Different ways of running R

a **statsTeachR** resource

The many ways of running R code

- an interactive session
- `knitr`
- `source(), for loading code`
- `R CMD BATCH` at command line, for running code

Working with R interactively

To run interactive sessions in R, we have been using RStudio, but you can also run the R GUI app that comes with the R download. Or can run at the command line by typing

```
$ R
```

Advantages of interactive sessions

- Encourages data exploration, tinkering, failure until you get it right
- Immediate feedback
- "Easy" to debug (or at least isolate problem code)
- Archiving of old commands for easy insertion into source files (RStudio)

Disadvantages of interactive sessions

- More difficult to reproduce your analysis
- When running long jobs, you can't work in R unless you start another R session
- Engourages sloppy, unsystematic coding practices

Using `knitr` to run R code

You can also use the `knitr` package to run R code and produce reports (HTML, PDF, DOC formats) or slides (Slidy, Beamer, ioslides). Knitr relies on you having written a .Rmd (RMarkdown) or .Rnw (technically, a Sweave file, but this is sort of outdated technology) file. Knitr is most easily used from within RStudio, but can also be called from within R, using for example:

```
R> rmarkdown::render("input.Rmd")
```

Advantages of using `knitr`

- Easy to produce nice-looking, reproducible reports
- Runs directly within RStudio
- Caching of large chunks of code
- Customizability in terms of how much code/raw output you show

Disadvantages of `knitr`

- More difficult to troubleshoot/debug
- If something breaks, you have to start over from scratch
- Not as nibmle for data exploration
- Extra baggage if all you want to do is execute some lines of R code

Sourcing R files directly

- From within R, you can source a file containing R code.
- Typically, this is done to "source" a bunch of functions that reside in a separate file.
- This is essentially what loading a package does too (with some other bells and whistles).

```
R> source("my-code.R")
```

Sourcing code: tips

- Keeps your code tidy and readable
- Keeps your code organized, e.g. one .R file for functions, another for code to run directly
- Best practice is to not operate on your existing workspace when you source a file, just load things in.
- `source()` files that **define** things, but not that **do** things.

Using `R CMD BATCH` to run R code

- You can run an R file without explicitly opening R.
- Typical use case: you have a script that runs an analysis and outputs some data. You don't need a report, just want the output to be saved somewhere.

```
$ R CMD BATCH my-script.R my-script.Rout
```

Advantages of using `R CMD BATCH`

- Can run files that take a long time without hanging up your R GUI.
- Results are reproducible
- You can see the output from the R session by looking at the .Rout file.
- The R script defines a clear workflow.

Disadvantages of `R CMD BATCH`

- Some people are unfamiliar with the command line interface
- As with knitr, harder to debug as you run, code needs to be polished

Summary

interactivity, `knitr`, `source()`, `R CMD BATCH`, ...

- I use each of these methods of running R on a daily basis.
- They each have their time and place!
- It takes time to understand how, when, and why to use each one.