RESOURCES for the FUTURE

Learning Lab: Data Management at RFF

Presented by Alexandra Thompson & Jordan Wingenroth

Data Governance Working Group

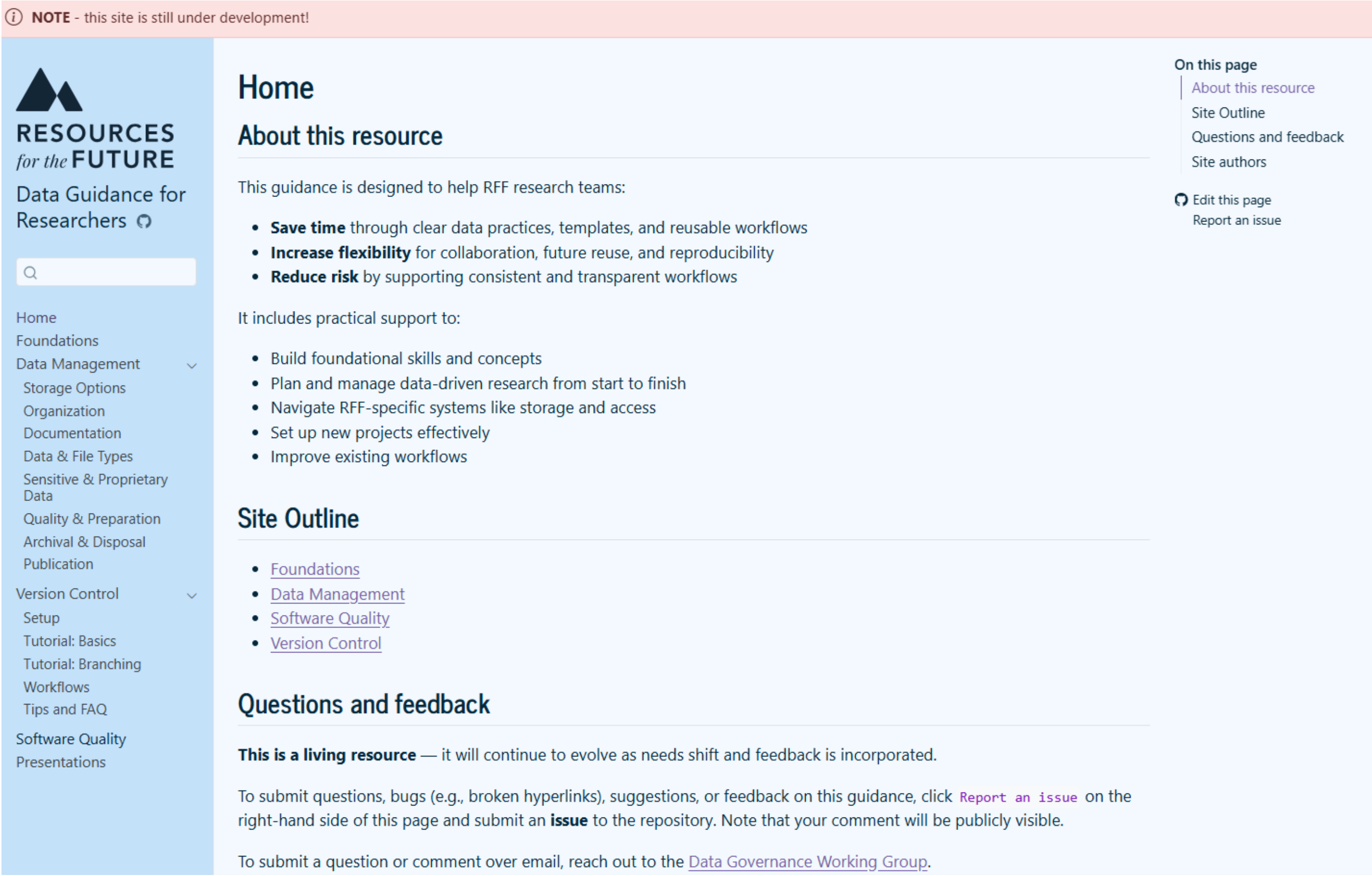
July 17, 2025

Data Governance Working Group

- Aris Awang
- Penny Liao
- Mike Raftery
- Ethan Russell
- John Valdez
- Matthew Wibbenmeyer
- Jordan Wingenroth
- Alexandra Thompson

Guidance website

- [Home – Data Guidance for Researchers (https://rff-data-projects.github.io/rff-data-gov-guidance/index.html)](https://rff-data-projects.github.io/rff-data-gov-guidance/index.html)



Outline for today

1. Organization
2. Storage
3. Documentation
4. Data & file types
5. Sensitive & proprietary data
6. Data quality & preparation
7. Archiving
8. Publication

Organization: General Principles

- **Use a simple, descriptive folder structure** that reflects project logic and is easy for new users to understand.

- **Design for flexibility**, allowing for new data, methods, or collaborators without major reorganization.

- **Name files/folders to be human- and machine-readable**: use clear names, avoid spaces and special characters, and include numeric prefixes for order.

- **Keep raw data** in a dedicated folder (*raw/*) and never modify it directly.

```
project_name/
  ├── data/
  ├── scripts/
  ├── results/
  ├── docs/
```

```
project_name/
  ├── data/
  │     ├── raw/
  │     ├── intermediate/
  │     ├── clean/ (optional)
  ├── scripts/
  │     ├── processing/
  │     ├── analysis/
  │     ├── tools/ (optional)
  ├── results/
  ├── docs/
```

Organization: Personal Repository Structure for Version Control

- Support **version control** by organizing files in a way that works with Git by creating separate repository folders for each user.
  - Allows users to work on files simultaneously without interrupting others’ workflows
- **Option 1: Version controlling scripts only**

```
project_name/
├── data/
│   ├── raw/
│   ├── intermediate/
│   ├── clean/ (optional)
├── results/
├── docs/
├── repos/
│   ├── baker/
│   │   ├── .git/
│   │   ├── scripts/
│   │   │   ├── processing/
│   │   │   ├── analysis/
│   │   │   ├── tools/
│   ├── johnson/
│   │   ├── .git/
│   │   ├── scripts/
│   │   │   ├── processing/
│   │   │   ├── analysis/
│   │   │   ├── tools/
```

Personal repository project folder structure to version control scripts

Organization: Personal Repository Structure for Version Control

- Support **version control** by organizing code and outputs in a way that works with Git by creating separate repository folders for each user.
    - Allows users to work on files simultaneously without interrupting others' workflows
- Option 1: Version controlling scripts only
- **Option 2: Version controlling scripts, results, and documents.** Useful for working with:
    - External collaborators
    - Small result files

```
project_name/
   ├── data/
   │    ├── raw/
   │    ├── intermediate/
   │    ├── clean/ (optional)
   ├── repos/
   │    ├── baker/
   │    │    ├── .git/
   │    │    ├── scripts/
   │    │    │    ├── processing/
   │    │    │    ├── analysis/
   │    │    │    ├── tools/
   │    │    ├── results/
   │    │    ├── docs/
   │    ├── johnson/
   │    │    ├── .git/
   │    │    ├── scripts/
   │    │    │    ├── processing/
   │    │    │    ├── analysis/
   │    │    │    ├── tools/
   │    │    ├── results/
   │    │    ├── docs/
```

Personal repository project folder structure to version control scripts, results, and docs

Storage Options

Component	Primary Use	Notes
L:/ Drive	Data and script storage	- Use for all projects - Regular backups - Large storage capacity - Secure (RFF-only access) - See Guidance for setup instructions
GitHub	Script version control, script sharing	- **Use for all projects** - Supports collaboration, history tracking, and (optionally) project management
Additional tools for projects with external collaborators		
OneDrive	Sharing essential data with external partners	- Good for smaller files - Shared access with external collaborators
Other cloud options	Sharing large datasets externally	- Azure, AWS S3, Dropbox, Google Bucket - Contact ITHelp@rff.org for Azure setup
ArcGIS Online	Sharing spatial data	- Share and visualize spatial data over web browser - Contact Thompson@rff.org for setup

Storage Options

Component	Primary Use	Notes
L:/ Drive	Data and script storage	- **Use for all projects** - Regular backups - Large storage capacity - Secure (RFF-only access) - See Guidance for setup instructions
GitHub	Script version control, script sharing	- **Use for all projects** - Supports collaboration, history tracking, and (optionally) project management
Additional tools for projects with external collaborators		
OneDrive	Sharing essential data with external partners	- Good for smaller files - Shared access with external collaborators
Other cloud options	Sharing large datasets externally	- Azure, AWS S3, Dropbox, Google Bucket - Contact ITGhelp@rff.org for Azure setup
ArcGIS Online	Sharing spatial data	- Share and visualize spatial data over web browser - Contact Thompson@rff.org for setup

Storage Options

Component	Primary Use	Notes
L:/ Drive	Data and script storage	- **Use for all projects** - Regular backups - Large storage capacity - Secure (RFF-only access) - See Guidance for setup instructions
GitHub	Script version control, script sharing	- **Use for all projects** - Supports collaboration, history tracking, and (optionally) project management
Additional tools for projects with external collaborators		
OneDrive	Sharing essential data with external partners	- Good for smaller files - Shared access with external collaborators
Other cloud options	Sharing large datasets externally	- Azure, AWS S3, Dropbox, Google Bucket - Contact ITHelp@rff.org for Azure setup
ArcGIS Online	Sharing spatial data	- Share and visualize spatial data over web browser - Contact Thompson@rff.org for setup

Storage Options

Component	Primary Use	Notes
L:/ Drive	Data and script storage	- **Use for all projects** - Regular backups - Large storage capacity - Secure (RFF-only access) - See Guidance for setup instructions
GitHub	Script version control, script sharing	- **Use for all projects** - Supports collaboration, history tracking, and (optionally) project management
Additional tools for projects with external collaborators		
OneDrive	Sharing essential data with external partners	- Good for smaller files - Shared access with external collaborators
Other cloud options	Sharing large datasets externally	- Azure, AWS S3, Dropbox, Google Bucket - Contact ITHelp@rff.org for Azure setup
ArcGIS Online	Sharing spatial data	- Share and visualize spatial data over web browser - Contact Thompson@rff.org for setup

Storage Options

Component	Primary Use	Notes
L:/ Drive	Data and script storage	- **Use for all projects** - Regular backups - Large storage capacity - Secure (RFF-only access) - See Guidance for setup instructions
GitHub	Script version control, script sharing	- **Use for all projects** - Supports collaboration, history tracking, and (optionally) project management
Additional tools for projects with external collaborators		
OneDrive	Sharing essential data with external partners	- Good for smaller files - Shared access with external collaborators
Other cloud options	Sharing large datasets externally	- Azure, AWS S3, Dropbox, Google Bucket - Contact IThelp@rff.org for Azure setup
ArcGIS Online	Sharing spatial data	- Share and visualize spatial data over web browser - Contact Thompson@rff.org for setup

Storage Options

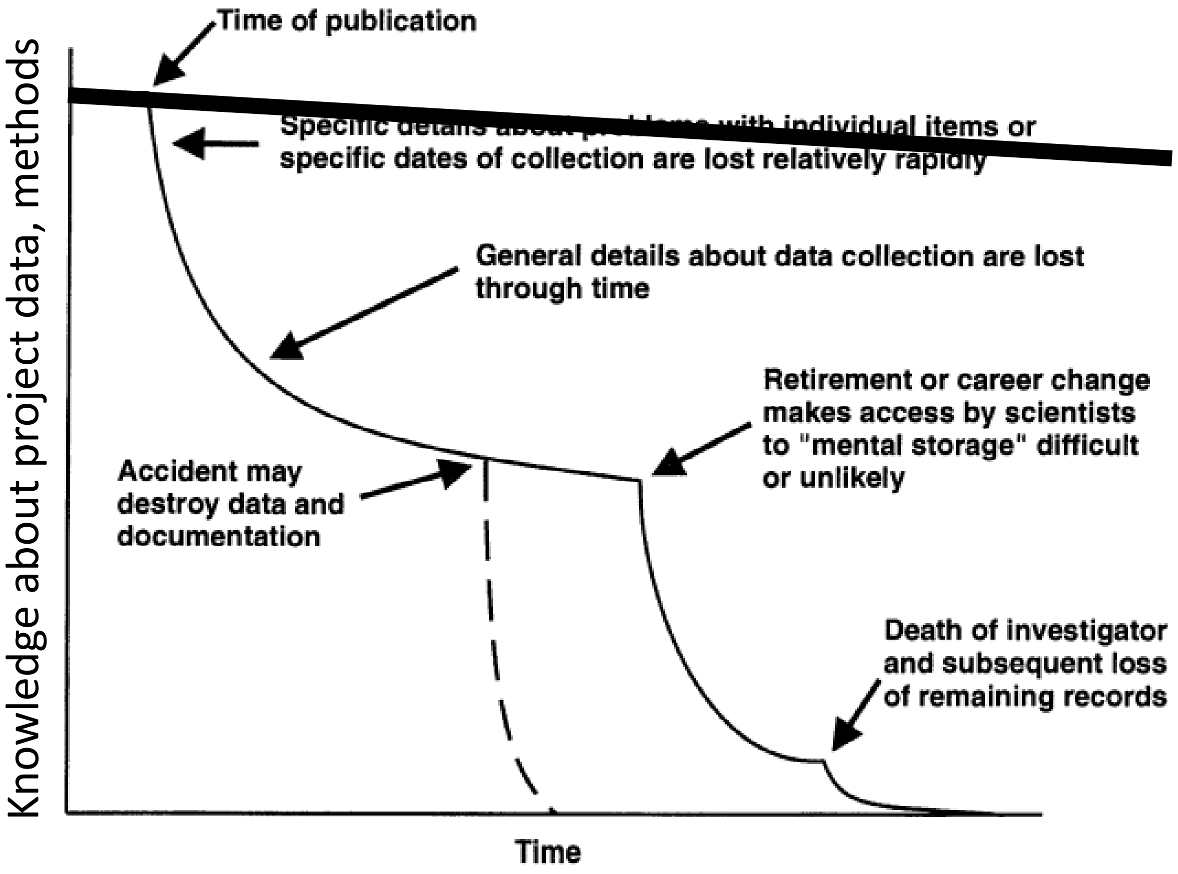
Component	Primary Use	Notes
L:/ Drive	Data and script storage	- **Use for all projects** - Regular backups - Large storage capacity - Secure (RFF-only access) - See Guidance for setup instructions
GitHub	Script version control, script sharing	- **Use for all projects** - Supports collaboration, history tracking, and (optionally) project management
Additional tools for projects with external collaborators		
OneDrive	Sharing essential data with external partners	- Good for smaller files - Shared access with external collaborators
Other cloud options	Sharing large datasets externally	- Azure, AWS S3, Dropbox, Google Bucket - Contact IThelp@rff.org for Azure setup
ArcGIS Online	Sharing spatial data	- Share and visualize spatial data over web browser - Contact Thompson@rff.org for setup

Documentation

- ***"If we are not conscientious documenters, we can easily end up… without the ability to coherently describe our research process up to that point."***
  — Stoudt, Vásquez, and Martinez (2021)
- Enables **reuse** and interpretation by others and your future self
- Supports **reproducibility**, project **continuity**
- Reduces **errors**, prevents **duplication of effort**
- Helps projects **retain value**
- Documentation along the way makes it **faster and easier** to compile final products

Without documentation

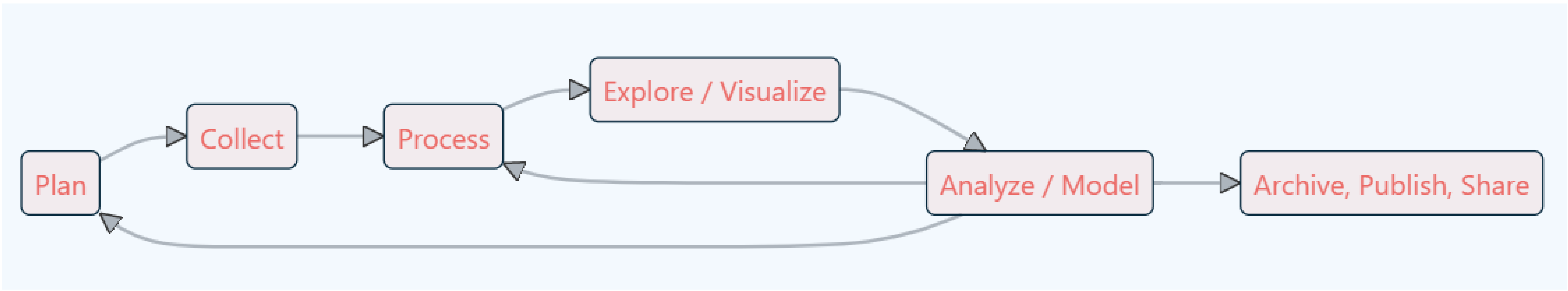
With documentation



Michener, W. K., J. W. Brunt, J. J. Helly, T. B. Kirchner, and S. G. Stafford. 1997. "Nongeospatial Metadata for the Ecological Sciences."

Documentation

Documentation phase



Preliminary
- Data management plan
- Project-level README *(first draft)*

Process
- Version control
- Code comments
- Raw data READMEs

Product
- Metadata
- Code comments (*refinement*)
- Project-level README *(final draft)*

Documentation

- Data management guidance provides specific instructions for four crucial pieces of documentation:

    1. Data management plan
       (10 questions with helpful links)
    2. Project-level READMEs
    3. Raw data READMEs
    4. Metadata

- Coming soon

    5. Version control
    6. Code comments

The first step: Data management planning for reproducibility

Data management planning, a form of **preliminary** documentation, is the process of thinking ahead about how your team will access, use, create, modify, store, share, and describe data related to your research project. Data management plans …nhance collaboration by establishing baseline expectations, make projects resilient to turnover, and save time …ng run.** In addition, many funders require data management plans be submitted with grant proposals, so thinking …ese issues early can facilitate the proposal process.

…dance resource provides general instructions for data practices and addresses many of the core questions that are part …MP process. ***At a minimum, the questions below should be reviewed at the start of a project.*** Associated guidance

…re will data/code be stored and how will it be organized?

…will the team use Microsoft Teams for file storage and communication? What types of files will be stored in the …ns folder versus the project's L:/ drive folder? What will be communicated over Teams chat versus email or GitHub?

…will be responsible for disposing / archiving data?

…will be responsible for publishing data/code and attaching appropriate documentation and use licenses?

…t will the version control / git / GitHub workflow be?

…t coding software and main libraries will be used?

…will code be reviewed for quality?

…t are expectations around data quality and code quality?

…will data sources, code, and major methodological decisions be documented?

… the appropriate budget been allocated to implement this DMP?

…rojects may require additional considerations. For development of a more thorough DMP, refer to the UCSB NCEAS …

Data-level documentation (metadata)

It is recommended to attach **metadata** files to published datasets. Metadata ("data about data") documents the "who, what, when, where, how, and why" of a data resource. Metadata not only allows users (your future self included) to understand and use datasets, but also facilitates search and retrieval of the data when deposited in a data repository.

Below are the key components of metadata. They can be stored in a simple text or markdown file.

- Title: Descriptive name of the dataset.
- DOI number: Associated with the final publication, dataset, or both
- Abstract: Summary of the dataset's content
- Keywords: Relevant terms for search and discovery
- Temporal Extent: Time period covered by the data
- Data Format: File format
- Data Source(s): Origin of the data
- Accuracy and Precision: Information about data quality
- Access Constraints: Restrictions on data use
- Attribute / field definitions: Define all abbreviations and coded entries
- Additional geospatial metadata components, if applicable
- Spatial Extent: Geographic coverage (bounding coordinates)
- Projection Information: Coordinate system details

3.2 Raw data

Best practices:

- The source of all downloaded raw datasets should be documented in a README file.
- Create a README file associated with each raw data file, or each logical "cluster" of related raw data files, in the same folder as the data.
- If there are multiple data files in a folder, name the README so that it is easily associated with the data file(s) it describes (e.g., `README_PRISM_Daily_Temperature.txt`).
- Format README files consistently.
- Write the README document in an plain text and open source file format, such as .txt or .md

3.1 The project-level README file

At a minimum, the project should have a README file in text … information listed below. This can be an evolving, living docu… **product documentation**, it's easiest to start it early in the pr…

- Project name and description
- Project PI and contact information
- List of staff responsible for data management and code development
- Associated final product(s), date of release, and DOI (if applicable)
- Link to published data/code (if applicable)
- License associated with final product(s)

Data Types

- Important for
  - Accuracy
  - Precision
- Considerations
  - Memory
  - Software behavior, changing data types
  - Floating point rounding errors

5.1.1 General data types

Data type	Definition	Typical memory	Precision	Common Operations	Examples
Character (string/text)	text or string values	~1 byte per character	Not applicable (stores characters, not numeric values)	Concatenation, substring, pattern matching	`"Hello, world!"`, `'#23'`, `"'Why?', they asked."`
Integer	Whole numbers (no decimal point)	4 bytes (32-bit)	Exact for values within allowed range	Arithemtic, comparisons, indexing	`0`, `42`, `-6`, `2e+30` (scientific notation)
Floating point	Numbers with decimals (real numbers, including fractions)	4 bytes (*float*), 8 bytes (*double*)	Approximate - may introduce small rounding errors	Arithmetic, scientific calculations	`-54.3`, `3.14159`
Boolean / Logical	Binary values	Typically 1 byte	Exact	Logical operations (e.g., `AND`, `OR`, `NOT`)	`TRUE`, `FALSE`, `0`, `1`
Date/time	Calendar dates and/or clock times	Varies by system (~8+ bytes)	High precision in supported range	Often require specialized functions; format inconsistencies can cause import errors	`2025-05-29`, `1990-01-24 14:30:00`

Data File Formats: Openness

- Proprietary
  - Designed for use with a specific software/language
  - Can be costly (licensed, closed-source)
  - Often encrypted, preventing effective version control
- Non-proprietary (**recommended**)
  - More easily moved from one language or software tool to another
  - Compatible with open-source tools
  - Better for collaboration, long-term access, and reproducibility

Data File Formats

- Avoid creating proprietary file types when possible
- Figures: use vector formats when feasible (.svg, .eps)
- Limitations of working with text-based file formats (.csv, .tsv, .txt)
    - No metadata retention (type guessing on import)
    - Precision loss on export
    - See guidance for best practices

Characteristics of tabular formats

Format	Extensions	Open-source or proprietary	Retains data types	Level of structure
Comma or tab-separated values	.csv, .tsv	Open-source	No	Structured
Plain text	.txt	Open-source	No	Semi-structured
Microsoft Excel spreadsheet/workbook	.xls or .xlsx	Proprietary	Yes	Structured
Feather	.feather	Open-source	Yes	Structured
Parquet	.parquet	Open-source	Yes	Structured
RData	.rdata or .rds	Open-source	Yes	Structured
Lightning Fast Serialization of Data Frames	.fst	Open-source	Yes	Structured
SQLite	.sqlite, .db	Open-source	Yes	Structured
Stata data file	.dta	Proprietary	Yes	Structured
SAS dataset	.sas7bdat	Proprietary	Yes	Structured
Database File	.dbf	Open-source	Yes	Structured

Characteristics of hierarchical formats

Format	Extensions	Open-source or proprietary	Retains data types	Level of structure
Hierarchical Data Format version 5 (HDF5)	.h5, .hdf5	Open-source	Yes	Structured
Network Common Data Form (NetCDF)	.nc	Open-source	Yes	Structured
JavaScript Object Notation	.json	Open-source	No	Semi-structured
eXtensible Markup Language	.xml	Open-source	No	Semi-structured
YAML	.yml or .yaml	Open-source	No	Unstructured

Data Sensitivity & Data Quality

Sensitive & Proprietary Data

- Categories
  - Proprietary
  - Regulated
  - Confidential

- See guidance for steps on how to identify and secure this data

Data Sensitivity & Data Quality

Sensitive & Proprietary Data

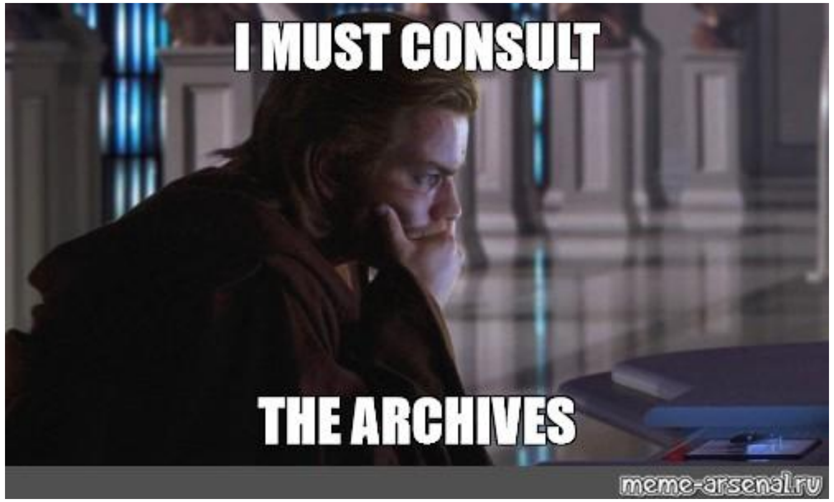
- Categories
  - Proprietary
  - Regulated
  - Confidential
- See guidance for steps on how to identify and protect this data

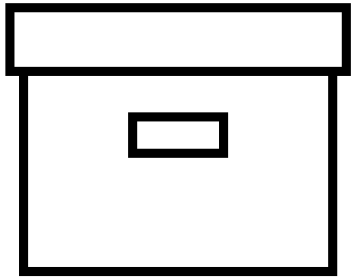
Data Preparation and Quality

- *Quality assurance (QA):* ensuring the accuracy, consistency, and reliability of data
- See guidance for
  - QA practices
  - Tidy data
  - Data type conversion and standardization
  - Links to helpful resources

Archiving: Secure Long-Term Storage of Final Data

- Archiving: moving finalized project data into a state of **long-term storage**.
  - Preserves data for **reproducibility and future use**
  - Improves **organization and accessibility** of finished projects
  - Frees up **computing/storage resources** for new projects
- At RFF, archived project folders:
  - Remain on the **L drive**
  - Are set to **read-only** to preserve content
  - Content can still be viewed and copied by **authorized users**

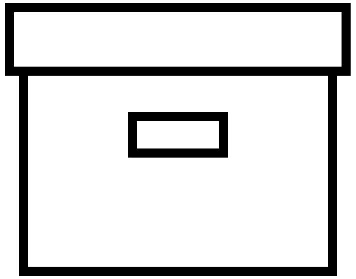


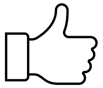


Archiving: Secure Long-Term Storage of Final Data

- Archiving: moving finalized project data into a state of **long-term storage**.
  - Preserves data for **reproducibility and future use**
  - Improves **organization and accessibility** of finished projects
  - Frees up **computing/storage resources** for new projects
- At RFF, archived project folders:
  - Remain on the **L drive**
  - Are set to **read-only** to preserve content
  - Content can still be viewed and copied by **authorized users**

- Archiving steps
  1. Finalize file **organization**
  2. Complete **documentation**
  3. Contact IT to set up **security** and long-term **folder access**





Yes, projects can be un-archived if needed!

Publication: Overview

Publication is making code/data available to the broader community through formal dissemination channels such as data repositories, journal articles, or public databases.

Key Benefits:

- Discoverable and accessible data
- Reproducibility and collaboration
- Visibility and impact of research
- Meeting journal and funder requirements

Publication: Licensing

Using a **license** when publishing code or data clarifies permissions, prevents misunderstandings, and encourages responsible use.

- First, make sure that you or your team own the **intellectual property** (IP) rights for the code or data
  - When the research team has joint IP ownership with funders or collaborators, licensing options must be agreed upon by consensus
- After you have confirmed ownership, the next step is to choose a license
- Default to using **permissive** licenses except in special circumstances

Publication: Software License Options

The two most popular software licenses in the academic community are:

- The MIT License **(more permissive)**
- The GNU General Public License

Unlike most commercial patents and copyrights, these licenses are designed with the intent to make a product (source code for software) available for others to use freely

Publication: The MIT License

- What you see is what you get:
- Users, including commercial entities, can view, use, modify, distribute the work freely
- Requires **attribution**
- Has consistently been the most widely used open-source license for many years

```
Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy,
modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Publication: The GNU General Public License (GPL)

- Far too long to fit on a slide
- Also provides the same permissions as the MIT license (view, use, modify, distribute, etc.), also requires **attribution**
- The key difference is that it is a **copyleft** license
  - Any derivative works (modifications or adaptations) must be released under the same GNU General Public License
  - This makes it **viral**, because as software is expanded upon and repurposed, the same license applies to new versions (and their subsequent "offspring")
  - For comparison, the MIT license allows for derivative works to be restricted under additional licensing policies and used in proprietary software, referred to as **enclosure**

Publication: Choosing a License

	MIT License	GNU GPL
Characteristic:		
Requires attribution	✓	✓
Requires use of same license	✗	✓
Modified versions must be open source	✗	✓
Very unlikely to cause future headaches	✓	✗
Use case:		
Packages available for use in **any** dependent models and software	✓	✗
Models and software intended for use and adaptation but **not enclosure** (takeover)	✗	✓

Publication: Data Licenses

Two of the most commonly used data licenses are produced by Open Data Commons (ODC)

- ODC-By:
  - The more permissive and flexible
  - Users, including commercial entities, can view, use, modify, and distribute the work freely
  - Allows for commercial use and **enclosure**
  - Requires **attribution**
- ODbL:
  - Allows for the same free access, use, and modification
  - Carries **share-alike** (similar to **copyleft**), which requires future distributions to be released under the same license
  - Ensures open access
  - Also requires **attribution**

Publication: Publishing Code

- Some journals may require authors to:
  - Make the codebase publicly accessible
  - Include a link to the GitHub code repository in the manuscript
  - Reference the codebase DOI in the manuscript
- If code is already managed with up-to-date version control on GitHub, this process can be effortless
- If code is not on GitHub, starting to track it now will pay off later

Publication: Publishing Code

More details for publishing code are available on the guidance site

Recommended Publishing Workflow

1. If you are not already a member of the RFF GitHub organization, request membership by emailing DGWG.
2. Attach the appropriate license to the repository, if you have not done so already.

If your repository is already in the RFF GitHub organization:

3. Make the repository public once you're ready to publish if it isn't already.
4. Tag the publication version (e.g., v1.0-publication-name) as described here. (Note: only admins can do this, request admin privileges if needed.)

If your repository is under a personal GitHub account:

3. Decide between **forking** or **transferring** the repo to the RFF GitHub organization (see explanation of these options above).
4. Contact the RFF GitHub organization admins at DGWG@rff.org to initiate the fork or transfer and to add a tag (*note: tags do not transfer automatically when a repo is forked*).
5. (Optional step) Archive the tagged version on Zenodo and link the DOI in your README. This enables persistent citation, version-specific reference, and supports scholarly credit.

Publication: Publishing Data

- Modern journals often request data to be publicly available when possible so that other researchers can reproduce and extrapolate upon results

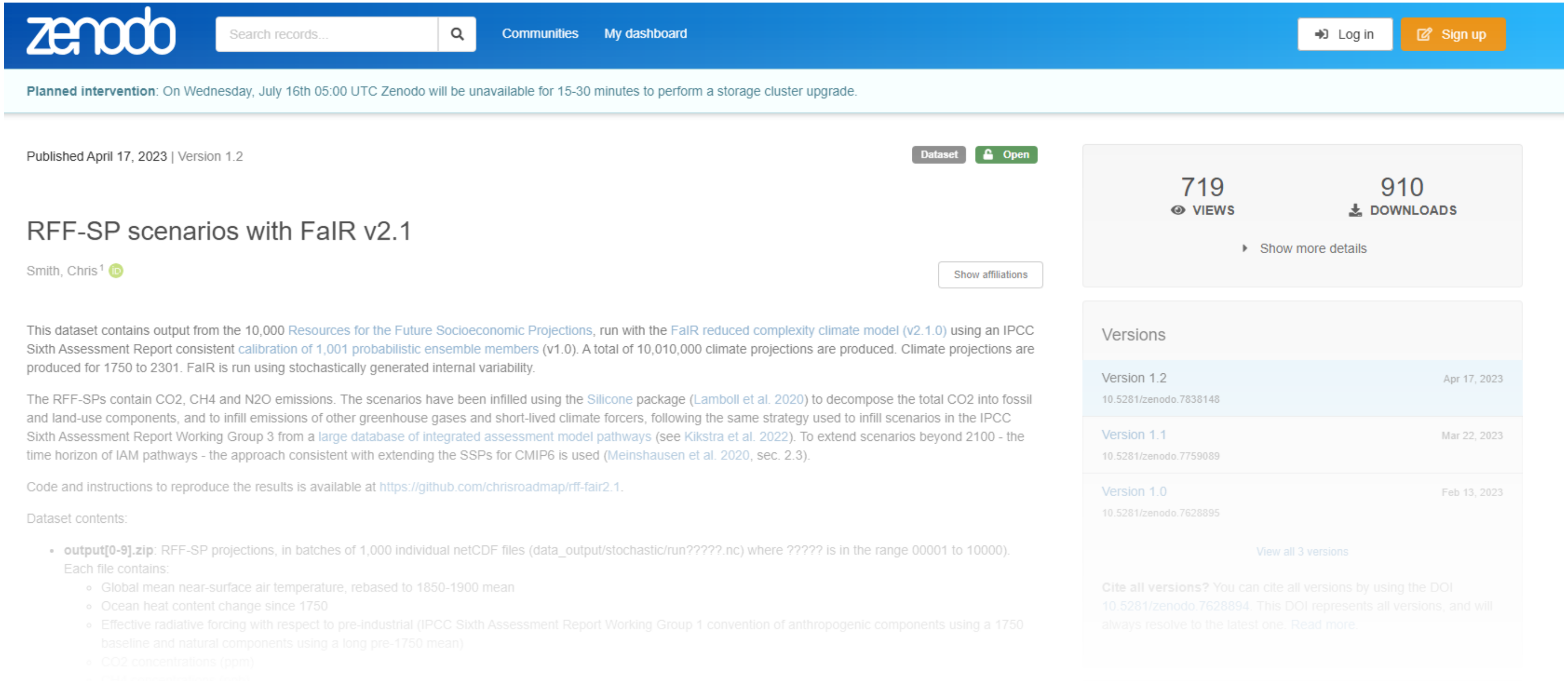
- To make data useful for others, it is key to include **metadata**:
  - documents the "who, what, when, where, how, and why" of a data resource
  - allows users (your future self included) to understand and use datasets
  - facilitates search and retrieval of the data when deposited in a data repository

Publication: Publishing Data

- Examples of metadata elements that are good to include:
  - Title of dataset
  - Keywords
  - Time range
  - Data format
  - Data sources
  - DOI
  - Link to a code repository used to generate the data
  - Descriptions of variables/fields (definitions, units of measure, etc.)

Publication: Publishing Data

One great resource for publishing data is Zenodo (zenodo.org)



Publication: Publishing Data

More details for publishing data are available on the guidance site

Uploading data to Zenodo

Zenodo is an online repository for sharing research data, software, and other scientific outputs. It has a broad disciplinary focus and is safe, citeable (every upload is assigned a DOI), compatible with GitHub, and free for up to 50GB of storage.

Step 1: Prepare the research data

Before uploading, ensure that:

- The data is well-organized (e.g., structured folders, clear file names).
- Metadata files are prepared for each data file or sets of data files.
- A license is attached.
- Any sensitive or restricted data is removed or anonymized (if applicable).
- The project-level README is up to date.

For guidance on choosing which files to publish and how to handle API-accessed data, see Finalize data organization.

Step 2: Create a Zenodo account & access the upload dashboard

- Go to Zenodo and sign in (or create an account). Note that you can create an account using your GitHub profile.
- Click the "New Upload" button on the Zenodo dashboard.

Step 3: Upload the data files, fill in metadata, set access

- Upload data, metadata, and the project-level README file.
- Enter metadata information in applicable fields (contributors, associated journal article or conference presentation, etc.)
- Include a link to the GitHub code repository.
- Choose an Access Level
  - Open Access: Publicly available for anyone.
  - Embargoed: Set a release date if the data must remain private for a certain period.

Please fill out the survey!

Thank you.

- Find out more about RFF online: **www.rff.org**
- Follow us on **LinkedIn**
- Subscribe to receive updates: **rff.org/subscribe**