Open source development and Rayleigh

Rene Gassmoeller, UC Davis
Reasons for open-source codes like Rayleigh

- Reduce duplication of efforts
- Use upstream improvements
- Community as force multiplier:
  - More diverse ideas
  - More eyes for the same problem
  - More robust testing
- Support the idea of open science, open data, open access
- Improve reproducibility
- Learn about new scientific and technical ideas
- Teach others what you know
- Meet future collaborators, employees, employers
- It's fun!

Ben Balter, Senior Product Manager at Github.com
Challenges for open-source projects

• Acknowledgment of contributions
• Interactions between community and software architecture
  • Software architecture determines size of a community
  • Work on architecture is crucial, but not sufficiently acknowledged
• Tradeoffs between competing project goals
  • Performance vs flexibility, individual interests need to be balanced
• Lack of software development skills
• Community, Leadership and Governance
  • A software project needs growth both horizontally (user base) and vertically (hierarchy and user engagement) to prevent burnout of maintainers and maintain influx of new users
  • New users need to feel welcome and introduced to the community

https://opensource.guide
Forming a community around Rayleigh

• Learn about how to modify the software
• Establish collaborations on short-term development tasks.
• Establish work flows for contributing to the repository.
• Establish a core-development team that is authorized to approve changes to the code moving forward.
• Establish long-term objectives for the continued development of Rayleigh.

https://opensource.guide
Forming a community around Rayleigh

• Get to know each other
• Create a place to discuss
• Share a common goal
• Form connections and capabilities
• Allow progress in responsibility

https://opensource.guide
How to get and give help

• Public discussions are better than private discussions (teaching, archival)
  • cig-geodyn@geodynamics.org
  • Github issues and pull requests (see Github tutorial later today)
• Follow a code of conduct, and be careful to create a welcoming community (I recommend reading: https://opensource.guide/building-community/)
• Create an atmosphere where it is not frightening to ask questions, and answering questions feels rewarding
• Adjust level of detail to the knowledge of the person contributing/asking, but teach to allow growth
• Create ways to acknowledge non-scientific (but crucial) development efforts:
  • E.g. set up an automatic newsletter or changelog that tells everyone who contributed something (I can help with that)
  • Create roles for people who help others (e.g. principal developer, core developer, ...)