Updates on Initial Stock-Recruit Model Results

Presenter: Cole Brookson *Date:* 2022-02-17

Model

Followed the form in Peacock (2013)

$$\ln[R_{i,t}/N_{i,t-2}] = r - b_i N_{i,t-2} - cW_{a,t-1} + \theta_t + \theta_{a,t} + \epsilon_{i,t}$$

- Fit in `glmmTMB`
  - Null model: `survival ~ scale(S):population_name + (1|ye`
  - Alternative model: `survival ~ scale(S):population_name + scale(lice) + (1|year/area)`
- Reason for using `scale()` on the predictors is the models didn't converge without it - non-positive-definite Hessian
- Final dataset (only kept populations with > 20 stock-recruits per populations):
  - 77 populations (even/odd)
  - 1752 S-R pairs
  - 45 rivers

Why is This Different Than Stephs Results

I think there are some actually very small errors in the code that introduced some problems (**I have not fully tested this**)

Problem 1 - Sample size

In the paper (and in the final dataset generated in the code), there are: * 179 populations * 2307 S-R pairs * 99 rivers * But, I think this is actually too

large of a dataset. There is code to exclude populations with <20 S-R pairs:

```
#-------------------------------------------------------------------------------
# 4) Only keep populations with a minimum of 20 spawner-recruit pairs
#-------------------------------------------------------------------------------

Min.N<-20; Z.lengths<-c(); L<-sort(unique(ZZ$Popn));
for(i in 1:length(L)){
  z1<-subset(ZZ,Popn==L[i]);
  z2<-which(is.na(z1$Survival));
  Z.lengths[i]<-length(z1$S)-length(z2);
  }

R.LongEnough<-which(Z.lengths>=20)
R.River<-c(); R.Area<-c();
for(i in 1:length(R.LongEnough)){
  z1<-subset(ZZ,Popn==R.LongEnough[i])
  R.Area<-c(R.Area,z1$Area[1])
  R.River<-c(R.River,z1$Popn[1])
  }
```

Why Git from the command line?

- It's the only place you can run *all* Git commands
- If you know the command line version you can probably figure out a GUI version - the opposite is not necessarily true
- You might have a preference of GUI, but *all* users can use command line tools
- Interacting with servers needs to be done via command line, so you might as well learn how to do it on your own machine
- Language-specific plug-ins (i.e. Git for RStudio) force you to open the IDE for that language every time you need to make a change to a file, even if it's not in that langauge

---

Cloud-based Git repository hosting service (GitHub)

- A for-profit company that hosts Git repositories
- Free to use for public repositories (makes it *very* popular for open-source projects)

- Provides a nice interface for viewing your repositories contents
- Allows you to publish items with DOIs (links with Zenodo for this)

Important Concept: Merging

- Merging is what allows us to make the changes that happened on the "feature branch" present on the main branch, once we're sure we like them
- This can get complicated with large numbers of files, but the great thing about Git is you can **always** go back if you mess up!

Important Concept: Reverting

- We might make mistakes, and it's important to know how to "undo" those mistakes
- There are often two scenarios:
  - You want to keep some of the work you did since the "bad" commit
  - You don't want to keep any of it (usually one or two commits back)
  -