# Installing and Using R Packages

## What is a package?

A package is an installable library of functions. If there is something you want to accomplish in R, someone has probably created a package to do most of the heavy lifting already.

> *Example*
>
> You want to access info from a REST API over the web using HTTP calls from R, but you don't want to figure out how to connect to the internet, send a GET request with a payload, etc. You can use the `GET` function from the httr package or the `getURL` function from the RCurl package instead, and reduce the process to a few lines of code.

Packages come from a variety of sources including:

- The Comprehensive R Archive Network (CRAN)
- Version Control (Git, GitHub, Subversion)
- The internet (in broad terms)
- Your own computer

Many people write and maintain packages, from professional programmers to hobbyists, the point being that the work that goes into building a package is made up by decreased code complexity down the road. Like browser extensions or home improvements, you should choose the packages you install and use based on your programming needs, whether the package will be compatible with your project, and whether you trust the developer. Whenever possible, it is best to install packages from the most trustworthy possible source: Any packages on CRAN can be trusted. Open source packages are always open for scrutiny, so you should be able to determine how much you trust an open source package.

## Installing a package

Installing packages is generally simple. Packages hosted locally or on CRAN can be installed using the `install.packages` function. The help files can tell you more about how to use this function. This will also install any dependencies your requested package needs: If the developer uses functions from other packages in their packages (which is common) those other packages will also be installed.

*Example*

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

The devtools package allows installs from CRAN and locally, but it also has functions available for installing from other sources like version control repositories.

*Example*

```
> devtools::install_github("joe_exmaple/ex_package")
```

## Using a package

Before using the functions in a package, you have to designate where to find them. If you aren't concerned about duplicate function names in your namespace, you can run a line like:

```
library("devtools")
```

Then you can call the functions in devtools like `install_github` without having to define them. It is good R programming practice to indicate the package that a function comes from to avoid confusion later. This is accomplished using the `::` operator. In the case above, you would call `devtools::install_github`. The benefits of doing this are:

- You can easily track what dependencies your project has, lending to code portability.
- You can easily track which functions need to be replaced if you find a better package for your needs.
- You can easily see which code is yours to change and which is not.
- You can easily make attributions as necessary.
- You can still run your code if you forget to load the library.
- You can have functions with similar names from different places without getting confused.

## Resources

- The Comprehensive R Archive Network (CRAN)
  https://cran.r-project.org/
- GitHub
  https://github.com/
- Some Common Packages (all available on CRAN)
  - dplyr
  - devtools
  - foreign
  - cluster
  - Rcpp
  - ggplot2
  - stringr
  - plyr
  - digest
  - RCurl
  - httr
  - bitops
  - knitr
  - R6
  - data.table