

Project and data management: Tips for the ~~lazy~~ industrious researcher

Ross Markello
Network Neuroscience Lab

Advanced Analytics for Neuroscience
IPN Summer School 2021
28 June 2021

The norm in project management



DATA ORGANIZATION TIP:
JUST GIVE UP.

**Your closest
collaborator is you six
months ago, but you
don't reply to emails**

Overview

1. Project + data management standards

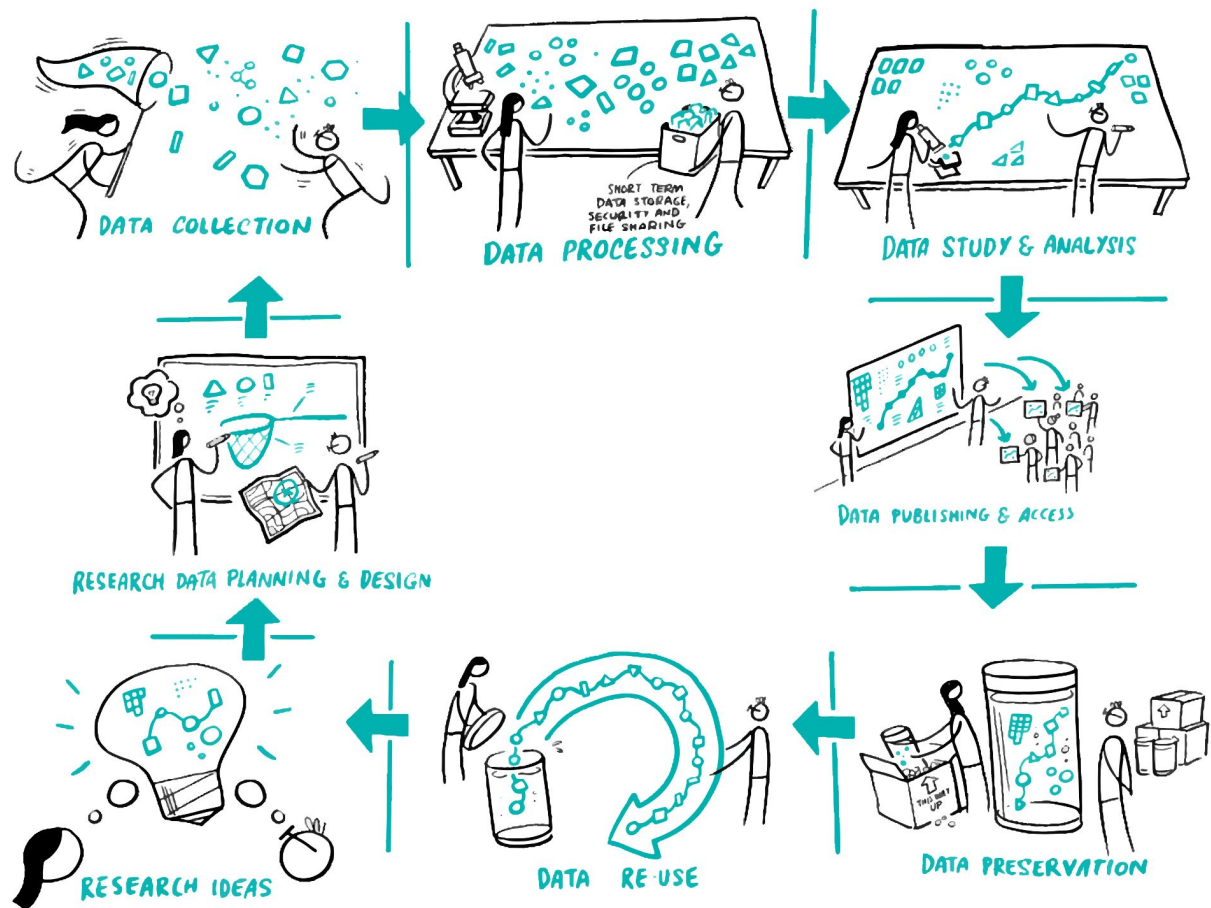
- Data standards: BIDS and NWB
- Project standards: Templating with cookiecutter

2. git + GitHub: The single-user ("selfish") tutorial

- Using git for version control
- Backing up code with GitHub

Project + data management

Standards are helpful
(probably)



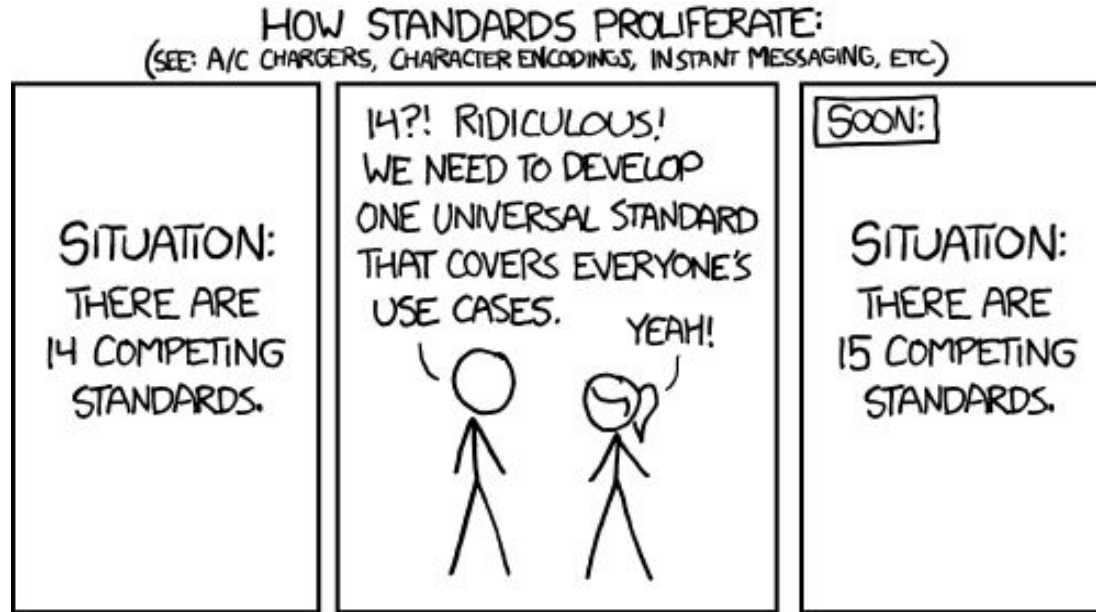
Scriberia

The Reproducibility Project

- Collaboration of 270 authors to repeat 100 published psychological studies
- Didn't have fully standardized project organization !

| Name ^ v | Modified ^ v |
|--|--|
| Replication of Correll (2008, JPSP, Stu... | |
| - OSF Storage (United States) | |
| ReplicationReport_Correll2008... | 2013-12-11 11:5... |
| ReplicationReport_incl_Metho... | 2013-12-11 11:5... |
| + Data | Name ^ v Modified ^ v |
| + Study Materials | Replication of Bressan & Stranieri (2... |
| + Analysis Audit | - OSF Storage (United States) |
| + Independent Direct Replication #... | BRESSAN COMMENTARY.docx 2015-06-08 08:5... |
| | + Study Materials |
| | + Cleaning & Analysis Scripts |
| | + Dataset |
| | + Replication Reports |
| | + Analysis Audit |

Let's make a standard!



Introducing:
Community-based
project + data
management standards

Community-based standards

- Developed openly
- Strive for consensus in decision-making
- Designed to empower and equip community members

The Brain Imaging Data Structure (BIDS)



Brain Imaging Data Structure v1.6.0



Search

Brain Imaging Data Structure v1.6.0

The BIDS Specification >

The BIDS Starter Kit >

The Brain Imaging Data Structure

This resource defines the Brain Imaging Data Structure (BIDS) specification, including the core specification as well as many modality-specific extensions.

To get started, [check out the introduction](#). If you'd like more information on how to adapt your own datasets to match the BIDS specification, we recommend exploring the [bids-specification starter kit](#).

For an overview of the BIDS ecosystem, visit the [BIDS homepage](#). The entire specification can also be [downloaded as PDF](#).

Table of contents

BIDS: Example layout

```
[rmarkello-home: ds001]$ tree
.
├── CHANGES
├── dataset_description.json
├── participants.json
├── participants.tsv
├── README
├── sub-01
│   ├── anat
│   │   ├── sub-01_inplaneT2.nii.gz
│   │   └── sub-01_T1w.nii.gz
│   └── func
│       ├── sub-01_task-balloonanalogrisktask_run-01_bold.nii.gz
│       ├── sub-01_task-balloonanalogrisktask_run-01_events.tsv
│       ├── sub-01_task-balloonanalogrisktask_run-02_bold.nii.gz
│       ├── sub-01_task-balloonanalogrisktask_run-02_events.tsv
│       ├── sub-01_task-balloonanalogrisktask_run-03_bold.nii.gz
│       └── sub-01_task-balloonanalogrisktask_run-03_events.tsv
├── sub-02
│   ├── anat
│   │   ├── sub-02_inplaneT2.nii.gz
│   │   └── sub-02_T1w.nii.gz
│   └── func
│       ├── sub-02_task-balloonanalogrisktask_run-01_bold.nii.gz
│       ├── sub-02_task-balloonanalogrisktask_run-01_events.tsv
│       ├── sub-02_task-balloonanalogrisktask_run-02_bold.nii.gz
│       ├── sub-02_task-balloonanalogrisktask_run-02_events.tsv
│       ├── sub-02_task-balloonanalogrisktask_run-03_bold.nii.gz
│       └── sub-02_task-balloonanalogrisktask_run-03_events.tsv
├── sub-03
│   ├── anat
│   │   ├── sub-03_inplaneT2.nii.gz
│   │   └── sub-03_T1w.nii.gz
│   └── func
│       ├── sub-03_task-balloonanalogrisktask_run-01_bold.nii.gz
│       ├── sub-03_task-balloonanalogrisktask_run-01_events.tsv
│       ├── sub-03_task-balloonanalogrisktask_run-02_bold.nii.gz
│       ├── sub-03_task-balloonanalogrisktask_run-02_events.tsv
│       ├── sub-03_task-balloonanalogrisktask_run-03_bold.nii.gz
│       └── sub-03_task-balloonanalogrisktask_run-03_events.tsv
└── task-balloonanalogrisktask_bold.json


9 directories, 30 files
```

Converting to BIDS

- Your "raw" data aren't going to be in BIDS format
 - Automated conversion tools are available (e.g., [HeuDiConv](#) and [BIDScoin](#))
- BIDS now supports additional modalities, including:
 - MEG, PET, ASL, iEEG, multi-echo MRI, genetics data, etc
- The [BIDS Starter Kit](#) can help get you set up and started!

Neurodata Without Borders (NWB)



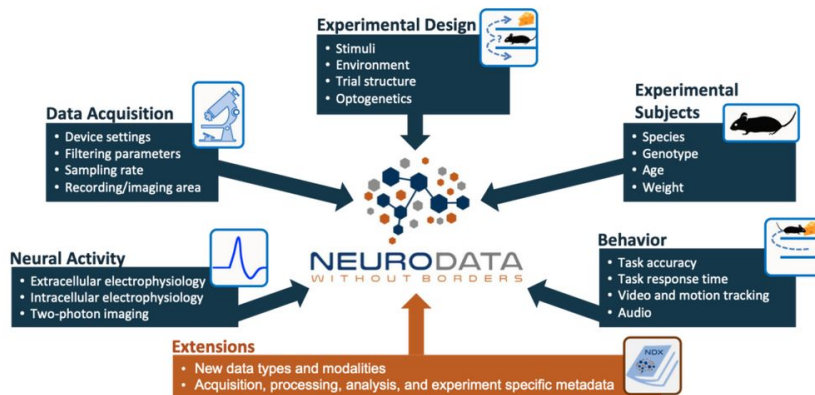
ABOUT COMMUNITY USING NWB DEVELOPER RESOURCES 

The NWB:N Data Standard

NWB: Neurophysiology

Unlike in other fields (i.e. genetics and cell biology), neuroscience does not have a standardized way to collect and share the wealth of existing data among researchers. The lack of a common format has made comparison across laboratories difficult and replication of specific experiments almost impossible, significantly slowing overall progress in the field.

Neurodata Without Borders: Neurophysiology (NWB:N) is a data standard for neurophysiology, providing neuroscientists with a common standard to share, archive, use, and build common analysis tools for neurophysiology data. NWB:N is designed to store a variety of neurophysiology data, including data from intracellular and extracellular electrophysiology experiments, data from optical physiology experiments, and tracking and stimulus data. The project includes not only the NWB format, but also a broad range of [software](#) for data standardization and application programming interfaces (APIs) for reading and writing the data as well as high-value [data sets](#) that have been translated into the NWB data standard.



The NWB way

- All NWB data live within a structured HDF5 file
- Different stores for "raw" time series data + processed data
- Automatic data validation using [PyNWB](#) or [MatNWB](#)
- More fun with [DANDI](#)

- `ElectricalSeries`
 - `SpikeEventSeries`
- `AnnotationSeries`
- `AbstractFeatureSeries`
- `ImageSeries`
 - `ImageMaskSeries`
 - `OpticalSeries`
 - `TwoPhotonSeries`
- `IndexSeries`
- `IntervalSeries`
- `OptogeneticSeries`
- `PatchClampSeries`
 - `CurrentClampSeries`
 - `IZeroClampSeries`
 - `CurrentClampStimulusSeries`
 - `VoltageClampSeries`
 - `VoltageClampStimulusSeries`
- `RoiResponseSeries`
- `SpatialSeries`
- `BehavioralEpochs`
- `BehavioralEvents`
- `BehavioralTimeSeries`
- `CompassDirection`
- `DfOverF`
- `EventDetection`
- `EventWaveform`
- `EyeTracking`
- `FeatureExtraction`
- `FilteredEphys`
- `Fluorescence`
- `ImageSegmentation`
- `ImagingRetinotopy`
- `LFP`
- `MotionCorrection`
- `Position`

Organizing with cookiecutter

- `cookiecutter`: tool for creating projects from pre-specified templates
 - Available for all types of projects
- Neuroscience projects as a form of "data science" projects

```
├── LICENSE
├── Makefile
├── README.md
├── data
│   ├── external
│   ├── interim
│   ├── processed
│   └── raw
├── docs
├── models
├── notebooks
├── references
├── reports
│   └── figures
├── requirements.txt
├── setup.py
├── src
│   ├── __init__.py
│   ├── data
│   │   └── make_dataset.py
│   ├── features
│   │   └── build_features.py
│   ├── models
│   │   ├── predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   ├── visualization
│   │   └── visualize.py
└── tox.ini
```

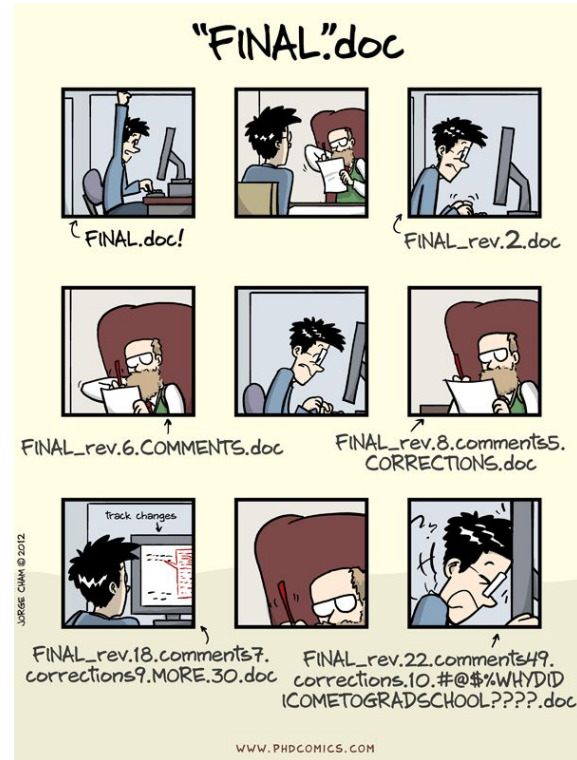
<- Makefile with commands like 'make data' or 'make train'
<- The top-level README for developers using this project.
<- Data from third party sources.
<- Intermediate data that has been transformed.
<- The final, canonical data sets for modeling.
<- The original, immutable data dump.
<- A default Sphinx project; see sphinx-doc.org for details
<- Trained and serialized models, model predictions, or model summaries
<- Jupyter notebooks. Naming convention is a number (for ordering), the creator's initials, and a short '-' delimited description, e.g. '1.0-jqp-initial-data-exploration'.
<- Data dictionaries, manuals, and all other explanatory materials.
<- Generated analysis as HTML, PDF, LaTeX, etc.
<- Generated graphics and figures to be used in reporting
<- The requirements file for reproducing the analysis environment, e.g. generated with 'pip freeze > requirements.txt'
<- Make this project pip installable with 'pip install -e'
<- Source code for use in this project.
<- Makes src a Python module
<- Scripts to download or generate data
<- Scripts to turn raw data into features for modeling
<- Scripts to train models and then use trained models to make predictions
<- Scripts to create exploratory and results oriented visualizations
<- tox file with settings for running tox; see tox.readthedocs.io

Git gud with git + GitHub

The case for a selfish
single-user workflow

Version control and you(r professor)

- Version control helps maintain a *clean* and *legible* history of your work
- Removes the fear of deleting codebits because "what if I might need it later?"
- The ultimate outcome should be version control helps *you*



git: Using a flamethrower to kill a fly?

- git is a version control system
 - It is very powerful + feature-intensive
 - (And that can be really overwhelming!)
- There are a few commands that can get you ~80% of the way
 - But it takes lots of practice!



Because git is so
notoriously frustrating:

<https://dangitgit.com/>

GitHub: distributed version control

- GitHub is like Dropbox for git
- **You do not need to use GitHub to work with git**
 - Buuuuut, backing things up is always a good idea
- GitHub adds a significant amount *more* jargon to your git workflow

git/GitHub jargon

repository : a directory that is tracked by git

commit (noun) : a snapshot of a git repository at a point in time

commit (verb) : the act of creating a commit (noun)

history : the complete series of commits in a repository

remote : a copy of a git repository hosted somewhere else (e.g., GitHub)

pull/push (verb) : the process of sending/receiving commits from a remote

The single-user workflow

1. Run `git init`
2. Make changes to your files
3. Run `git add`
4. Run `git commit`
5. Repeat steps 2-4 *ad infinitum*

The single-user workflow (with backup)

1. Run `git init`
2. Run `git remote add origin git@github.com:<user>/<repo>`
3. Make changes to your files
4. Run `git add`
5. Run `git commit`
6. Run `git push -u origin main`
7. Repeat steps 3-6 *ad infinitum*

git/GitHub resources

- <https://the-turing-way.netlify.app/reproducible-research/vcs>
- <https://github.com/git-guides/>
- <https://swcarpentry.github.io/git-novice/>

Let's git started!

Make sure you've made a GitHub account and have git installed on your computer if you want to follow along!

