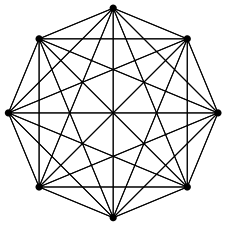
Chapter 28

Packages



The designers of R decided to organize functions and data sets into bundles which are known as *packages*. Even though it hasn't been apparent up to this point, every function and data set that has been introduced so far in the text belongs to a particular package. The previous 27 chapters have focused entirely on the data structures, functions, and data sets in R packages that are resident in a fresh R session.

This chapter will show you how to identify packages based on their capability, download packages onto your computer, and attach packages to your current R session. The topics that are introduced in this chapter are broken out by this classification of packages: (a) the big picture, (b) packages attached to my current R session, (c) packages installed on my computer, and (d) contributed packages. The first topic, the big picture, concerns *package management*, which includes how to move packages in and out of your current R session and how to download packages onto your computer. The second topic considers packages attached to a fresh R session, which has been the focus of the book so far. The third topic considers some other packages that are on your computer, but not in your current R session. The final topic considers *contributed packages* that can be downloaded onto your computer and then attached to your current R session.

28.1 The big picture

We begin with some definitions. A *package* is a collection of functions and data sets, along with associated documentation. The code in a package can be all R code, or can include code in other languages—such as C—oftentimes for faster execution times. A package is sometimes referred to as a *library* in other languages. Package names follow the same naming conventions as object names. These conventions were outlined at the beginning of Chapter 3.

A *contributed package* is a package written by a member of the R community that seeks to make a particular operation or application easier or to extend the functionality of R in a particular direction, for example, advanced statistical procedures or advanced graphical capability. All of the behind-the-scenes work on these contributed packages is done by the authors of the packages and a group of volunteers. New packages and updates of existing packages are coming online daily.

A *repository* is a central file storage location for contributed R packages. Although most contributed R packages reside at the repository known as CRAN (Comprehensive R Archive Network), they can also be located on other repositories such as Bioconductor, R-Forge, and Github. The number of contributed R packages on CRAN hit 10,000 on January 27, 2017.

A *mirror* is one location on a repository like CRAN, which consists of a worldwide network of servers. The mirrors are updated periodically with new packages that have been added to the CRAN servers. Choose a nearby mirror to download a contributed package onto your computer.

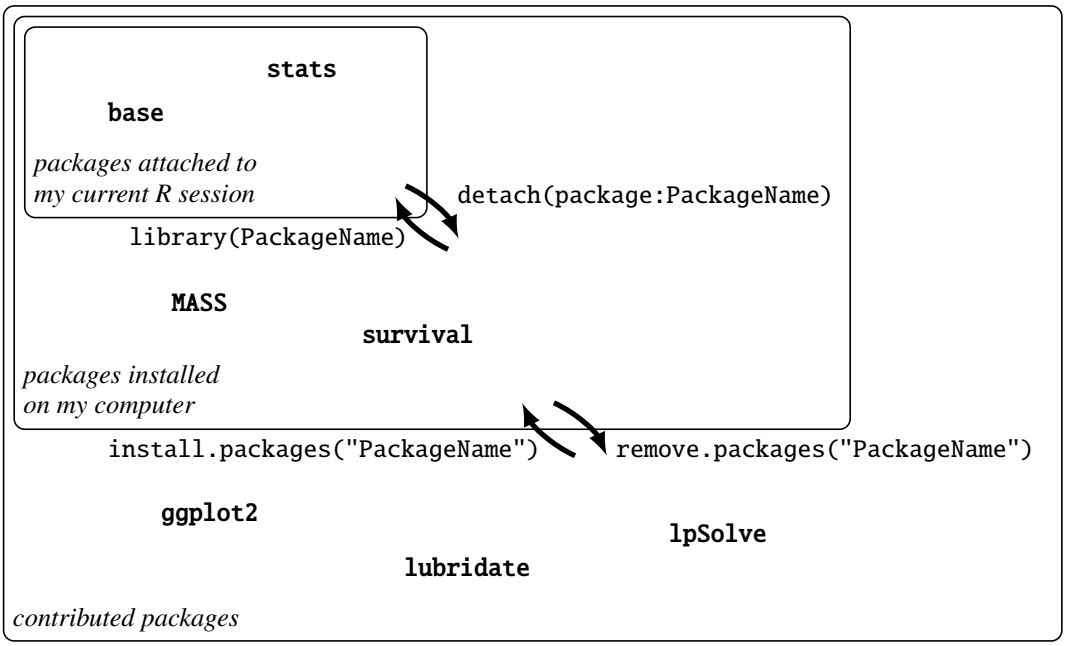
The diagram below depicts the various locations of packages and also serves as an outline of the packages that will be surveyed in this chapter. The three rectangles represent the location of packages, which will be described in the bulleted items below from the smallest to the largest rectangle.

- The smallest rectangle is labeled *packages attached to my current R session*. When you download R and enter a fresh R session, you have access to about a half dozen packages, which are known as *base packages*. Two of these packages, `base` and `stats`, are shown in the diagram.
- The medium-sized rectangle is labeled *packages installed on my computer*. The current distribution of R includes over a dozen additional packages that are resident on your computer, but have not yet been attached to your current R session. Two of these packages, `MASS` and `survival`, are shown in the diagram.
- The largest rectangle is labeled *contributed packages*. These are the thousands of packages available on the CRAN or another repository that can be downloaded onto your computer. Three of these packages, `ggplot2`, `lubridate`, and `lpSolve`, are shown in the diagram.

The arrows in the diagram represent the four key functions that are used in *package management*. These functions give you the ability to move packages between the three locations given in the diagram during an R session. The simplest syntax for a call to the `library` function is

```
library(PackageName)
```

which attaches a package named `PackageName` which is resident on your computer to your current R session. The `require` function has similar functionality. Once a package is in your R session, you can learn more about the package with the `help` function in the following fashion.



```
help(package = PackageName)
```

Help concerning a function or data set within a package is accessed with the `help` function in the usual fashion. The inverse of the `library` function is performed by the `detach` function with syntax

```
detach(package:PackageName)
```

which detaches a package named `PackageName` from your current R session.

The second pair of arrows in the diagram corresponds to downloading packages onto your computer and removing packages on your computer. The syntax for the `install.packages` function is

```
install.packages("PackageName")
```

The `install.packages` function will prompt you for a nearby mirror, and then download the package from that mirror on the CRAN onto your computer. If there are several packages to be downloaded, they can be placed in a vector of character strings containing the package names. The functions and data sets from the downloaded package(s) will not yet be accessible from your current R session; you will need to call the `library` function as described above to gain access to the package. Gaining access to the functions and data sets in a contributed package requires these two steps. Two problems can occur in the downloading of a package with the `install.packages` function. First, sometimes R needs other packages in order to use the contributed package. A function in one package, for example, might call a function from another package. These requirements are referred to as *dependencies*, and R will satisfy these dependencies as it downloads the package. Second, sometimes an object name used in one package is identical to an object name in an installed and attached package. In this case the original object name is said to be *masked*, and the syntax `PackageName::ObjectName` is required to access the masked object. The inverse of the `install.packages` function is performed by the `remove.packages` function with syntax

```
remove.packages("PackageName")
```

which removes a package named `PackageName` from your computer. Quitting an R session detaches all packages that were attached during a particular R session, but it leaves all installed packages resident on your computer.

There are a few other R functions that might be useful in package management. For example,

- `search()` displays all package names currently attached to your R session,
- `sessionInfo()` displays all package names currently attached to your R session and more,
- `installed.packages()` displays all installed packages on your computer,
- `old.packages()` displays all installed packages that are not in their current version,
- `update.packages()` updates all packages on your computer to their current version,
- `packageDescription("PackageName")` displays a brief description of `PackageName`, and
- `data(package = .packages(all.available = TRUE))` shows all data sets available in packages installed on your computer and `data()` shows all data sets attached to your current R session.