Bite Size R

The Package Ecosystem

Keith Hurley

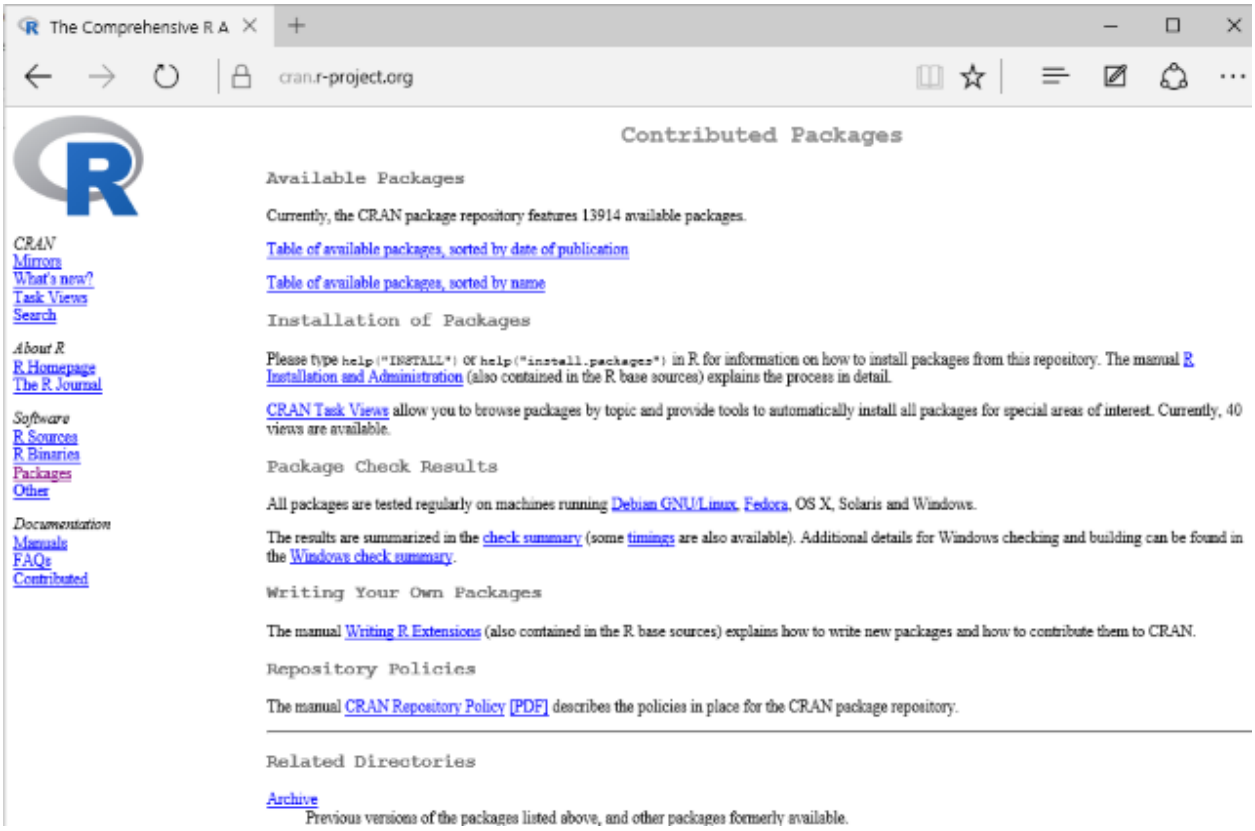
April 12, 2019

In today's snack we're going to take a look at the package ecosystem that's an integral part of using R.

The R software contains not only the R interpreter but also a large number of functions and capabilities that were part of R when it was first created. These functions are generally referred to now as "Base R".

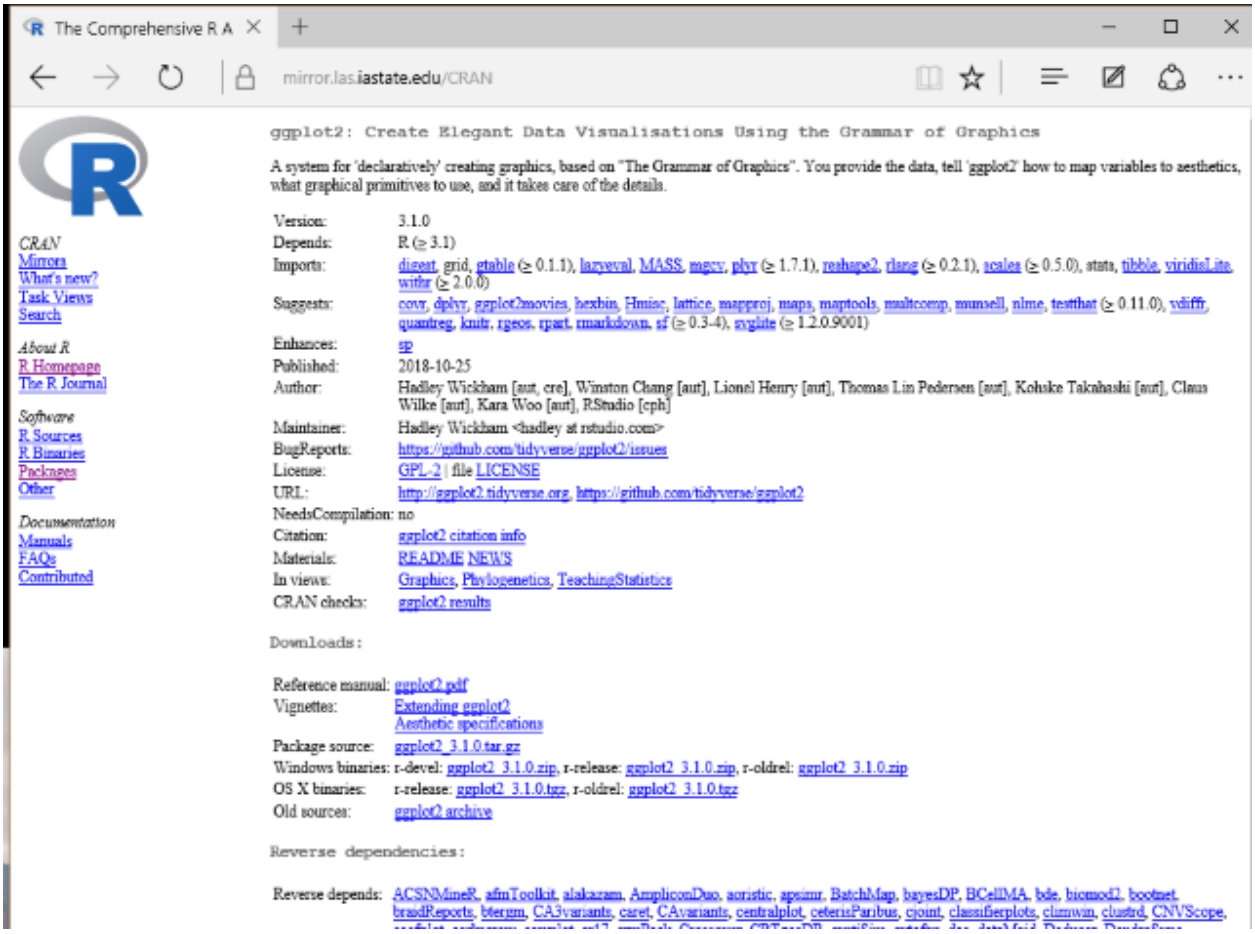
R Packages

Over time R has evolved and gained many new capabilities through it's package ecosystem. Packages are self-contained entities, usually revolving around a central topic or theme, that contain functions, code, datasets, and templates that expand the base capabilities of R. Not only that, but some packages will install other packages as well! CRAN (the Comprehensive R Archive Network) is the central repository for packages that have been submitted to the R-Project and meet some mandatory requirements and CRAN currently contains over 13,000 packages.



Many companies and organizations also have created their own packages specific to their domain which are distributed internally and NOT available publicly via CRAN. Many individuals, myself included, even maintain a personal package that encapsulates often-used pieces of code.

Part of any package accepted into CRAN is a standardized help file. A Google search on an R package name will usually lead you to the help file which contains some helpful information.



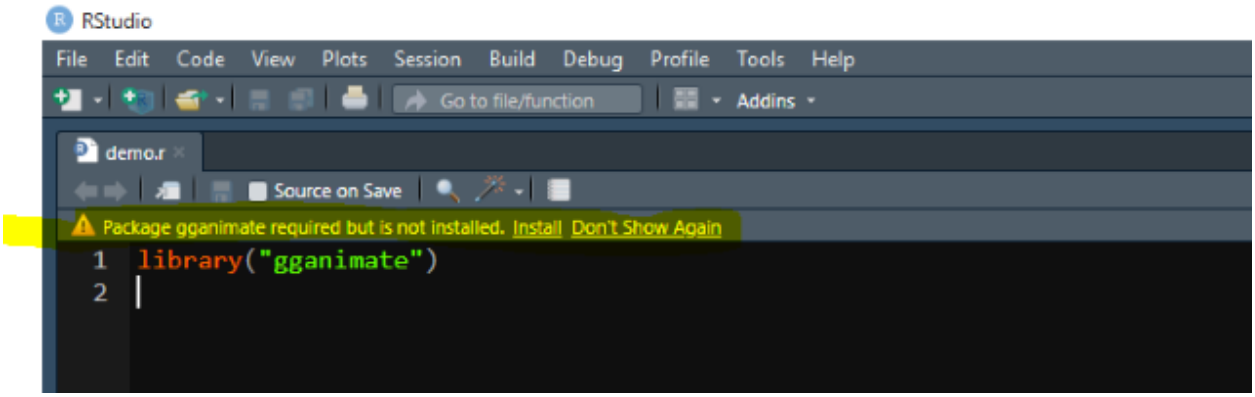
For example, the main package page provides the URL for the package (often times GitHub) which allows you to find in-development versions of the package, submit bug requests, or contact the author of the code.

Vignettes are tutorials to help you understand how to use the packages and a great way to get started with a new package.

The reference manual is the same standardized help file that can be viewed from within RStudio and provides details on every function, procedure, and dataset that the package offers.

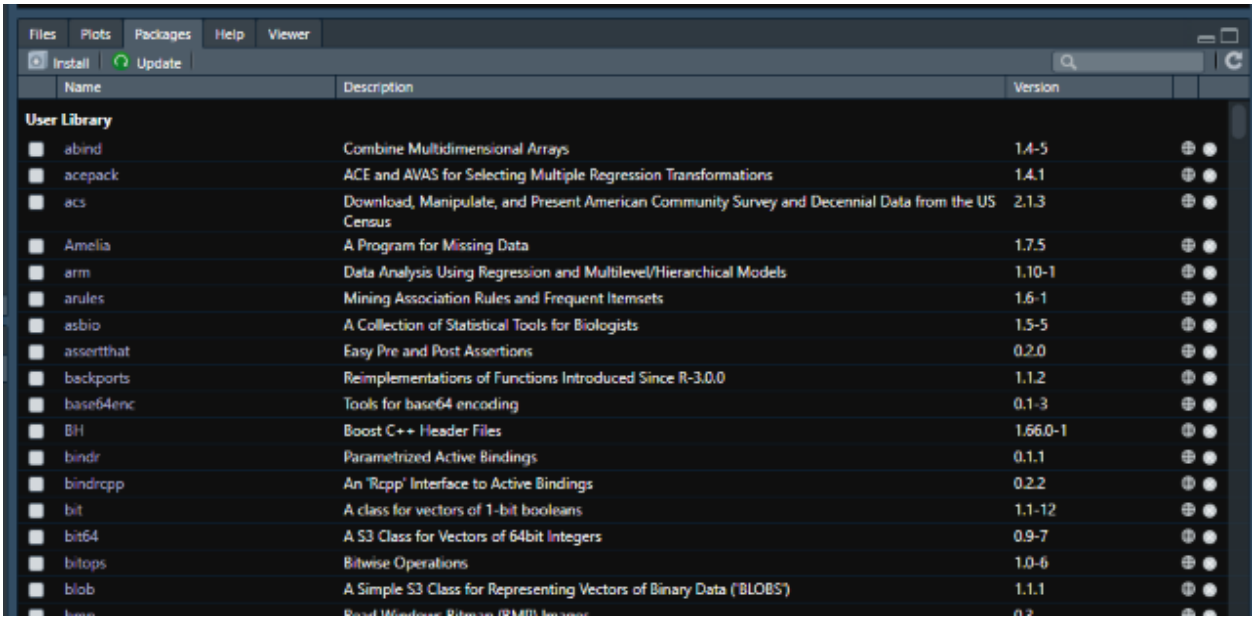
Installing R Packages

Before you can use R packages in your code, they must be installed on the computer that you are working on. This is a one-time function (barring updates and new versions) for each computer and I don't recommend including code to install packages in each of your scripts as it quickly becomes redundant and inefficient. In fact, the newest versions of RStudio even tell you if packages referenced by the code you have open are NOT loaded and gives you a convenient link to install them.

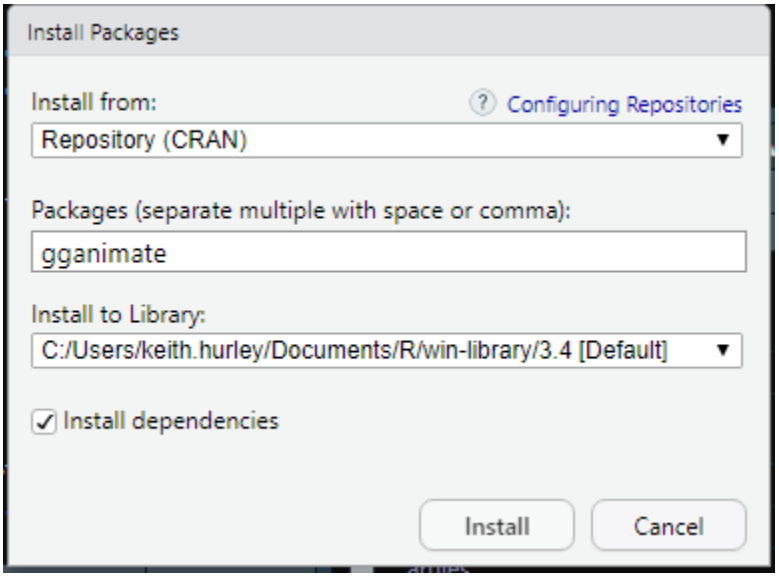


Install From The Package Tab

The package tab is normally located in the lower right corner of RStudio and provides a point and click method of installing packages. Packages that show up in this tab are already installed on your computer.

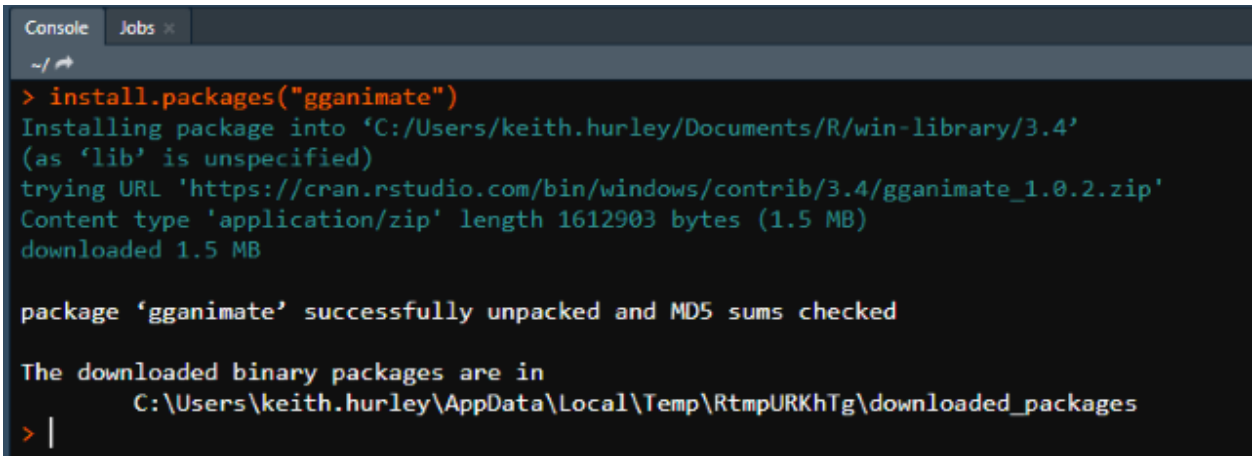


By clicking "Install", a pop-up window allows you to specify which package you want to install and then will make it happen!



Install From Code

It is also possible to install packages by writing it directly into r code - either in the console or a script file. When a package is loaded from the package tab pop-up window, you will notice that RStudio actually writes the code for you in the console.



To install an r package, the syntax is:

```
install.packages("gganimate")
```

Notice that the name of the package must be quoted so R knows that it's a name and not a variable that contains the name. It's also possible to load multiple packages at once by providing a vector of names:

```
install.packages(c("gganimate", "ggeffects", "ggedit"))
```

Loading a package

In order to use a package in your R code, you must load it into the memory of your R session. This session is reset every time you restart R. It's good practice to try to load only those packages that you'll be using rather than a laundry list of all of your favorites as they all take up memory that you may need for data! I

currently have over 400 packages installed on my desktop, but I usually will only use up to maybe a dozen at a time. For this reason, you DO need to load packages in each script or session of R.

The code for doing uses the library function like this:

```
library("gganimate")
```

Unlike packages, libraries can only be loaded one at a time. Also unlike packages, names do not need to be enclosed in quotes as the following will also work:

```
library(gganimate)
```

Wrap Up

That's it for this time. Now you know how to expand on R's capabilities using packages. If there is something that you want to do in R, it's likely a package exists that can make it happen! Happy snacking!