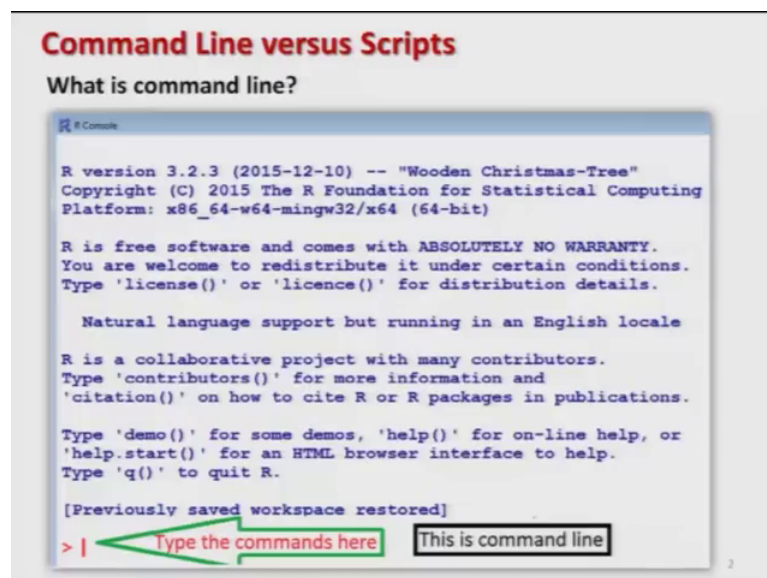


**Introduction to R Software**  
**Prof. Shalabh**  
**Department of Mathematics and Statistics**  
**Indian Institute of Technology, Kanpur**

**Lecture - 03**  
**Command line, Data Editor and R Studio**

Welcome to the lecture on introduction to R software. In this lecture we will continue with some introductory topics and you will talk about command line, data editors and say another software, R studio. So, let us try to start with one by one.

(Refer Slide Time: 00:37)



**Command Line versus Scripts**

**What is command line?**

```
R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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

Natural language support but running in an English locale

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

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

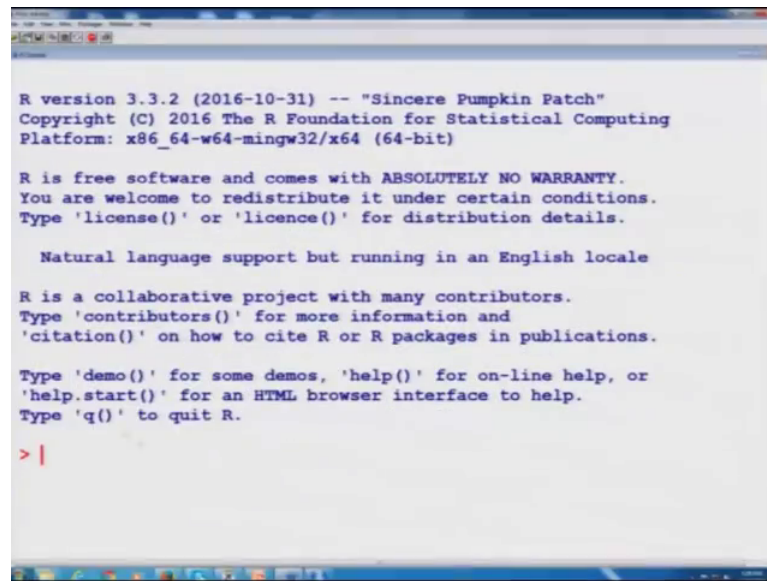
[Previously saved workspace restored]
> |
```

The screenshot shows the R command line interface. At the bottom, there is a prompt character '>' followed by a vertical bar '|'. A green arrow points to this prompt with the text 'Type the commands here'. A black box highlights the prompt area with the text 'This is command line'.

The first question comes what is a command line? Why command line? Because in order to write a program in R, there are two options. That I can write the program on the command line or I can write it inside a script file, right.

So, you have seen that when you start the R you get this type of screen over here. And here you will see that there is a sign something like greater than. This sign, greater than sign is actually the command prompt. For example, if you try to start here R this is here this thing you can see.

(Refer Slide Time: 01:13)



```
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 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

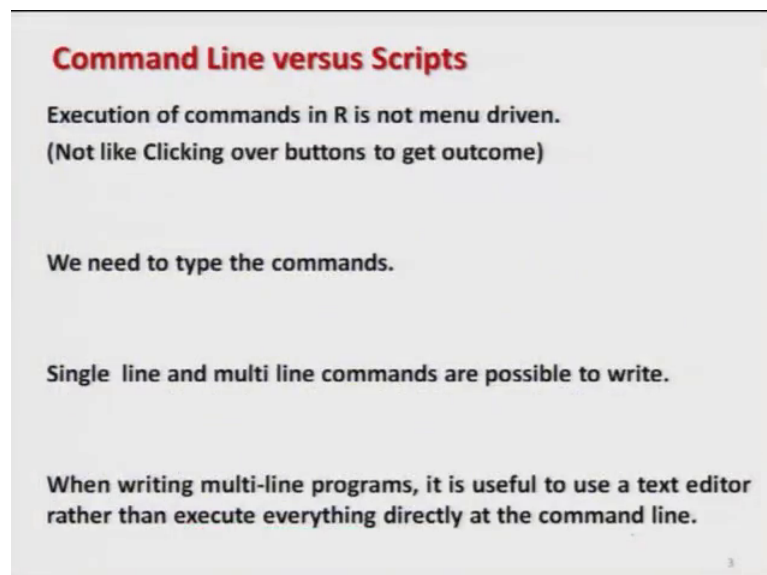
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

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.

> |
```

So, this is actually command line, and this is the place where we try to write out the syntax or command. For example, if I want to find out the mean I have to write down here and so on, right. So, the same thing is being demonstrated here. So, this is here the command line and here we try to type our commands, right.

(Refer Slide Time: 01:42)



**Command Line versus Scripts**

Execution of commands in R is not menu driven.  
(Not like Clicking over buttons to get outcome)

We need to type the commands.

Single line and multi line commands are possible to write.

When writing multi-line programs, it is useful to use a text editor rather than execute everything directly at the command line.

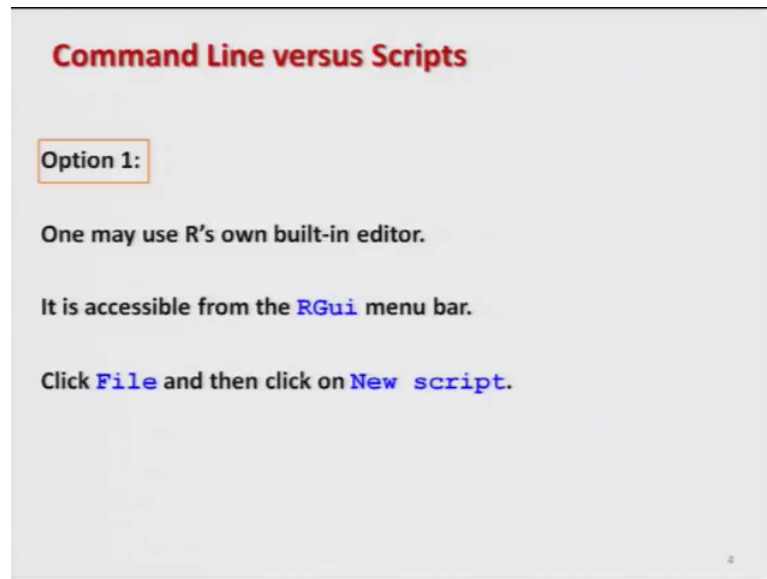
Now we come out another aspect, you might have used some software which are menu driven, what do you mean by menu driven? That there are some icons and there you have to simply click, whether you want to collect something, whether you want to copy

something where you whether you want to pay something. And there are some software which are not menu driven, but there you have to type the commands.

So, the R software is not a menu driven software, but here you have to type the commands, right. Now whenever we are trying to write down the commands there are two options, that the command can be written in a or it can be written in more than one line. So, single line commands and multi line commands both are possible to write in R programming. Whenever we are writing a single line program; that means, one command executed the same line and when we are trying to write down the multi line commands that means, we will try to write down or type one command at a time in the sequential order. So, whenever we are trying to write such multi line programs it is always better to write all the commands in a single file and then try to execute the entire file in a single shot.

So that means, whenever we want to write a program the program is essentially a combination of several syntax, several commands which are written in a logical order depending on the objective of the task. Whatever we want to do it we have to write down the program. For example, if you simply want to find out the arithmetic mean, in arithmetic mean the first step is to sum all the observation and the second step is to divide the sum by the total number of observation. So, these are two steps. So, if I want to write down a program for finding out the arithmetic mean first of all I have to write some lines to compute the sum and in the second step I have to write some lines to divide the sum by the number of observation. And this will complete the entire program for finding of the arithmetic mean.

(Refer Slide Time: 04:23)



**Command Line versus Scripts**

**Option 1:**

One may use R's own built-in editor.

It is accessible from the RGui menu bar.

Click **File** and then click on **New script**.

Now, whenever we want to write down the multi line program it is always easier in R to write the programs in a separate file and then execute it, right. So, in order to write such programs, we have got several option. First option is that I can use the built-in editor inside the r software that is Rs own editor and this software is easily accessible from the R graphic user interface window. And here you simply have to come on file and then you have to write down, I click on the new script. This I would like to show you here that how are you going to get it here. For example, if you come on the, this R software you can convert the file and then here you can click for the new script if this is on the second one. And here you can type here whatever you want here, something mean of this, say variance.

(Refer Slide Time: 05:17)

A screenshot of an R script editor window. The window title is "Untitled - R". The code editor contains two lines of R code: `mean(c(1,2,3))` and `variance()`. The window has a standard Windows-style title bar and a taskbar is visible at the bottom.

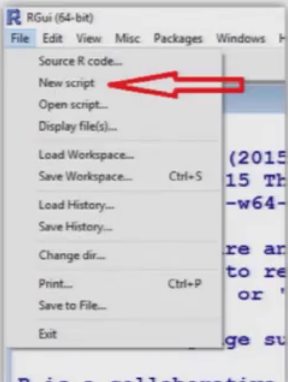
So, variance, whatever you want to type you can just type it and then you can save it and then later on you can execute it; that is the script editor which is available inside the R software, right.

(Refer Slide Time: 05:37)

### Command Line versus Scripts

At this point R will open a window entitled **Untitled-R Editor**.

We may type and edit in this.



If we want to execute a line or a group of lines, just highlight them and press **Ctrl+R**.

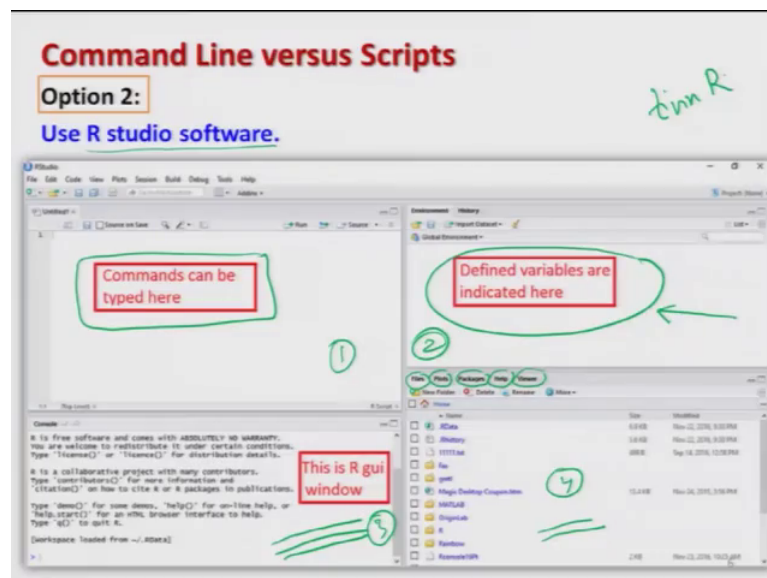
The image shows a slide titled "Command Line versus Scripts". On the left, there is text explaining that R will open a window titled "Untitled-R Editor" where users can type and edit code. On the right, there is a screenshot of the RGui (64-bit) File menu. A red arrow points to the "New script" option. Below the screenshot, there is text explaining that to execute a line or a group of lines, the user should highlight them and press Ctrl+R.

So, I have taken here the screen shot of the same window. So, first you need to click over here at this file, then you come over the new script and this will open a new window where you can type the program. And once you have written the program and if you want to execute it what you have to do? You simply have to highlight it and then press control

R. For example, here if I say that I have written here so, say here means suppose if I want to find out the mean of here 1, 2 and say 3, right. So, now, if you want to execute this command that means, I want to find out the mean of three numbers 1, 2 and 3, I simply have to highlight it and then I have to press control plus R from my keyboard. And now you will see here that in the RGUI window it is giving me this outcome. This is nothing but 1 plus 2 plus 3 divided by 3.

So, at this stage you may have an obvious question that how do I know that mean is a command to find out the arithmetic mean. I would say do not worry we are going to discuss all these things in the forthcoming lectures. Here I am simply trying to demonstrate it that how the things are being done step by step, right. So, here if you try to see using this mean function I am simply trying to find out the arithmetic mean of 1, 2 and 3 that is 1 plus 2 plus 3 divided by 3, the number of observations. So, similarly if you have here some more lines then you can simply highlight more than one lines together and you can execute it. And it is also possible to run the program without highlighting or the entire program in a single shot. So, as soon as we go further into the code I will try to show you all these things.

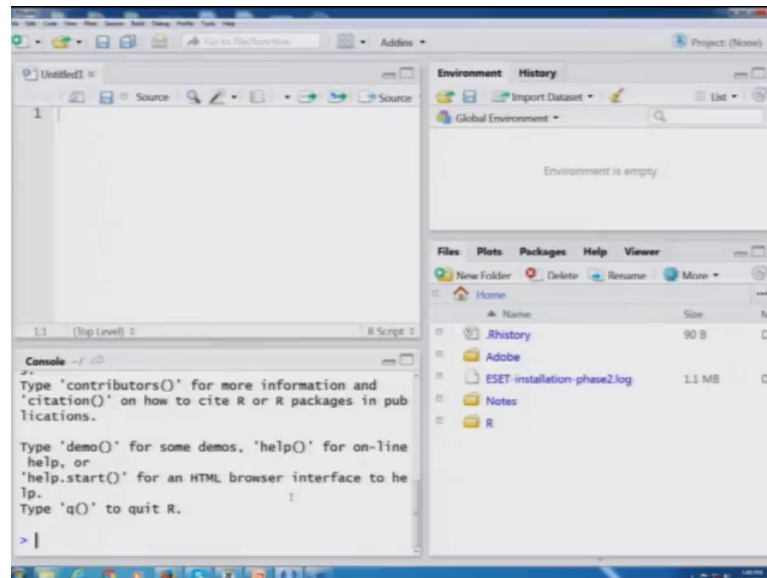
(Refer Slide Time: 07:45)



The second option is that we can use another software and as we had discussed in the earlier lecture that we are going to use here the softwares called as R studio and we had already installed it on our computer. So, here we are going to discuss that how to do all

those things inside the R studio software. So, if we are going into the explanation of the R studio software, first let us try to see that when I try to execute it how it will look like.

(Refer Slide Time: 08:34)



So, if you try to see on your say desktop there is going to be icon here like this and if you try to click over this icon you will get here a window like this one. So, you can see here that the entire screen is divided into 4 sections these four sections have got different types of utilities.

So, first we need to understand what are the utilities of these 4 windows and then you will try to understand how to do the programming later on. So, if you try to see here I have taken here a screenshot of the same R studio window with an objective to explain you all the things one by one. So, you can see here that this is the window where we try to type our commands. Whatever commands we want to execute either single line command or say multi line command or the entire file of the commands, we try to type over here. The second window is here, like this one. Here whatever variables you have defined in the program they will appear here. So, the advantage of using this R studio is that you can always see that what variable name you have given to any particular task.

Third window if you try to see here, this is a very familiar window. This is nothing but your R graphical user interface window. So, this is the same window which we obtain when we try to double click on the R icon or when we start the R software separately.

Now there is here another window here this one fourth one, this has different types of things; I can open here files, I can create here new folders, I can load here packages, I can do here some graphical things, I can obtain here different types of help and then I can view my different types of graphics over here. We will try to explain it in the next few slides, but here what I am trying to explain you here that when you try to open the R studio software there are 4 windows here; 1, 2, 3 and here 4 and these 4 windows are trying to give you all the important information about your program in a single shot.

Whereas in case if you try to work directly on the R software you may not have this facility. And then you may have to look into all this information one by one. I am not saying at all that that any of this information what is being presented in the R studio is not available in R. All this information is available in R, but working with this R studio particular for the beginners that is more easier, right. Similarly, I am not saying again at all that that R studio is the only software, there are some other software also. For example one another popular software it is tinn R, ti double n r, that is also a free software, one can download it from the website and can use it alright ok.

(Refer Slide Time: 12:09)

**Command Line versus Scripts**

Suppose we want to use following three functions:

Type them.

```
library(MASS)
attach(bacteria)
fix(bacteria)
```

Suppose we want to run only function: `library(MASS)`

Highlight it and click on Run

*Ctrl + R*

Then we get....

Now, we will come back again on the R studio software to understand that how we have to operate it. But before that I want to show you that how the things are being done. For example, I want to load a library mass. In the earlier lecture we had discussed that there is a library capital M, capital A and capital double SS called as mass which is from the



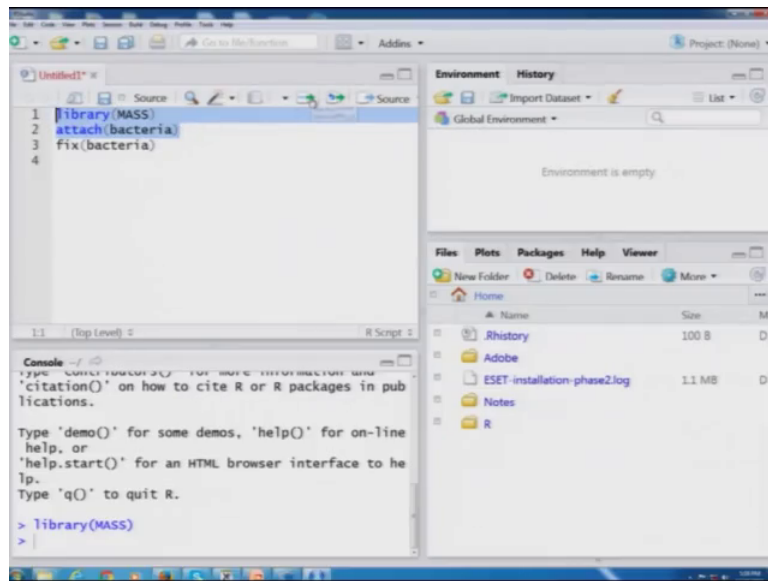
book on S plus which is available in the with base package of R. And suppose we want to use it, so I want to upload this library mass and inside that mass there is some information on the variable called as here I say bacteria. So, I want to use that variable, right. So, I have to write down here 3 commands, library MASS, attach bacteria and fix bacteria inside the brackets.

Now means obviously you will have another question that what is the meaning of attach and fix. So, again I would say that in the forthcoming lectures when we will try to explain you what are the meaning of these things. But here again as I said my objective is simply to show you that how to use the script line, command line and this R R studio. So, I have simply taken here these three simple commands and I will try to show you that how you can execute it over this, this function. So now, suppose I have typed these 3 commands on the R script window or inside the R studio. What I have to do? Suppose I want to execute only one command library mass. So, I have to simply say highlight it, just highlight it and like this on your computer and after that you simply have to say run or in the R script window you have to write control plus R or you have to type control plus R, right.

Once you try to do it you will get this type of thing in the R studio, right. So, let us try to do it over the R studio and see what are we going to get. So, I am simply trying to copy these 3 things into my R studio window over here and as I said we are going to copy these things over here. So, we highlight here to execute the command library mass and then you can see here that here on this side I am trying to click, here you have to click and this will execute the command library mass and you can see here the change. This I have highlighted in this window and you can see here that the command library mass is executed.

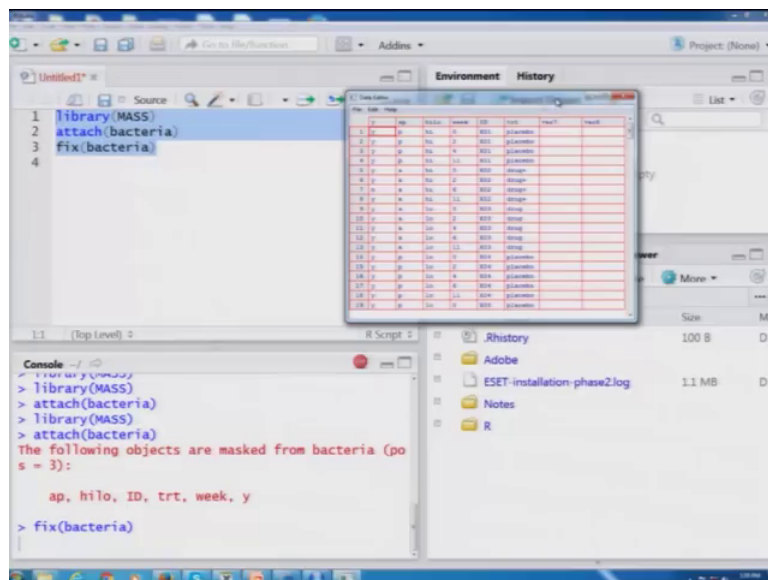
Now, suppose I want to execute these 2 commands together, what I have to do? I simply have to highlight it and then I have to click here.

(Refer Slide Time: 15:24)



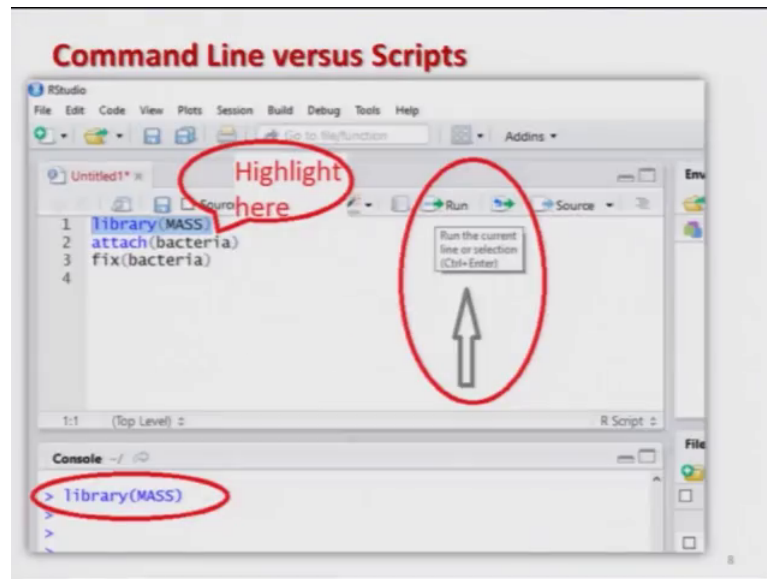
And then you can see here these 2 commands are a run together and suppose if I want to run the entire program.

(Refer Slide Time: 15:40)



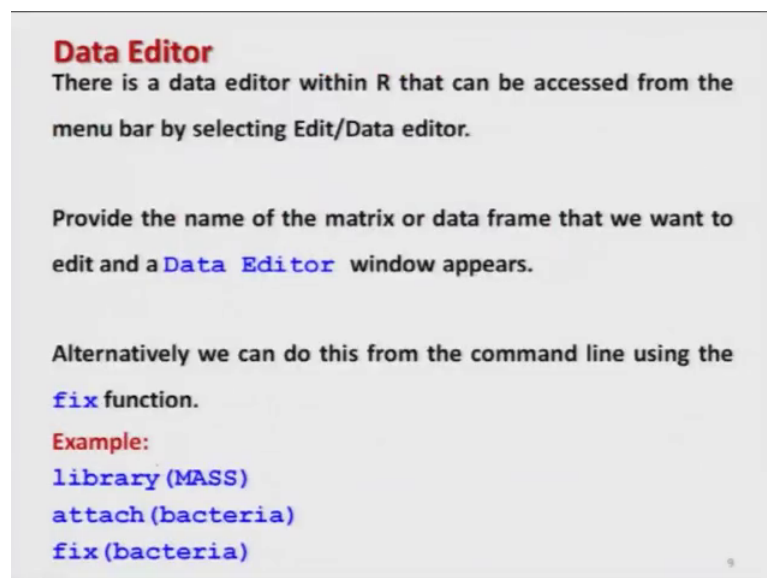
So, one option is this I can highlight it and then I can do it here and I can click here run. So, you can see here this is trying to give me the file of the data which is contained inside the bacteria here, right, because fix is a command to recall the file name bacteria. So, these things I have taken here and on my slides.

(Refer Slide Time: 16:16)



So, you can see here this is precisely what I am trying to see here. Now you can see it more clearly that I am trying to highlight it and then in the second stage you have to click here over this run and then you will see here the outcome which is here is mass right,.

(Refer Slide Time: 16:39)

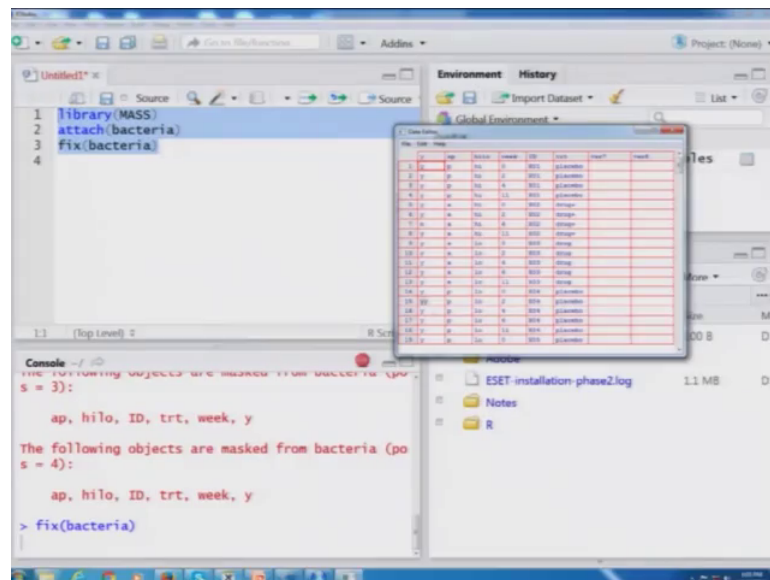


The next aspect of this whenever we are trying to do any data analysis we will certainly have a set of data and during the process we would like to edit our data. So, there are different ways by which I can call my data for editing. One option is that I can use the data editor which is within the R package and for that we simply have to go to the menu

bar and then I have to select this edit oblique say this data editor. Once you try to do it there will appear a window data editor and then you can choose whatever data you want. For example, if I try to show you here that this will be coming over here, here you can see here because in your file and then here this is here edit and inside this on the second life choice there is a data editor and if you try to open the data editor, here you have to give the name of your data. For example, if I say here mass, mass and it will try to well it is trying to show that mass is not available, now why? Because we have not loaded it.

So, first I have to load the library mass only then the data will appear, alright. The same thing we have done in the r studio also, so we try to see here that how it happens.

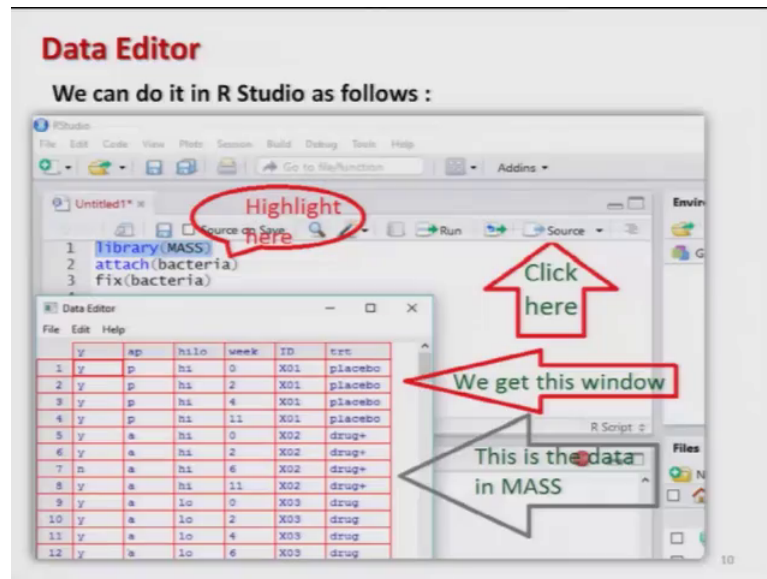
(Refer Slide Time: 18:20)



So, you can see here I already had copied these 3 commands over here and so I want to run all these things. So, I simply give over here and then you can see here that as soon as I say run, this window appears. And this is trying to give me all the data contained inside the file bacteria. So, you have both the options, either you can use the R software directly for the data editing or you can use the R studio software that will also help you in the data editing, right,.

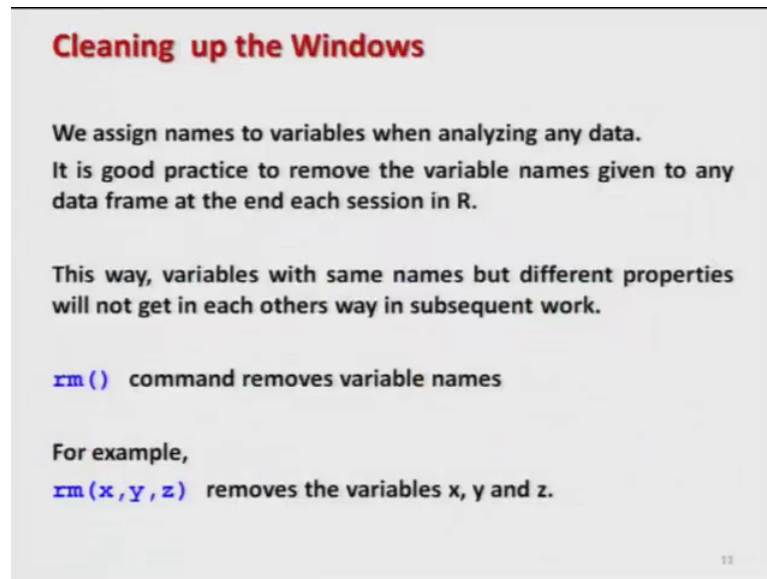
So, this is the window that you have seen that this means appears, right.

(Refer Slide Time: 19:02)



And this is all the data that is contained in the library mass. And suppose on the other hand I have another option. Suppose I have this data set and if I want to know what is the source of this data, from where this data is coming then I have another option. You see there is a command here source. So, I will try to click here this button source, here I am trying to move my cursor. So, that you can see where to click and if I try to click here you can see here all this information comes once again. That this is the information about this file and this file also appears over here and everything can be done in the reverse direction also. So, I would like that you also try to experiment all these things on your computer so that you can understand it better.

(Refer Slide Time: 20:03)



**Cleaning up the Windows**

We assign names to variables when analyzing any data.  
It is good practice to remove the variable names given to any data frame at the end each session in R.

This way, variables with same names but different properties will not get in each others way in subsequent work.

`rm()` command removes variable names

For example,  
`rm(x, y, z)` removes the variables x, y and z.

13

Now obviously, once you have done a program, right; you will you will get it ready for the next program. So, it is always better that you first clean up all the windows, whatever variable names you have defined whatever information you have defined that should be cleaned up. For example, suppose I am writing a program and in which I have used a variable name say age and by age I am trying to denote the ages of some older person. Now I am trying to use another program in which I want to find the age of some children. So obviously, I will try to define as my natural instinct the variable name to be age and I will try to enter my data, but then there can be some contradiction that that when I am trying to run the program possibly it might be using the information contained in the earlier defined variable age that was containing the ages of some elderly persons.

So, in order to make a good program we have to be a little bit careful and for that we should remove all the variables names whenever we are switching to write writes another program. So, in case if you want to remove this name we have a command what is called as rm and inside the bracket you have to write down the names of those variable which you want to remove. So, there can be 1 variable or there can be more than 1 variable. For example, if I have 3 variables say x y and here z. So, I can write down here rm inside the brackets x, y, z separated by comma. So, rm bracket x comma y comma z and bracket closed. So, if I try to do so, then the 3 variables x y z there will be a they will be removed from my workspace.

(Refer Slide Time: 22:01)

**Cleaning up the Windows**

`detach()` command detaches objects from the Search Path

It removes it from the `search()` path of available R objects.

Usually this is either a `data.frame` which has been attached or a package which was attached by `library`.

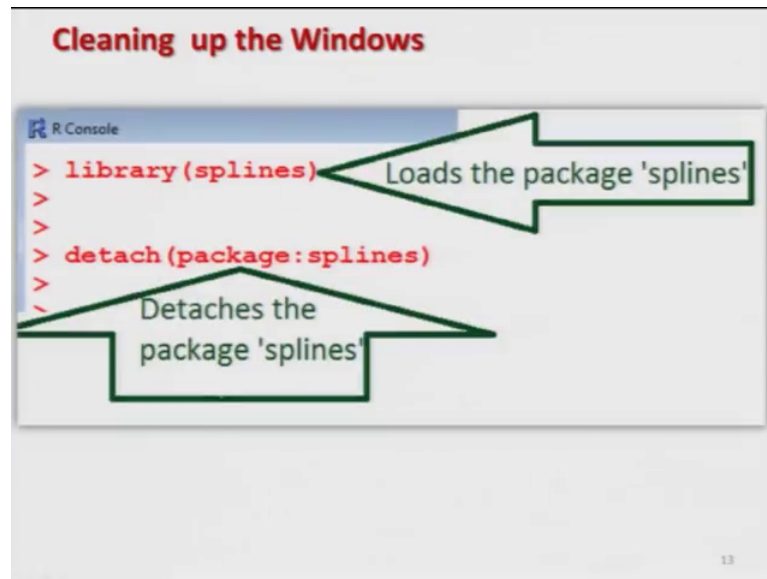
To get rid of everything, including data frames, type

```
rm(list=ls())
```

Then we get.... 11

Similarly, there is another command detach and this command detach detaches the object from the search path. And it removes those variable from the search path which is denoted by search inside the bracket, right. And usually this can be a data frame which was attached earlier in the program or this can be a package that was earlier attached to a program, right. So, this is what we try to do. Now in case if you want to remove everything suppose means data frame, data type and everything then one simple option is that you try to use the command say rm inside the bracket you can write list and then give the say quality sign and then write ls and inside the bracket whatever you want to remove means everything will be removed in a single shot, right

(Refer Slide Time: 23:11)

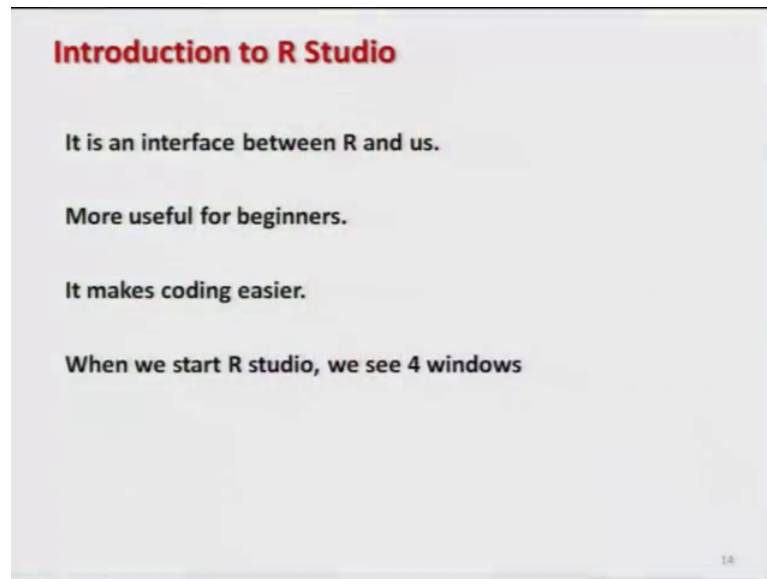


For example, in the R studio suppose I have uploaded a package say splines and I have used it in my program and in the next stage I want to remove this package so that I can use any other package, right.

So, the best option is that you simply try to write down detach package and with this colon sign you have to write splines. So, now, this command will remove or will detach the package whose name is splines. So, these are some good habits whenever you are trying to write down the program. Now we come on say this another aspect and I would like to give you some more details about the R studio software.

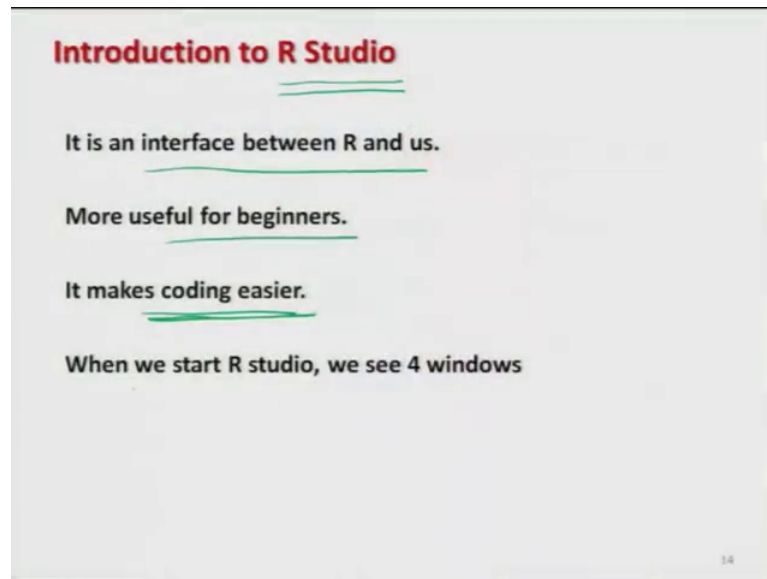


(Refer Slide Time: 24:08)



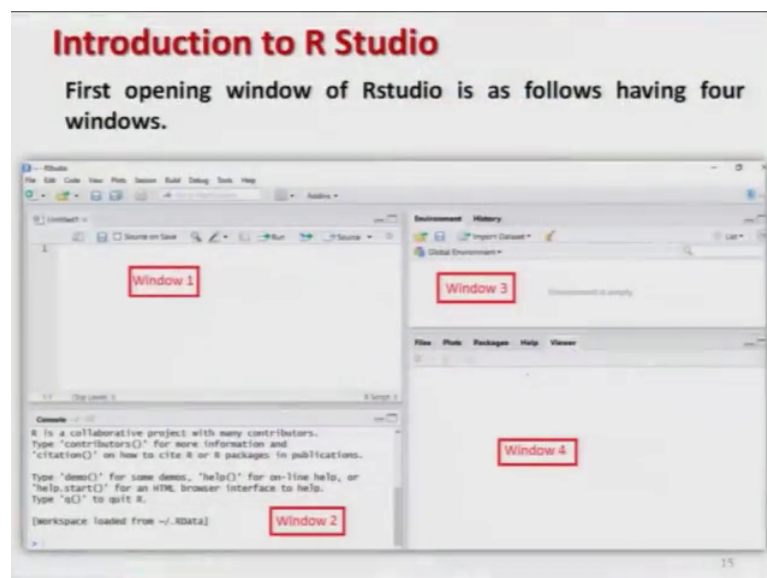
So, let us try to see. So, this R studio software as we had discussed earlier that is a free software that can be downloaded from the website and this is actually a sort of interface between R software and us. Whatever information is contained in R, whatever execution are being done in R, they can be seen through R studio also. And usually it is more helpful to work in R studio rather than working directly into the R software. And particularly if you are beginning to learn the R software, this is more helpful because you can see each and every thing just before your eyes in a single shot. And whenever you are trying to write down the program, you are trying to write down the code of a program then it is easier actually I can at every step. For example, you can highlight you can run and you can check whether your commands are working fine or not in case if you find any finally mistake at the same step you can correct it, right.

(Refer Slide Time: 25:20)



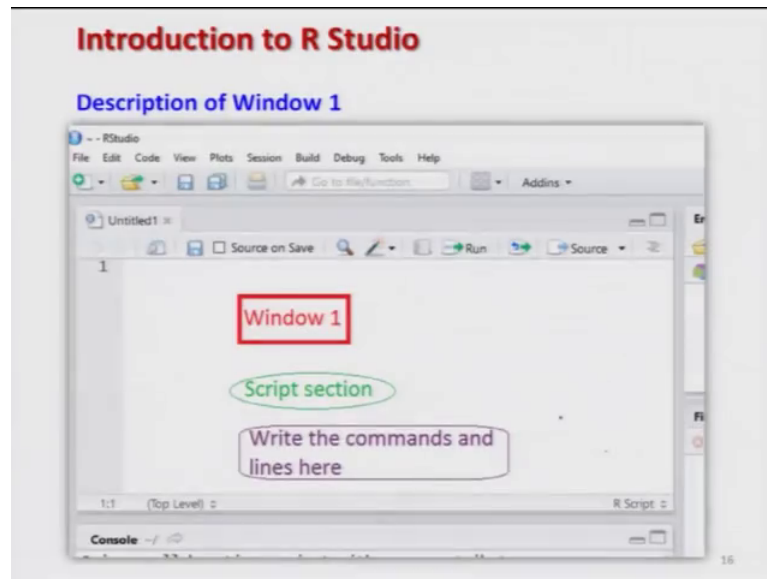
So, as we have seen earlier that whenever we start the R studio we have 4 windows. Now our objective is that we want to learn what are these 4 windows indicating, what type of information is being contained and provided by these 4 windows, right.

(Refer Slide Time: 25:45)



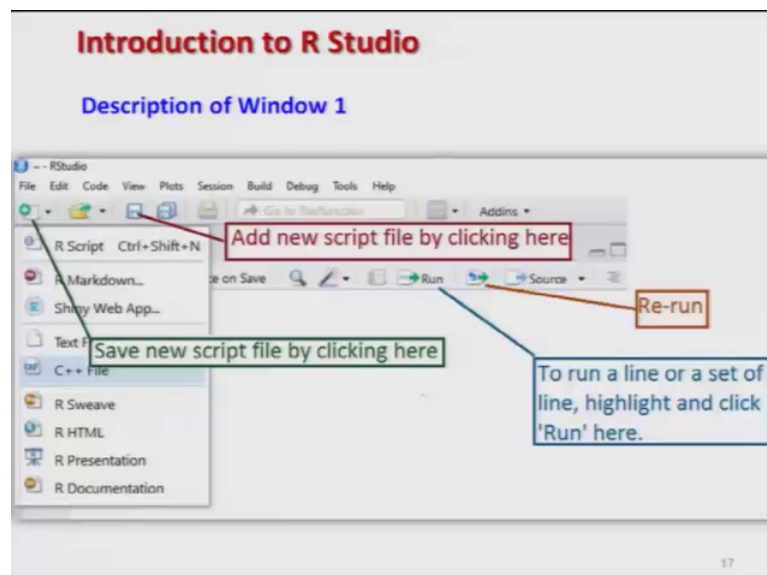
So, so you have seen here we have 4 windows. So, I am calling this as window 1, this as window 2, this has here say window 3 and this here as say window 4, right. And now let us try to understand the information provided by each of the windows one by one right.

(Refer Slide Time: 26:06)



So, first let us try to come to the window 1, right. This is a place where we try to write down the script or in simple words this is the place where we type our all the commands. They can be a single line command or they can be a multi level commands or that can be entire file containing the one program, right.

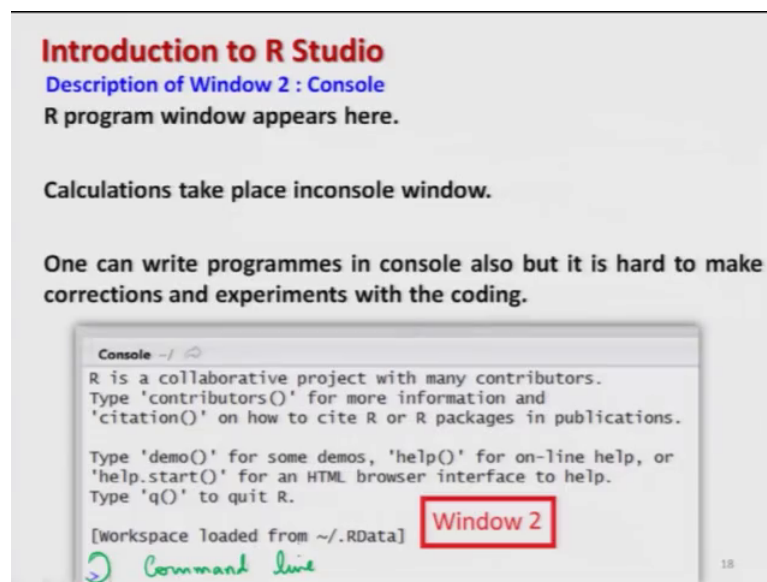
(Refer Slide Time: 26:36)



So, now you try to look at the minor details of this slide. So, you can see here that this is the place where we try to click to add a new script file and this is a place where we try to click to save the file and this is the place where we try to open an already existing file,

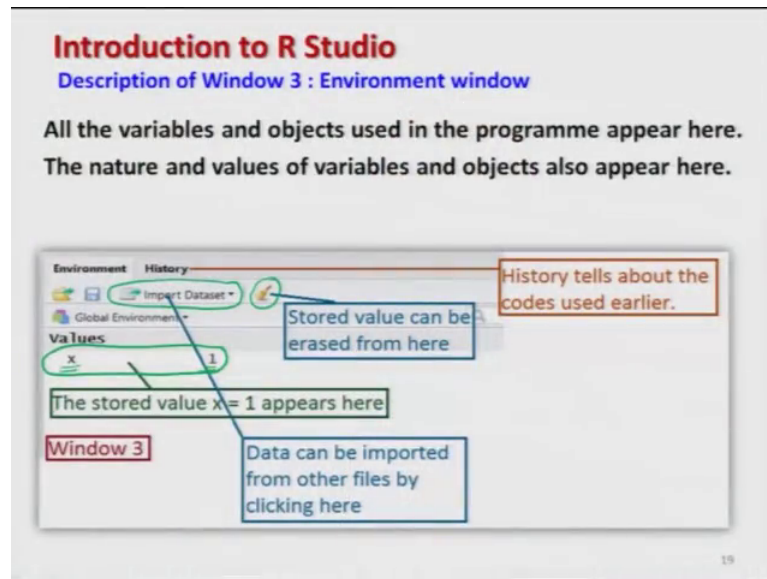
this is a place where we click to run the program and if we want to rerun the program then we need to click over here, right. These lines can be single level, single line or they can be multi lines which have to be run. So, for example, you can see here I can highlight it in the R studio software also. For example, you can see here if you try to click over here you get here R script and then you can say open here new R script and then you can see here first script and second script both are here. This is the place where you can save the details and here is the place where here by clicking here you can run the program, this is the place here way where you can rerun the program, right.

(Refer Slide Time: 28:02)



So, now I come to my window number 2. Window number 2, this is called as console window or this is simply called as console. Console is nothing means earlier we had used the terminology R graphic user interface, RGUI window, this is the same window over here. So, if you try to see this is the same place where you try to write down the commands and that was earlier called as command line. But now there is a difference, the difference is that you need not to type the commands on the command line, but you can write it over there script window and then you can directly run it. Well, in case if you want to write the program directly on the command line inside the R studio software, that is also possible, but that is not really actually needed.

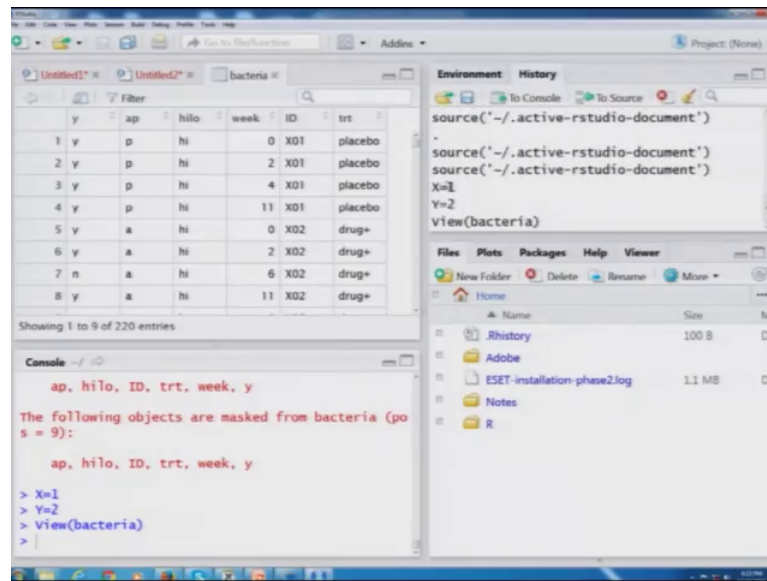
(Refer Slide Time: 29:01)



So, now I come to the window number 3. Window number 3 you can see here that it has different types of things for example, first you try to look at this part this part is writing x equal to here 1; that means, I already had defined a variable which takes value one. So, it is trying to indicate you that there exists a variable whose name is x and which is taking the value 1, right.

Similarly, there is another I can here import data set. So, in case if you already having a data set that you want to analyze, you can click over this window and can import the data set directly in the R studio software and which can be used further in the execution of a program over this data set, right. And in case if you want to and say erase any of the data are and that is already stored in this window, you have to click over here. And the stored values can be can be erased from here itself, right. And similarly, there is another icon here, history will try to give you an idea what other things that you use earlier. They are not existing in the current session, but earlier whatever variable name data sets packages whatever you have used it will try to give you the details about those things. For example, now if I try to use it here directly on the say here R studio software.

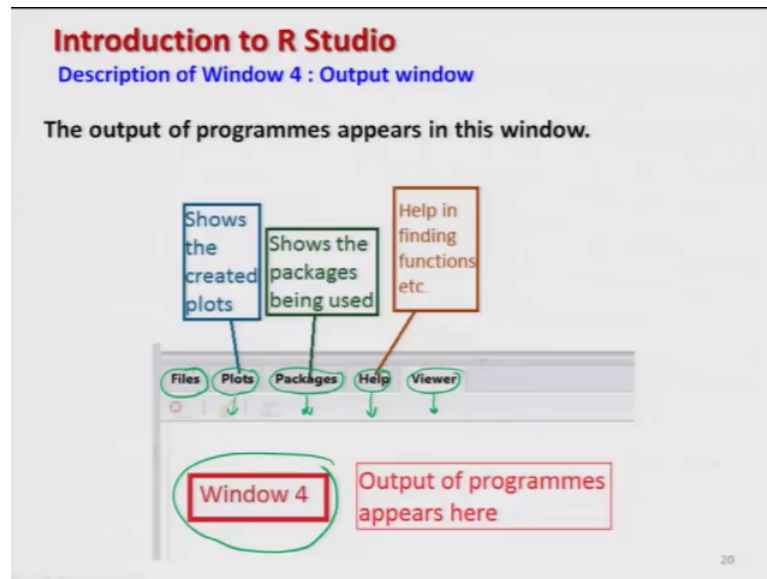
(Refer Slide Time: 30:55)



So, for example if I try to take here say write x equal to 1 and say here y equal to here 2, right. And suppose if I try to run this, so I can. So, now, in this window number 2 you can see here that it is assigning the values x equal to 1 and x equal to 2. And now this window number 3 it is trying to give us the same thing over here means I am trying to move my cursor here so that you can see that here is here x and here it is 1, it is here y it is here 2. And earlier if you try to use the information which I have used the data set on bacteria that is also given here which is trying to tell us that there are 22, sorry 220 observations of 6 variables. And if you want to see it, just click over here and you can see all this data over here, right.

So, you can see that how convenient it is to work with this R studio. Now if you try to click over the history you can see here that these are the commands which we have used earlier like a library mass, library bacteria and means we have assigned the values here x equal to 2, y equal to 2 and so on and then finally I had viewed the file name bacteria. So, this is trying to give us the history of the commands which I have used earlier, right.

(Refer Slide Time: 32:36)



But now we come to our next window which is our fourth window, right. So, now, you can see here that there are different icons; one here is file that will give you an idea about the file and there is another icon plots and another is here packages, then there is help and then there is viewer, right.

So, as far as you click first on say here viewer whatever is the outcome of your program say for example, any graphics or say any numerical values that will appear in window number 4 over here, right. I will try to demonstrate it by using a simple example over here. And similarly if I try to give you a some details quickly, help when you helps us in finding about the details about the function or say anything whatever you want to know in R. For example, earlier we had used different types of commands to find to take the help from the R software. But now this can be done very easily over here, I will try to demonstrate it. And then here there is a icon packages, this will when you try to click over here that we try to show you that what are the packages which is being used in this session. Then there is another here plots, this plots will show you whatever is the outcome if you have created any plots they will be shown over here.

(Refer Slide Time: 34:11)

### Introduction to R Studio

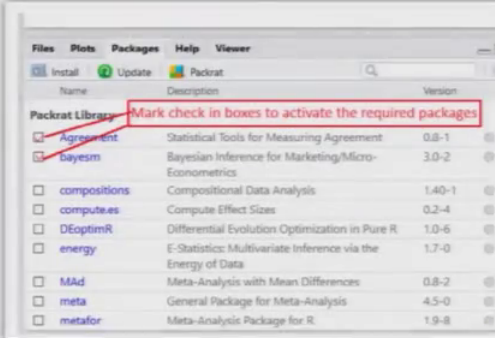
Description of Window 4 : Output window

**Packages:**

All the packages being installed appear here.

Packages are not active.

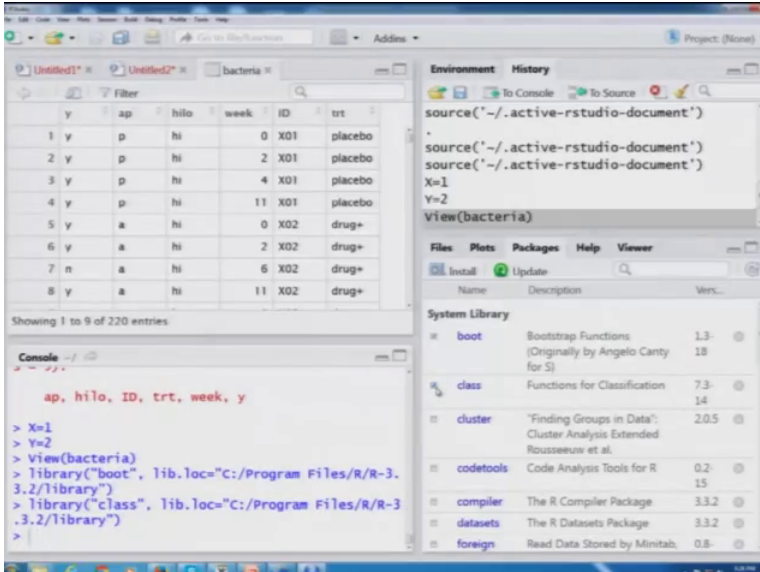
Check mark in the boxes to activate them.



Name	Description	Version
Agreement	Statistical Tools for Measuring Agreement	0.8-1
bayesm	Bayesian Inference for Marketing/Micro-Econometrics	3.0-2
compositions	Compositional Data Analysis	1.40-1
compute.es	Compute Effect Sizes	0.2-4
DEoptimR	Differential Evolution Optimization in Pure R	1.0-6
energy	E-Statistics: Multivariate Inference via the Energy of Data	1.7-0
MAd	Meta-Analysis with Mean Differences	0.8-2
meta	General Package for Meta-Analysis	4.5-0
metafor	Meta-Analysis Package for R	1.9-8

So, let us try to first see here what really happens. For example, here you can see here first you try to click on the first one which is here files. It is trying to give us different detail that whatever are the files which are being used over here. Here if you try to click they will be some details about here the plots, but it still we have not created any plots so, so there is no outcome over here, but we will certainly try to do it.

(Refer Slide Time: 34:37)



The screenshot shows the R Studio interface with the following components:

- Environment pane:** Shows the current environment with variables `X=1`, `Y=2`, and `bacteria`. The `view(bacteria)` function is being executed.
- Console:** Shows the R command prompt with the following code:

```
ap, hilo, ID, trt, week, y
> X=1
> Y=2
> view(bacteria)
> library("boot", lib.loc="C:/Program Files/R/R-3.3.2/library")
> library("class", lib.loc="C:/Program Files/R/R-3.3.2/library")
>
```
- Environment pane (System Library):** Lists installed system packages:
  - boot: Bootstrap Functions (Originally by Angelo Canty for S)
  - class: Functions for Classification
  - cluster: "Finding Groups in Data": Cluster Analysis Extended
  - codetools: Code Analysis Tools for R
  - compiler: The R Compiler Package
  - datasets: The R Datasets Package
  - foreign: Read Data Stored by Minitab

Then it is here trying to use, here it is trying to give us the details about the different types of packages which are being available in the software and whatever package you



want to use you simply have to click here. For example, here I can suppose the first name is coming here as a boot this is about the bootstrapping functions, right. So, suppose I want to use here another here function class so I have to simply class it here, say simply click here.

So, now earlier if you try to, so we have learned different types of command to upload the particular library or a particular packages, but inside the R studio they are they can be uploaded just by the use of a click, right. Similarly if you the fourth icon here is help. So, you can see here what this help contains the same website that we had seen in the earlier lectures. And suppose if you want to know about any function or say anything you can simply type here few words. So, for example, if I want to know about histogram, I am trying to type here say histogram and then it is trying to give you here that these are the different help available with the histogram. And simply if I want to play a print here save list over here, you can see the same thing over here right. So, now, let us try to come back to our slides. So, you can see here that this is the same thing which we have seen. So, these are the means you can see here that there are different say install, say update and say and send so on, right, about the packages right.

So, and they remain and then you have to keep in mind that unless and until you take over the packages what you want to use these packages will not remain as active. And if there is additional package that you want that you have to first download it and then it will appear in this list, right. Now we come to the fourth window and we try to see how to use this help.

(Refer Slide Time: 36:44)

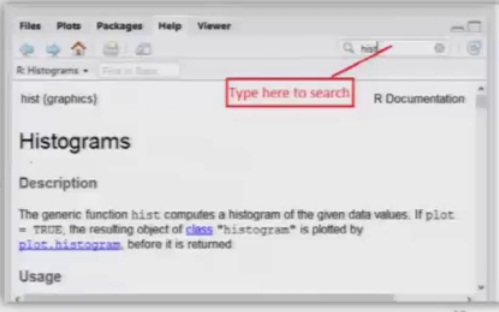
**Introduction to R Studio**  
Window 4 : Output window

**Help:**

Various types of help can be asked.

E.g., to know about histogram, type **hist**.

Information appears.



The screenshot shows the R Studio Help window with the search bar containing 'hist'. A red box highlights the search bar with the text 'Type here to search'. The search results show 'hist (graphics)' and 'R Documentation'. The 'Description' section states: 'The generic function hist computes a histogram of the given data values. If plot = TRUE, the resulting object of class "histogram" is plotted by plot.histogram before it is returned.' The 'Usage' section is also visible.

For example, whatever help you want you have to just type here few words. For example, if I want to know about histogram I will simply type here hist and it is trying to give me here all the details what are available with the histogram. So, I would request you that you please try to type all this command yourself and then try to use some more commands from your common sense and then try to see what do you get, right.

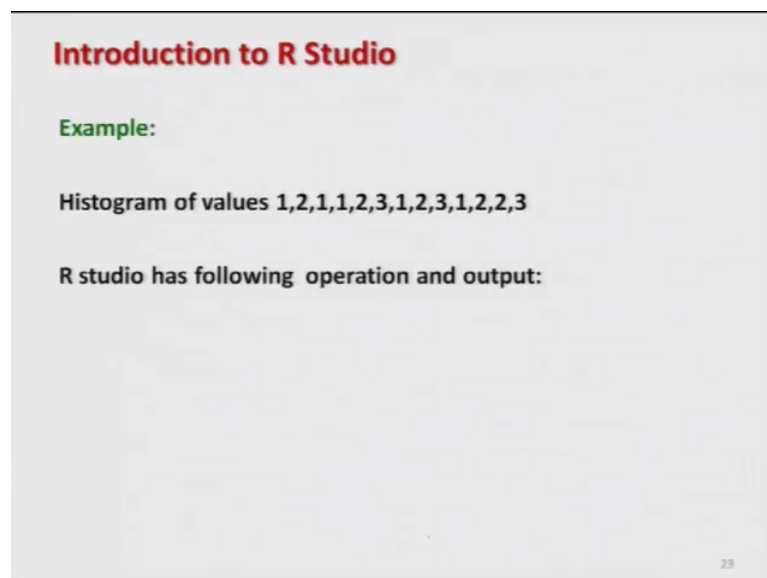
(Refer Slide Time: 37:11)

**Introduction to R Studio**

**Example:**

Histogram of values 1,2,1,1,2,3,1,2,3,1,2,2,3

R studio has following operation and output:

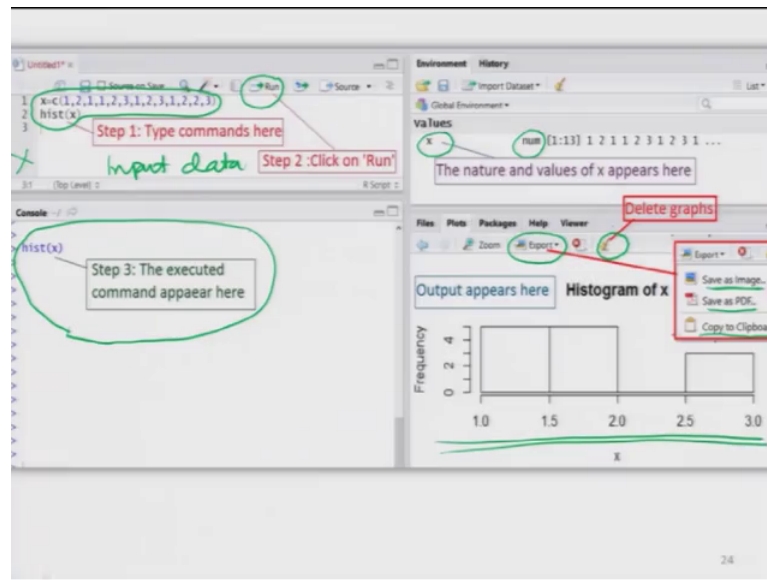


The screenshot shows the R Studio interface with the command 'hist(1,2,1,1,2,3,1,2,3,1,2,2,3)' entered in the console. The output shows a histogram plot. The plot has a title 'Histogram of values 1,2,1,1,2,3,1,2,3,1,2,2,3' and the x-axis is labeled 'x' and the y-axis is labeled 'Density'. The histogram shows the distribution of the data values.

Now, let me take a simple example and try to illustrate the use of all the things whatever I have explained up to now in this lecture. Suppose I have some data which is given over

here which contains 3 different values; 1, 2, and 3 and suppose I have obtained this data and suppose I want to create a histogram, right. So, let us try to first enter this data and we will try to create the histogram.

(Refer Slide Time: 37:44)



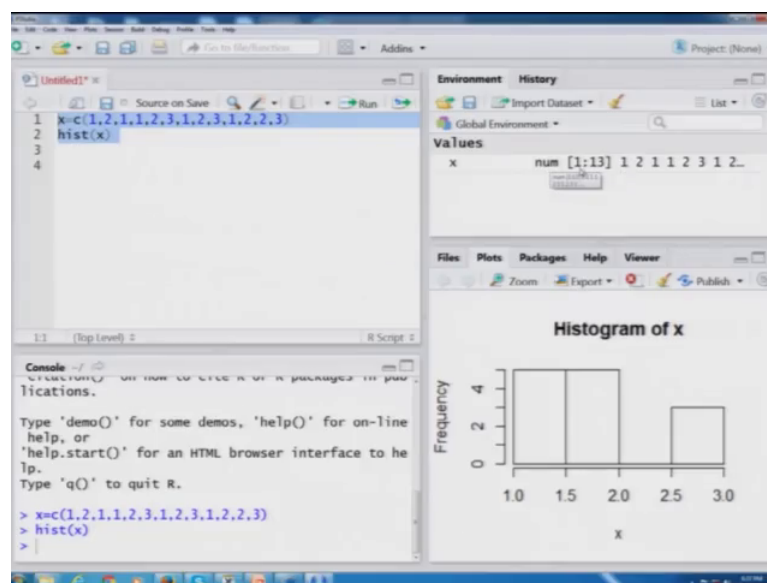
So, what we try to do here therefore, before I try to do something you have to keep in mind that in the first window you have to type here the ins input the data and you have to give the commands over here. So, what I have done here, I have given the data here and the command for creating the histogram like as here like this. For example, I have given all the values inside a variable say here x and then I am trying to create a histogram of here x.

Once I have tried this thing then I will click, I will try to highlight these things and then I will try to click on the icon run, right. And then as soon as I do here run, these things will appear in the window number 2. This I will try to show you and in the window number 3 whatever the values I have stored here inside the variable x they will appear. And it will not only give us the values, but it will also give us the nature of the data. For example, it is showing here num; that means, number these values are number. The other alternatives can be there can be alphabetic or they can be alphanumeric or they can be some logical values and so on, right.

Now, and in the fourth window you will see that this output appears, right. For example, here you can see here a histogram which is appearing in this window.

Now, in case if you want to save this window either you can copy and paste or you can click or export and then you it will ask you to save this as an image for example, jpeg file or say png file or save as pdf or you simply want to copy so that you can paste in paste it in save in some other software and all these things will come over here. And in case if you want to delete this graph you have to simply click over this last icon right.

(Refer Slide Time: 39:58)



So, now let us try to first create this graphic, right. So, if you try to see I have typed here these 2 lines in which I am trying to denote this set of data. And this data is being contained inside a vector, here x. Well you may like to know that what is here c, that we will try to discuss later on, but here I can say inform you that c is a command to combine all the values like this 1 inside 1 vector.

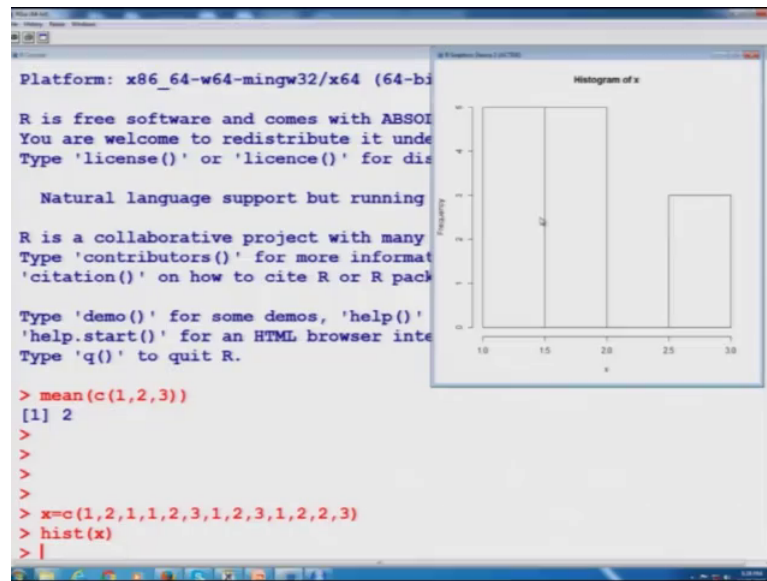
So, what now you have to do you have to concentrate wherever I am trying to move my cursor. So, you see now you have two options either you try to run these 2 lines, one by one or if I try to highlight both the lines together both the lines will be executed at the same time. So, the first step is to write all the programs and now I am trying to run the program. So, I try to come over here. Now as soon as I press here click, you have to keep an eye what happens in the remaining 3 windows and then we will try to discuss it. You

see now I am clicking it 1, 2 and 3, right. Now first let us come to the window number here 2 which is the console window. You can see here which I am highlighting that the same command which was executed from window 1, this appears here. This is equivalent to typing both the commands in the RGUI window and then is executing them and executing them one by one.

Now, in this window number 3 you can see here, you can see here that it is giving me all this information. Please try to concentrate on my cursor. This is here the variable `x`, its nature is being written here `num`; that means this is a number. And here it is trying to give me that this is a vector of 1 by 13 and it is and the values contained in the lecture are, sorry the values contained in these vectors are given here now this is here the fourth window where the histogram is created that you can see over here. Now suppose you want to zoom it, you can zoom it by clicking over here and suppose I want to save it now you can see here I have different options, I can save it as image for example, jpeg file or say png file. I can save it here as say here in the pdf format or I can simply copy it here, something like control C, right. And similarly, if you try to see here, but if I want to remove this graphic from say here, I simply have to click here. And it is asking me are you sure you want to clear all the plots in the history I will say yes, right.

So, now you can see that when I am trying to work with this with this R studio, all the things can be done in the same say screen. Now I try to do the same thing in the R window. So, if you can see here I am simply trying to first enter my data as here `x` and then I am trying to type here the command to create the histogram.

(Refer Slide Time: 43:42)



So, I have to type it again here and then you will see here this histogram appears here. But you can see here that now if you want to know that what is here x, you can type here x and it will give you that what is here x value but inside the R studio all this information that can be viewed in a single say screen.

So, that is the advantage of working with R studio, right. So, now I will stop here and I would request all of you that you please try to do some experiments with the softwares on the same lines which I have explained you. And try to get acquainted with the basic fundamentals and introductory part of this lecture. Now from the next lecture we will try to see how to do different types of calculation in the R software.

Till then goodbye.