# rstudioguide

*Asher Spector*

*February 24, 2018*

This guide will help users set up and install RStudio. It assumes they are already somewhat familiar with R: for Harvard students, at around the Gov 2000/2001 level. If you already have RStudio installed and are comfortable using it, feel free to skip this section.

# 1. Why use RStudio

RStudio has two main advantages, which is why we recommend it for package development.

First, RStudio is generally built to make users' lives easier. It automatically displays everything from the environment (the list of all defined variables) to your working directory, makes graphics more accessible, and also generally makes it easier to write scripts.

Second, RStudio has a suite of tools which make it easy to build well-documented and comprehensible packages. Since this guide discusses package development, RStudio is the natural tool of choice.

# 2. Setting up RStudio

## Setting up R

Presumably you've already been working with R, but you should make sure you have a recent enough version of R intalled. To do so, type "R.Version()" into the editor you've been working with previously. It should return an output titled "$version.string$", for example:

```
R.Version()
```

```
## $platform
## [1] "x86_64-w64-mingw32"
##
## $arch
## [1] "x86_64"
##
## $os
## [1] "mingw32"
##
## $system
## [1] "x86_64, mingw32"
##
## $status
## [1] ""
##
## $major
## [1] "3"
##
## $minor
## [1] "4.2"
##
## $year
## [1] "2017"
##
## $month
## [1] "09"
##
## $day
## [1] "28"
##
## $`svn rev`
## [1] "73368"
##
## $language
## [1] "R"
##
## $version.string
## [1] "R version 3.4.2 (2017-09-28)"
##
## $nickname
## [1] "Short Summer"
```

If the number after the R version string is anything lower than 3.0.1, you'll have to install a new version of R in order to use R Studio. To do so, follow the steps below:

1. Go to https://cran.rstudio.com/ (https://cran.rstudio.com/) and select "Download R for [Your Operating System]."
2. Click "base" and click "Download R [latest version] for [Your Operating System]" at the top of the page.
3. You can then let the installer run itself - the default settings should be fine for our purposes.

maybe use this? https://www.r-statistics.com/2015/06/a-step-by-step-screenshots-tutorial-for-upgrading-r-on-windows/ (https://www.r-statistics.com/2015/06/a-step-by-step-screenshots-tutorial-for-upgrading-r-on-windows/)

# Downloading Development Tools

You’ll also want to download some packages and tools that are important for package development. If you are running Windows, you can do so by following the instructions below:

1. Go to https://cran.rstudio.com/ (https://cran.rstudio.com/) and select “Download R for Windows.”"
2. Click “RTools” and download the latest version of the tools (or the tools that are compatible with your version of R).
3. Let the installer run itself (the defaults are fine).

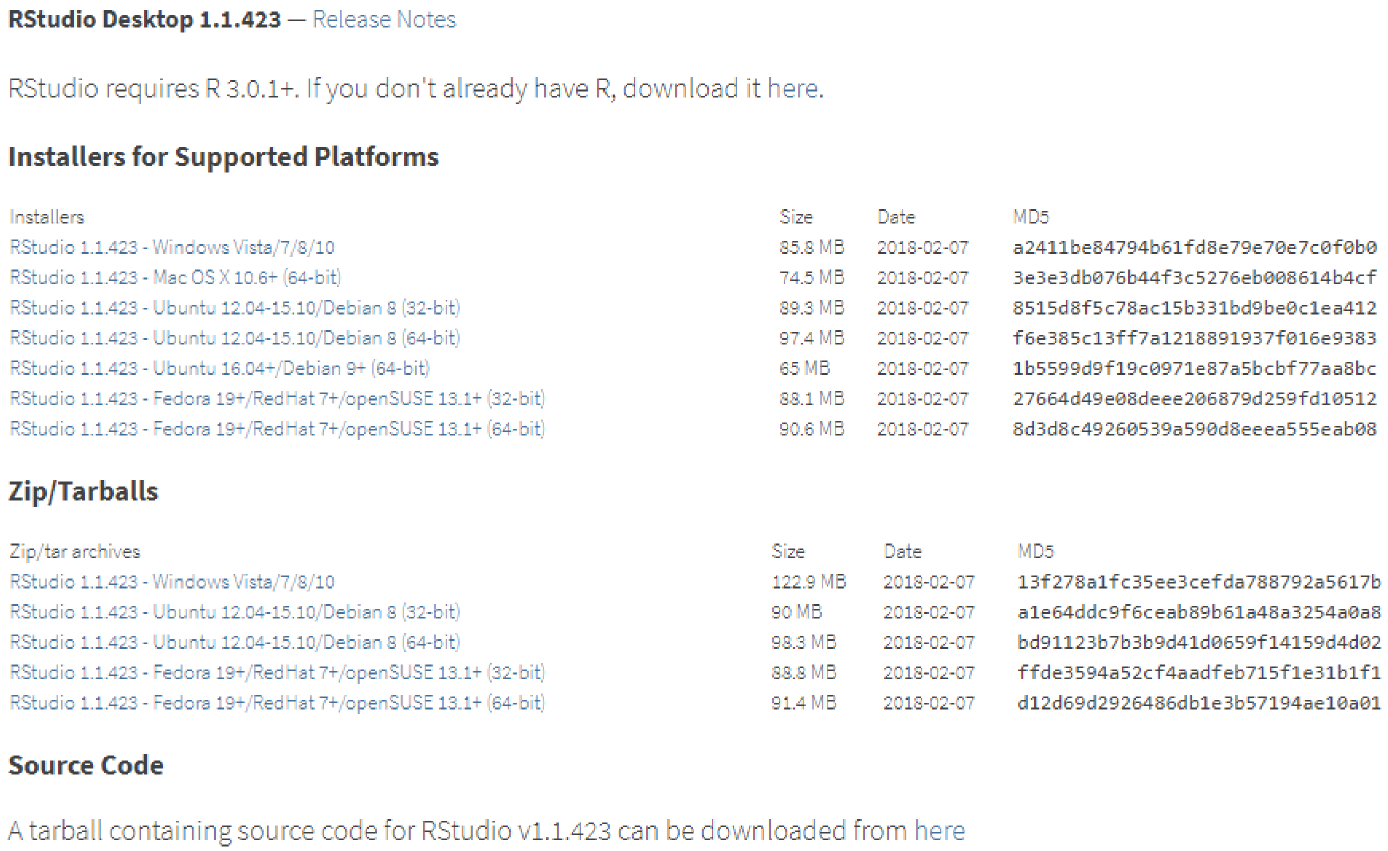
For Mac/Linux users, head to RStudio and run the following code to download the two primary development packages you’ll need:

```
install.packages('devtools')
install.packages('roxygen2')
```

## Setting up RStudio

To download and set up RStudio, follow the instructions below:

1. Head to https://www.rstudio.com/products/rstudio/download/ (https://www.rstudio.com/products/rstudio/download/) and click “download” for the free RStudio Desktop version
2. Download the *installer* for your operating system, not the *zip/tarball.* The picture below illustrates what the webpage should look like.



**RStudio Desktop 1.1.423** — Release Notes

RStudio requires R 3.0.1+. If you don't already have R, download it here.

**Installers for Supported Platforms**

| Installers | Size | Date | MD5 |
|---|---|---|---|
| RStudio 1.1.423 - Windows Vista/7/8/10 | 85.8 MB | 2018-02-07 | a2411be84794b61fd8e79e70e7c0f0b0 |
| RStudio 1.1.423 - Mac OS X 10.6+ (64-bit) | 74.5 MB | 2018-02-07 | 3e3e3db076b44f3c5276eb008614b4cf |
| RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (32-bit) | 89.3 MB | 2018-02-07 | 8515d8f5c78ac15b331bd9be0c1ea412 |
| RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (64-bit) | 97.4 MB | 2018-02-07 | f6e385c13ff7a1218891937f016e9383 |
| RStudio 1.1.423 - Ubuntu 16.04+/Debian 9+ (64-bit) | 65 MB | 2018-02-07 | 1b5599d9f19c0971e87a5bcbf77aa8bc |
| RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit) | 88.1 MB | 2018-02-07 | 27664d49e08deee206879d259fd10512 |
| RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit) | 90.6 MB | 2018-02-07 | 8d3d8c49260539a590d8eeea555eab08 |

**Zip/Tarballs**

| Zip/tar archives | Size | Date | MD5 |
|---|---|---|---|
| RStudio 1.1.423 - Windows Vista/7/8/10 | 122.9 MB | 2018-02-07 | 13f278a1fc35ee3cefda788792a5617b |
| RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (32-bit) | 90 MB | 2018-02-07 | a1e64ddc9f6ceab89b61a48a3254a0a8 |
| RStudio 1.1.423 - Ubuntu 12.04-15.10/Debian 8 (64-bit) | 98.3 MB | 2018-02-07 | bd91123b7b3b9d41d0659f14159d4d02 |
| RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit) | 88.8 MB | 2018-02-07 | ffde3594a52cf4aadfeb715f1e31b1f1 |
| RStudio 1.1.423 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit) | 91.4 MB | 2018-02-07 | d12d69d2926486db1e3b57194ae10a01 |

**Source Code**

A tarball containing source code for RStudio v1.1.423 can be downloaded from here

*RStudio Webpage*

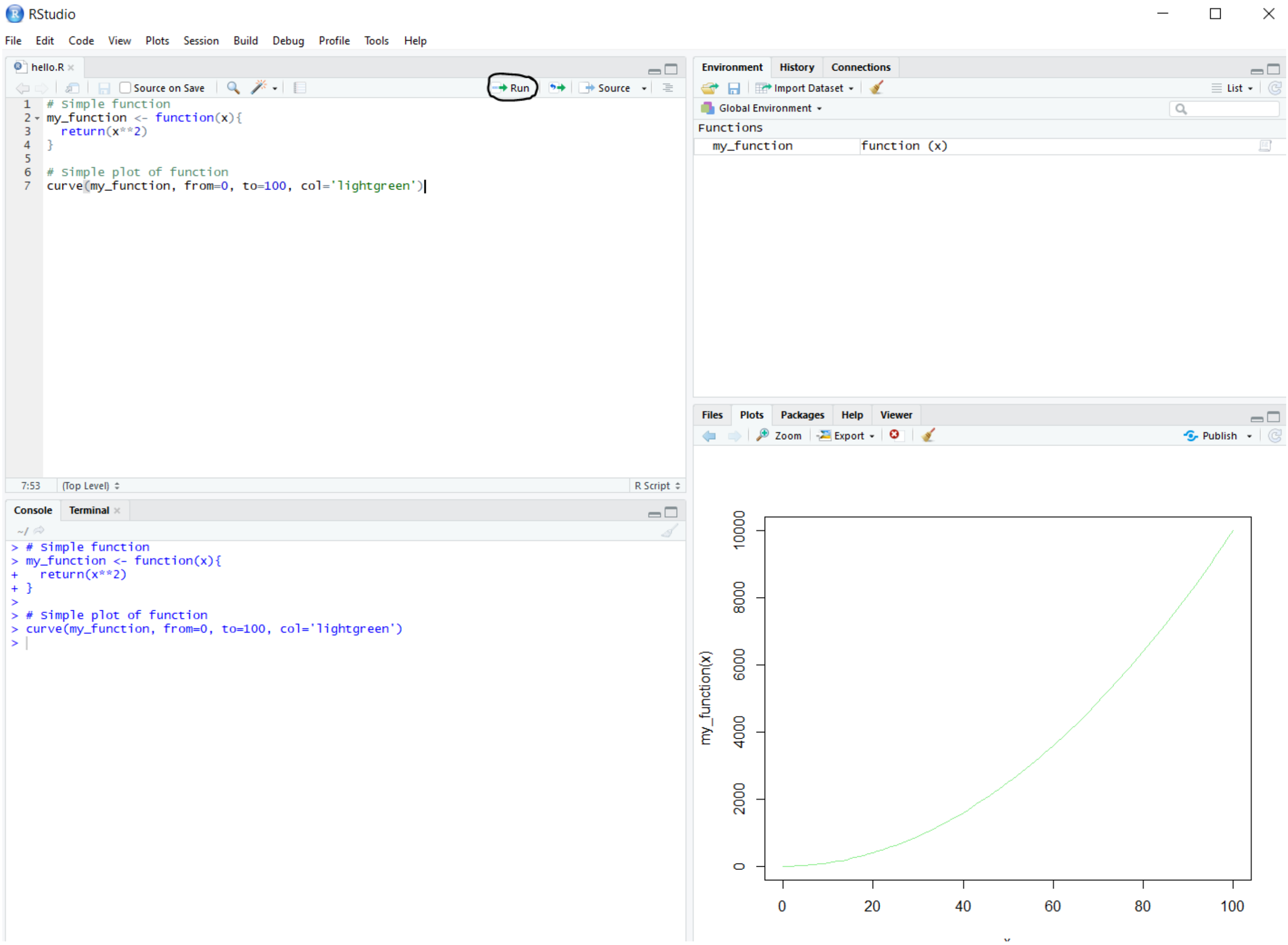
3. Let the installer run - it’s fine again to just use the default options.

and now you're ready!

# 3. Basic RStudio Use

## Screen Breakdown

RStudio will split your screen into four parts, as shown below. We'll refer to them as the *script*, the *environment*, the *shell* or *console*, and the *Viewer.*



```
# Simple function
my_function <- function(x){
  return(x**2)
}

# Simple plot of function
curve(my_function, from=0, to=100, col='lightgreen')
```

Functions

| | |
|---|---|
| my_function | function (x) |

```
~/
> # Simple function
> my_function <- function(x){
+   return(x**2)
+ }
>
> # Simple plot of function
> curve(my_function, from=0, to=100, col='lightgreen')
> |
```

The RStudio Interface

The *script* is in the upper left hand corner of the screen, and it's where you will do the bulk of your programming. To write an R script, click File>New File>R Script in the top left hand corner, and you can start programming! You can execute multiple lines of a script (or the entire thing) at a time by selecting the lines you want to run and clicking the green "Run" button on the right hand side of the script box.

The *shell* is in the lower left hand corner of the screen. Unlike scripts, which presumably you've worked with before, shells only run one line of code at a time. However, shells also remember what you did before. For example, if you define download some data into the shell, you can manipulate it repeatedly without having to download it again before each manipulation - it will stay in the shell's memory until you manually remove it.

All scripts that you run will be run through the shell - that is, if you run a script, all of the code from the script will appear in the shell. This means that you can run a script and manipulate the results directly from the bottom of the shell, without having to rerun the script each time you want to manipulate the results. You can also directly access the command prompt for your computer through the shell by clicking the “terminal” option.

Note that any errors caused by bugs in your code will show up in the shell.

On the upper right hand corner of the screen, you can see the *environment* you’re working in, which lists all the objects in the shell’s “memory” or *namespace.* For example, if you download some data in a script and then run the script, the data will show up in the right hand corner of the screen.

On the bottom right hand side is the *Viewer.* Any plots you generate in R will automatically show up there. Additionally, as we’ll discuss in a secnd, documentation of functions and packages will appear in the right hand corner. You can also use it to see the file structure of your working directory and the packages you’re using by clicking “files” or “packages.”"

# Basic Commands

Presumably you’re already familiar with R, but you should know a couple of commands specific to RStudio:

1. To remove an object from the environment, use the “rm” function as follows:

```
rm(remove_this_object)
```

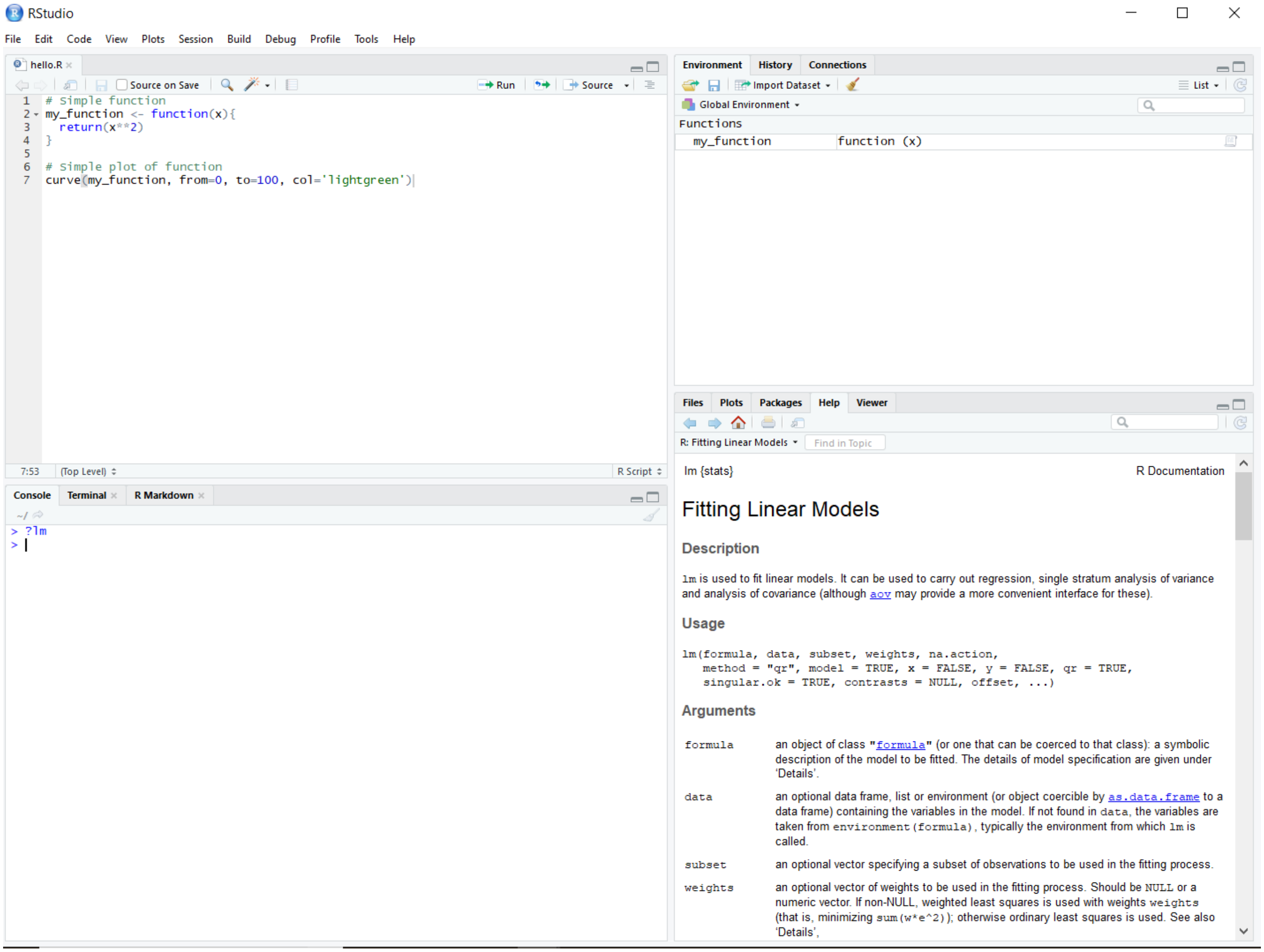
To remove all objects from the environment, you can use

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

2. To see the documentation of any specific function, you can type

```
?function_of_interest
```

In the example below, you’ll find a picture of the documentation for the lm function, which is used for linear modeling.



```
# Simple function
my_function <- function(x){
  return(x**2)
}

# Simple plot of function
curve(my_function, from=0, to=100, col='lightgreen')
```

```
> ?lm
> 
```

Functions

| | |
|---|---|
| my_function | function (x) |

lm {stats} R Documentation

Fitting Linear Models

Description

`lm` is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although `aov` may provide a more convenient interface for these).

Usage

```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

Arguments

| | |
|---|---|
| `formula` | an object of class "`formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'. |
| `data` | an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `lm` is called. |
| `subset` | an optional vector specifying a subset of observations to be used in the fitting process. |
| `weights` | an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If non-NULL, weighted least squares is used with weights `weights` (that is, minimizing `sum(w*e^2)`); otherwise ordinary least squares is used. See also 'Details', |

How to quickly access documentation

3. To get RStudio to stop executing some code, you can press the red “stop” button in the top right of the console section. Sometimes this won’t appear, so you can also just hit “escape” instead.

4. To clear the console, hit Ctrl+L.

5. To install packages, you can use the “install.packages(‘packagename’)” command. For example, to install the mosaic package, one could run the following code:

```
install.packages('mosaic')
```

A list of usefull packages can be found here (https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages).

# 4. Further Reading

To read more about using R, take a look at the following website, built by Chapman and Hall (http://adv-r.had.co.nz/)

This link (https://www.rstudio.com/resources/cheatsheets/) contains a number of truly fantastic cheat sheets, documenting everything from RStudio itself to data visualization and machine learning.

# 5. Sources cited

https://www.theanalysisfactor.com/the-advantages-of-rstudio/ (https://www.theanalysisfactor.com/the-advantages-of-rstudio/)