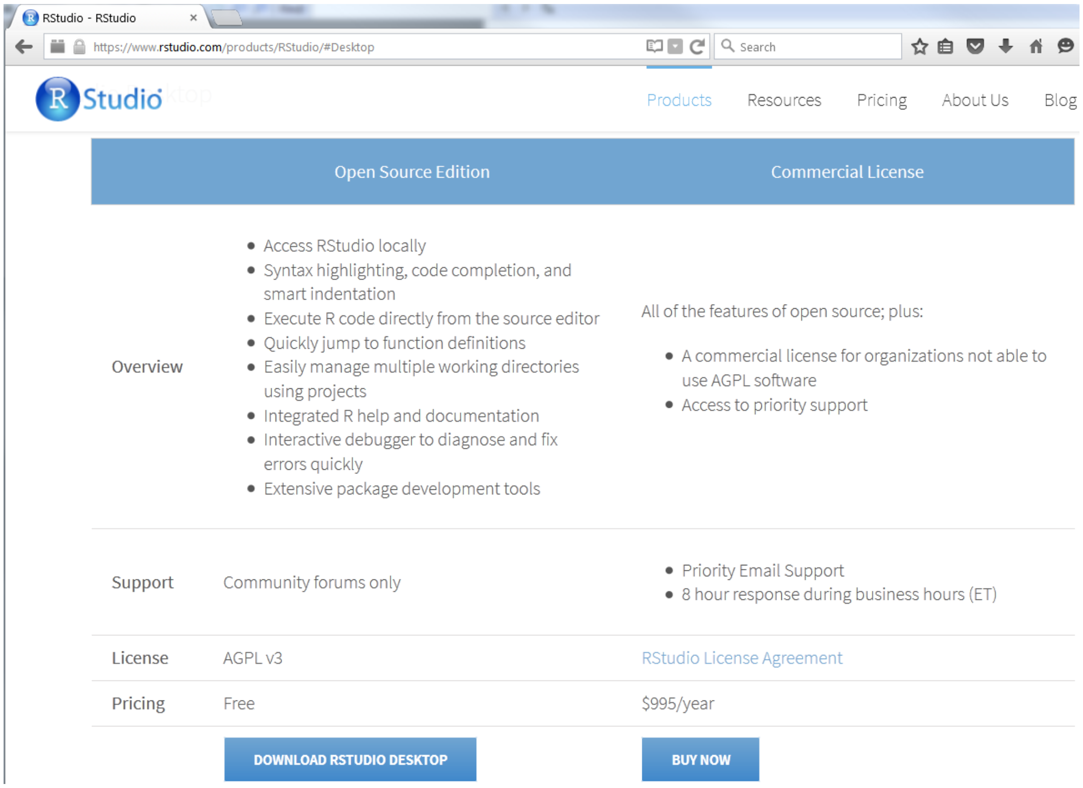
Installing, Preparing, and Introducing RStudio

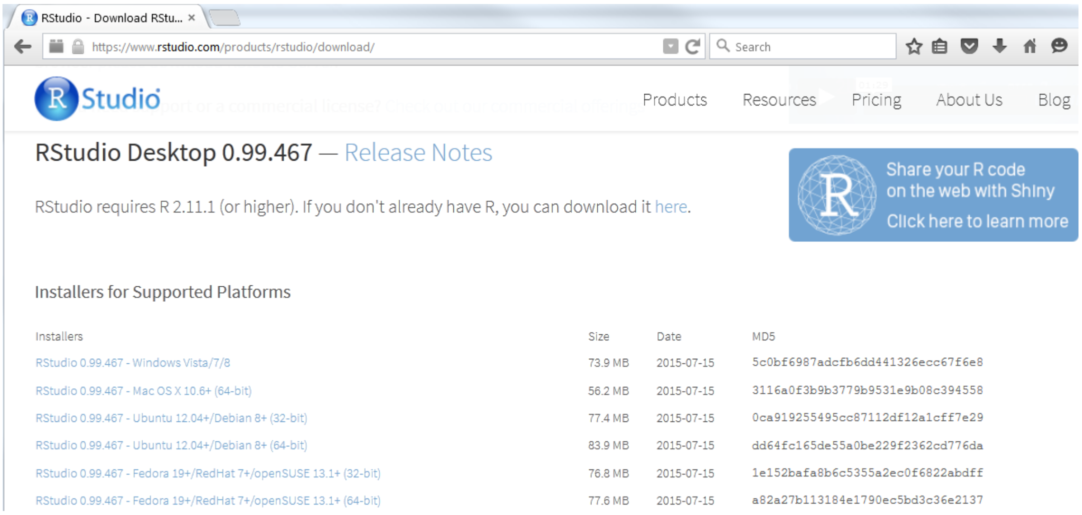
Basic RStudio Installation

1. Go to the R Studio download page at www.rstudio.com/products/rstudio/#Desktop. Press the "DOWNLOAD RSTUDIO DESKTOP" button/graphic (near bottom-left of the image below).



	Open Source Edition	Commercial License
Overview	• Access RStudio locally • Syntax highlighting, code completion, and smart indentation • Execute R code directly from the source editor • Quickly jump to function definitions • Easily manage multiple working directories using projects • Integrated R help and documentation • Interactive debugger to diagnose and fix errors quickly • Extensive package development tools	All of the features of open source; plus: • A commercial license for organizations not able to use AGPL software • Access to priority support
Support	Community forums only	• Priority Email Support • 8 hour response during business hours (ET)
License	AGPL v3	RStudio License Agreement
Pricing	Free	$995/year
	DOWNLOAD RSTUDIO DESKTOP	BUY NOW

2. Select the link from the "Installers for Supported Platforms" list that corresponds to the operating system appropriate for your computer. In the remainder of these directions I will demonstrate the installation for a WINDOWS operating system. Either run the program or note where this executable program is saved on your computer.


RStudio - Download RStu...
https://www.rstudio.com/products/rstudio/download/
Search
RStudio
Products
Resources
Pricing
About Us
Blog
Share your R code on the web with Shiny
Click here to learn more

RStudio Desktop 0.99.467 — Release Notes

RStudio requires R 2.11.1 (or higher). If you don't already have R, you can download it here.

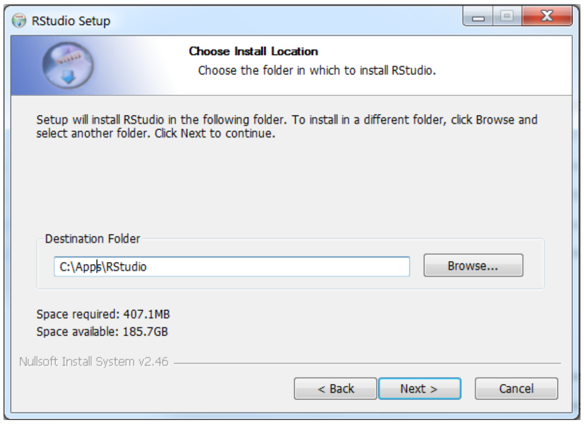
Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 0.99.467 - Windows Vista/7/8	73.9 MB	2015-07-15	5c0bf6987adcfb6dd441326ecc67f6e8
RStudio 0.99.467 - Mac OS X 10.6+ (64-bit)	56.2 MB	2015-07-15	3116a0f3b9b3779b9531e9b08c394558
RStudio 0.99.467 - Ubuntu 12.04+/Debian 8+ (32-bit)	77.4 MB	2015-07-15	0ca919255495cc87112df12a1cff7e29
RStudio 0.99.467 - Ubuntu 12.04+/Debian 8+ (64-bit)	83.9 MB	2015-07-15	dd64fc165de55a0be229f2362cd776da
RStudio 0.99.467 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	76.8 MB	2015-07-15	1e152bafa8b6c5355a2ec0f6822abdff
RStudio 0.99.467 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	77.6 MB	2015-07-15	a82a27b113184e1790ec5bd3c36e2137

3. If you did not run the program, then locate and run the downloaded file (called “RStudio-0.99.467.exe” or similar if the version number has changed).

4. Press “Next” on the first “Welcome” dialog box (depending on your version of Windows you may have received security warnings before this dialog box appears).

5. Select a location to install RStudio (simply use the default location if the location is not important to you – in the dialog box below I installed in a custom directory). Press “Next.”



RStudio Setup

Choose Install Location

Choose the folder in which to install RStudio.

Setup will install RStudio in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

C:\Apps\RStudio

Space required: 407.1MB

Space available: 185.7GB

Nullsoft Install System v2.46

6. Decide whether or not to create a shortcut in the Start Menu folder (I suggest you DO). Press “Install.”

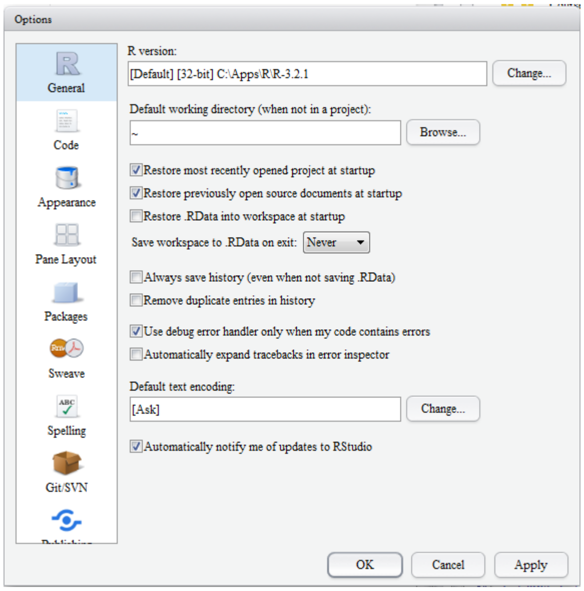
7. RStudio should then begin installing files into the directory you chose previously. If everything goes well then you should get one last dialog box noting such. Press “Finish.”

8. If you did not create a shortcut above then you will need to locate the “rstudio.exe” file inside the “RStudio/bin” folders inside the folder you chose to install RStudio in. On my computer, for example this file is inside of “C:/apps/RStudio/bin”.

Preparing RStudio

1. Open RStudio.

2. Select the "Tools" menu and then the "Global Options" submenu. In the ensuing dialog box select the "General" icon on the left (this should already be selected).



R version:
[Default] [32-bit] C:\Apps\R\R-3.2.1

Default working directory (when not in a project):
~

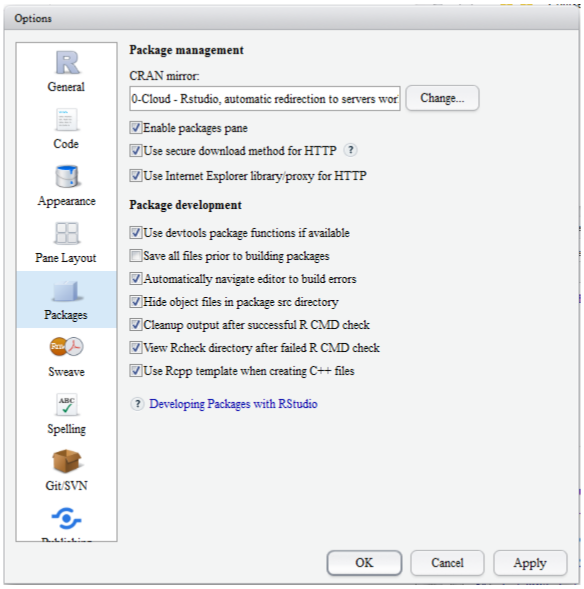
☑ Restore most recently opened project at startup
☑ Restore previously open source documents at startup
☐ Restore .RData into workspace at startup
Save workspace to .RData on exit: Never
☐ Always save history (even when not saving .RData)
☐ Remove duplicate entries in history
☑ Use debug error handler only when my code contains errors
☐ Automatically expand tracebacks in error inspector

Default text encoding:
[Ask]

☑ Automatically notify me of updates to RStudio

- Depending on your installation, the R version should read "[Default][32-bit]" followed by the path to the R program (as shown in the dialog box above). If you installed the 64-bit version of R, then select the "Change..." button and then "use your machine's default version of R64 (64-bit)".
- You can either leave the other selections at their defaults or change them as you see fit (my preferences are shown in the dialog box above). However, I strongly urge you to un-select "Restore .RData into workspace at startup" and make "Save workspace to .RData on exit:" is set to "Never."

3. Select the "Packages" icon in the "options" dialog box opened above. It is useful to set a CRAN mirror in this dialog box. I prefer the "0-Cloud - Rstudio ..." option but you may want to choose a location nearer to you (through the "change" button). All other options can remain at their defaults.



Package management

CRAN mirror:
0-Cloud - Rstudio, automatic redirection to servers wor

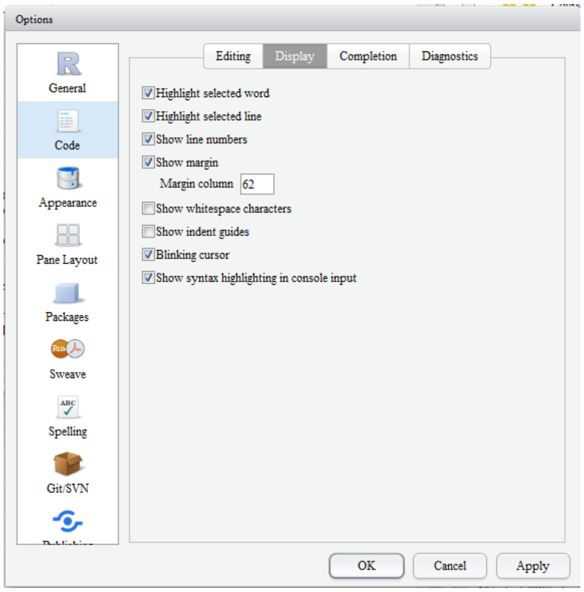
☑ Enable packages pane
☑ Use secure download method for HTTP ?
☑ Use Internet Explorer library/proxy for HTTP

Package development

☑ Use devtools package functions if available
☐ Save all files prior to building packages
☑ Automatically navigate editor to build errors
☑ Hide object files in package src directory
☑ Cleanup output after successful R CMD check
☑ View Rcheck directory after failed R CMD check
☑ Use Rcpp template when creating C++ files

? Developing Packages with RStudio

4. Select the “Code” icon in the “Options” dialog box opened above and the “Display” tab. I suggest, in addition to the default selections, selecting the “Highlight selected line”, “Show margin”, and “Show syntax highlighting in console input.”



☑ Highlight selected word
☑ Highlight selected line
☑ Show line numbers
☑ Show margin
Margin column 62
☐ Show whitespace characters
☐ Show indent guides
☑ Blinking cursor
☑ Show syntax highlighting in console input

5. At times I find the code completion options in RStudio irritating. If you do as well, you can either turn this option off or tweak its settings within the “Completion” tab under the “Code” icon in the “Options” dialog box opened above.

6. No other options need to be set for our purposes. Press “OK.”

Introducing RStudio

What is RStudio?

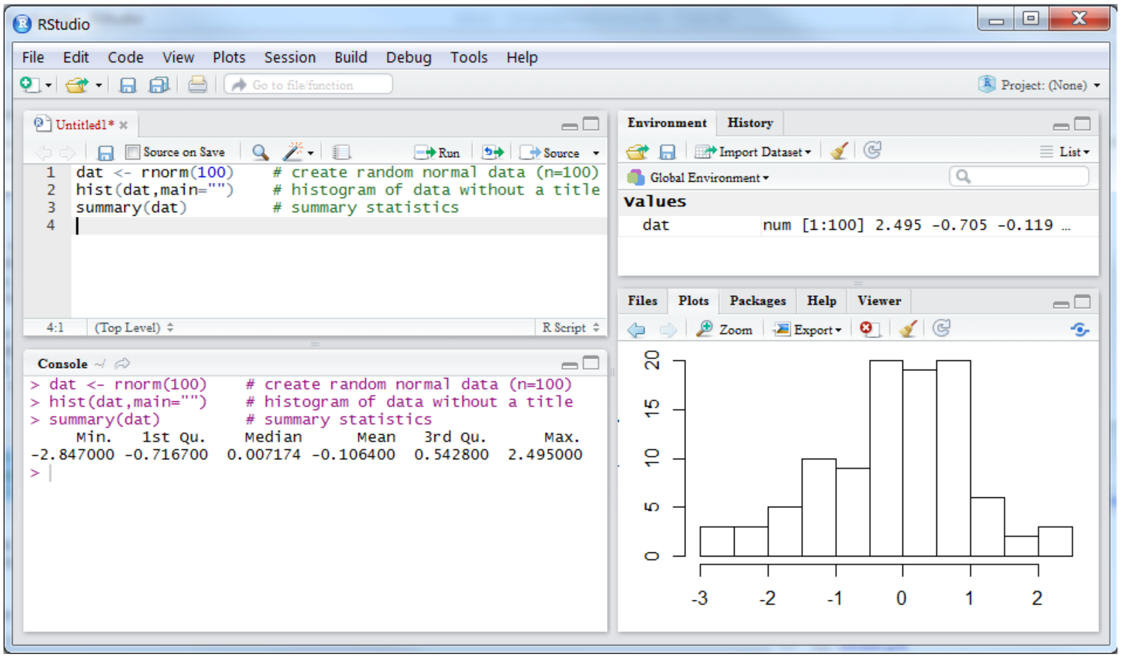
R is an open-source software environment for statistical computing and graphics that runs on Windows, Mac OS, and many UNIX platforms. Unlike many other programs, users interact with R through a command line rather than through a graphical user interface. While such an interface may be unfamiliar to many users, its primary strength is the ability for a user to develop scripts of commands to perform various analyses that can then be easily repeated.

RStudio is an open-source integrated development environment (IDE) that serves as a front-end “on top” of R. RStudio eases the user’s interaction with R by providing some of the conveniences of a GUI and, more importantly, a means for efficiently constructing and running R scripts. Among other conveniences, RStudio provides a four-panel layout that includes a feature-rich source-code editor (includes syntax highlighting, parentheses completion, spell-checking, etc.), a tight link to the R console, a system for examining objects saved in R, an interface to R help, and extended features to examine and save plots.

Using RStudio is easy to learn. This document is a very brief introduction to RStudio.

RStudio Design

RStudio is organized around a four-panel layout (Figure 1). The upper-left panel is the R *Script Editor*. R commands are typed into this panel and submitted to the R *Console* in the lower-left panel. For most applications, you will type R commands into the *Script Editor* and submit them to the *Console*; you will not type commands directly into the *Console*. The *Script Editor* is a high-level text editor, whereas the *Console* is the R program.



```
dat <- rnorm(100)     # create random normal data (n=100)
hist(dat,main="")     # histogram of data without a title
summary(dat)          # summary statistics
```

Values

dat   num [1:100] 2.495 -0.705 -0.119 …

```
> dat <- rnorm(100)     # create random normal data (n=100)
> hist(dat,main="")     # histogram of data without a title
> summary(dat)          # summary statistics
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. 
-2.847000 -0.716700  0.007174 -0.106400  0.542800  2.495000 
> 
```

Figure 1. Example of the RStudio layout with the *Script Editor* in the upper-left, *Console* in the lower-left, the *environment* tab shown in the upper-right, and the *Plot* tab shown in the lower-right panels.

The upper-right panel contains at least two tabs – *Environment* and *History*. Many items listed under the *Environment* tab can be double-clicked to open them for viewing as a tab in the *Script Editor*. The *History* tab simply shows all of the commands that you have submitted to the *Console* during the current session.

The lower-right panel contains at least five tabs – *Files*, *Plots*, *Packages*, *Help*, and *Viewer*. The *Plots* tab will show the high-level plots produced by commands submitted to the *Console*. One can cycle through the history of constructed plots with the arrows on the left side of the plot toolbar and plots can be saved to external files using the "Export" tab on the plot toolbar (Figure 1). A list of all installed packaged is seen by selecting the *Packages* tab (packages can also be installed through this tab as described in a separate document). Help for each package can be obtained by clicking on the name of package[1]. The help will then appear in the *Help* tab.

Basic Usage

Our primary interaction with RStudio will be through developing R scripts in the *Script Editor*, submitting those scripts to the *Console*, and viewing textual or tabular results in the *Console*, and graphical results in the *Plot* panel. In this section, I briefly introduce how to construct and run R scripts in RStudio.

One opens a blank file for an R script by selecting the "New" icon (⊕▾) and then `R Script`; selecting the `File` menu, `New` submenu, and `R Script` item; or with `<CTRL>` + `<Shift>` + `N`. In the newly created *Script Editor* pane, type the three lines exactly as shown below[2].

```
dat <- rnorm(100)       # create random normal data (n=100)
hist(dat,main="")       # histogram of data without a title
summary(dat)            # summary statistics
```

These commands must be submitted to the *Console* to perform the requested calculations. Commands may be submitted to the *Console* in a variety of ways:

- Put the cursor on a line in the *Script Editor* and press the "Run" icon (Run; altenatively press `<CTRL>` + `<Enter>`). This will submit that line to the *Console* and move the cursor to the next line in the *Script Editor*. Pressing "Run" again will submit this next line. And so on.
- Select all lines in the *Script Editor* that you wish to submit and press Run (or `<CTRL>` + `<Enter>`).

The RStudio layout after using the first method is shown in Figure 1.

The R Script in the *Script Editor* should now be saved by selecting the `File` menu and the `Save` item (alternatively, pressing `<CTRL>` + `S`). RStudio can now be closed (do NOT save the workspace). When RStudio is re-started later, the script can be re-opened (choose the `File` menu and the `Open file ...` submenu if the file is not already in the *Script Editor*) and re-submitted to the *Console* to exactly repeat the analyses[3].

[1] Help can also be obtained by typing a question mark and then the name of the package in the console – e.g., `?FSA`.

[2] For the moment, don't worry about what these lines "do."

[3] Note that the results of commands are not saved in R or RStudio; rather the commands are saved and re-submitted to re-perform the analysis.