



# Project Organization

Childhood Cancer Data Lab

# Sources!

All ideas come from somewhere.

These are some sources that have inspired us (and provided material directly)

- Vince Buffalo: [Bioinformatics Data Skills](#)
- Jenny Bryan: <https://speakerdeck.com/jennybc/how-to-name-files>
- Danielle Navarro: <https://slides.djnavarro.net/project-structure>

# Why does project organization matter?

- Finding things takes a lot of time and effort
- Standard and predictable organization saves time
- Be kind to yourself & others!
  - Make your stuff discoverable
  - Follow consistent patterns



# But I can just search!

- Google has trained us to search for content, and searching is great! Sometimes.

```
@071112_SLXA-EAS1_s_7:5:1:817:345
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC
+071112_SLXA-EAS1_s_7:5:1:817:345
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
@071112_SLXA-EAS1_s_7:5:1:801:338
G TTCAGGGATACGACGTTTGTATTTTAAGAATCTGA
+071112_SLXA-EAS1_s_7:5:1:801:338
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII6IBI
```

- File names can be uninformative (we'll come back to that!)
- Data files often don't have searchable content
  - Even when they do, you might not know what to search for!
- **Metadata** describing the content might not be part of the file



REPORT

# FILE NOT FOUND

*A generation that grew up with Google is forcing professors to rethink their lesson plans*

By **Monica Chin** | **@mcsquared96** | Sep 22, 2021, 8:00am EDT

*Illustrations by Micha Huigen*

<https://www.theverge.com/22684730/students-file-folder-directory-structure-education-gen-z>


# Where to start?

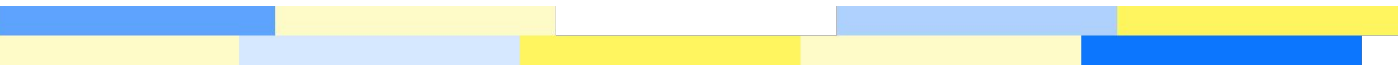
- **Use Folders/Directories!**
- Keep separate projects separated
- Separate sections for units within a project
  - Data
  - Code
  - Results
  - Reports
- **Documentation throughout!**
  - Describe what files do and how they are organized

*ABD*



# Relative vs. absolute paths

- The path is the address to a file or directory on your computer. There are two ways to formulate paths:
  - The **absolute (or full) path** represents the path of a given directory or file, *beginning at the root directory of the computer*
    - Because they begin at the root, absolute paths always start with **/**
  - The **relative path** represents the path of a given directory or file, beginning at (i.e. *relative to*) to the working/current directory 

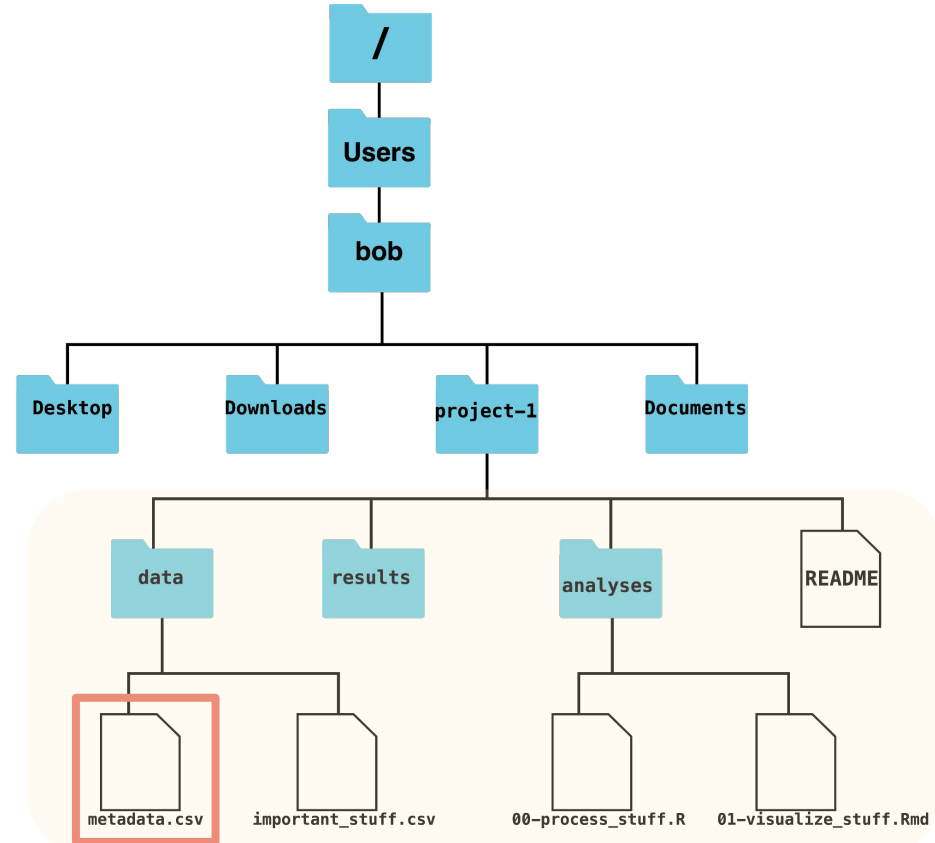
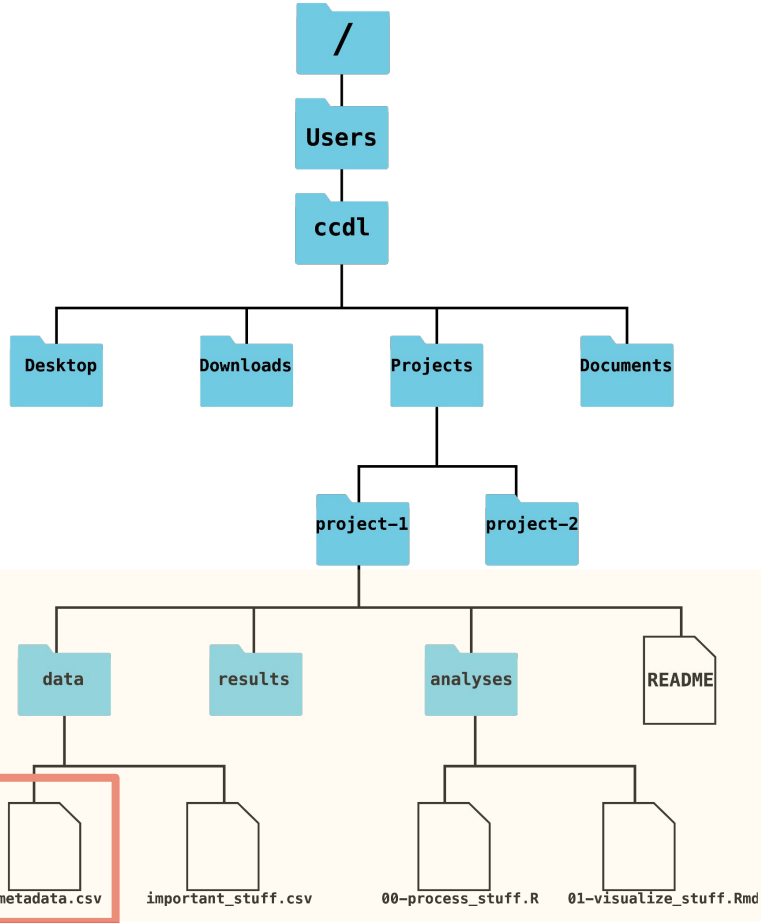


We prefer relative paths! Why do you think?





Consider each computer below. What is the **absolute path** to **metadata.csv**?  
What is the **relative path** to **metadata.csv** from **analyses**?

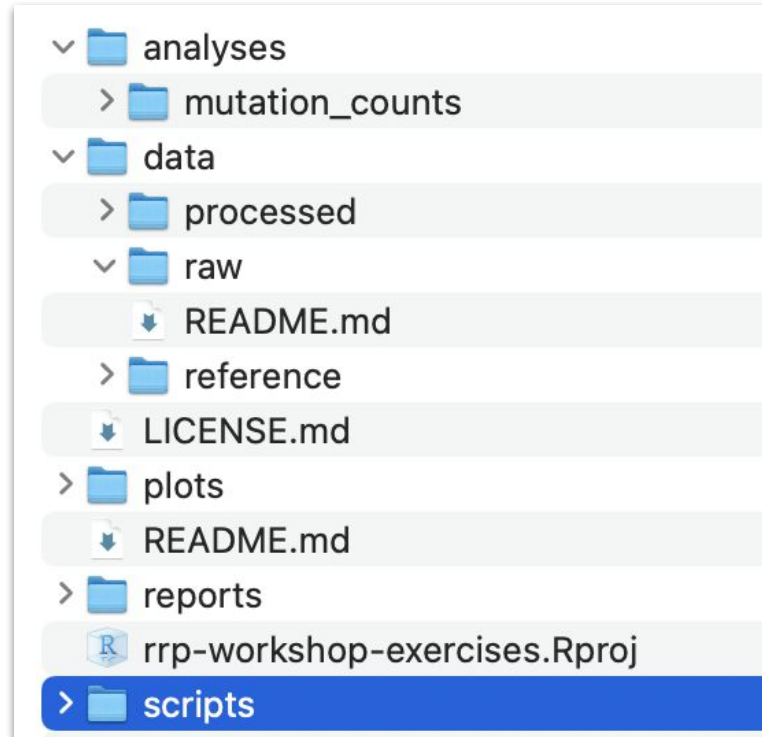


# Tools for path consistency

Defining paths in code and documentation relative to the base directory of a Project (or other fixed points in the project) allows for portable consistency

- In R
  - `here::here()` and the `{rprojroot}` package it is based on are good tools for finding the project root
- In Python:
  - Using the `gitpython` package, find the root of the git repository with:  
`git.Repo(".", search_parent_directories=True).working_dir`
- In Bash:
  - We often define paths relative to the script itself with:  
`cd "$(dirname "${BASH_SOURCE[0]}")"`

# A typical project folder (for me!)



You'll see more of this project later!

# The **data** folder

- This is where the big files go
- Often contains a **raw** subfolder
  - Files that came from external sources, untouched
- Maybe a separate subfolder or subfolders for **processed** files
  - trimmed, filtered, concatenated, etc.
- Use sub-subdirectories for organization:
  - by processing stage
  - by date
  - by sample
  - *Consider:* it is often easiest to process **all** the things in a folder together; organize by units of work

Spend time thinking about this organization! It will reward you later.



# The Naming of [Files] is a difficult matter

(apologies to T. S. Eliot)

Some “good” file names:

- `cheese-ratings.tsv`
- `2022-02-12_cheeseshop-inventory.tsv`
- `01_compile-ratings.py`

Some “not so good” file names:

- `script.py`
- `AVRS638GVEW4.fastq.gz`
- `tastingnotes (3).docx`
- `My favorite cheeses FINAL3 for Anatole UPDATED.docx`



image from *Anatole* by Eve Titus  
illustrated by Paul Galdone

# Jenny Bryan's principles for file naming (modified)

- machine friendly
- human friendly
- sortable and computable



# Machine friendly

- Avoid spaces
  - Old computer systems get confused by spaces
  - All computer systems are old underneath
  - Use underscores or dashes to separate words instead
- Use “standard” characters:
  - Letters, numbers, underscores, and dashes
  - Periods only for file extensions (`.txt`, `.tsv`, `.R`, `.tar.gz`)
  - Many characters have special meanings in code. Avoid them! (e.g. `*` `+` `?` `|` `$` `/` `"` )
- Be consistent with case
  - Don't *assume* case has meaning: on some systems it does, and on some it doesn't
  - But always *act* as if it does!
    - Never have two files that are the same but for case

# Human friendly

- Names should contain information about file content
- Short names are tempting, but you may regret choosing them!
  - **01.R**
  - **data.txt**
  - **tests.py**
- Use long descriptive names
  - **01\_download-ena-data.sh**
  - **fig01\_penguin-weight-histogram.png**

Which files do you want to look for before a deadline?

Which files do you want to get from your collaborator?





# Sortable

- Use numbers for consistent sorting
  - `fig01_project-overview.pdf`
  - `fig02_sample-descriptor-histogram.pdf`
  - `fig03_oncoprint.md`
  - Left pad with `0` for consistent number length; this helps the computer sort properly
    - `7` is sad when it gets sorted after `11`
- Dates: use ISO 8601
  - Year-month-day is unambiguous and sorts nicely!
  - `2000-05-04_jedi-council-attendance.tsv`
  - `2000-05-05_sith-council-attendance.tsv`

## PUBLIC SERVICE ANNOUNCEMENT:

OUR DIFFERENT WAYS OF WRITING DATES AS NUMBERS CAN LEAD TO ONLINE CONFUSION. THAT'S WHY IN 1988 ISO SET A GLOBAL STANDARD NUMERIC DATE FORMAT.

THIS IS ***THE*** CORRECT WAY TO WRITE NUMERIC DATES:

**2013-02-27**

THE FOLLOWING FORMATS ARE THEREFORE DISCOURAGED:

02/27/2013 02/27/13 27/02/2013 27/02/13  
20130227 2013.02.27 27.02.13 27-02-13  
27.2.13 2013.II.27.  $2\frac{7}{2}$ -13 2013.158904109  
MMXIII-II-XXVII MMXIII <sup>LVII</sup>/<sub>CCCLXV</sub> 1330300800  
 $((3+3)\times(111+1)-1)\times3/3-1/3^3$  2013 hisss  
10/11011/1101 02/27/20/13 0 1 2 3 7 2-27-13  
5 67 8

# Computable

- Use consistent name formats
  - Use file extensions
  - Separate “chunks” with underscores & keep consistent order
- Wildcards will be your friend:
  - `*` is the most common wildcard in UNIX
  - `*.txt`: refers to all files that end with `.txt` (hopefully all text files)
  - `2020-01-*`: all of the files from January 2020

# Files you didn't create

- All the guidelines and suggestions for file names are great for files you create, but sometimes files come from other sources
  - If you are lucky, they will follow nice conventions! 🎉
  - but often they won't 😞
- To rename or not to rename, that is the question
  - Leaving the name as it was sent can make it easier to track in correspondence
  - Reasons to rename:
    - uninformative generic names: `data.txt`
    - add source or date information
    - converting spaces or other special characters (but try to write code that can handle these!)
  - *If you choose to rename, do it with a script and document the original name and source*