Graphical Analysis of Biological Data

Notes 01

Contents

This web page is available as a PDF file

Who am I?

Dr. Mike Taylor
Department of Biology
Southeast Missouri State University

My educational background:

- B.S. in Biology (Chemistry minor) from Central Missouri State University.
- Ph.D. in Zoology from Louisiana State University in the lab of Dr. Mike Hellberg.
- Postdoctral Researcher at the University of Notre Dame in the lab of Dr. Jeff Feder.

My computer background:

My first computer was a Macintosh SE with 4 (!) megabytes (MB) of RAM, a 20 MB hard drive, and one 800K 3.5" floppy disk drive (they weren't actually "floppy"). Compare that your laptop or desktop computer. Or your smart phone. Back then (ca. 1988), I ran a bulletin board system on that computer with a U.S. Robotics Courier 14.4K Dual Standard modem. In comparison, typical broadband wireless is at least 17X faster. Back then, my system was the shiznit.

I have formal training in the Pascal and C programming languages. I am self-taught in R and in Python. If you find yourself wanting to learn another language after R, learn Python.

Who are you?

You will tell me and the rest of the class a little bit about yourself as part of the first homework assignment.

Approach the course with curiosity

I expect that you will be curious. The notes for this course will be brief but they will have links scattered throughout. Follow the links and read the material. Do not follow the tl;dr way of thinking. Others have

written tutorials, exercises, and other helpful information far better than I could write. Become a better you and take advantage of these resources. If you do not, you will not be successful in this course.

Follow the links! Read. Learn. Practice!

Data analysis requires coding. To learn proper coding, you will have to type R code. *Lots* of R code. Do not give in to the temptation to copy and paste from examples. You will not learn the R language well that way. Typing the code yourself will "drill" the language into your muscle memory and your brain memory. You will learn it better. You will retain it longer. You will have a better chance of getting a job. Just type it!

Make your work reproducible

A central tenet of science is reproducibility. Yet, often it is not. Read this introduction to reproducible science from rOpenSci. Follow the other links at your leisure. The tools you will use during this course emphasize the principle of open science. I feel strongly that you should adopt this philosophy in your scientific endeavors.

Your work should be reproducible. Document it, test it, and leave a breadcrumb trail (version control), even when you are the only collaborator.

Embrace text files

Use plain text files for all of your documents. Plain text files are small. Plain text files are portable. Text files can be opened and read on any computer plaform (e.g., OS X, PC, Linux) with any text editor, unlike (for example) Word documents, which use a proprietary format, and are subject to change at any time. Text files are the common currency of reproducibility and open science.

This course, and in fact nearly everything I use for teaching, is built from plain text. Text files are the base that can be converted into a variety of formats, including web pages, PDF files, and even (gasp!) Word documents. This webpage was generated from this markdown source file on GitHub.

RStudio, which you will use throughout this course, has a built-in text editor, and it will suffice for this course. In the future, or even during this course, you may decide to install a stand alone text editor. Browse this list of free text editors to find one that works on your computer platform and suits your needs. Atom works on all platforms and has a good reputation. I used it to write this part of these notes. Nice features are that you can live preview your markdown documents and it can interact directly with Git and GitHub (more on both of these later).

I have used BBEdit for OS X since version 4 (now at version 12+) as a good general purpose text editor. You can use the full feature set free for 30 days. After that time, you can still use most of its important functions for free.

For PC, Notepad++ has a good reputation but you may want to try a few to find a text editor that suits your style and needs.

Do not use Microsoft Word to export plain text files. They are not truly plain text. Cut your umbilical cord. Leave Microsoft for the business world and the less enlightened SEMO community.

Use Git and GitHub

During this course, you will write lots of R code. Your code will run. Eventually. But, you think you can do better. You pursue a thread of an idea, and edit your code. You follow that thread down the rabbit hole, and write even more code. You test your code and realize it does not do what you wanted. The Mad Hatter laughs at you. You try to back out of the rabbit hole but you are lost. You have a jumble of non-working code that even a hookah-smoking caterpillar cannot understand.

You can avoid falling through the looking glass by using Git. Git is a version control system (VCS) that tracks the changes you make to files through time. You can go back in time to any previous version. You can branch into a copy of a working program to try a new idea. Does your new code work? Awesome, merge it into to the main file. Disaster? No problem, because the original is intact. You laugh at the Mad Hatter and seek further adventures with the White Rabbit.

You may have used some form of version control when writing a report. You start the report, then you make a copy, add a date or version number, edit the copy, then make another copy with a new date or version number. You may have multiple copies spread across flash drives or even emailed to yourself. Git makes that process much more efficient, in terms of time and storage.

GitHub is a hosting service where you can store and access all of the projects that you maintain by Git version control. I use Git and Github to maintain all of my course materials and other bits of code that I write. I am using GitHub tools and web services to manage this course. We will touch on the basics of version control for this course but we will not go very far down the rabbit hole.

Git and GitHub have a bit of a learning curve. We will start gently and add something new every so often. You will get soon get familiar with the basics of Git. Also, RStudio integrates well with Git and GitHub, which makes version control of your R projects very easy.

An excellent document to read is Why Git? by Dr. Jennifer Bryan. She provides a good overview of how Git works and why you should use Git (even if not collaborating).

Save that which is real

As you interact with Git and GitHub, you will be tempted to commit and push all files. Do you really need to do that? As long as you have the source files, like the R scripts and R Notebooks, you can generate the output. If you lose the source, then you lose the output. Think in terms of source and output, and remember that *source is real.* Output can be recreated from the source. You only need to commit and push source files.

If you find yourself tempted to commit output files (e.g., HTML files), that is OK. You will get over it. Eventually.

Using the terminal or shell

We will avoid this when we can, but you will at least need it to install *git* as part of the first assignment. Here is the minimum need-to-know about the **terminal** or **shell**:

- Way to control your computer.
- OS-specific language!
  - Mac/linux/unix are about the same.
  - Windows has its own
- Navigate with `cd`, `pwd`, and `ls`. `.` and `..`

A great introduction to this stuff is in happygitwithr: Appendix A.

You can even use the shell from within RStudio.