Preview Questions

1. What are the computing tools and techniques that every researcher should adopt?
2. What activities comprise data management?
3. What is the best way to use software in an analysis?
4. How do I make it easy for new collaborators to come up to speed and contribute to a project?
5. How should I track changes for a research project?
6. What’s the recommended way for writing manuscripts?

Summary Notes

*Data Management*

* Document intermediate data and the process of preparing ready-to-analyze data.
* Strive for “tidy data.”
* Save the raw data.
* Back up the raw data in more than one location.
* Create data that facilitates machine readability.
  + Non-proprietary file types
  + Descriptive (meaningful) names
  + File names that include metadata
* Create data that is analysis-friendly.
  + One variable per column
  + One observation per row
* Record all steps used to process data.
* Represent unique identifiers for every record using the same format across multiple tables.
* Submit data to a reputable DOI-issuing (Digital Object Identifier) repository to facilitate sharing.

*Software*

* Adopting certain practices will make it easier for others to understand and reuse your code.
* At the beginning of every program, include a brief explanatory comment that includes at least one example.
* Decompose programs into functions.
* Eliminate duplicate wherever possible.
* Give functions and variables meaningful names.
* Meaningful variable names are no harder to use than terse names because of tab completion.
* Explicitly state dependencies and requirements for your program in a README file.
* DO NOT control program behavior by commenting and uncommenting sections of code.
* Provide a simple test data set so that users can check to make sure the program is working.
* Submit code to a reputable DOI-issuing repository to facilitate sharing.

*Collaboration*

* Collaborators could be other people or your future-self returning to a project after some period of time.
* Explain the purpose of your project in a README file.
* Create a CONTRIBUTING file explaining what people need to do to contribute to the project.
* Create a shared plaintext NOTES file that describes project tasks that need to be done.
* Decide how project members will communicate with each other and external users and collaborators.
* Create a LICENSE file in the project home directory that explicitly states the licenses that apply to the project’s software, data, and manuscripts.
  + The default is “all rights reserved” if nothing is stated.
  + Including “no commercial use” language can create unintended impediments to sharing and collaboration.
  + Permissive software licenses are easier to integrate into projects than the GNU General Public License (GPL).
* Include a CITATION file in the project home directory to easily communicate citation information.

*Project Organization*

* Put each project in its own directory.
* Put text documents in the doc directory.
* Put raw data and metadata the data directory.
* Put files generated during cleanup and analysis in the analysis directory.
* Put project source code in the src directory.
  + Subdirectory for core analysis
  + Subdirectory for controller or driver scripts that contain all the analysis steps
* Put compiled programs (executable programs compiled from code) in a bin directory.
  + Scripts are executed directly as is (i.e., code that is edited directly).
  + Programs are explicitly compiled before being used (i.e., code that is not edited directly).
  + Be consistent with the placement of directly executed scripts brought in from the outside.
* Name files to reflect their content and function.

*Keeping Track of Changes*

* Being able to reference specific versions of a project helps with publication, responding to reviewer comments, and providing support information to reviewers, editors, and readers.
* Recommended version control systems Git, Mercurial, and Subversion.
* Back up everything created as soon as it is created.
* Keep changes small.
  + Rule of thumb: a group of edits that you would want to undo in one step at some point in the future.
* Share changes frequently.
* Create and maintain a checklist for saving and sharing changes to the project.
* Store each project in a folder that is mirrored off the researcher’s working machine.
  + Synchronize daily.
* For manual versioning
  + Add a file called CHANGELOG.txt to the project’s docs subfolder.
  + Make dated notes in the CHANGLOG file in reverse chronological order.
  + Copy the entire project whenever a significant amount of change has occurred and store a copy in a subfolder whose name reflects the data.
* For version control systems
  + Use plain text files.
  + Raw data should not require version tracking because it shouldn’t change.
  + DO NOT include large data or results files (e.g., files over 100 MB).

*Manuscripts*

* Mirror good practices for managing software and data to make writing manuscripts scalable, collaborative, and reproducible.
* Use a single master document that is available to all coauthors at all times.
* All writers should agree on the workflow before the process starts.

*What Was Not Covered*

* Branches allow multiple independent changes to a project.
* Small projects can make use of shell scripts that rerun everything rather than build tools to regenerate data and entire papers when one or more raw input files change.
* Unit tests are small tests of one particular feature of a piece of software; they ensure that a change to one part of the software doesn’t break other parts (i.e., regression).
* Coverage is the set of lines that are (or aren’t) actually executed when a test case is run.
* Continuous integration tools automatically run a set of user-defined command whenever changes are made to a version control repository to make sure the software hasn’t regressed.
* Profiling is measuring where a program spends its time as is the first step in tuning the program to make it run faster.
* The semantic web.
* People don’t write comprehensive documentation unless they have collaborators who will use it.
* A bibliography manager or reference manager is used by researchers to identify themselves in their publications.
* Code reviews and pair programming.

Additional Questions of Interest

1. None