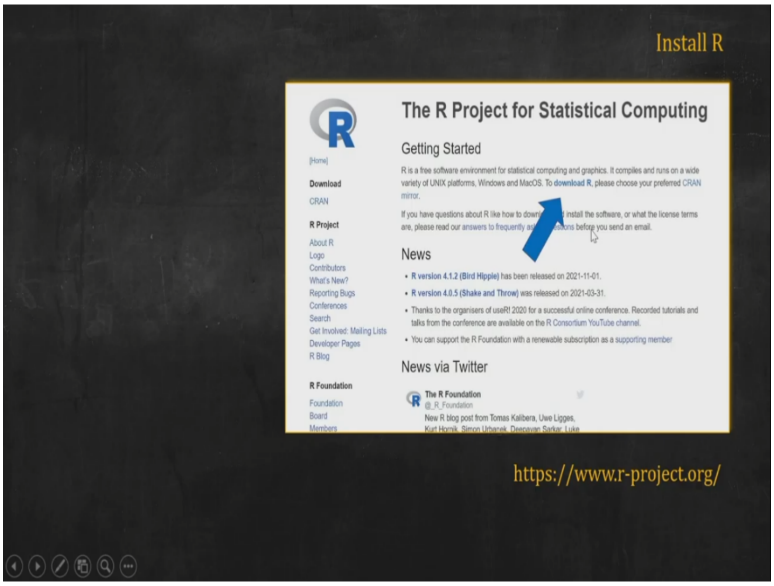
Data Analysis for Biologists
Professor Biplab Bose
Department of Biosciences and Bioengineering
Mehta Family School of Data Sciences and Artificial Intelligence
Indian Institute of Technology Guwahati
Lecture 14
Getting Ready with R

Hello everyone. In this course on Data Analysis for Biologist, we will use R to perform data analysis. So, in this video I will introduce you to R. How to install it and how to start working with that. You must be knowing R is a free open source language and environment for performing statistical computing. It is developed and maintained by an R core group and R foundation.

And at the same time, hundreds and thousands of people like you and me contribute to its development by, creating new packages and tools for R. It is part of new project and it is freely available for you to download and start using. It is not like other tools like Excel or Sigma Plot or SPSS those type of software where you have symbols.

So, you click and then you do some work. So, it is not GUI based. It is a command line based environment and language, so that is why it has a slightly difficult learning curve. Following this video, you will be able to learn to use R, the basic those and will be used, you will be able to use R for your data analysis works. So, let us start with first R. You have to download R.

(Refer Slide Time: 02:02)



Install R

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN mirror.

If you have questions about R like how to downl[illegible] install the software, or what the license terms are, please read our answers to frequently as[illegible]estions before you send an email.

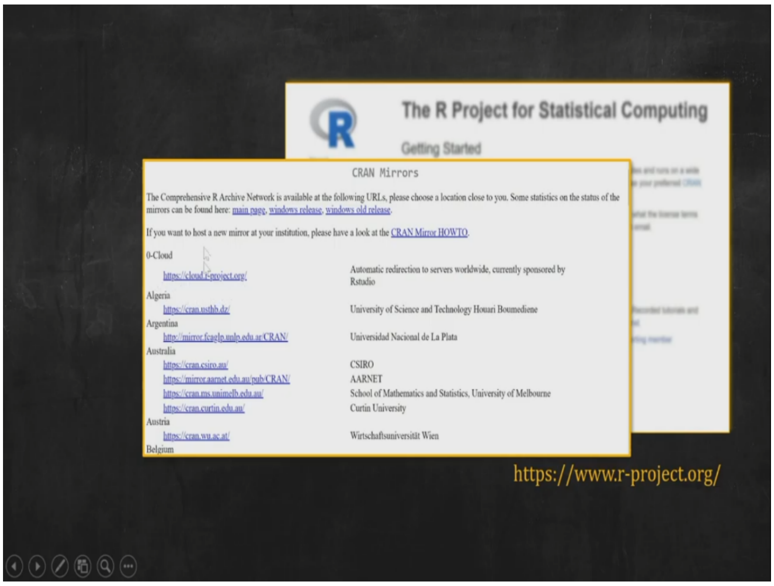
News

- R version 4.1.2 (Bird Hippie) has been released on 2021-11-01.
- R version 4.0.5 (Shake and Throw) was released on 2021-03-31.
- Thanks to the organisers of useR! 2020 for a successful online conference. Recorded tutorials and talks from the conference are available on the R Consortium YouTube channel.
- You can support the R Foundation with a renewable subscription as a supporting member

News via Twitter

The R Foundation
@_R_Foundation
New R blog post from Tomas Kalibera, Uwe Ligges, Kurt Hornik, Simon Urbanek, Deepayan Sarkar, Luke

https://www.r-project.org/



The R Project for Statistical Computing

Getting Started

CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.

If you want to host a new mirror at your institution, please have a look at the CRAN Mirror HOWTO.

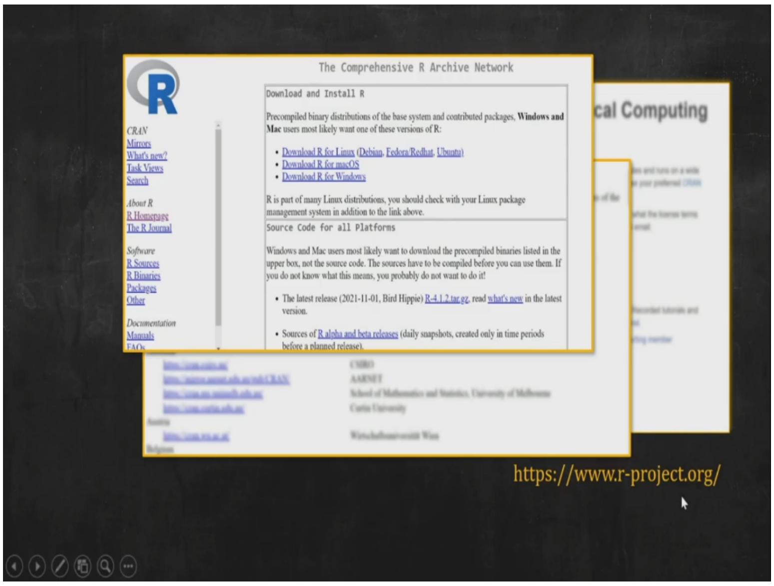
0-Cloud	
https://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio
Algeria	
https://cran.usthb.dz/	University of Science and Technology Houari Boumediene
Argentina	
http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	
https://cran.csiro.au/	CSIRO
https://mirror.aarnet.edu.au/pub/CRAN/	AARNET
https://cran.ms.unimelb.edu.au/	School of Mathematics and Statistics, University of Melbourne
https://cran.curtin.edu.au/	Curtin University
Austria	
https://cran.wu.ac.at/	Wirtschaftsuniversität Wien
Belgium	

https://www.r-project.org/

To download R, you should visit this R website. The link is given here and that is r-project.org Go there. In that page you have lots of information about R. And you go to the download page. Click on the download R text and that will take you to some sort of mirror sites.

R has multiple mirror site and you can download them from any of the mirror site. Maybe you can choose a mirror site in India. But, if you are confused about that you can choose the first one that is the zero cloud. And it will automatically redirect you to the nearest mirror and the best mirror for you.

(Refer Slide Time: 02:43)



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

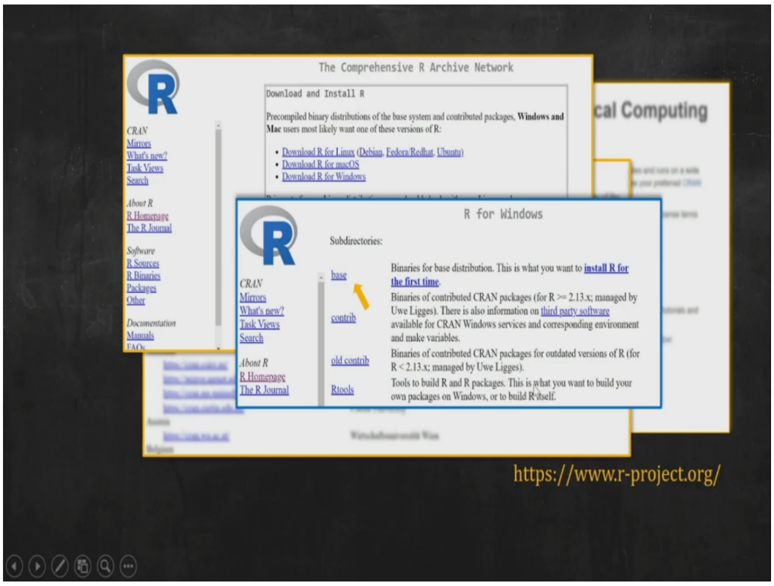
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2021-11-01, Bird Hippie) R-4.1.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).

https://www.r-project.org/


The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

https://www.r-project.org/

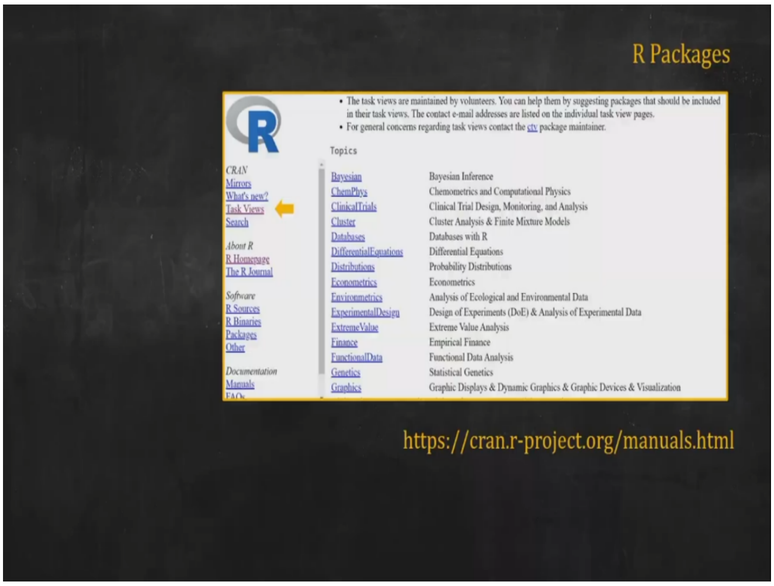
So, if you once you go there then you have multiple option. Now, remember, R is compatible with almost all the operating system Linux, Mac and Windows. So, most of us use windows here. So, you have to click on this download R for windows, so that will take you to the windows page for download.

And there are multiple options. So, choose, for most of us, to start working with this base package is good enough. So, please select click that and then it will download the file. You save that file somewhere and then rest of the part is self-explanatory.

You have to double click and run that file and it will get installed. So, once you have got installed, you have to understand that what you have installed is just the base R. The default R. It can do lots of thing. R in itself is a very powerful and versatile language and environment. So, when you download the base package, it itself comes with lots of strength. But, apart from that, hundreds of people over the years has developed different packages, different programs which are readily available for you to use for particular type of work.

So, those all of those may not come when you install the base thing, the default R. You have to install them, as you require to use. So, it makes sense because suppose I do not use analyze the financial data then why should I load my computer by installing the packages and software, R tools related to some financial data analysis. So, these packages are kept separate.

(Refer Slide Time: 04:29)



R Packages

- The task views are maintained by volunteers. You can help them by suggesting packages that should be included in their task views. The contact e-mail addresses are listed on the individual task view pages.
- For general concerns regarding task views contact the ctv package maintainer.

Topics

Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Databases	Databases with R
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
ExtremeValue	Extreme Value Analysis
Finance	Empirical Finance
FunctionalData	Functional Data Analysis
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

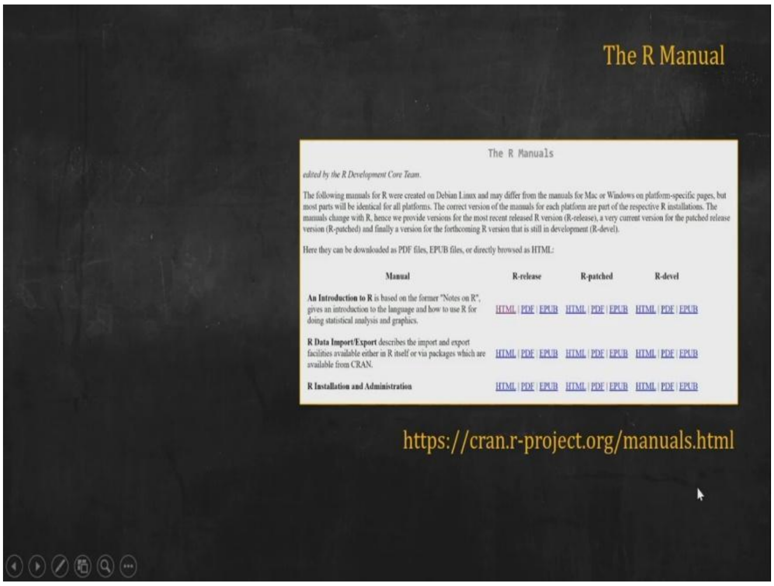
https://cran.r-project.org/manuals.html

If you go to the R web page and look click on the task view, you will get a consolidated list of different packages based upon their utility. For example, you can show here, it has been seen here that cluster, if you click on cluster, it will take you to the packages where packages for cluster analysis and finite mixture models are there.

So, suppose you want to perform cluster analysis and we have discussed that we need a particular package for that; you can go and download that. And I will show how to download and install an R package on your basic R. I will go into that. Now, along this course, we will have lectures by which I will show you, demonstrate you, how to use R to perform certain data analysis that we are learning in this course.

I believe following those lectures and videos you will be able to perform those operations those work. But, apart from there, there are lots of web resources available on, how to use R. There are books on how to use R for data analysis. There are books on how to use R specifically for biologists also. If you wish you can consult them.

(Refer Slide Time: 05:38)



The R Manual

The R Manuals

edited by the R Development Core Team.

The following manuals for R were created on Debian Linux and may differ from the manuals for Mac or Windows on platform-specific pages, but most parts will be identical for all platforms. The correct version of the manuals for each platform are part of the respective R installations. The manuals change with R, hence we provide versions for the most recent released R version (R-release), a very current version for the patched release version (R-patched) and finally a version for the forthcoming R version that is still in development (R-devel).

Here they can be downloaded as PDF files, EPUB files, or directly browsed as HTML:

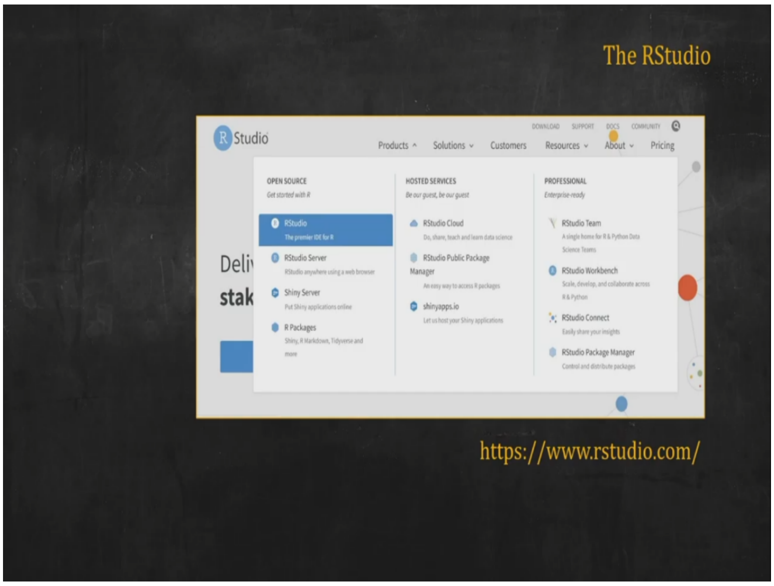
Manual	R-release	R-patched	R-devel
An Introduction to R is based on the former "Notes on R", gives an introduction to the language and how to use R for doing statistical analysis and graphics.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Data Import/Export describes the import and export facilities available either in R itself or via packages which are available from CRAN.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Installation and Administration	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB

https://cran.r-project.org/manuals.html

Apart from that, if you look into the R web page, there itself has a very organized manuals there. So, I have given you the link here. You can go there and you can download or look into the web page where you have the manuals for R. You can dig into them. And what you can do?

What I am teaching here in this course through these videos an R, you can complement that with the manual present in the webpage, R webpage. So, this is all about R. But what I will always suggest and insist in this course that apart from installing R, please move ahead and also install something called R studio. What is R studio?

(Refer Slide Time: 06:15)



The RStudio

https://www.rstudio.com/

R studio is an integrated development environment. In short it is called IDE for R. It is initially envisioned, as a you know IDE for making turnkey projects in R but for you and me, who do day-to-day data analysis from a lab data, this R studio is very useful. Because it provides all the R elements, all the R tools in one place.

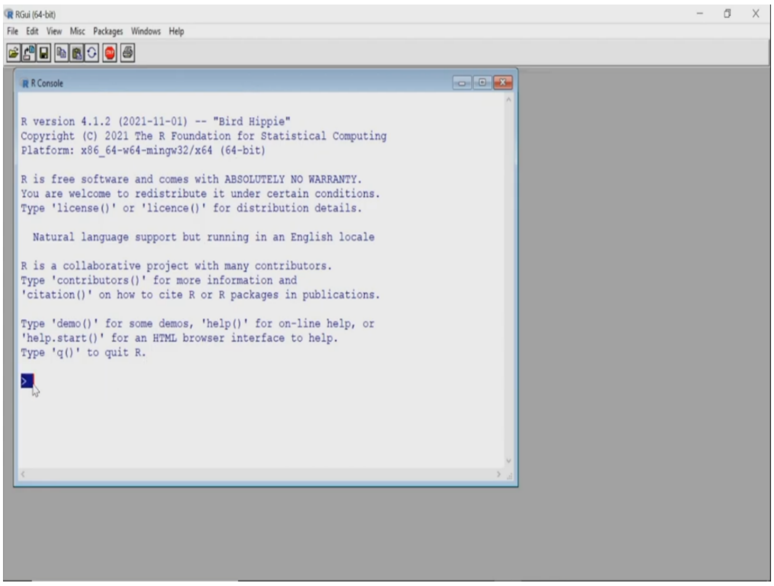
And I will show you, I will demonstrate you that how it becomes very useful. So, what I will suggest you that after you have installed R the way I have discussed till now, please also install R studio. And when you will do the data analysis along with me, you please use R studio.

After every video I will also provide you the script or the code for the data analysis that we are performing in this course. You can use those course code readymade. You can do that using the default R that you have installed and also you can use that in R studio. So, it is better to work in R studio. So, how do you install it? R studio is an IDE and it has different versions.

One version is for desktop that is for you and me, there is also server version. You can download the free desktop version free. You go to this download option in the R studio web page and it will take you to download page. And there if you scroll down you will find that the desktop version, there is a free version for the desktop version. You download and install it. Installation is easy and self-explanatory. I will not go into those details.

Now, if I assume that you have followed this instruction, downloaded R, installed R and then have installed R studio, let us move into R first and show you the basic aspects of R. How to use it. And then R studio. The basic things of R studio. Remember, we will have separate video on detail of different functions, different tools present in R. We will go one by one, not in one single video. So, I will first go to R.

(Refer Slide Time: 8:26)

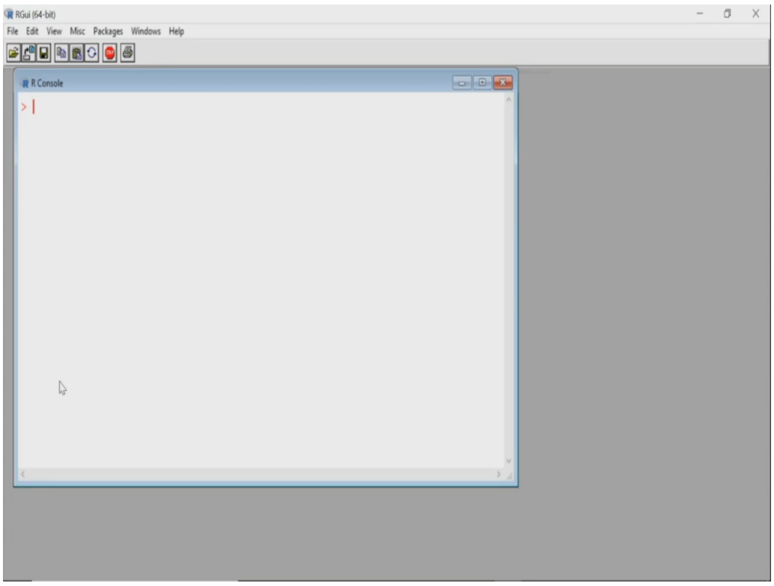


```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

>
```

I have already installed R in my machine. So, when you open R after successful installation, you get something like this. This type of thing. And in that you have some tool bars and the menus on top. And you have a windows here which is called R console. In R console, you have this greater than sign. This red color greater than sign which I am selecting now, this is called the command prompt. Beside this command prompt, I have to type the command.

Remember, R is a command tool. So, I have it is a command line tool. So, I have to write the command there and ask R to execute it. So, all the comments I will give in this command prompt. For example, the first command that I will write here, I will actually press my cursor there and I will press Ctrl L, because I want to clean this lots of information are there about the R version and all this thing. I want to clean it, so that I can start working on that.

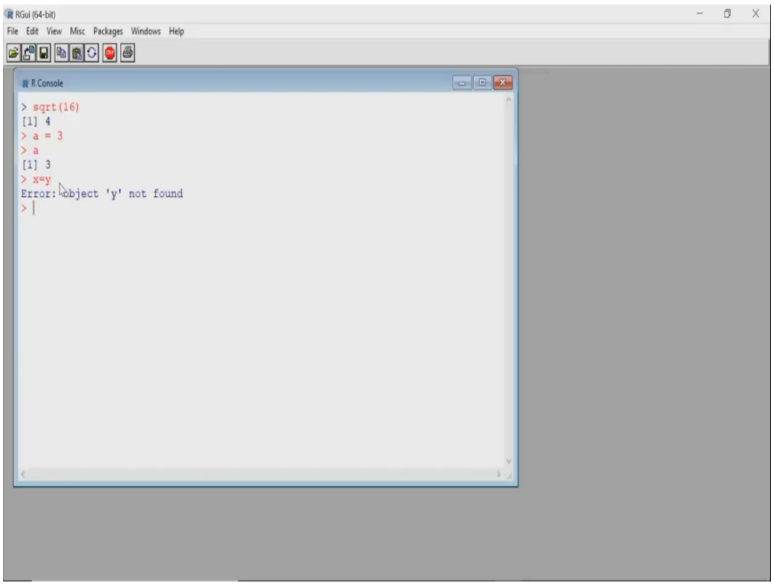
(Refer Slide Time: 09:27)



```
> 
```

So, I will press control and l simultaneously. It will get cleaned. Now, in the command prompt I want to write my first command. So, suppose, take a simple example. I want to get the square root of 16.

(Refer Slide Time: 09:39)



```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> 
```

So, what I will write sqrt then this round bracket and then within the round bracket I write 16 and then come out and then I will press enter. See, it has calculated 4. So, this sqrt(16), is the command that I have written beside the command prompt and then I pressed enter.

So, R has got received that command and send the output 4. There are lots of command. We cannot learn all of them in one go. We will learn step by step. We will learn which will be useful for our work. And in each video on the different aspect of data analysis, we will teach those.

Now, suppose I want to do something like this. I want to say, I want to make a new variable say, a. a equal to I want to assign a value 3. That means I am saying a = 3, the way you do in algebra. I can then I have written a = 3 and then press enter. Now, if I write a just beside the command prompt and press enter. It will print the value of a.

Now, suppose I say x = y. That means I am saying x, I am assigning the value of x to y. So, if I press it say it is error. Object y not found. If you have some experience in computer programming, you must be recognizing why R is giving error here. Because, remember, the computer program does not understand anything except numerical thing.

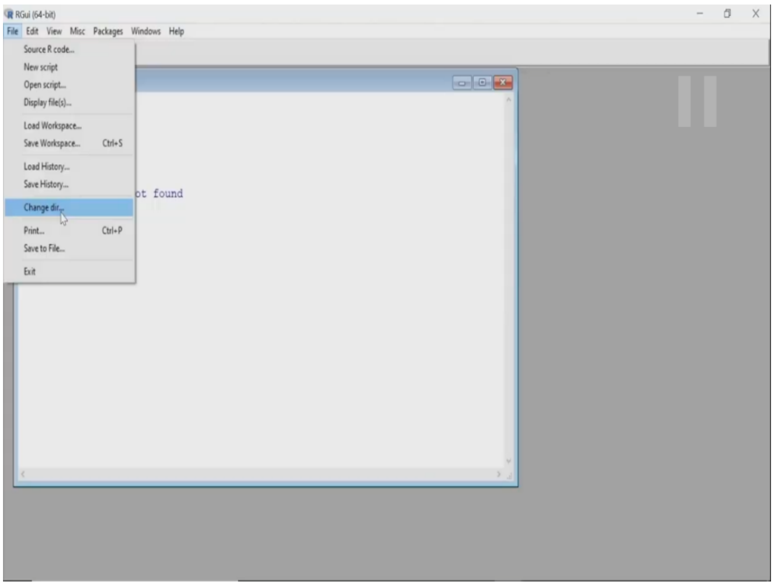
So, when I said a = 3, it has understood that I have to store the value of 3 to a and every time you call a, you will actually get 3. The number you will get. But, in the next step what I have done, I have written x equal to y. I have said y is assigned to x. y is also symbol x is also symbol. So, R could not understand, it does not have a numerical value to be assigned to x. So, it is giving error. So, that is typical of any programming language that we use. So, here also those are valid and you have to keep in mind.

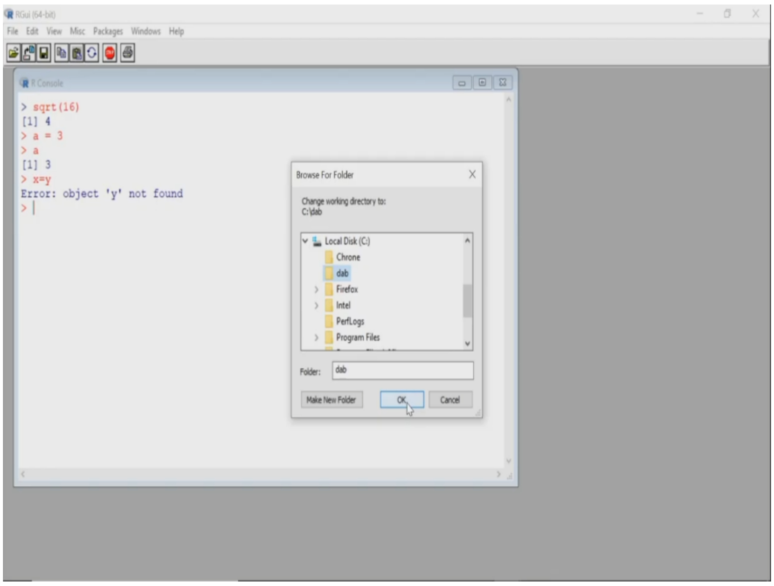
Now, this is where we write the commands. Now, you can easily understand suppose you are doing a large data analysis, where you have a data set which you will clean first. First load, then clean the data set and then you will perform, suppose, regression analysis. So, lots of command has to be given to R.

Will you do? Okay for one time, if you are doing it for the first time you can type it one by one and while doing it you may do some mistake in typing or you may use a wrong tool or wrong function. So, you have to correct them and then once you are done you are happy then you may get the result. But, if you want to reuse the same script, same code, same commands in future, what will you do? So that is where the role of scripts come. Scripts are nothing but programs codes.

So, what you do? You write a file where all the commands for a particular job, particular work are written one by one. And then you save it as a separate file, we call it R script, in this case. Now, when you want to perform a similar analysis, you will load that script and execute it. So, I will write an R script and I will show how to do that.

(Refer Slide Time: 13:00)

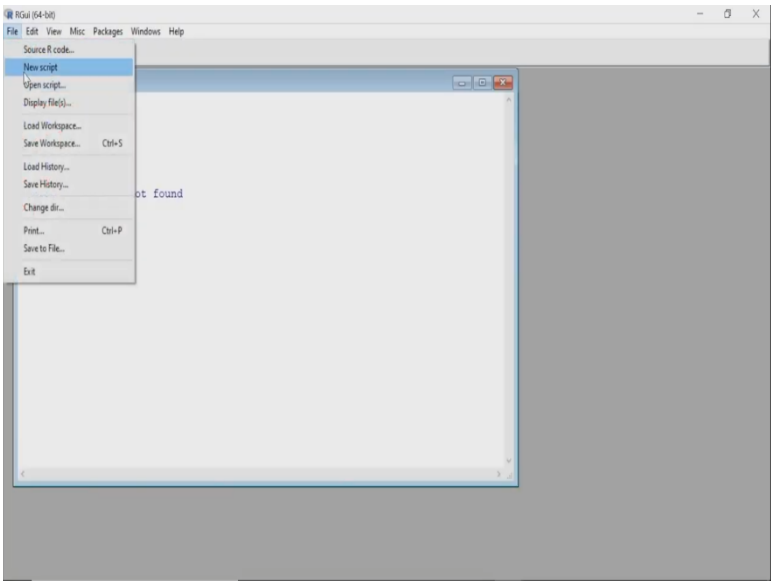




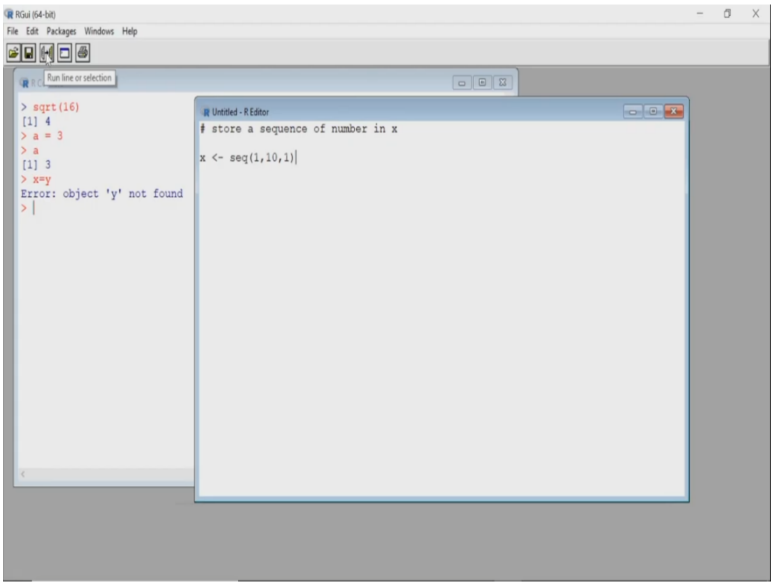
```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> |
```

For example, to do that what I will do? I will go to new script. Now, before I go to new script, one good practice is that I will set the directory where I want to work and save file and read file from. So, I will go to change directory and I will select c and this dab as my folder or directory. So, now, everything I save I will get saved there. I can read from that folder all this.

(Refer Slide Time: 13:32)

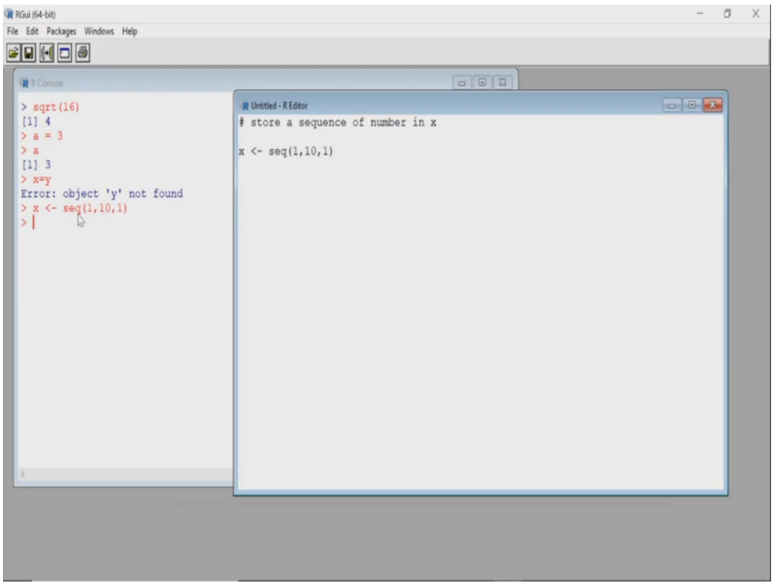


```
ot found
```



```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> |
```

```
# store a sequence of number in x

x <- seq(1,10,1)
```



```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> x <- seq(1,10,1)
> 
```

```
# store a sequence of number in x

x <- seq(1,10,1)
```

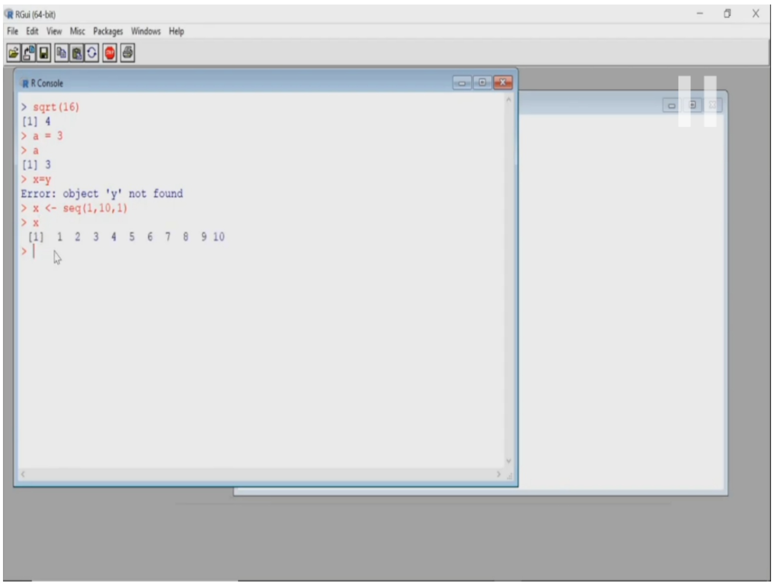
So, now, I will click on this new script and a new window will come. Now, notice it is written as R editor. It is not a console. So, here I can write a script. Script is nothing but a set of R commands.

So, the first thing I want is that I want to say x, the (⟵) back arrow is called assignment symbol, we will discuss that in a separate video, you have not to bother about this for the time being. Then I am writing sequence. I want to create a sequence of number from 1 to 10 with the interval of 1 and I am storing that to x. This is my first command. And it is a good practice when you are writing a script or a program you add comments.

Comments are for human being, you and me, so that we can later on understand which command of a particular script, which part of a particular script is doing what type of job. To write a command in R what I have to do? I have to use the # symbol then R will understand that it is not a command for it; it is a comment for you and me. It is a comment for you and me.

So, what I will write here, store a sequence of number in x, fine? So, I written that. So, I will create that, fine. So, now, let us check how it will work. So, what I can do? I can put my cursor just beside this line, that statement or I can select it, whatever you want and then you go to these toolbars. Now, you can see the toolbars has changed and you have this option called run line or selection. So, I am on this line and, this line, and I am clicking on run button. So, if I run, if you look into the console that command has been executed in the console.

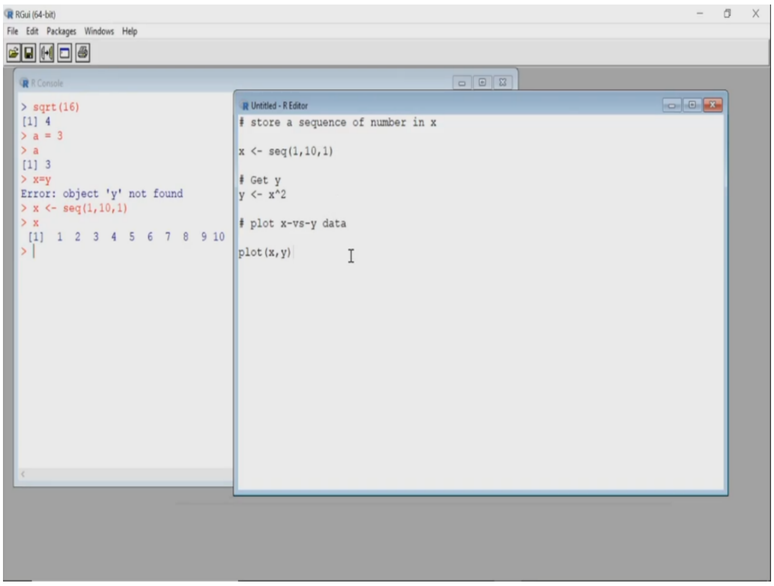
(Refer Slide Time: 15:39)



```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> x <- seq(1,10,1)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> |
```

So, now, let me go to the console for a moment and write x there and press enter, so that I can see what the previous command has done. You see what it has done. It has created a sequence of numbers starting from 1 to 10 with an increase in 1.

(Refer Slide Time: 15:57)

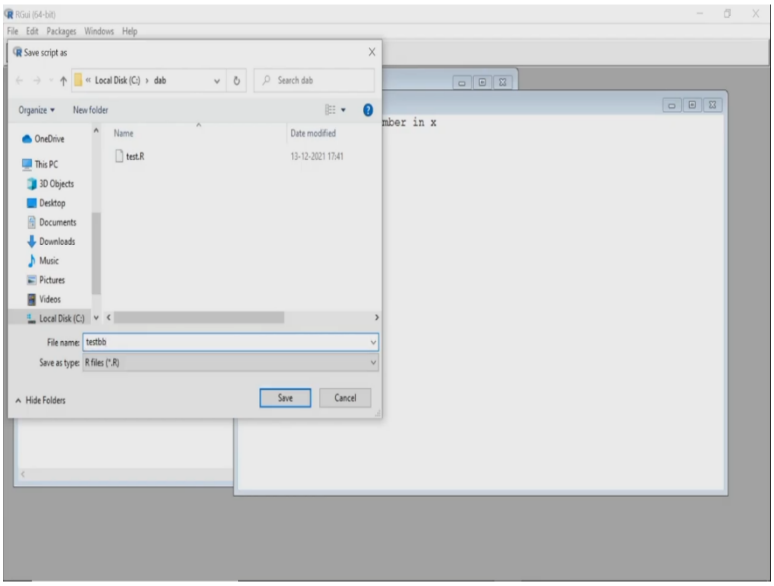


```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> x <- seq(1,10,1)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> |
```

```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

Now, I will go back to the editor, script editor and I will write y. I am assigning a value to y, a new variable. And I want that to x^2 that means I am saying $y = x^2$. So, let me make a comment. Get y. This is the step where I am getting the y. And now, I want to plot the data. So, I write a comment for you and me, plot x versus y data. And to achieve that I write a command. Plot (x,y) fine. Before I execute this script, let me save it.

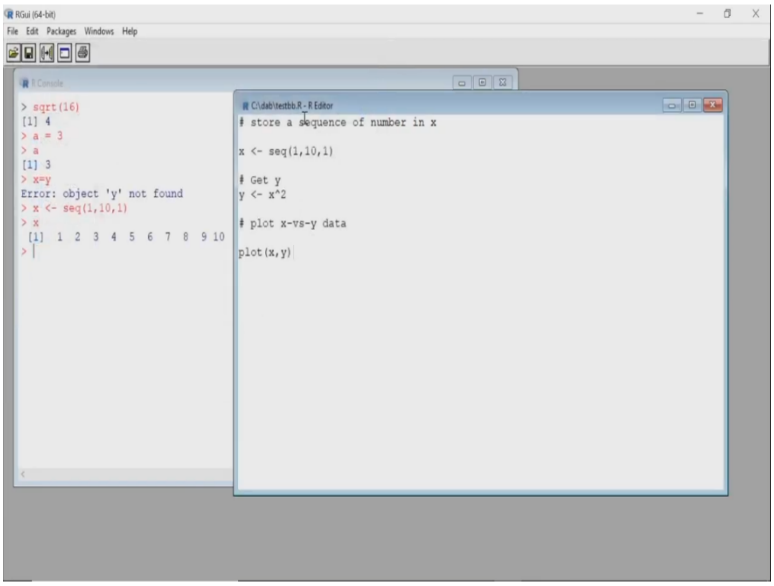
(Refer Slide Time: 16:42)



test.R 13-12-2021 17:41

File name: testbb

Save as type: R files (*.R)

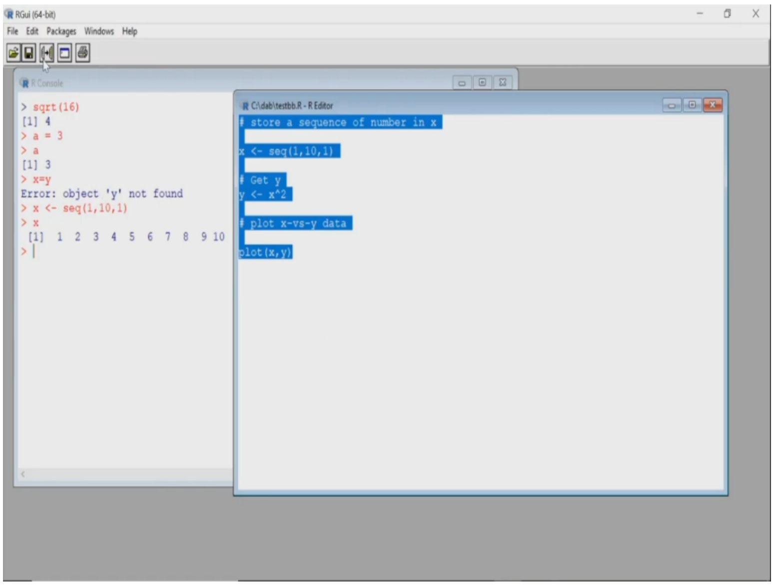


```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> x <- seq(1,10,1)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> 
```

```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

So, I click on save option. It by default takes me to this dab folder which is my current working directory and let me name it as testbb. So, it will be saved as testbb.R. So, now, what I can do? I want to run this whole script. So, what I will do?

(Refer Slide Time: 17:05)

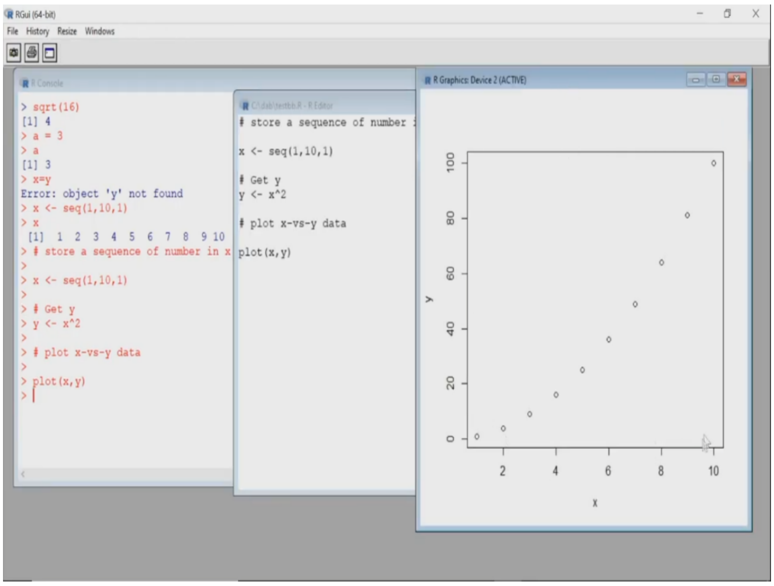


```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> x <- seq(1,10,1)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> |
```

```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

I can press Ctrl A, to select all or I can use my cursor to select all, whatever I want. So, I can also go by each line and press the run button to run up the, run each of them. But, rather than doing that what I am doing? I am selecting all of them now, and then pressing run.

(Refer Slide Time: 17:30)

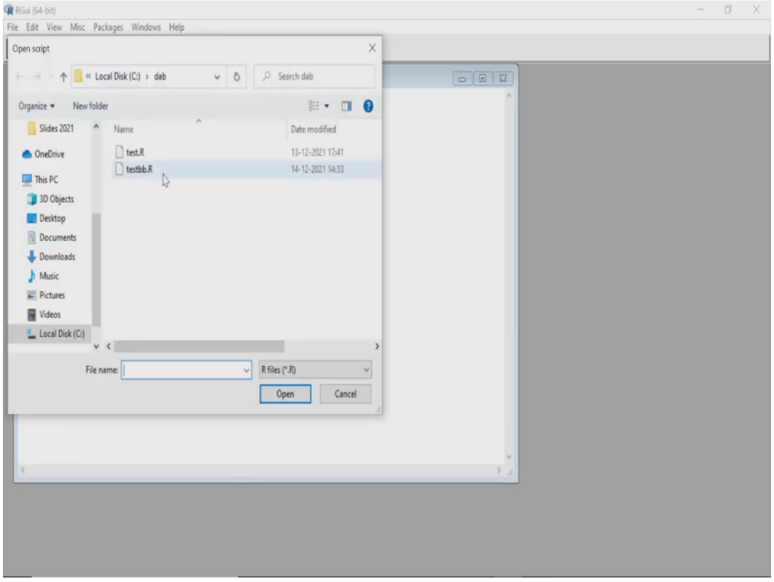


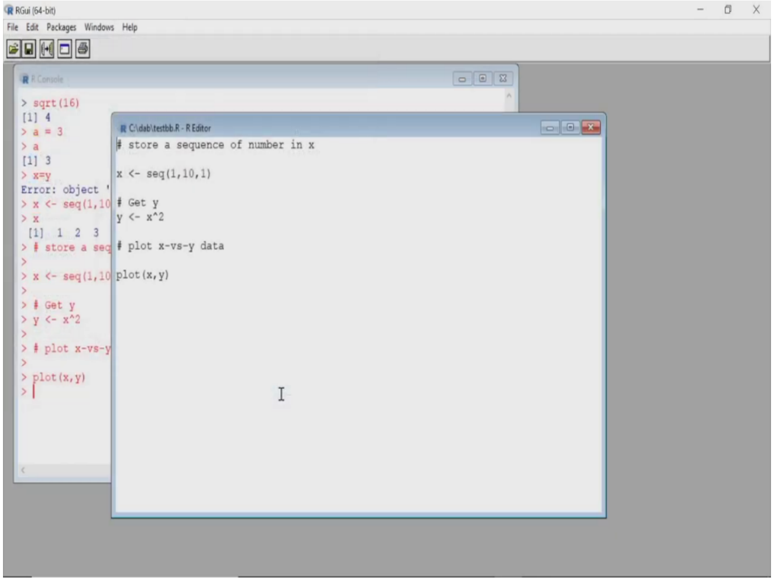
```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> x <- seq(1,10,1)
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> # store a sequence of number in x
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
> |
```

```
# store a sequence of number

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

So, now, you can see what has happened. First line executed here. x is created. Next line y is created and the last line data is plotted. And you can see the plot here. A new plot window has opened where I have x versus y plot. It is a quadratic one because $y = x^2$. So, this is how actually you can work.

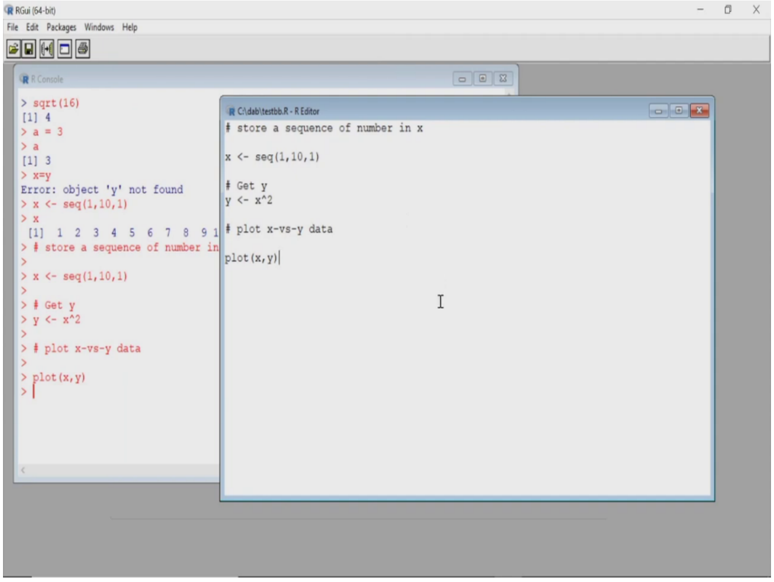
(Refer Slide Time: 18:04)





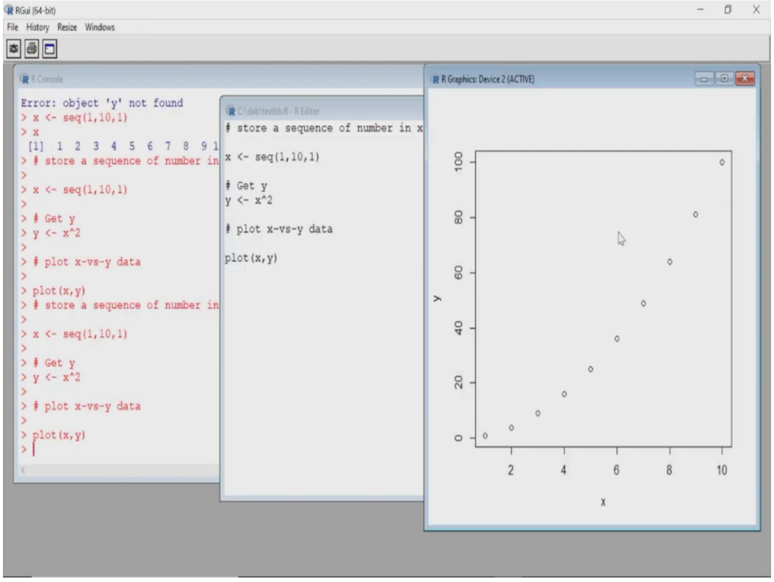
```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object '
> x <- seq(1,10
> x
 [1]  1  2  3
> # store a seq
>
> x <- seq(1,10
>
> # Get y
> y <- x^2
>
> # plot x-vs-y
>
> plot(x,y)
>
```

```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```



```
> sqrt(16)
[1] 4
> a = 3
> a
[1] 3
> x=y
Error: object 'y' not found
> x <- seq(1,10,1)
> x
 [1]  1  2  3  4  5  6  7  8  9 1
> # store a sequence of number in
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
>
```

```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```



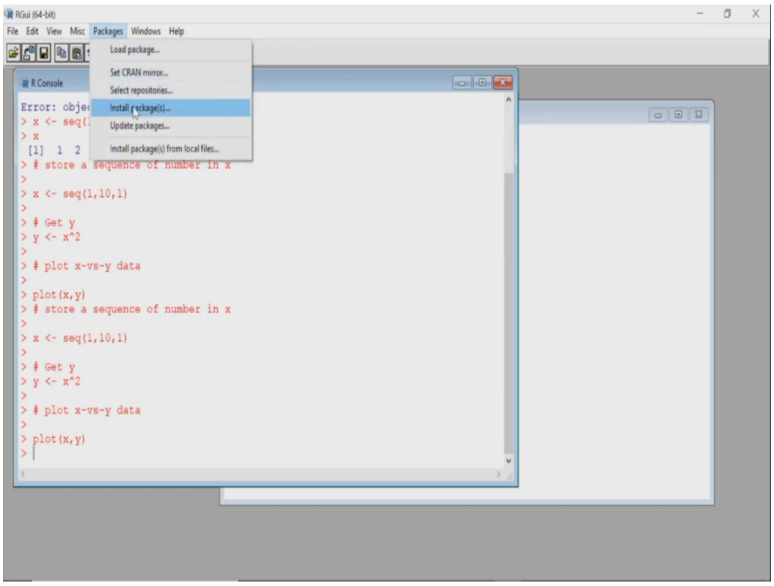
```
Error: object 'y' not found
> x <- seq(1,10,1)
> x
 [1]  1  2  3  4  5  6  7  8  9 1
> # store a sequence of number in
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
> # store a sequence of number in
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
>
```

```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

Now, what I can do? I can close these, the script also. And suppose I am working after three days and I want to create the same plot. So, what I can do? I can go to open file. Open this R script and the script is again here. Again I can place my cursor beside one line and press enter to execute that line or Ctrl A, select all and run. So, the whole script will be run.

So, this is the way you write a long script with lots of commands to perform a particular type of data analysis. Keep that saved in your folder and whenever you require, you execute it. And whenever you have to create a different modification of that you can simply modify that script and save it with a different name. And you have a set of codes now, set of scripts with you now, which you can use for different purposes.

(Refer Slide Time: 18:51)

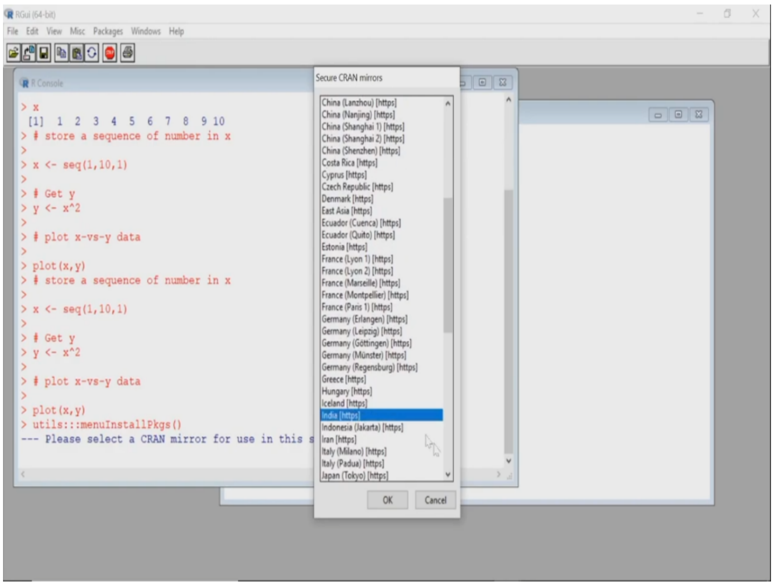


```
Error: obje
> x <- seq(
> x
 [1]  1  2
> # store a sequence of number in x
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
> # store a sequence of number in x
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
> |
```

So, once I have this, one important thing that left still now, is that what we have discussed. I have shown our console, how you write a script, how you open, write a script and execute it and save it all this thing. But, if you remember I said that when you install R, you are installing the base package. Beyond that there are lots of packages created by volunteers like you and me and which are curated and you can actually use them for specific purposes.

So, to do that you have to go to package. Suppose, in our course, we will in between we will say that you have to install this package to perform this particular analysis. So, then you have to install it separately. To do that what you have to do? Go to packages and then you click on this install packages option. These are very easy.

(Refer Slide Time: 19:39)

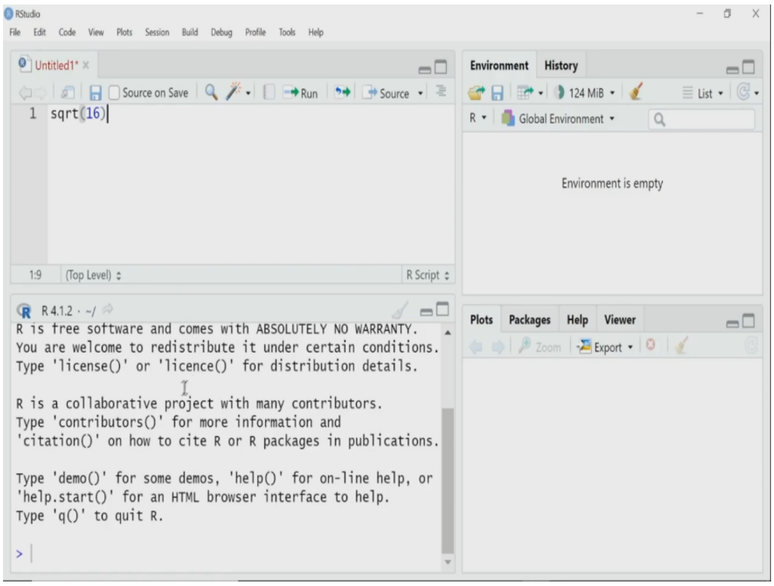


```
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> # store a sequence of number in x
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
> # store a sequence of number in x
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
> utils:::menuInstallPkgs()
--- Please select a CRAN mirror for use in this s
```

And then it will again take you to the mirror sites. And I may select, for example I may say select the India mirror and then ok then what will happen? It will get, give you a list of different packages available to you in that site. So, I can simply click one of them that I need, you have to know which one you need and then press ok. And automatically it will get installed.

For depending upon the packages, it may take some time. Wait for that to get completely installed. So, now, let me take you through R studio. Remember, R studio is working on R. So, you should have a default R installed first and then you should install R studio. And it is an IDE. So, it provides lots of thing at one place.

(Refer Slide Time: 20:25)

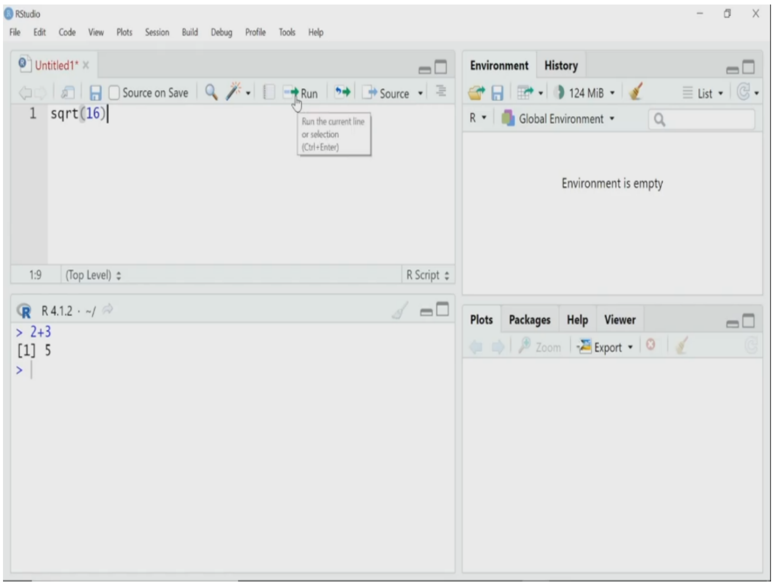


```
1  sqrt(16)
```

```
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.

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.

>
```

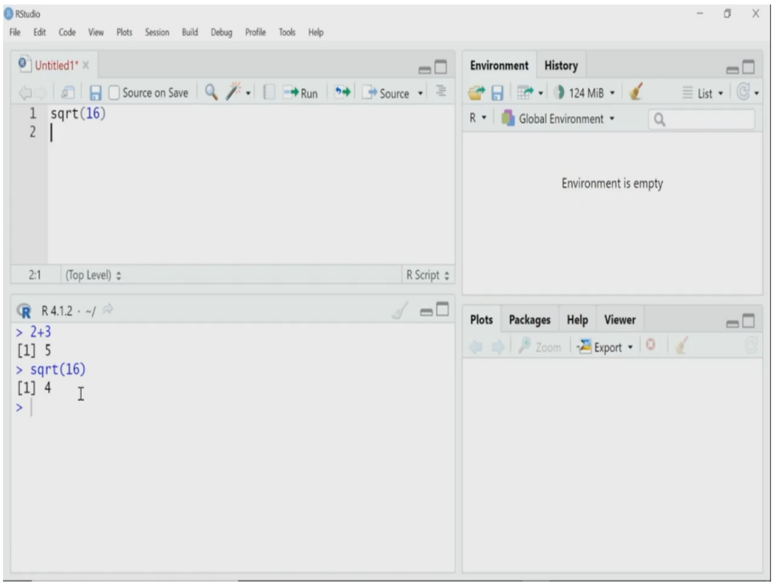
So, if when you start R studio, you get something like this, with lots of panes. So, these are multiple panes are here. One, two, three, four panes are there. And in the top pane here, this is where you write script. For example, I write sqrt(16). So, this is where you write a script something like that.

(Refer Slide Time: 20:50)



```
1  sqrt(16)
```

```
> 2+3
[1] 5
>
```



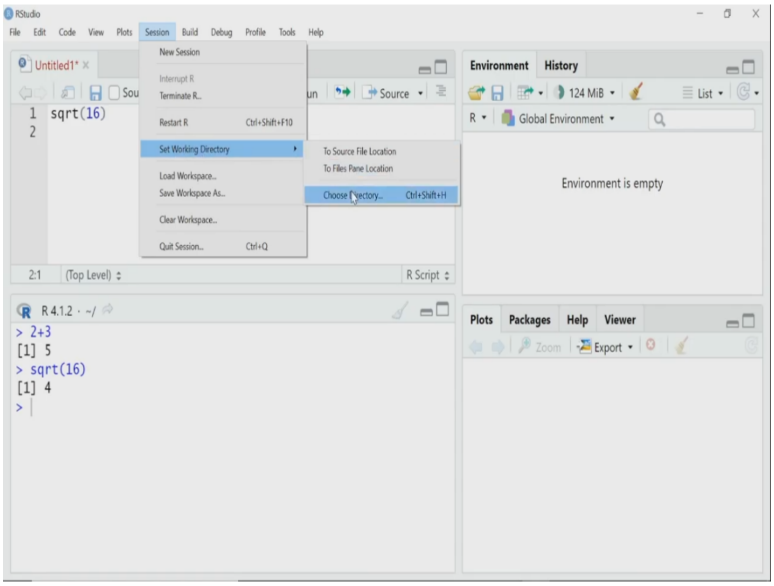
```
sqrt(16)
```

```
> 2+3
[1] 5
> sqrt(16)
[1] 4
>
```

The lower pane here is the R console that we have seen in R studio. So, let me place the cursor there on the command prompt for R console and place Ctrl L to clean it, okay. So, Now, I can write here, for example, 2 plus 3 or 3. So, it should give me 5. Just like the console it works. And where in the script here, what you have to do? Just like you have written script in R, I can type anything.

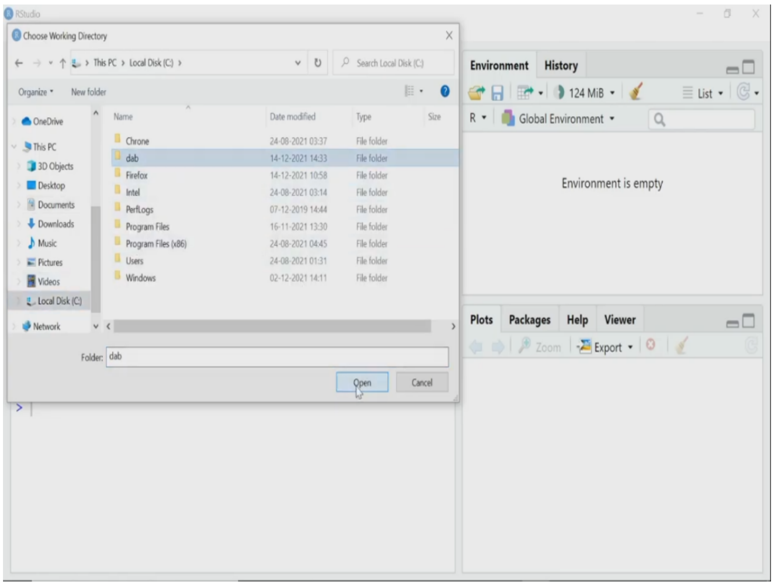
For example, I can write a command sqrt. So, to execute this particular line of the script, what you have to do? You have to put your cursor there and you have a run button, you can see, you can press that. So, it will get executed. Now, let me open that readymade script that we just now, wrote for testing the R. So, what I will do?

(Refer Slide Time: 21:51)



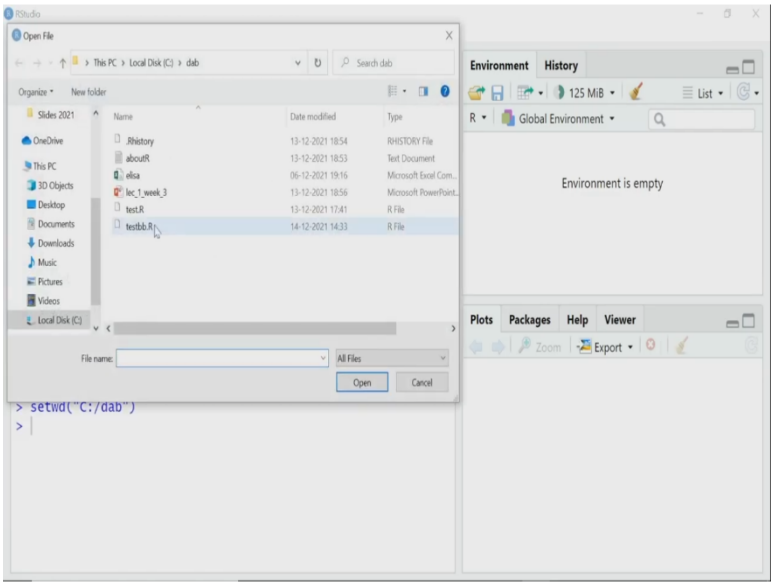
```
sqrt(16)
```

```
> 2+3
[1] 5
> sqrt(16)
[1] 4
>
```



Name	Date modified	Type	Size
Chrone	24-08-2021 03:37	File folder	
dab	14-12-2021 14:33	File folder	
Firefox	14-12-2021 10:58	File folder	
Intel	24-08-2021 03:14	File folder	
PerfLogs	07-12-2019 14:44	File folder	
Program Files	16-11-2021 13:30	File folder	
Program Files (x86)	24-08-2021 04:45	File folder	
Users	24-08-2021 01:31	File folder	
Windows	02-12-2021 14:11	File folder	

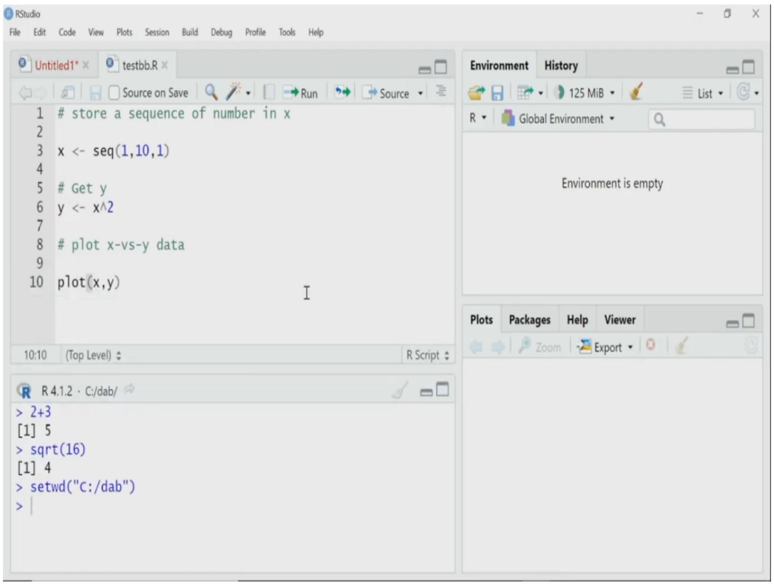
Folder: dab



Name	Date modified	Type
.Rhistory	13-12-2021 18:54	RHISTORY File
aboutR	13-12-2021 18:53	Text Document
elisa	06-12-2021 19:16	Microsoft Excel Com...
lec_1_week_3	13-12-2021 18:56	Microsoft PowerPoint...
test.R	13-12-2021 17:41	R File
testbb.R	14-12-2021 14:33	R File

File name:

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

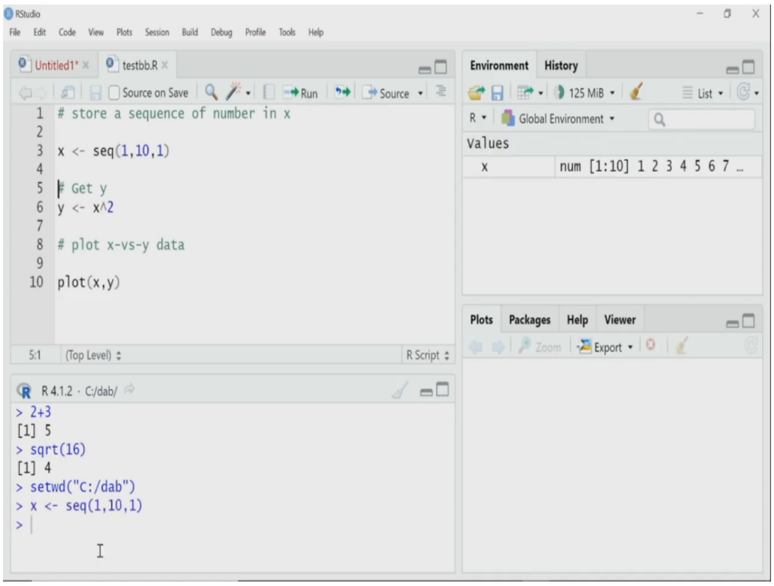


```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
> 2+3
[1] 5
> sqrt(16)
[1] 4
> setwd("C:/dab")
> 
```

First I will go to session, set my working directory, the similar thing we did for R also. So, I will choose the directory and here I will choose my directory, working directory. So, Now, I am in my working directory. And I can actually go, open file and go to that testbb script and open. Now, you can see it has come here. So, the script is now, with me. I can edit. I can run it.

(Refer Slide Time: 22:30)



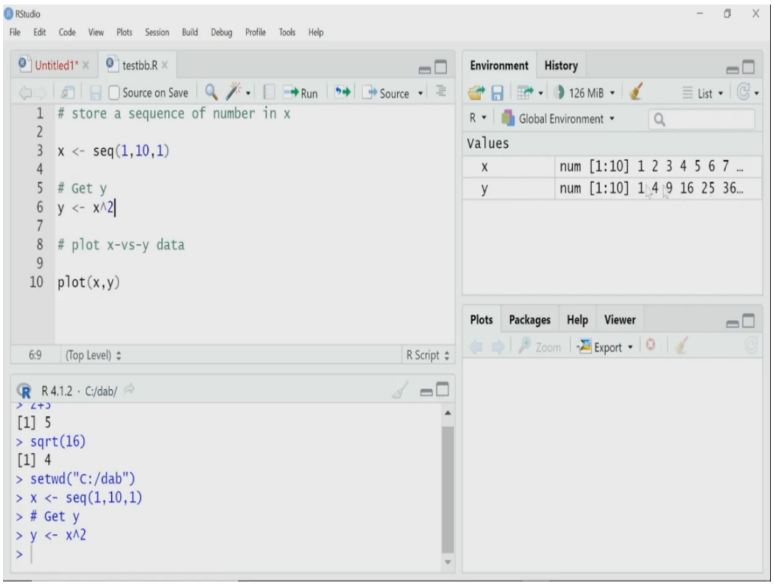
```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
> 2+3
[1] 5
> sqrt(16)
[1] 4
> setwd("C:/dab")
> x <- seq(1,10,1)
>
```

```
Values
  x               num [1:10] 1 2 3 4 5 6 7 ...
```

So, if I place cursor here and then press run, the first line for x will get executed. It has got executed. And immediately you see in this right hand side pane. This right hand side pane has two tabs. One is environment, another one is history. So, in environment, all the variables that we are created while working currently are actually shown here. For example, you can see, it is saying x is a sequence of number. So, it is saying x value, x is a variable. Its value there are 10 values. 1 to 10. And those values are 1, 2, 3, 4, 5, 6 something like this.

(Refer Slide Time: 23:09)



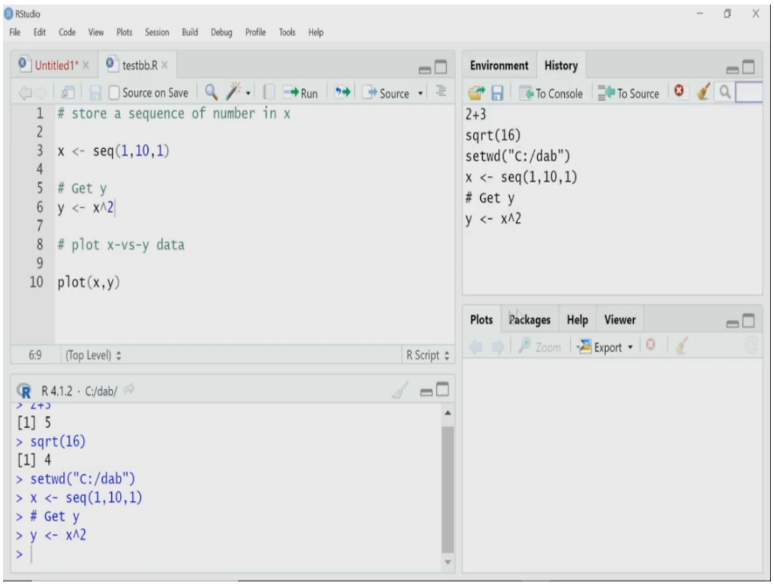
```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
Values
x     num [1:10] 1 2 3 4 5 6 7 …
y     num [1:10] 1 4 9 16 25 36…
```

```
[1] 5
> sqrt(16)
[1] 4
> setwd("C:/dab")
> x <- seq(1,10,1)
> # Get y
> y <- x^2
>
```

Let me execute the second line. So, the second line I will execute a bit different way. Rather than pressing this run, what I will do? I will press Alt enter simultaneously. So, then the second line will get executed. So, y value is created, $y = x^2$. In the environment plane you can see that though it is showing that, ok, y is a set of numbers starting with 1 4 9 sequentially like this and there are 10 numbers. What is the history tab?

(Refer Slide Time: 23:38)



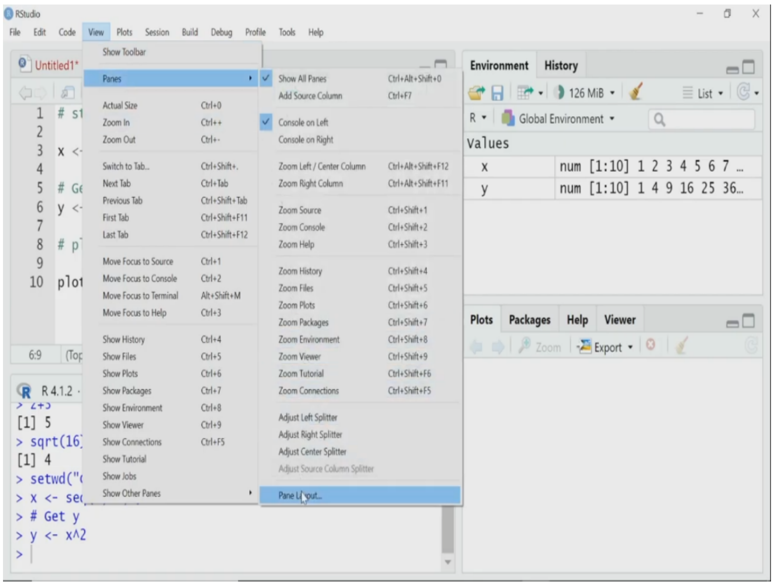
```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
2+3
sqrt(16)
setwd("C:/dab")
x <- seq(1,10,1)
# Get y
y <- x^2
```

```
[1] 5
> sqrt(16)
[1] 4
> setwd("C:/dab")
> x <- seq(1,10,1)
> # Get y
> y <- x^2
>
```

History tab shows all the comments that I have sequentially used.

(Refer Slide Time: 23:53)

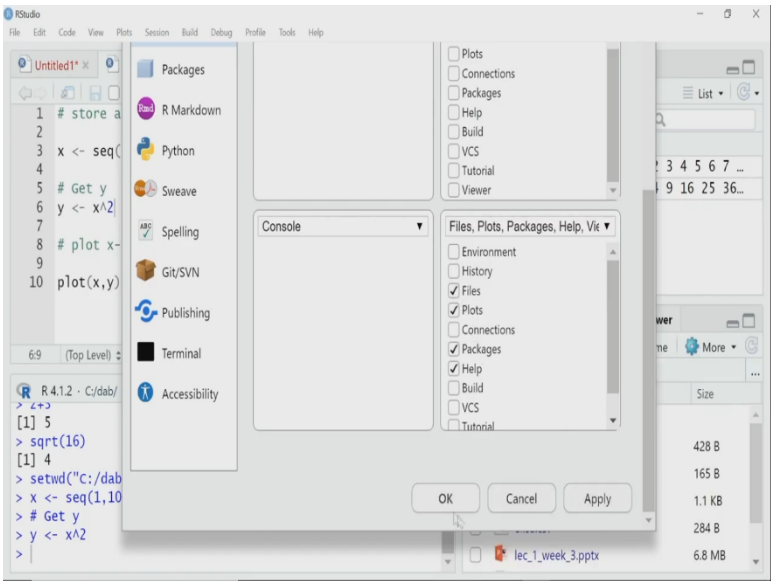


```
# st
x <-
# Ge
y <-
# p
plot
```

Values

x	num [1:10] 1 2 3 4 5 6 7 …
y	num [1:10] 1 4 9 16 25 36…

```
> 2+3
[1] 5
> sqrt(16)
[1] 4
> setwd("
> x <- se
> # Get y
> y <- x^2
>
```



☐ Plots
☐ Connections
☐ Packages
☐ Help
☐ Build
☐ VCS
☐ Tutorial
☐ Viewer

Console

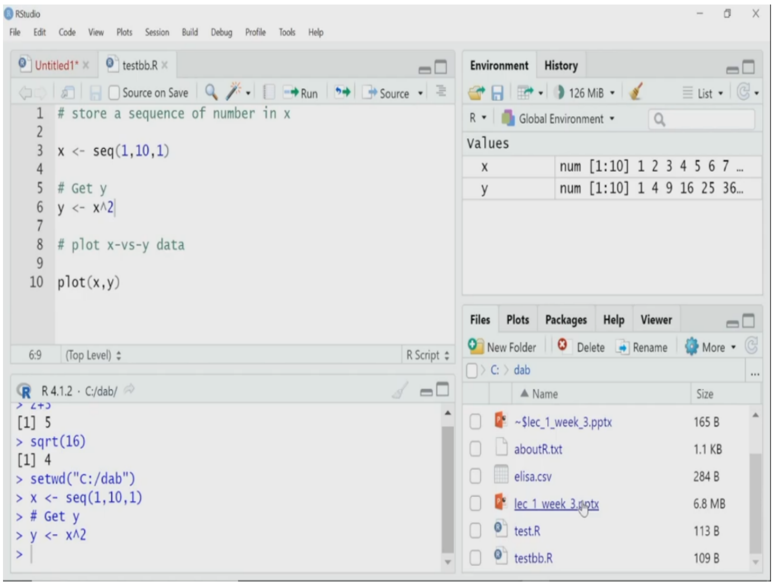
Files, Plots, Packages, Help, Vie

☐ Environment
☐ History
☑ Files
☑ Plots
☐ Connections
☑ Packages
☑ Help
☐ Build
☐ VCS
☐ Tutorial

```
# store a
x <- seq(
# Get y
y <- x^2
# plot x-
plot(x,y)
```

```
> 2+3
[1] 5
> sqrt(16)
[1] 4
> setwd("C:/dab
> x <- seq(1,10
> # Get y
> y <- x^2
>
```

	428 B
	165 B
	1.1 KB
	284 B
lec_1_week_3.pptx	6.8 MB



```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
[1] 5
> sqrt(16)
[1] 4
> setwd("C:/dab")
> x <- seq(1,10,1)
> # Get y
> y <- x^2
>
```

Values

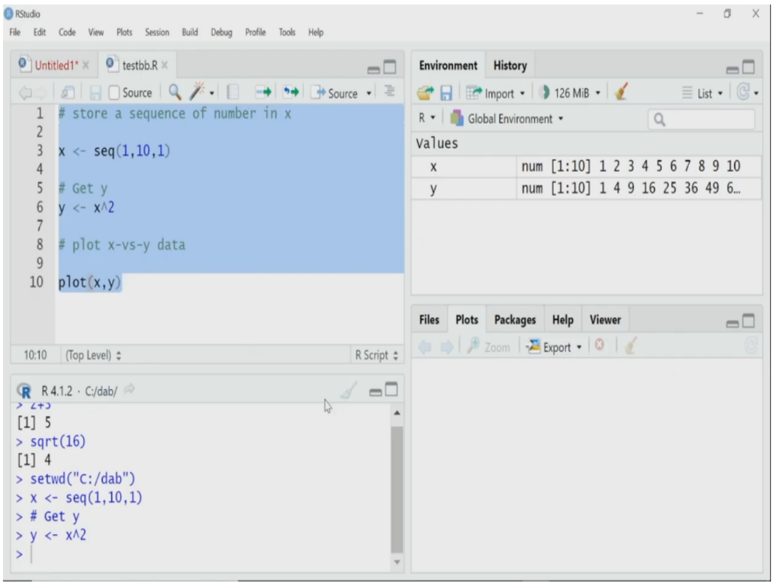
x	num [1:10] 1 2 3 4 5 6 7 ...
y	num [1:10] 1 4 9 16 25 36...

Name	Size
~$lec_1_week_3.pptx	165 B
aboutR.txt	1.1 KB
elisa.csv	284 B
lec_1_week_3.pptx	6.8 MB
test.R	113 B
testbb.R	109 B

Now, all these panes and the tabs you can actually adjust. To do that what you have to do? You have to go to view and from panes you can go to the panes layout and then you can adjust where you want a particular tab in which pane, which pane you have to remove and all this thing. Do it as per your requirement.

For example, I want that I want to have a file pane, file tab in my lower right hand side pane. So, I will click file here and press apply, okay. So, now, all the files in my current working directory are here. That becomes much easier to work.

(Refer Slide Time: 24:28)

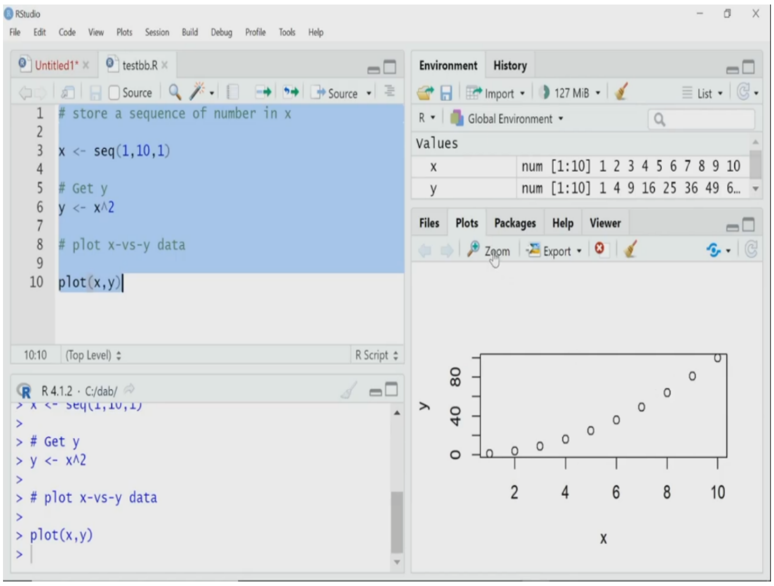


```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
[1] 5
> sqrt(16)
[1] 4
> setwd("C:/dab")
> x <- seq(1,10,1)
> # Get y
> y <- x^2
>
```

Values

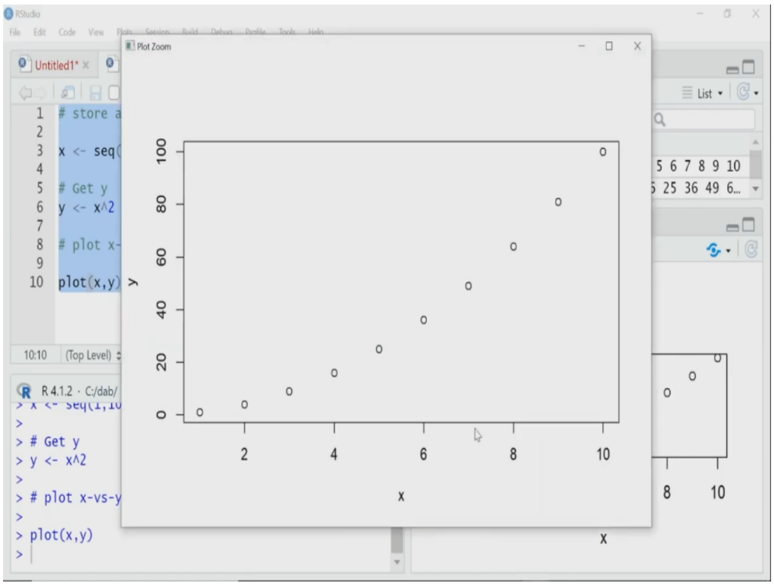
x	num [1:10] 1 2 3 4 5 6 7 8 9 10
y	num [1:10] 1 4 9 16 25 36 49 6...



```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
Values
  x    num [1:10] 1 2 3 4 5 6 7 8 9 10
  y    num [1:10] 1 4 9 16 25 36 49 6…
```

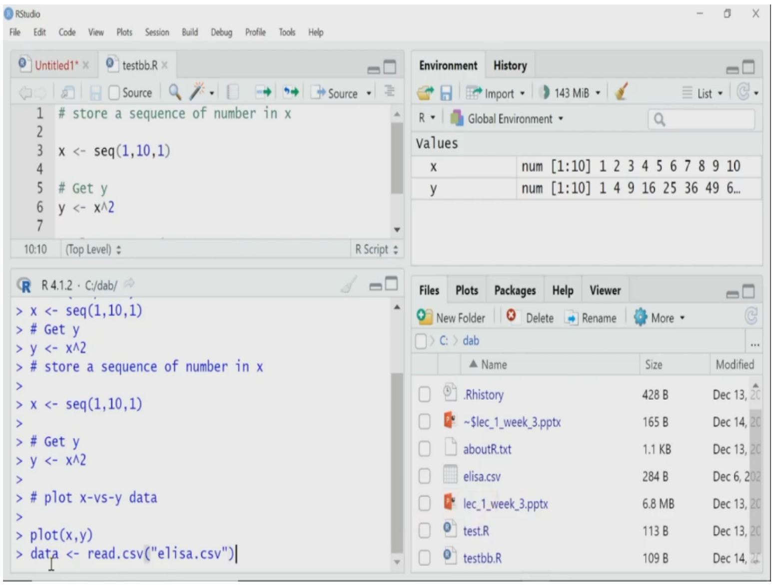
```
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
>
```



So, let me go back to my script. I have this script here which is supposed to generate a value of x and then the value of y and then plot them. And just like in R, to execute this whole script, what I will do? I will press Ctrl A or select all and then I can press run or I can press Alt enter and the whole script will be implemented. x and y value is created and the plot is created here in the plot tab on the right hand side pane, lower pane.

The plot tab is now active. You can increase the size of the tab. You can see x y plot. You may not be happy with the size of the plot. So, then you can click on zoom. You can see the plot. So, this is the way you can actually load a particular script, write a script here in R studio and then run it and you can see the data in different segment. For example, the plot will be in the plot tabs where the numerical values you can see here in the environment and all this thing.

(Refer Slide Time: 25:31)

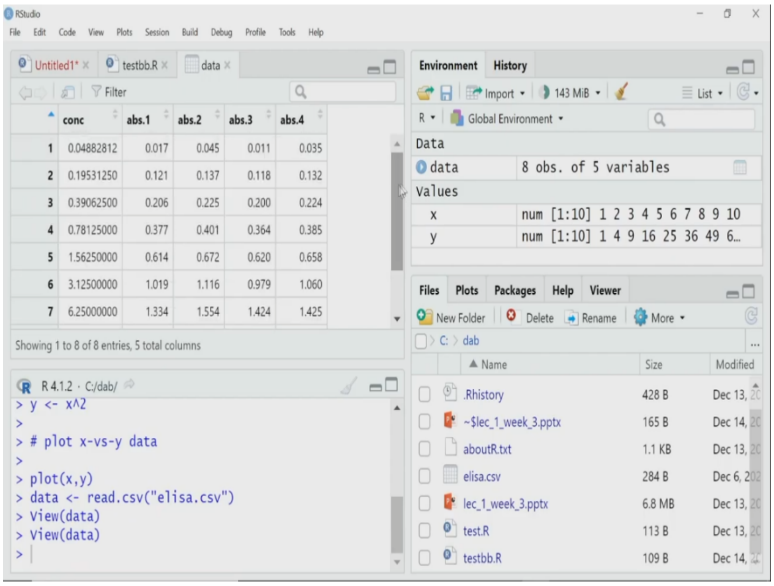


```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

```

```
> x <- seq(1,10,1)
> # Get y
> y <- x^2
> # store a sequence of number in x
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
> data <- read.csv("elisa.csv")
```

```
Values
  x      num [1:10] 1 2 3 4 5 6 7 8 9 10
  y      num [1:10] 1 4 9 16 25 36 49 6…
```

Name	Size	Modified
.Rhistory	428 B	Dec 13,
~$lec_1_week_3.pptx	165 B	Dec 14,
aboutR.txt	1.1 KB	Dec 13,
elisa.csv	284 B	Dec 6,
lec_1_week_3.pptx	6.8 MB	Dec 13,
test.R	113 B	Dec 13,
testbb.R	109 B	Dec 14,



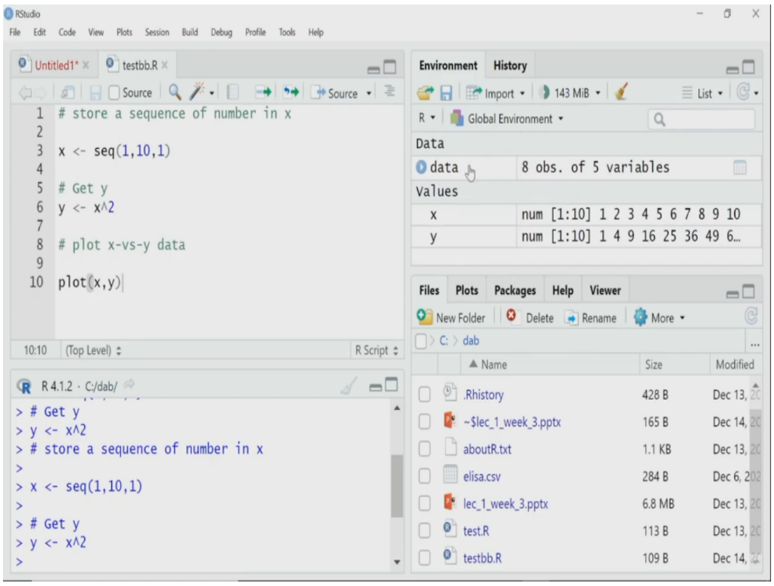
	conc	abs.1	abs.2	abs.3	abs.4
1	0.04882812	0.017	0.045	0.011	0.035
2	0.19531250	0.121	0.137	0.118	0.132
3	0.39062500	0.206	0.225	0.200	0.224
4	0.78125000	0.377	0.401	0.364	0.385
5	1.56250000	0.614	0.672	0.620	0.658
6	3.12500000	1.019	1.116	0.979	1.060
7	6.25000000	1.334	1.554	1.424	1.425

Showing 1 to 8 of 8 entries, 5 total columns

```
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
> data <- read.csv("elisa.csv")
> View(data)
> View(data)
>
```

```
Data
  data   8 obs. of 5 variables
Values
  x      num [1:10] 1 2 3 4 5 6 7 8 9 10
  y      num [1:10] 1 4 9 16 25 36 49 6…
```

Name	Size	Modified
.Rhistory	428 B	Dec 13,
~$lec_1_week_3.pptx	165 B	Dec 14,
aboutR.txt	1.1 KB	Dec 13,
elisa.csv	284 B	Dec 6,
lec_1_week_3.pptx	6.8 MB	Dec 13,
test.R	113 B	Dec 13,
testbb.R	109 B	Dec 14,



```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
> # Get y
> y <- x^2
> # store a sequence of number in x
>
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
```

```
Data
data      8 obs. of 5 variables
Values
x         num [1:10] 1 2 3 4 5 6 7 8 9 10
y         num [1:10] 1 4 9 16 25 36 49 6…
```

Name	Size	Modified
.Rhistory	428 B	Dec 13, 20
~$lec_1_week_3.pptx	165 B	Dec 14, 20
aboutR.txt	1.1 KB	Dec 13, 20
elisa.csv	284 B	Dec 6, 202
lec_1_week_3.pptx	6.8 MB	Dec 13, 20
test.R	113 B	Dec 13, 20
testbb.R	109 B	Dec 14,

Now, let me look into this environment thing where the variable values are shown. So, what I will do? I have a, if you go to the file, I have a file called elisa.csv. It has some elisa data. We will use that file later on in a lecture. So, let me write in some command in my console to read that data.

So, I will say,

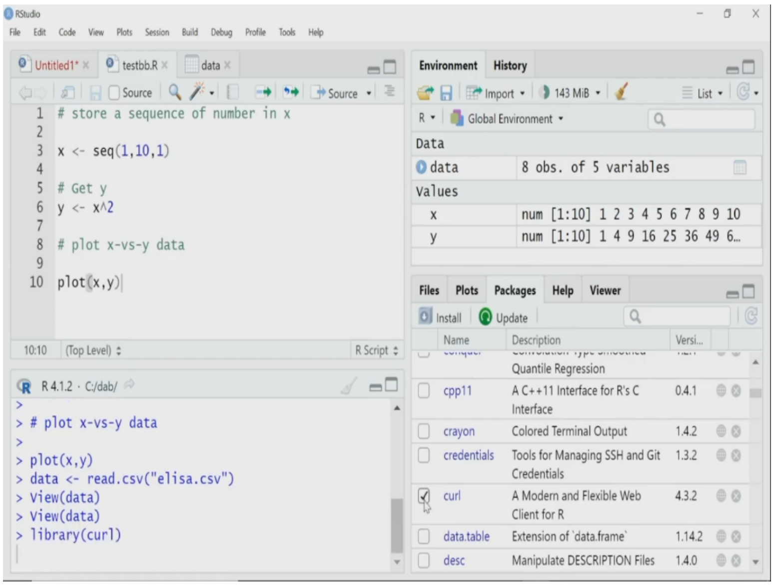
data ← read.csv(“elisa.csv”)

Let me execute it by pressing enter. So, the data has been read from my csv file, that elisa.csv file. And here it is updated in the environment tab, you can see here, on the upper right hand pane. So, if I click on this data variable, you can see here, the whole data is shown as it is as a table format in my csv file. So, that becomes very handy.

You do not have this feature in R. In R you have command by which you can look into the data file. But, that means clumsy, whereas here, you can easily, you can imagine, if you have a large file, the whole file will open up just like any other spreadsheet, here in R studio itself. Now, you can understand, okay, I have certain one column data for concentration then I have four columns for absorbance all this thing. And it becomes much easier to work with.

So, I have discussed about the script tab or pane, I have discussed the console in R studio, environment, history. We have seen the plot; we have seen the file. Then let me come to packages section.

(Refer Slide Time: 27:14)



```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

Data

data	8 obs. of 5 variables

Values

x	num [1:10] 1 2 3 4 5 6 7 8 9 10
y	num [1:10] 1 4 9 16 25 36 49 6…

```
>
> # plot x-vs-y data
>
> plot(x,y)
> data <- read.csv("elisa.csv")
> View(data)
> View(data)
> library(curl)
```

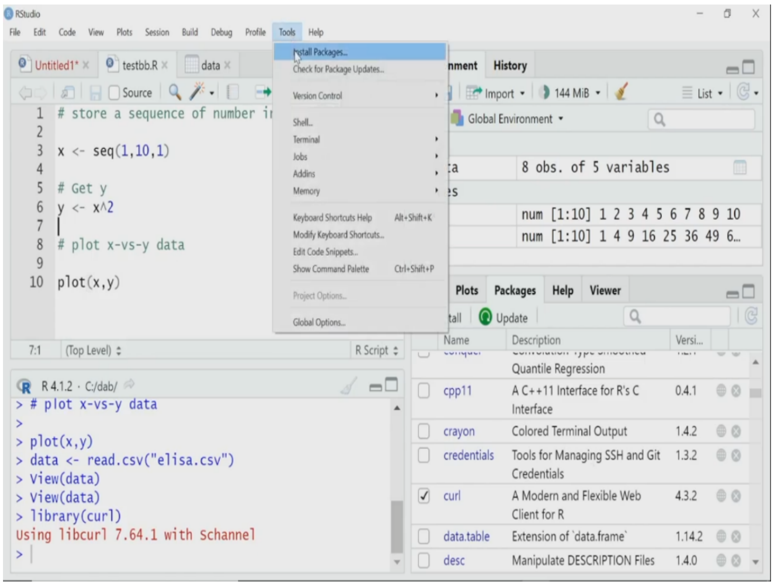
	Name	Description	Versi…
		Quantile Regression	
☐	cpp11	A C++11 Interface for R's C Interface	0.4.1
☐	crayon	Colored Terminal Output	1.4.2
☐	credentials	Tools for Managing SSH and Git Credentials	1.3.2
☑	curl	A Modern and Flexible Web Client for R	4.3.2
☐	data.table	Extension of `data.frame`	1.14.2
☐	desc	Manipulate DESCRIPTION Files	1.4.0

This packages tab now shows all the packages which are now installed in your machine. My packages I mean R packages. Now, you have to remember, when you have some packages installed in R that does not mean when you open R or R studio, automatically all those R packages will be you know integrated with your work environment at that moment. No.

To avoid the you know overloading your system what this, what R does it that, ok, even after installing the package, it automatically does not get loaded when you start R or R studio. Rather, when you require, you have to call it. So, there is a particular way to do that by a command. I will go to that separately in one lecture. But the easiest way to do that here is that suppose now, I need a particular package suppose I need this curl package.

So, what I will do? I will click on this curl. So, automatically this curl package will get loaded in my working environment right now. And I can be I will be able to use that package. So, this is how it becomes very easy to understand which packages I have right now, and to load that in your current working area. Now, suppose a package is not installed, you have to install that. To go do that, what you have to do?

(Refer Slide Time: 28:37)

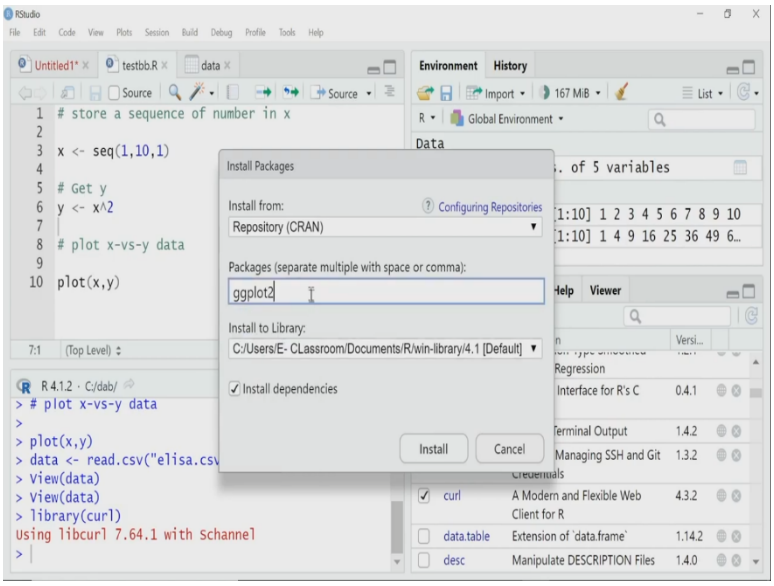


```
# store a sequence of number i

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
R 4.1.2 · C:/dab/
> # plot x-vs-y data
>
> plot(x,y)
> data <- read.csv("elisa.csv")
> View(data)
> View(data)
> library(curl)
Using libcurl 7.64.1 with Schannel
>
```

```
8 obs. of 5 variables
num [1:10] 1 2 3 4 5 6 7 8 9 10
num [1:10] 1 4 9 16 25 36 49 6…
```

	Name	Description	Versi...
		Quantile Regression	
☐	cpp11	A C++11 Interface for R's C Interface	0.4.1
☐	crayon	Colored Terminal Output	1.4.2
☐	credentials	Tools for Managing SSH and Git Credentials	1.3.2
☑	curl	A Modern and Flexible Web Client for R	4.3.2
☐	data.table	Extension of `data.frame`	1.14.2
☐	desc	Manipulate DESCRIPTION Files	1.4.0



```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
R 4.1.2 · C:/dab/
> # plot x-vs-y data
>
> plot(x,y)
> data <- read.csv("elisa.csv
> View(data)
> View(data)
> library(curl)
Using libcurl 7.64.1 with Schannel
>
```

Install from: Repository (CRAN)

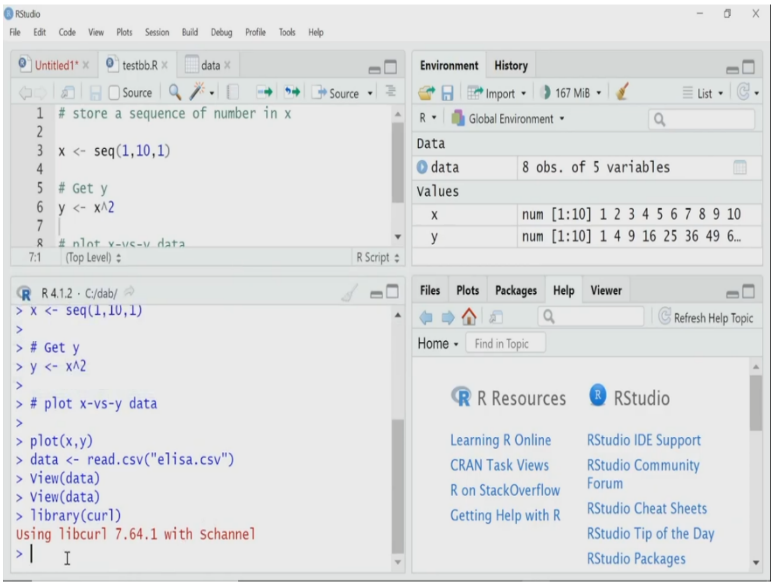
Packages (separate multiple with space or comma): ggplot2

Install to Library: C:/Users/E- CLassroom/Documents/R/win-library/4.1 [Default]

☑ Install dependencies

You go to tool, install packages, and it is very easy. You can select the repository. I can say CRAN and then I can type a suppose I want a particular, for example, I can ggplot2. So, this is ggplot2. I select and then I press install. Automatically the files will be downloaded from the mirror and it will gets installed. It is easy.

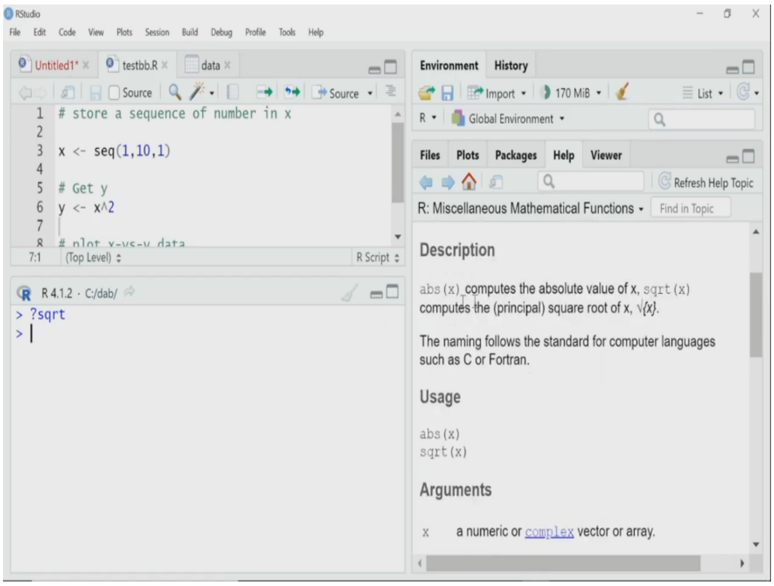
(Refer Slide Time: 29:12)



```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data
```

```
> x <- seq(1,10,1)
>
> # Get y
> y <- x^2
>
> # plot x-vs-y data
>
> plot(x,y)
> data <- read.csv("elisa.csv")
> View(data)
> View(data)
> library(curl)
Using libcurl 7.64.1 with Schannel
>
```

```
Data
data    8 obs. of 5 variables
Values
x       num [1:10] 1 2 3 4 5 6 7 8 9 10
y       num [1:10] 1 4 9 16 25 36 49 6…
```



```
> ?sqrt
>
```

R: Miscellaneous Mathematical Functions

Description

abs(x) computes the absolute value of x, sqrt(x) computes the (principal) square root of x, $\sqrt{x}$.

The naming follows the standard for computer languages such as C or Fortran.

Usage

```
abs(x)
sqrt(x)
```

Arguments

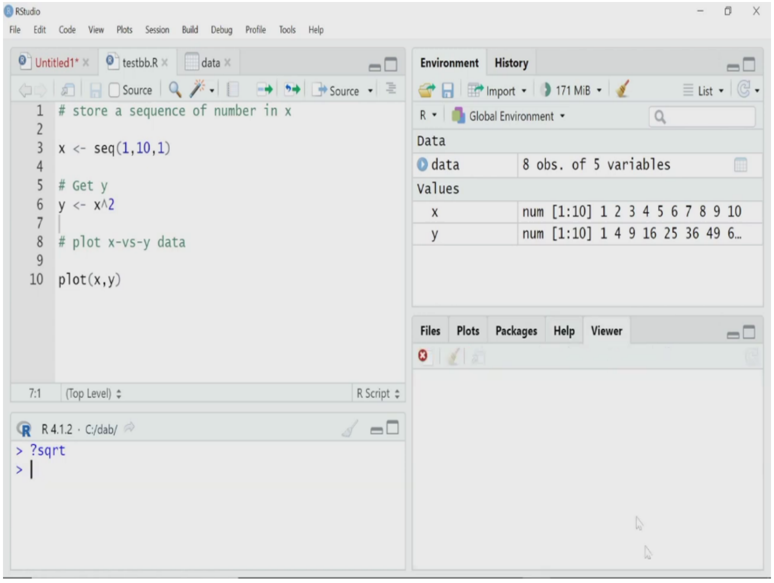
x a numeric or complex vector or array.

One thing very useful in R studio is that this help section. In normal R what will happen is that if you want to know about a particular function you have to use a particular command to get the help file for that. How you get that in R? Let me write in here in R console itself. You can have, you have to first write the question mark and then suppose you want to know about sqrt. So, you have to write s q r t. So, if you press that in R, a new window will open up.

A new page will open up where you will find the details about this function. Whereas in this case in R studio, if I execute this by pressing enter here, you can see, this help file is now here in this tab, help tab. So, what is written? It is written, it is a page about abs which is give the absolute value and the square root. And it is telling, ok, what it does and how it does and how to use it.

So, these help options helps in help information is very useful. And you may have to use it when you want to know more about a particular function and the arguments and the inputs for that for that particular function.

(Refer Slide Time: 30:30)



```
# store a sequence of number in x

x <- seq(1,10,1)

# Get y
y <- x^2

# plot x-vs-y data

plot(x,y)
```

```
> ?sqrt
> |
```

```
Data
data      8 obs. of 5 variables
Values
x         num [1:10] 1 2 3 4 5 6 7 8 9 10
y         num [1:10] 1 4 9 16 25 36 49 6…
```

The last tab that I want to discuss here for R studio is the viewer tab, where many a time there are some packages which will create an images and all this thing by creating a local web file. And that will be visible in this viewer tab. So, as I said there are many tabs and many options available. You can play with these different menus and tools here and use them as when required. That is all for this lecture. Thank you for learning with me today. See you in the next lecture.