ESA 2023 Talk outline

**How to make a sharable code workflow for reproducible and efficient science: Learnings from {galah} and ALA Labs**

* Many of us have probably heard the phrase “replication crisis” a lot. Over my PhD, was very much a part of the psychology research community and so I heard this phrase A LOT.
* Whether a finding can be **reproduced** given the same method and data, and whether others can **replicate** a result with the same method and new data, are fundamental to science. They allow us to evaluate whether evidence for or against a theory is robust, and if so, allow us to assess the limitations of a theory. In science, reproducibility and replication are pinnacles of empirical knowledge building.
  + But it turns out this is hard.
  + In my PhD I did a lot of psychology research, and psychology famous for having replication problems.
    - Anne Scheel paper
  + Ecology isn’t safe either
    - Biology & Ecology examples
* We often discuss this in terms of hypothesis testing in theory. But analyses, often in code, can sometimes get less attention because it is a computational problem, not a scientific one
  + In the end, there is so much variation in the organisms we study and how they interact with the environment around them that knowing anything is difficult. And so replication is very difficult for experiments already. Often when talking about this “replication crisis”, people discuss, experimental design, hypothesis testing, statistical power, p hacking, and Questionable Research Practices and their possible solutions.
  + However, in this talk I will focus on one aspect of science that can inhibit reproducibility and replication even with all of those issues solved. One that takes up a huge proportion of many researchers’ time but perhaps receives less attention in scientific literature – making an analysis, very often code, fully sharable (and open data).
* Examples of how reproducible data & code is quite rare
  + Lots of research has shown data isn’t shared after publication. Code is shared even less. From 96 ecology journals, 172 articles were checked for open data and open code in 2015-16 and 2018-19.
  + Even when shared, code doesn’t reproduce results. Of 62 Registered Reports drawn from the Center of Open Science database of Registered Reports, only 37 had code available, and of those, 31 scripts could be run (83.8%) and 21 reproduced results (56.8%)
* How do we solve these problems? If you look into many published papers, people often suggest complete workflow overhauls. This isn’t viable
  + The Invisible Workload of Open Research
* Working at the ALA, we have spent a lot of time to trying solve these problems with reproducible analysis and good coding practices
  + {galah} helps users download data in their code using readable syntax
  + ALA Labs shows users good coding practices with the goal that researchers can use the same code in their own workflows
* In creating these tools, we have also had to learn how to code more reproducibly and help teach interns to do it as well.

Here are tips intended to be small steps that will make a big difference to the reproducibility of your code, allowing you to share analyses more effectively with your colleagues and your future selves

* Make your work environment shareable: Use a repository, safe links, save package versions
  + Use a shareable repository
    - Repositories like GitHub share folders and content with version control
    - GitHub puts a README right in the front to explain context, file structure and metadata
    - GitHub Desktop is easy to understand and teach because it’s visual
  + Use R tools that preserve your work environment
    - R projects: Sets the working directory to what you need
    - The {here} package: safe folder urls
    - The {renv} package: save package versions for an analysis that can be loaded again in the future
* Readable notes & Readable code
  + Object names that are clear
    - Short code is not necessarily good code. Descriptive but simple names are the goal
      * (e.g. `bandicoots\_filtered`, `make\_map`)
  + We can’t all write perfect code, but we can write good notes
    - Explain what each chunk of code should do. Minor notes for unfamiliar functions are never unhelpful
    - Notes about what a test is, how to interpret it, and how you interpreted your results help you remember what you did
      * (e.g. Kurtosis measures skewed data, with values greater than 10 considered highly skewed. We found a k value of 11.32, suggesting data is highly skewed.)
* Rendered output with middle steps included
  + Having rendered output shows your code runs start to finish. It also documents your results for you to reference without having to render the workflow again
  + You *could* make it look extra nice, **but** rendering middle steps to see input and output means even if code breaks later on, you can probably reproduce that step without breaking the whole analysis
  + Quarto is making this fast and easy.
  + Sharing code publicly is one option. But even saving these rendered documents internally will allow you to reference them or share them with others later

Summary:

* + Make your work environment shareable:
    - Use an online repository (GitHub)
    - create safe links (the {here} package)
    - Save package versions (the {renv} package)
  + Readable code, Readable notes
    - Simple, clear object & function names
    - Clear notes with interpretations
  + Render output
    - Use Quarto
    - Save rendered files somewhere findable/shareable