This Workshop

• You’re not going to be an expert using R by the end of this workshop (unless you already are an expert)

• My goal: give you the tools and context so that you are not too frustrated to learn on your own

• Want to really learn? Find a ‘lover’…
Some common hangups

• Getting started
• Object types
• Loading data
• Indexing objects and efficient data cleaning
• Making it work end-to-end
• Graphics & Plotting
Quick Instructor Bio

• Doctoral student in Epidemiology

• Teaching assistant for doctoral-level R course

• Course instructor for 20-hour summer R class

• Formerly computer programmer
What is RStudio?

• RStudio is *not* R
• RStudio is an IDE (Integrated Development Environment)
• The IDE idea is derived from general purpose software development with the intent of making developers more efficient by integrating coding tools and output of code into a single interactive environment
Start RStudio

• This is probably obvious; let me help you if it's not

• Consider adding a desktop or start menu/Dock shortcut if you have trouble figuring out where to go to start RStudio.
Orientation: Console
Orientation: Console

- The R interpreter: The core of R. You can think of it as a combination of the SAS log window and output window with an editor twist.
- Try it:
  - `print("I am an awesome R programmer")`
  - `2 + 2`
Orientation: Environment/History
Orientation: Environment

• Shows objects present in your environment.
• Like the SAS dataset explorer but contains more than datasets – local variables, etc.
  – It’s probably empty right now
Orientation: History

• History (the tab next to workspace)
  – Condensed view of the console that shows only the commands you executed (not R’s response)
  – Try it: Do the commands you typed in the console show up here?
Orientation: Files/Plots/Packages/Help
Orientation: Files/Plots/Packages/Help
Orientation: Files/Plots/Packages

• Files
  – Like Explorer in SAS

• Plots
  – Like Graph Window in SAS

• Packages
  – Helps install, load, and track packages (we’ll cover packages in more detail later)

• Help
  – Shows help pages
Orientation: Code Editor

• Create a new script file:
  – File::New File::Rscript
Orientation: Code Editor
Orientation: code editor

• A place to edit your code: Comparable to SAS’s editor windows
  – You will usually want to use the code editor rather than the console for most of your work
Code editor

• Try it: Send a command from editor to console
  – First, type a command in the editor:
    • 2 + 2
  – Next, send to the console:
    • Control+R on a PC
    • Command+enter on a Mac
    • Or click Run

• Comparable to highlight/run in SAS
Code editor: an additional note

• Tab completion: takes advantage of workspace data to save keystrokes
  – Try it:
    • Type me then hit the tab key
      – What happened?
    • Now hit the tab key again
      – Now what happened?

• You don't have to use tab completion, but it may make you more efficient
Import Dataset

• On the Environment tab
Import Dataset

• Like SAS's import dataset wizard
  – Pick location, dataset type, etc.
  – Supports only plaintext datasets

• Try it:
  – Import from web URL:
    http://www.columbia.edu/~sjm2186/SER2014/titanic.csv
Import Dataset, cont’d
Import Dataset, cont'd

• Import wizard is a code generator
  – Look in console window to see code

• You *can* import other format (SAS, Excel, etc.) datasets into R
  – But you'll need to write this code by hand – we’ll get there later today
Data Viewer (Grid View)

• Import opens the grid viewer automatically

• (Or in Environment window, double-click on your imported dataset (titanic))
  – Data viewer opens in window with editor

• Comparable to SAS dataviewer
Takeaway points

• RStudio is a wrapper around an R console to help you use R effectively

• (Usually) write your code in the editor, send it to the console

• General orientation
Questions
Batch vs. interactive

• **SAS is fundamentally a batch system**
  – Analysis in SAS involves writing code (editor window), sending that code to a “central system” for processing, which generates results (output window) and metadata about the results (log window)
  – This model dates from the mainframe/time sharing era of computing

• **R is fundamentally interactive**
  – Analysis in R involves running code in an interpreter, which creates in-memory objects that can be affected by more code
  – Generating output (and output metadata) is a *side effect*, from the standpoint of functional programming
  – This model dates from the PC era of computing

• **Note:** This distinction is just a loose description of the model; we usually use SAS interactively, and R can be used for batch processing (e.g. in high-performance computing clusters)
Projects

• Basic unit of organization for RStudio
  – RStudio only lets you open one project at a time
  – Contains code, data, graphs, etc.

• Usually maps to your mental concept of the project, not necessarily to one dataset, one publication, etc.

• Projects can make collaboration easier
  – Per-project settings
  – Source control integration