Reference notes

# Basic Git workflow

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

--- open git bash on desired folder (i.e. Version 2 - Manuscript)

> git status

> git add .

> git commit –m “<commit messge goes here>”

> git push origin master

> nano file.txt } open in a text editor

> cd . } change directory to parent folder?

> ls -a } list all contents of the folder, -a show all including hidden folders

# PRISM Questions

## Don’t understand

* What is there difference between the columns Start\_time\_1, and Start\_time\_1.1?
* What is the difference between quality and quality\_old columns?
* What are quality codes 5,6,7,8,NA?
* Why are there 4 different GIS\_UTM\_Zones? (395 have different values in some of the columns, 2512 have NAs)
* How can final\_centroids have non-discrete numbers? Why are there so many NAs?
  + This probably only applies to intensive plots

## Weird things to deal with

* UTM\_x\_Type: Most of these are the SW corner, but a bunch of them are other corners, centre, perimeter points, NA etc

## Errors

## Analysis

* How should we deal with the plots that weren’t randomly selected? What were the reasons that they weren’t? When would they bias the results or not?
* Best way to approach spatial data
  + Do the UTMS in the spreadsheet match up with the GIS data the Tyler created? How to get the actual SW corner and appropriate corners? Might be better to entirely ignore the spatial data in the spreadsheet

Lab notebook

# 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

-set up and connected folder to a github repository

# Nov 15, 2019

## Setting up Git

-added a README file

-did another commit/push of edits

-created the reference notes at the beginning of this document

# Nov 19, 2019

## Preparing PRISM data

-added the new version of the raw prism data to data folder: unedited excel file and CSV file with the top two rows that were full of Jennie and Isabelle’s comments removed.

-Laurent gave me his script for cleaning the PRISM data

-created a library script for attaching packages, user defined functions, defining objects that will be used regularly such as coordinate reference systems

-started working on a prism data cleaning script, based on scripts Laurent and I have written already

# Nov 20, 2019

## Preparing PRISM data

-started a list of things that I don’t understand about the PRISM data, or errors, @ top of this document

-looking through columns I might want to find inconsistencies/surprises (recorded above)

# Nov 21, 2019

TO DO: add files starting with ~ to my git ignore file, untrack and delete the the temp file of workbook from by directory

-make a plan of what I want to accomplish over the next week, break down into steps