Week 12 Module 1: Packages

CSCI E-5a: Programming in R

Module Overview

In this week's lecture, we will study the `ggplot2` graphics package.

For this module, however, we will address a simpler issue – what is this "package" thing?

We'll also see how to install and load packages.

The material in this module isn't spectacular in and of itself, but it's nonetheless very important, because it makes a huge range of additional tools and methods available to us.

Section 1: What's a Package?

Before we get into the mechanics of installing and loading packages, it's a good idea to understand what these are.

Although the base R that we've been using offers a tremendous selection of tools and techniques for solving problems, it can't possibly address everything.

There is a core set of procedures that everyone wants to use.

But then everyone has their own specialized projects, and they will want to use specialized methods for these.

It's a little bit like buying actual tools in a hardware store.

Everyone will want a good quality hammer, and a variety of screwdrivers, and a set of wrenches, and so on.

But for your particular project, there might be some specialized tool for which most people will have little use.

For instance, maybe you are working with a very small system, and you need a tiny screwdriver.

If someone else is working on a giant tractor, they would not require such a tiny screwdriver.

So, aside from the core set of tools like a hammer and screwdrivers and wrenches, most people will want to have some special tools.

It's the same with R.

For instance, if someone is working on complicated financial data, they might not be interested in tools for monitoring clinical trials, and vice versa.

The solution for this is to develop what are called *packages*.

A package is a collection of R objects that are bundled together and can be distributed.

The idea is that if you need some special functionality, then perhaps you can find a package and download it.

This will save you the considerable effort of having to implement it yourself.

In fact, you might not even have the technical skill to implement the specialized tool that you want.

This ability to extend the functionality of a programming system has been around for a long time.

However, these add-ons have often been proprietary offerings, with licences that could cost thousands of dollars.

What's remarkable about R is that packages are generally freely available.

This has lead to an explosion in the popularity of the language, because it's made many techniques available to the regular user.

Modern computing is very much about working with packages, and there is no way that you can be fully productive without leveraging these assets in your own practice.

Let me issue one word of caution about the use of packages.

There is absolutely no form of centralized quality control with packages, other than the creator's own commitment to doing it right.

This unlike a situation such as Apple, which carefully monitors all apps available for its products.

In general, I don't think you have to worry about some terrible virus in your downloaded package (although I can't guarantee that).

But it's different matter with the quality and correctness of the bundled code.

Do the functions always do the right thing?

Do they have some sort of numerical instability?

Do they work well on some inputs, but poorly on others?

There's no way to tell what the answers are to these questions.

For well-established packages, it's generally safe to assume that they have been rigorously vetted by the community of users.

But for other packages, it's just not clear.

I'm not aware of anyone who has experienced a clear-cut disaster when using some package.

But I've seen many people use them without the slightest suggestion that there might be the possibility that something could go wrong.

Just be careful, that's all.

Section 2: Installing and Loading Packages

In order to use a package, you first have to install it.

You might have already installed one such package.

At the beginning of the course, one of the options for running R was to install it on your own personal hard drive.

In this process, you had to install the `rmarkdown` package.

To do this, you use the `install.packages()` function.

There are many different ways to use this function, but the standard approach is to download the package over the Internet and install it.

For CSCI E-5a, I recommend installing the `tidyverse` package.

You can do this by running this code:

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

Notice that I've set the code chunk option to `eval = FALSE`.

That means that when RStudio knits this R notebook to a PDF file, it will *not* run the code chunk.

However, if you manually run the chunk, then this code will indeed execute, and R will install the package.

I'll discuss the tidyverse more next week, but for now I recommend that you just install it.

If you're running RStudio Cloud, the tidyverse should already be installed.

When you install a package like this, what happens is that R will go to the CRAN website and attempt to download the package to your hard drive.

This is just the basic idea, and it's possible to modify this approach, but this is the standard procedure.

Once the package has been downloaded, then R installs the package on your hard drive.

This can take a few seconds, or it can take 20 minutes; it depends on a variety of factors.

Even if the package is installed on your hard drive, that doesn't mean that you have access to the tools provided in it.

Instead, you have to explicitly load the package into your current running system by using the `library()` command.

I'm going to load in the `ggplot2()` package:

```
library( "ggplot2" )
```

And now we are ready to use the `ggplot2` package.