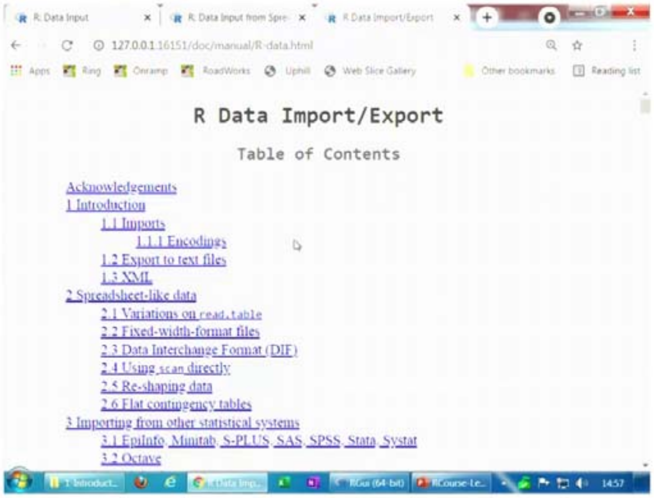
Miscellaneous Material

About R
License
NEWS
Authors
Frequently Asked Questions
User Manuals
Resources
Thanks
Technical papers

Material specific to the Windows port

CHANGES up to R 2.15.0
Windows FAQ

(Refer Slide Time: 22:20)

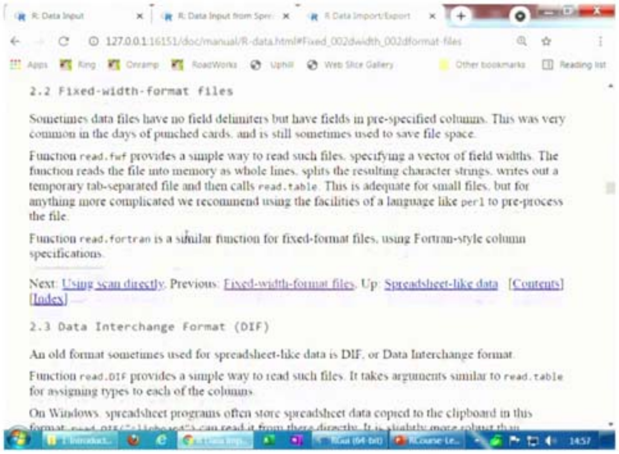


R Data Import/Export

Table of Contents

Acknowledgements
1 Introduction
    1.1 Imports
        1.1.1 Encodings
    1.2 Export to text files
    1.3 XML
2 Spreadsheet-like data
    2.1 Variations on read.table
    2.2 Fixed-width-format files
    2.3 Data Interchange Format (DIF)
    2.4 Using scan directly
    2.5 Re-shaping data
    2.6 Flat contingency tables
3 Importing from other statistical systems
    3.1 EpiInfo, Minitab, S-PLUS, SAS, SPSS, Stata, Systat
    3.2 Octave

(Refer Slide Time: 22:26)



2.2 Fixed-width-format files

Sometimes data files have no field delimiters but have fields in pre-specified columns. This was very common in the days of punched cards, and is still sometimes used to save file space.

Function `read.fwf` provides a simple way to read such files, specifying a vector of field widths. The function reads the file into memory as whole lines, splits the resulting character strings, writes out a temporary tab-separated file and then calls `read.table`. This is adequate for small files, but for anything more complicated we recommend using the facilities of a language like `perl` to pre-process the file.

Function `read.fortran` is a similar function for fixed-format files, using Fortran-style column specifications.

Next: Using scan directly, Previous: Fixed-width-format files, Up: Spreadsheet-like data [Contents] [Index]

2.3 Data Interchange Format (DIF)

An old format sometimes used for spreadsheet-like data is DIF, or Data Interchange format.

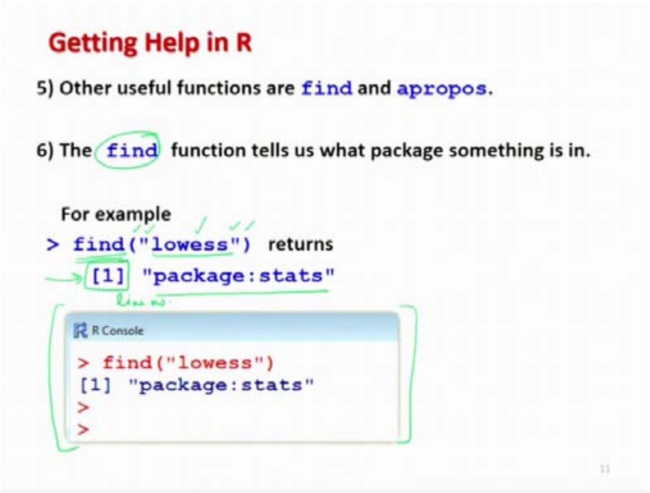
Function `read.DIF` provides a simple way to read such files. It takes arguments similar to `read.table` for assigning types to each of the columns.

On Windows, spreadsheet programs often store spreadsheet data copied to the clipboard in this format: `read.DIF("clipboard")` can read it from there directly. It is slightly more robust than

So, you can see here I can see here help dot start and then just open and close parenthesis and if I try to show you here I can reduce the font size. So, you can see here right now all the things are there what you want to know means this is about the data import and export in the R, it is giving you all sorts of format fixed with format how you can do it.

So, it is just like opening a book. So, once you have got a good book then you can just browse through the topic and then, but the only condition is that you have to read it. If you do not read, then nobody will come to help you out.

(Refer Slide Time: 22:44)



Getting Help in R

5) Other useful functions are `find` and `apropos`.

6) The `find` function tells us what package something is in.

For example

`> find("lowess")` returns

`[1] "package:stats"`

R Console

```
> find("lowess")
[1] "package:stats"
>
>
```

So, you can see here now after this I come to another approach. You see whenever we want to find or search something from in a text file or in a data file, we use the function for example, like control F or say find. Similarly, in the R software also we are using various types of functions. These functions are actually a special type of programs and which execute some special job and these functions are available inside the package.

And suppose we are in a situation that we have a function and we want to know which of the package contains this function. The reason is this although I will explain you later on in the forthcoming lecture that unless and until you install the package or load the package first you cannot use this function you cannot execute it.

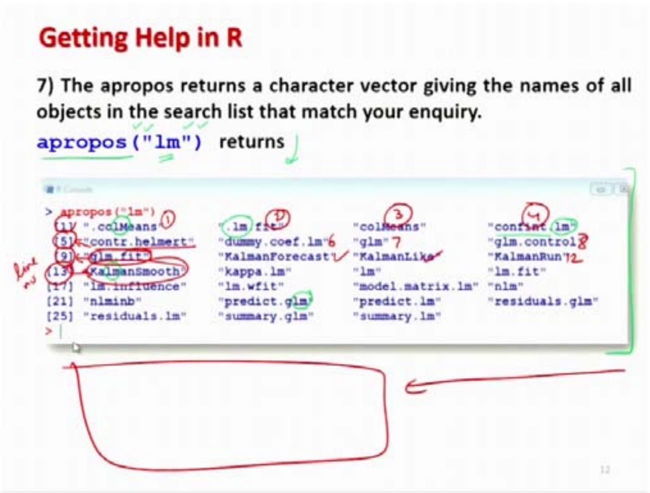
So, this is the first step while doing any analysis before using the function that one should know that which package is having it. So, suppose I know that there is some function and I want to find out where is this function in which package. Then for example, in this case suppose I want to know that there is a function lowess and we want to know that which statistical package or which R package has got this function for that we have a command here find.

So, the syntax is you simply try to write down here find, within the parenthesis, within the double quotes you simply write down this function name lowess and it will give you this type of outcome right and in case if you try to see it in the R console also it will give you this outcome.

So, if you try to read it this is saying package stats; that means, this function is available inside a package which is actually R package whose name is stats so; that means, you need to first upload or have the package stats in your computer and then you can do anything further. Here if you try to see here it is written like inside the parenthesis this is number 1. Actually this is only indicating the line number. So, this is first line and well this is at the first position.

So, I will try to explain you this concept when we have more than one values then we will try to show you that what this one is really indicating, but here I just thought that ok because it is there, so it should not create any problem. So, I should inform you, ok.

(Refer Slide Time: 25:51)



Getting Help in R

7) The apropos returns a character vector giving the names of all objects in the search list that match your enquiry.

`apropos("lm")` returns

```
> apropos("lm")
 [1] ".colMeans"        ".lm.fit"          "colMeans"           "confint.lm"
 [5] "contr.helmert"    "dummy.coef.lm"    "glm"                "glm.control"
 [9] "glm.fit"          "KalmanForecast"   "KalmanLike"         "KalmanRun"
[13] "KalmanSmooth"     "kappa.lm"         "lm"                 "lm.fit"
[17] "lm.influence"     "lm.wfit"          "model.matrix.lm"    "nlm"
[21] "nlminb"           "predict.glm"      "predict.lm"         "residuals.glm"
[25] "residuals.lm"     "summary.glm"      "summary.lm"
> |
```

Now, the second option suppose I want to use some function or command, but I cannot recall the exact command, but I know only some part, some fractional part of that command ok it looked like this. So, now how to find it out? One option is that if I can find out all the possible functions which contain that name then using my memory possibly I can find out which of the function I am trying to find.

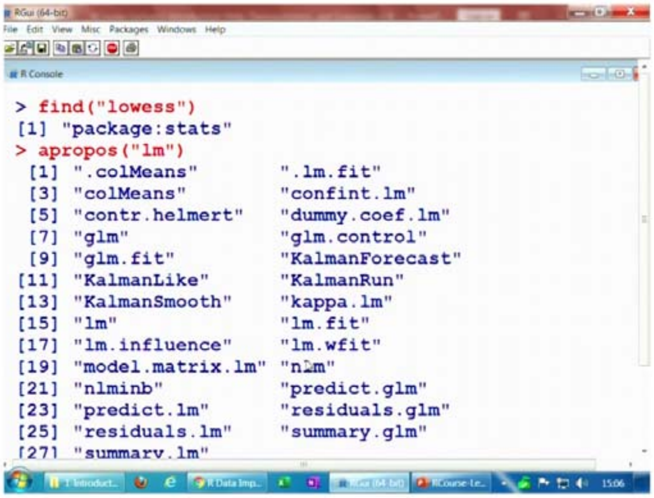
So, for that we have a command here apropos a p r o p o s and suppose I want to know that ok I can recall that ok sometime back I had used the command in which there were two words l m. So, now I will write down here apropos, within the parenthesis within the double quotes I will write l m and then if I try to enter here this will return me this type of screenshot.

So, you can see here it is now giving me all available functions on my computer within my R software which contain l m for example, you can see here this is here l m, but this is the part of the command dot call means, there is another value here where it is here dot l m, but this is the part of dot l m dot fit.

Similarly, if you try to see here there is another here l m, but this is the part of confint dot l m. Similarly if you try to take here anyone, so, for example, you can see here this is here l m, but this is the part of KalmanRun that is another function, right. Similarly, if you try to see here there is here function here predict dot glm in which the l m also occurs.

So, you can see here it is giving you all possible functions which are available on your computer and now it is your turn to look into these commands and try to recall if you can recall the earlier command which you want to use, right. So, let me try to show you these two things on the R console and then I will try to move forward.

(Refer Slide Time: 28:29)



```
> find("lowess")
[1] "package:stats"
> apropos("lm")
 [1] ".colMeans"          ".lm.fit"
 [3] "colMeans"           "confint.lm"
 [5] "contr.helmert"      "dummy.coef.lm"
 [7] "glm"                "glm.control"
 [9] "glm.fit"            "KalmanForecast"
[11] "KalmanLike"         "KalmanRun"
[13] "KalmanSmooth"       "kappa.lm"
[15] "lm"                 "lm.fit"
[17] "lm.influence"       "lm.wfit"
[19] "model.matrix.lm"    "nlm"
[21] "nlminb"             "predict.glm"
[23] "predict.lm"         "residuals.glm"
[25] "residuals.lm"       "summary.glm"
[27] "summary.lm"
```

So, if you try to see here, I try to clear the screen by pressing control L and then I try to say here find l o w e s s and if I say enter it is giving me here, this is package it is stats and this is the same thing what you have got here. So, now, you can gradually believe on me that whenever I am trying to show you the screenshots, this is exactly going to happen when you are trying to execute it on the R console.

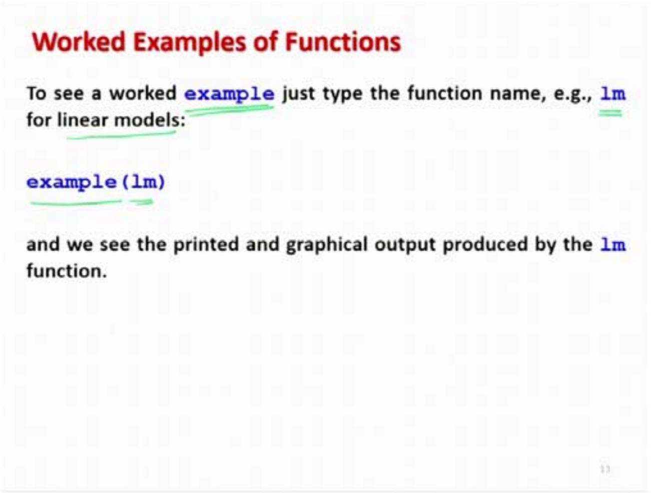
Now, you also understand what is the meaning of R console. Now similarly if you try to come here apropos, say here l m. You can see here it is giving you these many things you can see here these are here like this and here I want to show you here one thing more the reason I took it here if you try to see here, here we have got these numbers 1, then 5 and then 9.

What does this mean? 1 as I said earlier this is the line number. So, if you say this is here line number 1 and this is here 1, 2, 3, 4 and the 1st value on the second line is the 5th value which is written here 5, then here 6th, 7th, 8th and then here this is the value in the 3rd line which is the 9th value in the list similarly 10th, 11th, 12th and then this is going to be here the 13th value which is KalmanRun.

And now that is very obvious that in case if you try to shrink this window if you try to make it here only of this size then this number is going to increase and that is what exactly is happening here means if you try to see when you are trying to do here in the R console because you have increased the font size so, the number of elements that can be accommodated in a line they are they are becoming only two. So, this is there. So, it is. So, there are here 27 lines but in my slides there are only 25 lines.

So, and then it will also depend that what type of package you have on your particular computer. So, it may be possible that when you are trying to repeat the same commands on your computer this may not exactly match, but this should not confuse you I have explained you the reason why this is going to happen, right ok.

(Refer Slide Time: 30:58)



Worked Examples of Functions

To see a worked `example` just type the function name, e.g., `lm` for linear models:

`example(lm)`

and we see the printed and graphical output produced by the `lm` function.

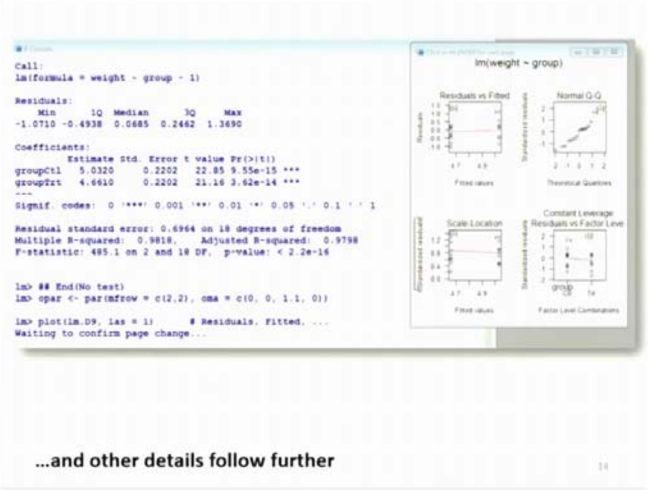
13

So, now I have given you sufficient idea that how you can seek the help. Now, I come to another aspect. Now you have got the book, you have you have searched the content that ok these things are available in the book and you start reading the book. Once you read a concept after that the first thing which you need is some solved examples. Solved example help us that how we have to proceed in the correct way.

So, similarly in the R software also there is a provision of examples and these examples on particular function command, etc. that can be viewed using the command example e x a m p l e. For example, in case if I want to know or we want to understand an example on the command l m.

Actually l m is a command to use the linear models which are quite popular in statistics although we are not going to do it here, but this is just for your information that suppose somebody wants to know that how to execute this l m command and how to interpret the outcome then simply type example inside the parenthesis just l m. And after that you will get all the details all the details about the outcome which are in the form of text and then graphics etc. etc.

(Refer Slide Time: 32:26)



```
Call:
lm(formula = weight ~ group - 1)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0710 -0.4938  0.0685  0.2462  1.3690

Coefficients:
          Estimate Std. Error t value Pr(>|t|)
groupCtl    5.0320     0.2202   22.85 9.55e-15 ***
groupTrt    4.6610     0.2202   21.16 3.62e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6964 on 18 degrees of freedom
Multiple R-squared:  0.9818,    Adjusted R-squared:  0.9798
F-statistic: 485.1 on 2 and 18 DF,  p-value: < 2.2e-16


lm> ## End(No test)
lm> opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))

lm> plot(lm.D9, las = 1)      # Residuals, Fitted, ...
Waiting to confirm page change...
```

...and other details follow further

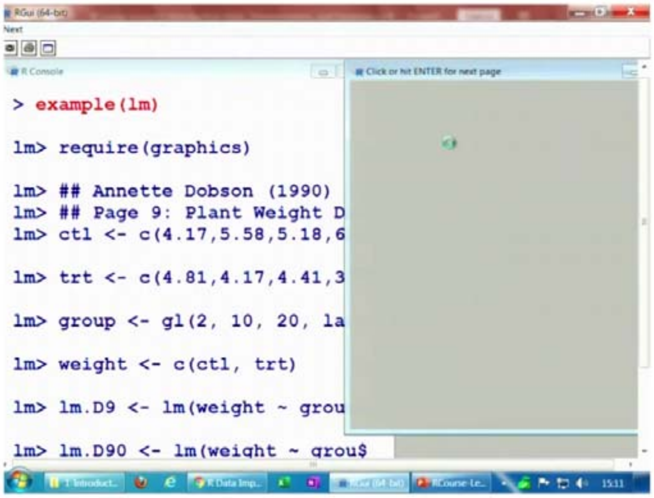
14

The only thing is this you have to just read it for example, you can see here and I try to do it on the R console, I get here this type of screenshot called then l m formula residual etc. etc., many many things and then graphics also you will get and there are more details although I am not going to print here all the screenshots but I will try to show you on the R console itself so, that you can see that what is really happenings right.

So, you can see here let me and there here you can also see that I can control the width of my this screen just by pressing or by just by shifting these things. So, I am trying to do it because you will see later on that when you want to do the graphical thing there are some graphic things also then they appear on the same screen.

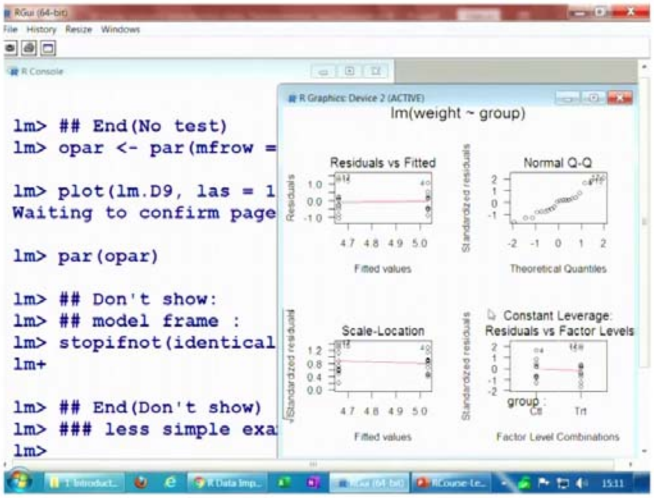
So, if you in a different window. So, you want to create some space. So, first let me try to clear the screen by pressing control L.

(Refer Slide Time: 33:29)



```
> example(lm)

lm> require(graphics)

lm> ## Annette Dobson (1990)
lm> ## Page 9: Plant Weight D
lm> ctl <- c(4.17,5.58,5.18,6

lm> trt <- c(4.81,4.17,4.41,3

lm> group <- gl(2, 10, 20, la

lm> weight <- c(ctl, trt)

lm> lm.D9 <- lm(weight ~ grou

lm> lm.D90 <- lm(weight ~ grou$
```

(Refer Slide Time: 33:38)

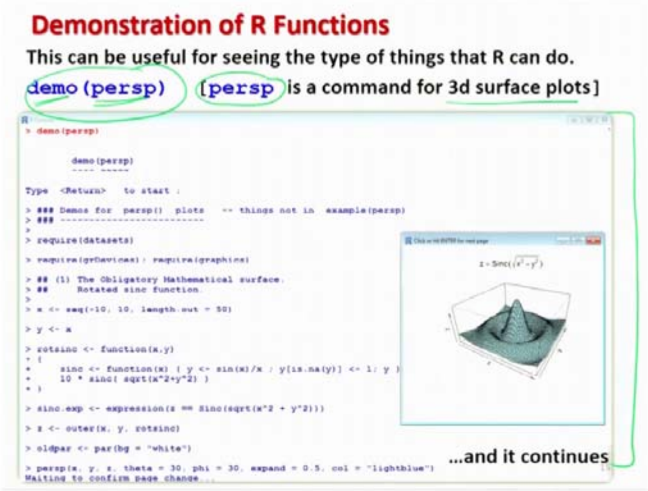


```
lm> ## End(No test)
lm> opar <- par(mfrow =

lm> plot(lm.D9, las = 1
Waiting to confirm page

lm> par(opar)

lm> ## Don't show:
lm> ## model frame :
lm> stopifnot(identical
lm+

lm> ## End(Don't show)
lm> ### less simple exa
lm>
```

Now, if I try to see here example see here l m, right and if I try to press it here it has started now it is giving here click, it is giving you here some details here that you can read in detail, it is yeah, well we are not going to do the here this thing, but if I try to say here enter you are getting here different types of here options from the application and their graphics etc. etc.

And if you try to see here just try to move forward you can see here different types of graphics etc. etc., they are given here right and if you want to close this window you simply have to just press here clear and I can clear the screen, well, I am not asking you to read it at the moment my objective is simply to show you that how these things can be can be obtained.

(Refer Slide Time: 34:25)



Demonstration of R Functions

This can be useful for seeing the type of things that R can do.

demo(persp) [persp is a command for 3d surface plots]

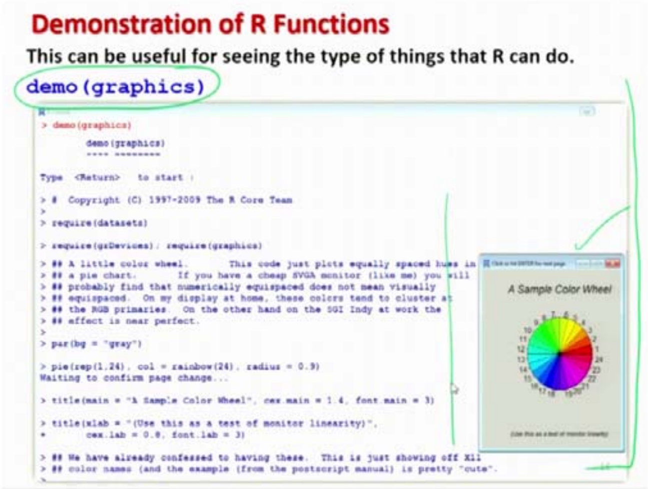
```
> demo(persp)


        demo(persp)
        ---- ------

Type  <Return>   to start :

> ### Demos for  persp()  plots   -- things not in  example(persp)
> ### -------------------------
>
> require(datasets)

> require(grDevices); require(graphics)

> ## (1) The Obligatory Mathematical surface.
> ##     Rotated sinc function.
>
> x <- seq(-10, 10, length.out = 50)

> y <- x

> rotsinc <- function(x,y)
+ {
+     sinc <- function(x) { y <- sin(x)/x ; y[is.na(y)] <- 1; y }
+     10 * sinc( sqrt(x^2+y^2) )
+ }

> sinc.exp <- expression(z == Sinc(sqrt(x^2 + y^2)))

> z <- outer(x, y, rotsinc)

> oldpar <- par(bg = "white")

> persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
Waiting to confirm page change...
```

...and it continues

Now, after this you would also like to have a demonstration for example, you always ask your teacher in the class sir, can you please solve some questions on the blackboard? Because you want to follow that how these things have been actually solved. So, the similar type of facility is available in the R software also for demonstrating the R function and for this thing, the command here is demo and within the parenthesis you have to write the R command for which you need a demonstration it may be possible that all the commands may not have a demonstration, but I am sure that most of the important commands do have a demonstration.

For example, here I would like to show you that suppose somebody wants to create a three d surface plot and in the R software there is a command here persp p e r s p. So, if you simply try to write down here demo persp here then you will get here this type of screenshot which I am trying to show you here, right. It has got all the details and it continues it is only the first screen I am giving you, but after that it will give you here more thing.

(Refer Slide Time: 35:31)



Demonstration of R Functions

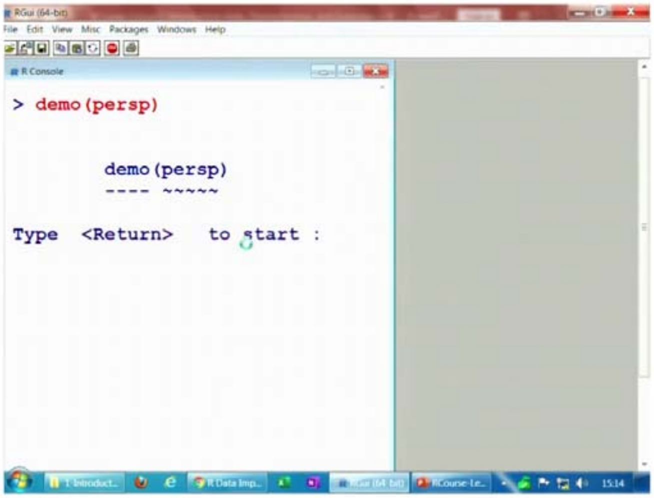
This can be useful for seeing the type of things that R can do.

demo(graphics)

```
> demo(graphics)

        demo(graphics)
        ---- --------

Type  <Return>   to start :

> #  Copyright (C) 1997-2009 The R Core Team
>
> require(datasets)

> require(grDevices); require(graphics)

> ## A little color wheel.       This code just plots equally spaced hues in
> ## a pie chart.       If you have a cheap SVGA monitor (like me) you will
> ## probably find that numerically equispaced does not mean visually
> ## equispaced.  On my display at home, these colors tend to cluster at
> ## the RGB primaries.  On the other hand on the SGI Indy at work the
> ## effect is near perfect.
>
> par(bg = "gray")

> pie(rep(1,24), col = rainbow(24), radius = 0.9)
Waiting to confirm page change...

> title(main = "A Sample Color Wheel", cex.main = 1.4, font.main = 3)

> title(xlab = "(Use this as a test of monitor linearity)",
+       cex.lab = 0.8, font.lab = 3)

> ## We have already confessed to having these.  This is just showing off X11
> ## color names (and the example (from the postscript manual) is pretty "cute".
```

And similarly if you want to see the demonstration of sender command here demo of some command graphics you can simply type here demo graphics- demo and inside the parenthesis graphics and then you will get here this type of outcome you will get here graphics you will get here all the text and you simply have to read it and then you can understand it how the things are happening.

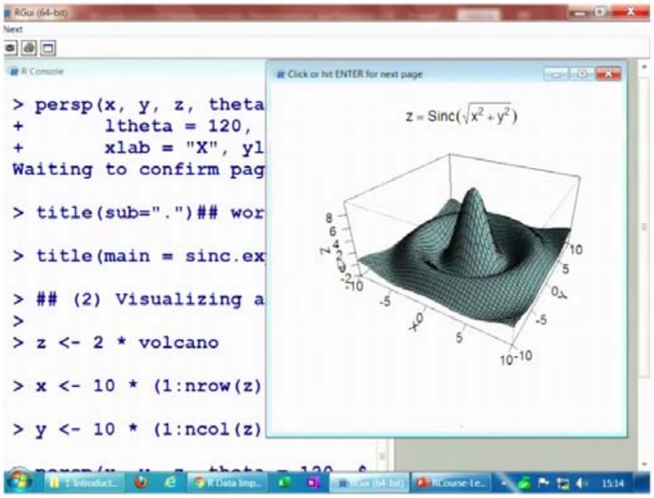
(Refer Slide Time: 36:05)



```
> demo(persp)


        demo(persp)
        ---- ~~~~~

Type  <Return>   to start :
```

So, let me try to show you it on the R console for example, demo persp and if you try to enter here it is trying to, now you have to follow the instruction it is giving with the demonstration it is writing type written to start, ok.

(Refer Slide Time: 36:21)

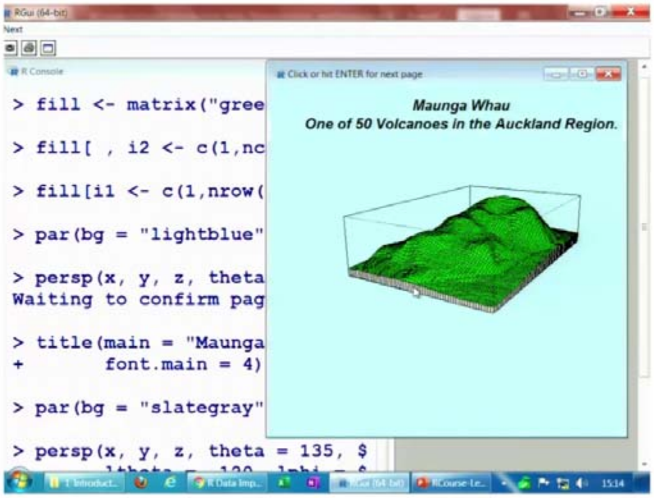


RGui (64-bit)
Next
R Console
Click or hit ENTER for next page
z = Sinc($\sqrt{x^2 + y^2}$)
8
6
4
2
0
-2
Z
-10
-5
0
X
5
10
-10
-5
0
Y
5
10

```
> persp(x, y, z, theta
+        ltheta = 120,
+        xlab = "X", yl
Waiting to confirm pag

> title(sub=".")## wor

> title(main = sinc.ex

> ## (2) Visualizing a
>
> z <- 2 * volcano

> x <- 10 * (1:nrow(z)

> y <- 10 * (1:ncol(z)
```

I will enter it and then it will give me here some details you have to read it and then after that it is showing me here that click or hit enter for the next page. So, I will keep on here entering and then the different types of 3D surfaces plots, they are coming over here.

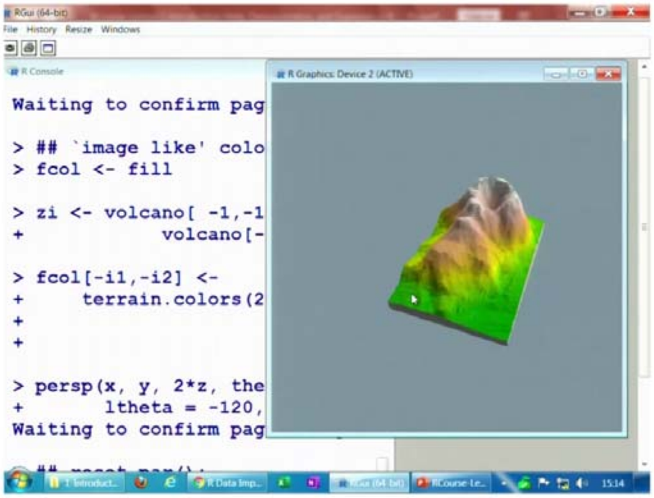
(Refer Slide Time: 36:35)



```
> fill <- matrix("gree

> fill[ , i2 <- c(1,nc

> fill[i1 <- c(1,nrow(

> par(bg = "lightblue"

> persp(x, y, z, theta
Waiting to confirm pag

> title(main = "Maunga
+         font.main = 4)

> par(bg = "slategray"

> persp(x, y, z, theta = 135, $
```

RGui (64-bit)
Next
R Console
Click or hit ENTER for next page
Maunga Whau
One of 50 Volcanoes in the Auckland Region.
15:14

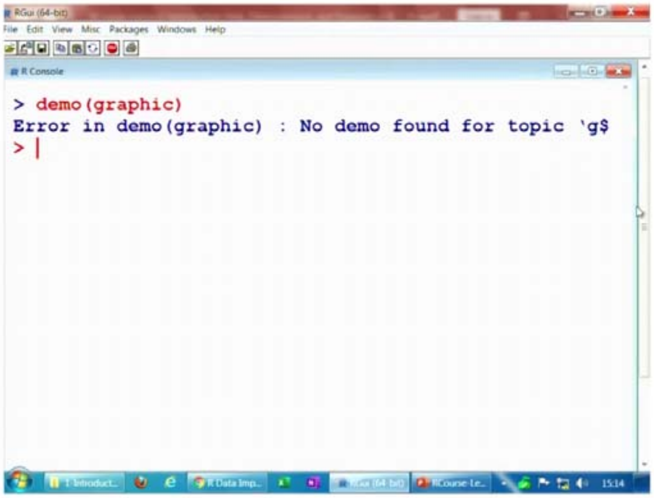
(Refer Slide Time: 36:44)



```
Waiting to confirm pag

> ## `image like' colo
> fcol <- fill

> zi <- volcano[ -1,-1
+            volcano[-

> fcol[-i1,-i2] <-
+     terrain.colors(2
+
+

> persp(x, y, 2*z, the
+        ltheta = -120,
Waiting to confirm pag
```

And you can see here that how beautiful curves and graphs can be prepared in the R software and that is also for free right.

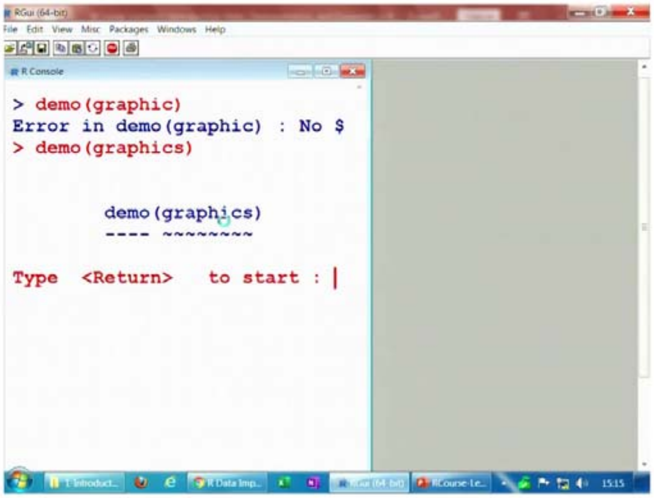
(Refer Slide Time: 36:50)



```
> demo(graphic)
Error in demo(graphic) : No demo found for topic 'g$
> |
```

Similarly, if you try to take here the demo of graphic. So, you can see here demo graphics, let me try to make here one mistake, suppose I try here only graphic I do not type here as see what happened it will show you here error in demo graphic no demo found for the topic graphic. So, as I told you that R always give you some error messages whenever you are trying to make some mistake and the language of the error message is quite understandable.

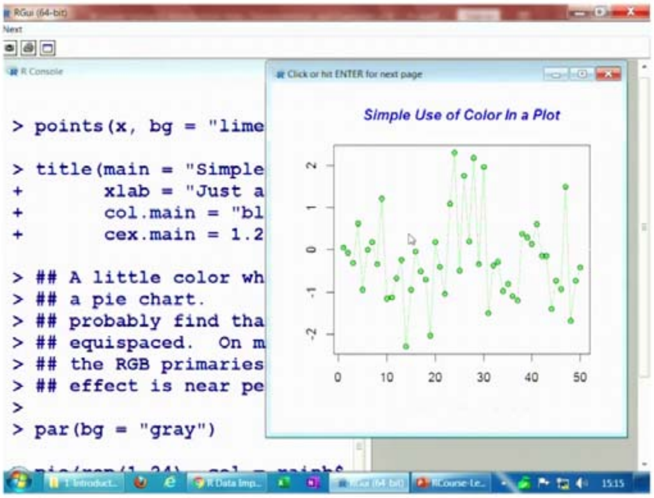
(Refer Slide Time: 37:26)



```
> demo(graphic)
Error in demo(graphic) : No $
> demo(graphics)


        demo(graphics)
        ---- ~~~~~~~~

Type  <Return>   to start : |
```

So, here it is trying to show you the same thing that ok no demo found for the topic graphic because graphic is not the correct command, the correct command is graphics. Now if I want to type here demo graphics I can use my arrow key and I can repeat the earlier command and if as soon as I type here demo graphics and I enter it will ask me to type return.

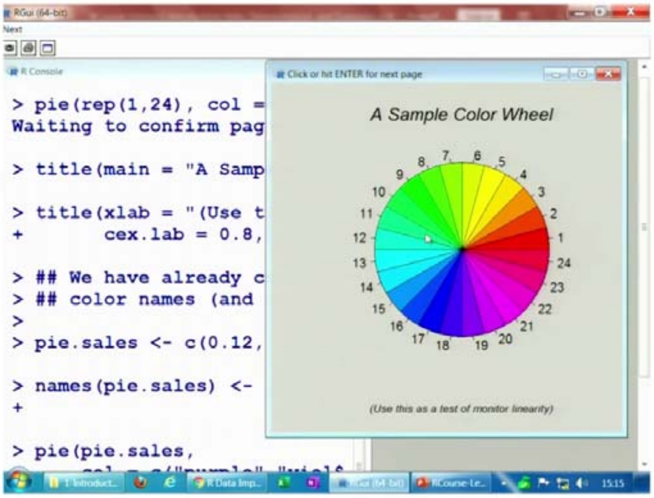
(Refer Slide Time: 37:42)



RGui (64-bit)
Next
R Console
Click or hit ENTER for next page
Simple Use of Color In a Plot
2
1
0
-1
-2
0
10
20
30
40
50

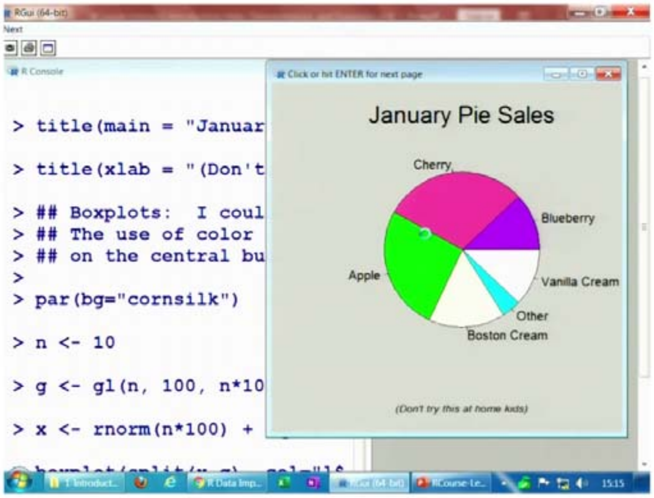
```
> points(x, bg = "lime

> title(main = "Simple
+       xlab = "Just a
+       col.main = "bl
+       cex.main = 1.2

> ## A little color wh
> ## a pie chart.
> ## probably find tha
> ## equispaced.  On m
> ## the RGB primaries
> ## effect is near pe
>
> par(bg = "gray")
```

(Refer Slide Time: 37:50)



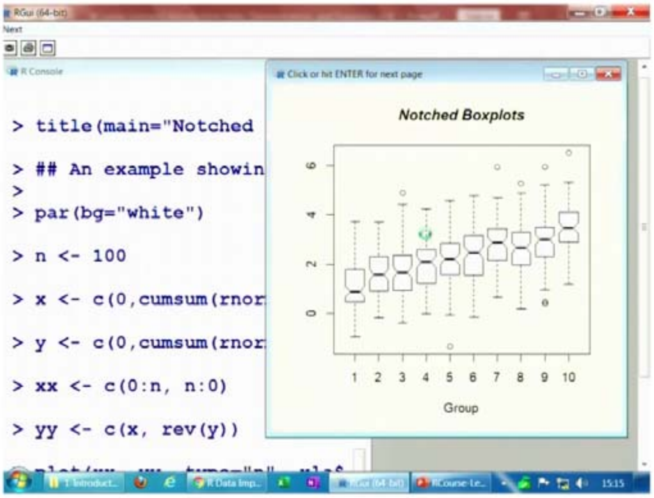
```
> pie(rep(1,24), col =
Waiting to confirm pag

> title(main = "A Samp

> title(xlab = "(Use t
+       cex.lab = 0.8,

> ## We have already c
> ## color names (and
>
> pie.sales <- c(0.12,

> names(pie.sales) <-
+

> pie(pie.sales,
```

(Refer Slide Time: 37:51)



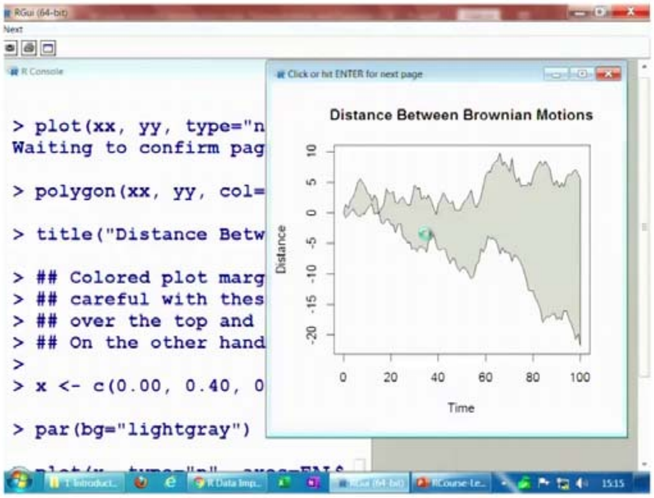
```
> title(main = "Januar

> title(xlab = "(Don't

> ## Boxplots:  I coul
> ## The use of color
> ## on the central bu
>
> par(bg="cornsilk")

> n <- 10

> g <- gl(n, 100, n*10

> x <- rnorm(n*100) +

> boxplot(split(x,g), col="l$
```

(Refer Slide Time: 37:51)



```
> title(main="Notched

> ## An example showin
>
> par(bg="white")

> n <- 100

> x <- c(0,cumsum(rnor

> y <- c(0,cumsum(rnor

> xx <- c(0:n, n:0)

> yy <- c(x, rev(y))

> plot(xx, yy, type="n", xla$
```

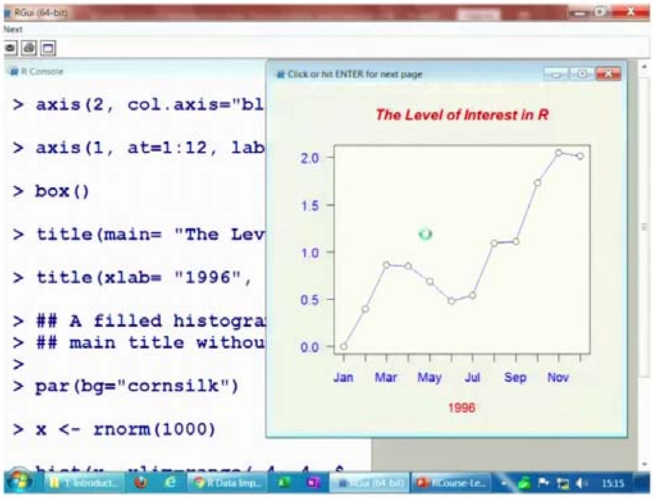
(Refer Slide Time: 37:52)



```
> plot(xx, yy, type="n
Waiting to confirm pag

> polygon(xx, yy, col=

> title("Distance Betw

> ## Colored plot marg
> ## careful with thes
> ## over the top and
> ## On the other hand
>
> x <- c(0.00, 0.40, 0

> par(bg="lightgray")

> plot(x, type="n", axes=FAL$
```

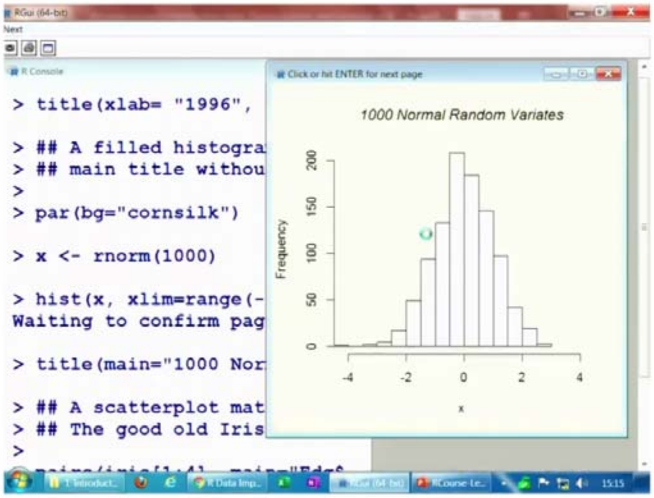
RGui (64-bit)
Next
R Console
Click or hit ENTER for next page
Distance Between Brownian Motions
Distance
Time
10
5
0
-5
-10
-15
-20
0
20
40
60
80
100

(Refer Slide Time: 37:53)



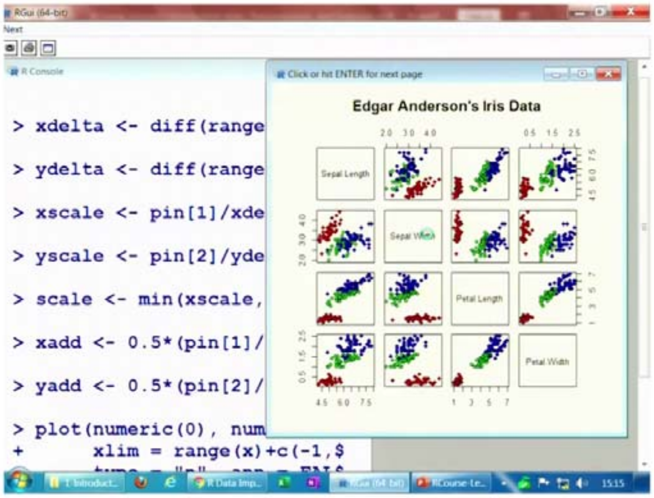
```
> axis(2, col.axis="bl

> axis(1, at=1:12, lab

> box()

> title(main= "The Lev

> title(xlab= "1996",

> ## A filled histogra
> ## main title withou
>
> par(bg="cornsilk")

> x <- rnorm(1000)

> hist(x, xlim=range(-4, 4, $
```

(Refer Slide Time: 37:53)



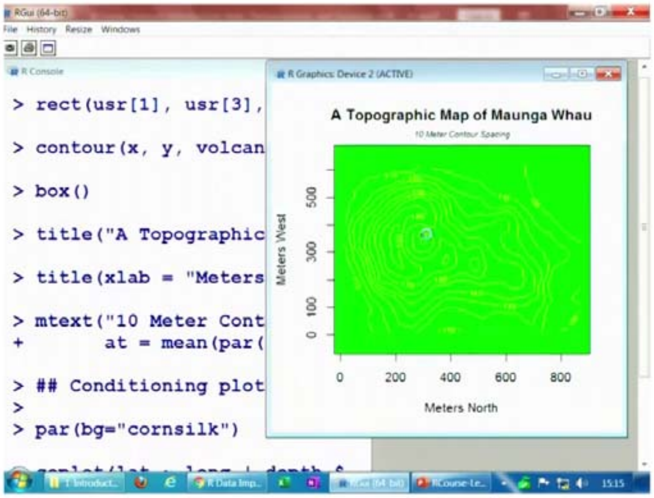
```
> title(xlab= "1996",

> ## A filled histogra
> ## main title withou
>
> par(bg="cornsilk")

> x <- rnorm(1000)

> hist(x, xlim=range(-
Waiting to confirm pag

> title(main="1000 Nor

> ## A scatterplot mat
> ## The good old Iris
>
> pairs(iris[1:4], main="Edg$
```

(Refer Slide Time: 37:54)



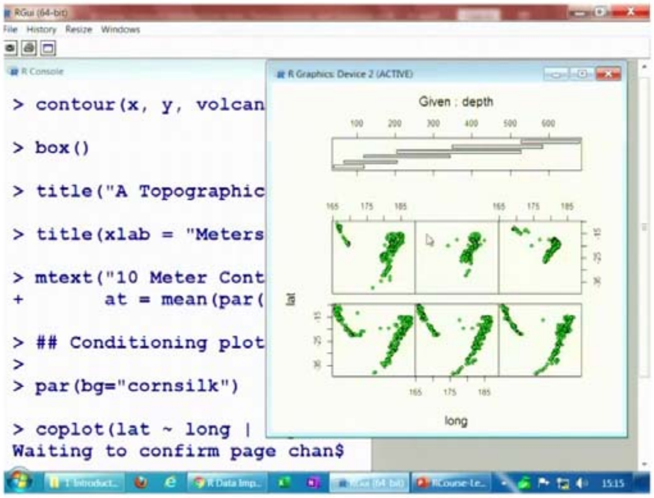
```
> xdelta <- diff(range

> ydelta <- diff(range

> xscale <- pin[1]/xde

> yscale <- pin[2]/yde

> scale <- min(xscale,

> xadd <- 0.5*(pin[1]/

> yadd <- 0.5*(pin[2]/

> plot(numeric(0), num
+       xlim = range(x)+c(-1,$
        type = "n", ann = FAL$
```

(Refer Slide Time: 37:56)



```
> rect(usr[1], usr[3],

> contour(x, y, volcan

> box()

> title("A Topographic

> title(xlab = "Meters

> mtext("10 Meter Cont
+       at = mean(par(

> ## Conditioning plot
>
> par(bg="cornsilk")
```

(Refer Slide Time: 37:56)



```
> contour(x, y, volcan

> box()

> title("A Topographic

> title(xlab = "Meters

> mtext("10 Meter Cont
+       at = mean(par(

> ## Conditioning plot
>
> par(bg="cornsilk")

> coplot(lat ~ long |
Waiting to confirm page chan$
```

And then you can see here on the left hand side this is theory and then it is writing here click or hit enter for the next page I try to do it here and then I keep on entering here and then I will get here that how to do all these graphics. You can see here how beautiful graphics can be created in the R software without doing actually much, right. So, now, let us stop in this lecture and we come to an end in this lecture.

So, if you try to see in this lecture, I have tried my best to give you an idea that whenever you want to begin with the new topic how you can seek the helps in different formats. I am not saying that these are the only formats, but there are many more things which are available, but my constraint is that if I start doing each and everything, then possibly it will be a very long course. So, as I said in the beginning you have to hold my hands and after that I will set you free.

So, my request is that you please try to have a look on this lecture try to revise it and then try to use some other functions and try to use the same command, for example, for this demo example etc. and try to see how you can get it. It may be possible that you may make some mistakes in the beginning, don't worry try again and, but at least try to use these commands for couple of times with different functions different examples so that these things are settled down in your mind.

So, at a later stage, whenever you want to use them these things will click back into your mind and you can use them very easily.

So, you try to practice them and I will see you in the next lecture with more topics, till then good bye.