Ten Simple Rules for running a Code Club to increasing computational skills among biologists

Patrick D. Schloss[1]

To whom correspondence should be addressed: pschloss@umich.edu

1 Department of Microbiology and Immunology, University of Michigan, Ann Arbor, MI 48109

Ten Simple Rules

Introduction

- We are awash in data
- Traditional undergraduate and graduate training is insufficient to develop data analysis skills needed to analyze one's own data
- Common to find exceptional bench scientists who are inexperienced at analyzing large datasets
- As new people join a research group there is a mix of coding experience in the group

Our solution

- This is not unlike the situation researchers find themselves in regards to an ever growing literature
- Developed a Code Club to parallel the traditional Journal Club
- Have experimented with different strategies
- Overall goal is to enable intra-research group learning surrounding topics in data analysis

Initial format

- Code review - project someone's script, go line by line through the code with each person in room stating the logic behind the code
- Pros
  - Emphasizes importance of readability
  - People get to see real, smelly code
- Cons
  - Can be hard to integrate logic across lines of code
  - Although beginners could contribute, experts dominated conversation
  - Could become adversarial, especially if the presenter didn't articulate their goal for the review

- Not easy for presenter to take information back to modify their own code

Current format

1. Help!
    - Clearly state problem the presenter would like to solve
    - Distribute code to pairs of lab members
    - Pairs work on problem
    - Diversity of solutions shared back to group and presenter
    - Ex: make code DRY, convert to tidyverse framework, go through issue tracker fixing code
2. Tutorial
    - Introduce new package or way of doing things
    - Start with brief presentation of package, workflow, book chapter
    - Break into pairs to work through exercises
    - Diversity of solutions shared back to group
    - Ex: base vs ggplot, implement make/snakemake workflows

Rule 1: Reciprocate respect (Group)

1. Sign up

- Establish a code of conduct

4. Check your attitude

    a. Be humble
    b. Be willing to incorporate feedback
    c. Ask clarifying questions

    d. Remember that this isn't a competition
5. Reciprocate respect
    a. Facilitation
        i. Lead the discussion
        ii. Have an action plan: Focus the conversation
        iii. Balance expertise in groups
        iv. Attempt to purposefully engage early learners
        v. Avoid letting (senior) individuals dominate
    b. Attitude:
        i. Both:
            1. Be humble
            2. Be willing to incorporate feedback
            3. Ask clarifying questions and note why you need the clarification
        ii. Participants:
            1. Remember that this isn't a competition
            2. Focus on the speaker's goals
            3. Avoid distractions and engage with the presenter
            4. Allow others to contribute
            5. Give feedback & coding suggestions at a novice explanation level
            6. Give compliments

Rule 2: Set specific goals (Presenter)

Preparation 5. Lead the discussion a. Have an action plan c. Focus the conversation 1. Set specific goals for each code review - Try to be as explicit and clear as possible for what you are asking the participants to do a. I have tried to be (what felt to me) painfully clear but I still was falling short making it clear to some people. b. People will always tend to notice or ask about things that are not the focus of the exercise, being specific about the goal may help. Also setting up a simplified scenario can be helpful (although requires much more prep time)

Rule 3: Keep it simple (Presenter)

2. Practice exercise
    a. When in doubt try it out! Run it by someone prior to lab meeting to make sure your request/goal is understandable
    b. Running on different systems will inevitably run into issues. Send out basic script ahead of time to try the dependencies/functions of focus
3. Create a *minimal working example* - Working directly with your own raw code can be challenging
    a. It is likely filled with code specific to you/your project
    b. It can take a lot of time to explain details and working of code
    c. Code can have multiple steps that are needed to run to get to script focus
    d. Using a whole script opens up the risk of getting off topic
4. Keep It Simple Silly (KISS)
    a. There is never enough time
        i. I typically have multiple activities planned, but rarely get through them all
        ii. Modularize the activity so that each can be its own and if accomplished you are able to move onto the next
    b. One activity per code review is typically sufficient
        i. This will take up most of the time
            1. Intro - 10 min
            2. Think/pair - 30 min
            3. Share in group - 10 min
            4. Debrief - 5 min
        ii. I have not been successful in trying to do more than that in different ways because it is going to be taking time away from one of the other steps

Rule 4: Give group time to prepare (Presenter)

3. Communicate with the group in advance so they can prepare
    a. Provide code chunk and necessary data
    b. Provide instructions for appropriate dependencies
    c. Provide use case (where available/appropriate)
    d. Specify goals & expectations

Rule 5: Prepare in advance (Audience)

1. Prepare
    a. Review the code in advance
    b. Download datasets, install packages
    c. Google what you don't understand

Read chapter, paper, etc.

Rule 6: Don't give busy work (Presenter)

2. For Help! format, select your code
    a. That you are willing to receive and incorporate feedback on
    b. That is digestible in an hour (<100 lines)
    c. Use "vanilla" R to check for dependencies on a "fresh" environment
    d. Prepare a small example dataset

Rule 7: Include all levels of learner (Presenter)

- Ask questions that are appropriate for different levels

- Be purposeful about how you pair people off
  - Use think/pair/share
  - Senior/Beginner
  - Senior/senior and Beginner/beginner
  - Self-selected
  - Navigator/driver with a switch midway through

Rule 8: Engage with the material (Audience)

3. Participate
   a. Focus on the speaker's goals
   b. Avoid distractions and engage with the presenter
   c. Allow others to contribute
   d. Ask clarifying questions and note why you need the clarification
   e. Give feedback & coding suggestions at a novice explanation level
   f. Give compliments

Rule 9: Let the material change you (Audience)

- Reflection
  - Use what you've learned to refactor your own code / improve new code

Rule 10: Speak up (Group)

- Speak up!
  - Topics - check interest level of potential topics - check group is at sufficient skill level needed - need for specific skills/methods in group - May need to circle back to "beginner" stuff with natural cycle of research group development

- What works and doesn't

Conclusion

Acknowledgements

References

https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004526

Figure 1. XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX. XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX XXXXX XXX.