Lab Carpentry Documentation

Release 0.1

DDD

Sep 28, 2017

Contents

Lab Carpentry provides management tools for academic research labs. We have extracted practices that we find useful, such as the construction of an onboarding document, and we provide templates of such practices.

Contents:

CHAPTER 1

Meta-Guidelines for Onboarding Document

- Simple: Use straightforward language and avoid jargon.
- Concise: The document should be readable in a single sitting.
- Specific: It should be targeted to the lab in question.
- Enforceable: Failure to meet expectations should have consequences.
- Living: The document is expected to change.
- Local Resources: (should this be part of a separate document?)

CHAPTER 2

Mission Statement

Core Purpose: What will we accomplish?

Core Values: What would we keep doing, even if it put us at a disadvantage?

Mission: Summarize the above into what your lab does and how it does it?

CHAPTER 3

Expectations

Code of Conduct: What rules are all lab members expected to follow?

Authorship: How do we confer authorship? What constitutes collaboration or work on a paper that results in authorship?

Project Ownership: When a project is developed in the lab, how does that project (from a social, rather than licensing/IP/technical perspective) grow or change if the primary developer leaves the lab?

Project Leadership: When software development is funded, how is the leadership direction established for projects developed in the lab? Is there a consensus-based approach, is there a strategy for governance, does it include external participants? How is leadership broken down between technical and social?

Ethics: How do we decide what is right and wrong / who lives and who dies, and how do we influence our decisions about projects, interactions, and scientific actions based on these ethics? This concerns things like scooping, being worried about scooping, actions with respect to software and data as it concerns "competitor" groups. When developing software, how are citations to other pieces of software handled? When writing papers, how are citations handled?

CHAPTER 4

Infrastructure Guide

Compute Resources: Where do we run stuff?

Shared Storage: Where do we save stuff?

Version Control: Where do we develop digital artifacts?

Accounts: How are accounts managed, and for what?

Tools: What tools do we use and how do we obtain them?

Purchasing: How do we buy stuff?

CHAPTER 5

Communication

Venue: Where do we communicate? Is it required? Ephemeral, archived, etc.

Detail Level: How often should lab members communicate with the PI, and what level of detail is expected in those communications? Are there guidelines for what types of problems to immediately escalate to the PI?

Project: What standards of communication should there be for interfacing between projects (which may have external contributors and participants) and lab members? For instance, is it acceptable etiquette to say, “I will come by your office and work through this with you”? Can external project participants influence development conducted by lab members? Does all software development need to be immediately upstreamed, or are there patterns for submitting development?

Social Media: Do we share our work? If so, how? Any restrictions?

CHAPTER 6

Coding and Software

Commenting: How should comments be included in code written for lab software, and to what standards? Should docstrings be included?

Coding Standards: How do we shape our active lines of code in form and function?

Using Other Code: Under what circumstances should code external to the lab be used? (i.e., both fundamental libraries like NumPy / LAPACK and higher-level codes that build on them) How should interactions between the lab and upstream communities be managed How should changes be contributed, which users should issue those changes (i.e., lab organizations or individual users)?

Licensing: What licenses (permissive, open source) should be used? For examples of non-copyleft licenses, BSD, MIT, X11 and Apache are commonly used. If copyleft licenses are to be used (which are sometimes discouraged by funding agencies, and patent provisions may interfere with institutional policy) are GPLv3, GPL2, GPL2+, MPL, AGPL to be used? For hybrid models, is LGPL appropriate?

Languages: What languages do we use in the lab? Under what circumstances? (i.e., "Python for most aspects of development, using C/C++ for underlying performance improvement, with Node.js for web apps.")

Data Management: When data is utilized by the lab, how is it stored? How are files named? Where would relational databases and their tables be stored? Is there a naming convention? When data is generated, how is it stored? How is access to data managed (i.e., is data on a filesystem accessible to all members of the lab? Is there an assumption of privacy?) Is data cataloged, are notes kept, and how is the cataloging system handled?

IP/Openness: Is there an assumption that development occurs in public, in private, or in a mixture?

Record Keeping: How do we record our actions? Are there specific places that the lab has access to, and is it something shared between lab members? Are notes open?

Reproducibility: How do we ensure our analyses are reproducible?

CHAPTER 7

How to Modify this Document

The skeleton repository for this document is at http://github.com/lab-carpentry/blueprint-onboarding/ . To make changes, fork the document, make the changes you wish to see to the template, and issue a pull request.

CHAPTER 8

How to Use this Document

You can copy and paste this into your own onboarding document. If you'd like to use this structure and the ReadTheDocs build system, you can fork the repository, and in a branch with your lab name, extend the text (and add text as necessary) where appropriate.