Lecture 11 – Providing Critical Feedback

Learning Objectives:

4. **Produce code that is reproducible and produces results that are replicable.**

    4.6 Become proficient at giving and receiving critical feedback.

Discussion: Why do we give feedback?

To improve the person's work receiving the feedback

To gain a new perspective on your work

To improve the product

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 (see CodeReviews.md for assignments)

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

- Suggested areas to look:

    - Does the code run for you? Are there tests, and if so, do they pass? 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? Does the documentation need to be improved?

    - Do you think the code needs to be refactored? If so, where and for what purpose?

- Each group will make notes together and submit as the same review for each reviewee. (Each group member please submit separately or “done” as a comment. This is for Canvas bean counting.)

Having reviewed code, what should we be assessing?