**Guide to using packages in R**

Benjamin Huber

Packages are an essential part of data analysis in R. While the base R programming language can read, manage, and analyze data, there is a lot of functionality that must be loaded via optional packages. There are currently 20,740 packages available to download via CRAN, the database of R packages, and even more available on GitHub. This abundance is one of the best reasons to choose R for your data analysis because there is probably a package, or multiple packages, that can do anything you want to do. But you need to exercise caution when choosing which packages to use because anyone can create a package. Below are some of my opinions about when to use packages and how to choose them.

How to use R packages:

* Limit package dependencies as much as possible
  + Updates (or lack thereof) can change the behavior of packages. This can break your code if you try to run it or send it to someone else who has different versions installed.
* Load R packages carefully
  + Many packages share function names. Pay attention to the order you load the package and ensure you load the function you want last.
  + Example: I always use the select function from the dplyr package to subset variables in a dataset. Sometimes I also use the MASS package. If you load the MASS package after the dplyr package the select function from the MASS package overwrites the select function from the dplyr package.
  + Don’t load packages you don’t use in that file. It makes it hard to figure out which functions come from which packages.

How I evaluate an R package:

* How popular it is:
  + If a lot of people depend on it then it is unlikely to suddenly lose support.
  + The CRAN page for the package shows the number of downloads per month.
* How old it is:
  + Brand new packages might still be working out all the bugs, but they also may have more modern features.
* How well it is maintained:
  + A lot of the packages I use are created and maintained by Posit, the company behind RStudio. They have a team for each package ensuring that everything is properly tested.
  + Packages created by grad students are unlikely to have long term support.
* How well it is documented:
  + R packages created by individuals frequently lack sufficient documentation.
  + Make sure that the functions you need are explicitly documented to ensure they are doing exactly what you think they are doing. This is especially important for niche analyses.

How to prevent code from breaking when they update packages:

* Sometimes things just break and there is nothing you can do except re-write the code.
* Taking note of which versions of each package, and which version of R were used when you originally ran the code can help you replicate it later.
* Most of the time using the renv package is sufficient.
  + renv() command will list all the packages you have loaded and all the versions.
* Use Docker when reproducibility is critical to the project
  + “Containerize” your project so that it downloads all the appropriate software and packages every time you run it.

**My Favorite R packages**

Note: Many of the packages included below are part of the tidyverse family of packages, created by Posit. Most people either heavily use tidyverse or don’t use it at all. I use it so much that I almost consider to just be part of R and I do almost everything within the tidy paradigm of R coding.

* Import Data
  + readr: CSV, Excel, Text
    - https://readr.tidyverse.org/
  + haven: SPSS, SAS and Stata data sets
    - https://haven.tidyverse.org/
* Wrangling:
  + dplyr & tidyr
    - Data Wrangling and cleaning packages
      * <https://dplyr.tidyverse.org/>
      * <https://tidyr.tidyverse.org/>
  + lubridate: dates
    - https://lubridate.tidyverse.org/
  + stringr: strings
    - https://stringr.tidyverse.org/
  + forcats: factors
    - https://forcats.tidyverse.org/
  + Hmisc
    - Frank Harrell’s miscellaneous functions package. Includes several useful functions for insights into data, especially *contents* and *describe*. This is a large package with a wide variety of well done functions and is worth exploring.
    - https://hbiostat.org/r/hmisc/
* Plots
  + ggplot2
    - Universal plotting package
    - https://ggplot2.tidyverse.org/
  + RColorBrewer & viridis: color scales for plots
  + patchwork
    - layout multiple ggplots
    - <https://patchwork.data-imaginist.com/>
  + cowplot
    - https://github.com/wilkelab/cowplot
  + leaflet
    - Interactive maps
    - https://rstudio.github.io/leaflet/
* Tables
  + gtsummary
    - Easily make neat tables and calculate simple statistics
    - https://www.danieldsjoberg.com/gtsummary/
  + flextable
    - Table Formatting for Microsoft compatible output
    - https://ardata-fr.github.io/flextable-book/
  + broom/broom.mixed
    - Convert analysis output to dataframes. Use when you want to reorganize the output into a custom table.
    - https://broom.tidymodels.org/
  + sjPlot
    - Alternative to Gtsummary for outputting regression tables
    - https://cran.r-project.org/web/packages/sjPlot/index.html
* Output
  + openxlsx
    - Formatting and tools for creating excel native R outputs
    - https://ycphs.github.io/openxlsx/index.html
* Analysis
  + purrr
    - Functional programming for R! Excellent for repetitive analyses
    - https://purrr.tidyverse.org/
  + lavaan: Structural Equation Modelling and latent variable analysis
    - https://lavaan.ugent.be/
  + psych: More latent variable analysis
    - https://personality-project.org/r/psych/
  + lmerTest/lme4: mixed models
    - <https://github.com/lme4/lme4/>
    - <https://github.com/runehaubo/lmerTestR>