**Nov 14**

**Notes about project structuring**

* Pick a naming convention and stick with it. Do not mix them. Perhaps consider writing your own style guide.
* Use functions whenever you can, and keep them short.
* Use **packrat** to manage dependencies to keep things consistent and reproducible.
* Versioning systems like git keep you from holding onto old versions out of fear of needing them in the future. Use versioning software.

https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000424

In practice, therefore, the scripts that I write tend to fall into these four categories:

1. **Driver script.** This is a top-level script; hence, each directory contains only one or two scripts of this type.
2. **Single-use script.** This is a simple script designed for a single use. For example, the script might convert an arbitrarily formatted file associated with this project into a format used by some of your existing scripts. This type of script resides in the same directory as the driver script that calls it.
3. **Project-specific script.** This type of script provides a generic functionality used by multiple experiments within the given project. I typically store such scripts in a directory immediately below the project root directory (e.g., the msms/bin/parse-sqt.py file in [Figure 1](https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000424#pcbi-1000424-g001)).
4. **Multi-project script.** Some functionality is generic enough to be useful across many projects. I maintain a set of these generic scripts, which perform functions such as extracting specified sequences from a FASTA file, generating an ROC curve, splitting a file for *n*-fold cross-validation, etc.

* I’m interested to try writing my manuscript as an r markdown file. One of the biggest barriers seems to be sharing with collaborators, but apparently rmd files can be knit as a word document: https://rmarkdown.rstudio.com/articles\_docx.html

**Setting up Git**

Links from the Software carpentry collaborate document

**Resources**

<https://swcarpentry.github.io/git-novice/reference>

<https://www.atlassian.com/git>

**Git cheat sheets**

<https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet>

<https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>

Following the first link

* Set user name, email, line endings
* Come back to setting the default text editor if it’s not what I like when I try it

-I can click on a folder and say Git Bash here to start in that directory

-Could use GUI or Bash. I think for now I will use Bash since that’s what I learned before and what I know I have good instructions for. Maybe when I’m ready I can write down a list of the commands that I would use regularly when committing etc

-Created a .gitignore file. Set it up to ignore everything that is in the “data” folder

-what adding all of the files that I had in the repository (at this point, just folders, r project files, and the git ignore file), I got the following warning message from git bash. I am going to ignore it based on what I read on the internet.

warning: LF will be replaced by CRLF in .Rhistory.

The file will have its original line endings in your working directory