


**Dealing with Materials Data
Collection, Analysis and Interpretation
Professor MP Gururajan
Department of Metallurgical Engineering and Materials Science
Indian Institute of Technology, Bombay
Lecture 09
R: Demos and getting help**

(Refer Slide Time: 00:20)

Dealing with Materials Data: collection, analysis and interpretation

M P Gururajan and Hina A Gokhale

Indian Institute of Technology Bombay, Mumbai


 NPTEL Guru and Hina Dealing with Materials Data IITB 1/4

Welcome to dealing with materials data collection, analysis and interpretation. I am Gururajan and we are doing a module on R programming language.

(Refer Slide Time: 00:31)

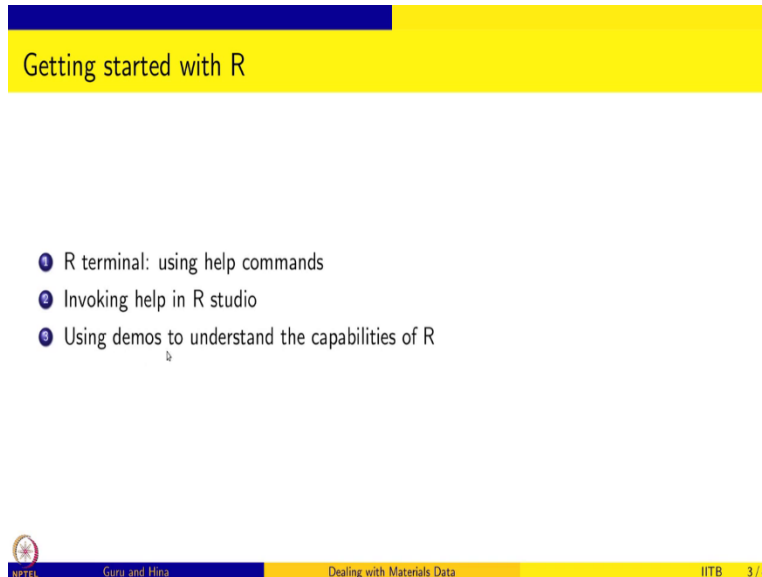
Module: Introduction to R

Demos and getting help

 NPTEL Guru and Hina Dealing with Materials Data IITB 2/4

And, so this is an introductory session. This is an introduction to R. And in this session we are going to look at demos and how to get help when you are working with R that is the aim of this session to let you know how to get help and how to look at some of the demos.

(Refer Slide Time: 00:49)



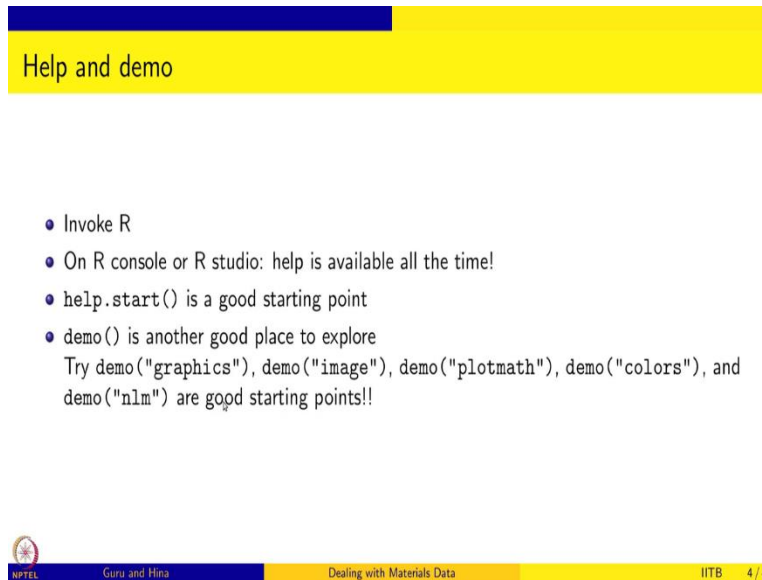
Getting started with R

- 1 R terminal: using help commands
- 2 Invoking help in R studio
- 3 Using demos to understand the capabilities of R

NPTEL Guru and Hina Dealing with Materials Data IITB 3 / 4

So, in R terminal using help command, how to use help command to get information is the first thing you are going to do. I will also show how to invoke help in R studio. I told you about R studio, but we have not used R studio yet. So, we will use and we will use some of the demos to understand the capabilities of R.

(Refer Slide Time: 01:15)



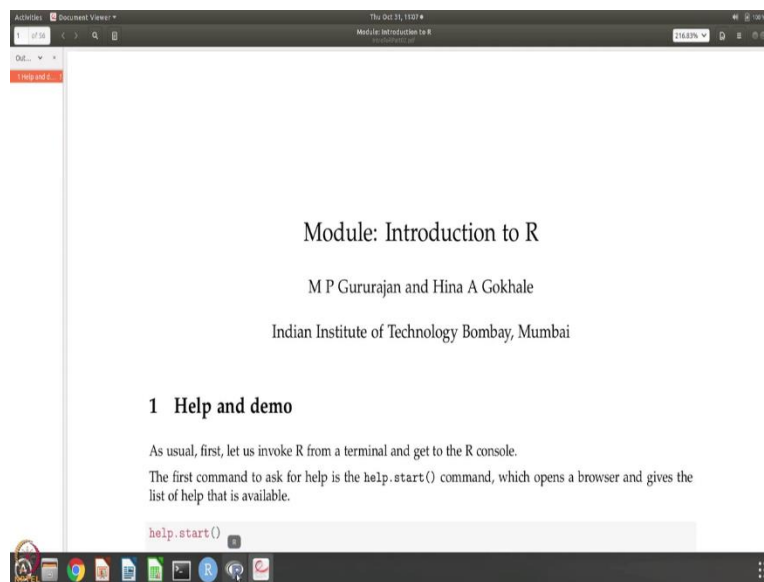
Help and demo

- Invoke R
- On R console or R studio: help is available all the time!
- `help.start()` is a good starting point
- `demo()` is another good place to explore
Try `demo("graphics")`, `demo("image")`, `demo("plotmath")`, `demo("colors")`, and `demo("nlm")` are good starting points!!

NPTEL Guru and Hina Dealing with Materials Data IITB 4/4

Specifically, what we are going to do is to invoke R and, and the help is available all the time in the R console, or if you are an R studio, and `help dot start` is a good starting point. If you do not know anything, that is the best place to start the help. And the `demo` is another good place to explore. `Demo` basically gives you some idea of the capabilities of R. So, `graphics`, `image`, `plot math`, `colors`, and `nlm` are some of the demos that we are going to look at in this.

(Refer Slide Time: 02:00)



Module: Introduction to R

M P Gururajan and Hina A Gokhale

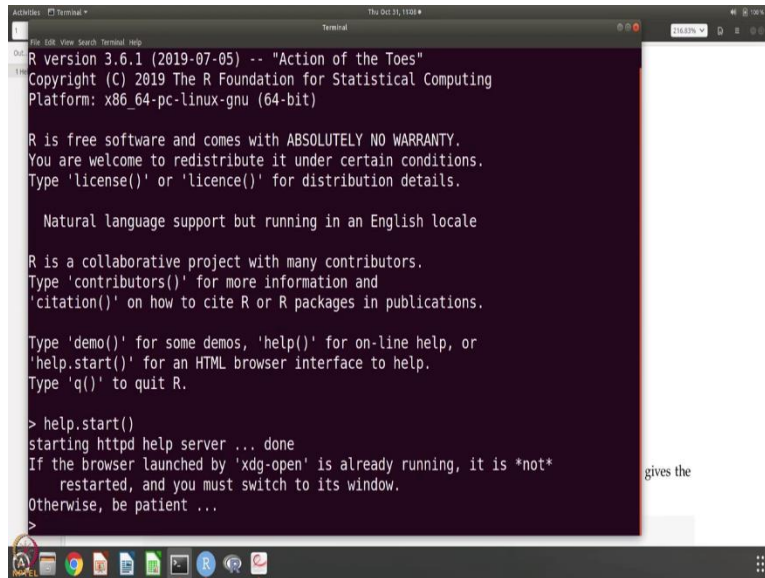
Indian Institute of Technology Bombay, Mumbai

1 Help and demo

As usual, first, let us invoke R from a terminal and get to the R console.

The first command to ask for help is the `help.start()` command, which opens a browser and gives the list of help that is available.

```
help.start()
```



```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (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

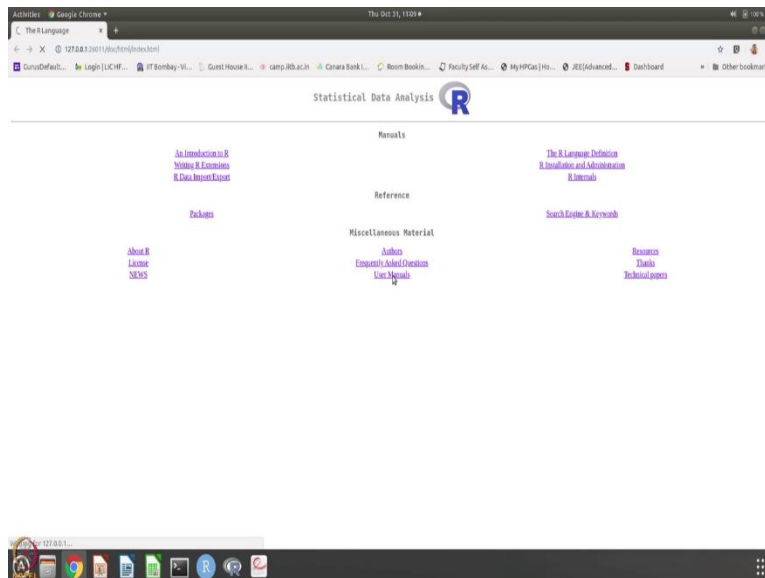
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

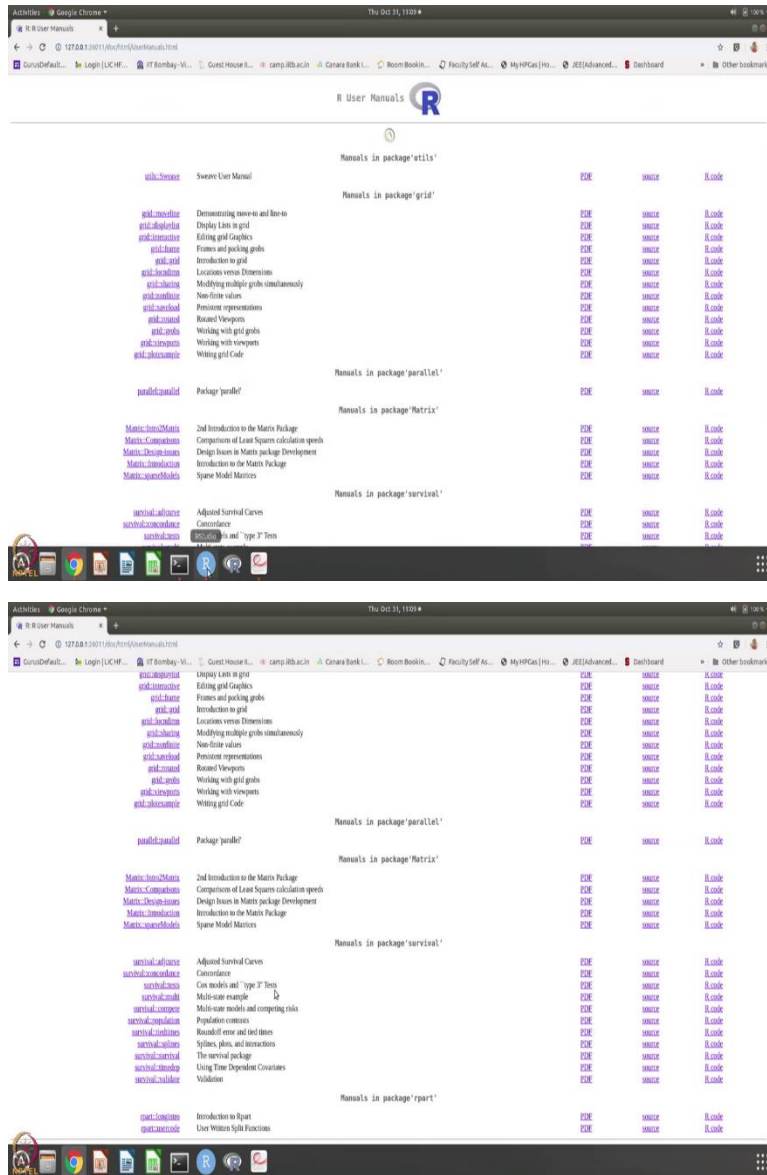
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> help.start()
starting httpd help server ... done
If the browser launched by 'xdg-open' is already running, it is *not*
restarted, and you must switch to its window.
Otherwise, be patient ...
>
```

So, like I said, I have some notes prepared. Let us do that. And this is the notes. So let us invoke R last time I showed how to invoke R from the terminal. It is also possible that if you have this shortcut, and then you just click on this, you get the R console. And like I said, R version is 3.6.1 action of the toes is what we are using. So, let us say help that we want to get so help dot start is a good starting point.

(Refer Slide Time: 02:28)





And it will open browser and in the browser and in the browser it will give you all the information so as you can see, this is very very detailed help. There is an introduction to R, this is what I mentioned in the manuals, this is the material that we want to want I want you to explore and that this is the kind of material that we are going to cover.

And there are also other information like how to write R extensions. So, if you are a developer or the language definition, and installation and administration. So, this is where I recommended that you use spoken tutorials for example, but it is possible to get that help from the R documentation itself and what are the, how does R work?

So, this is about the internals and how to get data into R and get data out of R. So, there are other references what are the packages and search engine keywords and about license and news and resources and lots of other material. User manuals is another thing that I mentioned. And if you go to user manuals, for example you will get lots of information, right. So, what are the manuals in package survival? So, what are the manuals? Manuals in package R part what are they? So, there is plenty of information that you can get by using this command, help that start.

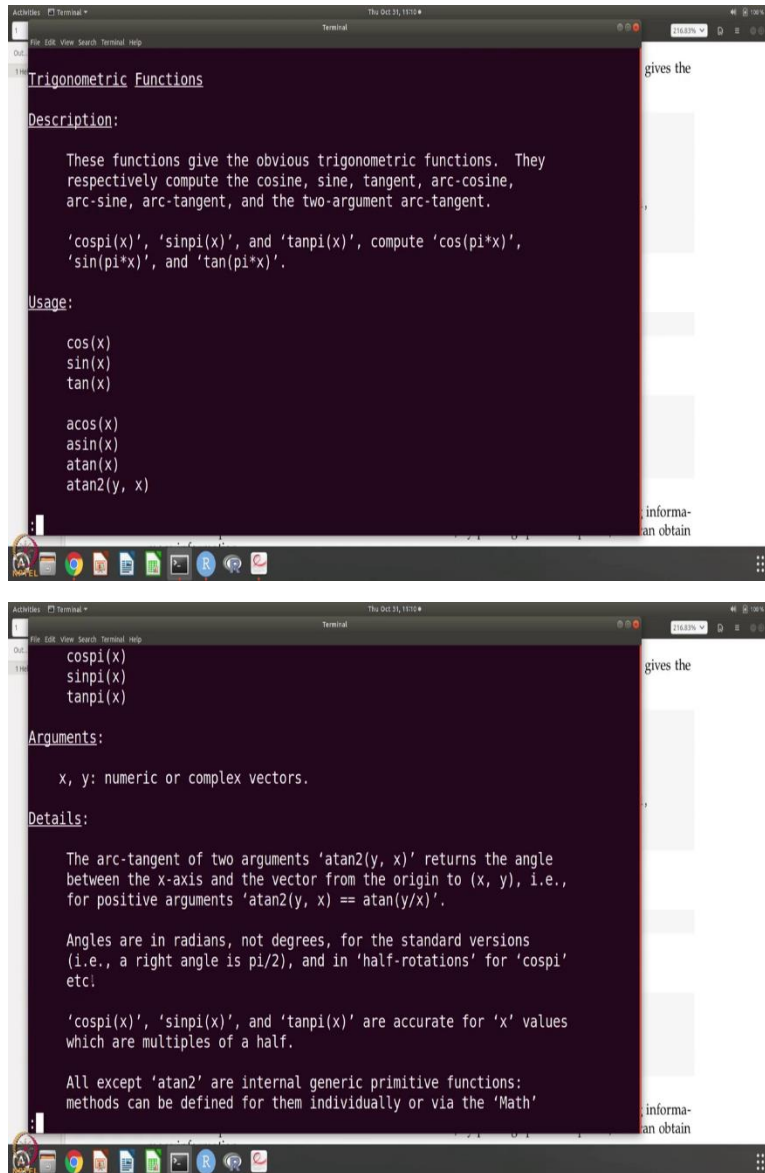
(Refer Slide Time: 03:52)

The top screenshot shows a Document Viewer window displaying a help page. The text on the page includes:

```
As usual, first, let us invoke it from a terminal and get to the R console.  
The first command to ask for help is the help.start() command, which opens a browser and gives the list of help that is available.  
  
help.start()  
  
## starting httpd help server ... done  
  
## If the browser launched by 'xdg-open' is already running, it is *not* restarted,  
## and you must switch to its window.  
## Otherwise, be patient ...  
  
One can also ask for specific help if a command or key word is known.  
  
help("sin")  
  
The help command is useful even if you do not know the commands well; here is an example:  
  
help("axes")  
  
## No documentation for 'axes' in specified packages and libraries:  
## you could try '??axes'  
  
Even though the help file does not give any information on axes, it suggests a way for obtaining information on all help documents that contain the word axes. Now, by probing specific help files, one can obtain
```

The bottom screenshot shows a Terminal window running R. The output is as follows:

```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"  
Copyright (C) 2019 The R Foundation for Statistical Computing  
Platform: x86_64-pc-linux-gnu (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.  
  
> help.start()  
starting httpd help server ... done  
If the browser launched by 'xdg-open' is already running, it is *not*  
restarted, and you must switch to its window.  
Otherwise, be patient ...  
> help("sin")
```

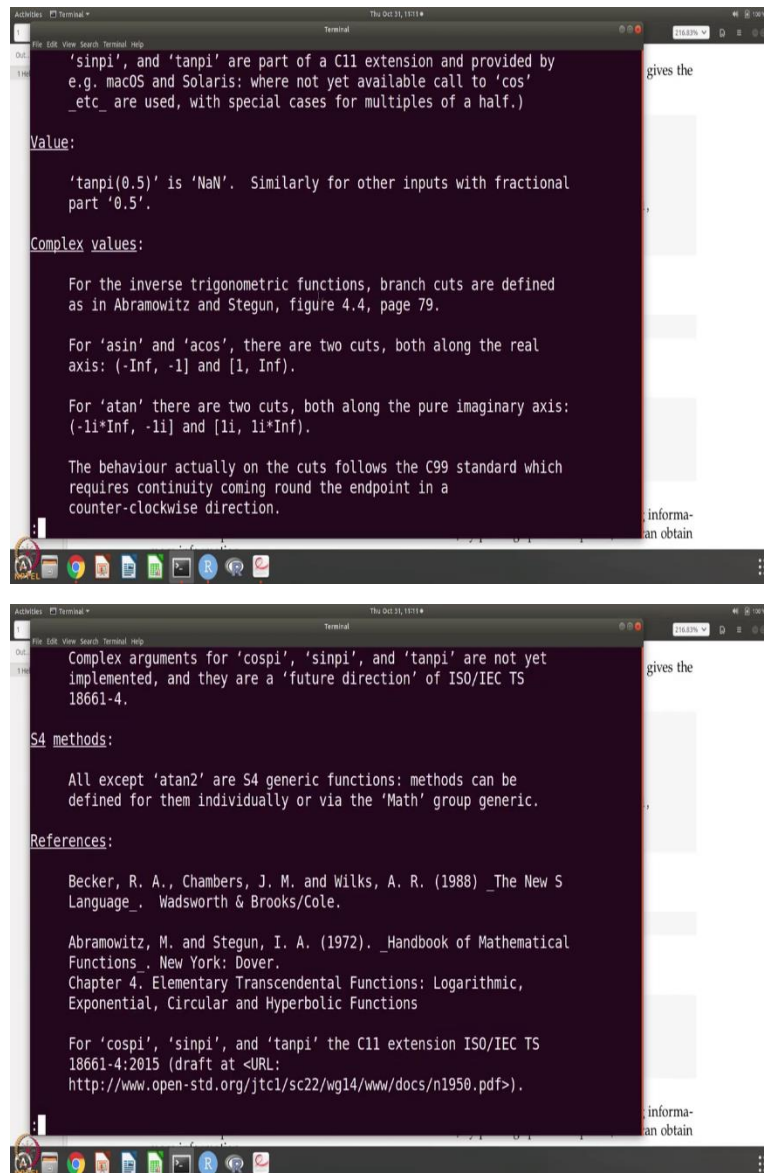


You can also get specific help like I show here. For example, If you know let us say that I want to know what is the sin function. So, I say help sin and then I get information. So, these are trigonometric functions these functions give the obvious trigonometric functions because it is sin, cosine etc. And they respectively compute the cosine, sin, tangent, arc-cosine, arc-sin, arc-tangent and the 2 argument arc-tangent.

So, you have this cos pi x, sin pi x, tan pi x etc. Compute cos pi into x, sin pi into x, tan pi into x. So, this is another information that is given. So, it is cos, sin, tan, A cos, A sin, A tan2 and cos pi, sin pi, tan pi. So, these are numeric or complex vector. So, the arguments that it takes for example, there is a y here that is why y is given and so they are numeric or complex vectors.

So, you can calculate sin for a bunch of values and that is more informations and, and the most important information for example, angles are in radians not degrees for the standard versions, right. So, which means if you want to calculate any of these trigonometric functions, then you have to change the angle value if it is given in degrees into radians. So, so information like this what is the input in what unit it is there? What is the command?

(Refer Slide Time: 05:31)



The image shows two screenshots of a terminal window displaying documentation for C11 trigonometric functions. The top screenshot covers the 'Value:' and 'Complex values:' sections, while the bottom screenshot covers the 'S4 methods:' and 'References:' sections.

```
'sinpi', and 'tanpi' are part of a C11 extension and provided by
e.g. macOS and Solaris: where not yet available call to 'cos'
etc_ are used, with special cases for multiples of a half.)

Value:

'tanpi(0.5)' is 'NaN'. Similarly for other inputs with fractional
part '0.5'.

Complex values:

For the inverse trigonometric functions, branch cuts are defined
as in Abramowitz and Stegun, figure 4.4, page 79.

For 'asin' and 'acos', there are two cuts, both along the real
axis: (-Inf, -1] and [1, Inf).

For 'atan' there are two cuts, both along the pure imaginary axis:
(-1i*Inf, -1i) and [1i, 1i*Inf).

The behaviour actually on the cuts follows the C99 standard which
requires continuity coming round the endpoint in a
counter-clockwise direction.
```

```
Complex arguments for 'cospi', 'sinpi', and 'tanpi' are not yet
implemented, and they are a 'future direction' of ISO/IEC TS
18661-4.

S4 methods:

All except 'atan2' are S4 generic functions: methods can be
defined for them individually or via the 'Math' group generic.

References:

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The New S
Language_. Wadsworth & Brooks/Cole.

Abramowitz, M. and Stegun, I. A. (1972). _Handbook of Mathematical
Functions_. New York: Dover.
Chapter 4. Elementary Transcendental Functions: Logarithmic,
Exponential, Circular and Hyperbolic Functions

For 'cospi', 'sinpi', and 'tanpi' the C11 extension ISO/IEC TS
18661-4:2015 (draft at <URL:
http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1950.pdf>).
```



```
Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The New S
Language_. Wadsworth & Brooks/Cole.

Abramowitz, M. and Stegun, I. A. (1972). _Handbook of Mathematical
Functions_. New York: Dover.
Chapter 4. Elementary Transcendental Functions: Logarithmic,
Exponential, Circular and Hyperbolic Functions

For 'cospi', 'sinpi', and 'tanpi' the C11 extension ISO/IEC TS
18661-4:2015 (draft at <URL:
http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1950.pdf>).

Examples:

x <- seq(-3, 7, by = 1/8)
tx <- cbind(x, cos(pi*x), cospi(x), sin(pi*x), sinpi(x),
           tan(pi*x), tanpi(x), deparse.level=2)
op <- options(digits = 4, width = 90) # for nice formatting
head(tx)
tx[ (x %% 1) %in% c(0, 0.5) ,]
options(op)

(END)
```

```
Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The New S
Language_. Wadsworth & Brooks/Cole.

Abramowitz, M. and Stegun, I. A. (1972). _Handbook of Mathematical
Functions_. New York: Dover.
Chapter 4. Elementary Transcendental Functions: Logarithmic,
Exponential, Circular and Hyperbolic Functions

For 'cospi', 'sinpi', and 'tanpi' the C11 extension ISO/IEC TS
18661-4:2015 (draft at <URL:
http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1950.pdf>).

Examples:

x <- seq(-3, 7, by = 1/8)
tx <- cbind(x, cos(pi*x), cospi(x), sin(pi*x), sinpi(x),
           tan(pi*x), tanpi(x), deparse.level=2)
op <- options(digits = 4, width = 90) # for nice formatting
head(tx)
tx[ (x %% 1) %in% c(0, 0.5) ,]
options(op)

(END)
```

```

x cos(pi * x... cospi(x) sin(pi * x... sinpi(x) tan(pi * x... tanpi(x)
[1,] -3.0 -1.000e+00 -1 -3.674e-16 0 3.674e-16 0
[2,] -2.5 3.062e-16 0 -1.000e+00 -1 -3.266e+15 NaN
[3,] -2.0 1.000e+00 1 2.449e-16 0 2.449e-16 0
[4,] -1.5 -1.837e-16 0 1.000e+00 1 -5.444e+15 NaN
[5,] -1.0 -1.000e+00 -1 -1.225e-16 0 1.225e-16 0
[6,] -0.5 6.123e-17 0 -1.000e+00 -1 -1.633e+16 NaN
[7,] 0.0 1.000e+00 1 0.000e+00 0 0.000e+00 0
[8,] 0.5 6.123e-17 0 1.000e+00 1 1.633e+16 NaN
[9,] 1.0 -1.000e+00 -1 1.225e-16 0 -1.225e-16 0
[10,] 1.5 -1.837e-16 0 -1.000e+00 -1 5.444e+15 NaN
[11,] 2.0 1.000e+00 1 -2.449e-16 0 -2.449e-16 0
[12,] 2.5 3.062e-16 0 1.000e+00 1 3.266e+15 NaN
[13,] 3.0 -1.000e+00 -1 3.674e-16 0 -3.674e-16 0
[14,] 3.5 -4.286e-16 0 -1.000e+00 -1 2.333e+15 NaN
[15,] 4.0 1.000e+00 1 -4.899e-16 0 -4.899e-16 0
[16,] 4.5 5.511e-16 0 1.000e+00 1 1.815e+15 NaN
[17,] 5.0 -1.000e+00 -1 6.123e-16 0 -6.123e-16 0
[18,] 5.5 -2.450e-15 0 -1.000e+00 -1 4.082e+14 NaN
[19,] 6.0 1.000e+00 1 -7.348e-16 0 -7.348e-16 0
[20,] 6.5 -9.803e-16 0 1.000e+00 1 -1.020e+15 NaN
[21,] 7.0 -1.000e+00 -1 8.573e-16 0 -8.573e-16 0
>
options(op)

```

And the help file is also very useful because you will see lots of other information. So, and there is also the reference and what I liked the best, the examples. So, you will be able to directly take these examples and run them. So, you can copy and you can paste them here. You can, let us copy so, the up key command actually gets you the information that you got in the previous command and let me try to copy this commands, copy and let me paste. So, you can just use this and so it is a very easy and fast way of getting used to some of these commands.

(Refer Slide Time: 06:39)

```

AS usual, first, let us invoke R from a terminal and get to the R console.

The first command to ask for help is the help.start() command, which opens a browser and gives the
list of help that is available.

help.start()

## starting httpd help server ... done

## If the browser launched by 'xdg-open' is already running, it is *not* restarted,
## and you must switch to its window.
## Otherwise, be patient ...

One can also ask for specific help if a command or key word is known.

help("sin")

The help command is useful even if you do not know the commands well; here is an example:

help("axes")

## No documentation for 'axes' in specified packages and libraries:
## you could try '?axes'

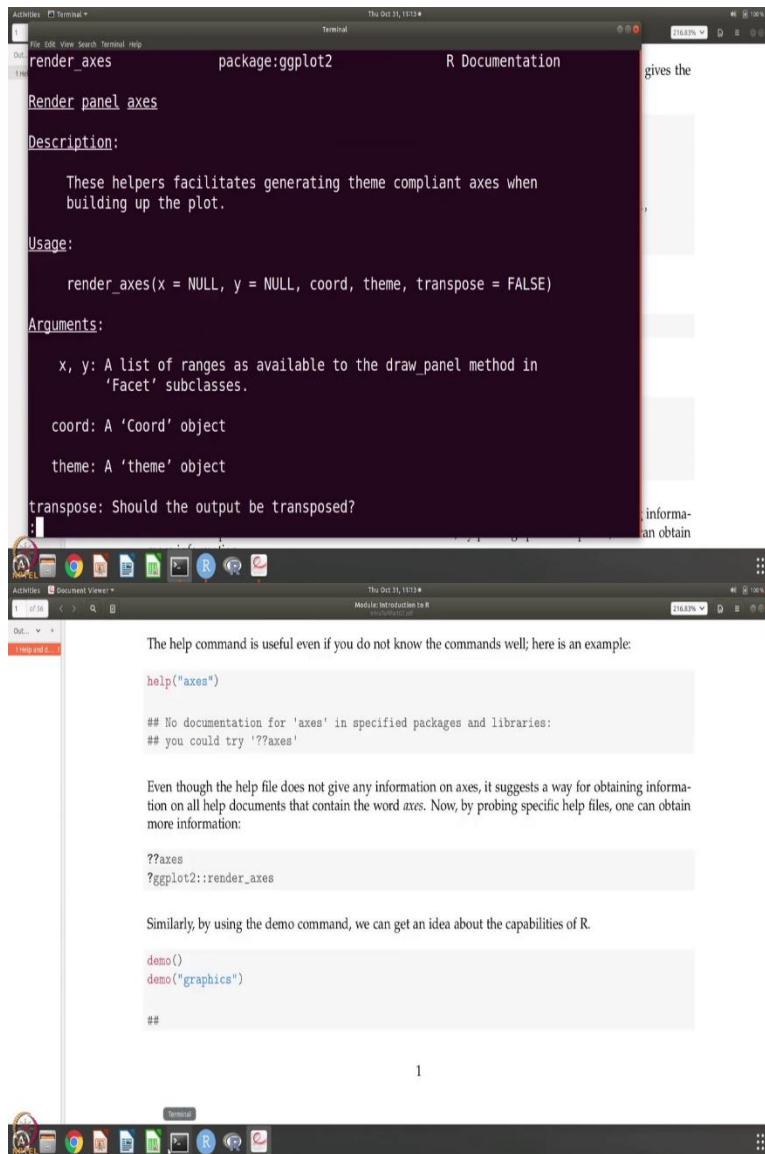
Even though the help file does not give any information on axes, it suggests a way for obtaining information
on all help documents that contain the word axes. Now, by probing specific help files, one can obtain

```

```
Terminal
Thu Oct 31, 10:12 AM
[4,] -1.5 -1.837e-16 0 1.000e+00 1 -5.444e+15 NaN
[5,] -1.0 -1.000e+00 -1 -1.225e-16 0 1.225e-16 0
[6,] -0.5 6.123e-17 0 -1.000e+00 -1 -1.633e+16 NaN
[7,] 0.0 1.000e+00 1 0.000e+00 0 0.000e+00 0
[8,] 0.5 6.123e-17 0 1.000e+00 1 1.633e+16 NaN
[9,] 1.0 -1.000e+00 -1 1.225e-16 0 -1.225e-16 0
[10,] 1.5 -1.837e-16 0 -1.000e+00 -1 5.444e+15 NaN
[11,] 2.0 1.000e+00 1 -2.449e-16 0 -2.449e-16 0
[12,] 2.5 3.062e-16 0 1.000e+00 1 3.266e+15 NaN
[13,] 3.0 -1.000e+00 -1 3.674e-16 0 -3.674e-16 0
[14,] 3.5 -4.286e-16 0 -1.000e+00 -1 2.333e+15 NaN
[15,] 4.0 1.000e+00 1 -4.899e-16 0 -4.899e-16 0
[16,] 4.5 5.511e-16 0 1.000e+00 1 1.815e+15 NaN
[17,] 5.0 -1.000e+00 -1 6.123e-16 0 -6.123e-16 0
[18,] 5.5 -2.450e-15 0 -1.000e+00 -1 4.082e+14 NaN
[19,] 6.0 1.000e+00 1 -7.348e-16 0 -7.348e-16 0
[20,] 6.5 -9.803e-16 0 1.000e+00 1 -1.020e+15 NaN
[21,] 7.0 -1.000e+00 -1 8.573e-16 0 -8.573e-16 0
> options(op)
> help("axes")
No documentation for 'axes' in specified packages and libraries:
you could try '?axes'
> ??axes
```

```
Terminal
Thu Oct 31, 10:12 AM
Help files with alias or concept or title matching 'axes' using regular
expression matching:
ggplot2::render_axes Render panel axes
Aliases: render_axes
labeling::extended An Extension of Wilkinson's Algorithm for
Position Tick Labels on Axes
labeling::extended.figures Generate figures from An Extension of
Wilkinson's Algorithm for Position Tick Labels
on Axes
rgl::axes3d Draw boxes, axes and other text outside the
data
Aliases: axes3d
Type '?PKG::FOO' to inspect entries 'PKG::FOO', or 'TYPE?PKG::FOO' for
entries like 'PKG:FOO-TYPE'.
```

```
Terminal
Thu Oct 31, 10:12 AM
Help files with alias or concept or title matching 'axes' using regular
expression matching:
ggplot2::render_axes Render panel axes
Aliases: render_axes
labeling::extended An Extension of Wilkinson's Algorithm for
Position Tick Labels on Axes
labeling::extended.figures Generate figures from An Extension of
Wilkinson's Algorithm for Position Tick Labels
on Axes
rgl::axes3d Draw boxes, axes and other text outside the
data
Aliases: axes3d
Type '?PKG::FOO' to inspect entries 'PKG::FOO', or 'TYPE?PKG::FOO' for
entries like 'PKG:FOO-TYPE'.
```

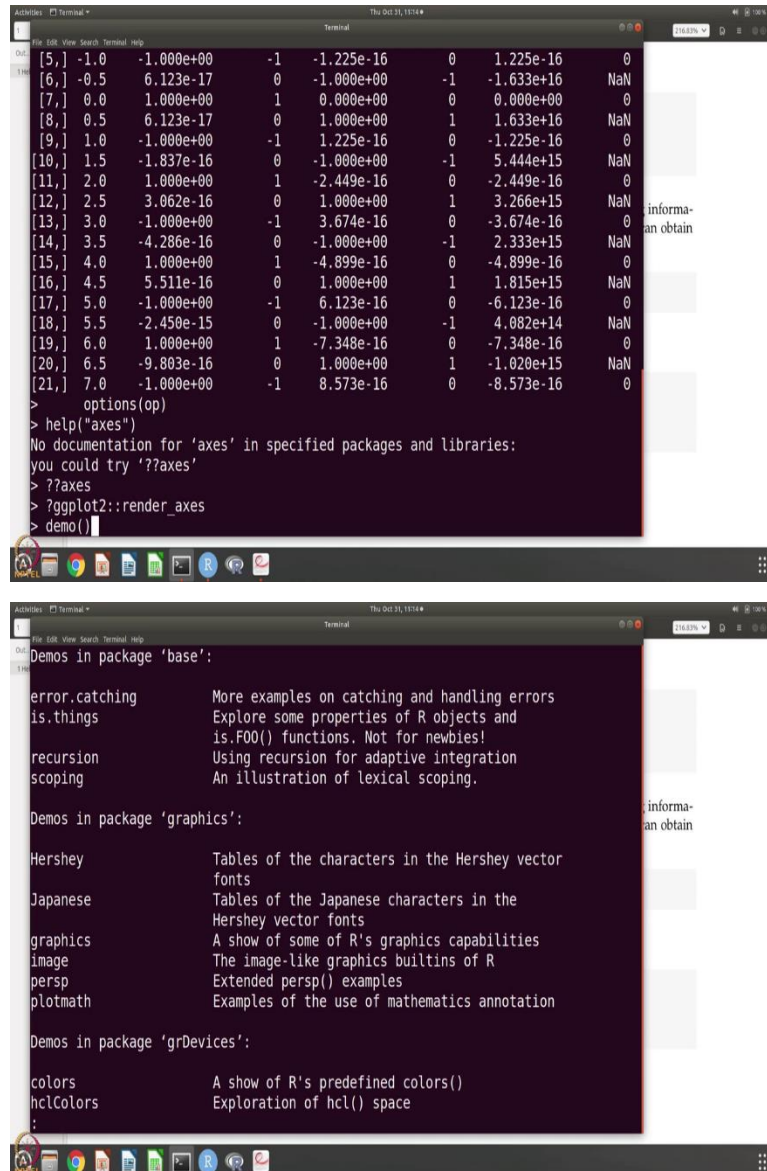


So, what next suppose I want to know how to get the axes information. So, you say axes, now there is no documentation for axes right? So, maybe you have a plot you want to know how the axis y, x are and you thought that there is something like axes, like there was sin, and then it is very helpful. So, it says try axes.

When you do so it actually basically looks for all help files and concept, R title matching axes, it actually gets. Now, if you want to know more about this, for example, let us say GGplot2 render axes is what I want to know more about. Let us copy this and then you put this question mark and paste that. And then you get information on that.

So, if this is another way of getting information from R and you can try to look up some keyword and if the keyword is not a specific keyword, like we did first time sin was a specific keyword. But axes is not appearing but it will still look up the help files and find information wherever this keyword appears. And we will give that information to you. And from that you can choose what you want to explore. So, this is the other way that you can use help. So, that is what we have done.

(Refer Slide Time: 08:14)



```

[5,] -1.0 -1.000e+00 -1 -1.225e-16 0 1.225e-16 0
[6,] -0.5 6.123e-17 0 -1.000e+00 -1 -1.633e+16 NaN
[7,] 0.0 1.000e+00 1 0.000e+00 0 0.000e+00 0
[8,] 0.5 6.123e-17 0 1.000e+00 1 1.633e+16 NaN
[9,] 1.0 -1.000e+00 -1 1.225e-16 0 -1.225e-16 0
[10,] 1.5 -1.837e-16 0 -1.000e+00 -1 5.444e+15 NaN
[11,] 2.0 1.000e+00 1 -2.449e-16 0 -2.449e-16 0
[12,] 2.5 3.062e-16 0 1.000e+00 1 3.266e+15 NaN
[13,] 3.0 -1.000e+00 -1 3.674e-16 0 -3.674e-16 0
[14,] 3.5 -4.286e-16 0 -1.000e+00 -1 2.333e+15 NaN
[15,] 4.0 1.000e+00 1 -4.899e-16 0 -4.899e-16 0
[16,] 4.5 5.511e-16 0 1.000e+00 1 1.815e+15 NaN
[17,] 5.0 -1.000e+00 -1 6.123e-16 0 -6.123e-16 0
[18,] 5.5 -2.450e-15 0 -1.000e+00 -1 4.082e+14 NaN
[19,] 6.0 1.000e+00 1 -7.348e-16 0 -7.348e-16 0
[20,] 6.5 -9.803e-16 0 1.000e+00 1 -1.020e+15 NaN
[21,] 7.0 -1.000e+00 -1 8.573e-16 0 -8.573e-16 0
> options(op)
> help("axes")
No documentation for 'axes' in specified packages and libraries:
you could try '??axes'
> ??axes
> ?ggplot2::render_axes
> demo()
```

```

Demos in package 'base':

error.catching      More examples on catching and handling errors
is.things            Explore some properties of R objects and
                    is.FOO() functions. Not for newbies!
recursion            Using recursion for adaptive integration
scoping              An illustration of lexical scoping.

Demos in package 'graphics':

Hershey              Tables of the characters in the Hershey vector
                    fonts
Japanese             Tables of the Japanese characters in the
                    Hershey vector fonts
graphics             A show of some of R's graphics capabilities
image               The image-like graphics builtins of R
persp                Extended persp() examples
plotmath             Examples of the use of mathematics annotation

Demos in package 'grDevices':

colors               A show of R's predefined colors()
hclColors            Exploration of hcl() space
:
```



```
Activities Terminal Thu Oct 31, 10:04
[6,] -0.5 6.123e-17 0 -1.000e+00 -1 -1.633e+16 NaN
[7,] 0.0 1.000e+00 1 0.000e+00 0 0.000e+00 0
[8,] 0.5 6.123e-17 0 1.000e+00 1 1.633e+16 NaN
[9,] 1.0 -1.000e+00 -1 1.225e-16 0 -1.225e-16 0
[10,] 1.5 -1.837e-16 0 -1.000e+00 -1 5.444e+15 NaN
[11,] 2.0 1.000e+00 1 -2.449e-16 0 -2.449e-16 0
[12,] 2.5 3.062e-16 0 1.000e+00 1 3.266e+15 NaN
[13,] 3.0 -1.000e+00 -1 3.674e-16 0 -3.674e-16 0
[14,] 3.5 -4.286e-16 0 -1.000e+00 -1 2.333e+15 NaN
[15,] 4.0 1.000e+00 1 -4.899e-16 0 -4.899e-16 0
[16,] 4.5 5.511e-16 0 1.000e+00 1 1.815e+15 NaN
[17,] 5.0 -1.000e+00 -1 6.123e-16 0 -6.123e-16 0
[18,] 5.5 -2.450e-15 0 -1.000e+00 -1 4.082e+14 NaN
[19,] 6.0 1.000e+00 1 -7.348e-16 0 -7.348e-16 0
[20,] 6.5 -9.803e-16 0 1.000e+00 1 -1.020e+15 NaN
[21,] 7.0 -1.000e+00 -1 8.573e-16 0 -8.573e-16 0
> options(op
> help("axes")
No documentation for 'axes' in specified packages and libraries:
you could try '??axes'
> ??axes
> ?ggplot2::render_axes
> demo()
> demo(Hershey)
```

```
Activities Terminal Thu Oct 31, 10:04
[12,] 2.5 3.062e-16 0 1.000e+00 1 3.266e+15 NaN
[13,] 3.0 -1.000e+00 -1 3.674e-16 0 -3.674e-16 0
[14,] 3.5 -4.286e-16 0 -1.000e+00 -1 2.333e+15 NaN
[15,] 4.0 1.000e+00 1 -4.899e-16 0 -4.899e-16 0
[16,] 4.5 5.511e-16 0 1.000e+00 1 1.815e+15 NaN
[17,] 5.0 -1.000e+00 -1 6.123e-16 0 -6.123e-16 0
[18,] 5.5 -2.450e-15 0 -1.000e+00 -1 4.082e+14 NaN
[19,] 6.0 1.000e+00 1 -7.348e-16 0 -7.348e-16 0
[20,] 6.5 -9.803e-16 0 1.000e+00 1 -1.020e+15 NaN
[21,] 7.0 -1.000e+00 -1 8.573e-16 0 -8.573e-16 0
> options(op
> help("axes")
No documentation for 'axes' in specified packages and libraries:
you could try '??axes'
> ??axes
> ?ggplot2::render_axes
> demo()
> demo(Hershey)

demo(Hershey)
-----
Type <Return> to start :
```

So, now let us say demo. And then it gives you all possible demos that are available. Let say that I want to try this demo, which was called the Hershey. So, it says return to start. So, it opens graphic device. Enter and then I can go see what it is, let me move it here. And I can keep entering and I will keep getting all this information.

So, this is just to show you the capabilities of R, I mean it is not really something that we are going to use but it. So, as you can see things keep changing. So, now it has come out. So, we have come back to the R prompt. As you could see, when I am executing commands on the console, I have to keep going to these figures or images. And sometimes they are not even seen, I have to locate them first and see.

(Refer Slide Time: 09:37)

The image displays two screenshots of the RStudio environment. The top screenshot shows the R console with the following text:

```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (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

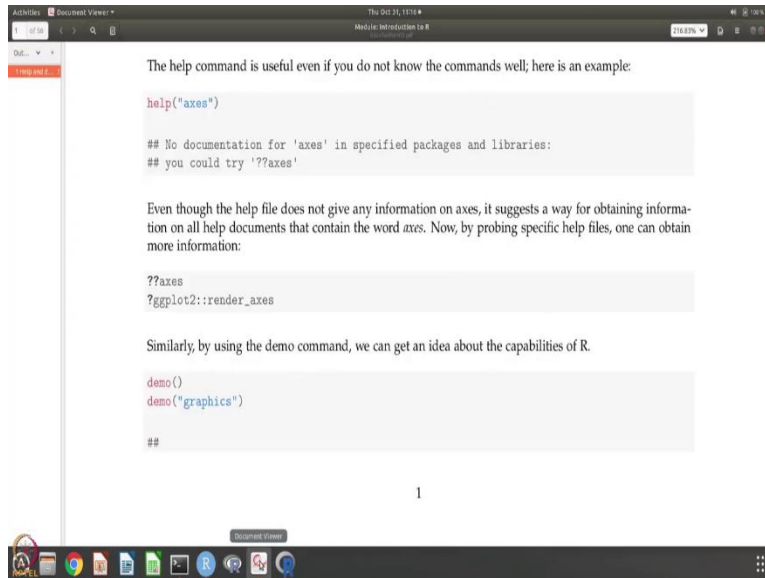
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

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.

> |
```

The right-hand pane shows the "Environment" tab with the text "Environment is empty". Below it, the "Files" pane shows the "Trigonometric Functions" package documentation, including a description and usage instructions.

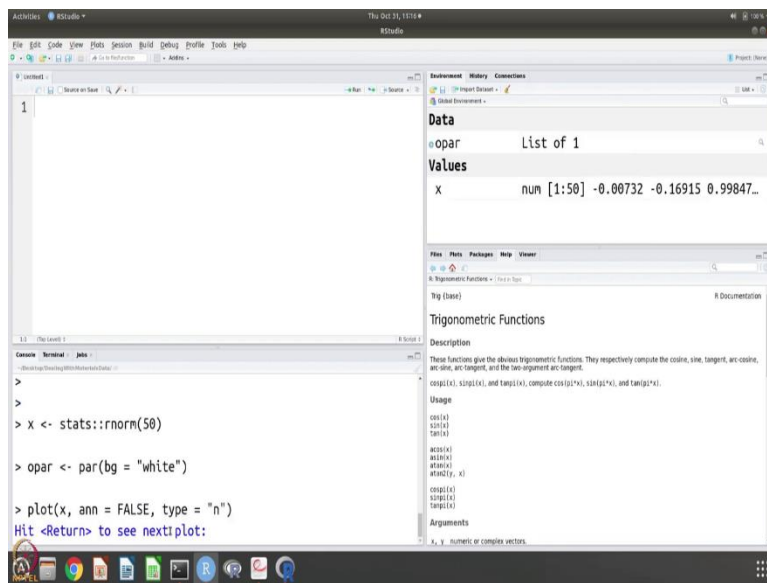
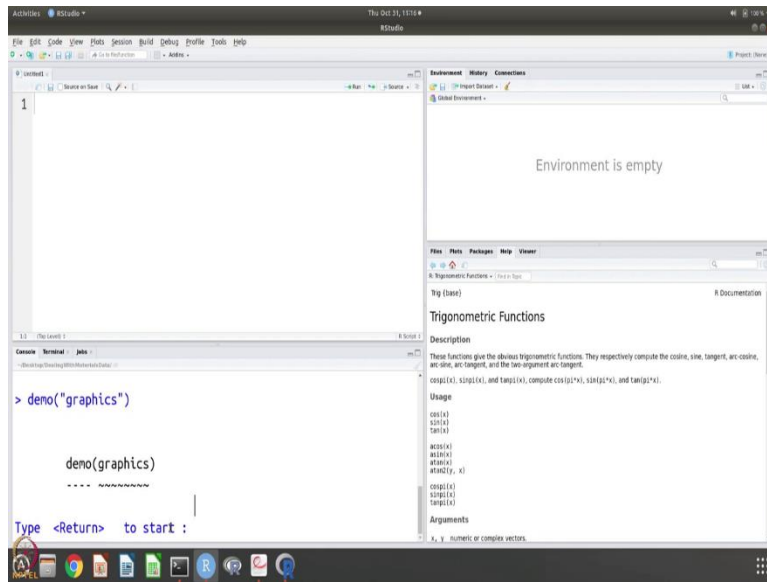
The bottom screenshot shows the same RStudio interface. The console now displays the command `> demo("graphics")` and a small window titled "1" is visible in the top-left corner.



Now that is where the integrated development environment comes very handy. And this is the IDE. Now, it actually consists of 4 panes. So let me let me do this, let say that I want to write a new R script or something. So, there is this editor, which you can use to write the R script. And here is the console because this is exactly like the R console that we got R version 3.6.1 action of the tools etc, and see the help files are shown here, right.

Help files and if you make some plots, plots will be shown here and the packages will be shown here and this is information about the history and the environment and things like that. So, the R studio pane has consists of four panes and the editor, the console, and you can even get a terminal here, for example and then you have all this help files and the environment information. Let us use the demo here and so there you go here you will see that let us say that the demo that I want to get so, so I mentioned some 4, 5 demos, right?

(Refer Slide Time: 10:53)



Activities RStudio Thu Oct 31, 11:04 RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Data

- o par List of 1

Values

- x num [1:50] -0.00732 -0.16915 0.99847...

File Plots Packages Help Viewer

Simple Use of Color in a Plot

```

> ## 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)
Hit <Return> to see next plot:

```

Activities RStudio Thu Oct 31, 11:04 RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Data

- o par List of 1

Values

- pie.sales Named num [1:6] 0.12 0.3 0.26 0.16 0...
- x num [1:50] -0.00732 -0.16915 0.99847...

File Plots Packages Help Viewer

A Sample Color Wheel

```

> names(pie.sales) <- c("Blueberry", "Cherry",
+ "Apple", "Boston Cream", "Other", "Vanil
+ la Cream")
>
> pie(pie.sales,
+ col = c("purple", "violetred1", "green3", "cornsilk", "cyan", "white"))
Hit <Return> to see next plot:

```

Activities RStudio Thu Oct 31, 11:17 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

```

opar      List of 1
Values
g         Factor w/ 10 levels "1","2","3","4",...
n         10
pie.sales Named num [1:6] 0.12 0.3 0.26 0.16 0...
x         num [1:1000] 0.521 1.255 0.684 1.381...

```

File Plots Packages Help Viewer

January Pie Sales

Cherry
Raspberry
Vanilla Cream
Other
Boston Cream
Apple

(Built by Mike at Newt Ltd)

```

> n <- 10

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

> x <- rnorm(n*100) + sqrt(as.numeric(g))

> boxplot(split(x,g), col="lavender", notch=TRUE)
Hit <Return> to see next plot: |

```

Activities RStudio Thu Oct 31, 11:17 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

```

n         100
pie.sales Named num [1:6] 0.12 0.3 0.26 0.16 0...
x         num [1:101] 0 1.6 3.77 4.32 4.38 ...
xx        int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
y         num [1:101] 0 1.74 2.38 2.22 2.77 ...
yy        num [1:202] 0 1.6 3.77 4.32 4.38 ...

```

File Plots Packages Help Viewer

Notched Boxplots

Distance

Group

```

> y <- c(0,cunsun(rnorm(n)))

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

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

> plot(xx, yy, type="n", xlab="Time", ylab="Distance")
Hit <Return> to see next plot: |

```

Activities RStudio Thu Oct 31, 11:17 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

```

n          num [1:6] 0.12 0.3 0.26 0.16 0...
pie.sales Named num [1:6] 0.12 0.3 0.26 0.16 0...
x          num [1:12] 0 0.4 0.86 0.85 0.69 0.48...
xx         int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
y          num [1:101] 0 1.74 2.38 2.22 2.77 ...
yy         num [1:202] 0 1.6 3.77 4.32 4.38 ...

```

Distance Between Brownian Motions

```

> x <- c(0.00, 0.40, 0.86, 0.85, 0.69, 0.48, 0.54, 1.09, 1.11,
1.73, 2.05, 2.02)

> par(bg="lightgray")

> plot(x, type="n", axes=FALSE, ann=FALSE)
Hit <Return> to see next plot:

```

Activities RStudio Thu Oct 31, 11:17 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

```

pie.sales Named num [1:6] 0.12 0.3 0.26 0.16 0...
usr       num [1:4] 0.56 12.44 -0.082 2.132
x         num [1:1000] -0.647 1.453 0.162 1.87...
xx        int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
y         num [1:101] 0 1.74 2.38 2.22 2.77 ...
yy        num [1:202] 0 1.6 3.77 4.32 4.38 ...

```

The Level of Interest In R

```

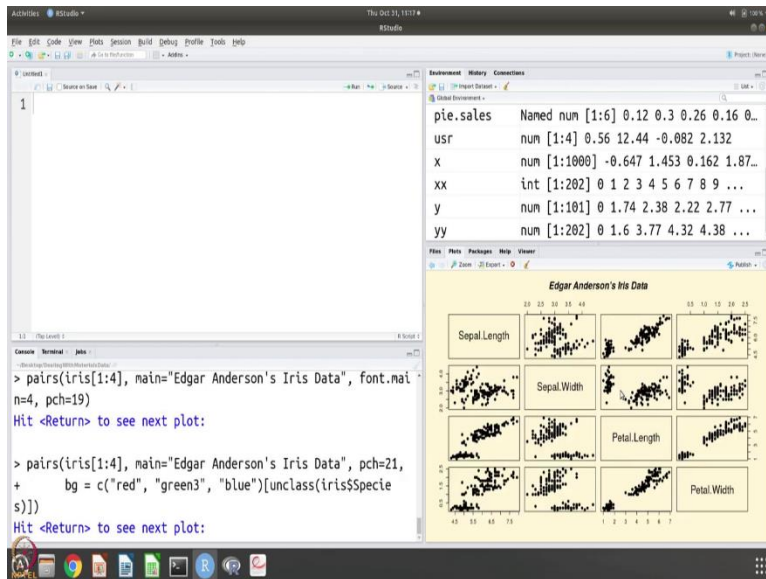
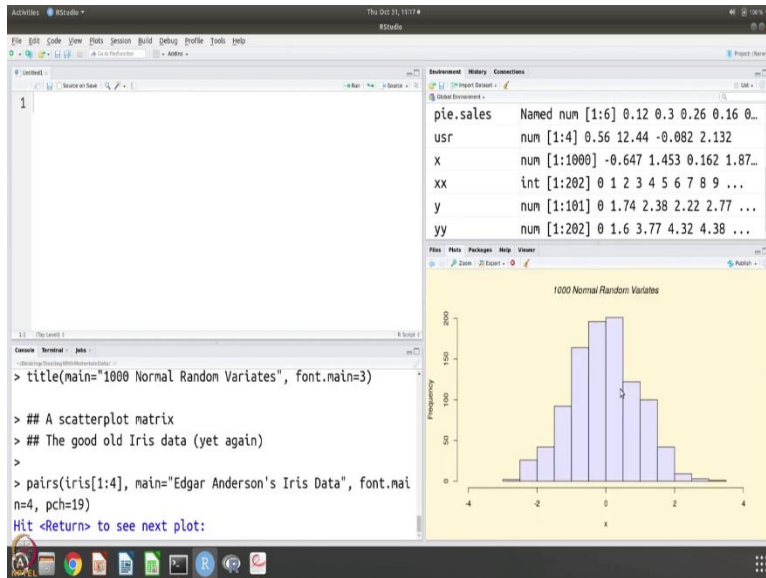
> ## main title without changing the other annotation.

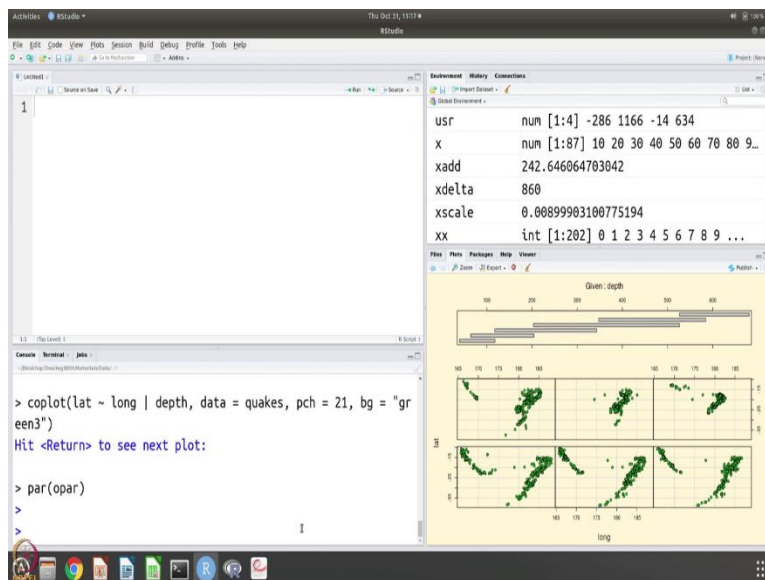
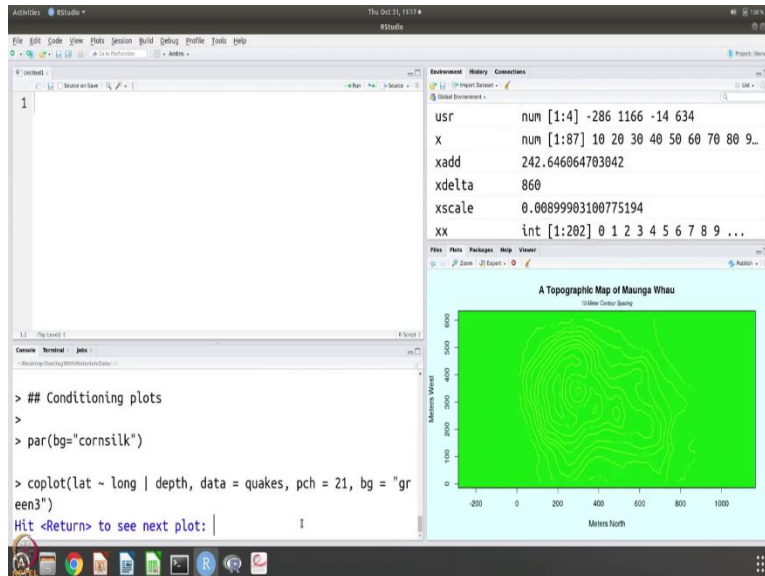
> par(bg="cornsilk")

> x <- rnorm(1000)

> hist(x, xlim=range(-4, 4), col="lavender", main="")
Hit <Return> to see next plot:

```





So, demographics is what first we want to do. Let us do demographics. Now, you can see that, see the plot shows here. So, I can keep skipping through. So, this is a sample color wheel, and see the, the sales as a pie chart and, and the notch boxplots and the time versus distance in a Brownian motion and so how R has been becoming popular, so lots of information, right.

So, this is some normal random variable, so it is a distribution box, the histogram plot, and so there is some data which can be plotted, these data are already built in into R so these are used as example data for learning. And we will, we might occasionally use but most of the times, we will make our

own data because we want materials data and use them. And so you can of course, get nice color plots. And you can get these kind of contour plots. And you can give the information in many different ways. So, it has come to the prompt. So, we have completed this demo.

(Refer Slide Time: 12:18)

The screenshot shows the RStudio interface. The console window contains the following R code:

```
> ## main title without changing the other annotation.  
>  
> par(bg="cornsilk")  
> x <- rnorm(1000)  
> hist(x, xlim=range(-4, 4, x), col="lavender", main="")  
Hit <Return> to see next plot: |
```

The Environment pane on the right shows the following objects:

Object	Class	Attributes
pie.sales	Named num	[1:6] 0.12 0.3 0.26 0.16 0...
usr	num	[1:4] 0.56 12.44 -0.082 2.132
x	num	[1:1000] -0.647 1.453 0.162 1.87...
xx	int	[1:202] 0 1 2 3 4 5 6 7 8 9 ...
y	num	[1:101] 0 1.74 2.38 2.22 2.77 ...
yy	num	[1:202] 0 1.6 3.77 4.32 4.38 ...

The plot window displays a line graph titled "The Level of Interest in R" for the year 1996. The x-axis represents months from Jan to Dec, and the y-axis represents interest levels from 0.0 to 2.0. The plot area has a yellow background. The data points are connected by a line with open circles at each month.

The screenshot shows a document viewer displaying the help text for the 'axes' command in R. The text includes the following sections:

The help command is useful even if you do not know the commands well; here is an example:

```
help("axes")  
  
## No documentation for 'axes' in specified packages and libraries:  
## you could try '??axes'
```

Even though the help file does not give any information on axes, it suggests a way for obtaining information on all help documents that contain the word axes. Now, by probing specific help files, one can obtain more information:

```
??axes  
?ggplot2::render_axes
```

Similarly, by using the demo command, we can get an idea about the capabilities of R.

```
demo()  
demo("graphics")  
  
##
```

The page number 1 is visible at the bottom of the document.


```
## demo(graphics)
## ----
##
## > # Copyright (C) 1997-2009 The R Core Team
## >
## > require(datasets)
##
## > require(grDevices); require(graphics)
##
## > ## Here is some code which illustrates some of the differences between
## > ## R and S graphics capabilities. Note that colors are generally specified
## > ## by a character string name (taken from the X11 rgb.txt file) and that line
## > ## textures are given similarly. The parameter "bg" sets the background
## > ## parameter for the plot and there is also an "fg" parameter which sets
## > ## the foreground color.
## >
## >
## > x <- stats:rnorm(50)
##
```

The similar fashion, you can have several demos. And actually, this is one of the biggest files in the notes so far 56 pages, because it gives you all that we have seen through now. And this is what we have run through up to this.

(Refer Slide Time: 13:09)

The screenshot shows the RStudio environment. The console window contains the following R code:

```
(op)
+ plot(1:n,1:n, type="n", axes=FALSE)
+ text(col(cm), rev(row(cm)), cm, col = cl, cex=cex, srt=
+ srt)
+ }

> showCols1()
Hit <Return> to see next plot:
```

The Environment pane on the right shows the following objects:

xdelta	860
xscale	0.00899903100775194
xx	int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
y	num [1:27] -12.57 -11.6 -10.63 -9.67...
y.at	num [1:6] 100 200 300 400 500 600
yadd	0

The plot window displays a heatmap with a color scale ranging from -0.5 to 1.0. The plot shows a grid of colored cells, with a prominent red cell in the center, indicating a high value. The color scale is shown on the right side of the plot.

Activities RStudio Thu Oct 31, 11:19 RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

x.at	num [1:8]	100 200 300 400 500 600 70...
xadd		242.646064703042
xdelta		860
xscale		0.00899903100775194
xx	int [1:202]	0 1 2 3 4 5 6 7 8 9 ...
y	num [1:27]	-12.57 -11.6 -10.63 -9.67...

```

1

```

Console Terminal Jobs

```

Hit <Return> to see next plot:
> demo("colors")

demo(colors)
...
Type <Return> to start :

```

Activities RStudio Thu Oct 31, 11:19 RStudio

Environment History Connections

x.at	num [1:8]	100 200 300 400 500 600 70...
xadd		242.646064703042
xdelta		860
xscale		0.00899903100775194
xx	int [1:202]	0 1 2 3 4 5 6 7 8 9 ...
y	num [1:27]	-12.57 -11.6 -10.63 -9.67...

```

1

```

Console Terminal Jobs

```

fllid.> filled.contour(cos(r^2)*exp(-r/(2*pi))), axes = FALSE)
Hit <Return> to see next plot:

fllid.> ## rather, the key *should* be labeled:
fllid.> filled.contour(cos(r^2)*exp(-r/(2*pi))), frame.plot = F
FALSE,
fllid.> plot.axes = {}
Hit <Return> to see next plot:

```

Activities RStudio Thu Oct 31, 11:19 RStudio

Environment History Connections

x.at	num [1:8]	100 200 300 400 500 600 70...
xadd		242.646064703042
xdelta		860
xscale		0.00899903100775194
xx	int [1:202]	0 1 2 3 4 5 6 7 8 9 ...
y	num [1:27]	-12.57 -11.6 -10.63 -9.67...

```

1

```

Console Terminal Jobs

```

fllid.> ## Persian Rug Art:
fllid.> x <- y <- seq(-4*pi, 4*pi, len = 27)

fllid.> r <- sqrt(outer(x^2, y^2, "+"))

fllid.> filled.contour(cos(r^2)*exp(-r/(2*pi))), axes = FALSE)
Hit <Return> to see next plot:

```

Activities RStudio Thu Dec 31, 11:19 RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

Data

- a 400 obs. of 2 variables
- b int [1:20, 1:20] 2 3 4 5 6 7 8 9 10...
- opar List of 1

Values

- g Factor w/ 10 levels "1","2","3","4",...

Console

```
fllid.> a <- expand.grid(1:20, 1:20)

fllid.> b <- matrix(a[,1] + a[,2], 20)

fllid.> filled.contour(x = 1:20, y = 1:20, z = b,
fllid.> plot.axes = { axis(1); axis(2); points
(10, 10) })
Hit <Return> to see next plot: |
```

The Topography of Maunga Whau

Activities RStudio Thu Dec 31, 11:19 RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

- x.at num [1:8] 100 200 300 400 500 600 70...
- xadd 242.646064703042
- xdelta 860
- xscale 0.00899903100775194
- xx int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
- y num [1:61] 10 20 30 40 50 60 70 80 9...

Console

```
a Whau",
fllid.> xlab = "Meters North", ylab = "Meters West"),
fllid.> plot.axes = { axis(1, seq(100, 800, by = 100))
fllid.> axis(2, seq(100, 600, by = 100)) },
fllid.> key.title = title(main = "Height\n(meters)"),
fllid.> key.axes = axis(4, seq(90, 190, by = 10)) # mayb
e also asp = 1
Hit <Return> to see next plot: |
```

Activities RStudio Thu Dec 31, 11:19 RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

- xadd 242.646064703042
- xdelta 860
- xscale 0.00899903100775194
- xx int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
- y num [1:61] 10 20 30 40 50 60 70 80 9...
- y.at num [1:6] 100 200 300 400 500 600

Console

```
> ## Filled Contours are even nicer sometimes :
> example(filled.contour)

fllid.> require("grDevices") # for colours

fllid.> filled.contour(volcano, asp = 1) # simple
Hit <Return> to see next plot: |
```

Maunga Whau Volcano
colgray(100:200:200)

Activities RStudio Thu Oct 31, 11:19 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

xadd	242.646064703042
xdelta	860
xscale	0.00899903100775194
xx	int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
y	num [1:61] 10 20 30 40 50 60 70 80 9...
y.at	num [1:6] 100 200 300 400 500 600

```

> title(main="Maunga Whau Volcano", sub = "col=terrain.colors
(100)", font.main=4)
>
# Using Heat Colors
> image(x, y, volcano, col=heat.colors(100), axes=FALSE)
Hit <Return> to see next plot:

```

Activities RStudio Thu Oct 31, 11:19 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

xadd	242.646064703042
xdelta	860
xscale	0.00899903100775194
xx	int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
y	num [1:61] 10 20 30 40 50 60 70 80 9...
y.at	num [1:6] 100 200 300 400 500 600

```

> title(main="Maunga Whau Volcano", sub = "col=terrain.colors
(100)", font.main=4)
>
# Using Heat Colors
> image(x, y, volcano, col=heat.colors(100), axes=FALSE)
Hit <Return> to see next plot:

```

Activities RStudio Thu Oct 31, 11:19 RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

usr	num [1:4]	-286 1166 -14 634
x	num [1:87]	10 20 30 40 50 60 70 80 9...
xadd		242.646064703042
xdelta		860
xscale		0.00899903100775194
xx	int [1:202]	0 1 2 3 4 5 6 7 8 9 ...

Files Plots Packages Help Viewer

Given: depth

Console Terminal Jobs

```

> deno("image")

deno(image)
.....
Type <Return> to start :

```

Activities RStudio Thu Oct 31, 11:19 RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

yscale		0.0057527777777778
yy	num [1:202]	0 1.6 3.77 4.32 4.38 ...

Functions

plotCol	function (col, nrow = 1, ncol = cei...
showCols1	function (bg = "gray", cex = 0.75, ...
showCols2	function (bg = "grey", cex = 0.75, ...

Files Plots Packages Help Viewer

Console Terminal Jobs

```

> ## A Chocolate Bar of colors:
> plotCol(c("#CC8C3C", paste0("chocolate", 2:4),
+         paste0("darkorange", c("", 1:2)), paste0("darkgolde
+         "nrod", 1:2),
+         "orange", "orange1", "sandybrown", "tan1", "tan
+         2"),
+         nrow=2)
Hit <Return> to see next plot:

```

Activities RStudio Thu Oct 31, 11:19 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

yy num [1:202] 0 1.6 3.77 4.32 4.38 ...

Functions

nearRcolor function (rgb, cSpace = c("hsv", "r...
 plotCol function (col, nrow = 1, ncol = cel...
 showCols1 function (bg = "gray", cex = 0.75, ...
 showCols2 function (bg = "grey", cex = 0.75, ...

File Plots Packages Help Viewer

Zoom Export

13 The Console 6 Script 1

```

> plotCol(nearRcolor("tomato", "hsv", dist=.12), nrow=3)
Hit <Return> to see next plot:

> plotCol(nearRcolor("tomato", "Luv", dist= 25), nrow=3)
Hit <Return> to see next plot:

> plotCol(nearRcolor("tomato", "Lab", dist= 18), nrow=3)
Hit <Return> to see next plot:

```

Activities RStudio Thu Oct 31, 11:19 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

yy num [1:202] 0 1.6 3.77 4.32 4.38 ...

Functions

nearRcolor function (rgb, cSpace = c("hsv", "r...
 plotCol function (col, nrow = 1, ncol = cel...
 showCols1 function (bg = "gray", cex = 0.75, ...
 showCols2 function (bg = "grey", cex = 0.75, ...

File Plots Packages Help Viewer

Zoom Export

13 The Console 6 Script 1

```

> plotCol(nearRcolor("tomato", "rgb", dist= 50), nrow=3)
Hit <Return> to see next plot:

> plotCol(nearRcolor("tomato", "hsv", dist=.12), nrow=3)
Hit <Return> to see next plot:

> plotCol(nearRcolor("tomato", "Luv", dist= 25), nrow=3)
Hit <Return> to see next plot:

```


Activities RStudio Thu Oct 31, 11:19 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

yy num [1:202] 0 1.6 3.77 4.32 4.38 ...

Functions

nearRcolor function (rgb, cSpace = c("hsv", "r...
 plotCol function (col, nrow = 1, ncol = ce...
 showCols1 function (bg = "gray", cex = 0.75, ...
 showCols2 function (bg = "grey", cex = 0.75, ...

File Plots Packages Help Viewer

Zoom Export

Console Terminal Jobs

```
> plotCol(nearRcolor("deepskyblue", dist=.1))
Hit <Return> to see next plot:

> plotCol(nearRcolor("tomato", "rgb", dist= 50), nrow=3)
Hit <Return> to see next plot:

> plotCol(nearRcolor("tomato", "hsv", dist=.12), nrow=3)
Hit <Return> to see next plot: |
```

Activities RStudio Thu Oct 31, 11:19 AM RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

yy num [1:202] 0 1.6 3.77 4.32 4.38 ...

Functions

nearRcolor function (rgb, cSpace = c("hsv", "r...
 plotCol function (col, nrow = 1, ncol = ce...
 showCols1 function (bg = "gray", cex = 0.75, ...
 showCols2 function (bg = "grey", cex = 0.75, ...

File Plots Packages Help Viewer

Zoom Export

Console Terminal Jobs

```
Hit <Return> to see next plot:

> plotCol(nearRcolor("deepskyblue", dist=.1))
Hit <Return> to see next plot:

> plotCol(nearRcolor("tomato", "rgb", dist= 50), nrow=3)
Hit <Return> to see next plot: |
```

Activities RStudio Thu Oct 31, 11:19 AM RStudio

File Edit Code View Data Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

yy num [1:202] 0 1.6 3.77 4.32 4.38 ...

Functions

nearRcolor function (rgb, cspace = c("hsv", "r...
 plotCol function (col, nrow = 1, ncol = cel...
 showCols1 function (bg = "gray", cex = 0.75, ...
 showCols2 function (bg = "grey", cex = 0.75, ...

File Edit Packages Help Viewer

Zoom Font

Console Terminal Jobs

```
> ## Now, consider choosing a color by looking in the
> ## neighborhood of one you know :
>
> plotCol(nearRcolor("deepskyblue", "rgb", dist=50))
Hit <Return> to see next plot:
>
> plotCol(nearRcolor("deepskyblue", "rgb", dist=.1))
Hit <Return> to see next plot: |
```

Activities RStudio Thu Oct 31, 11:19 AM RStudio

File Edit Code View Data Session Build Debug Profile Tools Help

Environment History Connections

Global Environment

yy num [1:202] 0 1.6 3.77 4.32 4.38 ...

Functions

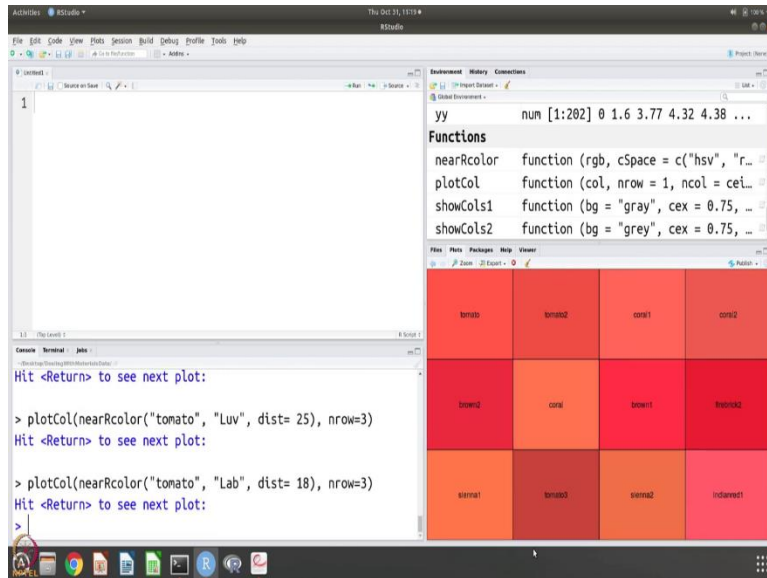
nearRcolor function (rgb, cspace = c("hsv", "r...
 plotCol function (col, nrow = 1, ncol = cel...
 showCols1 function (bg = "gray", cex = 0.75, ...
 showCols2 function (bg = "grey", cex = 0.75, ...

File Edit Packages Help Viewer

Zoom Font

Console Terminal Jobs

```
0.0867
"darkslategray"
> ## Now, consider choosing a color by looking in the
> ## neighborhood of one you know :
>
> plotCol(nearRcolor("deepskyblue", "rgb", dist=50))
Hit <Return> to see next plot:
```

So, that is the next demo. So, these I think R from demo, colors, let us go what are the commands. This is the image. So, graphics and then demo image. Sorry, let me go to the studio, right. So, this is the demo image and we start and then we start seeing this, right. So, that is what you saw in the tutorial and demo, I think colors. So, this is the colors demo, and it shows you all possible color schemes that you can get and all possible colors that you can get.

(Refer Slide Time: 13:50)

```
> code <- c(252,254,256,262,269,275,278,281,284,745,746,750,768,796,802,
+          804,807,809,814,828,830,834,840,844)
> draw.title("Raw Hershey Escape Sequences", i=0, nr, nc)
> for (i in 1:75)
+   draw.vf.cell(tf, fi, paste("\\#H",formatC(code[i],wid=4,flag="0"),sep=""),
+             i-1, nr)
> make.table(nr, nc)
Hit <Return> to see next plot:
> code <- c(845:847,850,856,860,874,899,909,2296:2299,2318:2332,2367:2382,
+          4014,4109)
> draw.title("More Raw Hershey Escape Sequences", i=0, nr, nc)
> for (i in 1:73)
+   draw.vf.cell(tf, fi, paste("\\#H",formatC(code[i],wid=4,flag="0"),sep=""),
+             i-1, nr)
> par(oldpar)
>
```

```
1
> y <- seq(-1, 1, length.out=50)
> z <- apply(as.matrix(expand.grid(x, y)), 1, function(x) f(c
(x, 0)))
> contour(x, y, matrix(log10(z), 50, 50))
Hit <Return> to see next plot:
```

Variable	Value
y.at	num [1:6] 100 200 300 400 500 600
yadd	0
ydelta	600
yscale	0.005752777777777778
yy	num [1:202] 0 1.6 3.77 4.32 4.38 ...
z	num [1:2500] 1423 1383 1341 1297 125...

Activities RStudio Thu Dec 31, 11:19 RStudio

File Edit Code View Data Session Build Debug Profile Tools Help

Environment History Connections

yy num [1:202] 0 1.6 3.77 4.32 4.38 ...

Functions

nearRcolor function (rgb, cSpace = c("hsv", "F...
 plotCol function (col, nrow = 1, ncol = ce...
 showCols1 function (bg = "gray", cex = 0.75, ...
 showCols2 function (bg = "grey", cex = 0.75, ...

File Edit Packages Help Viewer

tomato tomato2 coral coral2
 tomato2 coral tomato1 tomato2
 tomato1 tomato3 tomato2 indiana1

Console Terminal Jobs

```
> deno("nln")
```

deno(nln)

 |

Type <Return> to start :

Activities RStudio Thu Dec 31, 11:19 RStudio

File Edit Code View Data Session Build Debug Profile Tools Help

Environment History Connections

yy num [1:202] 0 1.6 3.77 4.32 4.38 ...

Functions

nearRcolor function (rgb, cSpace = c("hsv", "F...
 plotCol function (col, nrow = 1, ncol = ce...
 showCols1 function (bg = "gray", cex = 0.75, ...
 showCols2 function (bg = "grey", cex = 0.75, ...

File Edit Packages Help Viewer

tomato tomato2 coral coral2
 tomato2 coral tomato1 tomato2
 tomato1 tomato3 tomato2 indiana1

Console Terminal Jobs

```
> plotCol(nearRcolor("tomato", "Luv", dist= 25), nrow=3)
Hit <Return> to see next plot:
> plotCol(nearRcolor("tomato", "Lab", dist= 18), nrow=3)
Hit <Return> to see next plot:
> deno("nln")
```

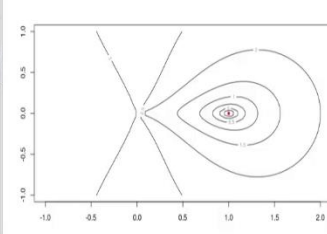
Activities RStudio Thu Dec 31, 11:20 RStudio

File Edit Code View Data Session Build Debug Profile Tools Help

Environment History Connections

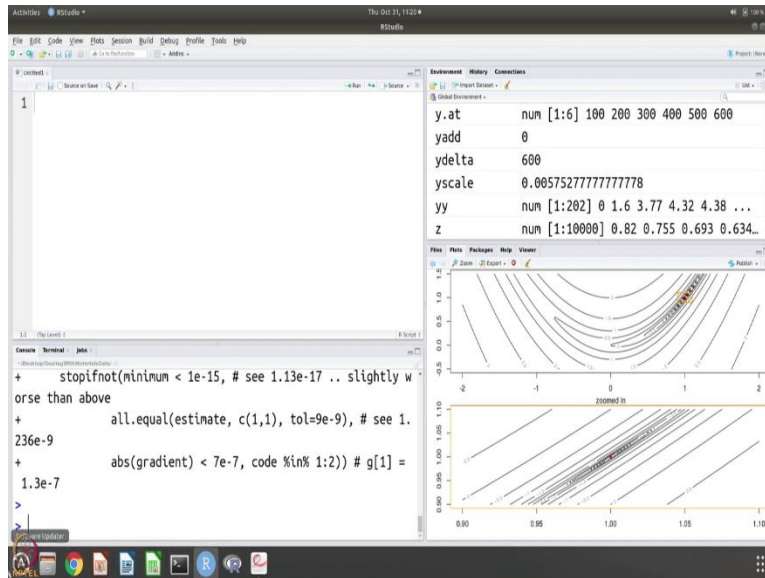
y.at num [1:6] 100 200 300 400 500 600
 yadd 0
 ydelta 600
 yscale 0.005752777777777778
 yy num [1:202] 0 1.6 3.77 4.32 4.38 ...
 z num [1:10000] 2034 1892 1759 1632 15...

File Edit Packages Help Viewer



Console Terminal Jobs

```
> y <- seq(-0.5, 1.5, length.out=100)
> z <- fx(expand.grid(x, y))
> op <- par(mfrow = c(2,1), mar = 0.1 + c(3,3,0,0))
> contour(x, y, matrix(log10(z), length(x)))
Hit <Return> to see next plot:
```



So this, these are the demos that are available. I want you to explore I have given a few more demos here in the in the documentation just for you to explore. So, this is a demo called nlm. So, let us do this, yeah. So, this is the nlm demo. So, and the R has come out of the nlm. So, either in the R console that is here, R in the R studio, you can use both help and the demos for example, if you ask for help here and so, as you can see, R studio also has this advantage.

For example, do you want to have help for tan, or tan hyperbolic or tan pi. So, you can get these information for example, let us say I want help for tan pi, hyperbolic, so it gives you hyperbolic function so cos, sinh, tanh, acosh, asinh, atanh, etc., and again, the x is numerical or complex vector and there are these other information. And, so it is very helpful sometimes to use R studio because everything is in the same screen and you can switch between one or the other.

And for example, you can even write a small script for example here, and then run the script here and the look at the result here in the plot, and help keep track of the information, the history etc. Here on this pane. So, this way, using the help and demo is very useful and it gives you a feel and it also familiarizes you with the R and R console and R studio.

So, I strongly recommend that you explore all the demos that are available and they will also tell you about your installation if it is complete and if you are able to see all the things that you should see. So, it is a it is a good exercise to begin with. Thank you.