Lecture 15 – Providing Critical Feedback

Learning Objectives:

3. Learn the basic principles of software design.

    3.4. Learn about optimization and profiling code.

Install packages: pryr

Discussion: Why do we give feedback?

For next slide: Read Waldrop-PeerReviews.rft.

- Which reviewer was most helpful?
- Which reviewer was least helpful?
- Pick out one example of useful criticism and one example of not useful criticism.

Discussion: What makes feedback constructive (or not)?

Qualities of Constructive Feedback

Qualities of **NON-**Constructive Feedback

Tips for Providing Constructive, Professional Feedback

- **Start with something positive**
- **Focus on the problem, not the person**
- **Avoid using second-person**
- **Be specific**
- **Provide solutions (if possible)**
- **Remember, your goal is to help!**

Tips for Receiving Feedback (Constructive or Otherwise)

- You are not your work
- Learn to recognize nonconstructive criticism
- Get back at bad feedback by getting better
- Bad feedback is not your fault (it’s the reviewer’s!)

Code Reviews

- Three students per team.

- Each will make notes and send to me.

- Together you will create a summary of two good points and two points where improvement is needed.

- Suggested areas to look:

  - Does the code run for you? If not, what are the errors? (Be sure to note what platform and R version you are using.

  - How does the code read to you? Was it easy to figure out or difficult? Do you think the code needs to be refactored? Does the documentation need to be improved?

  - How is the speed and memory usage? Profile if you wish. Is there anything glaring that could be improved? What's one thing that could help them improve speed and/or memory usage?

Having reviewed code, what should we be assessing?