Module 2.1

R packages: expanding your skill set

TWS-WS 2022

In this module we will talk about how you can continue learning, gaining confidence, and expanding what you can do in R using contributed **packages**.

Whether you are an expert or a novice, you will always encounter situations where you don't know exactly what to do. The great thing about R is that there is *nearly always a way to do it* (whether in base R or via **packages** contributed by R users from around the world), and once you get familiar with the basics, it's usually not that hard to figure out how to accomplish your analysis objectives!

Load script for module #2.1

1. Click here to download the script! Save the script to a convenient folder on your laptop.
2. Load your script in RStudio. To do this, open RStudio and click on the folder icon in the toolbar at the top and load your script.

Packages

How is it possible to have so many features implemented in a single programming environment?

The answer is that you have a whole community of researchers adding new functionality *all the time*! This new functionality is implemented in the form of **R Packages**. Each package includes a set of functions and other objects for accomplishing specific tasks (and associated help files!).

Mode example

For a simple example (borrowed from Prelude in R), base R does not include a function for computing the *mode* of a vector. BUT, some R user decided that functionality would be useful! They coded up a function for computing the mode (in many different ways!), and then they documented it and wrapped it up into a package, called "modeest", and made the package available in the CRAN repository (the 'official' repository for R packages).

Packages in the CRAN repository can be installed using the function "install.packages()":

```
##############
# PACKAGES!
##############

install.packages("modeest")    # run this if you haven't yet installed the package from CRAN!
                                   # you only need to install this once, so you can comment it out if
                                   # you already have this package installed
```

```
library(modeest)    # load the package: This is package 'modeest' written by P. PONCET.
```

```
## Warning: package 'modeest' was built under R version 4.1.2
```

```
   # install.packages("BiocManager")
   # BiocManager::install("genefilter")    # you might need to install a dependency in order to make th
```

Now that we have installed and loaded the package in our current R session, we can learn more about it! One quick way to get a quick overview of the package is to use the following syntax:

```
library(help = "modeest")    # get overview of package
```

Can you use the package overview to identify a function for computing the mode of a data vector?

Let's try to use it! If you haven't already loaded the "data.missing.txt" dataset, download it here and save to your working directory.

```
newdf <- read.table(file="data_missing.txt", sep="\t", header=T)

# ?mlv   # learn more about the function for computing the mode (most likely value). Who knew there wer

  # lets find the most frequent value(s) in the "Export" column:
mlv(newdf$Export, method="mfv", na.rm = T)
```

```
## [1] 0
```

Removing a package from your environment Not that you need to do this very often, but if you want to remove a package from your current session, you can use the "detach()" function:

```
detach("package:modeest")  # remove the package from your current working session
```

3d scatterplot example!

Before we move on, the "car" package (_c_ompanion to _a_pplied _r_egression) gives us an example of the endless, amazing things we can do with R packages. Here we will make our own interactive 3d plot! (example from "prelude in R").

Here is a data set that represents the number of times a dog barks in a day, and two potential explanatory factors: food given (in lbs) and cars passing by the house.

```
#########
# 3D Plotting example
#########

#########
# Data: dog barks per day (and two explanatory variables)

Cars= c(32, 28, 9, 41, 23, 26, 26, 31, 12, 25, 32, 13, 19, 19, 38,
        36, 43, 26, 21, 15, 17, 12, 7, 41, 38, 33, 31, 9, 40, 21)
Food= c(0.328, 0.213, 0.344, 0.339, 0.440, 0.335, 0.167, 0.440, 0.328,
        0.100, 0.381, 0.175, 0.238, 0.360, 0.146, 0.430, 0.446, 0.345,
        0.199, 0.301, 0.417, 0.409, 0.142, 0.301, 0.305, 0.230, 0.118,
        0.272, 0.098, 0.415)
Bark=c(15, 14, 6, 12, 8, 1, 9, 8, 1, 12, 14, 9, 8, 1, 19, 8, 13, 9,
       15, 11, 8, 7, 8, 16, 15, 10, 15, 4, 17, 0)
```

Let's install and load the "car" package:

```
install.packages("car")
install.packages("rgl")   # you need this one as well!

library(car)
```

Now for the fun part- interactive 3-D graphics! Run the following code and take a minute to play around with the resulting scatterplot.

```
##########
# interactive 3-D graphics!

car::scatter3d(Bark~Food+Cars,surface=TRUE)
```

Loading packages from GitHub

Many package developers use GitHub to make their packages available. Sometimes these packages are not yet on CRAN because they are not yet ready for prime time (so be careful!). Other times developers may choose to bypass the CRAN repository entirely, and instead just use GitHub. Most of the time, developers will use both GitHub and CRAN, where the version available on CRAN is not the latest version – for the latest version, you will need to use GitHub. There are some good reasons to put an R package on GitHub:

1. It's easier for others to peruse your code.

2. People can actually fix any problems and send you a patch, which you can easily test and then incorporate into your package. Rather than having someone say, "There's a typo in your documentation" they can say "Here, I've fixed a typo in your documentation."

3. With the install_github() function in the 'devtools' package, it's easy for people to install your package directly from GitHub. It doesn't have to be on CRAN (getting your package on CRAN can be a bit difficult)

The syntax for using "install_github()" is just install_github("[author]/[package]") Here's an example using the "install_github()" function:

```
###########
# install package from GitHub:

 # install.packages("remotes")    # run this if you haven't already installed the "remotes" package
library(remotes)
install_github("kbroman/broman")  # install a random package from GitHub!
```

Finding the right package for the job

Usually a Google search is the best way to find packages for doing something you want to do! But you may also read about new packages in a journal (e.g., Methods in Ecology and Evolution) or textbook, or conference, or just talking with other researchers!

Learning more about how a package works

It's not always perfectly obvious how to get started in actually using a new package. Here are three tips.

Built-in package overview We've already seen this one- just use the following syntax to pull up an overview of any package:

```
library(help = "[package name]")
```

For example,

```r
###########
# Learning more about packages
###########

###########
# Package overview

library(help = "car")     # help file for the useful "car" package for applied regression
```

Package vignettes "Vignettes" are provided in many packages, and are tremendously useful, providing worked examples. Usually the package overview (above) will tell you which vignettes are available (see above). Use the following syntax to pull up a vignette:

```
vignette('[name of vignette]', '[name of package]')
```

For example,

```r
##########
# package vignette

browseVignettes('car')
vignette('embedding','car')   # pull up the "embedding" vignette in the 'car' package
```

Or let's take a look at the "dismo" package, which is for species distribution modeling in R:

```r
################
#### GENERAL TIPS
################

#############
# 1. Use code examples provided by others

# install.packages("dismo")     # install "dismo" for species distribution modeling
```

```r
browseVignettes('dismo')
vignette('sdm','dismo')      # pull up one of the helpful vignettes from the 'dismo' package, with usef
vignette('brt','dismo')      # and another one!
```

Package demos Some packages include "demos", which are worked examples that are available with a single click. Here's an example:

```r
########
# package demo

demo(package="stats")    # list all demos for package 'stats', which is included in base R
demo('nlm','stats')
```

Package documentation (html or PDF) The 'master' documentation for any package is often packaged as a PDF file that can be downloaded from CRAN (each package on the CRAN repository has its own webpage- for example, here's the one for the raster package)

Also, you can load the documentation for any package you have installed using the "help.start()" function:

```r
###########
# Load html documentation for R and all installed packages

help.start()
```

Learning by example! To me, there is no better way to learn (1) general coding, (2) best practices for statistical analyses, and (3) how to take advantage of specialized functions in packages than to try to work through and understand code examples that others have written.

One way to do this- for base R and for packages, is to access the examples provided in the package documentation. These examples are provided at the end of the help file for any function (using 'help()' or '?'). You can also run these examples using the "example()" function. For example:

```r
#########
# Built-in examples

example(lm)   # run examples for "lm" function (included in base R)
```

Unfortunately, the code examples provided in help files (e.g., when using the leading question mark to pull up a help document) are often surprisingly unhelpful. Instead, you can try the package vignettes, or just use your favorite search engine to find more helpful examples on the web!

Package citations Most packages include one or more citations, which are often peer-reviewed papers that document the novel algorithms included in the package, often in more detail than the help files. These are also the papers you should cite if you use the package for a manuscript. You can find these citations using the "citation()" function:

```r
########
# package citations

citation('car')   # citation for the 'car' package
```

```
##
## To cite the car package in publications use:
##
##   John Fox and Sanford Weisberg (2019). An {R} Companion to Applied Regression, Third
##   Edition. Thousand Oaks CA: Sage. URL:
##   https://socialsciences.mcmaster.ca/jfox/Books/Companion/
##
## A BibTeX entry for LaTeX users is
##
##   @Book{,
##     title = {An {R} Companion to Applied Regression},
##     edition = {Third},
##     author = {John Fox and Sanford Weisberg},
##     year = {2019},
##     publisher = {Sage},
##     address = {Thousand Oaks {CA}},
##     url = {https://socialsciences.mcmaster.ca/jfox/Books/Companion/},
##   }
```

```r
citation()    # and here's the citation for R in general- useful for when you use R for manuscripts
```

```
##
## To cite R in publications use:
##
##   R Core Team (2021). R: A language and environment for statistical computing. R
##   Foundation for Statistical Computing, Vienna, Austria. URL
##   https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2021},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please cite it when using it
## for data analysis. See also 'citation("pkgname")' for citing R packages.
```

Online forums In some cases you may run into issues with a particular package that you just can't seem to solve. It's still likely that others have had this problem! If you use some appropriate keywords in a google search you may find a link to an online forum (like stackoverflow) where someone has answered a similar question. It's amazing how many knowledgeable people are answering questions on these forums - they are a tremendously valuable resource!

And of course, if you still can't find an answer to your question, you can ask your question on one of these online forums. The best ones are probably Nabble, Stack Overflow, and Cross Validated.

Ask the package author!! When all else fails, you can always reach out to the package author! The author's name and contact info are listed in the package overview, and I have found they are generally very responsive and willing to help!!

'Cheat sheets'

There are lots of "cheat sheets" available for reminding you of basic R functionality. It can be useful to tape these to the wall next to your desk!

For example, here is a good cheat sheet for using base R.

see Links page for more resources, including cheat sheets!

Google!

I don't even understand why it works so well to google R help ('r' is just a single letter after all, how does a single letter help a search?), but it works pretty well!! With some basic programming knowledge, basic R know-how including how R packages work (and some cheat sheets by your desk), and the internet, you can pretty much always find a solution!

More helpful links for going further with R are provided on the Links page.

DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS, AND OTHER "NOT COMPUTER PEOPLE."
WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:
START
FIND A MENU ITEM OR BUTTON WHICH LOOKS RELATED TO WHAT YOU WANT TO DO.
I CAN'T FIND ONE
PICK ONE AT RANDOM.
I'VE TRIED THEM ALL.
OK
OK
CLICK IT.
GOOGLE THE NAME OF THE PROGRAM PLUS A FEW WORDS RELATED TO WHAT YOU WANT TO DO. FOLLOW ANY INSTRUCTIONS.
NO
HAVE YOU BEEN TRYING THIS FOR OVER HALF AN HOUR?
NO
DID IT WORK?
YES
YES
ASK SOMEONE FOR HELP OR GIVE UP.
YOU'RE DONE!
PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN. CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

Any questions???

–go to next module–