25 Recipes for
Getting Started with R

Excerpts from the R Cookbook

Paul Teetor
Learn how to turn data into decisions.

From startups to the Fortune 500, smart companies are betting on data-driven insight, seizing the opportunities that are emerging from the convergence of four powerful trends:

- New methods of collecting, managing, and analyzing data
- Cloud computing that offers inexpensive storage and flexible, on-demand computing power for massive data sets
- Visualization techniques that turn complex data into images that tell a compelling story
- Tools that make the power of data available to anyone

Get control over big data and turn it into insight with O'Reilly’s Strata offerings. Find the inspiration and information to create new products or revive existing ones, understand customer behavior, and get the data edge.

Visit oreilly.com/data to learn more.
25 Recipes for Getting Started with R

Paul Teetor
See Also

These are just the point estimates of the predictions. Use the `interval="prediction"` argument of `predict` to obtain the confidence intervals.

1.25 Accessing the Functions in a Package

Problem

A package installed on your computer is either a standard package or a package downloaded by you. When you try using functions in the package, however, R cannot find them.

Solution

Use either the `library` function or the `require` function to load the package into R:

```r
> library(packagename)
```

Discussion

R comes with several standard packages, but not all of them are automatically loaded when you start R. Likewise, you can download and install many useful packages from CRAN, but they are not automatically loaded when you run R. The `MASS` package comes standard with R, for example, but you could get this message when using the `lda` function in that package:

```r
> lda(x)
Error: could not find function "lda"
```

R is complaining that it cannot find the `lda` function among the packages currently loaded into memory.

When you use the `library` function or the `require` function, R loads the package into memory and its contents immediately become available to you:

```r
> library(MASS)
> lda(f ~ x + y) # Load the MASS library into memory
> lda(f ~ x + y) # Now R can find the function
```

Before calling `library`, R does not recognize the function name. Afterward, the package contents are available and calling the `lda` function works.
Notice that you needn’t enclose the package name in quotes.

The `require` function is nearly identical to `library`, but it has two features that are useful for writing scripts. It returns `TRUE` if the package was successfully loaded and `FALSE` otherwise. It also generates a mere warning if the load fails—unlike `library`, which generates an error.

Both functions have a key feature: they do not reload packages that are already loaded, so calling twice for the same package is harmless. This is especially nice for writing scripts. You can write a script to load needed packages while knowing that loaded packages will not be reloaded:

The `detach` function will unload a package that is currently loaded.

```r
> detach(package:MASS)
```

Observe that the package name must be qualified, as in `package:MASS`.

One reason to unload a package is if it contains a function whose name conflicts with a same-named function lower on the search list. When such a conflict occurs, we say the higher function `masks` the lower function. You no longer “see” the lower function because R stops searching when it finds the higher function. Hence unloading the higher package unmasks the lower name.

**See Also**

See the `search` function for more about the search path.