BIOST 561: welcome + tidyverse + product & workflow

Lecture 1

Welcome!

BIOST 561 treats advanced programming and computing skills for biostatistics.

We introduce a sophisticated use of `R`, as well as other programming techniques and tools, including an introduction to using UW Biostatistics' computing resources.

About the course. . .

I know from your previous courses that everyone is quite comfortable with `R` already! The purpose of this course is to

- get faster and more proficient at things you already know how to do
- go a bit deeper into the details of `R`
- learn how to write code that other people will run
- scale up!

How to address me

“Kevin” is great. If that feels weird for you, “Professor Lin” is also great.

Agenda for today?

- The “real” goals of the course
- Discussion of what problems an R package or GitHub are trying to solve
- Going over the syllabus
- Showcasing a demo GitHub repositories – simple example of what you will make for your final project
- Teaching the basics of git and GitHub

Discussion: “Software engineer”

Discussion: “Software engineer” = “Someone who codes”

Discussion: “Software engineer” = “Someone who codes”

- (In class: Pause for discussion)
- I will not proceed without at least 3 people to voice an opinion

So. . . what's *really* the goal of the course?

There is a ocean's worth of difference between: 1) knowing how to write code, and 2) having the vision and execution to bring an entire vision's into a stable code-base that other people can easily use.

You wouldn't call:

- Anyone who can play the notes of Rachmaninoff No. 2, a master pianist
- Anyone who can sing Taylor Swift's "Anti-Hero", a star performer
- Anyone who can memorize Chris Murphy's lines in Oppenheimer, a Oscar-winning actor

Specific topics (Part 1)

- Separating product from process and gaining familiarity with RStudio and R projects
- Using Git and GitHub for version control and code distribution and integrating these tools into your typical coding workflow
- Writing R packages, including writing unit tests to ensure your code is properly maintained
- `tidyverse`, including `dplyr`, `magrittr` (i.e., pipes %>%), `tibble`, and `ggplot`
- Comfort in writing functions,

Specific topics (Part 2)

- Debugging, profiling code for speed and memory
- Writing simulation studies
- Basic familiarity with using the terminal
- Submitting code to be run on the server (specifically, the Bayes Biostatistics server)
- Creating a Pkgdown website and a personal website via Github.io

Give me the vision

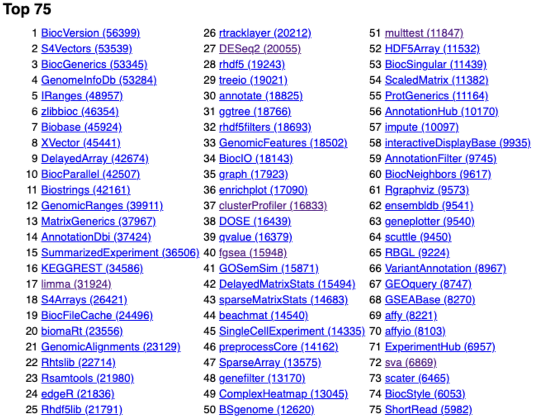
- A huge portion of the course is specifically about building familiarity with R packages and GitHub
- Why?
  - GitHub allows for efficient deployment of code so others can use it
  - There are tons of great software engineering tools in RStudio that are relevant for only R packages

R packages: How to distribute your ideas

- You can install anyone's R package via `install.packages()` or `devtools::install_github()`
- Nowadays, it's often the case that what drives a paper's success after publication:
  - **is not**: the intellectual nuances, the journals, the scientific novelty, the benchmarking
  - **is**: whether an bystander can easily install and run your code to help advance the science they're interested in

R packages: How to distribute your ideas

- Most biostatistics-related methods have humble coding beginnings
- See: https://bioconductor.org/packages/stats/



Top 75

1 BiocVersion (56399)
2 S4Vectors (53539)
3 BiocGenerics (53345)
4 GenomeInfoDb (53284)
5 IRanges (48957)
6 zlibbioc (46354)
7 Biobase (45924)
8 XVector (45441)
9 DelayedArray (42674)
10 BiocParallel (42507)
11 Biostrings (42161)
12 GenomicRanges (39911)
13 MatrixGenerics (37967)
14 AnnotationDbi (37424)
15 SummarizedExperiment (36506)
16 KEGGREST (34586)
17 limma (31924)
18 S4Arrays (26421)
19 BiocFileCache (24496)
20 biomaRt (23556)
21 GenomicAlignments (23129)
22 Rhtslib (22714)
23 Rsamtools (21980)
24 edgeR (21836)
25 Rhdf5lib (21791)
26 rtracklayer (20212)
27 DESeq2 (20055)
28 rhdf5 (19243)
29 treeio (19021)
30 annotate (18825)
31 ggtree (18766)
32 rhdf5filters (18693)
33 GenomicFeatures (18502)
34 BiocIO (18143)
35 graph (17923)
36 enrichplot (17090)
37 clusterProfiler (16833)
38 DOSE (16439)
39 qvalue (16379)
40 fgsea (15948)
41 GOSemSim (15871)
42 DelayedMatrixStats (15494)
43 sparseMatrixStats (14683)
44 beachmat (14540)
45 SingleCellExperiment (14335)
46 preprocessCore (14162)
47 SparseArray (13575)
48 genefilter (13170)
49 ComplexHeatmap (13045)
50 BSgenome (12620)
51 multtest (11847)
52 HDF5Array (11532)
53 BiocSingular (11439)
54 ScaledMatrix (11382)
55 ProtGenerics (11164)
56 AnnotationHub (10170)
57 impute (10097)
58 interactiveDisplayBase (9935)
59 AnnotationFilter (9745)
60 BiocNeighbors (9617)
61 Rgraphviz (9573)
62 ensembldb (9541)
63 geneplotter (9540)
64 scuttle (9450)
65 RBGL (9224)
66 VariantAnnotation (8967)
67 GEOquery (8747)
68 GSEABase (8270)
69 affy (8221)
70 affyio (8103)
71 ExperimentHub (6957)
72 sva (6869)
73 scater (6465)
74 BiocStyle (6053)
75 ShortRead (5982)

Figure 1: Screenshot of the most popular BioConductor packages

R packages: How to distribute your ideas

- (In class: Looking at the folder organization of an R package)

GitHub: How to distribute your ideas

- GitHub serves a slightly different purpose
- In a project, for yourself:
  - Version control (We will discuss this in detail later today)
- In a project, for others:
  - Stay up-to-date on what changes happens to a codebase
  - Take a quick peek to view the source code in a convenient way

GitHub: How to distribute your ideas

- When it's time for you to find a job:
  - Your GitHub profile is a great way to demonstrate your coding portfolio to future employers
  - Your GitHub provides a natural way to host your personal webpage

GitHub: Examples of webpages

- https://linnykos.github.io/
- https://gqi.github.io/
- https://anna-neufeld.github.io/
- https://yezhengstat.github.io/
- https://rdevito.github.io/web/
- https://ekatsevi.github.io/
- https://zhimeir.github.io/
- https://larrylehan.github.io/
- https://frkoehle.github.io/
- https://satarupa3671.github.io/
- https://lsn235711.github.io/
- https://mcelentano.github.io/
- https://yingma0107.github.io/

GitHub: Examples of webpages

- To see how these websites were built, type in: https://github.com/[[github username]]/[[github username]].github.io.
- For example, https://github.com/linnykos/linnykos.github.io

GitHub and R: The secret (and powerful) combination

- By combining GitHub and R, you can also make a website for your **project**
- This is called Pkgdown, which will cover near the end of the quarter
  - https://linnykos.github.io/tiltedCCA/
  - https://anna-neufeld.github.io/countsplit.tutorials/
  - https://katsevich-lab.github.io/sceptre/
  - https://linnykos.github.io/561_s2024_example/. This is our demo package for this course. (Your final project is to make something like this.)

Structure and expectations

- Weekly lectures
  - Live and recorded
- Homeworks
  - Most of your learning will happen through completing the homeworks!
  - Approximately 4 over the course of the quarter
  - Final project about using the skills you've learned during this course on any codebase you care about
  - Graded pass/fail
    - Pass: a good faith effort at every question that reflects a command of the concepts covered in - Lecture
  - To get credit for the class, you must pass all homeworks
- Weekly office hours

More details in Syllabus

Going over the syllabus

- (In class: Pulling up the syllabus)
- Main things to discuss:
    - Expectations inside the classroom
    - Collaboration policy
    - Usage of ChatGPT
    - Submission of homeworks
    - Mental health

Rest of today's class

1. A tour of RStudio
2. A tour of https://github.com/linnykos/561_s2024_example
3. A conception of what GitHub is
4. Discussion of homework 1

Pause

Any questions? (About the whirlwind demos or the syllabus)

- I will not proceed without at least 2 questions

Tour of RStudio

- (In class: Showcasing RStudio)
- If you would like some reading material:
  - https://book.cds101.com/a-guided-tour-of-rstudio-servers-default-interface.html
  - http://mercury.webster.edu/aleshunas/R_learning_infrastructure/Tour%20of%20RStudio.html

Tour of an example GitHub of an R repository

- (In class: https://github.com/linnykos/561_s2024_example)
- Observe:
  - The website associated with this demo package: https://linnykos.github.io/561_s2024_example/
  - R code: https://github.com/linnykos/561_s2024_example/tree/main/R
  - Loading in the entire package via `library(UW561S2024Example)`
  - Documentation via `?run_example`
  - Unit tests via `devtools::test()` (if I open the project in RStudio)
  - Pushing and pulling to sync changes with GitHub
  - Ability to install the package from GitHub via `devtools::install_github("linnykos/561_s2024_example")`

A broader view: What is “version control”?

Version control refers to how you handle different versions of files.

A common approach to version control is changing file names:

- papersims.R
- papersims-v2.R
- papersims-tmp.R
- papersims-comments.R
- papersims-hellfire.R
- papersims-v5.R
- papersims-final.R
- papersims-finalMondayedits.R
- papersims-final_final.R
- papersims-final_final2.R

A broader view: What is “version control”?

1. How many versions until this becomes intractable?
2. *Date Modified* sorting does not always help
3. Tracking changes is very difficult… merging and reverting is even harder…
4. More and more files with more collaborators
5. A dreaded computer crash

Dropbox and Google Docs can help with some of these issues, but generally not (3) or (4)!

We want to have a general approach to version control that can handle many different types of files (`.R`, `.Rmd`, `.tex`, etc.)

Git

`git` is an open-source version control system (VCS):

- Track changes to code and documents
  - *What* changes by *who* and *when*?

GitHub is a code-repository service that uses git's VCS:

- Collaborate with others effectively
- Distribute code
- Solicit improvements (*pull* requests)
- Track issues & feature requests
- Try out new things and *revert* if they don't work
- Remote backup of files
- Build your coding portfolio!

Git with life

I use git and GitHub for all my work*! [Kevin's note: I took this line from Amy, so she uses git and GitHub for everything as well]

- Collaborations
  - All my papers and R projects are on git
  - I honestly don't know why you would sensibly not use git for any project
- Portability
  - As long as I have any computer, I can work on my latest paper/draft/lecture *anywhere*, because I back them up with git
- Keep track of your "to-do" list
  - Never forget anything again! Remind yourself what still needs doing with "Issues"

{* Except data. GitHub doesn't like files over 500 Mb. And definitely not sensitive human genetic data.}

Git with life

Personal story: I've destroyed or lost my laptop (or have had close calls) many times since high school.

- Left it in Uber
- Forgot my carry-on when deboarding my flight, and my laptop was in my carry-on
- Spilled water on my keyboard
- Got oil into my laptop when I put purchased take-out food and my laptop in my backpack on the way home

Since everything was on Dropbox and GitHub, the only consequence was that I: 1) needed to spend money buying a new laptop, and 2) was angry at myself for being so careless. But I did **not** lose many months/years of my work.

Git with life

[[Kevin's note: These comments are mainly from Amy, but I wholeheartedly agree.]]

- Customize existing work
  - Do you like a tool package but want to rename something? You can `fork` it
- Visibility: social/professional network
  - Recruiters will likely look at yours
  - I always look at job candidates' GitHub pages to see if they are serious about developing good software
- Necessity
  - Every software company uses some form of version control (and some use systems even more complex than git)

How does GitHub work?

“Git” is version control on your laptop. “GitHub” is putting this version control repository on the internet.

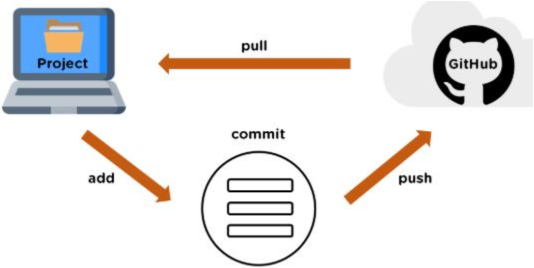


Figure 2: From https://www.simplilearn.com/tutorials/git-tutorial/git-push-command

Tour of using GitHub from RStudio

For some extra reading material

- https://rfortherestofus.com/2021/02/how-to-use-git-github-with-r

How does GitHub and R packages work?

- For now, you can use git and GitHub within RStudio
- Later in the quarter, I will show you I (and most people) use git and Github, which is through the Terminal
- I will also tell you how R packages work (bit by bit) over the quarter

With the remaining time...

- (In class: Get everyone set up on GitHub, to start Question 1 of Homework 1)
- Next week: Reviewing the basics (and some more fancy things) in `R` by default
  - data frames
  - `apply` family
  - indexing elements
  - `factor` variables
  - and more!