RDM Handbook DRAFT

2022-06-09

Contents

Welcome!

This resource is currently in **DRAFT** and not intended for public use. It is being made available for demonstration purposes only.

Outline

4 potential bigger buckets

- Naming and organization
- Data Storage
- Roles and Responibilities
- The Wrap Up

Research Data Management (RDM) at its core is about being organized, which means being intentional and coordinated - when working with others - about where we put things, how we name things, how we describe things, and - with everything likely stored on a computer - how we do this all in a way that is both human and machine accessible. The spin off of this is reasearch that is transparent, reproducible, and, if desired, shareable.

It can seem like a lot. But working with a couple of principles and with a little planning, it's not hard to tackle.

Roles & Responsibilities

It's important to know - when working in a team - who is resonsible for what. This chapter walks you through some of the questions a research team should be asking themselves at the outset about who will be doing what and when. These are then addressed in more detail in each subsequent section.

Offboarding

Tightly wed to roles and responsibilities, when someone leaves a research project, things need to be passed along and access to components of a research project need to be updated. This chapter covers considerations and approaches to handling offboarding.

File Naming

There are few things more frustrating then when preparing for a submission and opening a folder to find somethign like the following list of files…

```
Anlaysis to be reviewed.docx
Analysis to be reviewed MVD.docx
Analysis to be reviewed final.docx
Analysis to be reviewed FINAL.docx
Analysis FINAL.docx
Analysis FINAL - Updated.docx
Analysis - Last update - FINAL.docx
```

This chapter covers common conventions to avoid this headache - allowing you to look at a file and determine it's author, it's place in the chain of edits, and if needed, to use code to extract files of a specific designation from across and within complex nested folder structures.

Directory Structures

When projects get complex, file naming alone is not sufficient. When we're working across multiple projects, having fundamentally different organizational structures can be frustrating. This chapter covers common convetions in setting up directory structures that help streamline how new projects are started, who has access to what kind of content, and with the use of a template sets a common structure across projects.

Permissions & Ownership

Not everyone needs access - nor shoudl everyone require access - to everything related to a research project. Data in particular may be sensitive. This chapter covers approaches to establishing ownership over files and directories and setting permissions - the abaility to access or edit - on these for different partners in the research project.

File Formats

One person works in `.csv` another in `.xls` another in `.xlsx` and yet another in `.ods`. This limits portability of the information, risks introducing file errors as they are passed between computing environments, and causes headaches when you're not working in your preferred format in your preferred application. This chapter covers approaches to standardizing file formats used across a research project.

Documentation

How roles are assigned for a specific project, what naming conventions and directory structures should be used, how specific aspects of a research project should be undertaken should all be articulated. This serves as a reminder, it keeps people accountable, and it simplifies onboarding new research partners to an existing project. This chapter covers conventions for migrating tacit knowledge to documentation throughout a research project to ensure the project is comprehensible.

Version Control

Sometimes we need to track changes in certain documents over time. Version control let's us know what the currrent working copy is and how a file has changed over time. Version control can be handled by software, but also through naming conventions. This chapter covers common approaches to tackling version control with naming conventions.

Data Storage

Research data often has restrictions or obligations for where it's stored and how long it's stored for. Research data is also incredibly value, so we'd be loath to keep only one copy of it. This chapter covers outlining storage solutions from the outset that address potential restrictions and obligations. It also covers approaches to keeping multiple copies of research data to protect against accidental loss.

Wrapping up a project

Retention & Deposit

Retention schedules

Roles & Responsibilities

A lab manual is a way to communicate the protocols, practices, and procedures (among other things) of a research lab or group, with the goal of avoiding confusion and misunderstandings. When creating a lab or team manual, defining roles and responsibilities, and ensuring that everybody is very clear about their role in managing a project's data, is of primary importance.

The roles and responsibilities of your research will come into play in all other modules in this guide, and while we all want things to go perfectly, life happens and things can slip off track. If things do start to become disorganized, this shouldn't be a source of shame or embarrassment, and you can use this guide as a reference to help get things back in order.

General Rules

While not technically "rules", below are some suggestions and things to consider when developing your lab manual.

- **Designating Roles**: If you have research or lab manager in your group, designating responsibilities can largely fall to this person. However, if your group does not have this as a formal role, it is important that the team discusses and designates roles and responsibilities amongst themselves, and that everybody has a clear idea of what others are doing, and what is expected of them. These roles should be documented somewhere. See the chapter on documentation.
- **Access Permissions**: It's very possible that everybody within a research group won't need access to all the group's data. It's also possible that due to data sensitivities and restrictions, there will be team members who explicitly should not have access to some data. In addition to designating who will provision access permissions, it's also important to clearly outline and manage each team member's data access. This should be incorporated in both the onboarding and offboarding of team members.

- **Onboarding**: When a person first joins a research group, things can be overwhelming and not all information will be retained. While it's important to ensure there is a person or person who is responsible for onboarding, and that roles and responsibilities are explained to new members, it is equally important for there to be a documentation for reference (and that the documentation be updated as necessary).
- **Updating**: Just as much as it's valuable to develop naming conventions, folder hierarchies, versioning practices, and documentation, it's equally (and potentially more) important to ensure that the documentation of any new additions or practices be updated. Determining who will update the documentation will be specific to every team, and again, ensuring that expectations are clearly communicated is key.
- **Setting a Schedule**: Whether it's daily, weekly, bi-weekly, etc., having regularly scheduled tasks that can apply to specific group members or to everybody, will help keep things consistent and updated. This can apply to things like backups and reviewing any documentation that may need to be updated due to changes or additions.
- **Regular Check-ins**: Everybody is busy, but it's useful to have regular check-ins to make sure everything is staying on track, and to address any issues or questions people may have.

Questions

- What roles currently exist in your lab team?
- How are roles and responsabilties communicated?

File Naming

File naming isn't exactly fun, but it is crucial for organizing, describing, and managing research.

First, some general rules and a quick reference, then we break these down and explain the processes behind them.

General Rules

- File names should only contain letters in the English alphabet, numbers from `1-9`, dashes `-`, and underscores `_`.
- Do not use spaces or special characters, including `# % & < > : " / | ? * { } $ ! ' @ + =`
- File names should be broken down into conceptual components that are separated by underscores `_`.
- If more than one word is needed in each conceptual component, these are separated by dashes `-`.
- All file names should start with the last name of the creator of the file, followed by the project name, and then what the file contains. Additional blocks may be added as needed, these may include dates, versions, if a file is restricted etc.
- All of these components should be meaningful (read on for what it means for a file name to be meaningful!)
- All file names should end with a file type extension.

Depending on what the file pertains to, how exactly these guidelines are implemented will differ slightly.

Quick Reference

Type	Format & Example
Maunscript	**F** LastName_Project_File-contents_Version.file-type**E** VisDunbar_SMS-DB-Prev_Manuscript_V0.docx
Feedback	**F** LastName_Project_File-contents_Version_Editor-Initial.file-type**E** VisDunbar_SMS-DB-Prev_Manuscript_V0_NR.docx
Figure, Plots, Images	**F** LastName_Project_Figure-title_Version.file-type**E** VisDunbar_SMS-DB-Prev_Fig1_V0.docx
Scripts	**F** LastName_Project_Script_Version.file-type**E** VisDunbar_SMS-DB-Prev_Data-Clean_V0.docx
Data	**F** LastName_Date_Project_Data-file-description.file-type**E** VisDunbar_20210901_SMS-DB-Prev_P-Data.docx

What's in a name

File names need to achieve two primary goals, they need to make sense to a human reading them and they need to be constructed in a way that allows a computer to parse or process them. That is, file names should be **human interpretable** and **machine readable**.

Human interpretable

To be human interpretable, a file name needs to be meaningful. To do this, it needs to convey some basic information to a person reading it. We do this by integrating metadata into the file name. The metadata elements we include are:

- Who created the file
- The date on which it was created
- The project to which it is connected
- The nature of the contents of the file
- If it's been modified

- The type or format of the file

That is, we should be able to look at a file name and tell, who created it, when it was created, what it is related to, what is inside of it, if it has been updated, and what application we should expect to be able to open it with.

By using underscores _ to break up the metadata in a file name and by always using the same patterns, we quickly become atune to these patterns and picking out specific files from a long list becomes increasingly easy.

All said, that's a fair bit of information to hold in a file name. However, using standard truncation can be useful to manage overly long file names.

Machine readable

A file name that is machine readable or machine interpretable means creating file names in such a way that they can be sorted by an application along specified delimiters. This means building our names according to set patterns, which can then be parsed, and if necessary, described. Lastly, it means building our names in such a way that if we move them from one computer to another, from one application to another, or from one operating system to another, the files remain interpretable in exactly the same way.

By using underscores _ to break up the metadata in a file name and by always using the same patterns, we can computationaly isolate files based on each of these attributes irrespective of the directory structure in which we store them, and, for example, find all files from a particular creator, project, or created on a specific date.

Special characters

Different computing environments (applications, operating systems etc) handle character encoding in idiosyncratic ways. If you've ever opened a document and seen what look like odd characters or numbers in boxes in the middle of words, this is becuase the original encoding does not match the encoding of the system you're using.

This same thing can happen in file names and even produce errors when trying to read files. For this reason, we want to work with the safest subset of characters available to us; those base characters recognized by all systems. This means not using special characters.

Special characters are all characters except:

- Any character that is a part of the English alphabet
- Numbers from `0 - 9`

- Dashes `-`
- Underscores `_`

A space is a special character, which means that your file names should not have spaces.

When operating in a multi-lingual or non-English environment, this can prove problematic, but it is an unfortunate legacy of the development of computer standards that has yet to be fully resolved.

File Naming Conventions

Order

Convention often has file naming proceed in the following order, with each element separated by an underscore `_`, and words within an element joined with a dash `-`. The file type is generally added with a period `.` and is usually automatically generated when an application creates a file. More elements can be added as needed.

Element-1	_Element-2	_Element-3	_Element-4	_Element-5	.Element-6
Last-Name	_Date	_Project	_File-Contents	_Version	.File-type

Dates

Dates should be written in the following format `yyyymmdd`. They should contain no spaces, no dashes, no words, just 8 numbers. Months and Days that are from `1 - 9` should be led by a `0`. For example, January 23, 2020, should be written **20200123**. When written this way, your computer will always sort your files from the earliest date to the latest date when sorted in ascending order.

Versions

Manual version tracking through file nameing convetions is achieved by adding `_Vn` where `n` is the version number. With each major change, we increase `n` by `1`. So version 1 would read `_V1`, and when updated, it would read `_V2`.

Versions are very important for things like manuscripts and interpretations of data, such as figures and other visualizations, which we will continue to change and modify or update throughout a project. Data, however, while it has a

collection date, should not be 'updated', and should not then be versioned. If new data sets are derived from the original data, they should be named and dated accordingly.

Version control through file naming is described in more detail in the manual version control section on version control.

Roles and Responsibilities

It's possible that there may be multiple projects occuring within your lab, and it's worth noting that these different projects don't need to adhere to the same naming conventions (in fact, depending on the projects, it may be preferable to have different conventions). However, what is important is that all files within a project are named consistently, and that there is documentation for file naming that all project members are introduced to when onboarded, and can easily be found for reference. Designating a person to create, introduce, and update the file naming documentation will be extremely valuable in keeping your project organized.

It's also worth noting that things can get chaotic from time to time, and consistent naming may begin to slip. This happens to the best of us, but designating a person to periodically check on files and update any inconsistencies is a great safeguard to keep things from slipping too far.

Questions

1. What files do you anticipate creating across the projects in your lab?
2. What baseline file naming conventions do you expect team members to use when creating files?
3. Who (what role) will be responsible of informing new lab members about file naming conventions to employ?
4. Who should questions about file naming be directed to?
5. Who is responsible for ensuring that file naming across projects is kept up to a minimum standard? How often will this be reviewed?

Directory Structures

While the way we currently organize folders and files on our computer may make sense to us, the structure may become promblamtic if and when we need to share these folders and files with other people. As well, a folder name that is meaningful for us may make no sense to another person.

Like with file naming, conventions for directory structures and directry names will help organize the files on your computer in a way that is meaningful to both you and others. It will also allow for programmatically parsing the files if needed.

General Rules

- Directory names should only contain letters in the English alphabet, numbers from `1-9`, dashes `-`, and underscores `_`.
- Do not use spaces or special characters, including `# % & < > : " / | ? * { } $ ! ' @ + =`
- Directory names should be broken down into components that are separated by underscores `_`.
- If more than one word is needed in each component, these are separated by dashes `-`.

Quick Reference

A naming conventions for common directories should be used:

- `Data/` for all data directories
- `Manuscript/` for all authored papers
- `Conferences/` for all files realted to conferences
- `Scripts/` for all scripts

Directory Hierarchies

The highest-ranking folder is generally called the root directory. This folder will contain all of the subfolders and files related to a particular project, including its data, analysis, manusrcipts etc. It will also contain what is called a `readme` file.

The structure should look something like the following:

```
Project-Folder/Data/File-1
Project-Folder/Data/File-2
Project-Folder/Analysis/File-1
Project-Folder/Maunscript/File-1
Project-Folder/Manuscript/File-2
```

Here, our root folder is called `Project-Folder` and it contains three subdirectories, one for data, `Data`, one for analyses, `Analysis`, and one for a manuscript `Manuscript`. Each subdirectory then contains one or many files.

Directory Naming

Key file naming conventions, such as avoiding special characters, are equally as important for directories as they are for naming files. A special character is anything other than letters in the English alphabet, numbers from `1-9`, dashes `-`, and underscores `_`.

When we name the root folder we want to clearly communicate what the project is. And similarly, within this root folder, we want clearly labeled subfolders for each relevant aspect of the project. Common discrete subdirectories include ones for figures, data, manuscript etc.

Example

LAB SPECIFIC

Roles and Responsibilities

Much like file naming, the key to keeping your directory structure organized is by being consistent and having regularly updated documentation.

Depending on your project team, it's possible that not everybody should have the ability to add folders at every level of the hierarchy. When new directories

are created, the person who creates the folder should be responsible for documenting the new addition in the appropriate place. There should also be a person who is designated to regularly check on the documentation and folders to ensure that everything is accurate, and to update any inconsistencies.

Questions

1. Will you:
    a. Use a common directory structure across all projects in your lab?
    b. Use a set of common directory structures across all projects in you lab?
2. What will the directory structures look like? Will there be a template that can be copied and pasted? Will the template include placeholder documentation files (readme, data dictionary etc)?
3. What will file naming conventions for diretories be?
4. Who (what role) will be responsible of informing new lab members about the directory structure to employ?
5. Who should questions about directory structure be directed to?
6. Who is responsible for ensuring that directory structures across projects are kept up to a minimum standard? How often will this be reviewed?

Version Control

Version control can be tackled in a couple of different ways.

Version control - although not using this wording - was arguably first born in the context of records management, reliant originally on prescribed annotations, but more recently on naming conventions and multiple copies – or versions – of documents being created and managed.

But software developers working in distributed environments really pioneered the way forward in how we track changes in a document and across a project, especially in a digital environment, using version control systems.

In research, version control comes into play in a variety of ways, sometimes in the context of software, sometimes data, as well as in our methods, protocols, and manuscripts.

Version control helps to keep you sane as you track changes through a project in a logical, stepwise manner. Version control also increases the transparency of your work, providing a window into the evolution of a document, concept, or project.

Undo vs Version Control

We should differentiate between version control for sanity and transparency and version histories for recovery or tracked changes.

Tracking every change made is more about recovery than versioning. Versioning instead is very much about capturing moments of significant change or phases of implementation.

When we think recovery, it's more like having an undo function: the whoops moment, I made a mistake — I didn't mean to delete that paragraph or I didn't mean to put that file in the trash — let me undo that.

When we think of version control, we should be thinking of a timestamp to capture significant decisions made in the transition or evolution of a project, ideally accompanied by a description of what these changes were.

This is a continuum, and the difference between recovery and version control is more nuanced than this, but this dichotomy helps to highlight the role of version control in tracing change over time versus recovering from a mistake. In fact, wedding these two concepts together in the implementation of a version control system can provide for the best of both worlds.

Manual

Manual version control is handled by file naming convetions, and was touced on in the section on file naming.

Depending on the file type, or data / information held within a file, the file naming conventions used may differ. For files where a date is extremely relevant, such as with data, the date may form the primary means of versioning. This may then be supplemented with a version number to indicate modifications to the date versioned file.

For example, say I've collected data over several days, each collected in it's own file. I might have something like the following using our file naming conventions:

```
VisDunbar_20210901_SMS-DB-Prev_P-Data.csv
VisDunbar_20210905_SMS-DB-Prev_P-Data.csv
VisDunbar_20210912_SMS-DB-Prev_P-Data.csv
```

Now, I should keep an unmodified version of each, so I might call those V0 to supplement the date:

```
VisDunbar_20210901_SMS-DB-Prev_P-Data_V0.csv
VisDunbar_20210905_SMS-DB-Prev_P-Data_V0.csv
VisDunbar_20210912_SMS-DB-Prev_P-Data_V0.csv
```

You may wish to set the file permissions on your V0 data to read only to ensure that they cannot be edited accidentally.

I might then merge and go through and standardize the data, ensuring any blank values are set to `NA` and all character data matches on upper and lower cases. To do this, I would make a copy of each V0, indicate that these are the working copy somehow, maybe appeninding `V1`, and then merge them and make these changes to a new file that captures this information in a meaningful way, updating the date component to reflect the full span and resetting the version to `V0`.

```
VisDunbar_20210901_SMS-DB-Prev_P-Data_V0.csv
VisDunbar_20210905_SMS-DB-Prev_P-Data_V0.csv
```

```
VisDunbar_20210912_SMS-DB-Prev_P-Data_V0.csv
VisDunbar_20210901_SMS-DB-Prev_P-Data_V1.csv
VisDunbar_20210905_SMS-DB-Prev_P-Data_V1.csv
VisDunbar_20210912_SMS-DB-Prev_P-Data_V1.csv
VisDunbar_20210901-20210912_SMS-DB-Prev_P-Data_V0.csv
```

If there are meaningul ways in which the amalgamated data should be parsed for analysis, I may choose to update the version number to capture these modifications to the data file. In this scenario `V > 0` indicates a working copy and `V = 0` indicates a copy of record.

```
VisDunbar_20210901_SMS-DB-Prev_P-Data_V0.csv
VisDunbar_20210905_SMS-DB-Prev_P-Data_V0.csv
VisDunbar_20210912_SMS-DB-Prev_P-Data_V0.csv
VisDunbar_20210901_SMS-DB-Prev_P-Data_V1.csv
VisDunbar_20210905_SMS-DB-Prev_P-Data_V1.csv
VisDunbar_20210912_SMS-DB-Prev_P-Data_V1.csv
VisDunbar_20210901-20210912_SMS-DB-Prev_P-Data_V0.csv
VisDunbar_20210901-20210912_SMS-DB-Prev_P-Data_V1.csv
VisDunbar_20210901-20210912_SMS-DB-Prev_P-Data_V2.csv
```

For files where date is of less relevance, such as a manuscript, a version number may be the primary means of versioning changes. This may then be supplemented with a date and / or an initial to indicate modifications to this version. So, say we have the following file

```
VisDunbar_SMS-DB-Prev_Manuscript_V0.docx
```

I only want to update the version between major edits, so if I distribute the file for review by my co-authors, I would expect those to be returned to me with dates and initials so that I can keep track of things.

```
VisDunbar_SMS-DB-Prev_Manuscript_V0.docx
VisDunbar_SMS-DB-Prev_Manuscript_V0_20210902_MM.docx
VisDunbar_SMS-DB-Prev_Manuscript_V0_20210910_NR.docx
```

After recieving these suggestions and edits, I would copy and rename `V0` to `V1` and carry on implementing the edits into `V1`.

```
VisDunbar_SMS-DB-Prev_Manuscript_V0.docx
VisDunbar_SMS-DB-Prev_Manuscript_V0_20210902_MM.docx
VisDunbar_SMS-DB-Prev_Manuscript_V0_20210910_NR.docx
VisDunbar_SMS-DB-Prev_Manuscript_V1.docx
```

For some files it may be necessary to indicate a working copy. For instance, a script to process data may have a current working version, but you want to test something out. This test should not be done directly to the working version, but it also hasn't passed quality control and ready to be released as the next version. One way to implement this is by appending additional information onto the version. So say I have the following script

```
VisDunbar_SMS-DB-Prev_Data-Clean_V2.docx
```

And we have an idea to update it, to test this out I would copy `V2` and then indicate that this copy is a working copy by appending a `1`

```
VisDunbar_SMS-DB-Prev_Data-Clean_V2.docx
VisDunbar_SMS-DB-Prev_Data-Clean_V2-1.docx
```

or a `W` for `working`

```
VisDunbar_SMS-DB-Prev_Data-Clean_V2.docx
VisDunbar_SMS-DB-Prev_Data-Clean_V2-W.docx
```

Once the working copy is shown to work and has been ok'd by the team, I'll rename it to the next version

```
VisDunbar_SMS-DB-Prev_Data-Clean_V2.docx
VisDunbar_SMS-DB-Prev_Data-Clean_V3.docx
```

```
If you anticipate there ever being more than 10 versions of any document, ensure that y
```

Automatic

Automated version control systems save us from having to update file names. Instead the metadata that manual naming systems use to record version information is stored elsewhere as part of the system managing the versions.

For some work, or some projects, you may be asked to work with platforms like OSF or GitHub. The former is desigbned to provide you with version control for binary files - files like Excel, Word, PDF etc - while the latter is designed for plain text files.

Using these systems is beyond the scope of this manual, suffice to say, that manual naming conventions as they relate to versioning can be ignored.

Implementation

Version protocol	Meaning
V0	Copy of record. Do not edit.
V1…Vn	Staged version. Represents a significant change or update.
Vn_YYYYMMDD_FL	Working document of verson n, includes dates and First name Last Name initials of editor

Roles and Responsibilities

Whatever system, or systems, of version control you decide to use, ensuring that there is someone in the group to document and update the relevant procedures. Also, designating a team member to periodically check for and correct any inconsistencies is very helpful in keeping project data and files organized.

Questions

1. What system of version control will you use for:
    a. Manuscripts
    b. Code
    c. Data
    d. Documentation
    e. Figures and images
2. For manual naming conventions, what convention will you use for
    a. Manuscripts
    b. Code
    c. Data
    d. Documentation
    e. Figures and images
3. How will you ensure that raw data files are not edited / worked on?

File Formats

For every research project, it's important to consider which file formats you'll use to save and store your data and files. In some instances, the file formats you use will be dictated by software, instruments, or other factors in your research project or field, but in other cases, you may have the option to choose between several different formats. The type of data that you're working with will also play a factor in determining the file formats you use, but considering your workflows and project goals should play a large role in selecting formats.

File Format Best Practices

- If possible, collect data in open and non-proprietary formats
- If not possible to collect open / non-proprietary formats, try to pick a format that can be easily converted to an open format
- Consider software and formats that are widely used within your discipline
- Be aware of software, hardware, and licensing requirements for viewing and working with data
- Be aware of using elements such as colour or highlighting as a form of metadata, as they may be lost if converted to a different format

0.0.1 Recommended formats for sharing, resuse, and preservation

Data Type	Recommended Formats	Accepted Formats
Tabular data with extensive metadata (variable labels, code labels, defined missing values	.por (SPSS portable format)	.sav, .dta, .mdb,.accdb
Tabular data with minimal metadata (column headings, variable names)	.csv, .tab	.txt, .xls, .xlsx, .mdb, .accdb, .dbf, .ods
Textual data	.rtf, .txt, .xml	.html, .doc/.docx
Image data	.tiff (TIFF 6.0)	.jpeg, .jpg, .jp2, .gif, .tif, .tiff, .raw, .psd, .bmp, .png, .pdf
Audio data	.flac	.mp3, .aif, .wav
Video data	.mp4, .ogv, .ogg, .mj2	.avchd
Documentation and scripts	.rtf, .pdf, .xhtml, .htm, .odt	.txt, .doc/.docx, .xls, .xlsx, .xml
Geospatial data (vector and raster data)	.shp, .shx, .dbf, .prj, .sbx, .sbn, .tif, .tfw, .dwg, .gml	.mdb, .mif, .kml, .dxf, .svg
Chemistry data (spectroscopy)	.jdx (JCAMP)	

Source: Oregon State University

0.0.2 Raw Data

Raw data referse to data that has been gathered or collected in its initial state, and has not yet been processed, cleaned, organized, or analyzed in any way. Depending on the type of data you are working with and how its generated, you may or may not have the option to choose your raw file formats (for example, if your data is dependent on instruments or software).

General recommendations for raw data file formats: * Save a copy of your raw data in its original format, and do not alter or clean in any way. In case things go sideways, it's always good to have to have a fallback. * If possible, save a copy of your raw data in one of the recommended formats (preferable) or accepted formats above.

0.0.3 Processed Data

Processed data refers to data that has been cleaned, organized, and or manipulated in ways that allow the data to be analyzed. When actively processing data, select file formats that allow you and your team to to easily share, open, and read files. If you are creating any code or scripts to clean or analyze your data, follow the same guidelines in regards to the formats in which they are saved. For both your data and code, if you are using proprietary or non-open formats, consider how you may convert these files to the recommended formats above once processing is complete.

0.0.4 Data for Deposit

Deposited data refers to data that is place into a digital repository for long-term archiving. For this type of data, the recommended formats from above are most condusive to the long-term access of your data and files.

Manuscrips

When preparing and working on manuscripts, one of the most common file formats is .doc/.docx, which generally requires Microsoft Word in order to create, edit, or view, but other applications like Open Office and LibreOffice can read these files as well. While these are proprietary formats, they are widely accepted formats and are thus in the 'accepted formats' category.

Depending on the nature of your manuscript and how you intend to develop, edit, share, and disseminate, the formats in the "Textual data" section are all viable options.

Documentation files

Documentation files, such as READMEs and data dictionaries, should be written in plain text so that they can be easily opened by any computer at any point in time. .txt is a popular plain text format, and widely used for documentation files and other instructional documents.

In addition to .txt files, Markdown, which is saved as .md files, is an authoring language that makes it easy to create HTML documents without having to type all the extra characters of HTML. Markdown can be read by any application that will open a .txt file, and is increasingly being used for technical documentation. With that said, there may be systems that will not render an .md file with the desired formatting, so depending on your workflow and how you plan to disseminate and potentially archive your work, this may not be the ideal file format.

Questions

- What file formats will be used for what content types?
- Are any of your files bound to specific software or instruments? If so, do all your collaborators have access to these?
- Have you looked into how to convert any proprietary file formats you may have into non-proprietary formats?

Documentation

When naming files we embed metadata into our file naming conventions to encode relevant information for the reader. But we can only store so much information in a file name. So we also include three additional files:

- A `_README` file which resides in our **root folder** and elaborates on the contents of our folder structure.
- A `_README` file that resides in our **data directory** and discusses some of the particulars of the how, where, and who did the actual data collection.
- A `_DATA-DICTIONARY` file that also resides in our **data directory** and elaborates on how our data is stored and organized.

These files - containing a brief description of the major folder contents, naming conventions, and data structure - are critical for transparency and reproducibility because they allow others to easily understand the contents of your directory and data without needing to ask you. This is especially helpful when working with a group or sharing directories with others.

General Rules

A readme file and data dictionary should:

- Exist in at least two locations, the root directory and the data directory.
- Be prepended with an underscore `_`. This will push these files to the top of the directory for easy access.
- Be written as plain text.
- `_README` and `_DATA-DICTIONARY` files should be in all caps, so they really stand out; this should be the first thing you look at when looking at any directory or folder, as this is your guide to its contents.

Occasionally, certain systems and applications (such as GitHub), will not properly recognize a readme file that is preceded by an underscore _. In these situations, the underscore should be ommitted.

Readmes

File Formats

Readme files and data dictionaries should be written in plain text; this will ensure that the files describing your project can be opened on any computer at virutally any point in time. A word processor document from 15 years ago might be a challenge to open today. A plain text file from the 1980s won't have the same issue. You will often see readme files called `_README.txt` or `_DATA-DICTIONARY.txt`

Markdown

While plain text works well, using a markup language in plain text can be beneficial. Markup is a way of formatting plain text files, allowing us to provide additional meaning to our content. For example, in plain text, if we want to emphasize content, we don't really have a way of doing this. In a popular markup language, Markdown, we can use italics and bolding if needed. We can also create lists and tables. So you kind of get the best of both worlds.

Learn the basic syntax of Markdown here.

Root Folder

To create a root folder `readme` file, use any markdown or text editor (e.g. Typora, notepad etc.), open a new file and save it to the root folder for your project, ensuring the file type is `.md`.

Name your readme file `_README.md`.

Next, we want to add some content to our `_README.md`. The purpose of this document, at a bare minimum, is to describe the directory structure of our project. To adequately describe our directory structure we should include:

- A brief description of the project or purpose of the root folder
- Date when the root folder was created and who created it
- Date when the readme file was last updated and who updated it
- A brief description of the contents of each major folder within the root folder
- A brief description of file naming conventions used within the directory

To see an example root directory readme file click here.

Data directory `readme`

Next, we want to create another `readme` file, but this file will be placed within the subfolder that contains our project's data. To do this, open any markdown or text editor (e.g. Typora, notepad etc.), open a new file and save it to the data subfolder for your project, ensuring the file type is `.md`. Name your readme file `_README.md`.

The purpose of this readme file is to provide a description of data collection methods. We will include, at a minimum:

- Date when the data directory was created and who created it
- Date when the readme file was updated and who updated it
- A brief description of each data that was collected, the methods used for collection, and the date range for when each dataset was collected
- A description of who was involved in data collection
- A brief description of where the data was collected

To see an example data directory readme file click here.

Data Dictionaries

Lastly, we need to create a data dictionary which elaborates on how our data is stored and organized. To do this, open any markdown or text editor (e.g. Atom, Typora, notepad etc.), open a new file and save it to the data subfolder for your project. This time we will save the file as `_DATA-DICTIONARY.md`.

A data dictionary helps others understand the meaning of each element in your datasets within the broader context of the project. Typically you will have an individual readme file for each dataset. This file should include:

- Date when the data dictionary was created and who created it
- Date when the data dictionary file was updated and who updated it
- A description of the raw data file
- A description of each variable for all datasets including data type, units, number of levels if categorical, and a description of variable levels where relevant
- When describing variables you need to provide the full names and definitions of each variable because often variables are abbreviated in datasets

To see an example data dictionary click here.

Roles and Responsibilities

It's important to remember that having a documentation system that your team can easily update throughout a project is more benefical than one that begins with great detail, but is difficult to maintain. As the project grows and new files and folders are created, clearly identifying what information team members need to document, and where that information needs to be put, will help keep things on track. It is also worth considering having a person who periodically will check the documentation to ensure that it's accurate, and can update any inconsistencies.

Questions

1. What baseline documentation are team members expected to create? (readme, data dictionary, code books, code documentation…)
2. Where should these documentation file reside?
3. What would a template for each of these look like? Or what baseline information would you expect each piece of documentation to include?
4. What format should this documentation be stored in?
5. Who is responsible within a given project for ensuring adequate documentation is created and maintained?
6. Who is responsible for ensuring that documentation across projects is kept up to a minimum standard? How often will this be reviewed?
7. Where will roles and responsibilities for lab processes be tracked?
8. Where will roles and responsibilities for specific projects be tracked / articulated?

Permissions & Ownership

Managing access and permissions at the project, directory, and file level is not an easy task. However, not everyone involved in a given project needs or should have access to every aspect of that project.

When preparing for a research project, and mapping out the directory structure and storage locations that will be used, we should think about the following roles and which should have access to what parts of a project:

- Primary investigator
- Lead researcher
- Lab manager
- Research collaborators
- Graduate assistants
- Undergraduate assistants
- Funders
- Editors and Reviewers
- The public

Depending, some of these actors may not be relevant, and some may need further breakdown when thinking about roles in the research project and access and permission needs.

Most systems allow three levels of access:

- Administrator or owner
- Read and write
- Read only

It's worth considering two other potential levels of access:

- Knowing something exists
- No knowledge

Generally, **administrators** have full control over what ever they preside on, wheterh this be at the project, directory, or file level. At the project level, they can generally invite and remove collaborators and make public or delete files. **Read and write** will allow a user or person to open (read) and edit (write) existing files. Depending on the system, read and write may allow users to delete or make public files and directories. It may also allow them to create new files. Generally read and write does not allow invitation or deletion of collaborators. However, it's always wise to ask these questions of whatever storage solution you're using to house your research. **Read only** generally only allows for the opening of a file, without the ability to edit, delete, or invite additional collaborators.

Even when someone is unable to read the contents of a file, we may wish to make them aware of its existence. Publishing, or making publicly available, a copy of your directory structure, is one way of doing this - one may be able to see the names of individual files and perhaps a `readme` file that describes the contents of the file, but does not actually reveal these contents. This may be useful in a data sharing scenario, where you may be willing or in a position to share your data upon request; a request can only come if someone knows there is something there to request.

When all data and metadata are stored on private media or are unpublished, non-collaborators will have zero knowledge of any of your research data or other files.

When prepaing for a project, access permissions for directories should be articulated based on collaborator roles. If needed, and if using a system that supports file level permissions, these too should be considered and articulated. A registry should then be maintained, listing all collaborators, their roles, and what they have access to. When leaving a project, collaborators should have their permissions and access updated.

Roles and Responsibilities

Questions

Data Storage

Data Storage is a term made up of three components. In this section we will walk you through determining the amount of storage, where to store your active research data, and finally, storing your data for the long term, generally known as data preservation. Backing up your data, while also a storage issue, is handled separately.

Data storage is important but becomes urgent when you don't have enough space to store all your data files. This is especially true if you are part way through a research project and are unable to save and/or capture data because you have run out of data storage space. If you are uncertain how to estimate the amount of storage you require, people in Research Computing and Advanced Research Computing are available to help you quickly and easily determine this.

General Rules

1. Know what the subject of your data is as this will determine where you can store it and whether or not additional security measures are recommended or required
2. Determine approximations for how much data your project will generate
3. Outline whether your collaborators (if any) are from within UBC or outside of UBC
4.

- Determine what level of security your files and data require. Working with human data *may* require different storage than data that is not of human origin or relating to human participants or subjects.
- Allocate storage for original (raw) data, cleaned or organized data, processed and analyzed data, documentation that informs users about your data, draft papers, charts, visualizations, photos, ans supplementary items that go within your papers. You will likely also need some, generally small, amount of space for the admimistrative documents of your project such as grant applications and the like.

- Allocate sufficient storage in different locations for back-ups.
- Arrange for sufficient or additional storage well before it is actually required. Sometimes you have instant access to needed additional storage, and in other cases, it takes time to gain access.
- Decide who needs access to which files. Principal Investigators (PIs) and lab managers often have access to all the files at all times while students beginning work in the lab may only have access to the files into which they are entering data. Graduate students and post docs may have access to more files. Record who has access to which files. Record all the locations (specific drive, or other exact location) and who has access to which locations.
- Apply standardized file names to all your files (see Chapter 2) and directory structures (see Chapter 3) to all storage locations

Depending on what types of information the file contains, how exactly these guidelines are implemented will differ slightly.

Quick Reference

Type	Stoage Location
Data about non-humans	**Teams** Good for collaborating on documents.**OneDrive** Cloud Storage good for many kinds of data and file formats
Non-sensitive Human data	**Teams** Good for collaborating on documents.**OneDrive** Cloud Storage good for many kinds of data and file formats
Big Data	**UBC Administered** Sockeye or Chinook.**Contact** Consult an ARC consultant about your requirements. www.arc.ubc.ca **Compute Canada** Consult an ARC consultant about your requirements. www.arc.ubc.ca

Storage locations NOT recommended by UBC

- **Drop Box** Cloud storage servers are located in the US and do not provide meet the Freedom of Information and Protection of Privacy Act (FIPPA) requirements for UBC researchers.

- **Google Drive** Cloud storage servers are located in the US and do not provide meet the Freedom of Information and Protection of Privacy Act (FIPPA) requirements for UBC researchers.

How *many* copies of files and the locations of these copies matters.

A **best practice** for files is to have ***3*** copies in ***2*** formats (i.e. laptop harddrive and removeable harddrive) ***1*** stored off-site

File Naming Conventions

Utilize a standardized file naming convention as outlined in the File Naming section.

Roles and Responsibilities

You may have multiple projects occuring within your lab concurrently and it's worth noting that these different projects may have different storage requirements. Additionally, file naming conventions will become specifically more important so files for separate projects do not become mixed.

Questions

1. Where will active project storage occur? (laptop, lab computer, OneDrive, Teams etc)
2. Where will active projects be backed up to? How often will back-ups be run?
3. At project completion, how long will the following be kept (UBC minimum requirements, other…)
    a. Data
    b. Code
    c. Manuscripts and posters
4. At project completion, where will files be stored?
5. At time of publication / presentation, where will files be shared? (Dataverse, OSF, OneDrive etc.)
6. Will the lab use Dataverse for data deposit?
    a. What will the structure of Dataverse look like?
7. Will you use disk level encryption? If so, whe, and what service?
8. Will you use file level encryption? If so, when, and what service?

Offboarding

Offboarding is a systematic process to ensure that when people leave your team you have all the information you require to continue your research in an orderly and smooth fashion.

The longer the amount of time your research project operates, the more likely that you will have people, such as students at various levels, join and leave the research team. Having a standard process for offboarding people will prevent or reduce:

- People leaving and taking needed passwords and/or files with them.
- Time lost locating files required for continued work.
- Possible file security breaches in the future.

This is especially helpful when working with a group or sharing data with others. Concretely outlining the steps everyone must take before they complete their term is good practice.

General Rules

An offboarding guide should contain:

- Any already existing administrative requirements for offboarding such as returning keys, and expiring accounts to specific computer drives or storage locations. Avoid duplicating procedures that already exist and that are sufficient for your purposes.
- Ensure the offboarding process be completed by every person prior to ending their term on your research project or in your research lab.
- All offboarding documents should be reviewed by either the lab manager, the PI, or their designate at least 5 days in advance of the final working day of the person leaving.
- Be written as plain text.

- Contain a checklist of actions specific to your research, including but not limited to
    1. Locations of all files worked on that are pertinent to the project and any passwords created by this person to secure those files.
    2. Are files accessible by other members of the team
    3. Specific feedback about the research project that might be useful to your team or lab in the future.
    4. Recommmendations from the person leaving about any processes or procedures that were good, that did not work, or that were outdated.
    5. Future contact information, if needed.

Roles and Responsibilities

It's important to remember that having a documentation system that your team can easily update throughout a project is more benefical than one that begins with great detail, but is difficult to maintain. As the project grows and new files and folders are created, clearly identifying what information team members need to document, and where that information needs to be put, will help keep things on track. It is also worth considering having a person who periodically will check the documentation to ensure that it's accurate, and can update any inconsistencies. The offboarding documents can be a valuable source of information to assist in keeping the documentation system up to date.