**Day 1 - Getting started with R**

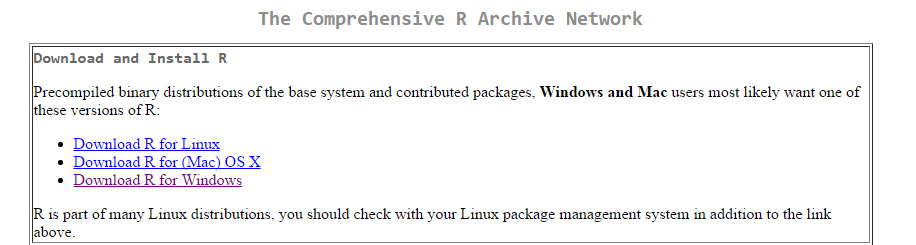
**Installing R and R Studio**

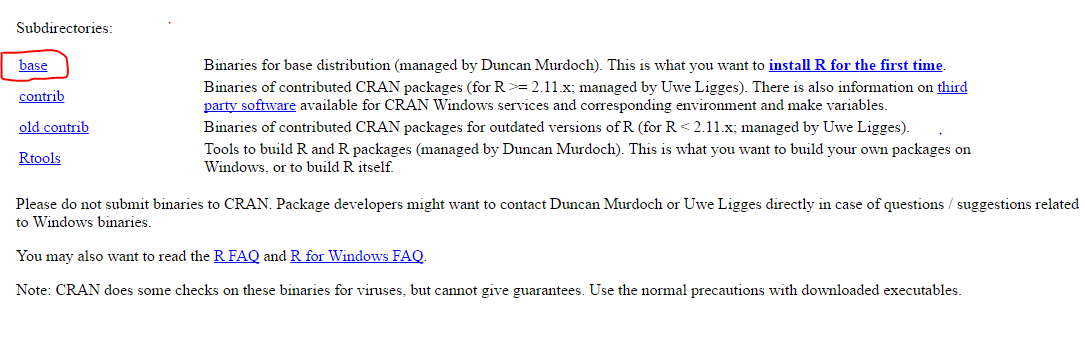
Before starting the course, let’s begin by Installing R and R studio. Installing R is a very easy process. We will show you how to install R for 3 commonly used platforms, Windows, Linux and Mac.

**Windows**

First we start by installing R. Load up your web browser and go to the following link - <https://cran.r-project.org/>. This is the link to Comprehensive R Archive Network, or CRAN. It contains a lot of documentation related to R and it packages, as well as, the installation files. We will talk about this later, but remember that this is the website you will visit often while working on R.

At the top you'll see that there are three options. There's Linux, a Mac and Windows. So you can go to the Windows version here.

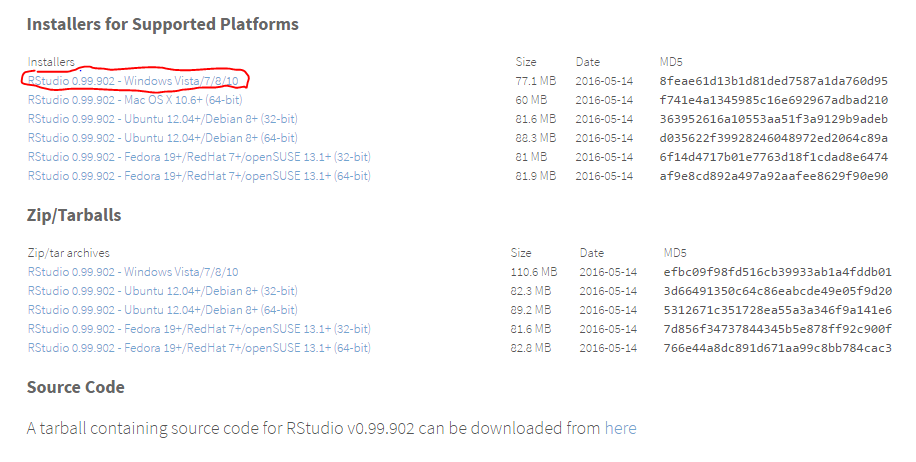


After clicking on Windows link you will see the following page. Click on base as shown in the following picture. Then you will get a download link for R. It should be around 60 megabytes. 

Once you download the file, just click on it and follow the instructions to install. I would recommend choosing the default options for the installation, so just blindly click “next” till the installation is completed. Since you will be using R Studio for doing your assignments you really don’t need to waste your time configuring R. Default options will suffice.

There is one thing that comes up during installation I want to bring to your notice. That is whether you want to install 64-bit or 32-bit version of R. You don’t have to pay much attention to it as far as this course is concerned. But if you are particular about it, 64-bit version is better than 32-bit one in regards that it can make use of more RAM space, which in turn will be useful when you want to deal with large amounts of data. So if your computer is 64-bit you might want to go for 64-bit version. If you don’t understand what I just said, do not worry. It is not that relevant to the course in any way.

Now we are ready to install R Studio. Go to the following link to download R Studio - <https://www.rstudio.com/products/rstudio/download/>. Once you open this link in your browser, just scroll down the page and you will find the following download links.



Download the first option, as marked in the picture. It will automatically start downloading the file. Once the download is completed click on the installer to begin installation. There aren’t many options for installing, so just click on “next” few times and then install. This is it! We are now done with the installing part.

**What is R?**

Since you have decided to take this course you might already have a pretty good idea of what R is, but let me explain it again to keep everyone on the same page.

R is a programming language and software environment for statistical computing and graphics. It is open source software, which means it is free to use, modify and redistribute as you like. This is one of the reasons for the increased popularity of R in recent times. It has a huge user base in both academics and industries. This course will get you started into the wonderful journey of R programming.

R is an ***interpreted language***. It means, users typically access it through a command-line interpreter, or command prompt. If a user types 2+2 at the R command prompt and presses enter, the computer replies with 4, as shown below:

**>** 2+2

[1] 4

Aside from the interactive command prompt we can also write scripts, i.e. a series of such commands and save it as a file and run that all together.

When you enter expressions into the R console (or run an R script), a program within the R system, called the ***R interpreter***, executes the actual code that you wrote. When you type 2+2 and press enter in command prompt the computation of 2+2 is done by R interpreterin the background and gives out the result 4. When you write a script and execute it, each and every line is executed in order.

If you are familiar with other programming languages like C or C++, you would know that those languages don’t have an interpreter, or command prompt where we can just type 2+2 and get result 4. There we just write code in a file and execute it. After executing the code is compiled into machine code and then the machine code is run by computer. These kinds of languages are called ***compiled languages***, since we first compile and then run. If you are not familiar with C or C++, do not worry. I am just making a comparison to show how R is different from the commonly used language such as C.

**Why R?**

It is easy, as well as, natural to use. In just a few lines we can complete data analysis. The reason for this is because R is developed by statisticians exclusively for the purpose of data analysis.

It also has nice graphics and visualizations. One of the design principles of R was that visualization of data through charts and graphs is an essential part of the data analysis process. As a result, it has excellent tools for creating graphics.

It is free and has a huge user base of around 2 million in both academia and industry alike. Because of its open source nature new features are developed by lot of users and programmers in R community. It’s not an exaggeration to say the possibilities with R are unlimited. With R, you're not restricted to choosing a pre-defined set of routines. You can use code contributed by others in the open-source community, or extend R with your own functions. And R is excellent for "mash-ups" with other applications.

It is easy to re-run previous work and make adjustments with R. Often times data analysis is a long process, hence re-running your previous code easily is a very useful feature. You can also make neat documents very easily and even publish them online for free.

**Brief History of R**

R originated from statistical programming language called S developed by John Chambers at Bell Labs in 1970s. The name was chosen because R is prefix to S. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand in 1991. They developed it because they wanted better statistical software for their Macintosh teaching laboratory. Their experience developing R is documented in a 1996 JCGS pager. It’s first announced to public in 1993. In 1995 R is made into free software under GNU General Public License. In 1996 public mailing list is created(R-help and R-devel). In 1997 R Core Group is formed. This group is responsible for the development and maintenance of R since then.

Due to its open source nature and increased requirement for data analysis R has gained huge popularity in recent times.

**Design of R System**

The R system can be divided into 2 conceptual parts.

1. The “base” R system you initially downloaded and installed from CRAN
2. Everything else

The base R system contains, among other things, the ***base*** package which is required to run R. It also contains the most fundamental functions. (If you are confused by the word package, they are just collections of R functions, data, and compiled code in a well-defined format.)

There are other packages like ***utils, stats, datasets, graphics, grDevices, grid, methods, tools, parallel, compiler, splines, tcltk, stats4*** included in base system. Then finally there are some recommended packages: ***boot, class, cluster, codetools, foreign, KernSmooth, lattice, mgcv, nime, rpart, survival, MASS, spatial, nnet, Matrix.***

Don’t get scared by the huge list of unexplained packages I listed. One thing you need to take away from this section is that R consists of a base system which we first downloaded and installed. That base system includes lot of packages which are essential to run R. The R Core group (formed in 1997) controls and oversees the source code for base R system.

Then there is everything else, to be more specific, the packages created by various users and programmers of R community. These packages serve as extinction to the base R system. Since R is an open source software, lot of users have created many useful packages which simplifies a lot of things you might do while programming in R.

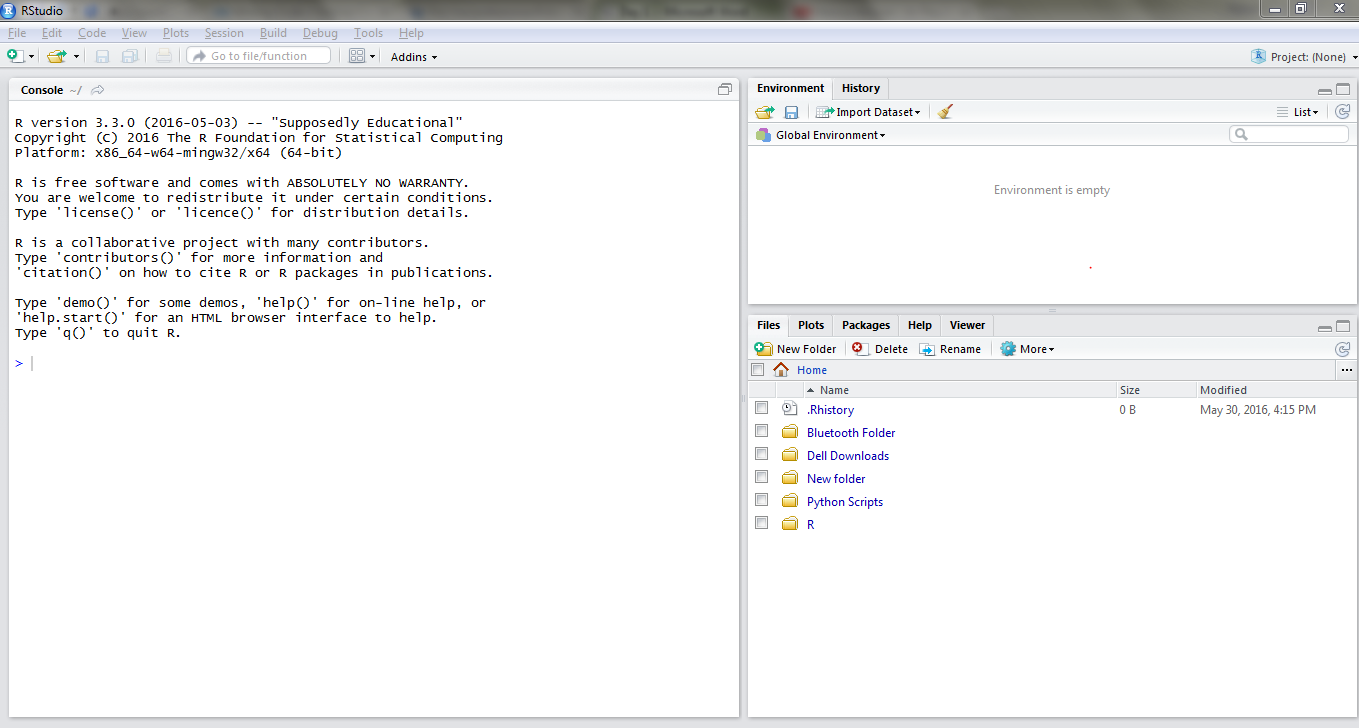
The documentation of these packages is present in CRAN website from where we downloaded R. CRAN has number of restrictions and standards that has to be followed in order to get the package on CRAN. These standards ensure that the packages are safe, easy to use and well documented.

By the end of the course you will learn how to use several widely used packages yourself and experience first-hand how these packages help enormously while programming in R.

**Getting Started with R Studio**

Before diving into more technical aspects, I will show how to get started with, as well as, configure R Studio. I will do it in windows platform, but it will be similar on Mac or Linux platforms.

When you first open R studio it looks like this. On the top of the window you can see the following tabs like - File, Edit, Code, View, Posts, Session, Build, Debug, Tools, and Help.



**R Console**

On the left-hand side you see the console window. This is the command prompt we talked about earlier. In your R console, you'll always type commands after the arrow or "greater than" sign. Try typing 2+2 in the console and press “enter”. The R interpreter will calculate 2+2 and give you 4 as result. Here in this console you can type a wide variety of commands to get useful results. For example you can do any arithmetic operation in R console. Try using multiplication using “\*” (2\*3 will give 6), division using “/” (8/2 will give 4) and power using “^” (2^3 will give 8).

If you type something in your R console but don't finish it properly. For example, try typing 2 and the caret symbol (^), and hitting Enter. R will show you a plus sign and will wait for you to finish the command. You can either finish the command in this case, by typing a number or you can hit Escape, and R will take you back to the arrow sign. So if you ever see a plus sign while working in R, it means that R is waiting for you to finish the line. If you're not sure how to finish it, you can always just hit Escape.

A nice feature of the R console is that you can scroll through your previous commands by using the up and down arrows on your keyboard. If you hit Enter, you can run this command again. You could also adjust previous commands, and hit Enter to run the adjusted comment. We'll use this approach to re-run or adjust commands many times in this course.

You can all view the history of commands you typed previously using ***“Ctlr + Up”*** for Windows or ***“Command + Up”*** for Mac.

Try typing “pi” in console, you will 3.141593 as result. This is the mathematical constant pi. We can use this pi in trigonometric functions. Try calculating values of sin(pi), cos(pi), tan(pi). For sin(pi) and tan(pi) you will get a crazy result 1.224606e-16. (If you are not aware of this notation it means 1.224606\*e^-16.) It’s an extremely low value which can be approximated as zero. As you already might know sin(pi) and tan(pi) value is zero. Sometimes in R you will find this strange value, just remember this can be approximated to zero in those cases.

Your console window might be filled with lot of commands by now. To clean the console use the command ***“Ctrl + L”.*** It is same for Windows, Ubuntu and Mac.

If you are not familiar with working on consoles, please read through this article. [Working in the Console](https://support.rstudio.com/hc/en-us/articles/200404846-Working-in-the-Console). Using the techniques mentioned in this article will make using R console easier. Also refer to [Keyboard Shortcuts](https://support.rstudio.com/hc/en-us/articles/200711853-Keyboard-Shortcuts) to get check out all the keyboard shortcuts for R studio (such as the one we discussed earlier to clean the window).

**Functions and Variables**

Generally, ***R works in terms of functions and variables***. A variable allows you to store values. A function can take in several arguments, or inputs, and returns an output value. An example is the square root, or “***sqrt***”, function. In your R console, type sqrt, and then in parentheses, the number 4, and hit Enter. The function is sqrt, the input, or argument is the number 4, and the output is 2.

There are thousands of functions in R. Some of them are built into R like this one and some can be added in by installing packages, which we'll do several times in this course.

Another example of a function is the abs, or absolute value function, which returns the absolute value of a number. So if we type abs and then in parentheses -65 and hit Enter, we should get the result 65 as our output.

Suppose we now want to save the output of a function. We can do this by saving it to a variable. In your R console, type SquareRoot4 and then an equals sign, and then sqrt, and in parentheses the number 4. And hit Enter. Now you don't see the output of sqrt(4) because we saved it to the variable named SquareRoot4. You can see the value of a variable by typing its name and hitting Enter. If you type *SquareRoor4* and hit Enter you should see that it takes the value 2.

SquareRoot4 is a name that we created, and we could have named it many other things. Generally, you have a lot of freedom in naming your variables, but there are a couple basic rules. One is that you should not use spaces in variable names. If you want to easily separate words in a variable name, popular strategies are using a mix of capital and lowercase letters or to separate the words using periods. Another basic rule is that you should not start variable names with a number. ***Keep in mind that variable names in R are case-sensitive. Capital and lowercase letters are seen differently by R.***

When we created our variable SquareRoot4, we used the equals sign for assignment, or to assign the value of sqrt(4) to the variable named SquareRoot2. You could instead use the following sign ***"<-"***.

For example, let's create a new variable called HoursWeek, and then type the less than sign and a dash, followed by 7 times 24. If you hit Enter and then type HoursWeek to look at its value, you should see 168 in output.

By doing this, the less than sign and the dash did the exact same thing as it would have done if we used the equals sign. In this course, we'll typically use the equals sign. But if you see this less than sign and a dash used, keep in mind that it means the same thing as an equals sign.

Lastly, if you type “***ls()”*** in your R console and hit Enter, you should see a list of all of the variables that you've created in your current R session. Here, we've created two variables HoursWeek and SquareRoot2. This can be useful if you forget exactly how you typed a variable name.

These are our basic building blocks in R. Functions like square root and absolute value, and variables, like HoursWeek and SquareRoot4.

> SquareRoot4 = sqrt(4)

> SquareRoot4

[1] 2

> HoursWeek <- 7\*24

> HoursWeek

[1] 168

> ls()

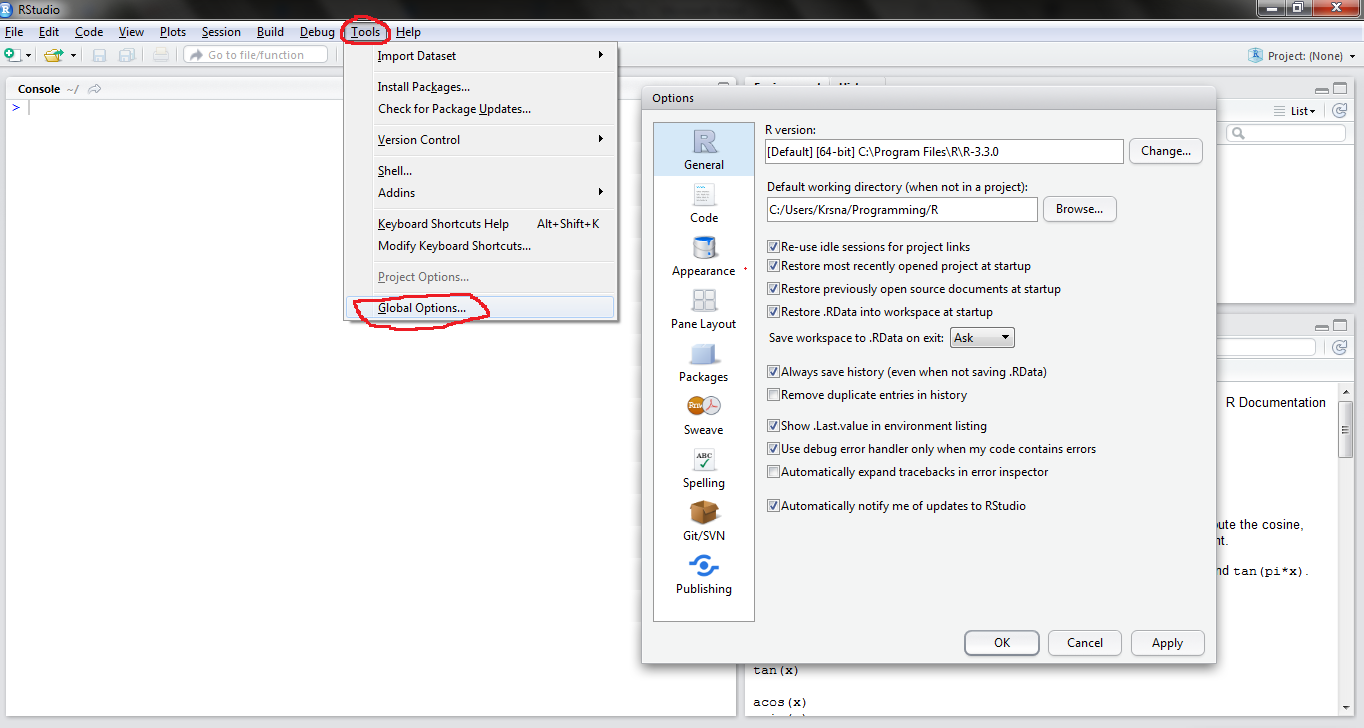
[1] "HoursWeek" "SquareRoot4"

**Working Directory**

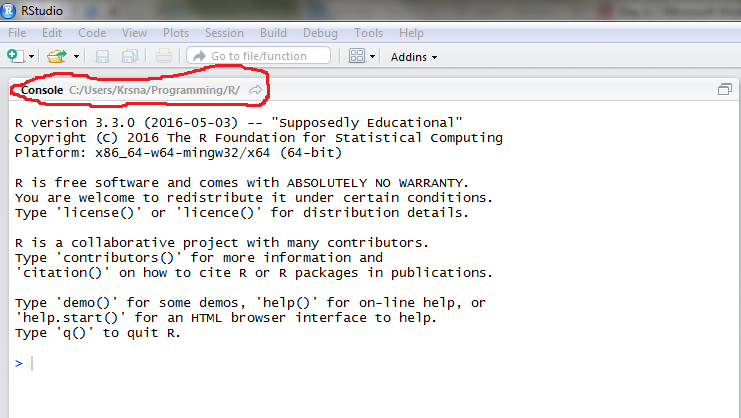
When working on R Studio you would want to do is set default working directory (folder). The working directory is basically the folder in which you are currently working on, i.e. if you save a file from R Studio it will be saved in that folder. By setting a default working directory whenever you want to work on R Studio, it automatically directs to that default directory. To set default directory go to “Tools” tab on the top, and then choose “Global Options”. Tools and Global Options tab is marked with red circle in the following picture.

Here in global options, you can configure R Studio to your needs. In the right side you find several tabs like General, Code, Appearance, Pane layout, Packages, Sweave, Spelling, Git/SVN, and Publishing. When you open global options the general tab is automatically selected. On the top there is the version of R which R studio is currently using is listed. You can have different versions of R installed in your computer, by using change button you can change versions. Since you just installed R you won’t need it right now. But just in case remember that you can do it.

Below version, you can change the default working directory. Choose a convenient directory for you and click on “Apply” in the bottom. Only after clicking “Apply” the changes you made will be saved.



Once you set default directory you can see that directory path beside console, as shown in the following picture.



Try using the command ***getwd()*.** This will give you the current working directory. By using ***setwd()*** you can set working directory. getwd() and setwd() are some of the several functions which we use regularly in R. Here is how you use setwd() –

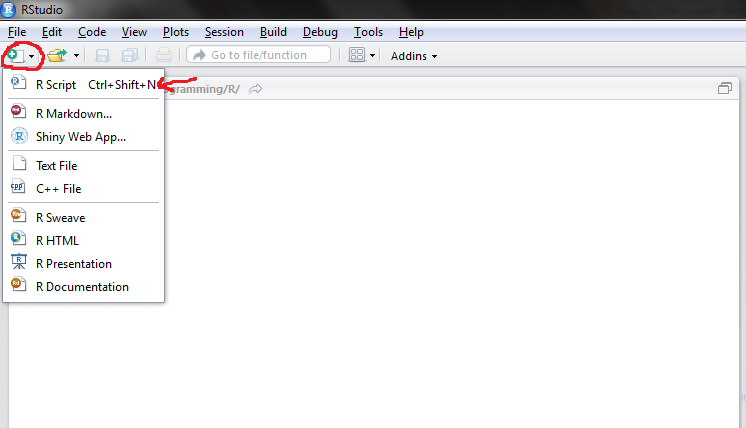
**>** setwd(“C:/Users/Krsna/Programming/R”)

> getwd()

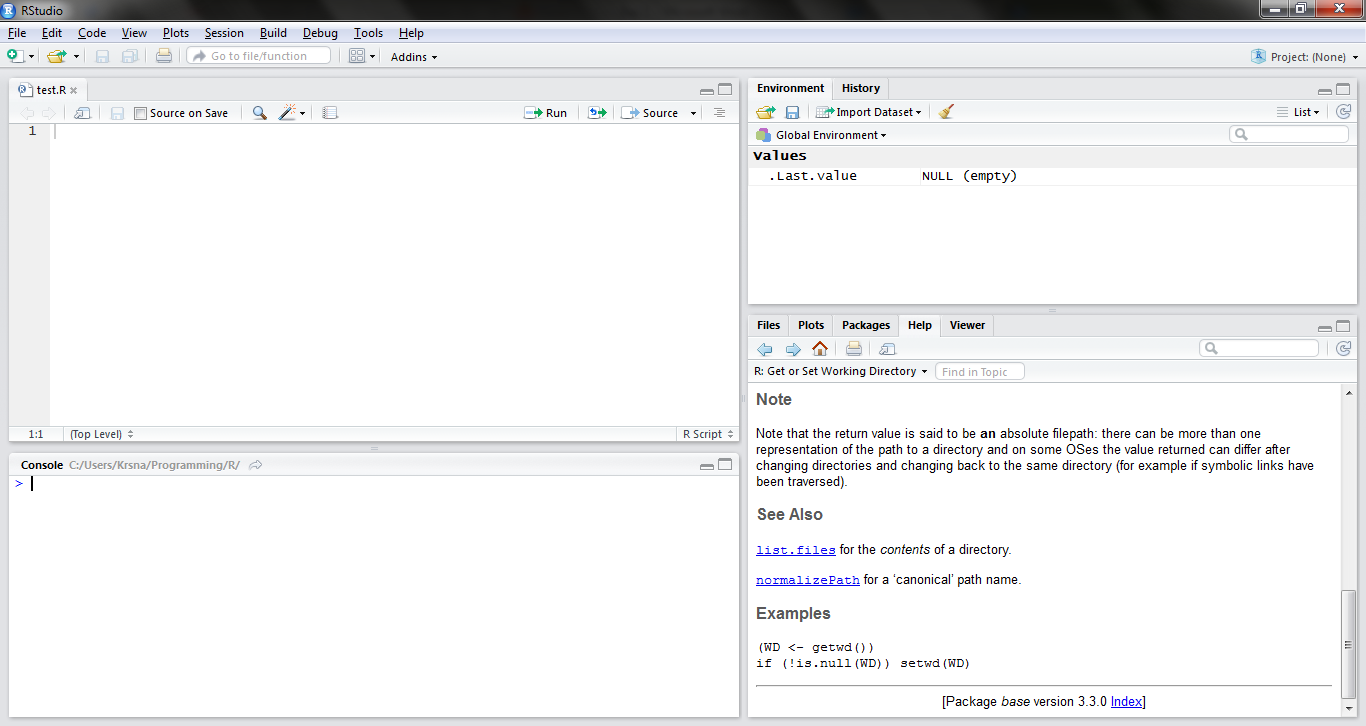
[1] "C:/Users/Krsna/Programming/R"

**R Scripts**

Now let us create a new file to write R code. On the left corner you can find an icon marked in the picture. Select the first option, R Script.



After you select it a new ***code editor*** window will open as shown in the following picture. To save this file press ***“Ctrl + S”*** on Windows and ***“Command + S”*** on Mac and choose a file name.



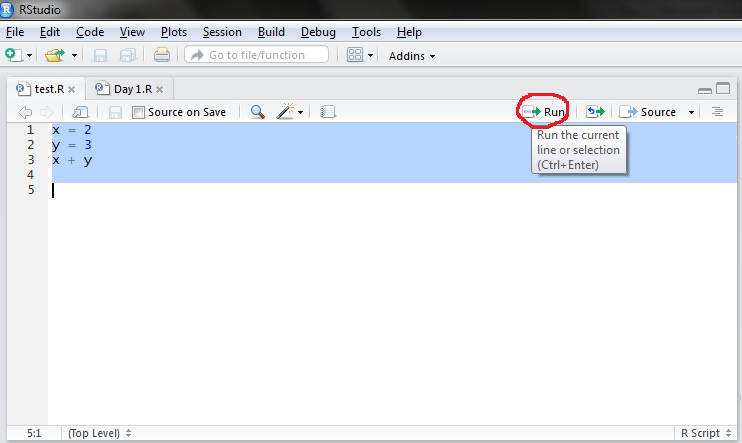
You will learn how to write elegant code in this script files a bit later, but for now just type a few simple likes like this and save the file.

X = 2

Y = 3

X + Y

Now in the editor press ***“Ctrl + A”*** if you are on windows or ***“Command + A”*** if you are on Mac, to select all the lines in the editor. As shown in the next figure after you select all the lines press on the run button on the top to run the code. This selecting and running process might seem a little counter intuitive, but in a way it’s pretty useful. Suppose if you wanted to test a few lines in a 100 lines code you can just select the lines you want to run and run them only, instead of running all the code. We will get to this point again in the course, so if you don’t understand the usefulness of this just yet, don’t worry.



After you run those commands in the console you can see the lines executed and the result. You can find more about how to edit and execute code here [Editing and Executing Code](https://support.rstudio.com/hc/en-us/articles/200484448-Editing-and-Executing-Code).

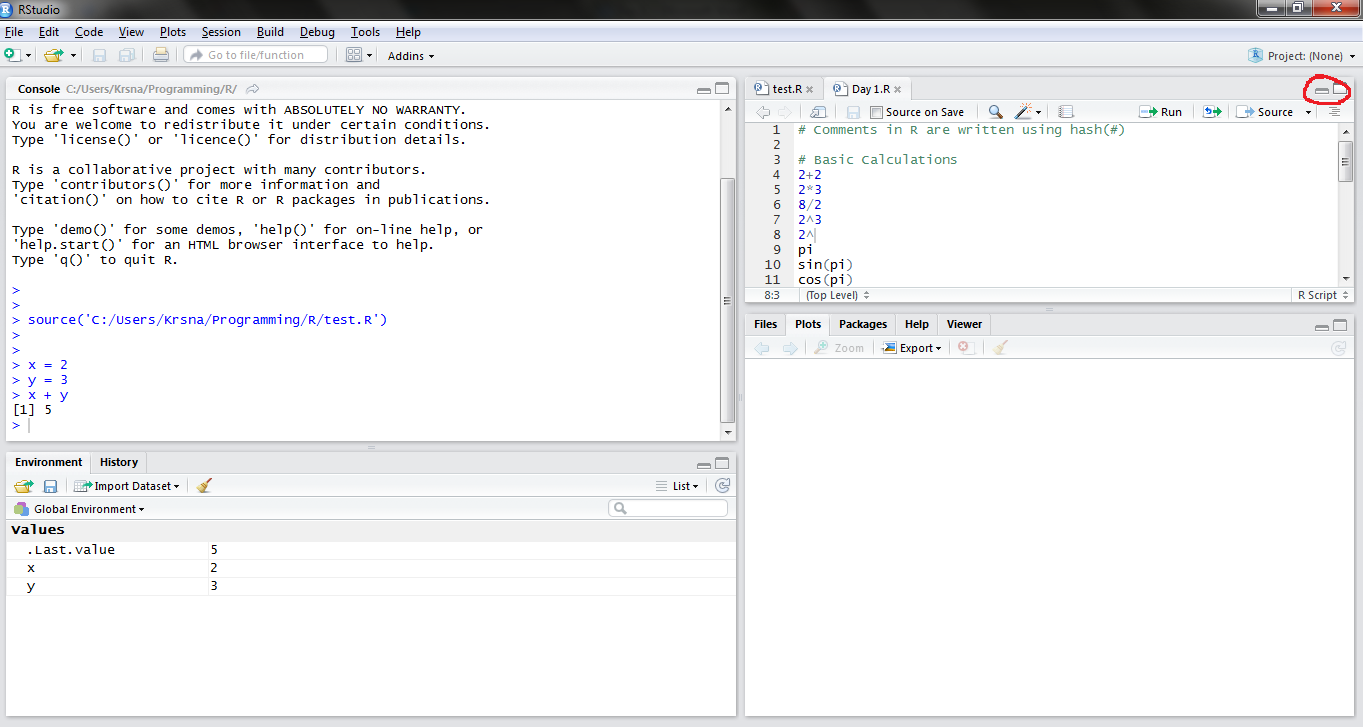
**Comments**

Usually when you write a lot of lines of code, you would want to give comments in your code to understand what the code does. A comment is a programmer-readable annotation in the source code of a computer program. They are added with the purpose of making the source code easier to understand, and are generally ignored by compilers and interpreters. In R you can start the comment with ***“#”*** and anything you type after # will be ignored by R interpreter when you run the code. The comments are there just for the convenience of user/programmer to read and understand code. It is a good practice for a programmer to use comments. The recommended style for commenting in R is having commented lines begin with # and one space. At the end of each day we will give you a script containing all the commands we used in the class. In today’s file is Day1.R you could see the recommended style for commenting and how having a proper commenting can help readability.

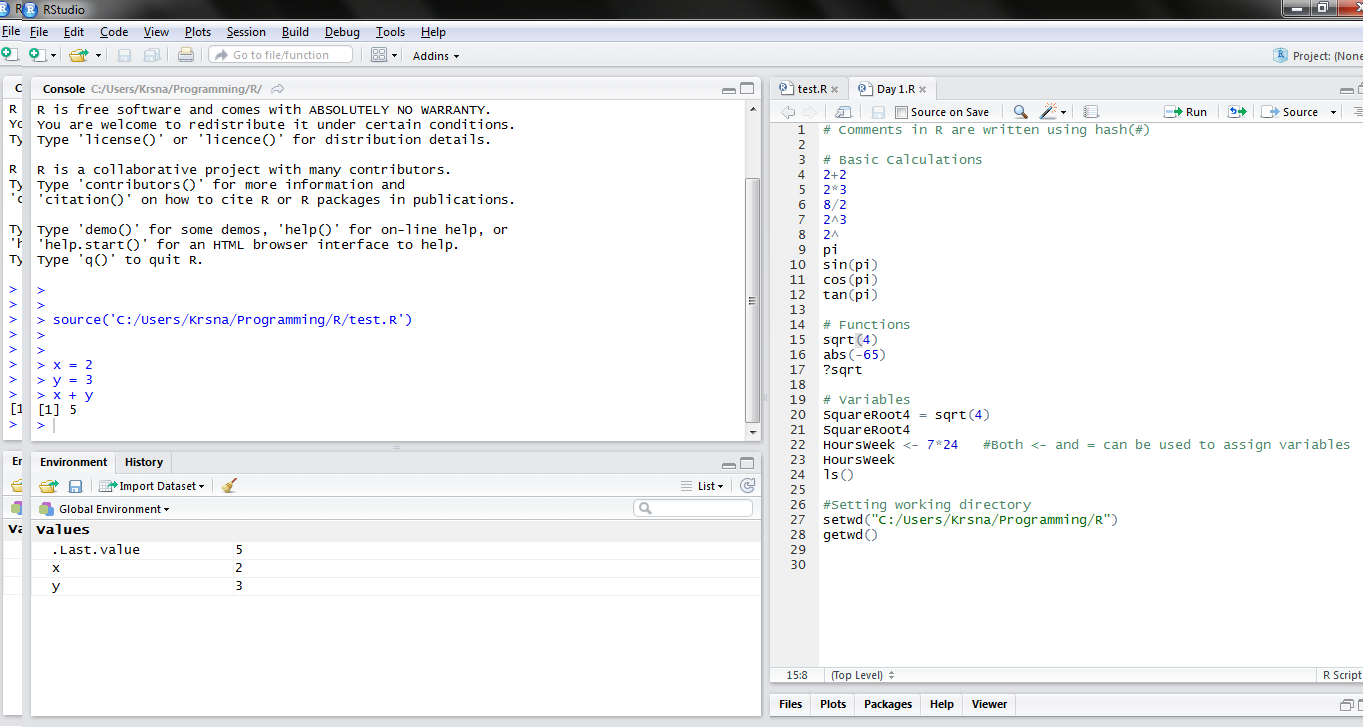
**Customizing R Studio**

If you want to see how to customize your R Studio looks, refer to this link [Customizing R Studio](https://support.rstudio.com/hc/en-us/articles/200549016-Customizing-RStudio). It covers all customization options in depth.

One thing I would recommend looking in these customization options is the ***Pane Layout*** option, you can rearrange the console and editors windows as you like with this option. Here in the following picture, I will show you how I usually customize my layout. Using the buttons at the right corner of each panel as marked in the picture, you can minimize or maximize the panels.



Usually when working on editor you can maximize it like this.



In the left bottom you can see environment variables tab, here all the variables you have created are listed. You could now see the X and Y variables we created earlier. The history tab will show the history of commands you typed.

On the right bottom there is Files tab when you can see all the files in your current working directory. Plots tab shows the plots you create. Packages tab will have a list of all the packages you installed. You can browse through those package and update them if necessary. Help tab will open up whenever you use ? or help command.

**Getting Help**

You can get help on any function in R by typing a question mark and then the function name. So if we type “***?sqrt”*** and hit Enter, you should see the R Help Page for the sqrt function on the right side. You can also use the help feature as ***help(sqrt).***

The help pages are often very useful, and if you want to learn more about a function, you should refer to the help page in R. Try to get familiarized by the format of this description in help, because you are going to read through these documents a lot. In the end of the description there are few examples demonstrating how to use the command, often times this this is the most useful part in the help description.

We also have the ***help.search()*** function to do a search engine type of search. We could use the ***??*** operator for this. Try doing the following and see what happens - ??histogram.

**Installing packages**

As we discussed earlier, R system is divided into 2 logical components. One is the base R, which you first installed from CRAN website. The second is the R packages made by community of R users. In this section you will learn how to install packages in R.

In the later parts of the course we extensively cover a package called ggplot2 which is widely used among R user community for plotting. To install this package, just run the following command ***install.packages("ggplot2")*** in your R console. That’s it! The package will be installed in a couple of seconds. To use this package you need to import it first, for that just type ***library(“ggplot2”)*** in your console. After importing the ggplot2 package, just type the following line in you R console. You won’t understand the command just yet, copy-paste it in your R console.

ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) + geom\_point(size=3)

After executing this command a plot will pop up on the right side. Isn’t it amazing that with just a single command we are able to generate a colorful plot.

**Summary**

* R is an interpreted programming language used for statistical computing and graphics.
* R system can be divided into 2 conceptual parts. The base system and packages made by users.
* Building blocks of R are functions and Variables.
* Commands to remember –
  + ***help (or ?)*** to get help
  + ***Ctrl + L*** (Windows) or ***Command + L*** (Mac) to clean window
  + Up and Down to scroll through previous comments.
  + To popup command history use ***Ctrl + Up*** on Windows or ***Command + Up*** on Mac.
  + ***setwd(“<--directory path-->”)*** to set working directory. (Give directory path in the quotes, this is the notation I will follow in this course.)
  + ***getwd()*** to get working directory
  + ***Install.packages(“<--package name-->”)*** for installing packages and to import the package use ***library(“<--package name-->”)***

**Resources**

* If you are new to working on console, please read through this material - [Working in the Console](https://support.rstudio.com/hc/en-us/articles/200404846-Working-in-the-Console). Using the techniques mentioned in this article will make using R console easier.
* On how to use R Studio code editor for writing scripts. [Editing and Executing Code](https://support.rstudio.com/hc/en-us/articles/200484448-Editing-and-Executing-Code)
* Link for Keyboard shortcuts – [Keyboard Shortcuts](https://support.rstudio.com/hc/en-us/articles/200711853-Keyboard-Shortcuts)
* Customizing R Studio – [Customizing RStudio](https://support.rstudio.com/hc/en-us/articles/200549016-Customizing-RStudio)
* [R Bloggers](http://www.r-bloggers.com/) - A community blog where a lot of experienced R users will share articles about recent updates in R, tutorials, or sometimes even job offers. Be sure to bookmark this blog and visit it occasionally if you are interested in learning more about R.
* [Stackoverflow R community](http://stackoverflow.com/questions/tagged/r) - You can ask questions here in case you get stuck while working on your future projects in R.

**Exercises**

Since there was a lot of installing and customizing today, which is honestly quite boring, there isn’t going to have many exercises. Complete all the following tasks listed below to make your journey from here on smoother.

1. Read the description of the following functions using ? or help command and understand them. In addition to learning what these functions do this exercise will also help you get used to reading help documentations.
   1. print
   2. cat
   3. ls
   4. rm
   5. abs
   6. floor
   7. ceiling
   8. log
   9. logx
   10. exp
   11. list.files
   12. letters
   13. LETTERS ( Remember R is case sensitive)
   14. q
2. Make sure you look into these following links and get to know R studio better.
   1. [Working in the Console](https://support.rstudio.com/hc/en-us/articles/200404846-Working-in-the-Console)
   2. [Editing and Executing Code](https://support.rstudio.com/hc/en-us/articles/200484448-Editing-and-Executing-Code)
   3. [Keyboard Shortcuts](https://support.rstudio.com/hc/en-us/articles/200711853-Keyboard-Shortcuts)
   4. [Customizing RStudio](https://support.rstudio.com/hc/en-us/articles/200549016-Customizing-RStudio)
3. At the end of each day we will give you a script containing all the commands we used in the class. Look at all he commands we used today in the file Day1.R. The script file will be properly commented to make it easy to read and understand. Take a look at the script to revise what we have done in the class.
4. Install couple of packages listed below, we will use this packages later in the session so it will save time installing them later on. If you curious as to what these packages are used for, go to the CRAN documentation page and read about these packages. You can just google any package name (like “dplyr cran”) and find the CRAN link to the package.

* dplyr
* dta.table
* markdown
* rmarkdown
* knitr
* shiny

Also, finally, visit R Bloggers blog, and check out few articles.