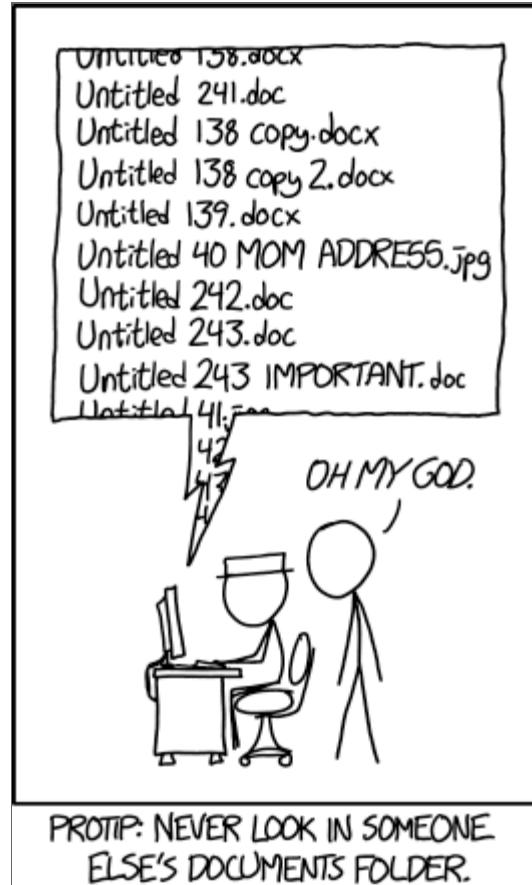


Good Data Habits

Starting as you mean to go on

29-Sep-2025



File naming conventions

Organizing files

Which of these files contains the most recent version of the data?

```
1 $ ls -l data/
2 -rw-r--r--. 1 hannah hannah 0 Sep 23 11:38 sample_metadata_clean.tsv
3 -rw-r--r--. 1 hannah hannah 0 Sep 23 11:37 sample_metadata.tsv
4 -rw-r--r--. 1 hannah hannah 0 Sep 23 11:38 sample_metadata_USE_THIS_ONE.tsv
5 -rw-r--r--. 1 hannah hannah 0 Sep 23 11:38 sample_metadataV2_final.tsv
6 -rw-r--r--. 1 hannah hannah 0 Sep 23 11:38 sample_metadataV2.tsv
```

File naming conventions

What makes a file name useful? **Metadata**

- Project name or acronym
- Study title
- Location
- Data type
- Researcher initials
- Date
- Data stage (raw, filtered, etc.)
- Version number
- File type
- If script, names should describe what they do

Task of the day:

- ▶ rename 3 files

Directory structures - Get organized!

Last time on 10-minute data science...

We discussed file naming conventions. What were folks' take-aways?

Directory Structures

Where should you look to find the latest version of protocol you're interested in testing?

Our lab's sharepoint is a good example of what not to do...

Directory Structures

Which enzyme assay is the one you want?

Best practices

- Choose an organizational style; stick with it
- If sharing, document the organizational style
- Divide work into **project directories**.
 - Thesis
 - Chapters
 - Papers
 - Sections
 - Grants
 - subprojects or papers

Take home: Each project directory should be **self-contained** and hold all files needed to go from raw data to final results

Subdirectory choice

What subdirectories do folks use?

What questions should you ask when creating a new subdirectory?

Example 1: ARCSS Grant

- A project I joined when I started working here
- Organized around anything relevant to the grant
- Includes both sub-“project” directories, but also writing, administrative information, literature
- sub-projects are tracked with version control software, but not this directory

Conferences/	Conference presentations, travel administrative documents
Sean_qsip_tree/	Project file for creating a phylogenetic tree with Sean's qSIP project
Literature/	Relevant literature for ARCSS project (automatically integrated into Zotero/Mendeley libraries)
Senescence/	Project to identify likely senescence times for our sites
mimics_webapp/	Project for Stuart's hairbrained (but genius idea) to turn MIMICS into a webapp
Picarro Code/	Nacent code for processing Picarro outputs
useful_images/	Helpful images related to the project. Often useful in creating figures or presentations
Protocols/	Protocols related to lab work
Writing/	Writing folder; includes derived grants, manuscripts, etc.
qsip/	FICUS qsip project

Example 2: The temporal paper

- Self-contained project
- highly collaborative; structure is co-created with others
- designed to be tracked with version control software from day 1

Assembly-analysis/
cazyme_scraper/
pipeline
CN_versatility/
Core_microbiome/
data/
contains both shortcuts to large data sets and actual files
general_climate_weather/
GraftM-analysis/
in here
identifying-outlier-years/
identify-temp-WTD-responders/
Metabolic-analysis/
in here
metadata_availability/
quantify_stability_with_time_figure/
SingleM-analysis/
setup.R
cleaning
README.md
your own computer
temporal_paper.yml

Sub-analyses; files contain code, outputs, figures
Shortcut to a different project file, where I wrote a code
Sub-analyses; files contain code, outputs, figures
Sub-analyses; files contain code, outputs, figures
Raw data; files never edited; common across collaborators;
Sub-analyses; files contain code, outputs, figures
Collaborators's sub-analyses; I don't have to edit anything
Sub-analyses; files contain code, outputs, figures
Sub-analyses; files contain code, outputs, figures
Collaborators's sub-analyses; I don't have to edit anything
Sub-analyses; files contain code, outputs, figures
Sub-analyses; files contain code, outputs, figures
Sub-analyses; files contain code, outputs, figures
Common analysis script that takes raw data and does initial
Readme file; describes how to setup the code and data on
Contains instructions for installing the software necessary

Example 3: The dada2 pipeline

- Purpose: Tutorial/ pipeline
- Doesn't have unique raw data
- Output folders generated by code
- Emphasis on portability to other computers

```
R/           Rscripts live here - they include documentation in the form of R-markdown  
slurm/       slurm scripts for submitting to supercomputer live here  
dada2_ernakovich.yml Installation and software information  
README.md/   Tutorial information
```

Not sure which is best? Templates exist!

Heidi Seibold's Research Project Template

```
.  
├── README.md  
├── analysis          <- all things data analysis  
│   └── src             <- functions and other source files  
├── comm                
│   ├── internal_comm  <- internal communication such as meeting notes  
│   └── journal_comm    <- communication with the journal, e.g. peer review  
├── data                 
│   ├── data_clean      <- clean version of the data  
│   └── data_raw        <- raw data (don't touch)  
├── dissemination        
│   ├── manuscripts       
│   ├── posters           
│   └── presentations     
├── documentation      <- documentation, e.g. data management plan  
└── misc                <- miscellaneous files that don't fit elsewhere
```

Taking project folders to the next level

Project folders allow you to take advantage of coding and project management tools

- Most IDEs (Integrated Development Environments, e.g. Rstudio) are set up to allow users to work in and switch easily between projects
- RStudio projects
- git version tracking - For tracking your code and files, you set up version tracking in a project folder.
- Sharing a project is easy - simply share the project folder with the collaborator

Mark calendars for Ernakovich Lab Discussion

lab meeting on 10/27/25: . . .

- Determine organization norms
- Reorganize Ernakovich Sharepoint
- Create a Guide ('readme') for directories

Metadata (aka readmes)

Last time on 10-minute data science...

We discussed directory structures

What is metadata?

What should metadata include?

- Units
- Resolution
- Meaning of column names
- Description of caveats, issues, or missing values
- How data was collected
- Filtering or processing steps the data has been through (if applicable)

What do you do if you don't know what kind of metadata to include?

- look it up (many data types have standards)
- MIMARKS (Minimum information about a marker gene sequence)
- MIMAGS (Minimum information about a metagenome-assembled genome) for microbial genomics data)
- phone a friend

Weekly and Daily Checklists

Last time on 10-minute data science...

We discussed metadata and README files

What are some habits you have at work?

- checking email
- wearing gloves when handling chemicals
- maintaining a lab notebook

Establishing Good Data Habits

Good data habits can be implemented regardless of your experience or computational skill level

Today we'll go through some check-lists you can use to help cultivate good data and coding habits

When starting a project

- Create a dedicated **project directory**, named for the project
- Create **subdirectories** for data, analysis, and documentation
- Create a “**README**” document
- Decide how you will **record** your decisions and data analysis process
(e.g. commenting, an electronic lab notebook, documentation, or all of the above)
- Decide how you will **keep track of changes** (and document that decision in the README)
- Decide how you will **name your files** (and document that decision in the README)
- Decide how you will **backup changes** to the project directory [store it in >1 place, ideally mirrored]

When you receive (or collect) data

- Save at least **2 copies** of the data in more than one place
- Create a **metadata file**, (data about the data) store it with the data
- The metadata includes information about **how** the data was obtained and **who** is responsible for it
- The data is stored and organized in an **analysis-friendly** way
- Look at your raw data** before any processing
- Perform **quality control first**. Before analysis (and document what quality control you performed)

When beginning to analyze data

- Store all analysis scripts and code in the project directory
- Explain the purpose of each script/code/software at the top of the file
- Use comments or other “self-documenting” code practices
(e.g. rmarkdown, jupyter, quarto notebooks)
- Pick a **code style guide** and stick to it
- Use meaningful variable and function names
- Create an outline (pseudocode) of analyses to break analysis into smaller steps
- Ruthlessly resist duplication** (avoid copy and pasting code)
- Try to write in functions, and scripts, rather than one, long piece of code

At the end of the day

- Back up data; (never overwrite originals!)
- Save data, code, figures, etc. to appropriate locations in the project directory (not “Downloads”, or “Desktop”)
- Check that all new files have clear, consistent names, following your conventions
- Move temporary files, scripts, and other analyses you’re unsure you’ll keep to a “sandbox” folder
- Briefly (2-3 bullets) document the day’s work in your lab notebook, readme, change log, etc.
- Write down your next steps for tomorrow; including any errors you may have left un-addressed
- if using version control software, “commit” your changes with a message

Quality Assurance

Exploring your data

Speaker notes