Lecture 2.4 – Scripting & Troubleshooting Compound Code

Specific Learning Objectives:

3.5 – Think and work independently with code.

3.5.1 – Learn basic skills in debugging and troubleshooting error messages.

3.5.2 – Search for effective solutions and tools using online resources.

Up Until Now…

- We've been focusing on the basic functions of R and running single lines of code.

- The **real power** of computing is in getting the computer to do complex things for you, which often requires more than one line of code to work together in sequence.

- We'll call this **compound code** because each line compounds the next to produce a new analysis.

- Scripting (or writing) compound code is difficult, but like writing a long paper, it helps to know some strategies for how to start and build on your code!

Data Explains Scripting Compound Code

- The **key to creating complex, compound code** is to start simple and build on it!

  - Absolutely NO ONE writes 1,000 lines of code top to bottom on the first try.

  - The more experience you have, the easier it will be to structure the code you write (what you should work on first, second, last, etc).

- Everyone starts with *a simple thing that works* and builds on that example.

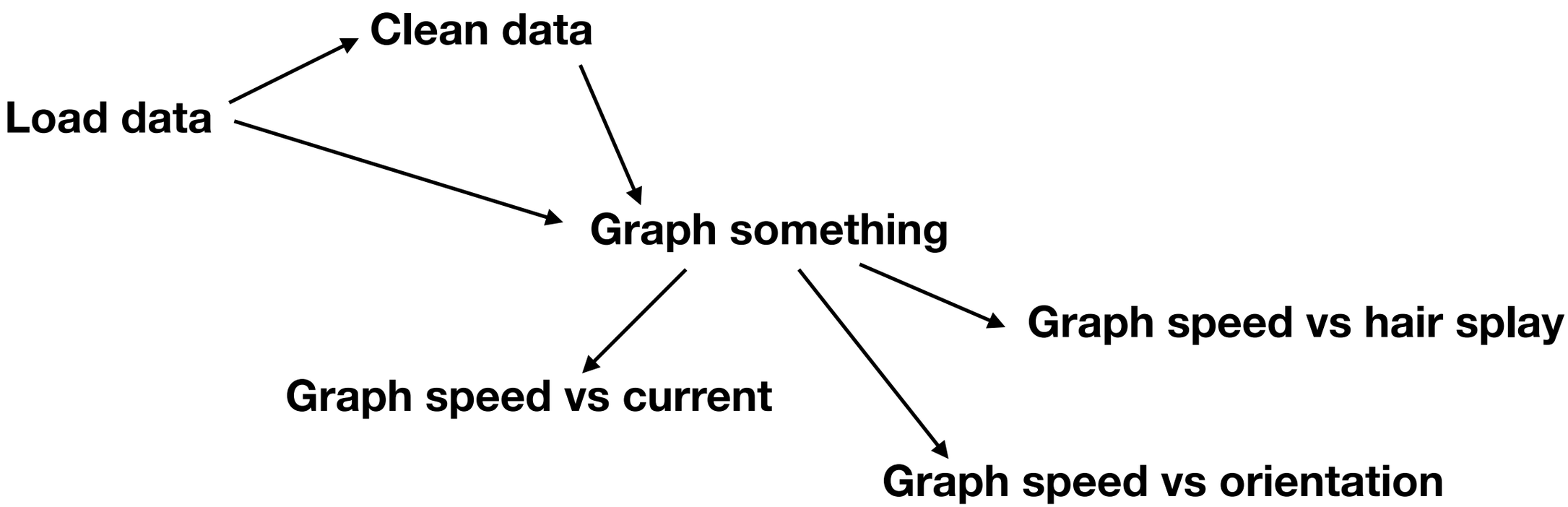
- We've already been practicing this through the first few assignments in Unit 2. Today, we'll formalize it a bit more to create a longer chunk of code.

Example Problem Walkthrough

In my dissertation, I made models of crab antennae hair arrays and measured fluid speed through the spaces in between the hairs. I had a bunch of conditions: splayed or clumped arrays, backward or forward facing, and how fast external fluid was moving.

I need to visualize these data and decide which factors are important. The data are in `bluecrab.csv` in the data zip.

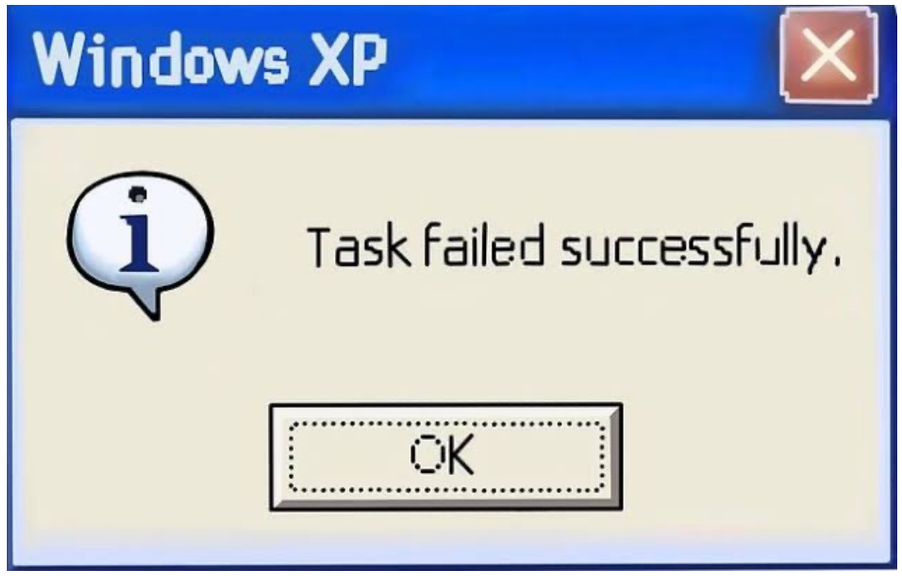
How should we approach this problem? Where to start?



Check Your Understanding

Import the data set in `tomatohaul2021.csv` as `tomato.data`. Graph tomato harvest (in lbs) versus date. Make the paste tomatoes blue and the heirloom tomatoes red. Be sure there are both points and lines on the graph. Include a legend. Add an additional element!

What if things break?



- The best part of compounding code is that you know the spot when things worked – and then when they stopped working.

  - Building code layer-by-layer and running it/fixing it at each layer will give you a good idea of what is working and what is not (usually the thing you just added).

  - Be sure to SAVE your script regularly so if things break (and R Studio crashes) you won't lose progress.

  - Don't forget about our 4 steps of debugging (spelling, case, syntax, environment)!

  - If those don't work, split the code until you find the issue. (There may be more than one error!)

  - Focus on the code you suspect doesn't work, but always be willing to sanity-check stuff you think works!

What if things break?

- What happens if you STILL can't figure out the problem?? Try searching the error message online.

- Use part of error message that is general (not specific)
- Read several results from your search (not just one)
- Read to make sure the error and solution are close to what your problem is.
- Be sure you understand why the fix works.
- Try several solutions and check the output to see if it what you want.

Action Items

1. **Complete Assignment 2.3 using R Markdown.**

2. **Read Davies Chapter 8 and Chang Chapters 1-2 for next time.**