

ESTTA Tracking number: **ESTTA411750**

Filing date: **05/30/2011**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE TRADEMARK TRIAL AND APPEAL BOARD

Proceeding	91193335
Party	Plaintiff Embarcadero Technologies, Inc.
Correspondence Address	MARTIN R GREENSTEIN TECHMARK A LAW CORPORATION 4820 HARWOOD RD, 2ND FLOOR SAN JOSE, CA 95124-5273 UNITED STATES MRG@TechMark.com, MPV@TechMark.com, AMR@TechMark.com, LZH@TechMark.com
Submission	Plaintiff's Notice of Reliance
Filer's Name	Leah Z Halpert
Filer's e-mail	MRG@TechMark.com, MPV@TechMark.com, LZH@TechMark.com, AMR@TechMark.com
Signature	/Leah Z Halpert/
Date	05/30/2011
Attachments	RSTUDIO-91193335-Embarcadero Notice of Reliance-Rebuttal Period.pdf (3 pages)(44060 bytes) Exhibit A-2.pdf (58 pages)(2268965 bytes)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE TRADEMARK TRIAL AND APPEAL BOARD

EMBARCADERO TECHNOLOGIES, INC.

Opposer

v.

RSTUDIO, INC.

Applicant.

Opposition No.: 91-193,335

Trademarks: RSTUDIO

Serial Nos.: 77/691,980

77/691,984

77/691,987

OPPOSER'S NOTICE OF RELIANCE - REBUTTAL

Pursuant to Trademark Rule 2.122(e) Opposer, EMBARCADERO TECHNOLOGIES, INC., (“Embarcadero”, or “Opposer”), by its attorneys, hereby gives notice that it will or may rely on the following materials relevant to the issues in the captioned proceeding, copies of which are attached to the Notice. All of the following materials are specifically entered into the record in order to rebut testimony and evidence submitted by Applicant during their testimony period.

The Notice of Reliance is being submitted prior to the close of Opposer’s Rebuttal Testimony Period, pursuant to the Trademark Trial and Appeal Board Manual of Procedure (TBMP) § 702.02, *See Sports Authority Michigan Inc. v. PC Authority Inc.*, 63 USPQ2d 1782, 1786 n.4 (TTAB 2001) (notices of reliance must be filed before closing date of party's testimony period).

1. RStudio Inc.’s current website as of May 25 2011, attached hereto as Exhibit A to show the manner in which Applicant has altered the site since their testimony period, and to show the manner in which Application is currently using and/or referencing to its RSTUDIO marks.
2. Working Paper No. 15 from the Meeting of the Management of Statistical Information Systems titled “R: An Open Source Statistical Environment Invited Paper” prepared by Valentin Todorov, UNIDO, dated March 28, 2008, available at:

- <http://www.unece.org/stats/documents/ece/ces/ge.50/2008/wp.15.e.pdf>, attached hereto as Exhibit B. The Working Paper shows the connectivity between the R computing language and relational databases as well as shows that statistical systems are never used in isolation, but rather must communicate with other systems.
3. The online book “Using R for Actuarial Science” by Shyamal Kumar, dated March 5, 2007, available at: www.soa.org/files/pdf/UsingRforActuarialScience.pdf, attached hereto as Exhibit C. The online book shows the connectivity between the R computing language and relational databases.
 4. The online article “Scenarios for Using R within a Relational Database Management System Server” by Duncan Temple Lang, dated April 12, 2001, available at: www.omegahat.org/RSPostgres/Scenarios.pdf, attached hereto as Exhibit D. The article shows the connectivity between the R computing language and relational databases.
 5. Excerpts from the online manual “R Data Import/Export” edited by the R Development Core Team at the Comprehensive R Archive Network (CRAN), Version 2.13.0, dated April 13, 2011, available at: <http://cran.r-project.org/doc/manuals/R-data.pdf>, attached hereto as Exhibit E. This manual shows how the R language is well adapted to work with relational databases, such as ER/Studio.
 6. The article “Improving the analysis, storage and sharing of neuroimaging data using relational databases and distributing computing”, by Uri Hasson, et al, published in *NeuroImage*, a Journal of Brain Function, Volume 39, Issue 2, 15 January 2008, Pages 693-706, available at: http://www.behaviormetrix.com/public_html/Hasson07.distrib.analysis.pdf, attached hereto as Exhibit F. This article shows how the R language is well suited to conduct analysis on relational databases

7. Portions of Embarcadero's current website as of May 6, 2011, as well as excerpts from various ER/Studio user guides, attached hereto as Exhibit G to show the relationship, interface, and/or interoperability between the ER/Studio product line and flat file or non-relational databases.

Dated: May 30, 2011

Respectfully Submitted,

EMBARCADERO TECHNOLOGIES, INC.
By /Martin R. Greenstein/
Martin R. Greenstein
Mariela P. Vidolova
Leah Z. Halpert
TechMark a Law Corporation
4820 Harwood Road, 2nd Floor
San Jose, CA 95124-5273
Tel: (408) 266-4700; Fax: (408) 850-1955
E-Mail: MRG@TechMark.com
Attorneys for Opposer

CERTIFICATE OF SERVICE

I hereby certify that a true and correct copy of the foregoing **OPPOSER'S NOTICE OF RELIANCE - REBUTTAL** is being served on May 30, 2011, by first class mail, postage prepaid on Applicant's Attorney of Record at his address below:

Charles E. Weinstein, Esq.
Julia Huston
Joshua S. Jarvis
Anthony E. Rufo
FOLEY HOAG LLP
155 Seaport Blvd, Ste 1600
Boston, MA 02210-2600
Tel: (617) 832-1000
E-Mail: CEW@foleyhoag.com

/Leah Z Halpert/
Leah Z Halpert

Exhibit A

Introducing RStudio

RStudio™ is a new integrated development environment (IDE) for R. RStudio combines an intuitive user interface with powerful coding tools to help you get the most out of R.



Productive

RStudio brings together everything you need to be productive with R in a single, customizable environment. Its intuitive interface and [powerful coding tools](#) help you get work done faster.

Runs Everywhere

RStudio is available for [all major platforms](#) including Windows, Mac OS X, and Linux. It can even run alongside R on a server, enabling multiple users to access the RStudio IDE using a web browser.

Free & Open

Like R, RStudio is available under a [free software license](#) that guarantees the freedom to share and change the software, and to make sure it remains free software for all its users.



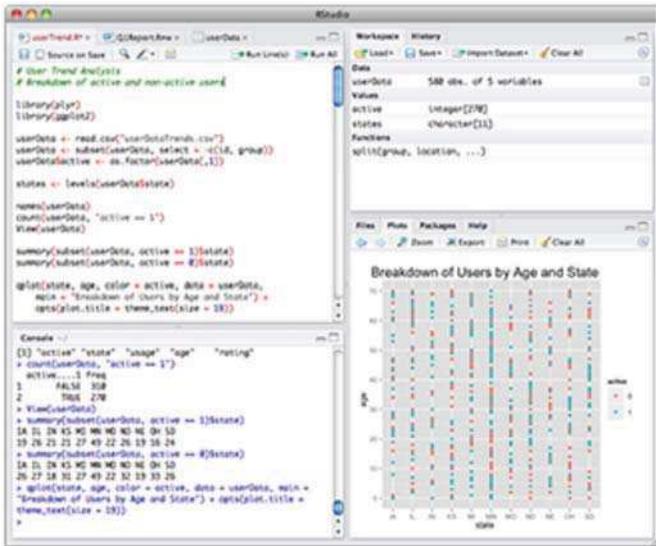
News

RStudio v0.93 Available (4/11/2011)

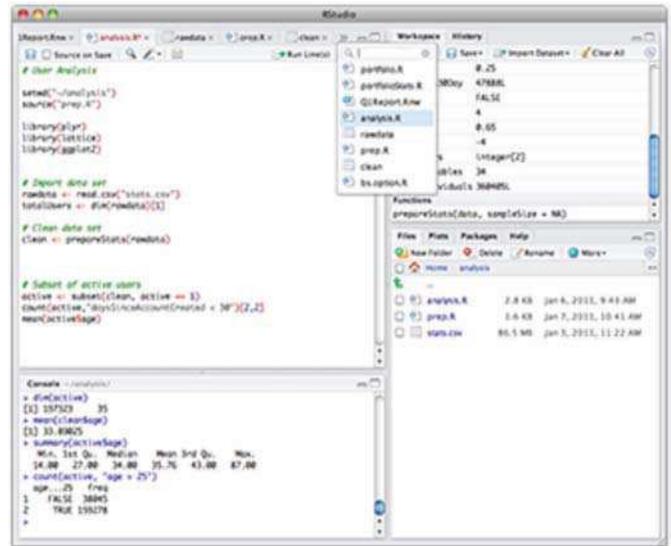
RStudio v0.93 is now available. This release includes source editor enhancements, options for customizing pane layout and appearance, an interactive plotting package, improved handling of Unicode characters, and many more small enhancements and bug fixes. All of the details on the new release can be found in our [release notes](#)

Announcing RStudio (2/28/2011)

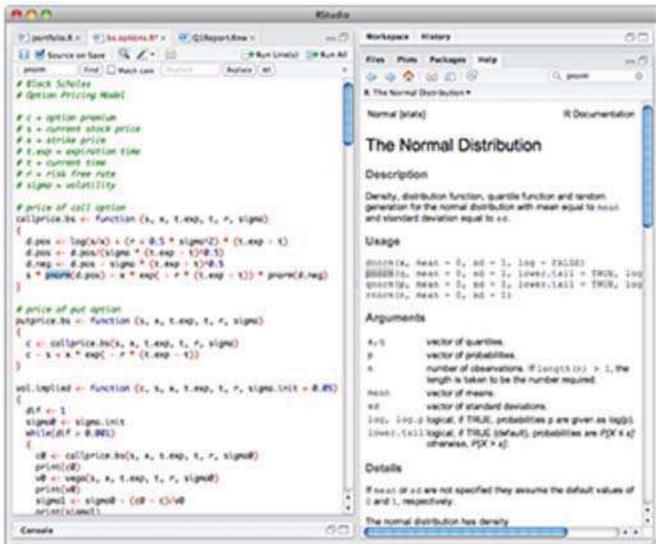
We are pleased to announce availability of the beta version of RStudio! RStudio is a new open-source IDE for R that runs either on your desktop or on a server (where it is accessed using a browser). We invite everyone to check out the [screenshots](#) for more details; download the product to try it out, and follow its ongoing development on [github](#).



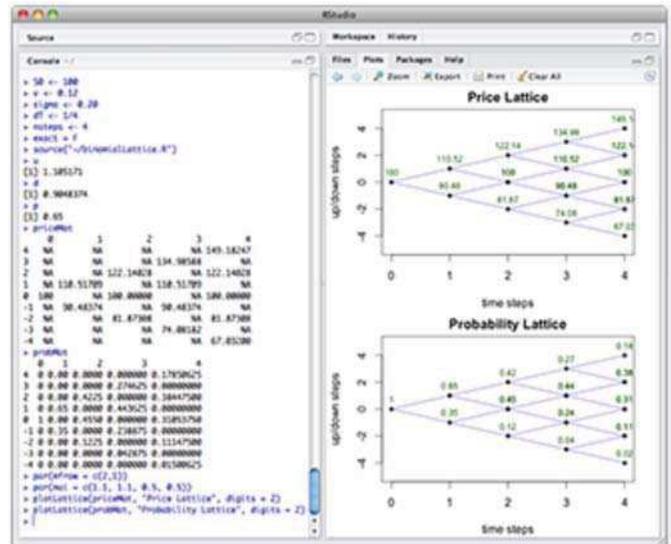
Source, Console, Workspace, and Plots



Tabbed Source Editor

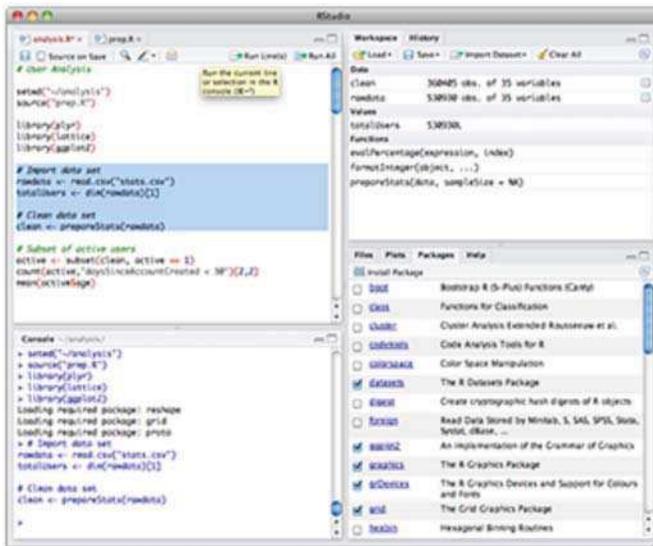


Customized for Coding



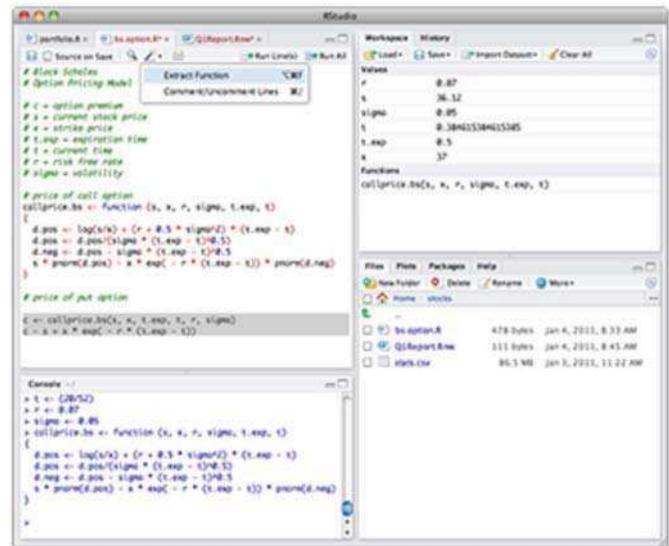
Customized for Plotting

Searchable History



Execute From Source

Code Completion



Code Transformations

RStudio supports authoring TeX and Sweave documents:

```

Price and Probability Matrix\
p = S/rnorm(100)/rnorm(100)/rnorm(100)/rnorm(100)/rnorm(100)
v = S/rnorm(100)/rnorm(100)/rnorm(100)
d = S/rnorm(100)/rnorm(100)/rnorm(100)

if (n(E) = 100, v = 12 %%, sigma = 20 %%, S0 = 100)
  S/rnorm(100)

# Code Chunk 2
rCode Chunk 2, fig=FALSE, include=FALSE, echo=FALSE,
results=hide x=
# setup
S0 = 100 # initial price
r = 0.12 # expected annual growth rate
sigma = 0.20 # annual standard deviation
dt = 1/4 # length of time step (in years)
nsteps = 4 # number of time steps
exact = 1 # 1 for exact, 0 for approximate

f[["exact"]] {
  p = 0.5 + 0.5*sqrt(sigma^2*(r^2*dt^2)+1)
  u = exp(sqrt(sigma^2*dt)*(r+sigma*p))
  d = exp(sqrt(sigma^2*dt)*(r-sigma*p))
} else {
  p = 0.5 + 0.5*(sigma^2*dt)
  u = exp(sigma*sqrt(dt))
  d = exp(-sigma*sqrt(dt))
}

# create and plot price matrix
priceMat = matrix(NA, ncol=nsteps+1, nrow=2^nsteps+1,
  dimnames=list(as.character(nsteps+1:nsteps),
  as.character(0:nsteps)))
priceMat["0",0] = S0
for(colnum in seq(colnum, colnum, by=2))

```

Syntax Highlighting Editor

```

# Function to plot lattice
plotLattice = function(mat, title="Price Lattice", digits=2) {
  nsteps = ncol(mat) - 1
  plot(NA, type="n", xlim=c(0, nsteps),
  title=title)
  text(x=colnum, y=rownum, label=round(mat[colnum, rownum],
  digits=digits), col="k",
  pos=c("left", "top"), cex=1.5, col="darkgreen")
  points(x=colnum, y=rownum, pch="k")
}
title(title)
}
# compilePDF("~/binomialLattice.Rnw")
Writing to file binomialLattice.tex
Processing code chunks ...
1 : term hide (label=Code Chunk 1)
2 : term hide (label=Code Chunk 2)
3 : echo term hide (label=Code Chunk 3)

```

One-Click PDF

Download RStudio v0.93

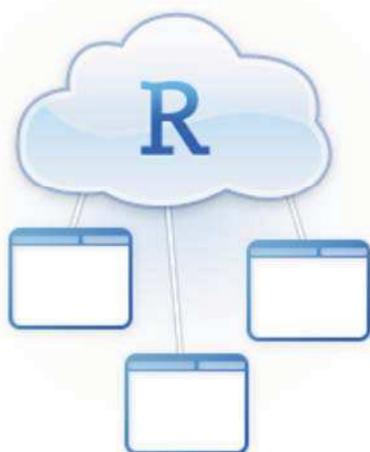


If you run R on your desktop:



[Download RStudio Desktop](#)

OR



If you run R on a Linux server and want to enable users to remotely access RStudio using a web browser:



[Download RStudio Server](#)

Download RStudio Desktop

RStudio v0.93 — Release Notes

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

Recommended For Your System	Size	Date	MD5
 RStudio Desktop 0.93.92 - Windows XP/Vista/7	15.4 MB	2011-05-21	45cdab67f601462e749fb4126a76fa10
All Platforms			
RStudio Desktop 0.93.92 - Windows XP/Vista/7	15.4 MB	2011-05-21	45cdab67f601462e749fb4126a76fa10
RStudio Desktop 0.93.92 - Mac OS X 10.5+	39.6 MB	2011-05-21	dd874ab38c3ae774a84da0cb265a15c7
RStudio Desktop 0.93.92 - Ubuntu 10.04+ (32-bit)	22.7 MB	2011-05-21	dce58c79ec849948fc0d07b1d65e2496
RStudio Desktop 0.93.92 - Ubuntu 10.04+ (64-bit)	23.1 MB	2011-05-21	585aec371628605b2d570e2d0b969dde
RStudio Desktop 0.93.92 - Fedora 13+ (32-bit)	22.7 MB	2011-05-21	d99d3dee98ff585903be32dc4047cccc
RStudio Desktop 0.93.92 - Fedora 13+ (64-bit)	23 MB	2011-05-21	8e9ef279bcae6c34df629978b0a05e07

Source Code

A tarball containing source code for RStudio v0.93.92 can be downloaded from [here](#).

© 2011 RStudio, Inc.

Download RStudio Server

RStudio v0.93 — Release Notes

Ubuntu (10.04 or higher)

Prerequisites

RStudio requires a previous installation of R version 2.11.1 (or higher). To install the latest version of R you should first add the CRAN repository to your system as described here: [Ubuntu Packages for R](#).

You can then install R using the following command:

```
$ sudo apt-get install r-base
```

Note that if you do not add the CRAN Ubuntu repository as described above this command will install the version of R corresponding to your current system version (R 2.10 for Ubuntu 10.04, R 2.11.1 for Ubuntu 10.10).

Download and Install

To download and install RStudio Server open a terminal window and execute the commands corresponding to the 32 or 64-bit version as appropriate.

32-bit Size: 5.8 MB MD5: 1bb3f29aa34c2a01ca677230913e139d

```
$ wget https://s3.amazonaws.com/rstudio-server/rstudio-server-0.93.92-i386.deb
$ sudo dpkg -i rstudio-server-0.93.92-i386.deb
```

64-bit Size: 5.9 MB MD5: bd21457b8a1c8362ea48d7d630a4a73f

```
$ wget https://s3.amazonaws.com/rstudio-server/rstudio-server-0.93.92-amd64.deb
$ sudo dpkg -i rstudio-server-0.93.92-amd64.deb
```

After installing please see the [Getting Started](#) document for information configuring and managing the server.

RedHat/CentOS (5.4 or higher)

Prerequisites

RStudio Server has several dependencies on packages (including R itself) found in the [Extra Packages for Enterprise Linux](#) (EPEL) repository. If you don't already have this repository available you can add it to your system using the following command:

```
$ sudo rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm
```

You should ensure that you have installed the version of R available from EPEL. You can do this using the following command:

```
$ sudo yum install R
```

Download and Install

To download and install RStudio Server open a terminal window and execute the commands corresponding to the 32 or 64-bit version as appropriate.

32-bit Size: 6 MB MD5: 08c992f841cbc9f6b3e14b11790aac08

```
$ wget https://s3.amazonaws.com/rstudio-server/rstudio-server-0.93.92-i686.rpm
$ sudo rpm -Uvh rstudio-server-0.93.92-i686.rpm
```

64-bit Size: 6.2 MB MD5: ec846ee17568d42a2f38c2edfd4d36ce

```
$ wget https://s3.amazonaws.com/rstudio-server/rstudio-server-0.93.92-x86_64.rpm
$ sudo rpm -Uvh rstudio-server-0.93.92-x86_64.rpm
```

After installing please see the [Getting Started](#) document for information on configuring and managing the server.

Other Platforms

To install RStudio on other platforms you will need to build and install from source. Note that while it is likely that RStudio will correctly compile and install on your target platform the only platforms currently tested and certified are Ubuntu and RedHat/CentOS.

A tarball containing source code for RStudio v0.93.92 can be downloaded from [here](#).

The INSTALL file within the source distribution contains additional instructions on how to build and install from source. After successfully performing a `make install` you should also be sure to install the additional scripts and configuration files found within the `src/server/extras` directory (the INSTALL file describes these extras in more detail).

© 2011 RStudio, Inc.

RStudio Documentation

Using RStudio

-  [Working in the Console](#)
-  [Editing and Executing Code](#)
-  [Using Command History](#)
-  [Working Directories and Workspaces](#)
-  [Customizing RStudio](#)
-  [Keyboard Shortcuts](#)

RStudio Server

-  [Getting Started](#)
-  [Configuring the Server](#)
-  [Managing the Server](#)
-  [Running with a Proxy](#)

Advanced Topics

-  [Interactive Plotting with Manipulate](#)
-  [Using Different Versions of R](#)
-  [Character Encoding](#)
-  [Optimizing your Browser for RStudio](#)
-  [Uploading and Downloading Files](#)

More

-  [About RStudio](#)
-  [Release Notes](#)
-  [Frequently Asked Questions](#)
-  [Getting Help with R](#)

Working in the Console

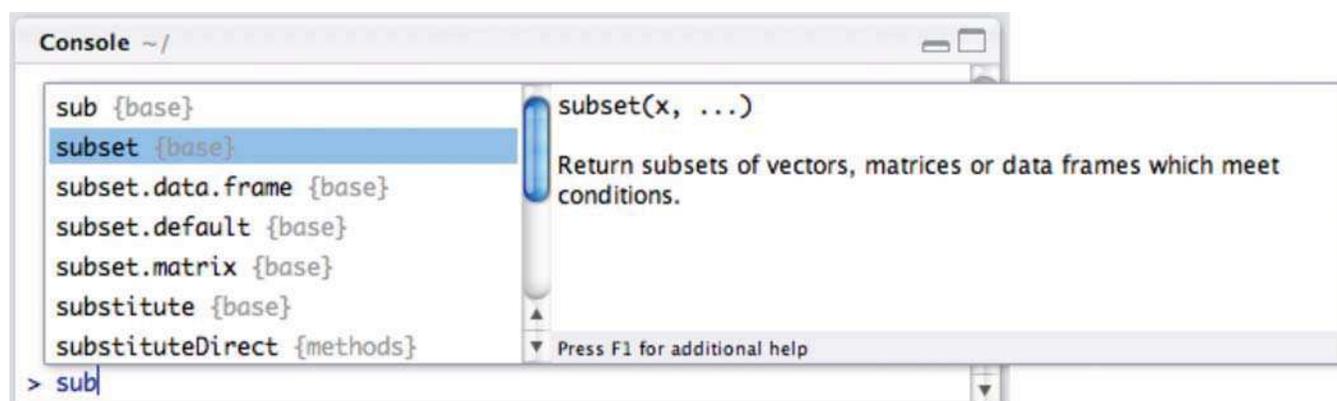
Overview

The RStudio console includes a variety of features intended to make working with R more productive and straightforward. This article reviews these features. Learning to use these features along with the related features available in the **Source** and **History** panes can have a substantial payoff in your overall productivity with R.

Code Completion

RStudio supports the automatic completion of code using the **Tab** key. For example, if you have an object named `pollResults` in your workspace you can type `poll` and then **Tab** and RStudio will automatically complete the full name of the object.

The code completion feature also provides inline help for functions whenever possible. For example, if you type `sub` then pressed **Tab** you would see:



Code completion also works for function arguments, so if you typed `subset(` and then pressed **Tab** you'd see the following:

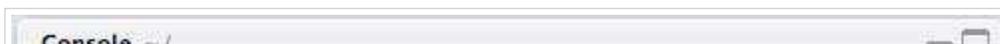


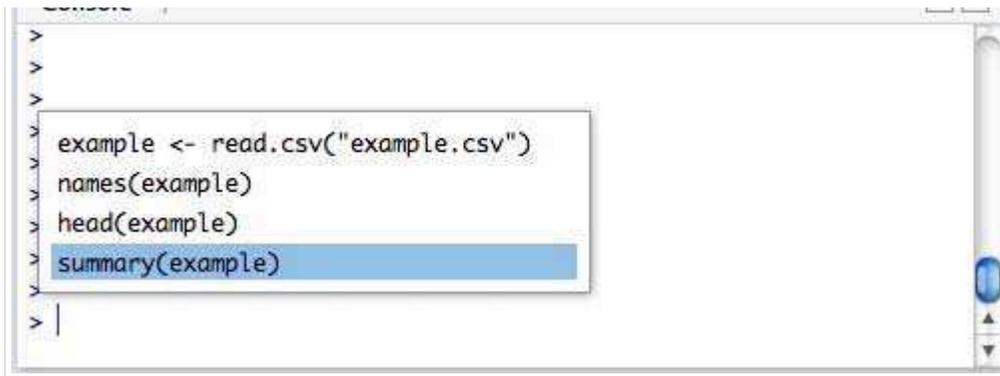
Retrieving Previous Commands

As you work with R you'll often want to re-execute a command which you previously entered. As with the standard R console, the RStudio console supports the ability to recall previous commands using the arrow keys

- **Up** — Recall previous command(s)
- **Down** — Reverse of Up

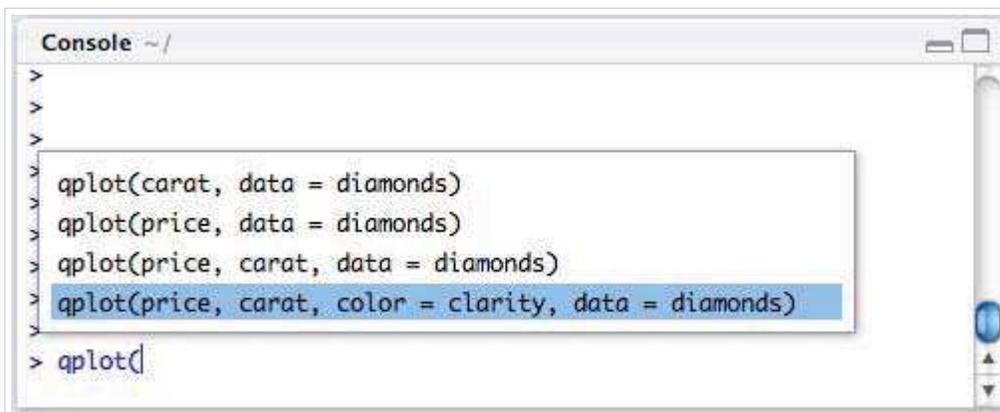
If you wish to review a list of your recent commands and then select a command from this list you can use **Ctrl+Up** to review the list (note that on the Mac you can also use **Command+Up**):





A screenshot of the RStudio Console window. The console shows a list of commands: `example <- read.csv("example.csv")`, `names(example)`, `head(example)`, and `summary(example)`. The `summary(example)` command is highlighted in blue, indicating it is the current selection.

You can also use this same keyboard shortcut to quickly search for commands that match a given prefix. For example, to search for previous instances of the `plot` function simply type `plot` and then `Ctrl+Up`:



A screenshot of the RStudio Console window. The console shows a list of `qplot` commands: `qplot(carat, data = diamonds)`, `qplot(price, data = diamonds)`, `qplot(price, carat, data = diamonds)`, and `qplot(price, carat, color = clarity, data = diamonds)`. The last command is highlighted in blue, indicating it is the current selection.

Console Title Bar

This screenshot illustrates a few additional capabilities provided by the Console title bar:

- Display of the current working directory.
- The ability to interrupt R during a long computation.
- Minimizing and maximizing the Console in relation to the Source pane (using the buttons at the top-right or by double-clicking the title bar).



A screenshot of the RStudio Console title bar. The title bar text is `Console ~/reports/analysis/`. To the right of the text are three icons: a red circle (stop/interrupt), a minus sign (restore down), and a square (restore up/maximize).

Keyboard Shortcuts

Beyond the history and code-completion oriented keyboard shortcuts described above, there are a wide variety of other shortcuts available. Some of the more useful shortcuts include:

- `Ctrl+1` — Move focus to the Source Editor
- `Ctrl+2` — Move focus to the Console
- `Ctrl+L` — Clear the Console
- `Esc` — Interrupt R

You can find a list of all shortcuts in the [Keyboard Shortcuts](#) article.

Related Topics

- [Editing and Executing Code](#)
- [Using Command History](#)

Editing and Executing Code

Overview

RStudio's source editor includes a variety of productivity enhancing features including syntax highlighting, code completion, multiple-file editing, and find/replace.

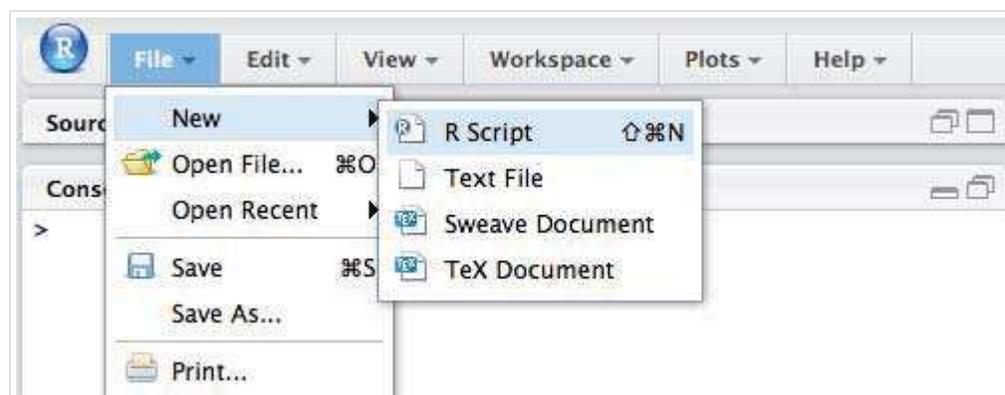
RStudio also enables you to flexibly execute R code directly from the source editor. For many R developers this represents their preferred way of working with R. By executing commands from within the source editor rather than the console it is much easier to reproduce sequences of commands as well as package them for re-use as a function. These features are described in the [Executing Code](#) section below.

Managing Files

RStudio supports syntax highlighting and other specialized code-editing features for the following types of files

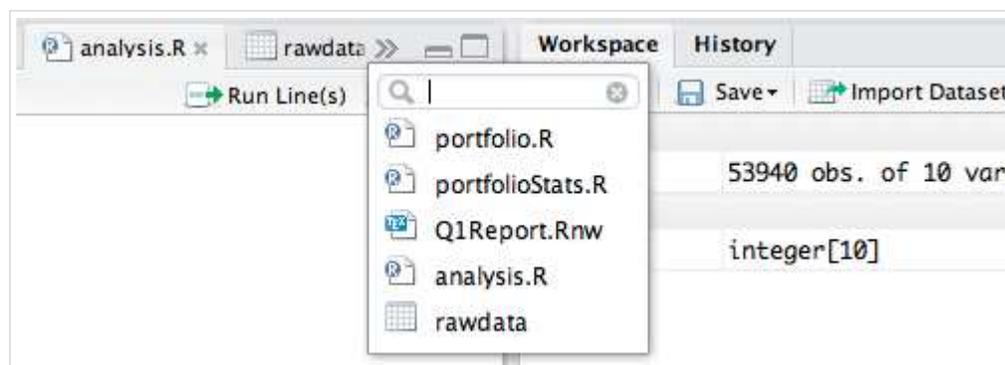
- R scripts
- Sweave documents
- TeX documents

To create a new file you use the File -> New menu:



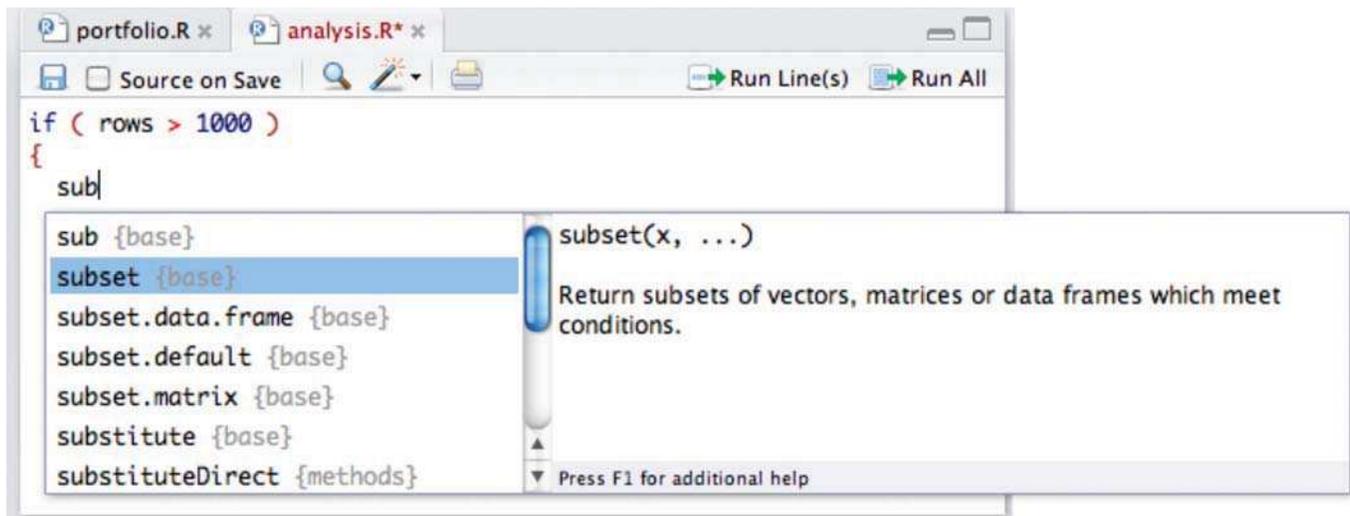
To open an existing file you use either the File -> Open menu or the Open Recent menu to select from recently opened files.

If you open several files within RStudio they are all available as tabs to facilitate quick switching between open documents. If you have a large number of open documents you can also navigate between them using the >> icon on the tab bar or the View -> Switch to Tab) menu item:



Code Completion

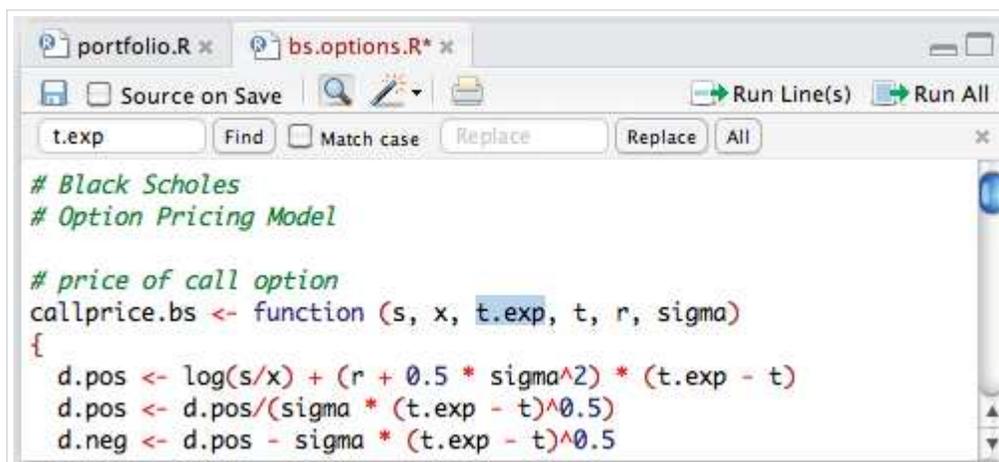
RStudio supports the automatic completion of code using the Tab key. For example, if you have an object named pollResults in your workspace you can type poll and then Tab and RStudio will automatically complete the full name of the object.



Code completion also works in the console, and more details on using it can be found the console [Code Completion](#) documentation.

Find and Replace

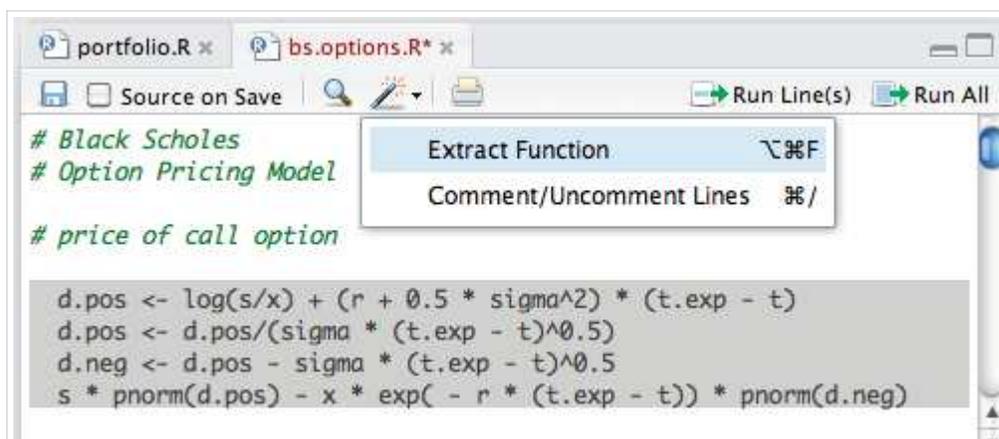
RStudio supports finding and replacing text within source documents:



Find and replace can be opened using the Ctrl+F shortcut key, or from the Edit -> Find and Replace menu item.

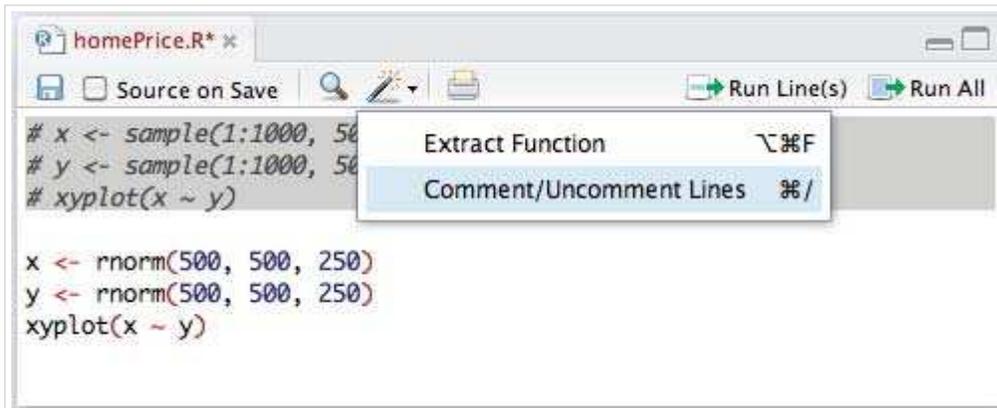
Extract Function

RStudio can analyze a selection of code from within the source editor and automatically convert it into a re-usable function. Any "free" variables within the selection (objects that are referenced but not created within the selection) are converted into function arguments:



Comment/Uncomment

You can comment and uncomment entire selections of code using the Edit -> Comment/Uncomment Lines menu item (you can also do this using the Ctrl+/ keyboard shortcut):

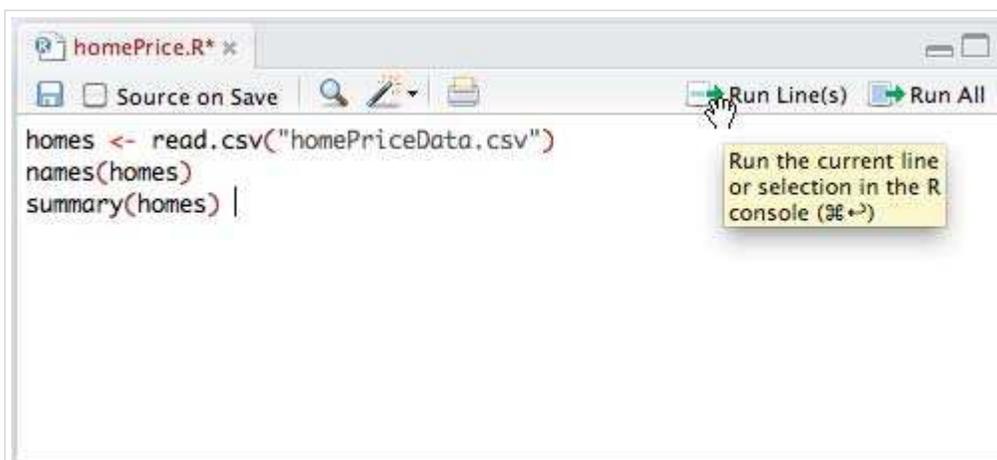


Executing Code

RStudio supports the direct execution of code from within the source editor (the executed commands are inserted into the console where their output also appears).

Executing a Single Line

To execute the line of source code where the cursor currently resides you press the Ctrl+Enter key (or use the Run Line(s) toolbar button):



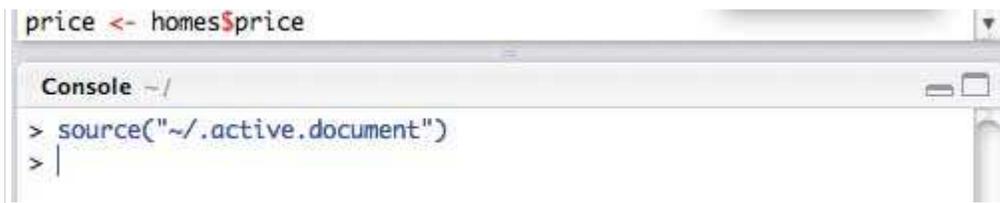
After executing the line of code, RStudio automatically advances the cursor to the next line. This enables you to single-step through a sequence of lines.

Executing Multiple Lines

There are two ways to execute multiple lines from within the editor:

- Select the lines and press the Ctrl+Enter key (or use the Run Line(s) toolbar button); or
- To run the entire document press the Ctrl+Shift+Enter key (or use the Run All toolbar button).





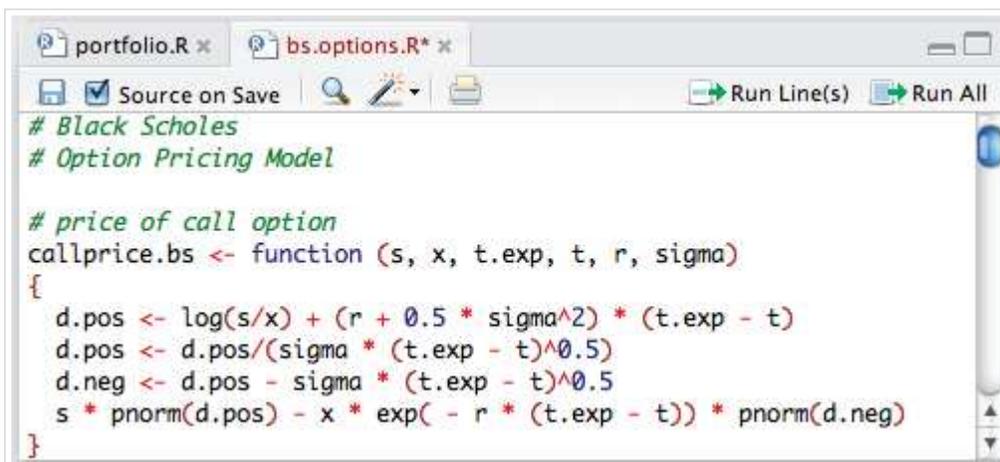
```
price <- homes$price

Console ~/
> source("~/active.document")
> |
```

The difference between running lines from a selection and invoking Run All is that when running a selection all lines are inserted directly into the console whereas for Run All the file is saved to a temporary location and then sourced into the console from there (thereby creating less clutter in the console).

Source on Save

When editing re-usable functions (as opposed to freestanding lines of R) you may wish to set the Source on Save option for the document (available on the toolbar next to the Save icon). Enabling this option will cause the file to automatically be sourced into the global environment every time it is saved:



```
# Black Scholes
# Option Pricing Model

# price of call option
callprice.bs <- function (s, x, t.exp, t, r, sigma)
{
  d.pos <- log(s/x) + (r + 0.5 * sigma^2) * (t.exp - t)
  d.pos <- d.pos / (sigma * (t.exp - t)^0.5)
  d.neg <- d.pos - sigma * (t.exp - t)^0.5
  s * pnorm(d.pos) - x * exp(- r * (t.exp - t)) * pnorm(d.neg)
}
```

Setting Source on Save ensures that the copy of a function within the global environment is always in sync with its source, and also provides a good way to arrange for frequent syntax checking as you develop a function.

Keyboard Shortcuts

Beyond the keyboard shortcuts described above, there are a wide variety of other shortcuts available. Some of the more useful ones include:

- Ctrl+Shift+N — New document
- Ctrl+O — Open document
- Ctrl+S — Save active document
- Ctrl+1 — Move focus to the Source Editor
- Ctrl+2 — Move focus to the Console

You can find a list of all shortcuts in the [Keyboard Shortcuts](#) article.

Related Topics

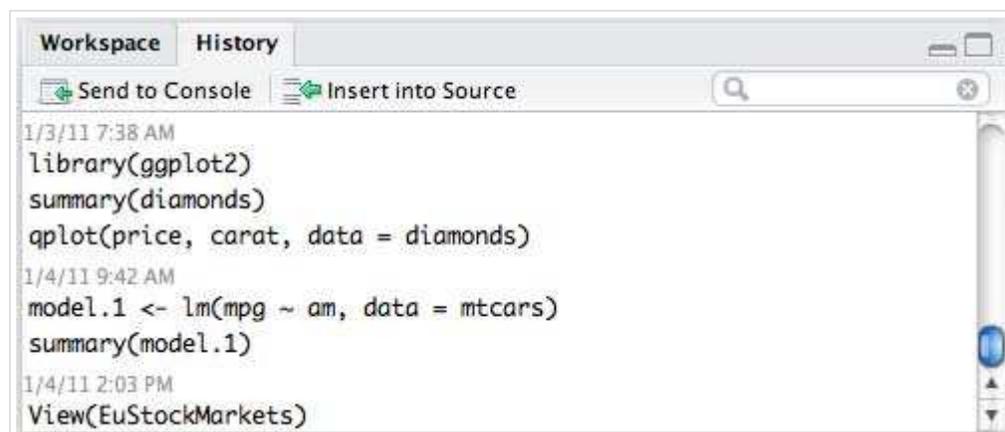
- [Working in the Console](#)
- [Using Command History](#)

Using Command History

RStudio maintains a database of all commands which you have ever entered into the Console. You can browse and search this database using the History pane.

Browsing History

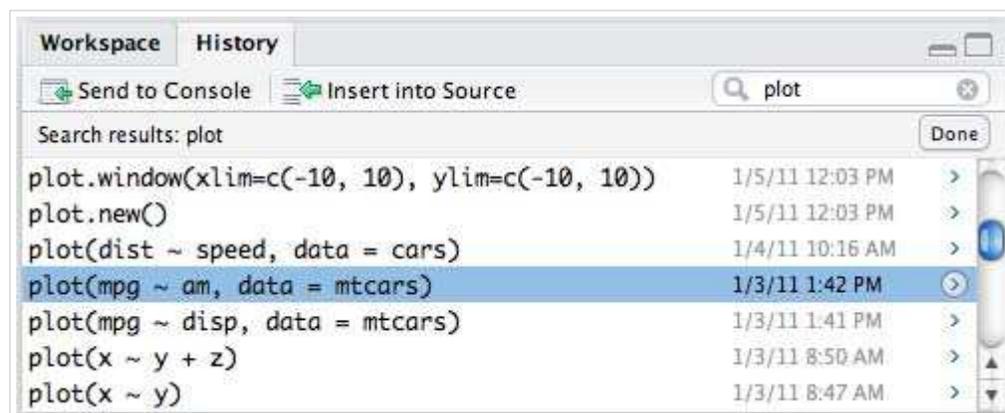
Commands you have previously entered in the RStudio console can be browsed from the History tab. The commands are displayed in order (most recent at the bottom) and grouped by block of time:



Searching History

Executing a Search

You can use the search box at the top right of the history tab to search for all instances of a previous command (e.g. plot). The search can be further refined by adding additional words separated by spaces (e.g. the name of particular dataset):



Showing Command Context

After searching for a command within your history you may wish to view the other commands that were executed in proximity to it. By clicking the arrow in the right margin of the search results you can view the command within its context:



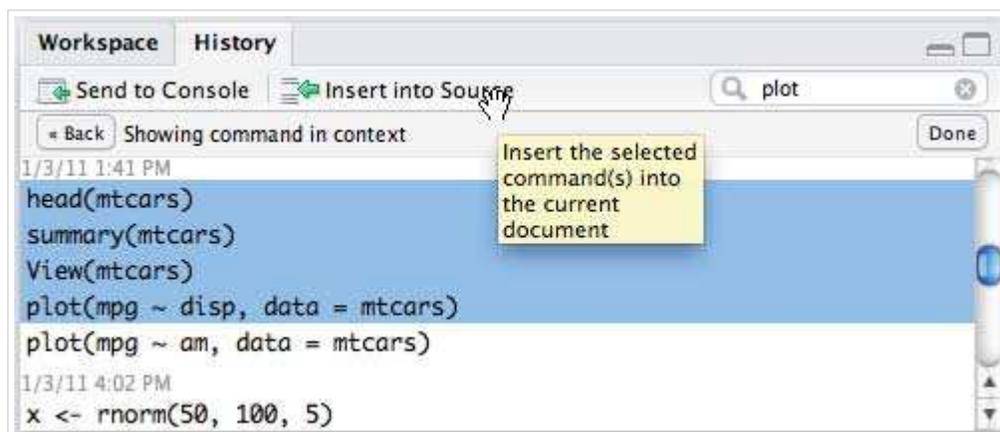
```
plot(mpg ~ disp, data = mtcars)
plot(mpg ~ am, data = mtcars)
1/3/11 4:02 PM
x <- rnorm(50, 100, 5)
```

Using Commands

Commands selected within the History pane can be used in two fashions (corresponding to the two buttons on the left side of the History toolbar):

- **Send to Console**— Sends the selected command(s) to the Console. Note that the commands are inserted into the Console however they are not executed until you press Enter.
- **Insert into Source**— Inserts the selected command(s) into the currently active Source document. If there isn't currently a Source document available then a new untitled one will be created.

Within the history list you can select a single command or multiple commands:



Related Topics

- Working in the Console
- Editing and Executing Code

Working Directories and Workspaces

The default behavior of R for the handling of .RData files and workspaces encourages and facilitates a model of breaking work contexts into distinct working directories. This article describes the various features of RStudio which support this workflow.

Default Working Directory

As with the standard R GUI, RStudio employs the notion of a global default working directory. Normally this is the user home directory (typically referenced using `~` in R). When RStudio starts up it does the following:

- Executes the `.Rprofile` (if any) from the default working directory.
- Loads the `.RData` file (if any) from the default working directory into the workspace.
- Performs the other actions described in [R Startup](#).

When RStudio exits and there are changes to the workspace, a prompt asks whether these changes should be saved to the `.RData` file in the current working directory.

This default behavior can be customized in the following ways using the RStudio Options dialog:

- Change the default working directory
- Enable/disable the loading of `.RData` from the default working directory at startup
- Specify whether `.RData` is always saved, never saved, or prompted for save at exit.

Changing the Working Directory

The current working directory is displayed by RStudio within the title region of the Console. There are a number of ways to change the current working directory:

- Use the `setwd` R function
- Use the Tools | Change Working Dir... menu. This will also change directory location of the Files pane.
- From within the Files pane, use the More | Set As Working Directory menu. (Navigation within the Files pane alone will not change the working directory.)

Be careful to consider the side effects of changing your working directory:

- Relative file references in your code (for datasets, source files, etc) will become invalid when you change working directories.
- The location where `.RData` is saved at exit will be changed to the new directory.

Because these side effects can cause confusion and errors, it's usually best to start within the working directory associated with your project and remain there for the duration of your session. The section below describes how to set RStudio's initial working directory.

Starting in Other Working Directories

If all of the files related to a project are contained within a single directory then you'll likely want to start RStudio within that directory. There are a number of ways (which vary by platform) to do this.

File Associations

On all platforms RStudio registers itself as a handler for `.RData`, `.R`, and other R related file types. This means that the system file browser's context-menu will show RStudio as an Open With choice for these files.

You can also optionally create a default association between RStudio and the `.RData` and/or `.R` file types.

When launched through a file association, RStudio automatically sets the working directory to the directory of the opened file. Note that RStudio can also open files via associations when it is already running—in this case

RStudio simply opens the file and does not change the working directory.

Shortcuts (Windows)

On Windows, you can create a shortcut to RStudio and customize the "Start in" field. When launched through this shortcut RStudio will startup within the specified working directory.

Drag and Drop (Mac)

On Mac, dragging and dropping a folder from the Finder on the RStudio Dock icon will cause RStudio to startup with the dropped folder as the current working directory.

Run from Terminal (Mac and Linux)

On Mac and Linux systems you can run RStudio from a terminal and specify which working directory to startup within. Additionally, on Linux systems if you run RStudio from a terminal and specify no command line argument then RStudio will startup using the current working directory of the terminal.

For example, on the Mac you could use the following commands to open RStudio (respectively) in the '~/projects/foo' directory or the current working directory:

```
$ open -a RStudio ~/projects/foo
$ open -a RStudio .
```

On Linux you would use the following commands (note that no '.' is necessary in the second invocation):

```
$ rstudio ~/projects/foo
$ rstudio
```

Handling of .Rprofile

When starting RStudio in an alternate working directory the .Rprofile file located within that directory is sourced. If (and only if) there is not an .Rprofile file in the alternate directory then the global default profile (e.g. ~/.Rprofile) is sourced instead.

Loading and Saving Workspaces

If you want to save or load a workspace during an RStudio session you can use the following commands to save to or load from the .RData file in the current working directory:

```
> save.image()
> load(".RData")
```

Note that the load function appends (and overwrites) objects within the current workspace rather than replacing it entirely. Prior to loading you may therefore wish to clear all objects currently within the workspace. You can do so using the following command:

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

Note that since loading is handled at startup and saving is handled at exit, in many cases you won't require these commands. If however you change working directories within a session you may need them in order to sync your workspace with the directory you have changed to.

The RStudio Workspace menu also includes items that execute the above described commands, as well as enables you to load or save specific .RData files.

Handling of .Rhistory

The `.Rhistory` file determines which commands are available by pressing the up arrow key within the console. RStudio handles the `.Rhistory` file differently than the standard R GUI. This is because in addition to the `.Rhistory` file RStudio also includes a searchable history database (accessible via the History tab). For the sake of simplicity RStudio attempts to keep these two history contexts in sync.

The conventional handling of `.Rhistory` files is as follows:

- Load and save `.Rhistory` within the current working directory
- Only save the `.Rhistory` file when the user chooses to save the `.RData` file

Whereas the RStudio handling of `.Rhistory` files is:

- Load and save a single global `.Rhistory` file (located in the default working directory)
- Always save the `.Rhistory` file (even if the `.RData` file is not saved)

As a result, the contents of the History pane always match the up arrow history within the console.

Customizing RStudio

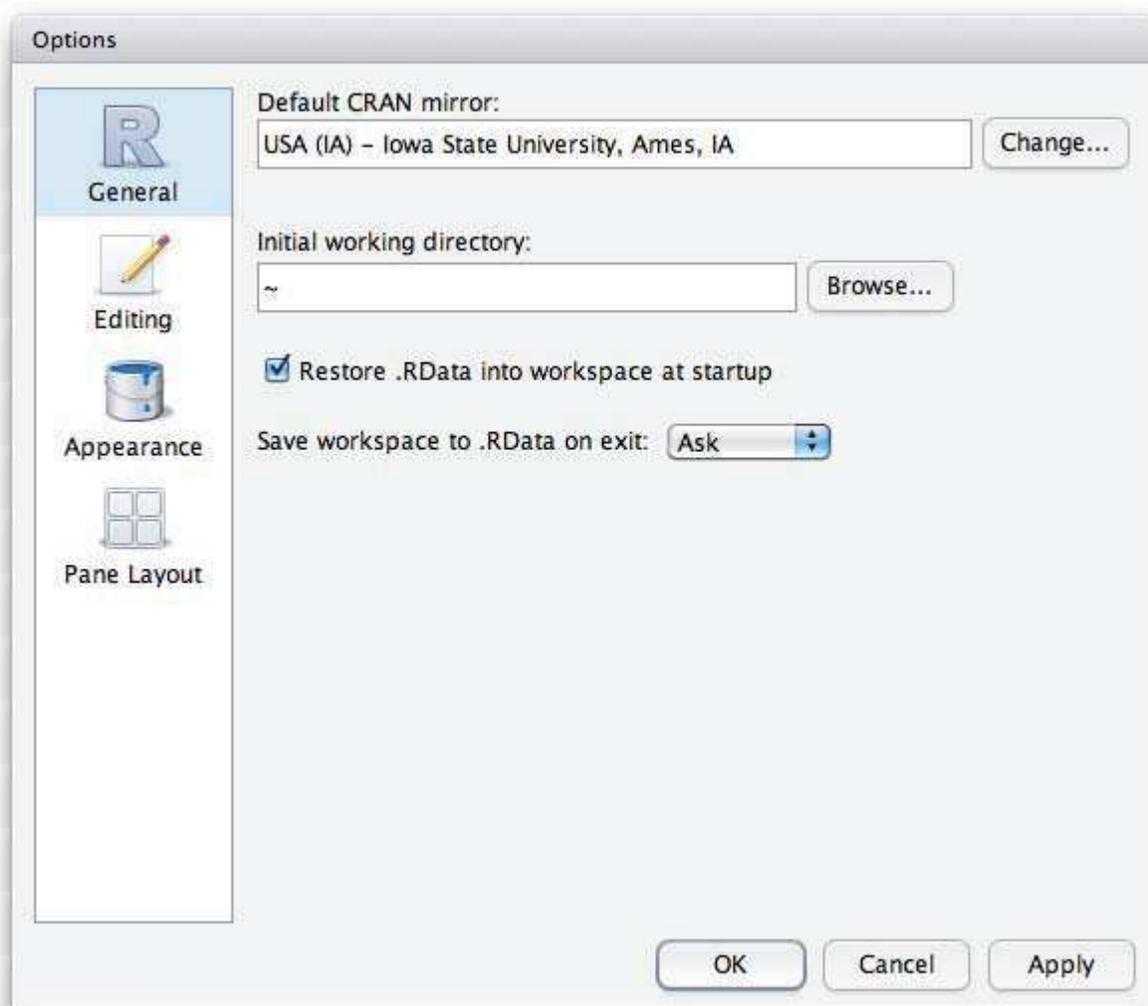
Overview

RStudio options are accessible from the Options dialog Tools : Options menu and include the following categories:

- **General R Options** — Default CRAN mirror, initial working directory, workspace save and restore behavior
- **Source Code Editing** — Enable/disable line numbers, line highlighting, soft-wrapping for R files, and right margin display; configure tab spacing; set default text encoding.
- **Appearance and Themes** — Specify font size and visual theme for the console and source editor.
- **Pane Layout** — Locations of console, source editor, and tab panes; set which tabs are included in each pane.

Details on the various settings are provided in the sections below.

General R Options

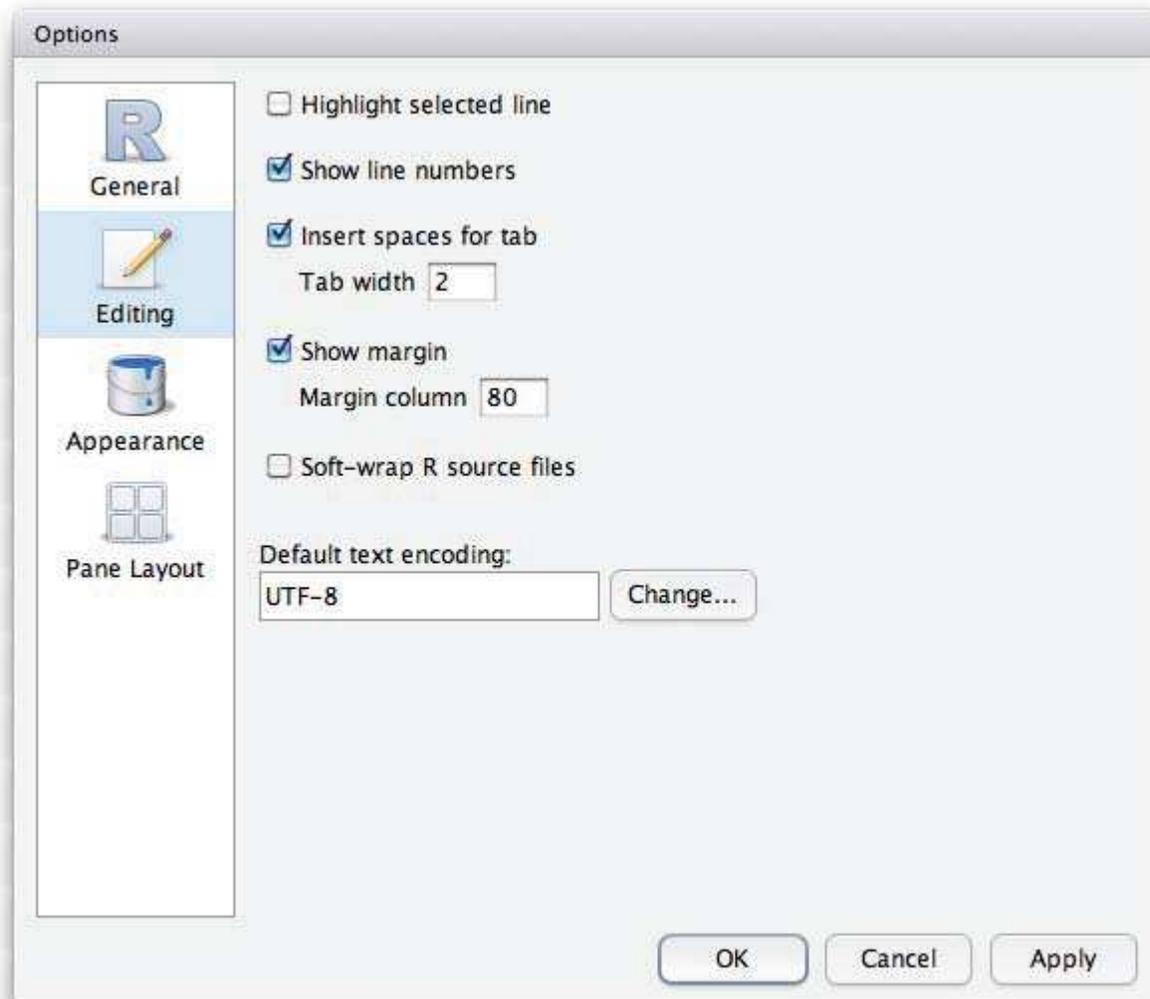


- **Default CRAN mirror** — Set the CRAN mirror used for installing packages (can be overridden using the repos argument to install.packages).
- **Initial working directory** — Startup directory for RStudio. The initial .RData and .Rprofile files (if any) will be read from this directory. The current working directory and Files pane will also be set to this directory.

Note that this setting can be overridden when launching RStudio using a file association or a terminal with a command line parameter indicating the initial working directory.

- Restore .RData into workspace at startup — Load the .RData file (if any) found in the initial working directory into the R workspace (global environment) at startup. If you have a very large .RData file then unchecking this option will improve startup time considerably.
- Save workspace to .RData on exit — Ask whether to save .RData on exit, always save it, or never save it. Note that if the workspace is not dirty (no changes made) at the end of a session then no prompt to save occurs even if Ask is specified.

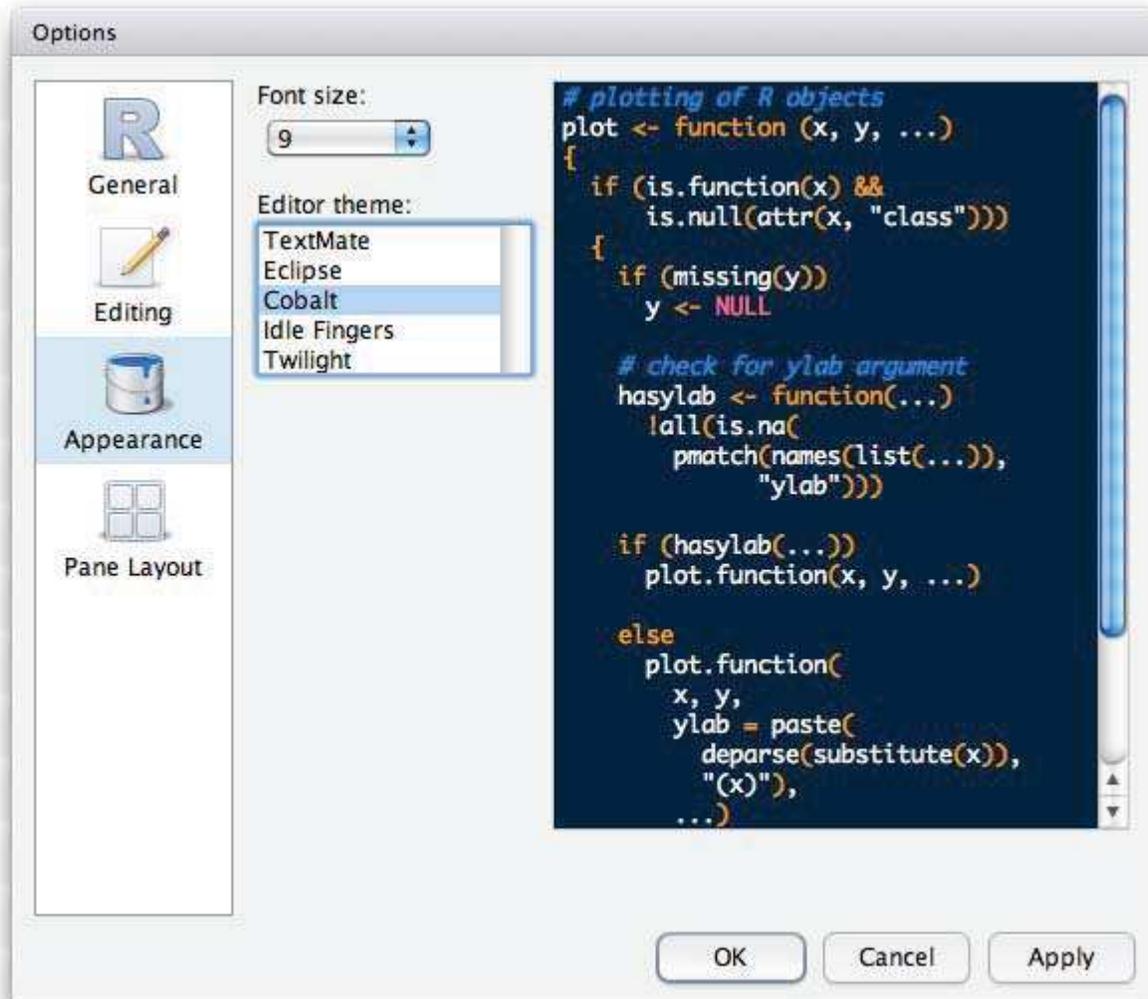
Source Code Editing



- Highlight selected line — Add a background highlight effect to the currently selected line of code.
- Show line numbers — Show or hide line numbers within the left margin.
- Insert spaces for tab — Determine whether the tab key inserts multiple spaces rather than a tab character (soft tabs). Configure the number of spaces per soft-tab.
- Show margin — Display a margin guide on the right-hand side of the source editor at the specified column
- Soft-wrap R source files — Wrap lines of R source code which exceed the width of the editor onto the next line. Note that this does not insert a line-break at the point of wrapping, it simply displays the code on multiple lines in the editor.
- Default text encoding — Specify the default text encoding for source files. Note that source files which don't match the default encoding can still be opened correctly using the File : Reopen with Encoding menu

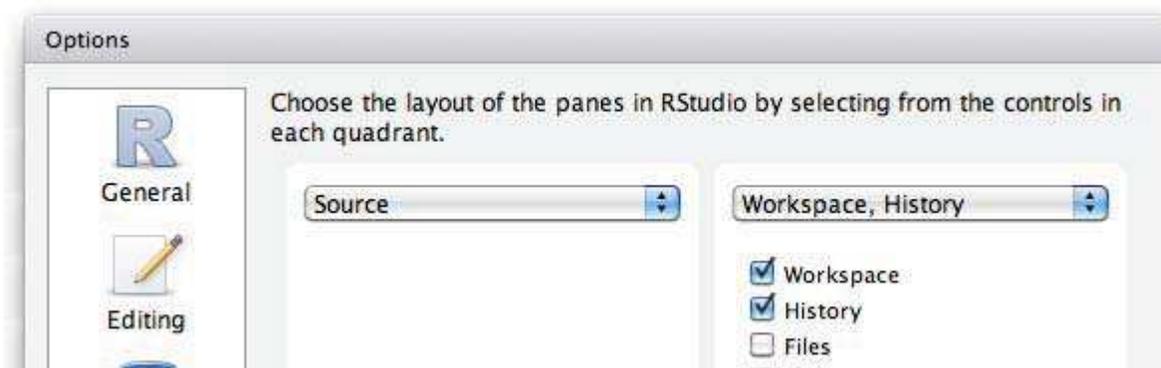
command.

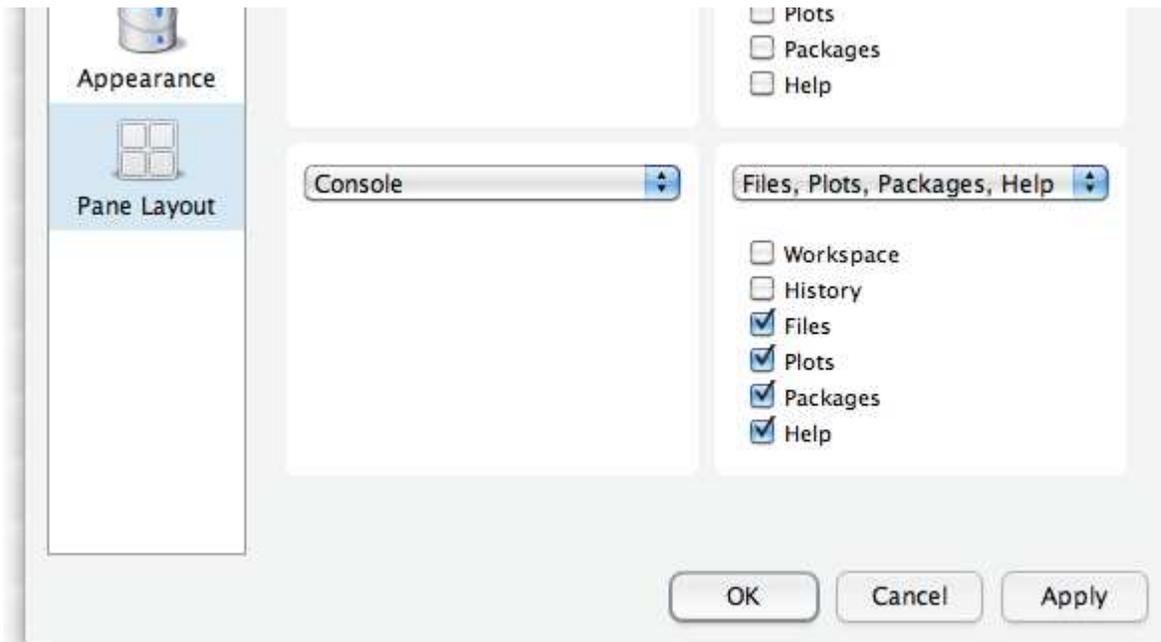
Appearance and Themes



- **Font size** — Set the font size (in points) for panes which display code (Console, Source, History, and Workspace).
- **Editor theme** — Specify the visual theme for the Console and Source panes. You can preview the theme using the inline preview or by pressing the Apply button.

Pane Layout





- Specify the location and tab sets of panes within RStudio.
- Each of the 4 panes is always displayed (it isn't currently possible to hide a pane).

Keyboard Shortcuts

Console

Description	Windows & Linux	Mac
Move cursor to Console	Ctrl+2	Ctrl+2
Clear console	Ctrl+L	Command+L
Move cursor to beginning of line	Home	Command+Left
Move cursor to end of line	End	Command+Right
Navigate command history	Up/Down	Up/Down
Popup command history	Ctrl+Up	Command+Up
Interrupt currently executing command	Esc	Esc
Yank line up to cursor	Ctrl+U	Command+U
Yank line after cursor	Ctrl+K	Command+K
Insert currently yanked text	Ctrl+Y	Command+Y
Insert assignment operator	Alt+-	Option+-

Source

Description	Windows & Linux	Mac
Move cursor to Source Editor	Ctrl+I	Ctrl+I
New document (except on Chrome/Windows)	Ctrl+Shift+N	Command+Shift+N
Open document	Ctrl+O	Command+O
Save active document	Ctrl+S	Command+S
Close active document	Ctrl+Shift+L	Command+Shift+L
Compile PDF (TeX and Sweave)	Ctrl+Shift+P	Command+Shift+P
Run current line/selection	Ctrl+Enter	Command+Enter
Run current document	Ctrl+Shift+Enter	Command+Shift+Enter
Switch to tab	Ctrl+Alt+Down	Ctrl+Option+Down
Previous tab	Ctrl+Alt+Left	Ctrl+Option+Left
Next tab	Ctrl+Alt+Right	Ctrl+Option+Right
First tab	Ctrl+Shift+Alt+Left	Ctrl+Shift+Option+Left
Last tab	Ctrl+Shift+Alt+Right	Ctrl+Shift+Option+Right
Extract function from selection	Ctrl+Shift+F	Command+Shift+F
Comment/uncomment current line/selection	Ctrl+/ /	Command+/ /
Insert assignment operator	Alt+-	Option+-
Transpose Letters		Ctrl+T
Jump to Word	Ctrl+Left/Right	Option+Left/Right

Jump to Start/End	Ctrl+Home/End or Ctrl+Up/Down	Command+Home/End or Command+Up/Down
Delete Line	Ctrl+D	Command+D
Move Lines Up/Down	Alt+Up/Down	Option+Up/Down
Copy Lines Up/Down	Ctrl+Alt+Up/Down	Command+Option+Up/Down
Select	Shift+[Arrow]	Shift+[Arrow]
Select Word	Ctrl+Shift+Left/Right	Option+Shift+Left/Right
Select to Line Start	Shift+Home	Command+Shift+Left or Shift+Home
Select to Line End	Shift+End	Command+Shift+Right or Shift+End
Select Page Up/Down	Shift+PageUp/PageDown	Shift+PageUp/Down
Select to Start/End	Ctrl+Shift+Home/End or Shift+Alt+Up/Down	Command+Shift+Up/Down
Delete Word Left	Ctrl+Backspace	Option+Backspace or Ctrl+Option+Backspace
Delete Word Right		Option+Delete
Delete to Line End		Ctrl+K
Delete to Line Start		Option+Backspace
Indent	Tab (at beginning of line)	Tab (at beginning of line)
Outdent	Shift+Tab	Shift+Tab

Editing (Console and Source)

Description	Windows & Linux	Mac
Undo	Ctrl+Z	Command+Z
Redo	Ctrl+Shift+Z	Command+Shift+Z
Cut	Ctrl+X	Command+X
Copy	Ctrl+C	Command+C
Paste	Ctrl+V	Command+V
Select All	Ctrl+A	Command+A

Completions (Console and Source)

Description	Windows & Linux	Mac
Attempt completion	Tab or Ctrl+Space	Tab or Command+Space
Navigate candidates	Up/Down	Up/Down
Accept selected candidate	Enter, Tab, or Right	Enter, Tab, or Right
Show help for selected candidate	F1	F1
Dismiss completion popup	Esc	Esc

Views

Description	Windows & Linux	Mac
--------------------	----------------------------	------------

Move cursor to Source Editor	Ctrl+1	Ctrl+1
Move cursor to Console	Ctrl+2	Ctrl+2
Show workspace	Ctrl+3	Ctrl+3
Show data	Ctrl+4	Ctrl+4
Show history	Ctrl+5	Ctrl+5
Show files	Ctrl+6	Ctrl+6
Show plots	Ctrl+7	Ctrl+7
Show packages	Ctrl+8	Ctrl+8
Show help	Ctrl+9	Ctrl+9

Plots

Description	Windows & Linux	Mac
Previous plot	Ctrl+PageUp	Command+PageUp
Next plot	Ctrl+PageDown	Command+PageDown

RStudio Server: Getting Started

Overview

RStudio Server enables you to provide a browser based interface (the RStudio IDE) to a version of R running on a remote Linux server. Deploying R and RStudio on a server has a number of benefits, including:

- The ability to access your R workspace from any computer in any location;
- Easy sharing of code, data, and other files with colleagues;
- Allowing multiple users to share access to the more powerful compute resources (memory, processors, etc.) available on a well equipped server; and
- Centralized installation and configuration of R, R packages, TeX, and other supporting libraries.

RStudio Server works with recent versions of popular Linux distributions including Red Hat and Ubuntu. RStudio Server can also be built and installed from source on other platforms (see notes on this below).

Download and Install

RStudio Server binary packages are available for recent versions of popular Linux distributions including Ubuntu (version 10.04 or higher) and RedHat/CentOS (version 5.4 or higher). For other platforms it is also possible to build and install from source.

Instructions for downloading and installing RStudio Server can be found on the [server downloads](#) page.

Accessing the Server

By default RStudio Server runs on port 8787 and accepts connections from all remote clients. After installation you should therefore be able to navigate a web browser to the following address to access the server:

```
http://<server-ip>:8787
```

RStudio will prompt for a username and password, and will authenticate the user by checking the server's username and password database. Note that user credentials are encrypted using RSA as they travel over the network.

Configuration and Management

RStudio Server has a variety of configuration options (including the ability to change what port the server listens on) as well as a utility for managing the lifetime of the server and remote user sessions. You can find out more about these capabilities in the following articles:

- [Configuring the Server](#)
- [Managing the Server](#)

If you are running RStudio on a public network you may wish deploy RStudio behind another server (e.g. [Nginx](#) or [Apache](#)) which acts as a reverse proxy to it. You can find out more about doing this in the following article:

- [Running with a Proxy](#)

RStudio Server: Configuring the Server

Overview

RStudio is configured by adding entries to two configuration files (note that these files do not exist by default so you will need to create them if you wish to specify custom settings):

```
/etc/rstudio/rserver.conf
/etc/rstudio/rsession.conf
```

After editing configuration files you should perform a check to ensure that the entries you specified are valid. This can be accomplished by executing the following command:

```
$ sudo rstudio-server test-config
```

Note that this command is also automatically executed when starting or restarting the server (those commands will fail if the configuration is not valid).

Network Port and Address

After initial installation RStudio accepts connections on port 8787. If you wish to change to another port you should create an `/etc/rstudio/rserver.conf` file (if one doesn't already exist) and add a `www-port` entry corresponding to the port you want RStudio to listen on. For example:

```
www-port=80
```

By default RStudio binds to address 0.0.0.0 (accepting connections from any remote IP). You can modify this behavior using the `www-address` entry. For example:

```
www-address=127.0.0.1
```

Note that after editing the `/etc/rstudio/rserver.conf` file you should always restart the server to apply your changes (and validate that your configuration entries were valid). You can do this by entering the following command:

```
$ sudo rstudio-server restart
```

Specifying R Version

By default RStudio Server runs against the version of R which is found on the system PATH (using `which R`). You can override which version of R is used via the `rsession-which-r` setting. For example, if you have two version of R installed on the server and want to make sure the one at `/usr/local/bin/R` is used by RStudio then you would use:

```
rsession-which-r=/usr/local/bin/R
```

Note again that the server must be restarted for this setting to take effect.

Setting User Limits

There are a number of settings which place limits on which users can access RStudio and the amount of resources they can consume. This file does not exist by default so if you wish to specify any of the settings below you should create the file.

To limit the users who can login to RStudio to the members of a specific group, you use the `auth-required-user-group` setting. For example:

```
auth-required-user-group=rstudio_users
```

You can also limit the total memory, stack size, and number of simultaneous child processes for users using settings like the following:

```
rsession-memory-limit-mb=4000  
rsession-stack-limit-mb=10  
rsession-process-limit=100
```

Additional Settings

There is a separate `/etc/rstudio/rsession.conf` configuration file that enables you to control various aspects of R sessions (note that as with `rserver.conf` this file does not exist by default). These settings are especially useful if you have a large number of potential users and want to make sure that resources are balanced appropriately.

By default if a user hasn't issued a command for 2 hours RStudio will suspend that user's R session to disk so they are no longer consuming server resources (the next time the user attempts to access the server their session will be restored). You can change the timeout (including disabling it by specifying a value of 0) using the `session-timeout-minutes` setting. For example:

```
session-timeout-minutes=30
```

Note that a user's session will never be suspended while it is running code (only sessions which are idle will be suspended).

You can limit the size of file uploads using the `limit-file-upload-size-mb` setting. For example:

```
limit-file-upload-size-mb=100
```

If you are using the XFS filesystem and you have disk quotas enabled you can have RStudio notify the user when they are close to their soft and/or hard quota by specifying the `limit-xfs-disk-quota` setting. For example:

```
limit-xfs-disk-quota=1
```

Finally, you can set the default CRAN repository for the server using the `r-cran-repos` setting. For example:

```
r-cran-repos=http://cran.case.edu/
```

Note again that the above settings should be specified in the `/etc/rstudio/rsession.conf` file (rather than the aforementioned `rserver.conf` file).

Related Topics

- [Getting Started](#)
- [Managing the Server](#)
- [Running with a Proxy](#)

RStudio Server: Managing the Server

Overview

RStudio server management tasks are performed using the `rstudio-server` utility (installed under `/usr/sbin` in binary distributions). This utility enables the stopping, starting, and restarting of the server, enumeration and suspension of user sessions, taking the server offline, as well as the ability to hot upgrade a running version of the server.

Stopping and Starting

If you installed RStudio using a package manager binary (e.g. a Debian package or RPM) then RStudio is automatically registered as a daemon which starts along with the rest of the system. On Ubuntu this registration is performed using an Upstart script at `/etc/init/rstudio-server.conf`. On other systems an `init.d` script is installed at `/etc/init.d/rstudio-server`.

To manually stop, start, and restart the server you use the following commands:

```
$ sudo rstudio-server stop
$ sudo rstudio-server start
$ sudo rstudio-server restart
```

Managing Active Sessions

There are a number of administrative commands which allow you to see what sessions are active and request suspension of running sessions (note that session data is not lost during a suspend).

To list all currently active sessions:

```
$ sudo rstudio-server active-sessions
```

To suspend an individual session:

```
$ sudo rstudio-server suspend-session <pid>
```

To suspend all running sessions:

```
$ sudo rstudio-server suspend-all
```

The suspend commands also have a "force" variation which will send an interrupt to to the session to request the termination of any running R command:

```
$ sudo rstudio-server force-suspend-session <pid>
$ sudo rstudio-server force-suspend-all
```

The `force-suspend-all` command should be issued immediately prior to any reboot so as to preserve the data and state of active R sessions across the restart.

Taking the Server Offline

If you need to perform system maintenance and want users to receive a friendly message indicating the server is offline you can issue the following command:

```
$ sudo rstudio-server offline
```

When the server is once again available you should issue this command:

```
$ sudo rstudio-server online
```

Upgrading to a New Version

If you install RStudio using a package manager binary (e.g. a Debian package or RPM) and a version of RStudio Server is currently running on the system, then the current running version is automatically upgraded. This includes the following behavior:

- Running R sessions are suspended so that future interactions with the server automatically launch the updated R session binary
- Currently connected browser clients are notified that a new version is available and automatically refresh themselves.
- The core server binary is restarted

Related Topics

- [Getting Started](#)
- [Configuring the Server](#)
- [Running with a Proxy](#)

© 2011 RStudio, Inc.

RStudio Server: Running with a Proxy

Overview

If you are running RStudio on a public network it is strongly recommended that you deploy RStudio behind another web server (e.g. Nginx or Apache) which acts as a reverse proxy to it. Doing this allows you to benefit from the the robust HTTP protocol handling built into the web server. This has both performance (e.g. keep-alive) and security (e.g. rejection of maliciously malformed requests) benefits.

Nginx Configuration

On Ubuntu a version of Nginx that supports reverse-proxying can be installed using the following command:

```
sudo apt-get install nginx
```

To enable an instance of Nginx running on the same server to act as a front-end proxy to RStudio you would add commands like the following to your `nginx.conf` file:

```
location / {
    proxy_pass http://localhost:8787;
    proxy_redirect http://localhost:8787/ $scheme://$host/;
}
```

After adding this entry you'll then need to restart Nginx so that the proxy settings take effect:

```
sudo /etc/init.d/nginx restart
```

Apache Configuration

To enable an instance of Apache running on the same server to act as a front-end proxy to RStudio you need to use the `mod_proxy`. The steps for enabling this module vary across operating systems so you should consult your distribution's Apache documentation for details.

On Ubuntu systems Apache can be installed with `mod_proxy` using the following commands:

```
sudo apt-get install apache2
sudo apt-get install libapache2-mod-proxy-html
sudo apt-get install libxml2-dev
```

Then, to update the Apache configuration files to activate `mod_proxy` you execute the following commands:

```
sudo a2enmod proxy
sudo a2enmod proxy_http
```

Once you have enabled `mod_proxy` in your Apache installation you need to add the required proxy commands to your `VirtualHost` definition. For example:

```
<VirtualHost *:80>

    <Proxy *>
        Allow from localhost
    </Proxy>

    ProxyPass          / http://localhost:8787/
    ProxyPassReverse  / http://localhost:8787/
```

```
</VirtualHost>
```

Note that if you want to serve RStudio from a custom path (e.g. /rstudio) you would replace the ProxyPass directives described above to:

```
ProxyPass      /rstudio/ http://localhost:8787/  
ProxyPassReverse /rstudio/ http://localhost:8787/  
RedirectMatch permanent ^/rstudio$ /rstudio/
```

Finally, after you've completed all of the above steps you'll then need to restart Apache so that the proxy settings take effect:

```
sudo /etc/init.d/apache2 restart
```

RStudio Configuration

Once you are successfully proxying requests from Nginx or Apache to RStudio you should change the port RStudio listens on from 0.0.0.0 (all remote clients) to 127.0.0.1 (only the localhost). This ensures that the only way to connect to RStudio is through the proxy server. You can do this by adding the `www-address` entry to the `/etc/rstudio/rserver.conf` file as follows:

```
www-address=127.0.0.1
```

Note that this config file does not exist by default so you may need to create it if it doesn't already exist.

Related Topics

- [Getting Started](#)
- [Configuring the Server](#)
- [Managing the Server](#)

Interactive Plotting with Manipulate

RStudio includes a `manipulate` package that enables the addition of interactive capabilities to standard R plots. This is accomplished by binding plot inputs to custom controls rather than static hard-coded values.

Basic Usage

The `manipulate` function accepts a plotting expression and a set of controls (e.g. slider, picker, or checkbox) which are used to dynamically change values within the expression. When a value is changed using its corresponding control the expression is automatically re-executed and the plot is redrawn.

For example, to create a plot that enables manipulation of a parameter using a slider control you could use syntax like this:

```
library(manipulate)
manipulate(plot(1:x), x = slider(1, 100))
```

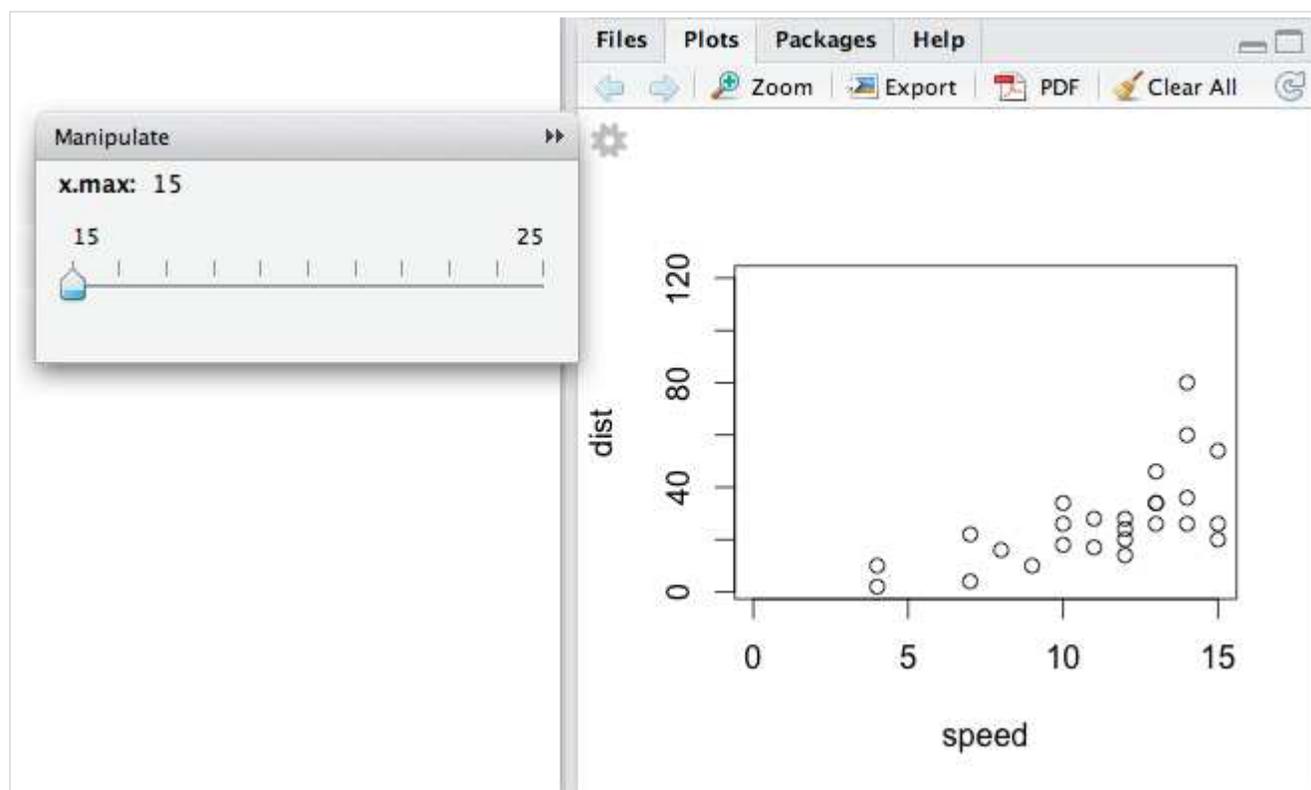
After this code is executed the plot is drawn using an initial value of 1 for `x`. A manipulator panel is also opened adjacent to the plot which contains a slider control used to change the value of `x` from 1 to 100.

Slider Control

The slider control enables manipulation of plot variables along a numeric range. For example:

```
manipulate(
  plot(cars, xlim=c(0,x.max)),
  x.max=slider(15,25))
```

Results in this plot and manipulator:



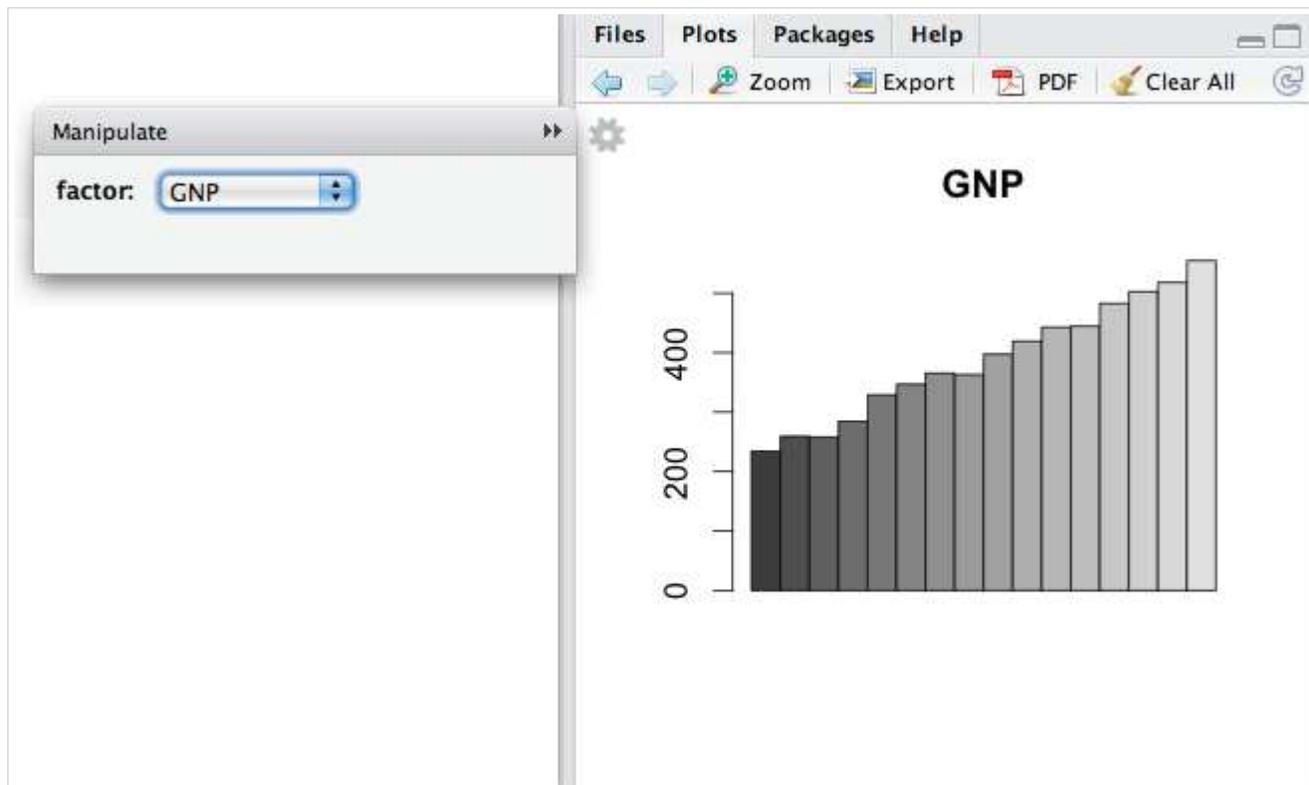
Slider controls also support custom labels and step increments.

Picker Control

The picker control enables manipulation of plot variables based on a set of fixed choices. For example:

```
manipulate(
  barplot(as.matrix(longley[, factor]),
    beside = TRUE, main = factor),
  factor = picker("GNP", "Unemployed", "Employed"))
```

Results in this plot and manipulator:



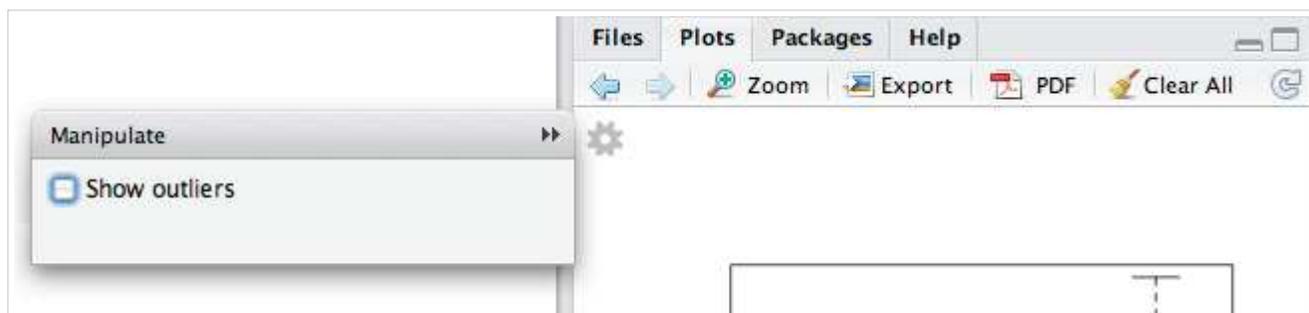
Picker controls support arbitrary value types, and can also include custom user-readable labels for each choice.

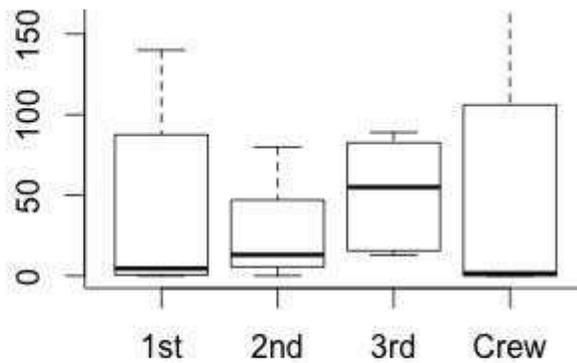
Checkbox Control

The checkbox control enables manipulation of logical plot variables. For example:

```
manipulate(
  boxplot(Freq ~ Class, data = Titanic, outline = outline),
  outline = checkbox(FALSE, "Show outliers"))
```

Results in this plot and manipulator:





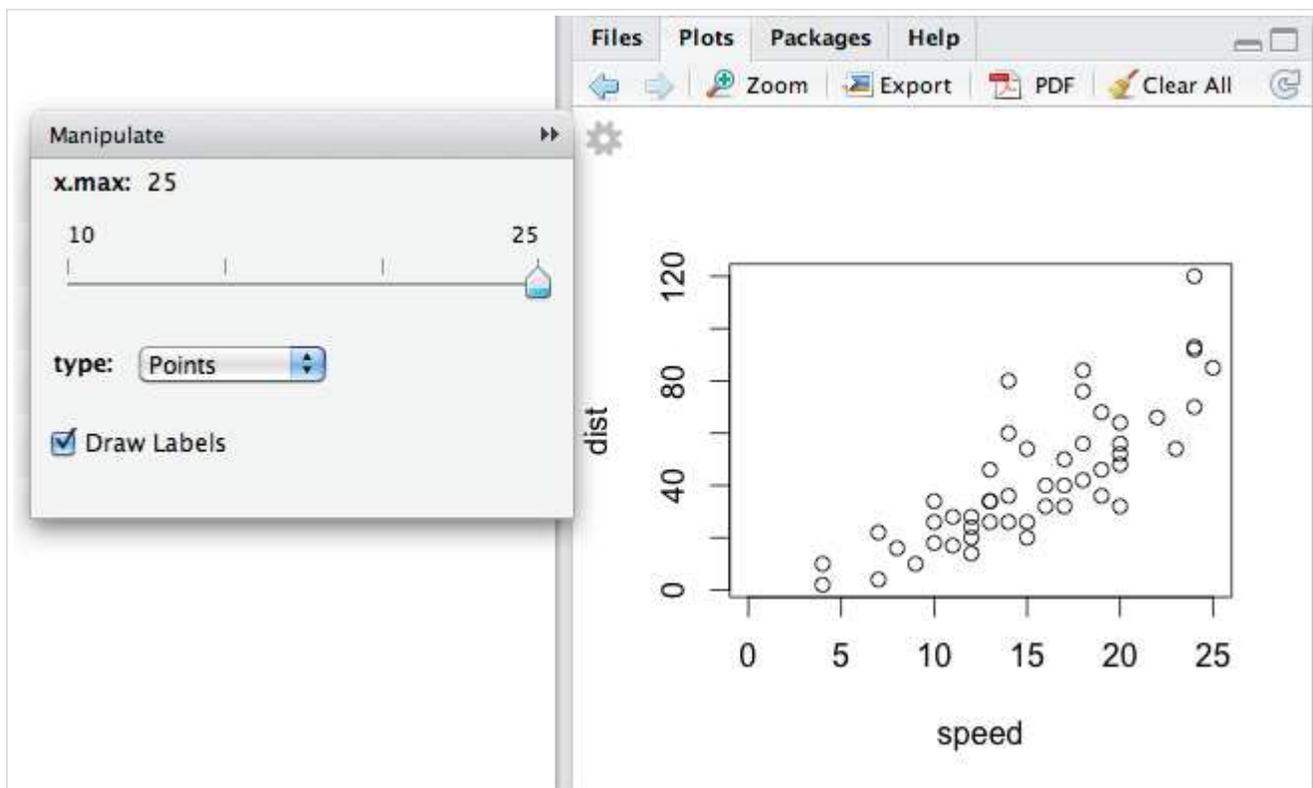
The `manipulate` package documentation contains additional details on all of the options available for the various control types.

Combining Controls

Multiple controls can be combined within a single manipulator. For example:

```
manipulate(
  plot(cars, xlim = c(0, x.max), type = type, ann = label),
  x.max = slider(10, 25, step=5, initial = 25),
  type = picker("Points" = "p", "Line" = "l", "Step" = "s"),
  label = checkbox(TRUE, "Draw Labels"))
```

Results in this plot and manipulator:



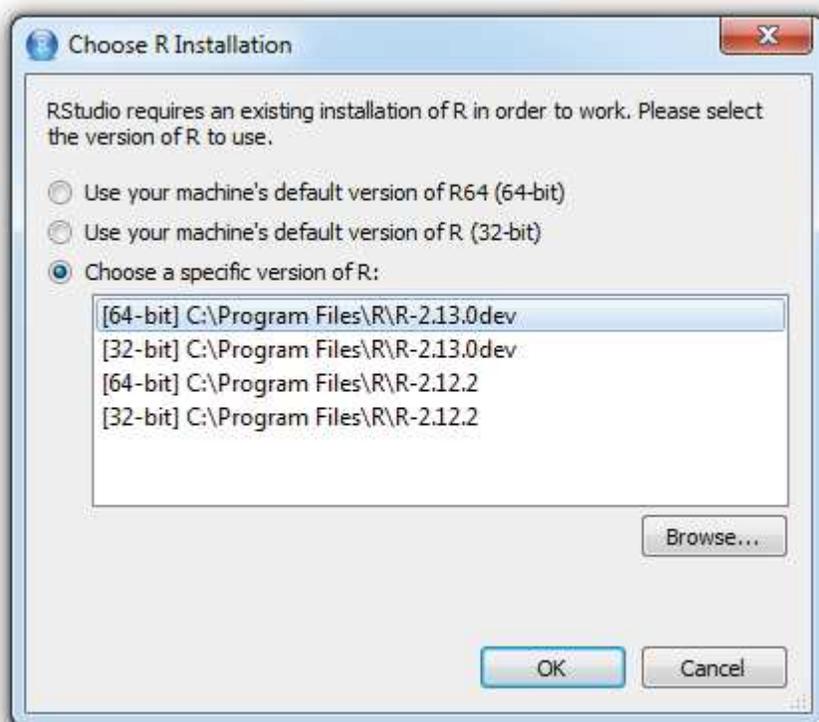
Using Different Versions of R

RStudio requires R version 2.11.1 or higher. Since R versions can be installed side-by-side on a system, RStudio needs to select which version of R to run against. The way this occurs varies between platforms—this article covers how version selection is handled on each platform.

Windows

On Windows, RStudio uses the system's current version of R by default. When R is installed on Windows it writes the version being installed to the Registry as the "current" version of R (the specific registry keys written are described [here](#)). This is the version of R which RStudio runs against by default.

You can override which version of R is used via General panel of the RStudio Options dialog. This dialog allows you to specify that RStudio should always bind to the default 32 or 64-bit version of R, or to specify a different version altogether:



Note that by holding down the Control key during the launch of RStudio you can cause the R version selection dialog to display at startup.

Mac OS X

R from CRAN

On Mac OS X if the only version of R you have installed is the standard R distribution from CRAN then RStudio will by default run against the current version of R.Framework. You can list all of the versions of R.Framework on your system and determine which one is considered the current one by executing the following command:

```
ls -l /Library/Frameworks/R.framework/Versions/
```

To change the current version of R.Framework you can either:

- Run the installer from CRAN for the R version you want to be current

- Use the RSwitch utility available at: <http://r.research.att.com/>
- Update the R.framework/Versions/Current directory alias directly using `ln -s`

R from source (including MacPorts and Homebrew)

When R is installed from CRAN on OS X the R executable is installed at `/usr/bin/R`. However, if R is installed directly from source or via a package manager like MacPorts or Homebrew, then the R executable is installed to either `/usr/local/bin/R` (Homebrew) or `/opt/local/bin/R` (MacPorts). In order to support these variations, RStudio scans for the R executable in the following sequence:

1. `/opt/local/bin/R`
2. `/usr/bin/R`
3. `/usr/local/bin/R`

This order is based on the conventional ordering of the OS X PATH environment variable, and therefore should normally yield the same version that is run when R is executed from a terminal.

If you want to override the version of R selected by RStudio's default behavior then you can set the `RSTUDIO_WHICH_R` environment variable to the R executable that you want to run against. For example, to force RStudio to use the R executable located at `/usr/local/bin`:

```
export RSTUDIO_WHICH_R=/usr/local/bin/R
```

Note that in order for RStudio to see this environment variable it needs to be added to the OS X `environment.plist` file. Instructions for editing this file are [available here](#).

Linux

On Linux, RStudio uses the version of R pointed to by the output of the following command:

```
which R
```

The `which` command performs a search for the R executable using the system PATH. RStudio will therefore by default bind to the same version that is run when R is executed from a terminal.

For versions of R installed by system package managers (e.g. `r-base` on Ubuntu) this will be `/usr/bin/R`. For versions of R installed from source this will typically (but not always) be `/usr/local/bin/R`.

If you want to override which version of R is used then you can set the `RSTUDIO_WHICH_R` environment variable to the R executable that you want to run against. For example:

```
export RSTUDIO_WHICH_R=/usr/local/bin/R
```

Not that in order for RStudio to see this environment variable when launched from the Ubuntu desktop Applications menu (as opposed to from a terminal) it must be defined in the `~/.profile` file.

Web

If you are running RStudio within a web browser then the version of R is determined by whatever version of R is running alongside RStudio Server. The version currently in use on the server can be printed using the following command:

```
> R.version.string
```

Character Encoding

Starting with version 0.93, RStudio supports non-ASCII characters for input and output.

Console

Unicode characters can be used for both input and output in the console.

Source Editor

The RStudio source editor natively supports Unicode characters. It will allow you to type or paste characters from any language, even ones that are not part of the document's character set. RStudio will allow you to save such documents, but will print a warning to the R console that not all characters could be encoded. If you close the document without re-saving in a more suitable encoding, those characters will be lost.

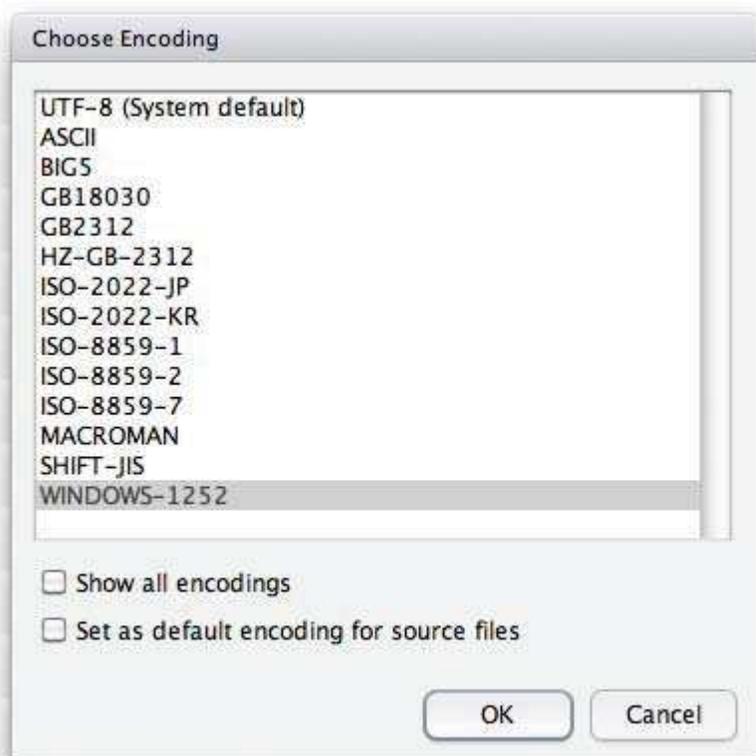
If in doubt about which encoding to use, use UTF-8, as it can encode any Unicode character.

Reading and Writing Files

The RStudio source editor can read and write files using any character encoding that is available on your system:

- You can choose the encoding for reading with File : Reopen with Encoding, which will re-read the current file from disk with the new encoding.
- You can also save an open file using a different encoding with File : Save with Encoding.

The Reopen and Save with Encoding commands both display the following dialog:



Setting the Default Encoding

If you frequently use the character set, check "Set as default encoding for source files". You can view or change this default in the Tools : Options (for Windows & Linux) or Preferences (for Mac) dialog, in the Editing section.

If you don't set a default encoding, files will be opened using UTF-8 (on Mac desktop, Linux desktop, and server) or the system's default encoding (on Windows). When saving a previously unsaved file, RStudio will ask you to

choose an encoding if non-ASCII characters are present.

Known Issues

- If you call `Sys.setlocale` with "LC_CTYPE" or "LC_ALL" to change the system locale while RStudio is running, you may run into some minor issues as RStudio assumes the system encoding doesn't change. If you are on Windows, we recommend you only call `Sys.setlocale` in `.Rprofile`. If you are on Mac or Linux and want to change the system locale, please visit the [support forum](#) and let us know your scenario.
- On Windows, R's `source` function does not work with files that include characters that aren't part of the current system encoding. You may have trouble with RStudio's Run All and Source on Save commands, as they rely on `source`.

© 2011 RStudio, Inc.

Optimizing your Browser for RStudio

NOTE: This article is only applicable if you are using the RStudio IDE within a web browser (as opposed to using RStudio as a standalone desktop application).

Run a Recent Browser Version

RStudio makes use of a number of advanced web browser features and as a result benefits from running within more up-to-date browser versions. The following are the recommend minimum versions for various browsers:

- [Firefox 3.5](#)
- [Safari 4.0](#)
- [Google Chrome 5.0](#)

Note that RStudio can also be run from within Internet Explorer using the [Google Chrome Frame](#) browser plugin

Disable Pop-Up Blockers

There are a number of instances where RStudio needs to show a new external popup window (e.g to display a PDF file). We therefore recommend that you disable pop-up blocking for the RStudio domain. Most browsers will prompt you regarding whether you want to enable popups for RStudio the first time one is blocked. Some browsers (such as Safari) may require you to globally enable and disable popups.

Safari and Chrome: Disable Browser Spell Checking

If you are using Safari or Chrome, you may find it desirable to disable "Check Spelling While Typing" (available from the Edit menu) since many of the words you enter in the Source and Console will not be in the built-in dictionary and thus will show up with a red underline when entered.

Firefox for Mac: Install Inline PDF Extension

Firefox for the Mac does not display PDFs inline by default (rather they are downloaded like any other file). RStudio uses PDFs for both printing plots as well as for Sweave/LaTeX documents and having them display inline is much preferred. To enable this you should install the following Firefox extension:

<http://code.google.com/p/firefox-mac-pdf/>

Uploading and Downloading Files

NOTE: This article is only applicable if you are using the RStudio IDE within a web browser (as opposed to using RStudio as a standalone desktop application).

Uploading Files

To upload datasets, scripts, or other files to RStudio Server you should take the following steps:

1. Switch to the Files pane
2. Navigate to the directory you wish to upload files into
3. Click the Upload toolbar button
4. Choose the file you wish to upload and press OK

Note that if you wish to upload several files or even an entire folder, you should first compress your files or folder into a zip file and then upload the zip file (when RStudio receives an uploaded zip file it automatically uncompresses it).

Downloading Files

To download files from RStudio Server you should take the following steps:

1. Switch to directory you want to download files from within the Files pane
2. Select the file(s) and/or folder(s) you want to download
3. Click More -> Export on the toolbar
4. You'll then be prompted with a default file name for the download. Either accept the default or specify a custom name then press OK.

Note that if you select multiple files or folders for download then RStudio compresses all of the files into a single zip archive for downloading.

About RStudio

The RStudio Project

We started the RStudio project because we were excited and inspired by R. The [creators of R](#) provided a flexible and powerful foundation for statistical computing; then made it free and open so that it could be improved collaboratively and its benefits could be shared by the widest possible audience.

It's better for everyone if the tools used for research and science are free and open. Reproducibility, widespread sharing of knowledge and techniques, and the leveling of the playing field by eliminating cost barriers are but a few of the shared benefits of free software in science.

RStudio is an integrated development environment (IDE) for R which works with the standard version of R available from CRAN. Like R, RStudio is available under a free software license. Our goal is to develop a powerful tool that supports the practices and techniques required for creating trustworthy, high quality analysis. At the same time, we want RStudio to be as straightforward and intuitive as possible to provide a friendly environment for new and experienced R users alike. RStudio is also a company, and we plan to sell services (support, training, consulting, hosting) related to the open-source software we distribute.

We're looking forward to joining the R community, learning from users, growing the product, and hopefully making a meaningful contribution to the practice of research and science.

Our Team



JJ Allaire

JJ Allaire is a software engineer and entrepreneur who has created a wide variety of products including [ColdFusion](#), [Windows Live Writer](#), [Lose It!](#), and RStudio.



Joe Cheng

Joe Cheng is a software engineer who has worked at a number of startups including Allaire, Upromise, and Onfolio. Most recently, he worked at Microsoft as the development lead for [Windows Live Writer](#).



Josh Paulson

Josh Paulson is a product manager who has been working with R for over 4 years, focusing principally on data visualization and financial modeling applications.



Paul DiCristina

Paul DiCristina is a designer with broad experience including consumer, enterprise, mobile, and web. Paul's design portfolio includes RStudio, Lose It! and many other apps and sites.

Contact Us

Support: <http://support.rstudio.org>

Feedback: feedback@rstudio.org

Inquiries: info@rstudio.org

RStudio v0.93 — Release Notes

April 11th, 2011

New Features

Source editor enhancements

We've added some new features & options to the source editor. We've also received lots of feedback on more advanced capabilities users want in the editor and we will definitely address this in upcoming releases. New stuff in the editor includes:

- Highlight all instances of selected text
- Insert spaces for tabs (soft-tabs)
- Customizable print margin line
- Selected line highlight
- Toggle line numbers on/off
- Optional soft-wrapping for R source files

The docs on [source code editing options](#) include more details.

Customizable layout and appearance

One of the most frequently requested features we've have is the ability to put the Console and Source views side-by-side. This configuration (and others) are now possible. New appearance and layout options include:

- Customize locations of panes and tabs
- Change default font size for code views
- Select from various editor themes including TextMate, Eclipse, and others.

For more details see the docs on [appearance and layout options](#).

Interactive plotting (manipulate)

This release includes a package called `manipulate` that can be use to create interactive plots within RStudio. `Manipulate` is very flexible and includes the following capabilities:

- Generate plots with inputs bound to custom controls (rather than being hard-coded to a single value)
- Variety of control types including slider, picker, and checkbox.
- Controls appear next to the plot and can be easily shown and hidden

More details as well as screenshots with examples can be found in the [manipulate documentation](#).

Works with R installed from source

The first beta of RStudio was compatible with the binary version of R distributed from CRAN. The current release works with any version of R, including:

- R built and installed from source using `make install`
- MacPorts or Homebrew versions of R on MacOS X

The docs on [using different versions of R](#) describe how RStudio determines which R to run against on each platform.

Character encoding

In this release we've significantly improved handling of non-ASCII characters, this includes:

- Unicode characters can be used for input and output in the console.
- The source editor supports Unicode characters and can open/save files using any character encoding.
- A product-wide default encoding can be set, and can be overridden on a per-document basis.

See the [character encoding](#) documentation for more details.

Improved management of working directories

We've added a number of new features to make it easier to switch between working contexts located in different directories. These include:

- Option to specify a default initial working directory
- Tools | Change Working Dir menu command to change both the working directory and Files pane.
- Optional file associations for .RData and .R that initialize RStudio within the opened file's directory
- Windows: Startup in working directory specified for shortcuts
- Mac: Startup in folder dragged and dropped on RStudio Dock icon
- Linux: Startup in terminal working directory when run from the command line

The docs on [working directories](#) and [workspaces](#) go into more depth on these features.

Other Enhancements

Console

- Recognize `\r` character in console so that `txtProgressBar` works as expected
- Lift restrictions on size of console input which can be sent to R (was 4K total, is now 4K per line)
- Jump to next non-blank line in source after executing via `Ctrl-Enter`

Source Editing

- Improved size and legibility of default fonts on Windows & Linux
- New keyboard shortcuts:
 - `Alt-` for inserting assignment ("`<-`") operator.
 - `Ctrl+Shift+Home/End` for select to start/end
 - `Ctrl+Shift+P` for Compile PDF
- Add "return" to list of symbols syntax highlighted as a keyword

Compatibility

- Compatible with R 2.11.1 on Mac (previously required R 2.12)
- Added `CFBundleSignature` to Mac version
- Correctly initialize `memory.limit` to available physical memory on 64-bit Windows
- Ensure that R uses Internet2 on Windows for interoperability with proxy servers.
- Compatibility with changes to the R 2.13 internal web server (pass headers to custom handlers).
- Allow RStudio desktop to run under root account
- Improved support for openSUSE (still requires install from source):
 - Added `install-dependencies-zypper` script
 - Added `init.d` script for daemon management

Packaging and Installation

- Added `RStudio.version` function to show current version of RStudio

- Changed name of RStudio binary from rdesktop to rstudio (avoid conflict with existing rdesktop binary)
- Added /usr/bin/rstudio soft-link
- Change DEB and RPM dependency on base R package to "recommends" rather than "depends"
- Made it more straightforward to install from source:
 - Eliminated git pull requirement (can now build directly from tarball)
 - Optionally use system package manager installed versions of Qt4 & Boost

Miscellaneous

- Added Tools menu with Interrupt R, Change Working Dir, and Options commands.
- Add support for loading .rda files into Workspace.
- Improved file icons including new custom icons for Rnw and Rd files.
- Respect both R_USER and HOME environment variables for determining location of R home directory
- Render plot changes on calls to Sys.sleep (enables animated plots)
- Workaround Ubuntu TeX ~ substitution bug by using pdflatex rather than texi2dvi

Bug Fixes

Console

- Workspace restored message prints at startup even if no workspace was restored
- Numeric keypad Enter and navigation keys not correctly interpreted by console
- Esc key not always correctly interpreted when attempting to exit from incomplete command.

Source Editing

- Source pane can become fully selected and impossible to unselect.
- Control-Enter to execute sometimes results in selection not updating properly
- Jump to Word (Ctrl+Right) doesn't navigate past '[' character.
- Ctrl+Backspace doesn't delete previous word on Windows
- Characters illegible when Lucida Grande is installed on Mac systems
- Active tab in source mode sometimes hidden or partially obscured
- Control+S repeats last Undo/Redo on Firefox 3.6
- Print from source view not working in Firefox 4
- Eliminate key binding conflicts for international keyboard layouts
- Incorrect shortcut key displayed in tooltip for Run from source commands
- Variables with dots (".") in their names not highlighting on double-click.

Plotting

- Graphics device not reselected after closing other device (e.g. pdf or png device)
- X11 device and rgl package not working properly on Mac.
- Save as PNG command not working on Linux
- Re-entrant plot rendering routine causes crash for plots which take a long time to be rendered.
- plotmath expressions not rendered correctly
- Plot history can grow arbitrarily large and cause disk/memory problems.

Compatibility

- rJava package not working on Linux due to incomplete LD_LIBRARY_PATH

- **Not always correctly detecting whether TeX is installed**
- **Dependency on psmisc package not specified for RStudio Server**
- **Incorrect file association for download of RPM on Fedora**

Miscellaneous

- **Failed to start when running behind some proxy server configurations.**
- **Unable to initialize from .Rhistory file that is owned by root**
- **Exit delay of 2-3 seconds in Mac version**
- **Crash when custom gtk theme contains missing or invalid images for standard icons**
- **Unresponsiveness when viewing extremely large data frames**
- **Conflicting RStudio instances running in parallel if launched from different executable paths**

© 2011 RStudio, Inc.

Frequently Asked Questions

What is RStudio?

RStudio is an integrated development environment (IDE) for R which works with the standard version of R available from CRAN. RStudio includes a wide range of productivity enhancing features and runs on all major platforms. RStudio can optionally also be run as server which enables you to provide a browser based interface to a version of R running on a remote system.

What versions of R is RStudio compatible with?

In order to run RStudio you need to have already installed R 2.11.1 or higher. You can download the most recent version of R for your environment from [CRAN](#).

What is the best way to get started with R and RStudio?

To get started with R there are a wide variety of learning and reference materials available online. We recommend you check out the [Getting Help with R](#) article to find the appropriate resources.

To get started using RStudio check out the [Product Overview](#) as well as the other articles linked to from the [documentation](#).

What is the difference between RStudio Desktop and RStudio Server?

RStudio Desktop is an R IDE that works with the version of R you have installed on your local Windows, Mac OS X, or Linux workstation. RStudio Desktop is a standalone desktop application that in no way requires or connects to RStudio Server.

RStudio Server is a Linux server application that provides a web browser based interface to the version of R running on the server. For more on why you might want to deploy an RStudio Server see the [server documentation](#).

Will RStudio be providing a hosted version of RStudio Server?

Yes, we do plan to provide a hosted version of RStudio Server so that customers can use RStudio over the web without having to deploy their own servers. We haven't announced a specific timeframe for this service yet, however if you are interested in testing it when the beta version becomes available please contact us at info@rstudio.org.

What license is RStudio available under?

RStudio is available under the [GNU Affero General Public License v3](#). The AGPL v3 is an open-source license that guarantees the freedom to share and change the software, and to make sure it remains free software for all its users.

Where can I find the latest updates and news about RStudio?

To keep up with the latest RStudio developments you can subscribe to our [blog](#) or [follow us](#) on Twitter.

Where can I get support for using RStudio?

Our [support web site](#) includes a knowledge base and discussion forum for reporting problems, asking questions and discussing new ideas and features. Our team and others in the community actively participates in the forum so it is a great place to go for help or answers.

Getting Help with R

There are a number of good resources available on the web for both learning R and seeking answers to questions about how to accomplish various tasks. This article summarizes a few of the more helpful ones.

Learning R

If you are just learning R there are a number of good places to start:

- This [basic R tutorial](#) takes you step-by-step through the core functions of R.
- The CRAN [Introduction to R](#) provides a more complete and detailed overview of the entire R language.
- This article provides a nice [introduction to R for those coming from other languages](#).
- [The R Reference Card](#) provides a useful quick reference for how to perform common tasks in R.
- The Carnegie Mellon Open Learning Initiative has a free online [Introduction to Statistics](#) course has an option to do the exercises and labs using R.

If you have some familiarity with R and want to learn about the system or particular features in more depth these resources might be helpful.

- This Stack Overflow question provides some pointers to good [books for learning the R language](#).
- The CRAN [Contributed Documentation](#) page lists other manuals, tutorials, etc. provided by users of R.
- Once you've gained some familiarity with R, [The R Inferno](#) provides an entertaining roadmap to some of the deeper subtleties of the language and how to work with it most effectively.
- The [Google R Style Guide](#) provides some guidelines for writing readable and maintainable R code.

Asking Questions

A great place to start for any question about R is the [RSeek meta search engine](#), which provides a unified interface for searching the various sources of online R information. If there is an answer to question already available there is a good chance that RSeek can locate it.

If you aren't able to locate an answer using RSeek, the following are good places to find an answer or ask a question:

- The [R-help mailing list](#) is a very active list with questions and answers about problems and solutions using R. Before posting to the list you can also [search the list archives](#) to see if an answer already exists.
- Stack Overflow is also becoming an increasingly important resource for seeking [answers to questions about R](#).
- If you have a question that is more about statistical methodology there are also plenty of R users active on the [CrossValidated Q&A community](#).

Finding Packages

There are thousands of R packages [available from CRAN](#) but navigating them all can be a challenge. The following resources can help find the packages most appropriate for your tasks:

- [CRAN Task Views](#) provides comprehensive summaries of the packages most commonly used in various disciplines.
- [crantastic](#) is a community site for R packages where you can search for, review, and tag CRAN packages.

News and Information

The R community is growing rapidly and there are lots of new things happening all the time. If you want to stay on top of what's happening we recommend keeping up with the following sites:

- [R-bloggers](#) is a news site that combines posts from over 140 R bloggers. Almost everything that happens in the R community is mentioned and discussed on R-bloggers.
- Users in the R community also frequently record videos of presentations, seminars, and user-groups. The

R Videos channel run by Drew Conway is a great way to keep up with all of the available videos.

© 2011 RStudio, Inc.



How can we help?

This is the home of support for RStudio.

You can search our knowledge base articles, browse public discussions, or create a new discussion if you're having trouble.

If you take the time to create an account, you'll be able to post private discussions (that only yourself and our support team can see) and track your existing discussions. However, if you don't want to create an account, you can still create a public discussion in our discussion area.

Support Staff

Representatives from the company here to listen to your issues

Joe Cheng Engineer

Josh Paulson Product Manager

JJ Allaire Engineer

Start a discussion

Browse discussions

Knowledge Base

RECENT DISCUSSIONS

- 25 MAY 10:16 Sweave Master Document
- 25 MAY 09:00 New install of RStudio can't find \$(R_HOME_DIR)\doc
- 25 MAY 08:58 build failed

RECENT ARTICLES

- RStudio Release History
- Creating a Portable Version of RStudio for a USB Drive
- Getting Help with R
- RStudio Application Logs
- How to Get Sweave Working with RStudio on Red Hat

rstudio is using GitHub to share code with you!

GitHub is more than just a place to share code. It's a place to keep tabs on your favorite developers and projects, easily contribute fixes and new features, and visualize what's going on inside your codebase!

Sign Up Now

FREE FOR OPEN SOURCE

rstudio / rstudio

Watch Fork 198 13

Source Commits Network Pull Requests (0) Graphs Branch: master

Switch Branches (17) * Switch Tags (133) * Branch List

RStudio is an integrated development environment (IDE) for R — Read more
<http://www.rstudio.org>

Downloads

HTTP Git Read-Only Read-Only access

make ExportPlot non-abstract

 jjallaire (author)
6 minutes ago

```
commit 27e0c32d14eb3f2dc13b
tree 35a78fceb4e90b1f4876
parent 67655f2f1d92c8c7701f5
```

rstudio /

name	age	message	history
cmake/	March 15, 2011	improvements to r version checking at configure ti...	[jjallaire]
dependencies/	April 12, 2011	Fix install-dependencies.cmd script [jcheng5]	
package/	April 11, 2011	debian packaging: add libc6 dependency; use fake...	[jjallaire]
src/	6 minutes ago	make ExportPlot non-abstract [jjallaire]	
.gitignore	April 12, 2011	ignore eclipse gwt log [jcheng5]	
.gitmodules	February 16, 2011	Improve ace packaging. Fix Safari 4 [jcheng5]	
CMakeGlobals.txt	2 days ago	correct detection of ubuntu for pam helper [jjallaire]	
CMakeLists.txt	February 09, 2011	update installer reference to README to include md...	[jjallaire]
COPYING	December 13, 2010	create SOURCE doc file and dynamically configure i...	[jjallaire]
INSTALL	March 15, 2011	clarifications for INSTALL doc [jjallaire]	
NOTICE	April 28, 2011	update NOTICE file for solarized theme [jjallaire]	
README.md	March 14, 2011	Edited README.md via GitHub [jjallaire]	
SOURCE.in	February 08, 2011	in SOURCE point to tree rather than tarball [jjallaire]	

README.md

RStudio

RStudio is an integrated development environment (IDE) for the [R programming language](#). Some of its features include:

- Customizable workbench with all of the tools required to work with R in one place (console, source, plots, workspace, help, history, etc).
- Syntax highlighting editor with code completion.
- Execute code directly from the source editor (line, selection, or file).
- Full support for authoring Sweave and TeX documents.
- Runs on all major platforms (Windows, Mac, and Linux) and can also be run as a server, enabling multiple users to access the RStudio IDE using a web browser.

For more information on RStudio please visit the [project website](#).

Getting the Code

RStudio is licensed under the AGPLv3, the terms of which are included in the file COPYING. You can find our source code repository on GitHub at <https://github.com/rstudio/rstudio>.

Documentation

For information on how to use RStudio check out our [online documentation](#).

For documentation on running your own RStudio Server see the [server getting started guide](#).

See also the following files included with the distribution:

- COPYING - RStudio license (AGPLv3)
- NOTICE - Additional open source software included with RStudio
- SOURCE - How to obtain the source code for RStudio
- INSTALL - How to build and install RStudio from source

If you have problems or want to share feedback with us please visit our [support website](#). For other inquiries you can also email us at info@rstudio.org.

RStudio Beta 2 (v0.93)

April 11, 2011 in [News](#) | by [jjallaire](#) | [23 comments](#)

RStudio Beta 2 (v0.93) is available for download today. We've gotten incredibly helpful input from the R community and this release reflects a lot of that feedback.

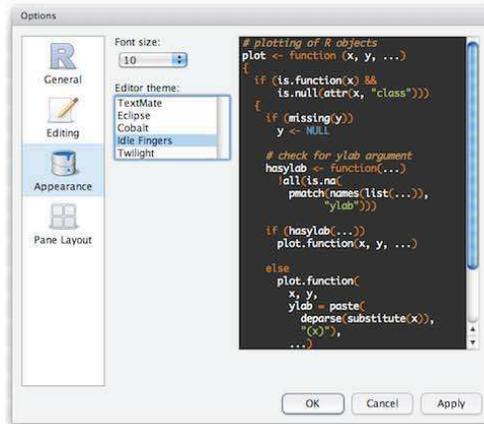
The [release notes](#) have the full details on what's new. Some of the highlights include:

Source Editor Enhancements

- Highlight all instances of selected text
- Insert spaces for tabs (soft-tabs)
- Customizable print margin line
- Selected line highlight
- Toggle line numbers on/off
- Optional soft-wrapping for R source files

Customizable Layout and Appearance

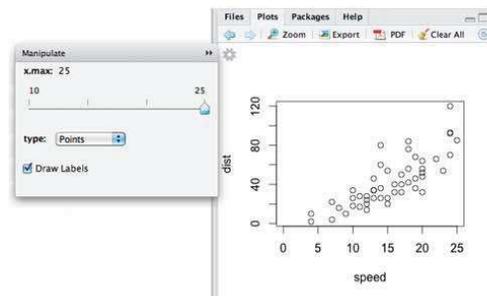
- The layout of panes and tabs is now configurable (enabling side-by-side source and console view, among others).
- Support for a variety of editing themes, including TextMate, Eclipse, and others.



Interactive Plotting

This release features `manipulate`, a new interactive plotting feature that enables you to create plots with inputs bound to custom controls (e.g. slider, picker, etc.) rather than hard-coded to a single value. For example:

```
manipulate(  
  # plot expression  
  plot(cars, xlim = c(0, x.max), type = type, ann = label),  
  # controls  
  x.max = slider(10, 25, step = 5, initial = 25),  
  type = picker("Points" = "p", "Line" = "l", "Step" = "s"),  
  label = checkbox(TRUE, "Draw Labels")  
)
```



More

- RStudio now works with versions of R installed from source (either via `make install` or packaged by MacPorts, Homebrew, etc.).
- Enhanced support for Unicode and non-ASCII character encodings.
- Improved working directory management including new options for default behavior, support for shell "open with" context menus, and optional file associations for common R file types (RData, R, Rnw).
- Many other small enhancements and bug fixes (see the [release notes](#) for full details).

We hope you try out the new release and keep talking to us on our [support forum](#) about what works, what doesn't, and what else you'd like RStudio to do.

RStudio, new open-source IDE for R

February 28, 2011 in [News](#) | by [jjallaire](#) | [75 comments](#)

SEARCH

ABOUT

RStudio™ is an open-source integrated development environment (IDE) for R.

LINKS

[RStudio Website](#)
[RStudio Support Development](#) @ [Github](#)
[Contact Us](#)

CATEGORIES

[Meta](#)
[News](#)

ARCHIVES

[April 2011](#)
[February 2011](#)



Twitter



RSS

EMAIL SUBSCRIPTION

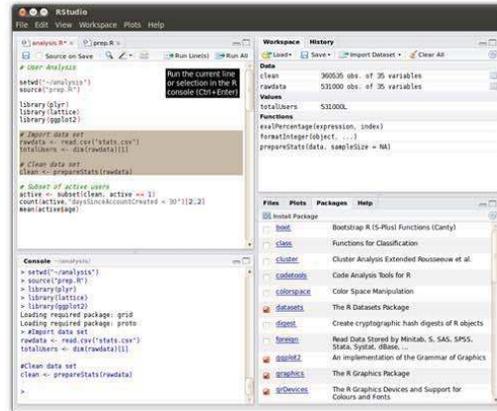
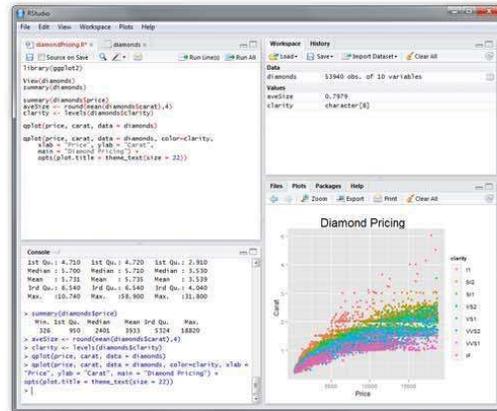
Enter your email address to subscribe to this blog and receive notifications of new posts by email.

Sign me up!

RStudio is a new open-source IDE for R which we're excited to announce the availability of today. RStudio has interesting features for both new and experienced R developers including code completion, execute from source, searchable history, and support for authoring Sweave documents.

RStudio runs on all major desktop platforms (Windows, Mac OS X, Ubuntu, or Fedora) and can also run as a server which enables multiple users to access the IDE using a web browser.

A couple of screenshots (click here for more screenshots):



The version of RStudio available today is a beta (v0.92) and is released under the GNU AGPL license. We're hoping for lots of input and dialog with the R community to help make the product as good as it can be!

More details on the project as well as download links can be found at: <http://www.rstudio.org>.

About the RStudio Project

February 27, 2011 in [Meta](#) | by [J.J. Allaire](#) | [Leave a comment](#)

We started the RStudio project because we were excited and inspired by R. The creators of R provided a flexible and powerful foundation for statistical computing; then made it free and open so that it could be improved collaboratively and its benefits could be shared by the widest possible audience.

It's better for everyone if the tools used for research and science are free and open. Reproducibility, widespread sharing of knowledge and techniques, and the leveling of the playing field by eliminating cost barriers are but a few of the shared benefits of free software in science.

RStudio is an integrated development environment (IDE) for R which works with the standard version of R available from CRAN. Like R, RStudio is available under a free software license. Our goal is to develop a powerful tool that supports the practices and techniques required for creating trustworthy, high quality analysis. At the same time, we want RStudio to be as straightforward and intuitive as possible to provide a friendly environment for new and experienced R users alike. RStudio is also a company, and we plan to sell services (support, training, consulting, hosting) related to the open-source software we distribute.

We're looking forward to joining the R community, learning from users, growing the product, and hopefully making a meaningful contribution to the practice of research and science.

Welcome to our Weblog

February 27, 2011 in [Meta](#) | by [J.J. Allaire](#) | [10 comments](#)

Welcome to the RStudio weblog! We'll use the weblog to talk about both the product and its features as well as broader issues that concern the R community.