# Installing R, R packages and RStudio

Instructions and Resources for new users and students

Bruce Dudek

2020-08-06

## Contents

# 1 Introduction

This document provides information for new R users on installation, updating and maintining an R installation. It can be broadly useful to a wide audience, but is intended for students in B. Dudek's statistics classes, and serves as the first of several R tutorial documents. Topics covered include installation/updates/maintenance of R, RStudio, and R packages. Some general background is provided along with instructions and links to detailed web postings that go into more detail if needed.

# 2 What is R? What is RStudio?

The R programming language (https://www.r-project.org/) has major strengths in data management, statistical analysis and data visualization. It has fast become a primary tool for data scientists, statisticians and researchers. It can be used on multiple operating systems and its installation is largely straight forward on all platforms. As an open source and free software ecosystem it provides a rich array of tools for a diverse audience of users.

This link to a page on the R project web site gives a more detailed overview: https://www.r-project.org/about.html

Downloads of R and more information are found on CRAN (Comprehenisive R Archive Network): https://cran.r-project.org/

R is installed as a relatively minimal configuration where the user passes R code to a command line environment called the R console (inside the R GUI). A majority of R users employ additional tool, RStudio (https://rstudio.com/) that is an Integrated Development Environment. Embedded in the RStudio configuration is the R console, file management and code writing capability, and several other useful components for displaying figures, managing add-on packages, etc. It is a very powerful way to use R, especially with its capabilities to use markdown, to build packages, and many other add-in capabilities. It is strongly recommended that new users become familiar with using RStudio very early in the R learning curve.

# 3 Why R?

The question of why R? is often posed in the context of comparing it to other programming languages that can handle data science needs, or in comparison to commercially available software that have long and established histories (e.g., SAS, STATA, SPSS). There is no need to dwell on these comparisons. The major utility of R is its rich array of add-on contributions in the form of packages created by statisticians around the world. New methods can appear very quickly in the R eco-system of add-on packages. There is also an extensive support community online in the form of forums, blogs, and help sites. For the scientific researcher in many disciplines, skills in R are becoming an expected part of training in degree programs. It is an important tool to add to an arsenal of data analytic capabilities.

These URL's expand on the points made above:

https://www.burns-stat.com/documents/tutorials/why-use-the-r-language/

http://www.econometricsbysimulation.com/2014/03/why-use-r-five-reasons.html

# 4 Why not just spreadsheets?

Spreadsheets such as Excel have major positive features and are often very helpful for data entry and initial data management. But R can do these things well too, and there is a reproducibility element with R that is important, since code can be saved, repeated, and documented. At its core, R is a complete data management and analytic system, with far more capabilities than spreadsheets. See the following commentary for additional perspectives:

https://www.burns-stat.com/documents/tutorials/spreadsheet-addiction/

# 5 What are "packages"?

The R language is very broadly capable, even in its simple original installation, but additional capabilities are found in packages.

The core R installation already has a set of add-on packages with capabilities suggested by their names. These are found in a subfolder of the R program installation called src/library/:

- base
- compiler
- datasets
- graphics
- grDevices
- grid
- methods
- parallel
- splines
- stats
- stats4
- tcltk
- tools
- translations
- utils

The R Project also "recommends" a set of additional packages that the user will have to install separately (see a later section in this document):

https://cran.r-project.org/src/contrib/4.0.2/Recommended/

- KernSmooth
- MASS
- Matrix
- boot
- class
- cluster
- codetools
- foreign
- lattice
- mgcv
- nlme
- nnet
- rpart
- spatial
- survival

These are all useful but most users will quickly compile a longer list of additional packages that they use for specific purposes. For example the **psych** package contains a large suite of functions useful to researchers in

the psychological sciences. Methods for installing these packages and a recommended set are described in a later section.

# 6 Installing R

For purposes of B. Dudek's classes, it is useful to install the most recent version of R so that we are all using the same version. As of this writing that version is 4.01, but it will likely be a later version by the time you read this. Also as of version 4.0, all previous packages you may have used on an earlier version need to be reinstalled (see the section below).

## 6.1 Major installation methods

On each of the major platforms, the base R installation occurs with a minimum of effort. The steps are:

1. Go to the R-project.org site, https://www.r-project.org/, and in the download section navigate with the CRAN link. This takes you to a page with a set of CRAN mirrors. Choose one.
2. Choose your operating system and then on the next page choose "base" which leads to a download.
3. Complete the download to your device and run the installer that was downloaded. Choose the default options at each point.

I used to provide a detailed document with screen captures showing each step but it is so simple that it is no longer necessary to generate that level of detail. If you want more detail, go to one of these sites that do provide additional guidance:

https://www.andrewheiss.com/blog/2012/04/17/install-r-rstudio-r-commander-windows-osx/

https://techvidvan.com/tutorials/install-r/

https://rstudio-education.github.io/hopr/starting.html

https://www.datacamp.com/community/tutorials/installing-R-windows-mac-ubuntu

https://www.dataquest.io/blog/tutorial-getting-started-with-r-and-rstudio/

## 6.2 64 bit vs 32 bit installations

On Windows platforms, both 32 bit and 64 bit R is installed by the procedures outlined above if you are using an x64 version of Windows (which by now, most all users are). The 32 bit version was created originally because older PC OS installations were only 32 bit - and there is a memory limitation that is now much higher in the 64 bit version. The 64 bit version is the one you will want to use.

On MAC OS and LINUX, the base R installation is 64 bit.

# 7 Installing RStudio

After you have installed R, then install RStudio. Go to the RStudio web site https://rstudio.com/ and choose the download button (or go directly to the download page (https://rstudio.com/products/rstudio/download/). Then download the free RStudio Desktop version for your operating system. Execute the installer that you downloaded.

# 8 Testing your R and R studio installations

In order to begin learning to use R and RStudio and test the installations, I suggest the following.

1. Open the 64 bit version of R that you installed (not RStudio at this point). It should look like this in Microsoft Windows and something similar in MAC OS:
2. At the command prompt in R Console, type the following to obtain the square root of 25:

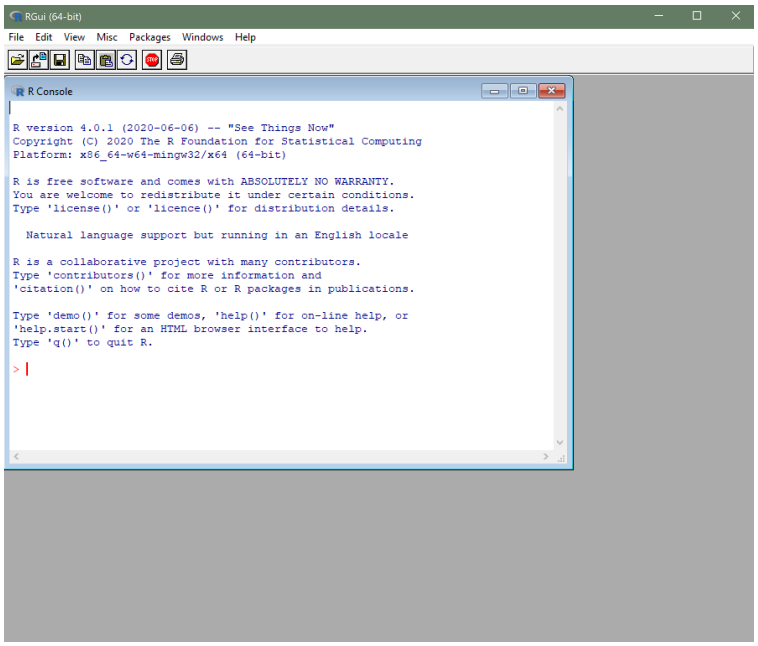



Figure 1: R Console inside the R GUI

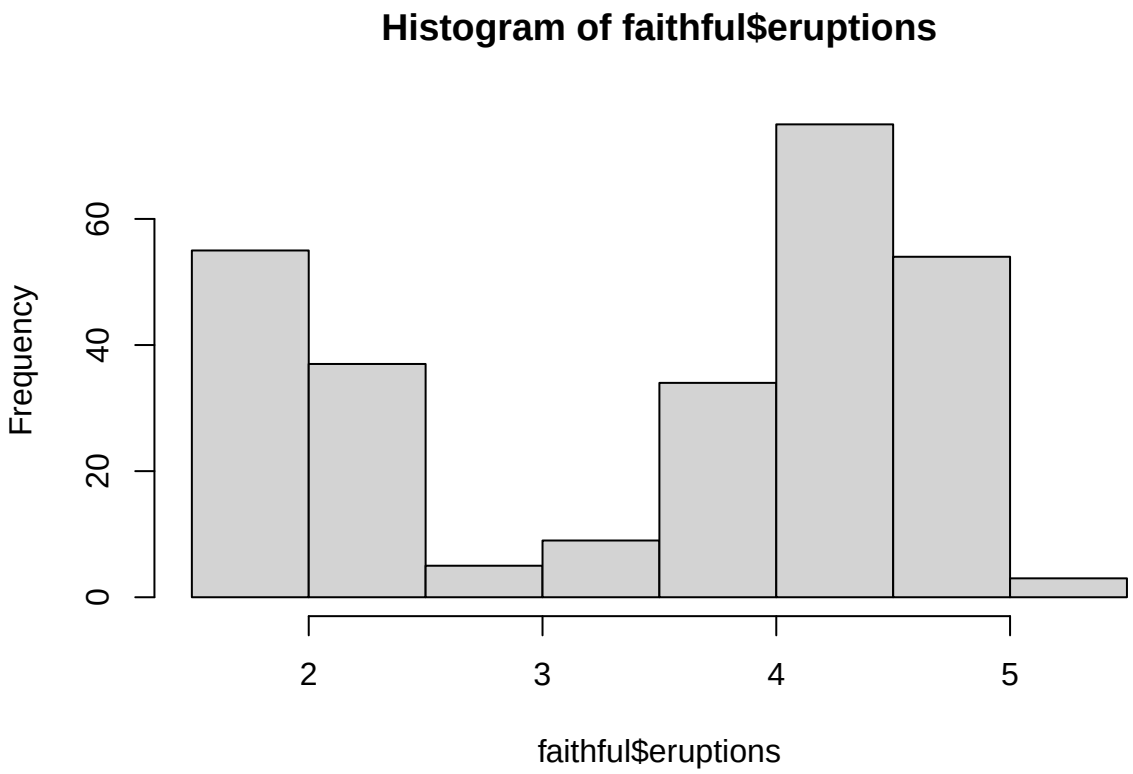
```
25^.5
```

```
## [1] 5
```

R has returned an "object" that contains one element and that first element is the numeric value of the answer.

3. Next type:

```
hist(faithful$eruptions)
```




This function has drawn a frequency histogram of the Old Faithful Geyser eruption durations using a data set that comes with R (called faithul).

4. Now close R and open RStudio. It should look something like this in Windows and something similar in MAC OS:

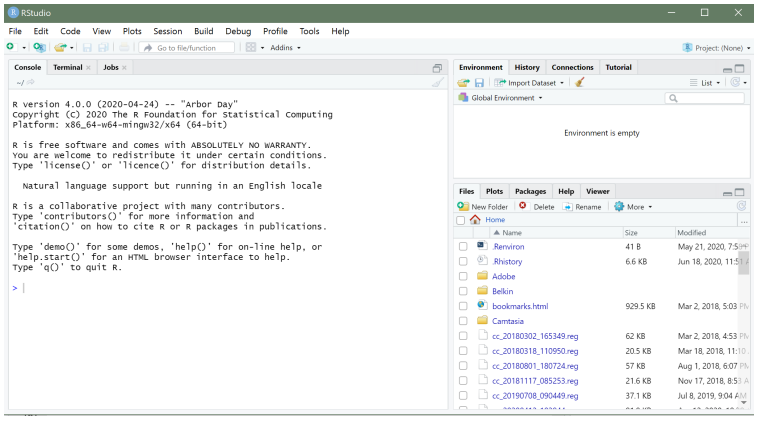



Figure 2: R Console inside the R GUI

Notice that the same R Console and command prompt appear in the left pane (I think that is still the default).

5. Type the same to lines of r code that you did above to see that it works the same way in RStudio. However, the histogram will appear in the viewer pane bottom right.

With these quick tests, you know that you have a functioning installation and are ready to add other packages and begin learning/using R.

# 9 Adding packages to your R installation

At the time of this writing, CRAN currently hosts 15806 add-on packages. Obviously there is no need to install all of them (or time!). Typically one adds packages when a need arises or you learn about a new package that is worth exploring.

One of the great things about R (and sometimes an infuriating thing) is that there are multiple ways of doing things. This is true with package installations as well. Add-on packages can be installed from the RGui menus, or from code submitted to the command prompt, and with the packages tab in RStudio. I still prefer generating the code to install packages so that I can keep a record of what I did.

The primary function to install packages is `install.packages()`. You can look up the help page for this function by typing the following at the command prompt:

```
?install.packages
```

If you are working in the RGui/Rconsole, this help page will pop up as a second window. If you are in RStudio, the help page appears in the help tab in the bottom right pane.

This help page will probably be overly challenging for the user who is new to R, so lets go slowly.

## 9.1 Installing packages from the command line

This process can be done either from within the R GUI/R Console, or from the console withing RStudio. Initially, I suggest that you do this first test from the basic R installation (RGui/RConsole). Let's begin by installing a single package from the command line, using the `install.packages()` function. This line will install the `psych` package. Each add-on package may have dependencies on other packages in order to function. Those dependencies are respected by `install.packages()` and will also be installed. If you look at the help page for `install.packages()` you will see that a "dependencies" argument is set so that dependencies are also installed by default.

```
install.packages("psych")
```

If you do this in RGui/RConsole, a dialog box will appear asking your to choose a CRAN mirror. I typically choose one in Michigan, assuming that closer ones might be quicker. If you do this in RStudio, a default mirror is already set. You can change that default in the global options section of RStudio.

## 9.2 Install more than one package at a time

It is possible to install multiple packages with a single `install.packages()` command. Let's assume that we want to install `yhat` and `rcompanion`. This code uses the `c` function called concatenate, which creates a string of two package names passed to the `install.packages()` function.

```
install.packages(c("yhat","rcompanion"))
```

## 9.3 Install packages from the menus in RGui/Rconsole.

In RGui, choose the "pacakges" tab and select the "install package(s)". You can choose from a list of all packages on CRAN. This is a menu/mouse way of executing the `install.packages()` function.

## 9.4 Install packages from the "Packages" tab in the RStudio pane (usually bottom right)

By choosing that "packages" tab, you have forced RStudio to show all packages that you have installed. But to install new packages, choose the "install" tab there. The dialog box permits you to install one or more packages by typing in their name in the available box. By default it goes to CRAN to look for those packages, but you may have a local file that contains a package and the option there is to specify the local location of that package file.

## 9.5 Installing packages from GitHub

Many package developers use GitHub repositories for their packages. Sometimes you may want/need to install a development version of a package that is not yet on CRAN, or you may have identified a useful package that the developer hasn't submitted to CRAN. That is the case for a package used by B. Dudek in his courses, a package called **bcdstats**.

In order to make this work, it is necessary to install another package first, **devtools**. A function in **devtools** called `install_github()` is used.

```
install.packages("devtools")
```

In order to install a github package you need to know the location which would be found in the developer's GitHub page. Information on the package page on GitHub probably repeats this sequence of commands. First load the **devtools** package and then execute the install function.

```
library(devtools)
install_github('bcdudek/bcdstats')
```

## 9.6 Install many packages for use in B. Dudek's course

For requirements in B. Dudek's courses an additional way of installing packages is available. A long list of packages are recommended for these courses (between one and two hundred). A separate file is provided, called something like "bcd_recommend.R" or "bcd_recommend2.R". That file is a text file of R code. The R code is a long series of "install.packages()" commands, each installing an individual package and its dependencies. In order to use this file, we can employ the `source` function in R. At the command prompt, using the source function (and the file name) asks R to read the R code in that text file. It will then execute all of the `install.packages()` lines in that file. In order to employ this approach, you have to know where the "bcd_recommend.R" file resides and navigate there. The `source` function is set up here to use another function called `file.choose()` which creates a dialog box where you can navigate to the correct folder and identify the "bcd_recommend.R" file to be passed to `source`. Once you do this, the process will take a while to install all of the packages. Some may not install and you can see this at the end by using the `warnings()` function as indicated by R if some package installations failed. You should copy/paste this list of uninstalled packages from the `warnings()` output, but don't worry about them at this point. Just share the list with B. Dudek.

```
# pass the "bcd_recommend.R" file to the source function.
source(file.choose())
```

## 9.7 Installing the tidyverse

A suite of packages motivated by the RStudio folks and Hadley Wickham, in particular, have gained much favor recently in the R user world. They provide a nice set of tools for data handling, data visualization and many other things. For example the **ggplot2** package has become a heavily used approach to graphing. The current number of tidyverse package is over 20 and you can read about them here: https://www.tidyverse.org/

The tidyverse packages can be installed simultaneously with this code. But note that if you used the "bcd_recommend.R" script above, one of the first things it did was to install the tidyverse suite of packages.

```
install.packages("tidyverse")
```

## 9.8 Source vs binary package installation methods

The syntax of the install.packages() function works on all OS platforms. And despite some superficial differences in appearance of the RGui/RConsole on Linux/MAC/Windows, the R program largely runs the same way. But some differences with regard to package installation are less apparent.

If you visit the home page of the package (e.g., **psych** at CRAN you will find that there are actually three different formats of the package

```
psych_1.9.12.31.tar.gz , the package source code
psych_1.9.12.31.zip, the Windows binary containing the package
psych_1.9.12.31.gz .tgz, the Mac binary containing the package
```

What actually happens when you run 'install.packages("psych") is that, on Mac or Windows, R downloads these compressed pre-compiled binaries and just unpacks them and places them in your R library folder. This speeds the process. On Linux, packages have to be compiled from source (they are not precompiled, so the .tar.gz file is used).

At times, you may want to install from source. R may give you this suggestion for some packages as indicated above, when the source version is newer than the compiled version. Other packages are not pre-compiled for the current version of R (especially versions later than R4.0), and you may still want to install them. Or some packages may not be currently maintained and are archived on CRAN - they may still be useful.

Installing packages from source is direct on Linux. On Mac OS it is also typically seamless. But on Windows, the normal R installation does not have the utilities for compilation that are built in to the MAC and LINUX OS's. This requires installation of the RTOOLS suite of tools (see section below).

## 9.9 The Rcmdr package

One of the R packages often used by inexperienced R users is the **Rcmdr** package (R Commander) written by John Fox. It is a bit different than the typical R package. It provides a menu driven interface that is quite capable in a wide range of descriptive statistics, EDA, and many basic inferential approaches. Often, new R users employ R Commander in the early stages of the learning curve because they can rapidly accomplish analyses without knowing detailed R code. The biggest advantage of R Commander, in this context, is that for every analysis, it converts the menuing operation to R code and provides that code. It is a great way to learn the style of R function usage.

R Commander would have been installed in the suite of packages that were installed from the "bcd_recommend.R" script as described above. If successfully installed, the user can call it by loading the package from the library. Mac OS users may need to do two extra things first, found in the following section.

```
library(Rcmdr)
```

### 9.9.1 Configuring R Commander capability on MAC OS

Mac OS X users may need to do two things to enable a functional **Rcmdr** install:

1. Download Tcl/Tk from <http://cran.r-project.org/bin/macosx/tools/ >http://cran.r-project.org/bin/macosx/tools/ (click on tcltk-8.x.x-x11.dmg) Install this tool. Note that in R versions 3.0 and above, the base R installation comes with Tcl/Tk, so this step should not be needed. But be aware of this if **Rcmdr** does not work and you get Tcl/Tk-related error messages.
2. In the applications folder there should be a sub-folder named Utilities. Make sure that you have a program named "X11" there. If not, then you wil need to download it from http://xquartz.macosforge.org/ and install it - make sure it is the latest version.

Special instructions for functionalizing **Rcmdr** on Mac OS can be found in John Fox' installation notes page, explaining this second point:

https://socialsciences.mcmaster.ca/jfox/Misc/Rcmdr/installation-notes.html

### 9.9.2 Some features and quirks of R Commander usage

Many plug-ins have been developed to extend the basic R Commander interface to enable more kinds of analyses. See https://www.rcommander.com/

R Commander will work from within RStudio. However I have found that it infrequently causes the R program to hang in a Windows OS installation. I have not been able to discern what the cause might be. However, it is known that R Commander works better in the version of the RGui that is called the "single document interface". By default, when you installed R it did not install itself in the SDI, rather it installed a "multiple document interface", which is normally preferred. In order to run R in SDI mode it is possible to create a modified shortcut that will open R in SDI. A copy of this can be provided by B. Dudek, or the user can examine the instructions from John Fox.

See the bottom of page 1 in the "Getting Started with R Commander" vignette/document on this CRAN page for information on this SDI issue:

https://cran.r-project.org/web/packages/Rcmdr/index.html

This SDI approach is also described in another web page on R Commander Installation that was referred to above for configuring Mac OS:

https://socialsciences.mcmaster.ca/jfox/Misc/Rcmdr/installation-notes.html

# 10 RTools for Windows

The RTools installation for Windows is a toolkit (called a toolchain) that provides added capabilities that normally are in the OS for Linux and Mac OS. As described above, one use of RTools is the provision of capability to install packages from source. But RTools has many other features and it is important for users who want to build their own package.

RTools can be downloaded from the following site, which also provides installation instructions. It is important to install a version that matches your installed version of R, and to upgrade RTools when R is upgraded to a newer version.

https://cran.r-project.org/bin/windows/Rtools/

Also see the history page for RTools for more guidance:

https://cran.r-project.org/bin/windows/Rtools/history.html

# 11 The bcdstats package and Installing packages from a locally saved file

Students in B. Dudek's courses will find the **bcdstats** package useful. It provides a few graphing and analytical functions on correlation/regression and it includes the shiny apps used in the course. The section above describes how to install it from GitHub. However, B. Dudek may provide binary files that can be installed. When packages are installed from a downloaded binary file the method requires use of the `install.packages()` function with a few additional arguments than those described above in the initial tutorial.

To install locally saved (downloaded) binaries (.zip or .gz), use the following syntax:

```
install.packages(file.choose(), repos=NULL)
```

To install packages from a locally saved source file (.tar.gz), use the following:

```
install.packages(file.choose(), repos=NULL, type="source")
```

# 12 The Bioconductor repository

The Bioconductor resource provides a great many tools for use in bioinformatics. Many of its package have broader utility and can be used by research scientists in other disciplines and by statisticians and data scientists. The Bioconductor site is well documented and can be found at:

https://www.bioconductor.org/

Bioconductor packages are not available on CRAN and usually have dependencies on two other Bioconductor packages: **biobase biogenerics**

An example of a broadly usefuly bioconductor package is the **multtest** package. It contains functions for resampling-based multiple comparisons.

Familiarizing oneself with Bioconductor is a useful exercise.

# 13 Updating R, RStudio and R Packages

Base R is updated several times a year. Keeping up with the latest version is useful, but some cautions are advised. RStudio updates frequently and it is useful to regularly update it. R packages are updated on unpredictable schedules. Accomplishing these updates requires a strategy. I suggest the following.

## 13.1 Updating/installing new R versions

When new R versions are released, updates can be accomplished simply by installing the new version. Older versions are retained unless uninstalled via the OS tools that you would normally use. When a new R version is installed, RStudio immediately switches to using that version. If you want to use an older version in RStudio, set the version in the global options.

This manual installation of updated R versions is simple enough to be the recommended approach. If you want to use a more automated approach that also manages updating packages then see the page for the **installr** package:

https://cran.r-project.org/web/packages/installr/index.html

## 13.2 Updating R packages.

One can update to the most recently released versions of packages in a variety of ways, each fairly simple to do - but it may take time if a lot of packages require updating.

### 13.2.1 Updating packages from within base R

There are menu-driven approaches both with RGui/RConsole and RStudio that enable updating all packages that have newer versions.

In RGui/RConsole, use the Packages tab and choose "Update Pacakges" from the menu list. After choosing a mirror site, a list of all packages that have newer versions than your installation is returned. You can select any or all of them. This process uses a function called `update.packages()` that could also be run from the command line. See help ('?update.packages()') for the function.

Doing this from the command line is also simple:

```
update.packages()
```

This code will result in a query for every possible updateable package about whether you want to update. To avoide the repetitive queries, add one argument to the function:

```
update.packages(ask=FALSE)
```

### 13.2.2 Updating packages from within RStudio

I find a simple way to update packages is to use RStudio. In the packages tab in the lower right viewer pane, choose "update". A list of updateable packages is shown and any/all of them can be updated.

### 13.2.3 Updating tidyverse packages

Some users will rely heavily on tidyverse packages and there is a way to check for updates and install them for only the tidyverse packages:

```
library(tidyverse)
tidyverse_update(recursive = FALSE, repos = getOption("repos"))
```

## 13.3 Updating RStudio

In the help tab of RStudio, one can check for upated version by choosing the menu item "check for updates". RStudio frequently updates to new versions and more and more capability is being built in to this IDE. It is worth checking this somewhat frequently.

# 14 Reproducibility

Given the changing versions of R and add-on packages, it is important to document versions used in any analysis. One can accomplish this at any point by executing a function in R that lists the versions of R, its packages in use in the project.

```
sessionInfo()
```

# 15 R Resources

**General**

- R Project for Statistical Computing The main R Project Web page
- CRAN mirrors Primary path to sites for download
- The R Journal
- R Blog Developer commentaries on new R methods
- RStudio RStudio IDE, Shiny, Hosted Services, etc
- Quick-R Robert Kabacoff's tremendously helpful Site - tutorials and templates

**Installation**

- Andrew Heiss' Site Install R, RStudio, and R Commander in Windows and OS X
- Tech Vidvan R Tutorials
- Hands On Programming with R Garrett Grolemund
- DataCamp R Installation tutorial
- Mixomics.org Installation Guide for R and RStudio

**Newbie help**

- Getting Help with R R Project Help Overview page - USEFUL
- CRAN Task Views
- Quick-R Tutorial Intro to R tutorial
- R for Beginners Emmanuel Paradis' introductory Document

- R Core Team Intro An Introduction to R
- DataCamp Intro to R tutorials
- R Commander
- Coding Club A positive peer-learning Community
- Random Stuff that Matters Starting with R: Things I wish I had been told

**Advanced help**

- RStudio Cheatsheets
- Advanced R Hadley Wickham's Important Book
- Contributed Documents R Project Site repository of documentation

**R community forums**

- StackOverflow Major R Q/A Site
- RStudio Community Forum Emphasis on RStudio and tidyverse with some general R support
- R Seek Fantastic search engine

**Blogs**

- Revolutions Blog
- R Blog List A list of R blogs contributing to R Bloggers

**Blog Aggregator**

- R Bloggers **A very important site for any R user**

**Data Visualization**

- ggplot2 web page
- R Graphics Cookbook Winston Chang
- DataVis with R Kabacoff's Data Visualization site
- The R Graph Gallery
- Towards Data Science Comprehensive Guide to Data Visualisation in R for Beginners
- r-statistics.co ggplot2 tutorial
- r-statistics.co Top 50 ggplot2 Visualizations
- Data Visualisation Chapter in Grolemund/Wickham book
- Compendium of Clean Graphs
- Visualizing Likert Items Jason Bryer

**Tutorials and Code Templates**

- R Companion Handbook Extensive topics
- Statistics with R Vincent Zoonekynd's compendium
- UCLA Statistics Data Analysis Examples
- Quick R
- The Personality Project Using R for Psychological Research

**Online courses**

- Coursera A listing of R programming courses
- DataCamp Free introductory course
- Udemy Free R basics course
- Class Central Listing of R Programming Courses

**Reproducibility and R Markdown**

- Open Science Foundataion
- Monash Bioinformatics Platform Reproducible Research in R
- R Markdown R Markdown home page
- R for Data Science Chapter on R Markdown
- Coding ClubGetting Started with R Markdown