

# **HDBI Data Resource**

**Data: Inception to Publication & Beyond**

Richard J. Acton

2023-01-31

# Table of contents

<b>About this resource</b>	<b>6</b>
Why to use this Resource . . . . .	6
How to use this resource . . . . .	10
How to contribute to this resource . . . . .	11
<b>1 What Constitutes Data?</b>	<b>13</b>
1.1 A brief overview of sources/types of data . . . . .	13
1.1.1 Results Outputs . . . . .	13
1.1.2 Process Outputs . . . . .	14
1.2 Metadata . . . . .	14
1.2.1 Ontologies & Controlled Vocabularies . . . . .	17
1.2.2 Metadata models, Schemas, and Standards . . . . .	19
1.2.2.1 The ISA (Investigation, Study, Assay) metadata model . . . . .	24
<b>2 When Should I Generate Data?</b>	<b>26</b>
2.1 When to speak to data analysts . . . . .	28
2.2 Exploratory analyses are not exempt from proper planning . . . . .	29
<b>3 How To Store Your Data</b>	<b>33</b>
3.1 Estimating capacity . . . . .	34
3.2 Categorizing / Stratifying your data . . . . .	35
3.2.1 Deleting Data . . . . .	38
3.2.1.1 Have Clear Policies on Data Deletion . . . . .	39
3.2.2 Practice Recovery . . . . .	41
3.2.2.1 Personal device loss . . . . .	41
3.2.3 Storing Code . . . . .	42
3.2.3.1 Making your git repo citable . . . . .	48
3.3 gitlab . . . . .	49
3.4 github . . . . .	51
3.5 Choosing Storage . . . . .	54
3.5.1 “The Cloud” . . . . .	54

3.5.2	Central/Institutional Storage Resources . . . . .	55
3.5.3	Operating Your Own Storage . . . . .	56
3.5.3.1	Redundancy and Raw vs Usable capacity . . . . .	57
3.5.4	Cost, Availability, & Integrity . . . . .	59
3.5.4.1	Data Availability . . . . .	60
3.5.4.2	Data Integrity . . . . .	61
3.6	Securing Data . . . . .	62
3.6.1	<b>What is Multi/Two-factor Authentication?</b> . . . . .	64
3.6.2	<b>What 2nd factor should I use?</b> . . . . .	65
3.6.3	Setting up a password manager & MFA . . . . .	66
3.6.4	Video Summary . . . . .	72
3.7	Naming Files . . . . .	72
3.7.1	Dates . . . . .	72
3.7.2	No spaces or special characters! . . . . .	73
<b>4</b>	<b>Working With Data</b>	<b>76</b>
4.1	Electronic Lab Notebooks (ELNs) . . . . .	76
4.1.0.1	eLabFTW . . . . .	79
4.1.0.2	openBIS . . . . .	80
4.1.0.3	OSF . . . . .	80
4.2	Code vs GUIs for provenance and repeatability . . . . .	81
4.3	Reproducible computational analyses . . . . .	83
4.3.1	Source Management . . . . .	84
4.3.1.1	Literate programming . . . . .	87
4.4	Jupyter/Quarto & vscode learning resources . . . . .	91
4.4.0.1	Using git . . . . .	91
4.5	Git learning resources . . . . .	100
4.5.1	Environment management . . . . .	100
4.5.1.1	Package & Environment management tools . . . . .	101
4.5.1.2	Containers . . . . .	103
4.5.2	Dataset Management . . . . .	104
4.5.2.1	Your Own Raw Data . . . . .	104
4.5.2.2	Raw Vs. Processed Data . . . . .	105
4.5.2.3	Public Data . . . . .	106
4.5.2.4	Large files in git repos . . . . .	107
4.6	git-lfs learning resources . . . . .	107
4.6.0.1	Imaging datasets . . . . .	108

4.6.1	Pipeline/Workflow Management . . . . .	110
4.6.1.1	Pipeline management tools . . . . .	110
4.7	Pipeline tools learning resources . . . . .	113
4.8	Renku - Bringing it all together . . . . .	114
4.8.1	Why Renku? . . . . .	114
4.8.2	Getting started with Renku . . . . .	115
4.8.2.1	Account setup . . . . .	115
4.8.2.2	Your first project . . . . .	116
4.8.2.3	Templates . . . . .	116
4.8.2.4	Running Renku Sessions Locally . . . . .	116
4.8.3	Renku Learning Resources . . . . .	117
<b>5</b>	<b>When To Publish Data</b>	<b>118</b>
5.1	The Fort Lauderdale Principles . . . . .	118
5.2	Registered Reports . . . . .	121
5.3	Counterpoints to publication on data generation . . . . .	124
5.3.1	Poor Quality data . . . . .	124
<b>6</b>	<b>Where To Publish Data</b>	<b>126</b>
6.1	Searching for Research data repositories . . . . .	126
6.2	Selected Public Data Repositories & Data Sharing Platforms . . . . .	126
6.2.1	Sequencing Data . . . . .	126
6.2.1.1	GEO (Gene Expression Omnibus) . . . . .	126
6.2.1.2	SRA (Sequence Read Archive) . . . . .	127
6.2.1.3	HCA (Human Cell Atlas) data portal . . . . .	127
6.2.1.4	GenBank . . . . .	128
6.2.1.5	ENA (European Nucleotide Archive) . . . . .	128
6.2.2	Imaging Data . . . . .	129
6.2.2.1	IDR (Image Data Resource) . . . . .	134
6.2.2.2	Bioimage Archive (EBI) . . . . .	134
6.2.2.3	EMPIAR (Electron Microscopy Public Image Archive) . . . . .	134
6.2.2.4	figshare . . . . .	134
6.2.3	Protocols . . . . .	134
6.2.3.1	Protocols.io . . . . .	135
6.2.3.2	JOVE . . . . .	135
6.2.4	Code & Computational Environments . . . . .	135
6.2.4.1	Software packages . . . . .	135

6.2.4.2	Bioinformatic analysis pipelines	136
6.2.4.3	Scripts, Notebooks and project specific workflows can be shared as git repositories. . . . .	136
6.2.5	Biological Materials access / Sharing . . .	137
6.2.5.1	HDBR (Human Developmental Biology Resource) . . . . .	137
6.2.6	Spatial transcriptomics . . . . .	138
6.2.7	Flow Cytometry . . . . .	138
6.2.8	Proteomics . . . . .	138
6.2.9	None of the above . . . . .	139
6.3	Integrated publishing - a possible future . . . . .	139
<b>7</b>	<b>What Data To Publish</b>	<b>141</b>
7.1	Additional Ethical & Legal Considerations . . . . .	142
<b>8</b>	<b>How To License Your Data</b>	<b>145</b>
8.1	Data . . . . .	145
8.1.1	Images . . . . .	146
8.2	Software . . . . .	147
8.2.1	Quick Primer on choosing a software license	148
8.2.1.1	Recommendations . . . . .	148
8.3	Retaining Rights . . . . .	151
8.4	Resources . . . . .	152
<b>9</b>	<b>How To Manage References</b>	<b>153</b>
9.1	Zotero . . . . .	153
9.1.1	RSS feeds in Zotero . . . . .	154
9.1.2	Zotero Web . . . . .	155
9.1.3	Extensions & Integrations . . . . .	156
9.2	Personal Knowledge Management . . . . .	158
9.2.0.1	Open Source Options . . . . .	158
9.2.0.2	Some popular proprietary r emo::ji("disappointed") solutions . . . . .	160
	<b>HDBI Logos, Links, &amp; Communities</b>	<b>162</b>
	Code Repositories . . . . .	162
	Protocols . . . . .	162
	Publications . . . . .	162

# About this resource

DOI 10.5281/zenodo.8021381

(a9e2a07) [2024-04-11]

**These are our aspirations for making the data generated by the HDBI adhere to the [FAIR](#) principles.**

Datasets should be accompanied by structured metadata which provides the details of who generated the data, how and for what purpose. It should also include a detailed account of the experimental design and how the data files map onto this design. Files should be identified with publicly available unique identifiers with which they can be retrieved from a repository wherever possible. Wherever suitable controlled vocabularies and ontologies exist these should be used to refer to methods, organisms, medical conditions and other such annotations. Any methods used in the generation of data should reference protocol level details and any sources of biological materials be clearly identified so that the data generation process could be readily reproduced with the same materials and processes. Any secondary data which are the product of analyses of primary data should be published with reference to openly licensed code which produced these outputs. The computational environment in which the code was run should also be described ideally in a form which permits its automated reproduction and details exact software versions.

## Why to use this Resource

“Ask not what you can do for reproducibility; ask what reproducibility can do for you!”

- Florian Markowetz

Firstly work which takes place as a part of HDBI *MUST* meet the stipulations for research data provided by our funder, Wellcome. Wellcome's Policy consists of 6 main points summarized below.

**i** Wellcome Data, software and materials management, and sharing policy summary

1. Maximise the availability of research data, software, and materials. At a minimum those underpinning research publications.
2. Have a plan for managing and sharing research outputs, especially those which might serve as a general resource of use to the wider academic community.
3. Data should be discoverable, shared using persistent identifiers according to the most recent community standards, and in appropriate repositories.
4. Users of research data, software and materials should cite those resources and abide by the terms and conditions of any resource used.
5. Wellcome recognizes a range of research outputs including: inventions, datasets, software, translation to health applications, and materials, in addition to conventional publications when assessing researchers.
6. **Successful sharing of research outputs will be considered critical by Wellcome in assessment of end of grant reporting.**

Beyond their data management guidelines, It is also worth reviewing Wellcome's guide on completing an [outputs management plan](#). Anyone applying for further funding from Wellcome might find it helpful to review Chapter 3 - [How to store your data](#) of this guide when writing a grant proposal so that they can request suitable funding for research data storage.

This guide aims to help you not only to meet but to exceed Wellcome's requirements in ways that benefit your research. It

is intended to provide the resources needed to make it as simple as possible for HDBI researchers to reach these goals.

A substantial portion of this guide relates to working reproducibly, either directly or indirectly. **The best practices for the care and feeding of research data, from choosing the right tools store and organised it to the right time and place to release it are important foundations for reproducible science.**

### **i** Reasons to work reproducibly

Florian Markowetz outlines why **working reproducibly is a good idea for pragmatic and self-interested reasons** beyond high scientific ideals in his excellent short piece: ‘[Five selfish reasons to work reproducibly](#)’ (Markowetz 2015 [cito:agreesWith] [cito:discusses]), I will briefly recap/paraphrase these here:

#### 0. **Idealism**

The ability of others to reproduce our work is a foundational idea in the philosophy of science ... and whatnot. Inspiring but not always motivating to action in the mundane day-to-day.

#### 1. **Avoidance of disaster**

The easier it is for you and your collaborators to understand and cross-check your work the easier it is for co-authors to spot errors (or even fraud) before you publish them and tarnish your reputation.

Not loosing years of hard work if you loose your laptop because you have things backed up properly - *you do have things backed up properly right?* .

#### 2. **An easier time writing papers**

You can avoid the laborious error prone process of transcribing numbers and updating figures when you change a data cleaning step and just have it



all update automatically at the click of a button. You may even be able to avoid having to manually reformat things to different journal's persnickety layout requirements.

### 3. **Easier for reviewers to see things your way**

Conversations with reviews tend to be more constructive when they can actually see what you did and even better if it's easy for them to understand and poke it themselves to see if they can find anything wrong with it.

It is much easier to avoid talking at cross-purposes when you have a common set of concrete facts. An imprecise description can confuse what you did, code & protocols clarify this.

### 4. **Continuity of work**

The ability to actually pick-up where you or someone else left off rather than spending months redoing stuff that's already been done but not adequately documented.

The ability to run old code on a different computer, or use an old protocol in a new lab and still have a good chance of it working.

### 5. **Building A Reputation**

People Know your work is in good faith because they can actually see it, even if it's not perfect at least they know your honest.

People will like working with you and your published data better because it's easy to use, a good dataset or software package can translate into a lot of citations.

If you publish, data and details of your analysis with every paper that's two more DOIs than normal and more published artifacts on a CV = CV more better ?

Learning more about the technical and systemic barriers to open and reproducible work as well as the tools and solutions to facilitate the adoption these workflows has been an interest of mine for about the last 8 years<sup>1</sup>, or more or less since the start of my academic career as a PhD student. This has progressed to the point where that number of years was calculated on the fly from the difference between the current date and the approximate date at which I started my PhD so that it will remain accurate in future revisions. I'm aiming here to codify as many of my learnings as possible and make them accessible to you so that you can do better than I have managed to do so far and avoid some of the pitfalls I've encountered along the way. As such it is at present based largely on my experience and influenced by my opinions, if & when this project gets more contributions that may change. A good resource with more contributors that is organised less linearly and covers more domains is [RDMkit](#) (Research data management kit) from the [ELIXIR-CONVERGE](#) project which aimed to "help standardize life science data management across Europe".

## How to use this resource

It should be **possible to consume this document out of order referring only to the sections relevant to your problem(s)**, cross-references will be present when another section provides useful context. Though it does aim to be readable in it's entirety by a general reader with an academic/technical background.

---

<sup>1</sup>Assuming that I have continued to be interested in this subject since I wrote this, which seems likely at time of writing .

### Searching this resource

**Press k to summon the search box** Or click the magnifying glass above the table of content in the top left of the page.

You can share a search by clicking the copy icon in the search dialog.

You can share a link to a section by using the link icon that

appears next to section headers when you hover over them.

### Collapsed blocks

You will find occasional ‘callout’ blocks like this colored box in the body of the text which are collapsed by default. You can click to expand them and read their contents. Sometimes when I’m delving into technical specifics I’ve hidden these by default so that they are available to anyone who needs them but won’t disrupt the flow for the general reader.

In each section there will be an **overview of the topic with links to external sources covering these topics in additional detail**. I aim to include **text and video resources on each topic** to suit different people’s preferences for the media from which they learn best. In addition I will attempt to provide **sources which range from basic/quickstart/TLDR introductions** to a topic for newcomers **to longer form and deeper dive content** so that beginners and advanced users all have something of value to refer to in each section. Another type of content I’ll aim include are high level overviews useful both to newcomers before they dive into the details and supervisors/collaborators who just need a high level summary to understand why the specialists they are working with are using these tools/methods and how to talk to them about it.



An example of a similar resource to this book which covers some of the same ground and a number of other areas not addressed here is [The Turing Way](#) a collaboratively authored book from [The Alan Turing Institute](#).

## How to contribute to this resource

The [source for this document](#) can be found in the HDDBI group on [renkulab.io](#). Input, feedback and suggestions are welcome. Anyone wishing to tell me that I am wrong and/or stupid/ignorant for anything I have written here is warmly invited to do so. As long as it is constructive, ideally with specific suggestions for improvement, and in accordance with the [contributor code of conduct](#). The best way of doing this is to [open an issue in the gitlab repository](#), or for small fixes

“Online, a book can be a gathering place, a shared space where readers record their reactions and conversations.”

- Jennifer Howard (2012)

like typos to directly suggest an edit. Please check existing open issues before opening a new one in case someone else has already spotted the same problem. In the web version of this resource you will see in the top right under the section headings of this chapter an edit () link and a view () link that will take you to the source for the current page if you would like to suggest an edit.

This site has the [hypothesis annotation viewer](#) enabled so feel free to add comments and annotations to this site there<sup>2</sup>. [Email me](#) if you'd like an invite to the HDBI hypothesis group.

---

<sup>2</sup>Quickstart: [Hypothesis - Web Annotation Tool Overview](#)  
Longform: [Hypothesis 101 - Social annotation for beginners](#)

# 1 What Constitutes Data?

It is reasonably common at this point in time in the biological sciences to **focus perhaps too much attention on the sharing of large datasets**, especially sequencing based datasets. These are important datasets to share, but **they are far from all the data that we generate. Indeed they are largely useless if not accompanied by sufficient experimental metadata** to contextualize these datasets and make it possible to interrogate them for biological differences.

“Data, data everywhere, but not a thought to think.”

- John Allen Paulos

## 1.1 A brief overview of sources/types of data

### 1.1.1 Results Outputs

- Sequencing Data
  - e.g. Genomic, RNA-seq, ChIP-seq, ATAC-seq, & their Single Cell variants
- Imaging Data
  - e.g. wide field, confocal, Electron Microscopy
- ‘Bench’ data
  - e.g. gels, blots, various forms of small tabular data of counts & scores
- Numerous others, e.g. Flow cytometry, X-Ray crystallography data

## 1.1.2 Process Outputs

- **Protocols**
  - Step-by-step procedure to reproduce the experimental work including any relevant details e.g.:
    - \* Biological resources e.g. strains, cell lines, tissue samples
    - \* Volumes, concentrations, container sizes, amounts, temperatures
    - \* Chemicals, Reagents, enzymes, & antibodies
    - \* Equipment/kit, make, model, software versions (where potentially relevant)
    - \* **Tutorial, Video, & Training content** conveying how a method is carried out in practice including details not readily captured in the written medium.
- **Code**
  - Scripts & software packages
  - pipelines, templates, & workflows
  - Computational Environments
    - \* Software dependencies and versions
  - Approximate computational resources required
    - \* minimum hardware requirements & run times

## 1.2 Metadata

Meta data is *the context needed to: find, use, and interpret data*

### ! FAIR data

Robust metadata is essential to making data [FAIR](#).

- **Findable**<sup>1</sup>
  - Have a globally unique Identifier e.g. a [DOI](#) (Digital Object Identifier)
  - Be indexed in searchable resources

“Metadata Is a Love Note to the Future”

- unattributed

- Have ‘Rich’ metadata
- **A**ccessible
  - Be retrievable using a standard open protocol
  - (Some) metadata remains accessible even when the data is not<sup>2</sup>
- **I**nteroperable
  - Use open formally specified languages or formats
  - Use vocabularies / [ontologies](#) which are themselves FAIR
  - Reference other (meta)data
- **R**e-usable
  - Be richly described with relevant attributes
  - Have clear usable [licences](#) (see: Chapter 8)
  - Have detailed provenance
  - Meet domain specific community standards

In more concrete terms the sort of information that needs to be present or referenced in a dataset’s metadata to make it most FAIR is:

- Experimental Design/Question/Motivation
- Organisms/Lines/Strains, Reagents etc.
- File format and structure information
- Technical information from Instrumentation
  - How the instrument was configured during the data acquisition.

---

<sup>2</sup>A strategy for gauging how find-able your data is: Get a student in your lab or a neighboring one with a passing familiarity with your project to try and retrieve your data from the internet **without** using your name. Can they do it at all?, how long did it take?, What would have made it easier?, How would you search for your dataset if you were trying to perform an analysis like yours on publicly available data?

<sup>2</sup>It’s useful to know that data exists even if it’s not publicly accessible so that you can ask to see it, if you don’t even know it exists you can’t ask.

- Its make, model and embedded software versions.
- Researcher Identity
  - **Human names are not globally unique identifiers** and there are plenty of researchers with the same name. If you do not have one already get at **ORCID** and have it associated with **all** of your publications.
- The protocols used in its generation
- The code used to analyse it

I used the term ‘Rich’ for succinctness under the description of what makes data findable. ‘Rich’ is unfortunately somewhat vague, this is because ‘richness’ is highly context dependent. The type of data and experiment that you have determines which metadata are essential for its effective FAIRness.

- Which keywords make it most findable? This might be biological, technical or social. You talked to some at a conference who mentioned a cool dataset and you remember their name and the name of the technology they used. Can you find the dataset with “John, Smith” and YASSA-seq (yet another stupid sequencing acronym sequencing)?
- Which biological information about the samples is most needed in a different analyses of the data?
- Which file format is most widely compatible, or has the best features for accessing subsets of large datasets over a network?
- Is the license on the data clearly indicated so that I know I’m legally allowed to re-use it? (see Chapter 8)

All of these details provide additional characteristics which enhance the findability of data by improving your ability to search for it and filter the results. If your metadata is ‘RNA-seq’ then there are millions of results. If it is RNA-seq (PolyA), Mouse <strain>, 12 samples 6 Control, 6 <Homozygous Mutant X>, aged 2 months etc. etc. I can, assuming it is well structured for search tools to parse, actually find out if I can use your data to answer my question. I may even be able to do so using a fairly



simple search tool and no manual trawling through hundreds or thousands of hits. On a self-interested note, **people can only cite your data if they can actually find it.**

**The Goal, In short, is full provenance of the results & conclusions leading step-wise from the exact materials and practical steps taken in the experiment, through the complete logic of the analysis steps to the conclusions drawn. In modern science this is long chain but it must be a complete one or it's not good science because it's not at least theoretically possible for someone to pick up the description of your work and check your conclusions independently for themselves.** I've read papers in the modern literature with less useful methods sections that [this alchemical description of the preparation of phosphorus written by William Y-Worth in 1692<sup>3</sup>](#) don't be one of these people!

### 1.2.1 Ontologies & Controlled Vocabularies

In philosophy the field of ontology concerns itself with the groups and categories of entities that exist, this often involves much disputation over which of these are basal and which merely comprised of collections of other things.

The practical application of these concepts in knowledge management and computer science is to have **shared common definitions of the things that we work on so that we don't run into the problems of talking about the same things with with different names or using the same name to refer to different things.** As well as having **standard categories of things so we can refer to groups of things with same names** as well as individual entities. These groups can also be **useful for analyzing, visualizing and summarizing large collections of entities.**

There are ontologies for Genes, proteins, organisms, reagents, methods, technologies, file types, clinical phenotypes and much

---

<sup>3</sup>Note that this published method for the preparation of phosphorus was a full 23 years after it was first produced by Hennig Brandt in 1669 - history doesn't repeat itself but it sure does rhyme.

more which makes it possible to refer to common entities and importantly **makes it possible to perform effective searches of data labeled with ontology terms.**

Almost everyone has encountered the problem of searching for all the instances of a word in a document and run into the issue of inexact matches. Consider the problem of finding all the instances of the word “color” in a document what about variants on this word like “colors”, “Color”, and the British English spelling “colour”<sup>4</sup>. What about conjugations like “colouring” or “coloured” There’s capitalization, plural, conjugation, international spellings etc. We rapidly run in to trying to use ‘fuzzy’ matching to find all the instances we may miss some (false negatives) or match some things that we didn’t intend to e.g. “discoloration” (false positives). These different versions may mean slightly different things and the problem quickly gets messy. Ontologies provide standardized [human \(NCBI:txid9606\)](#) and machine readable names to refer to the same things to avoid the many variants on this sort of matching problem. For each instance of and variant on the word “color” I’ve used here I linked to it’s latest definition in [Wiktionary](#) so that is clear that they all refer to the same concept.

Ontologies enable the searching of literature and databases, identifying the links and relationships between entities in literature, databases, data repositories and other places. This facilitates research as it becomes much easier to identify all the different places in different mediums which refer to the same entities letting you find sources and resources relevant to your work and letting computational analysts mine the many sources of data on a subject with much greater ease.

EBI maintains a database and search tool for biological ontologies at the [Ontology Lookup Service \(OLS\)](#). Many biological ontologies are maintained and developed at the [Open Biological and Biomedical Ontology Foundry \(OBO Foundry\)](#) Ontologies are living documents and must be updated and extended as new entities are described and definitions are revised etc. If an entity that you need to talk about in your work does not

---

<sup>4</sup>This text defaults to the American English spellings mostly because I was too lazy to change the default spell checker settings.

yet exist in an ontology you should consider contributing it to a suitable one via OBO Foundry.

A source for exploring more general web vocabularies is [linked open vocabularies](#) where you can find vocabularies for describing many different things. These vocabularies are generally represented in [Web Ontology Language \(OWL\)](#).

Another fun and relatively simple example of an ontology is [CiTO](#) ([Shotton 2010](#), [cito:discusses] [cito:citesAsAuthority]) an ontology for describing the nature of citations. It can represent things like if the author agrees or disagrees with the conclusions of the paper that they are citing, or are using methods or data from the cited source. This was recently adopted in a trial by the *Journal of Cheminformatics* ([Willighagen 2020](#), [cito:citesAsAuthority] [cito:agreesWith]). I include these examples inline here but they would typically be represented in the bibliography not in the body of the text, though having them prominently placed in a tooltip when a citation is hovered over might also be informative for the reader.

## 1.2.2 Metadata models, Schemas, and Standards

*I'm quite new to the theory and practice of more formal metadata standards & models contributions and corrections from those with more experience and expertise would be welcome.*

**Luckily for many domains someone has already done the work of devising suitable metadata models often based on the principles I'm going to briefly outline below and you can take these 'off the shelf' and use them to describe your data. Unfortunately there may be multiple competing standards and formats for metadata in your domain and whilst the standards may exist the tooling to use them is not always very user friendly especially for the non-programmer. Hopefully the following section will aid you in the selection of a good metadata standards to adopt in your own work and provide a starting point if you need to devise one from scratch.**

**Where you choose to make your data available often determines the minimum accompanying metadata and**

I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A “Semantic Web”, which makes this possible, has yet to emerge...

- Tim Berners-Lee  
(1999)

**in what standard or format it is made available. We will detail metadata requirements and guidelines associated with different data publishing venues in Chapter 6 [Where to publish data](#).** Many offer the option to include additional metadata or supplementary files which could contain metadata in standards other than those directly supported by the data repository but which might be valuable to others re-using your data. E.g. metadata that is not in the readily included in the format supported by the repository about your experimental design that is stored in a file with your data and parsed by your analysis code.

The aspiration of a ‘semantic’ web in which all data on it are meaningfully annotated, interconnected and readily both machine and human interpretable dates back to the same era as some of my earliest memories of using the web in the late 1990s. A little more than 20 Years on and this is still yet to be fully realized, but much work has been done and many people and organisations are still pushing towards this goal.

The artefacts of scholarly publication, be they articles or datasets, are a subset of the artifacts that exist on the internet and may need to refer to and interoperate with things outside confines of the network of scholarly citations. Consequently it makes sense to consider the problem of how to annotate them with structured metadata in the wider context of the ‘semantic web’. Firstly looking at the scope of this larger problem can be an excellent way of feeling better about the magnitude of your own problem space. Secondly, we can take some lessons and concepts from this work to inform our thinking about our particular subset of this wider problem.

The [World Wide Web Consortium \(W3C\)](#) develops standards for the web including the absurdly general problem of developing standards for the ‘semantic web’ or ‘web of data’. Their standards are generally constructed from the same set of primitives. Specifically ‘semantic triples’ the atomic unit of the Resource Description Framework (RDF) data model. A triple takes a Subject-predecate-object form, similar to an Entity-Attribute-Value form. For Example: “Alice Is 27”, “Alice Knows Bob”. Importantly triples can themselves

be objects allowing triples to be composed into arbitrarily complex knowledge graphs, e.g. “Bob Knows (Alice is 27)”.

RDF is not a particularly efficient representation in terms of storage or the computational ease with which it can be queried. It can be queried using the [SPARQL](#) language but this rapidly gets slow and verbose with higher complexity. The advantage of RDF is that it is essentially fully general, thus it lends itself to the direct representation of highly unstructured data about which you can make few assumptions and bridging or translating between disparate domains.

Developing an RDF based description of your domain specific data description problem can be a good conceptual tool. Even if it is not a representation that you will use day to day. Creating and RDF representation necessitates thinking about what explicit types of entities and relationships that your domain needs in order to be described. You can likely devise a more efficient representation which can make implicit some of the things that you would have to state explicitly to fully capture your system in RDF.

Whether you start with RDF and devise a simplified representation that makes more assumptions about your system or translate a simpler representation in RDF the ability to translate between RDF and a domain specific representation is useful for interfacing with and being indexed by more domain agnostic tools. Especially if your RDF has overlapping types with schemas from organisations like [schema.org](#) or can be readily translated into the [Wikidata](#) format. [Wikipathways](#) is a good example of a biological sciences project making use of Wikidata.

Founded by a number of large internet search providers [schema.org](#) provides a general purpose RDF schema with a set of types and classes of entities and relationships between them. There are also community extensions to the base schema.org schemas such as [bioschemas.org](#).

Conforming to an existing schema, or extension of a schema is a good way to enforce compliance of a metadata description with a standard model of that class of data. As automated tools can be used to check if a representation conforms to a

schema. The Shape Constraint Language [SHACL](#) is a more recent and rigorous language for describing the constraints on the properties of an RDF representation. Such constraints are useful when you are building tooling around metadata that has to make more assumptions about its structure than merely that it is RDF for reasons such as performance and ease of use.

It is often possible or necessary to attempt to translate from a domain specific niche representation often with more details than can be represented in a more general metadata model to such a more general model. This generally mean using a overlapping subset of features present in both models and some conceptual mapping that can related sufficiently similar categories that exist in both models

When representing metadata [json-ld](#) is the new format preferred by many developers for representing linked data as it is very human readable and relatively easy to work with ‘by-hand’ compared to XML-like formats for representing linked data like [RDFa](#).

**Including a description of your data in json-ld format in the header of an html document that conforms to the standards of schema.org permits that page to be more effectively indexed and mined by major search engines, aiding the discover-ability of pages marked-up in this fashion.**

#### metadata glossary

For more see the [linked data glossary](#).

- **Ontology** - A formal description of the things which exist (classes), the relationships between them (properties), and the logical properties by which they can be combined (axioms).
- **Schema** - A definition of the structure and contents of a data model. Often a description of constraints which would make a given representation of some data a valid instance of a specific data model. i.e. a file could be validated against a schema to check if it is a valid instance of a model.

- **Serialization format** - A file format for representing a conceptual model. RDF for example is an abstract conceptual model for representing data RDFa, Turtle, and JSON-LD are textual conventions for representing the same semantic information.
- **Standard** - agreed set of norms often accompanied by formal specifications to facilitate conforming to these norms.
- **Data model(ing)** - Creating a representation of domain specific knowledge which faithfully represents that knowledge so that it can be formalized into some form of standard representation.
- **Controlled Vocabulary** - A curated set of terms to describe units of information. Often a standardized word or phrase and unique identifier linked to a description and synonyms. Used taxonomies & ontologies.

**The importance of good metadata standards is scale independent from the entire internet to the intimate details of the structure of image file formats** which we will touch on when we discuss the OME-NGFF (Next Generation File Formats). If the world of formally structured metadata is a bit much you can start small by just adding some simple key value pairs to some table or entries that accompany your published data like “Species (key): [human \(NCBI:txid9606\)](#) (value)” or

**Pragmatically technical and theoretical considerations for what might be considered the best format for your metadata are often trumped by ease of use / good tooling and wide adoption in a particular community. As a general rule It is best to use the community standard and participate in community efforts at reforming, refactoring, or extending them if they are lacking in some way.** Beware the proliferation of standards problem:



Figure 1.1: XKCD 927 “Standards”

### 1.2.2.1 The ISA (Investigation, Study, Assay) metadata model

**How can we approach systematically organizing such a complex and heterogeneous collection of metadata?**

This is an extremely challenging problem which does not have a definitive answer. There are a number of [standards](#), none universal in scope. Systematic machine readable representation of this metadata is however essential to achieving the **F** in FAIR (Findable) data cannot be properly systematically indexed & searched without structured metadata.

#### **i** ISA

One schema proposed for the representation of biological experiments is the ISA model ([Sansone et al. 2012](#) [cito:citesAsAuthority]):

- **Investigation**
  - A high level concept linking together related studies
- **Study**
  - Information on the subject under study, it’s characteristics and treatments applied



- **Assay**
  - A test performed on material take from the subject of the assay which produces a qualitative or quantitative measurement

This provides a high level framework for structuring your experimental metadata which is extensible to accommodate a variety of more specialized descriptions within it or referenced by it.

## 2 When Should I Generate Data?

*After you've got a plan for how you are going to analyse it.*

If you will have a specific narrowly and well defined hypothesis to test which you will analyse yourself having a thorough plan before you generate your data may be relatively straightforward. However **it is easy to accidentally gloss over some detail in an inexactly formulated analysis plan.** I always advise that you **produce some dummy data** in the format that your experiment will generate. **Use this to work through the steps of your analysis in its entirety.** This will make it easier to spot any ambiguities or confusions in the planned analysis. In Chapter [4 working with data](#) we discuss some approaches you can take to make this process easier.

**Test your analysis with dummy data that supports your hypothesis as well as dummy data that goes against it.** If you play around a bit with your simulated data this will give you **qualitative impression of the statistical power of your experiment.** How strongly would your experimental data need to match your expectations for you to be able to see the results clearly in the statistics? Does this match up with the effect sizes that you saw in any preliminary data? **Do you have all the controls/calibrations that you need** and are you making best use of them in your analysis? **What can you tweak in your design to make it robust if your full dataset turns out more noisy than your test data?** In short **is your design actually up to the task of answering your question?**

At this stage you should **seek feedback on your plans** from your colleagues, indeed as we shall discuss in Chapter [5 when to](#)

“Data is like garbage. You'd better know what you are going to do with it before you collect it.”

- Mark Twain  
(*attribution questionable*)

“The first principle is that you must not fool yourself and you are the easiest person to fool.”

- Richard P. Feynman

[publish data](#) this might be a good time to think about publishing a [registered report](#) detailing your hypothesis(es), planned experimental protocols & analysis(es). Getting an outside perspective on your methods and design can be very helpful and lead to major improvements in your experiment. You may also get some bad takes. Even if some or all of the feedback proves of little value **merely going through the process of putting your work in a form that others can give feedback on at this stage is remarkably effective at helping you to catch errors in your own formulation of your ideas.** The advantage offered by getting someone external to look at your planed project and not just your supervisor or someone else close to it is that you get a ‘red team’ a concept popular in [defence](#), [cybersecurity](#), and other areas where the stakes are high for your models mapping well onto reality. Whilst a culture/system of red-teaming can be cultivated internally people often feel a bit awkward about being brutally honest about a colleagues work, an affliction often mitigated by simple expedience of not being likely to bump into the other person at lunch the next day. [The red team concept has been tried in academic contexts](#) but has yet to see wide adoption.

Importantly this **planning should take place before you have committed experimental resources to testing your ideas.** It is a responsibility of an ethical researcher to **ensure that when significant resources are going to be expended on an experiment that its design be sound and as close to optimal as is practically achievable.** This is especially true in the context of scarce and valuable experimental resources like donated human tissues.

**Why seek feedback at this time?** at the **grant stage plans** are often **insufficiently concrete to benefit from benefit from** this kind of **object level critique** and at the **publication stage** it is **too late to change anything really important** and you’ve already committed resources. **Grant scrutiny is for funders, publication scrutiny is for journals** so that they can protect their respective reputations if they attach their names to your work. **Red-teaming and registered reports** ([Section 5.2](#)) are for the researcher

they let you **make best use of your own resources and demonstrate the integrity of your process.**

## 2.1 When to speak to data analysts

If you are going to be **collaborating** with a bioinformatician/statistician or other analyst **on the analysis of your data you should speak to them before you pick up a pipette to generate your main dataset**<sup>1</sup>.

**Bioinformticians and statisticians frequently receive data to analyse with problems in the experimental design that cannot be properly addressed at the analysis phase.** An example common in my experience is that of technical or batch effects perfectly confounded with biological variables of interest to the experimenter. This is often readily resolved by using variants on a split-plot design / appropriate blocking. This can sometimes lead to a more logistically challenging experiment at the bench but often means that you need fewer repeats to robustly observe smaller effects. More importantly fixing confounding issues can allow you to properly separate technical from biological sources of variation something which cannot be fully addressed by attempting to correct for it statistically. **It is important to spend the time thinking about these trade-offs during the design process.**

A common symptom of this sort of design failure is over-reliance on batch and other effect corrections, it is now commonplace to see batch correction used in various ‘omics’ analyses essentially **by design**, this is almost always a bad idea. You should not assume that you can successfully correct for something in your analysis, you should especially not assume that you can

“Further, science is a collaborative effort.”

- John Bardeen

---

<sup>1</sup>You may find that your bioinformaticians and other analysts are surprised and even confused to be consulted at this early a stage in the process. This is likely because they are so used to be presented with the data as a fait accompli that they are not sure how to react when consulted at the proper time. You may need to be patient and encouraging with them in order to convince them that you are actually interested in their thoughts on your experimental design.

correct for multiple things in your analysis, and you should really not assume that you can correct for multiple things when you have a small number of samples and are running a large number of tests. You are very unlikely to be able to perform a robust statistical analysis of your data if you have made these assumptions.

These correction methods are tools that should ideally only show up in observational studies or meta-studies combining results from multiple experiments. It should not be a part of the design of a planned experiment unless there is no other recourse. These and similar approaches should not be used *in lieu* of proper experimental design.

## 2.2 Exploratory analyses are not exempt from proper planning

It is one thing to generate a dataset without knowing what it is going to show you, this is to be expected, it is quite another to generate a dataset without even knowing if it will be able to answer any of the questions that you are interested in. A valid reason to generate a dataset is also calibration: what can we see, and how reliably with a given method, in a given system? Calibrating your method, performing exploratory analysis and testing the resulting hypotheses are too often conflated into a single step. You don't want to end up 'betting the farm' on one set of experimental parameters that you hope will be able to answer your question(s) without a high degree of confidence that you will see the effect(s) you are interested in. Large 'calibration' experiments can be a public service to a research community as subsequent users of these methods are saved the bulk of the parameter tuning work and can use simple controls to check their implementation lines up with the reference. Such datasets can be expensive and a lot of work but often garner a lot of citations and good will.

People attempt, often due to resource limitations, to answer a biological question with with poorly calibrated methods where

"if it's worth doing, it's worth doing well"

- proverb

the baseline variability is insufficiently characterized and perhaps most common mistake: insufficient statistical power to properly test the question the experimenter wants to answer.

Exploratory and descriptive analyses are perfectly reasonable undertakings as hypothesis generating exercises. It can be helpful to have at least one concrete well defined hypothesis to test with the data you plan to generate. Because of the nature of null hypothesis significance testing an important thing for correctly interpretable statistical results is to draw clean separation between planned analyses testing specific hypotheses and exploratory analyses, we will return to this subject in [Chapter 5 When to publish data](#). It is important to have exacting clarity on what your study is setting out to achieve and what you can reasonably expect to measure with your experimental setup.

If you generate for example an RNA-seq dataset vague goals like: “I’m going to do differential expression analysis & functional enrichment analysis” are **not** a plan.

- What sort of effects do you want to be able to see?
- Will the design you have planned have the power to see the sorts of effect you are interested in?
- What FDR (False Discovery Rate) is acceptable for what you want to do with your results?

You might treat a large effect size and low p-value on an exploratory analysis that you did not plan as a sort of informal bayesian evidence and modify your qualitative priors as a result. This can inform subsequent experiments but the absolute value of a p-value from such an after the fact test is not meaningful, treating it as such is HARKing (Hypothesizing after the results are known). This is not in and of itself a problem as an approach to thinking up new hypotheses but mis-representing a hypothesis arrived at after the fact as the one originally being tested is.

You can design for this with simulated data and running analysis code before a single cell sees culture medium. Modern bioinformatic analysis & image processing pipelines are long complicated and difficult to comprehend in their entirety. It is a very challenging if not impossible task to intuit what sort of

changes in your input data may yield outputs with meaningful effect sizes and degrees of confidence.

One of the best ways to handle this is to design your analysis and feed it dummy / test data, be it simulated or from previous/pre-liminary experiments and see what you can actually reliably detect in the output. This needn't be the entire pipeline in the case of sequencing results for example it can be a lot easier and more practical to start at the count matrix phase if you are looking for differences in expression/occupancy.

This process is also very useful for being able to properly interpret your results. Play with the knobs on the black box of your complicated analysis and see if you can predict what happens to the dials at the other end, set the dials and see where the knobs had to be to produce that result. Does it pass the high level common sense checks? Does it break, or perhaps more worryingly not break, when you set it to weird edge-case values which often include 0, negative numbers, infinity, or missing values?

This is important for the verifiability of your work ([Hinsen 2018](#) [cito:citesAsAuthority] [cito:agreesWith]). If your work is reproducible but wrong it's nice that it'll be a bit easier for someone to figure out why it's wrong but it will still be wrong. Putting the time into understanding and stressing your analysis process makes it easier to spot conceptual, logical and technical errors in the final analysis. This is hard to do unless you have calibrated your intuitive expectations for how the system will behave when presented with a given input. You can develop the diagnostic tools, such as visualizations and summary statistics, you need to spot sources of error.

When you have a pipeline that you are going to re-use or that is of critical importance it can be useful to construct more formal tests of your analysis pipeline borrowing concepts and sometimes tooling from [integration](#) and [systems](#) testing in software development and applying some of these practices to your analysis pipeline(s). Good test coverage can be a valuable resource to help you catch mistakes when updating, extending or modifying a pipeline. It's easy to make inadvertent changes which have unexpected effects especially when re-visiting old projects, tests and checklists can help you avoid this. This needn't be

only in data analysis similar principles apply in the wet lab, refreshing cell-lines and reagents, re-validating mutants etc.



### 3 How To Store Your Data

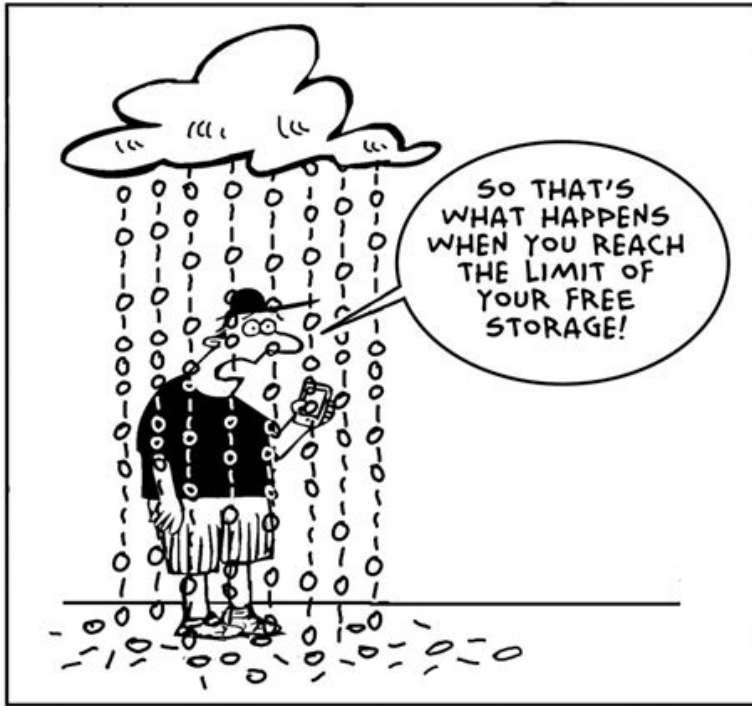


Figure 3.1: So that's what happens when you reach the limit for free cloud storage.

**Whenever you are embarking on a research project an important part of the planning process** especially in the era of massive multiomics datasets & high resolution n dimensional microscopy **is estimating how much data storage you are going to need.** Storage can intuitively feel cheap and easy to acquire more of. This is largely because of how much its price has dropped over time in recent decades and the advent of cloud storage solutions. Thus it can be tempting to

put off thinking about this problem until confronted with it. Ideally **you should think about this problem in reasonable depth at the grant application stage or before** and ensure that you have both sufficient funds and a solid plan in place to ensure the **integrity, availability and security of your labs research data for the lifetime of a given project and beyond** as needed.

### *Storage Space isn't everything*

Beyond how much space you will need there are other factors that you will need to consider. This includes but is not limited to a strategy for **backing up your data & recovering** your working state from those backups and performance requirements. Here we will cover various factors that you should consider in your data management planning.

## 3.1 Estimating capacity

A technique I recommend for estimating how much data you are going to need to store is **Fermi Estimation**. Those who've heard of it know this approach is a lot less intimidating than it sounds, it's just a loosely structured approach to do back of the envelope calculations to estimate a quantity, here is [A short list of nice examples](#). A tool useful in any number of situations where you need to make an estimate of a an uncertain quantity which might have a number of inputs is **Guesstimate**. I'd recommend using this to perform your Fermi estimates of how much data storage you are going to need. **Doing some basic research to set your priors on file sizes & numbers of files, based on previous datasets from similar methods is often the best way to get started with your Fermi estimate**. Here is [an example set of estimations in guesstimate for the UCL imaging hub](#).

Whatever you end up with for your **highest probability estimate you should probably add 50-100% more on top of that number, and/or lean towards the >95th percentile of the distribution** of your estimated value. This may seem like a lot but but I don't think I've ever met someone who overestimated how much space they would need and

regretted it. On the other hand I've met many people who undershot and regretted it despite thinking they'd built in enough headroom. **We often fail to account for everything, and it's helpful to have some excess capacity around for performing operations on your data**, moving data around etc. Some kind of *performance degradation of storage devices is almost inevitable as they approach full capacity*, a rule of thumb I use is to try and **avoid exceeding >~70% usable capacity** of my systems **before expanding** them.

**In order to backup you data properly, in line with the 3-2-1 backup rule you will need access to usable storage capacity 3x that of the total size of your dataset. You should budget for this.**

If you cannot immediately afford the data storage that you expect to need by the end of your project it is almost always wise to invest in storage technology which is capable of growing to accommodate your future needs even if you get slightly lower initial capacity. The alternative is convoluted cobbled together combinations of stop-gap storage solutions and/or greater expense in time & money when replacing and migrating from your initial improvised solution. Not to mention that trying to get sensible backups of this hot mess will make your storage admins cry tears of despair, and you don't want your storage admins to despair you want them to be quietly confident that whatever you did the backups are so rock solid that they can fix it for you.

## **3.2 Categorizing / Stratifying your data**

**It is often necessary to prioritize or stratify your data so that appropriate policies can be applied to different datasets.** Your raw data for example probably needs the most robust backups in terms of number of different copies and integrity of that data but it may not need to be very regularly updated excepting when new batches of data come in. Working data on the other hand, intermediate often ephemeral steps in analyses which change very regularly might want more frequent

backups but may not need as many different backups or to keep them for as long.

As a researcher your priorities are typically to secure, the **raw data, how you got from the data to your results and your results themselves**. The **first two are key and the latter a convenience** so you don't have to re-run your analyses to retrieve them. **You have succeeded at reproducible analysis if I can delete all your results and you don't really care, because getting them back is at worst a minor inconvenience.**

**Top priority generally goes to your raw data and it's associated metadata.** The **metadata is critical** as a directory full of raw sequencing data files is useless unless you know to which sample/experimental condition each file belongs. I would advise **keeping a structured copy of your experimental metadata at the same location as your raw data** even if you have another system to store your metadata for example in a more centralized database. co-locating the two reduces the risk of them becoming decoupled and vital metadata being lost. **Defend in depth against data loss.**

Next highest priority goes to the **code / compute environment which permits you to generate your results from your data & those final results**. If your code is under [source control](#) e.g. git (Section 4.3.1) this is quite easy to achieve. You can push your code to a remote git server on a platform like github and/or gitlab indeed setting up automated mirroring of git repositories between these platforms is quite simple (Section 3.2.3).

lower priority goes to **intermediate compute products which can be re-generated as long as the code and data survive.**

#### **i** RPO/RTO

Some non-technical definitions of two important concepts in planning how to recover from a problem with your data storage:

- **Recovery Point Objective (RPO)**

- From what point in time do you want to be able to recover?
- If there is an issue can you lose the last Minute/Hour/Day of data without too much of a problem?

- **Recovery Time Objective (RTO)**

- How long can you go between a problem occurring and you having restored your data storage to working order?
- How long can you spend trying to fix the problem before you have to cut your losses and get back to work?

Think about how these concepts apply to the different categories/strata of data that we considered above.

If your storage technology enables it it is often useful to keep snapshots of your data which you then prune as you get further back in time e.g. hourly snapshots kept for a day, dailies for a week and weeklies for a month etc.

If recovery time was your priority i.e. you need to get everything back up and running as fast as possible and you can afford to lose a little data then you might roll back to the last good snapshot and not spend time cherry-picking files from more recent snapshots to get yourself the most recent good versions of all your files. If recovering from the most recent point in time possible is your priority and you can afford a little downtime while you pick out all the most recent versions of your files then you might do this instead. **The important thing is to think through what your priorities are ahead of time and plan your backup and recovery strategies accordingly.**

**!** 3-2-1 rule

The **3-2-1 rule** of backup is generally formulated along the lines of:

- **3 Copies of your data**
- **On 2 different devices / in 2 different formats**

- *plus 1 off-site copy*

It *can* also be a good idea to use two different mechanisms for your off-site copy and your 2nd local copy. That way if something breaks with one of your approaches and there is an issue with one of your backups the other will not be affected by the same problem.

### 3.2.1 Deleting Data

You may be legally obligated to keep certain types of data for a specific period of time, or required to do so by your institution, or by a journal in which you have published your research. When thinking about a data deletion policy you should always consult your institutional policies and the relevant standards and practices for doing so. For example it is commonplace that data underlying a research publication is required to be preserved for 10 years.

“data underlying publications should be retained for 10 years from the date of any publication which fundamentally relies on the data”

- [UK Concordat on open research](#)

Trying to keep everything forever gets expensive and impractical quite fast. Consequently you will need policies in place to delete data. However, wherever possible it is best to design workflows which stratify data at the time it is generated. Place data that can safely be disposed of in locations that facilitate its disposal, such as explicitly temporary directories. What can you automatically flag, and what will require manual review, to be considered for deletion? How can you structure your workflow so as much as possible lands in the automatically detectable bucket?

Be wary of automated deletions, they make sense for some things like pruning snapshots from backups but any process

that can automatically delete your data is inherently dangerous and should be treated with caution. Automatically flagging data to be deleted and waiting for your review and approval before doing so is probably best, if there is any probabilistic component to the flagging, and you can keep on top of the review process.

What data should you consider deleting?

- Redundant copies of things (that are not part of your backup strategy)<sup>1</sup>
- Stuff that would be useless without provenance that doesn't have provenance.
  - If you generated a random temporary file whilst developing your analysis and you've got no idea how you made it or where it came from you can probably delete because if you can't explain where it came from you won't be able to use it anywhere anyway.
- Stuff that you can get back
  - Intermediate data files which you can deterministically re-generate from your code, raw data and compute environment.
  - **Versioned** files that you downloaded from the internet that you should be able to get back from the repository from which you originally downloaded it. This only applies to resources where they have versioned downloads that will let you get back the exact same data file from the metadata i.e. version information that you kept.

### 3.2.1.1 Have Clear Policies on Data Deletion

When deleting data, especially other people's data that has been entrusted to you it is essential to have clear and well com-

---

<sup>1</sup>A favorite utility of mine for identifying duplicate files is [rdfind](#). It can preserve an existing directory structure with multiple copies of a file but only keep one copy on disk with hardlinks. This is convenient for reducing the size of messily structured data where people have made 'backups' by copying everything to another directory periodically rather than doing a differential backup.

municated policies.

- Tell people up front what guarantees you can and cannot make.
  - How long can you keep it
  - How much can you keep
  - How fast can it be accessed (this can get a bit more complicated, but may be important for certain use cases)
  - What is your back-up policy, so they know what additional measures they may need to take
  - Make this information easy to find, going forward and reiterate it fairly often in communications
- Very **vigorously** communicate any changes in data retention policy so that people know what to expect and when.
  - Give lots of notice, follow up with reminders of increasing intensity as the deadline approaches
  - It might be a good idea to provide a means of affirmatively acknowledging that the user is aware that the data will be deleted and is OK with this
  - If possible use multiple channels of communication
  - Provide easy ways for people to get a copy of their data, get help getting a copy of their data, or extend the period for which you are able to store it if this is a possibility.

Even If you are going to delete data in the course of normal operations and without a change in policy for some things you may wish to notify the user of any impending deletions.

#### Security considerations for device disposal

If you are disposing of devices with data stored on them then caution is warranted.



If the data is not particularly sensitive and your entire drive is already encrypted at rest then you may not need to do anything as long as the encryption key is not located on the device anywhere and you are unconcerned about ‘store-now decrypt’ later attacks if you are not using particularly strong encryption or quantum safe cryptographic algorithms. If you want to exercise additional caution re-format the disk. If you want to exercise more caution and are OK with putting a bunch of additional writes on the device, write over the whole drive once or twice with random data after re-formatting before sending it off to a re-furbishing service.

When overwriting a device to attempt to ensure complete data deletion don’t use all 0s you want something basically incompressible so SSD controllers won’t lie to you about having written an entire disk’s worth of 0s and secretly only wrote 1 block claiming to be a whole disk’s worth of 0s.

If the data is critically sensitive, re-format the device and write over the entire device multiple times with randomly generated data, then physically destroy the device on which it was stored yourself with a power tool (wear eye protection) and spread the resulting debris out over a couple of waste collection cycles. If it is a hard drive ensure all the platters are shattered if it’s an SSD ensure all the NAND flash packages (chips) are broken. Physically destroy all apparently non-functional devices containing sensitive data as they may have a trivial defect that might permit a 3rd party to recover data from them if they fix the device. I’m not kidding. This may seem wasteful but if your data is actually critically sensitive this is the only way to be sure.

## **3.2.2 Practice Recovery**

### **3.2.2.1 Personal device loss**

*Game out your recovery process*

How would you recover if one of more of your personal computing devices disappeared?

What concrete steps would you take to get back to work if personal computer was no longer available to you. How would this go if you lost your phone at the same time?

Grab an old laptop and see how long it would take you to get back up to speed, if you even can completely? What would you lose? How important is it? how can you improve this?

If you haven't done this yet have a friend to 'steel' your laptop and refuse to give it back to you until you figure out how you would deal with this situation. If you have a friend who you know hasn't got a plan for this scenario 'threaten' to 'steel' their laptop until they've got one. (This should no be construed as advice to actually steel or threaten to steel your friends things but rather to apply social pressure to them to have good backup & recovery strategies and to leverage social pressure to force yourself to do the same.)

### 3.2.3 Storing Code

In Section [4.3.1 Source Management](#) we will cover `git` which is how I recommend that you manage your code and indeed other text. Here I will just detail some steps you can take to automatically keep a backup of your code on `git` hosting services.

#### Repository mirroring

A simple way to have another copy of your `git` repository is to set up mirroring of your repository to another `git` hosting provider. Let's say that you have your code in a gitlab instance like [gitlab.com](https://gitlab.com) or [renkulab.io/gitlab](https://renkulab.io/gitlab) but you also want a copy in [github](https://github.com). Gitlab supports repository mirroring your repository to other gitlab instances as well as to github and other providers, there are [detailed instructions for this in their documentation](#).

You will generally want to configure a 'push' mirror from gitlab to github if gitlab is your primary repository. If

github is your primary repository and you would like to 'pull' from github to gitlab this is a paid feature in gitlab. There are 3 main steps.

**1. create a repository to mirror your project to.**

Create an empty repo in github, no README no license just empty.

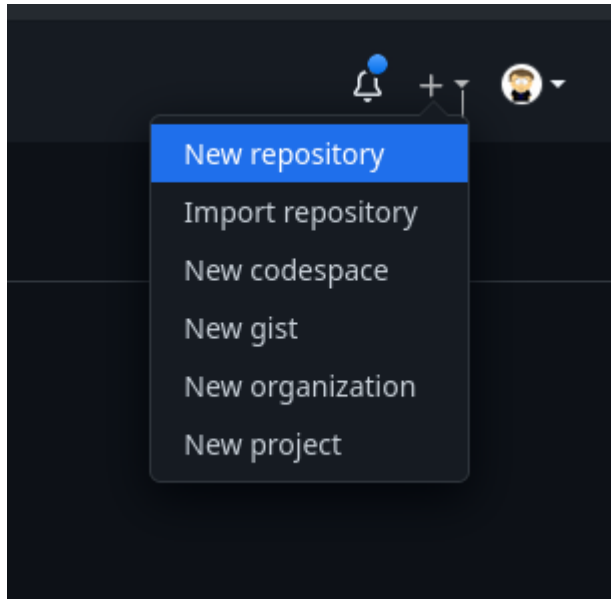


Figure 3.2: new repo on github

**2. Generate an access token to authenticate and authorize the connection between the repositories.**

In github go to:

```
Settings >developer settings >personal access  
tokens >tokens (classic) >generate new token  
>generate new token (classic)
```

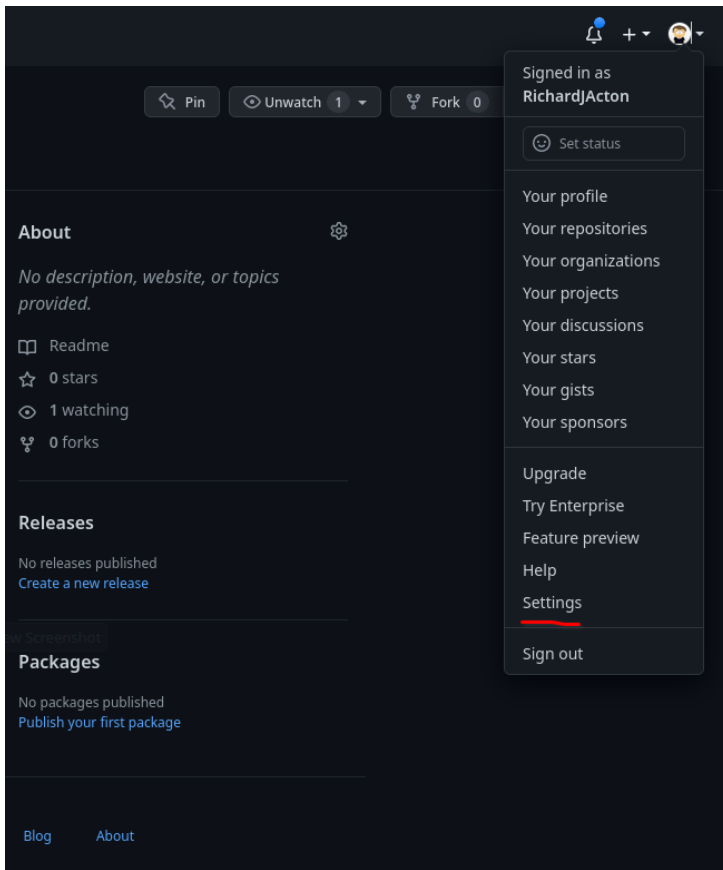


Figure 3.3: GitHub Settings

The developer settings are at the bottom of the settings menu on the left.

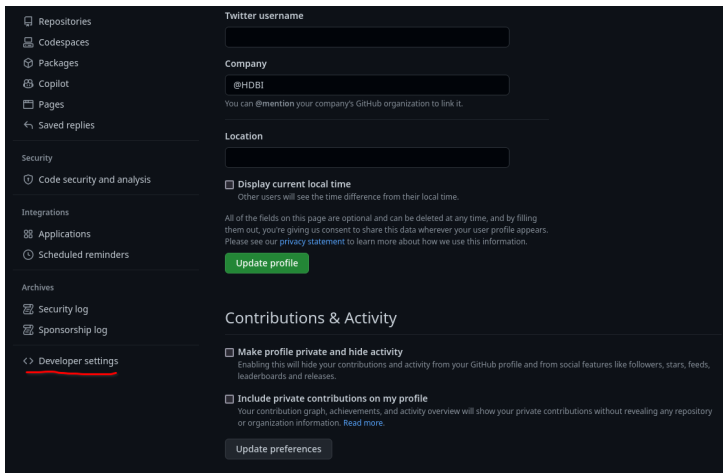


Figure 3.4: GitHub Developer Settings

You'll want a 'classic' token at time of writing the fine-grained scope tokens do not support repo syncing. (You may want to revisit this in the future in case you can get a token scoped to a single repo in accordance with the principle of least privilege.)

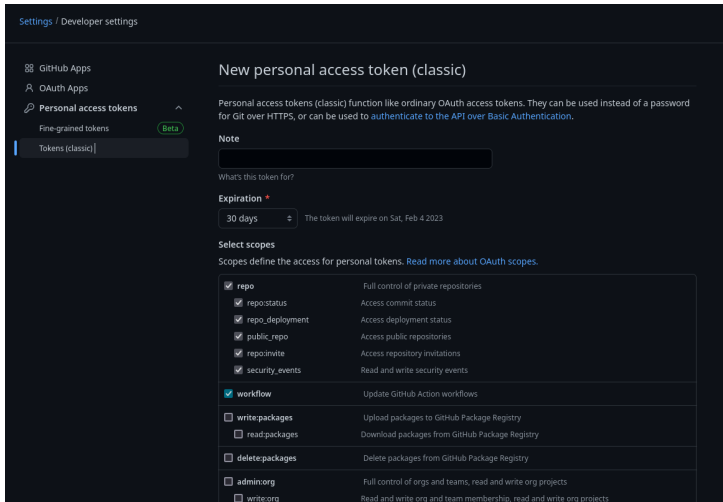


Figure 3.5: GitHub personal authentication token (PAT) Settings

***PATs SHOULD BE TREATED LIKE PASSWORDS*** keep them somewhere safe like in your bitwarden vault. ***The ‘scopes’ needed are the ‘repo’ permissions set and ‘workflow’.***

You can set an expiry date for your tokens you can balance the risk of leaking a token without an expiry date against forgetting to renew the token and having sync stop working without you noticing.

### 3. Configuring your original repository with the mirror repository URL and the access token.

Settings > Repository > Mirroring Repositories

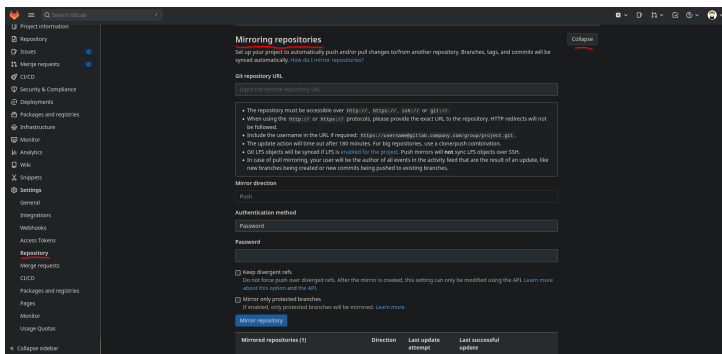


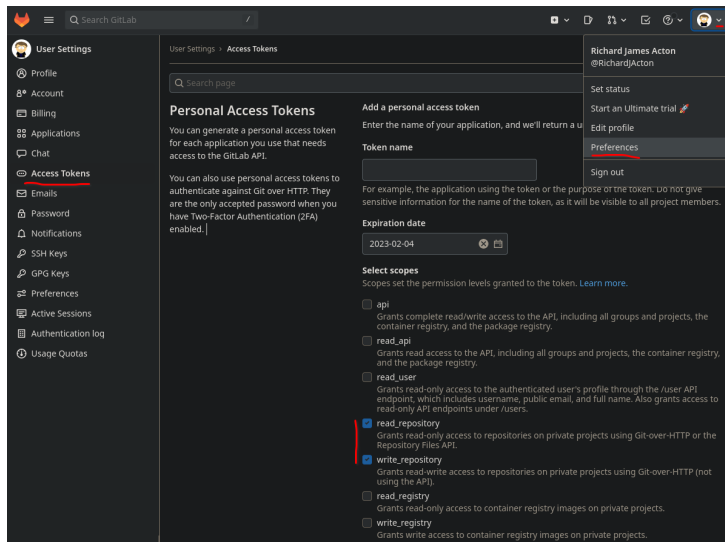
Figure 3.6: GitLab repository mirroring settings

The PAT from GitHub goes in the password field, and is also included git repository URL in the following form:  
`https://ghp_XXXXXXXXXX@github.com/username/reponame.git`

To sync to another GitLab repo instead of GitHub you will need a PAT from GitLab. To generate a PAT in GitLab got to:

User Icon Dropdown Menu > Preferences > Access Tokens

The scopes needed in gitlab are `read_repository` & `write_repository`.



This can then be used in the same way as a the GitHub PAT for the purposes of mirroring the repo. The URL format is a little different:  
`https://username@gitlab.com/username/reponame.git`  
The PAT does not need to be in the URL only in the password field.

## Pushing to multiple remotes

You can also push your repository to multiple remotes rather than connecting your remotes together and pushing to only one. This solution is probably more error prone as it is easier for your remotes to potentially get out of sync so I would advise mirroring over this.

```
git remote set-url --add --push origin git://original/repo.git
git remote set-url --add --push origin git://another/repo.git
```

(See this [StackOverflow answer](#) for additional explanation and discussion of this solution)

### 3.2.3.1 Making your git repo citable

To generate a DOI so that you can cite your code you can use [Zenodo](#) to store a snapshot of your repository. You can associate a commit in the history of your git repository with a named tag, a mechanism commonly used for releasing named versions of software or other content (see: [?@sec-semantic-versioning](#))

When planning to deposit something in Zenodo it is a good idea to test everything out and make sure that it is all working first at [sandbox.zenodo.org](https://sandbox.zenodo.org) where temporary records can be created for testing purposes.

To get started you need a `.zenodo.json` file which contains all the information needed to produce your citation. The `.zenodo.json` file must conform to the [JSON schema defining a valid .zenodo.json](#) which is described in more detail in the [Zenodo developer documentation](#) and can be checked using a [JSON schema validator](#). Alternatively the [json editor](#) tool lets you use a JSON schema to generate a web form which you can fill out, [this link](#) goes directly to json editor with the Zenodo schema pre-loaded.

It is also possible to automatically generate a `.zenodo.json` file from a `CITATION.cff` file using the [cffconvert](#) tool. [CITATION.cff files](#) are “plain text files with human- and machine-readable citation information for software (and datasets)” which work with several tools:

- [github](#) where they are parsed and information in them displayed on your repo page, This feature is not yet available for gitlab repos but there is [an open issue](#) about getting it added. `CITATION.cff` files also work directly with github’s zenodo integration skipping the need for a `.zenodo.json` file though it appears that some properties cannot be directly translated between these two formats, so the best zenodo results will come from a `.zenodo.json` file. [cff-initializer](#) is a useful tool for generating `CITATION.cff` files with a web form.



- [Zotero](#) can generate `CITATION.cff` files and the browser extension recognises them for importing citation information from repositories.

Note that when adding a grant to your project, Zenodo integrates with reporting lines for some grant awarding agencies. Consequently I would suggest filling this out in the Zenodo web UI as there is auto-completion for funding bodies and grant codes. There is often ambiguity in name and grant code format so if you want to get them correct for easy automated discovery I'd start here.

### 3.3 gitlab

1. The [gitlab2zenodo](#) tools makes it easier than using the [zenodo API](#) directly to deposit a tagged release of your project on [Zenodo](#) and generate a DOI for it. With it and gitlab CI/CD you can also automate updates to Zenodo whenever you release a new version. You can install [gitlab2zenodo](#) from [pypi](#) with: `pip install gitlab2zenodo`
2. To use [gitlab2zenodo](#) you will need a Zenodo API key. Use the drop-down menu next to your username in the top right of the Zenodo site and Navigate to: **Account > Applications > New token**. You will also need to do this separately for [sandbox.zenodo.org](#). To use the full functionality of [gitlab2zenodo](#) give it the scopes: `deposit:actions`, `deposit:write`, & `user:email`. Copy these tokens and store them securely in your password manager, be sure to name them so you know which one is for the sandbox (See: [Section 3.6](#)). [gitlab2zenodo](#) expects the environment variables `zenodo_token` to be set in order for you to use it. If you are using it interactively to test your `.zenodo.json` file you can do this with `export zenodo_token="YourKeyHere"`. If you are using it with gitlab CI/CD automation, then; from you repo page go to **Settings > CI/CD > Variables > Add Variable** use `zenodo_token` as the key and your token string as the value. This variable will now be

available in the environment of your CI/CD runners. Use the sandbox token while practicing and change it for the real one when your are ready

3. Craft a `.zenodo.json` file that you are happy with by using `gitlab2zenodo` to send it to [sandbox.zenodo.org](https://sandbox.zenodo.org). You can do with with the `-s` option: `g2z-send -s -m .zenodo.json`. Once you have run this you should see this submission staged in the `uploads` page on [sandbox.zenodo.org](https://sandbox.zenodo.org) where you can preview it.
4. Setup release automation with gitlab CI/CD. Setup you API token as described in step 2. This is an example `.gitlab-ci.yml` file:

```
# You will likely have other stages here like building and/or deploying your project
stages:
  - doi

deploy_zenodo:
  stage: doi
  image: python:3.6 ①
  rules:
    - if: $CI_COMMIT_TAG =~ /^v?[0-9]+\.[0-9]+/ ②
  script:
    - pip install gitlab2zenodo ③
    - git archive --format zip --output ${CI_COMMIT_TAG#v}.zip ${CI_COMMIT_TAG} ④
    # - g2z-send -s -m .zenodo.json ${CI_COMMIT_TAG#v}.zip ⑤
    - g2z-send -p -m .zenodo.json ${CI_COMMIT_TAG#v}.zip ⑥
```

- ① We are using the Python:3.6 image based on debian.
- ② This means that this will only run when the commit tag begins with a 'v' and is follow by some digits, a '.' and some more digits
- ③ Install `gitlab2zenodo` in the CI/CD runner
- ④ Make a `.zip` archive of the repo state at the current commit (tag), `${CI_COMMIT_TAG#v}` is a default environment variable that contains the name of the tag for example: 'v0.0.1'.
- ⑤ The `-s` flag indicates that `gitlab2zenodo` should use the sandbox, test things with this version.

- ⑥ `-p` means publish the result immediately - If you don't set this option you must manually review and publish the draft on the zenodo website. A subsequent run of the pipeline will fail if there is an unpublished version staged but not published.
- 5. Set `zenodo_record`. Once your first version has been published in order to update it and not simply create a new entry every time this pipeline runs you need to set the environment variable: `zenodo_record` (this can be done as described in step 2). The value of `zenodo_record` is the zenodo ID of your first entry, for this book it is: [8021382](#), it's at the end of the zenodo url. I would suggest testing your CI/CD with `-s` the sandbox. First check the first draft is correct at the sandbox site and publish it manually there, then set the `zenodo_record` value and use both the `-s` and `-p` flags on a new commit with an updated version tag to check that the next version is auto-published. Finally delete `zenodo_record` in preparation for publishing the real version.
- 6. Publication: Repeat the process in step 5 without the `-s` flag set. That's it your project is published and you've got your DOI!
- 7. Get the DOI badge from your zenodo record and include it in your project README etc. You will also find a 'Cite all versions' DOI on the zenodo page which will always point to the latest revision of your work as well as the version specific DOIs, which display a banner saying a newer version is available when visited if there is a newer option. It may be easiest to include the cite all versions DOI in your README so that it will always remains up to date, and encourage people to use the version specific citation when they cite your work.

## 3.4 github

Zenodo integration with github is a little better at the moment. Github has a concept called 'releases' which correspond to a git tag which marks a particular point in your repository's history

as a released version of the software. Checkout the [Github documentation on using zenodo integration](#)

### git tagging

- list tags: `git tag` or to filter tags: `git tag -l "v*"`
- Create an annotated tag: `git tag -a v1.0.0 -m "Description..."`
- Create a lightweight tag: `git tag v1.0.0-tmp` (lacks author information, can't be signed)
- Get details of a tag: `git tag show v1.0.0`
- Retrospectively tag a commit `git tag -a v0.0.1 <commit hash>`
- Sharing tags `git push <remote> <tagname>`, multiple `git push <remote> --tags`
- Deleting *local* tags `git tag -d <tagname>`
- Deleting remote tags `git push origin --delete <tagname>` (use with caution if other might use these tags)
- Checkout a specific tag in a branch `git checkout -b <branch> <version>`, you can just do `git checkout <tag>` but this will leave you in a detached HEAD state. (Checkout [git worktrees](#) if you'd like to have multiple branches checked out at the same time).

### semantic versioning

A popular convention in the naming of software releases is the concept of [semantic versioning](#) which is summarized as:

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API<sup>2</sup> (Application programming Interface) changes
2. MINOR version when you add functionality in a backwards compatible manner
3. PATCH version when you make backwards compat-

ible bug fixes

For example a piece of software might have the version 1.8.12 (said: ‘one dot eight dot twelve’, not: ‘one dot eight dot one two’). This is the 12th set of bugfix patches, for the 8th set of backwards compatible changes e.g. adding some new features that don’t affect the old ones. This is also probably technically the 2nd version of the software as the 0.X versions of the project were probably development and testing versions with 1.0 being the first stable release. A future 2.X release might change the way some of the things in the 1.0 series worked known as making ‘breaking changes’. Generally in a larger software project it is a good idea to signal these changes to your users with code ‘lifecycle’ information. Checkout the [{lifecycle}](#) R package docs for some good general advice on this that applies beyond R.

**BEWARE** Not all programs that use a X.Y.Z naming convention for their releases follow semantic versioning conventions and even for those that do what constitutes a ‘breaking change’ can be a tricky and context dependent question.

A critical point in the history of an academic piece of software might be the version of an analysis used in a published piece of research. This is probably worth marking in the release notes if you are both making the software and publishing about the new analysis/workflow that it enables.

---

<sup>2</sup>An API (application Programming Interface) is a set of rules the two pieces of software use to communicate with one-another. You can think of it as a sort of layer between two programs. In the context of a software package it often means changes to how functions from that package interpret arguments given to them by the programmer. Lets say I have a function `subtract()` that takes 2 arguments, `x` and `y` and subtracts `y` from `x`, in version 1.0 of my software. In version 2.0, however, I change it to subtract `x` from `y`. This was a breaking API change as the order of the arguments now has the opposite meaning. This will change the output of the `subtract` function in peoples code and will likely break lots of things.

## 3.5 Choosing Storage

Now that we've estimated how much data we are going to generate and thought about our priorities for preserving different subsets of it **we are at the stage of thinking about purchasing storage capacity.**

You will need to think about the question of how your data backup, recovery and integrity needs are met by whatever solutions that you choose to use.

### 3.5.1 “The Cloud”

When considering cloud storage solutions remember that **the cloud is just someone else's computer** and you'll have to pay to rent and maintain the hardware underpinning what resources you use. **If all your data is in the cloud you are at the mercy of your cloud provider's pricing** and will have to pay whatever rate they decide to charge you if you cannot readily move your data. I would **advise against using cloud storage as your primary storage medium. Have local storage and use the cloud as a backup.** In addition **avoid proprietary cloud storage solutions, make use of standardized approaches** that will let you move your storage to a different vendor with relative ease if pricing changes.

I try to adhere to the rule “Never do in the cloud that which you could not (theoretically) do on premises.” This translates to only using cloud services which you could host your own drop-in replacements for. Be as close to ‘just changing some config files for locally hosted API endpoints and restoring from backup and you can pickup where you left off’ as possible. This gives you maximum optionality in choice of hosting provider and bargaining power on prices.

For very large datasets the ability to ‘bring the compute to the data’ offered by cloud solutions and the ease of collaboration in cloud computing environments is indeed highly valuable. This utility is not however limited to the ‘public’ cloud infrastructure operated by major providers like Amazon web services

and Microsoft Azure. Operating your own ‘private’ cloud infrastructure is increasingly accessible and there are some emerging standards in cloud computing which increase interoperability. Opening the door to ‘hybrid cloud’ solutions which might make use of local resources as well as resources from multiple ‘public’ cloud providers. If you are going to use a ‘public’ cloud solution I would recommend avoiding the likes of Google/AWS/Azure in favor of independent cloud providers with less proprietary ecosystems and vastly superior customer service such as [Linode](#).

### 3.5.2 Central/Institutional Storage Resources

Understanding your institution’s research data policies, infrastructure & resources is important to inform your planning. Speak with your IT, core facility, and often HPC (High performance computing) teams about their data policies and understand what data resources they can and cannot offer you. For longer term archival storage another place that often has some help to offer are any library sciences specialists at your institution.

- [Cambridge University Research Data Management Site](#)

Unfortunately there is often a lack of clarity about where the responsibility to ensure that you have adequate data storage arrangements for your research data lies. **This can easily lead to data loss if responsibility is not taken at the level of the research group with the PI setting a clear policy and ensuring that it is adhered to by lab members.** This article ([McInturff and Adenis 2022](#) [cito:citesAsRecommendedReading] [cito:discusses] [cito:agreesWith]) has some useful practical advice for setting up standard procedures for the handling of data in a research group. Important components of this can be seemingly simple things like [agreeing on file and directory/folder naming conventions](#) (see: Section 3.7).

**Is there sufficient capacity at a central resource or can it be expanded according to your needs?**

**Is their backup policy sufficient for your needs?** How do you access and use backups as an end user, what kind of control do you have over the data recovery process? Test it and see how recovering from a data loss actually goes.

**Does the central storage meet your performance requirements?** Data storage only accessible over a relatively slow internal network may not be up to the task of storing and serving data from an instrument that generates a lot of data quickly, or an analysis which reads a lot of data quickly.

Users of institutional facilities should ask these questions to ascertain how quickly and completely a central facility should be able to recover in the event of an incident. Is this recovery time compatible with doing your job during any downtime whilst things are fixed? Still closer attention should be paid to this question if you are [operating your own storage](#).

### 3.5.3 Operating Your Own Storage

As a **last resort** if you need to manage your data yourself here is some advice. I stress that **if you can get a professional storage admin in charge of managing your data** through the resources available at your institution **do this before embarking on operating your own storage**. Managing your own storage is **not a passive process it requires active maintenance and monitoring**.

Hard Drives (HDDs) die and **data on them becomes corrupt if not actively maintained**. Solid State Drives (SSDs) and other solid state storage media also die and **data stored on them also degrades over time without active maintenance**. Simply placing your research data on an external storage medium or keeping it on your local system and leaving it there is a bad idea for several reasons.

1. You may lose your data if there is an issue with that device and you do not have other backups.
2. Your data may be corrupted without you even knowing about it through 'bitrot' random bitflips in your imaging or sequencing data that change the file's meaning but do



not make the file unreadable. These events are rare in absolute terms because of the reliability of modern storage media but because of the sheer size of many of the files they are common enough to warrant taking precautions against them.

3. Other people who might need to access your data cannot readily do so, if they do not have an understanding of the idiosyncrasies of how you organised it.

**If you are looking to purchase your own small to medium sized storage solution (10s of Tb) I recommend a NAS (Network attached storage) appliance over an external HDD/SSD** as a NAS can be configured with some redundancy to help ensure the integrity of your data. There a number of commercial offerings in this space (e.g. [Synology](#), [QNAP](#)), few are terribly satisfactory in their implementations of data redundancy & integrity though. Importantly they are also not as readily scaled up to larger systems. In my view **one of the best options currently available is a device running a [TrueNAS operating system](#), or [XigmaNAS](#)** a set of free and open source operating systems developed specifically for storage appliances. The main reason for this is **they make use of the [ZFS file system](#) which has probably the best data integrity features of any filesystem** currently available. If you ever get yourself into such a mess that you need [professional help](#) to recover your data the professionals will thank you for having used ZFS.

**You will need some cooperation from your institution's IT team to setup a NAS on their network so be sure to consult with them first.**

If you need a small NAS for Your Lab pick yourself up a [TrueNAS Mini](#) from a local iXsystems distributor, you should also talk to them if you need a large storage appliance.

### 3.5.3.1 Redundancy and Raw vs Usable capacity

There are a variety of ways to implement data redundancy the term I'm using for various approaches to error detection and

correction. This refers to a whole family of mathematical tricks to detect and correct for errors in messages, ideally more efficiently than having to duplicate the whole message. The canonical example of this is a [Hamming code](#) named for their inventor for Richard Hamming, other examples include [erasure encoding](#). These techniques let you **construct storage systems such that if you lose some number of drives to technical failures you can still infer their contents from the information on the remaining drives**. Thus with redundancy you are resilient to data loss up to a level of simultaneous hardware failures that you can tune according to the level of risk you want to take with your storage.

If you are managing your own storage **you will need to account for how much redundancy you want in your data when planning how much storage you will need**. You will need to use some of your **raw** capacity if you add redundancy to your data. Thus the **usable** capacity that you have to actually keep files in will be reduced. If you are taking the almost always preferable route of using [central/institutional data storage resources](#) they have likely already given thought to data redundancy though you should always inquire about what measures have been taken so that you can plan accordingly.

**Redundancy is not a backup**, it **can** when used properly, **improve up-time** and **ensure data integrity**. However, **just because your data is redundant** within one system it **does not mean that it is properly backed-up**. Choice of redundancy method can also significantly impact on performance. For example a 2 way mirror roughly doubles read speed but not writes and means you need 2x usable capacity in raw storage - storage layout is a complex trade-off. Recovering from a hardware failure can still take a long time and impact on performance. Even if you don't lose data you still have to expend compute cycles inferring what was on the failed hardware and writing a new copy of it out to the replacement drive.

### **i** RAID & Drive Topologies

You will hear the term **RAID** (Redundant Array of Inexpensive/Independent Disks) used a lot in discussions of

configuring redundant storage, there are a variety of types of RAID.

Selecting a particular layout and degree of redundancy will depend on a number of factors and detailed discussion of which drive topologies to chose for which purpose are out of scope for this document. Here are some resources to refer to when making this decision. They emphasize making this choice for TrueNAS/ZFS systems which is what I recommend if you are operating your own storage appliance:

- Soliciting advice for your specific use case on these fora is likely to yield good results:
  - General [Level1Techs](#)
  - TrueNAS specific [TrueNAS Community Forums](#)
- ZFS specific advice for the more technical user:
  - [Choosing the right Zpool layout](#)
  - [Practical ZFS forum](#)

### 3.5.4 Cost, Availability, & Integrity

The more available your data and the higher your confidence in its integrity, the more it will cost to store and the more energy will be expended to store it. An archival copy on a slow offline tape drive is not very available and it's integrity may be questionable, but it's cheap in cost per Tb. If a hash is taken of the file when it was stored and the hash is stored in multiple copies so that you can have a high degree of certainty that the hash is correct then you can compare the hash of the file on reading to the one computed when it was originally stored. If they match the file is very likely fine if they don't you may need to go try another backup copy as there's likely no way of fixing the missing data and it may be difficult to assess the scale and importance of any corruption. (A well implemented system automates such hash comparisons so the end user doesn't have to do this manually - but it is a useful concept to be aware of

when thinking about these problems, and not all systems are well implemented.)

#### **3.5.4.1 Data Availability**

If my data is archived on a slow tape drive and I can't easily query it will it every be used? Depending on the nature of the data this might matter more or less. If I'm just going to be batch process the files then it matters less. If I've got raw sequencing data for example there's not a lot I'm likely to do with a fastq file that does not involve all of the file. This means that I only really need sequential access to the files and it might be fine to stick them in a tape archive with relatively minimal availability trade-offs. Loading the data from the tape onto temporary storage for alignment will probably take less or a similar amount of time to computing a full sequence alignment for that data (with current storage and compute speeds) so the time taken to retrieve the data is not massively out of proportion to the time that it will take to perform other tasks on the data once it is retrieved. This also requires that I have a fast and accurate record of the metadata for the files on the tape that I can search to see which ones I want to retrieve.

On the other hand if I've got sequencing data that's already been aligned then I might be interested in constructing queries that look a the number of reads meeting certain criteria over the same set of genomic coordinates in many different samples even potentially from numerous different experiments. This requires the ability to read data from within the alignment files but not the whole file, known as random reads. If I've got to read the whole file sequentially from a tape such queries would take a very long time. I might be able to speed things up by fast forwarding over the bits that don't have data in them from which I want to read based on an index of my file. If they are on a hard drive I can skip around between the needed chunks by moving the read head - which is faster than linearly scrubbing back and forth along a tape. Better yet if they are on an array of hard drives I might be able to read different chunks from different physical disks in parallel to speed things up. Lots of SSDs e.g. many using the NVMe protocol, can

parallelise reads within a single ‘drive’. Individual SSD devices are essentially small arrays of solid state storage devices, and so can retrieve the data still faster than an arrays of HDDs. This lets me improve not just the linear speed at which I can write a continuous block of data but the speed at which I can read and write at random within the data known as “IOPs”. **All of this is abstracted away by various performance specification metrics and we don’t usually need to worry about the implementation details but a high level picture of these considerations helps to put the performance numbers in context.**

### 3.5.4.2 Data Integrity

If I’ve got 3 copies of some data on 3 drives it’s a bit of additional work to compare their hashes to the original hash which I hopefully also stored in multiple copies when I originally archived the files. If all of the hashes don’t match what can I do about it? I might be able to painstakingly manually assemble a complete file from the copies assuming different parts are corrupt in each, but this is probably beyond my skill and both slow and expensive.

Hard drives are flakey they wear out, they break and occasionally bits stored on them get flipped at random this is also true of many SSDs especially the high density NAND flash in common use today. However it is possible to make a reliable whole from unreliable parts. This is discussed in [Section 3.5.3.1 Redundancy and Raw vs Usable capacity](#) advanced file system like ZFS can automatically compare the hashes of blocks of data on disk to copies of those hashes stored in the filesystem metadata and look for or infer the correct value of this block from redundant versions of your data. This takes energy as you must periodically ‘scrub’ your data looking for and fixing such inconsistencies, but this is readily automated.

Data storage is essentially transmitting data forward in time and is thus subject to the same kinds of physical constraints on ensuring the integrity of a transmitted message originally identified by Claude Shannon. Data integrity is probabilistic, you must decide the level of risk that you are willing to take

with your data becoming corrupted and choose a technology accordingly.

## 3.6 Securing Data

*This advice is good general cyber-security advice that anyone using a computing device could benefit from following given the modern threat landscape of just being a human on the internet.*

If you handle personally identifying information, or other data to which general access is restricted you should adhere to the policies of the institution granting you access to this data. Most of the time as a researcher you should not find yourself responsible for administering the security for access to data to which access must be restricted for one reason or another. This requires IT security expertise and will generally be performed for you by a 3rd party, if such expectations are being placed on you and securing them properly is out of the scope of your responsibilities and expertise you should raise this with your supervisor/institution/funders. This is one of the reasons to be hesitant about operating your own storage, you are assuming the burden of responsibility for properly securing and backing up your data. If you are not operating storage with sensitive data but merely accessing it you still have security responsibilities. **There are some general counter measures that you can take to improve your personal electronic security and that of any potentially sensitive data that you handle.** Specifically you should take care to follow best practices with your access credentials to restricted data to ensure that unauthorized persons cannot use your credentials to access the data. In addition you should take great care to secure any local copies that you may have even temporarily made of such data.

! Use a Password manager

**Why?** Every account should have a different unique password so a breach of one is not a breach of them all. No one

can generate a different long random alphanumeric password with special symbols for every account you create and remember them all indefinitely.

**What do they do?** A password manager provides a *secure place to store a long complicated and randomly generated password for all of the things you need to login to*. But many have additional useful features. The *convenience of auto-filling passwords* for you. Additional features like securely sharing credentials for shared accounts within organisations including TOTP codes, (Those 6 digit codes authenticator apps generate - more on these later), which can otherwise be a challenge to share securely. Checking to see if passwords in your accounts have appeared in lists of breached credentials. The ability to *store files* with private keys or other secrets.

**Which one should I use?** I would advise against using the password management features built into your web browser, disable these in favor of a dedicated tool which does this better. Whilst many are available Bitwarden [?@sec-bitwarden](#) is the only password management solution that I would currently recommend to the general user, [KeePassXC](#) is also a reasonable choice for the more technically inclined.

**How should I use one?** Secure your vault with a master password or passphrase which you have **never used anywhere else before** so that it cannot be found in any past data breaches. Secure your vault with a second factor (more on these below). *Install the app(s) and browser extensions on the systems that you use. Gradually migrate all of your existing accounts over to your vault, each time you sign into a new site take a moment to add it to your vault. Start with major accounts that you are likely to be logged into all the time e.g. Google and work accounts, it may be helpful to disable browser auto-filling if you use it. This will help prompt you to switch to your password manager, though you may want to wait until you have completed your migration to delete any passwords stored in your browser*

**If you must share credentials for an account it is**

**best to do so through a password manager, or failing that a secure link. Never send plaintext credentials in email!** If you can avoid making use of a shared account you should, use groups where everyone who needs permission to perform an action on a service has their own account with those permissions. If this is not possible 1. complain to the service 2. store and share your credentials properly. The best option is something like a [bitwarden organisation](#) where all the parties who need access have account and permissions on a shared collection of credentials. If you must send credentials to someone who is not in a Bitwarden organisation you should use temporary secure links such as those provided by [bitwarden send](#) or similar services, these can be set to expire after a certain time and number of downloads, that way the link is useless to someone who later gains access to the communication. These should ideally be sent through an encrypted messaging app such as [Signal](#), [SimpleX](#) or a [Matrix Client](#) and not plaintext email as extra surity against their interception. You can also split a one time password and an access link between two separate communication channels - there is little point sending emails with a password and a link. It is best to have such out of band communication channels established **in advance and in person** so that you can be confident the intended recipient is the one at the other end. If sharing things in groups remember to practice the [Principle of least priviledge](#) and only give people access to the credentials that they need.

! Use Multifactor authentication (MFA/2FA)

### 3.6.1 What is Multi/Two-factor Authentication?

A password is one factor, additional factors mean you need more than just 1 credential to login. Adding another factor means that if a malicious actor gets your password they still need this 2nd factor to access your account. You are not always asked to login every time using an additional factor, when you have logged in to a service from a device



once it may remember that device and only prompt you for your password and not your additional factor on most subsequent login attempts.

### 3.6.2 What 2nd factor should I use?

**The best second factors are hardware security tokens** such as Yubikeys. *In these devices the secrets are physically stored on the device and (usually) nowhere else this mean someone has to physically steal your particular key to login to your account. This does mean that it is advisable to have more than one and keep the other(s) in secure locations in case you loose or break your key.*

**Second best are those 6 digit codes generated by an authenticator app** such as [Google authenticator](#) or Aegis ([play](#), [F-droid](#)) **these are called TOTP (Time based One Time Passcodes)**. For more details expand the box below on TOTP

**SMS (text messaging on your phone) is better than nothing but much less secure than the other options.** If you have other MFA methods available SMS should be disabled where possible as *your security is only as good as your weakest 2nd factor.* (if someone can spoof your phone number and intercept an SMS MFA code *your hardware token does nothing to help you if you are allowing both methods to work as a 2nd factor* on your account).

**Avoid push notifications on your MFA**, set yourself up to only authorize logins when you are making a careful deliberate choice to do so. Popular MFA solutions including those from Google and Microsoft often offer a prompt on your mobile device that will let you simply tap a notification to authorize a login attempt. This convenience feature has been exploited by attackers to bombard people with notifications until they authorize a login by accident in an attempt to stop the notifications. Stick with the classic TOTP codes if you can, these give you a moment to think about what are doing compared tapping a notification.

To reiterate: **MFA is very effective and dramatically re-**

duces how easy it is to break into your account, use it wherever you can!

### 3.6.3 Setting up a password manager & MFA

#### Password choice

Constructing a good master password for your vault.

- One that you have **NEVER** used before, not even a variant on it
- Does **NOT** contain any potentially guessable or public information about you, e.g. birthdays of yourself or relatives, pets names, address information, this includes any clever interleavings, remixes, or anograms of these values etc.
- Long
- Memorable
- Easy to type

These last three often seem to be in conflict however they need not be. Passphrases, that is a series of dictionary words, with perhaps a couple of numbers and special characters are the easiest way to achieve all of these features at the same time. This  [XKCD](#) has some sage advice on password creation:

**WARNING** modern GPUs can now perform password guessing at incredible rates on the order of millions per second so adjust accordingly and for highly sensitive applications which might be subject to a targeted brute force attack consider passphrases of seven or more words in length.

(Whatever you do don't use correct horse battery staple as your passphrase though - I just wrote that from memory long after the last time I read the comic so that passphrase is well and truly burned)

## **i** Bitwarden

### Getting Started with Bitwarden

- **Text** [Bitwarden official getting started documentation](#)
- **Video** [Bitwarden Password Manger Beginners Guide](#)

For more general information see [?@sec-use-password-manager](#), and [?@sec-mfa](#).

## **i** TOTP (Time based One Time Passcodes)

TOTP uses a secret key stored on your device combined with the current time through a hash function to generate the one time codes. This key is stored on your device and should be securely backed up, **google authenticator does this via your google account so be sure to secure your google account or people may be able to steal your TOTP keys by ‘restoring’ your google account to their device.** Note that using google authenticator means if you loose access to your google account you likely loose access to all your other accounts as well. **The same applies to Microsoft’s authenticator.** For an app like Aegis with no cloud backup of your MFA keys export an encrypted copy of your TOTP database and securely back it up to protect from device loss.

**You will often also be given a set of backup one time codes to use in the event that your device is lost,** be sure to make a secure copy of these, it may be safest to simply print these directly (don’t save any intermediate files) and keep the paper copies in a secure location. If you make encrypted electronic copies of these **avoid placing them only in the same place where your TOTP secret keys are backed up or in the same place as your password is stored.** They are a backup for your 2nd factor so if you store them with your password and that storage is breached you no longer have

2 independent factors.

**TOTP is an open standard (see: [RFC-6238](#)) not everyone implements it according to the standard and some try to add their own stuff on top if you can stick to strict implementations of the standard.**

### **i** Hardware Security Tokens

Hardware security tokens such as the Yubikey products produced by Yubico can implement a number of standards to provide cryptographic functions that include authentication i.e. letting you login to things. The practical upshot is a cryptographic secret of some kind exists *only*<sup>3</sup> on the physical token and *never* leaves it such that in order to steal the secrets and breach your account an attacker must physically steal the key. This is in contrast to TOTP for example where the secrets whilst well protected could in theory be stolen remotely from the device on which they are stored, though this would likely require exploiting unpatched software vulnerabilities. Choose security tokens that implement open standards see the [FIDO2 alliance](#).

- **Text** (*quickstart*) [Official Yubico Yubikey setup guide](#)
- **Video** (*quickstart*) [Setup a 2FA Key for MAXIMUM Online Security! \(Yubikey Tutorial\)](#)
- **Video** (*longform*) [YubiKey Complete Getting Started Guide!](#)

---

<sup>3</sup>Yubikeys can store cryptographic secrets which are backed up elsewhere e.g. you could generate a PGP key and save it to your Yubikey and must therefore take appropriate precautions when generating and backing up that key.

E.g. generating that key in an ephemeral environment that only uses a RAM disk so it is only saved to the persistent storage that you want and never your main drive (see [tails OS](#)).

## 💡 Other General Security Advice

**Keep yourself up to date** This is a rapidly moving space advice can go out of date quite quickly here are some good but accessible news sources that you might be able to work into your media diet.

- **Podcast** (*The ‘Most Entertaining’ security podcast*) [Smashing Security](#)
- **Pocast** (“*Computer security & privacy for non-techies*”) [Firewalls don’t Stop Dragons](#)
- **Text** (*in-depth investigative*) [Krebs on Security](#)

Other measure that you can take to ensure that your security practices best protect any sensitive data that you are handling include:

- **Keep your software up to date** in particular your operating system and web browser
- **Use full disk encryption** on your computing devices. Note that you should always backup before enabling this. You should also take great care with the storage of the password because if you loose it, and the encryption is properly implemented, there should be no way of recovering/resetting your password. This is now standard on most newer mobile devices.
  - **Windows** Windows’ whole disk encryption options leave something be be desired. On non-pro tier editions ‘Device Encryption’ which stores an encryption key in onedrive is available with the more fully featured ‘bitlocker’ only available available to those with professional tier licences (see: [Device Encryption in Windows](#)) 3rd party options like [VeraCrypt](#) exist but being non-native solutions not explicitly ‘blessed’ by the Windows developers they are more prone to breaking when Microsoft occasionally changes things which affect the way that they function. Unfortunately Windows’

solution also conflates 2 distinct functions that of encrypting your data and that of tying your drive to a specific system so recovering data encrypted on one system by putting the drive into another may be a problem even if you have your encryption password.

- **MacOS** This can be enabled with FileVault2
  - **Linux** Many modern and popular Linux Distributions such as Ubuntu & Fedora present you with a whole disk encryption option using the LUKS system when you install them. As always in Linux if you know what you are doing there is more than one way to to it and it should be possible to set up full disk encryption on any Linux setup. I would keep an eye on [shuffle cake](#) in the future for plausibly deniable encrypted partitions in case you are ever in jurisdictions that can legally compel you to disclose your encryption keys.
- **Disable any cloud based backup systems which sync your data not end-2-end encrypted** to a remote server
    - Your shiny new fully encrypted disk does you no good whatsoever if you are now syncing the plaintext versions of all that data to a non end-2-end encrypted cloud backup with a service like Google drive or iCloud.
    - 3rd party services exist to enable local encryption of your data using keys that you hold before it is synced to the cloud e.g. [boxcryptor](#) & [cryptomator](#) who support multiple platforms/cloud ecosystems
    - Alternatively you could use cloud storage platforms which are open source and end-2-end encrypted by design e.g. [Internxt](#)
  - **Use only trusted machines on trusted networks** to login to sensitive systems If you are on public Wi-Fi use a secure VPN to provide extra pro-

tection of your web traffic

- **Secure your home network** determined attackers may try to steal login credentials from you when you are on a less well secured network such as on your home wi-fi
  - If you have any IOT/smart devices such as TVs or other appliances keep them on a separate virtual network. An easy way to do this is to use the guest network that is usually available on commercial home routers.
  - Use a strong passphrase on your network and router admin interface.
    - \* WPA2 passwords have a max length of 63 characters so I advise using a random alphanumeric with caps/lowercase and symbols generated by your password manager here to maximise entropy in the available space. If you can move to WPA3 only which has better encryption, be sure to disable automatic WPA2 fallback to prevent older incompatible devices from compromising the improved security if possible.
  - Keep your home router firmware up to date.
  - If you host any services on your home network it is best to have your own VPN rather than opening ports on your home network to the internet at large. I recommend [tailscale](#) a [wireguard](#) based mesh VPN.
- **Mobile**
  - keep your OS up to date, upgrade your device if it no longer receives security patches from its manufacturer (support windows are often criminally short, pay attention to security support windows when purchasing a device).
  - Install as few apps as possible on your phone to minimize the attack surface, use the mobile browser versions of services instead of the app

wherever you can. mobile web-apps running in your browser are better sand-boxed than ones installed natively.

### 3.6.4 Video Summary

This covers most of the operational security advice that I've given above if you want it in a video medium. I cover some points this misses in the text though so I'd recommend reading it as well.

NB DNS over HTTPS can cause issues resolving intranet sites and interfere with DNS based site filtering which might cause you inconveniences, it is quite effective at protecting your DNS queries but may not be worth the inconvenience.

## 3.7 Naming Files

*For an easy life and happy bioinformatics collaborators*

### 3.7.1 Dates

If your file contains a date **ALWAYS** use the [iso-8601](#) date format YYYY-MM-DD **NEVER** any other date format!

The main reason for this is a standard semantic sort operation on the filename will put this date format in the correct order unlike other date formats. For this reason If your filename contains a date it is often a good idea for the date to come first in the filename.

“There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors.”

- Leon Bambrick



### 3.7.2 No spaces or special characters!

This is for at least **3** reasons.

Spaces and special characters can have special meanings in command line interfaces (CLI) and cause command-line users inconvenience trying to ‘escape’ them. ‘My Document’ must be referred to as `My\ Document` in most CLIs, this gets tedious if there are lot of spaces. You can quote a file with special characters in their names e.g. “My Document” but this usually breaks our filename auto-completion meaning we have to type the whole name by hand .

Using underscores or `_` or hyphens `-` in place of spaces is best. Specifically it is best to use hyphens to delimit ‘sections’ of your filename and underscores to split up multi-word ‘sections’. A good rule of thumb is **can I turn my filenames into a table by splitting them at the hyphens?** Think of each ‘section’ as a column<sup>4</sup>.

Experiment\_1-Sample\_1-WT-DrugA

Experiment-1\_Sample-1\_WT\_DrugA

**Always use explicit missing or negative values in file names!**

If you have an experiment where you are adding something to some samples also add the lack of that thing to the other files, aim for an explicit value in every row and column of the table.

Experiment\_1-Sample\_2-WT-No\_Drug

Experiment\_1-Sample\_2-WT

At the end of a filename this makes less of difference as you get a missing value in your last column but in the middle it’s more of a problem as now it can end up in the wrong column!

---

<sup>4</sup>One reason for this is keyboard navigation CLI people don’t like to use the mouse if we can avoid it, it slows us down. If you hold down `Ctrl` and use your left and right arrow keys you can jump through text 1 word at a time in many text editors, `Ctrl+Shift` permits you to select text with the keyboard in this fashion. `_` does not count to this action as the end of a word but `-` does. Thus I can highlight each section / column within a filename with a couple of keystrokes - very handy .

Experiment\_1-Sample\_2-Mutant-DrugA

Experiment\_1-Sample\_2-DrugA

This might work but an explicit value is better:

Experiment\_1-Sample\_2-DrugA (note the double hyphen)

File names with missing components that confuse which columns parts of the name should end up in can be a real pain to parse and result in your bioinformatician having to do something *manually* (we hate that).

For example these file names can very easily become a nice table with this naming convention:

```
filename <- c( ①
  "Experiment_1-Sample_1-WT-DrugA",
  "Experiment_1-Sample_2-WT-No_Drug"
)
## filename <- fs::dir_ls("directory/with/files") # <2>
filenames <- tibble::tibble(filename = filename) ③

filenames %>% ④
  tidyr::separate( ⑤
    filename,
    into = c("Experiment", "Sample", "Genotype", "Treatment"),
    sep = "-"
  ) %>% ⑥
  mutate(across(everything(), ~gsub("_", " ", .x))) %>% ⑦
  gt::gt() ⑧
```

- ① Making a table with a column called 'filename' which contains the names of all the files.
- ② In reality we'd just read all the files from the folder rather than explicitly listing them like below.
- ③ List of filenames
- ④ Give the filenames table to the next function
- ⑤ Split the filenames column into: Experiment, Sample, Genotype, & Treatment columns by '-'
- ⑥ Give the new table with separated columns to the next function

- ⑦ We can even easily make it prettier by swapping out the `_` for spaces! Alter all the columns by substituting `'_'` for `' '`,
- ⑧ Draw a pretty version of the table

Experiment	Sample	Genotype	Treatment
Experiment 1	Sample 1	WT	DrugA
Experiment 1	Sample 2	WT	No Drug

To retain as much information as possible in your filenames you can devise alternative encodings from which better formatted names can be re-constructed e.g.:

*Gene1*<sup>+/-</sup> to **Gene1\_p\_m**

*Gene1*<sup>+/-</sup>; *Gene2*<sup>-/-</sup> to **Gene1\_p\_m\_\_Gene2\_m\_m**

## 4 Working With Data

*The challenges of working collaboratively on data and some solutions*

### 4.1 Electronic Lab Notebooks (ELNs)

*Note: The ELN function is also frequently combined with a LIMS (Laboratory information/inventory management system) solution as the two are often closely interrelated. They are however distinct functions and you may wish to pick different tools for these functions.*

*See also the turing way sub-chapter on ELNs which I have contributed to*

ELNs are incredibly useful and massively expand on the utility of a paper lab notebook. Links, images, multimedia, collaboration, search, and sharing, integration with LIMS systems, and much more. It is, however, important when choosing an ELN solution that you **do not give up the advantages offered by a paper lab notebook.**

There are an immense and baffling array of options in the Electronic Lab Notebook space. Many organisations offer software that purports to solve the problem of electronic lab notebooks. So choosing a suitable solution can be a major headache. **The choice of ELN is an incredibly important decision and one that your lab/institution will likely have live with for years or even decades. You are putting the record of your research into the hands of the tool that you choose, and entering into a long-term relationship with the provider of your ELN solution.**

**Consider the differences between a paper copy of a lab notebook (PLN) and an electronic copy. Consider also**

“rigour, reproducibility and robustness. These remind us of the reason why we became scientists in the first place.”

- Levi Garraway (2017)

**which are the properties of a paper copy that it is important that you are able to retain when adopting an electronic alternative.**

A paper lab notebook is **physically under your control** you (or more likely your institution) own it. You can control access to it physically. Your physical possession of this resource means that it would be **difficult for anyone to prevent you from accessing it**. It would also be **difficult for anyone to charge you a fee in order to continue using it**. You **do not need any specialist tools in order to access it's contents**. You are not dependent on the functioning of any complex systems like computer networks in order to be able to use your paper lab notebook. You **do not have to agree to a 'terms of service' or 'end-user license agreement' with a 3rd party**, (the terms of which are likely subject to unilateral alteration by that 3rd party), **in order to use and retain access to your lab book**.

**If the provider of my paper lab books goes out of business it has almost no bearing on my ability to continue doing my work**. One paper notebook is much like another, **finding a new provider is pretty easy**. **Changing providers does not impact on my ability to access my past notebooks or to continue operating with the same workflow** in my future ones. This is not necessarily true of ELNs. **Few active measures are needed to maintain the data in your notebooks**, they are vulnerable as they exist in only one copy but as long as they are kept in a cool, dry and dark spot they will likely last decades. **Electronic data requires much more active upkeep**.

**Lab notebooks perform an archival function** and proprietary formats are antithetical to this as they assume the institution which can act as a gatekeeper to the use of the proprietary format will outlive the need to archive the material. **When choosing an archival format one seeks to maximize the likelihood that one can recover the relevant information from that format**. Using a proprietary solution is talking a needless risk with the future of your data. Your data's fate can become tied to that of the firm, or project within a

firm, that develops and operates the software that you use to store your lab notes.

When looking for any piece of software the first question that I ask is: **“Is there an Libre / open-source solution to this”**, If it is a web app I ask: **“Can I host my own fully featured instance, should I need to?”**. I also ask: **“is there a large community using the project, does it have some institutional backing of some kind?”** This might take the form of a company which sells service contracts, or offers paid hosting ideally with feature parity<sup>1</sup> with a self-hosted option. Or perhaps a foundation or other non-profit/academic organisation **with robust funding**.

Open solutions provide me the assurance that If I do the appropriate preparatory work I should be able to access all of my data in it's native form by running the ELN application in a VM or similar reproducible computational environment in the future should I need to. Even if the tools are no longer maintained and in a state that can be used in production. They can still be used to read the data and interact with it in the same way. The data will also likely be stored in an open format from which it can relatively easily be extracted and ported to a new format.

This **recent review (Higgins, Nogiwa-Valdez, and Stevens 2022 [cito:citesAsRecommendedReading] [cito:critiques]) provides a good overview of considerations when adopting an ELN solution. It cover such things a regulatory compliance that I have not touched on here.** It does occasionally appear to conflate open-source solutions with self-hosted ones which need not necessarily be the case. This [guide on choosing an ELN from Simon Bungers of Labfolder](#) is also worth a read. Some companies will let you host proprietary apps on premises and you can pay for 3rd party hosting and administration of open source applications. This is important as **if you don't have the expertise or**

---

<sup>1</sup>Feature Parity - meaning that the self-hosted version offers all the same features as the paid hosted version. I.e. there are not features locked behind a pay-wall, this business model tends to trend increasingly closed and can lead to tension between the in house dev team and the community over implementation of paid features in community versions.

**internal resources to administer a self-hosted instance of an open-source ELN solution you can still pay a 3rd party to do this for you in which case you get the benefits of professional support and the reassurance of an open solution.** You should still take regular local backups of exports from your hosting provider from which you could restore your ELN system with different hosting. This means that you **retain the option to change providers as the hosting and support are no longer vertically integrated parts of the software as a service (SaaS) experience** for you.

#### **i** ELN picks

The only ELN/LIMS software solutions that I have so far identified that meet my initial screening criteria are listed here. They are each quite different but have many of the same core features. For example rich text editing in a web browser. The ability to upload files. Sharing and permissions based on roles/groups.

##### **4.1.0.1 eLabFTW**

- Laboratory resource scheduling feature for booking things like hoods and microscopes, automatic mol file previews for molecules and proteins & support for free-hand drawing.
- The eLabFTW [site](#) and [documentation](#) there is also a [demo](#) deployment that you can try out
- Self-hosting is *relatively* simple [according to the documentation](#). There is also a [paid support tier](#) which would be recommend for any larger deployment to support the ongoing development of the project.
- [Paid cloud hosting](#) is available from the developer in a geographical region suited to your needs, a more expensive tier with hosting in France compliant with additional security and privacy certifications is available.

#### 4.1.0.2 openBIS

- Good features for integrated metadata management e.g. linking to ontologies / controlled vocabularies. This is based in a flexible object system for making similar entries.
- openBIS has an API and can integrate with jupyter-hub for electronic lab notebooks.
- Very feature rich LIMS system with optional integration of stores management with protocols and experiments including keeping track of bar-coded stocks.
- You can get a feel for it in the [demo](#) deployment.
- openBIS is a bit more complex to administer [based on its documentation](#). It's a slightly older and more complex project than the others on this list, meaning it is very featureful and well tested against the needs of the groups at ETH Zurich.
- Developed at ETH Zurich openBIS can be hosted for you under the openRDM service operated by ETH Zurich scientific IT services. No fixed pricing is available cost would be dependent on your specific needs

#### 4.1.0.3 OSF

- OSF is oriented towards sharing and collaborating on your work, including the ability to generate DOIs and host pre-prints directly on the main instance.
- It is free to use OSF at the main instance at [osf.io](#) so you can try it out there directly. For larger data you must provide your own additional [storage addons, available from a number fo cloud storage providers](#).
- Whilst OSF can be hosted yourself this is presented by the project as for the purposes of development, and is not directly available as a paid service.
- Strong sharing features, makes it easy to take your ELN and make it, or parts of it, public.



### 💡 Notes Beyond ELNs

There are additional sections covering the management of types on information which don't necessarily fit into an ELN solution for more general, personal or informal notes see the short section on Personal Knowledge management Section 9.2, for bibliographic information management the section on Zotero Section 9.1, and for passwords the section on ?@sec-bitwarden.

## 4.2 Code vs GUIs for provenance and repeatability

The **choice of** analyzing your data in a **graphical or GUI (Graphical User Interface) tool** such as a spreadsheet, or statistical analysis and plotting tool **or** doing so in **code in a programming language such as R or Python** is a significant one, though the two are **not always mutually exclusive**. This is sometimes a choice made for us by the availability of tools to solve our specific problem being only graphical or only command line (i.e. code). On many occasions however we are presented with a choice between the two.

Working in a graphical tool is typically, though not always, faster to pickup and learn for those with no prior coding experience than interacting through code and can make it easier to quickly get started with data analysis with a shallower learning curve.

Working in **code typically provides much better provenance** for the data and **implicitly documents every step taken in the analysis of your data**. If working with a **graphical tool** it is commonplace for the steps taken to **require manually and often in-exact instructions to repeat the same analysis**. This requires that such steps be carefully documented, and meticulously followed by someone attempting to reproduce your work. Both of these steps leave **significant room for** a class of **'operator' error**, failing to unambiguously document a step, misinterpreting or having to

“Code is text, code is readable, code is reproducible”

- [Hadley Wickham](#)

guess at an ambiguous step or just random mistakes. Whilst these difficulties are hard to avoid in lab protocols where physical steps must be described they are **theoretically avoidable in computational analyses** which reduces to a series of unambiguous mechanically executed steps. This can result in its own class of errors, bugs not spotted which lead to widespread errors if undiscovered in widely used tools for example so there are some trade-offs.

**A graphical tool which permits you to define a series of operations, export these instructions to a file and import that process into a different session is a step up over one in which the user must repeatedly manually specify an action.** However such approaches are usually not quite as robust as code, software versions change and user interfaces no longer match the instructions or can no longer import the files from older versions. **This can be especially difficult with proprietary or SAAS (software as a service) solutions where access to older version of the software is not available.** *It is much easier to maintain the equivalent of a lab notebook for you computational analyses if you are able to do so in code than it is do the equivalent when using a GUI tool.*

Some **GUI tools which generate/edit code snippets based on a GUI wrapper** or produce a file containing manual annotation information can provide a bridge for things that are just easier to do graphically and purely code based solutions. These sorts of **hybrid solution are available for certain tasks and make it possible to have a primarily code first workflow augmented by GUI assistance when needed/desired.** This document is an example of this sort of workflow. I'm currently writing it in [RStudio's visual editor mode](#) that resembles an ordinary WYSIWYG (what you see is what you get) word processor with all the bells and whistles like automated reference management integrated with [Zotero](#) but it's actually generating well-formed Rmarkdown syntax.

Another useful example of this for generating reproducible figures with imaging data and graphs in [Inkscape](#) with [imageJ](#) is

Jérôme Mutterer's<sup>2</sup> [inkscape-imagej-panel](#) plugin<sup>3</sup> .

*A section from Hadley Wickham's 2019 Keynote at EMBL covering the merits of computational notebooks for reproducible science.*

## 4.3 Reproducible computational analyses

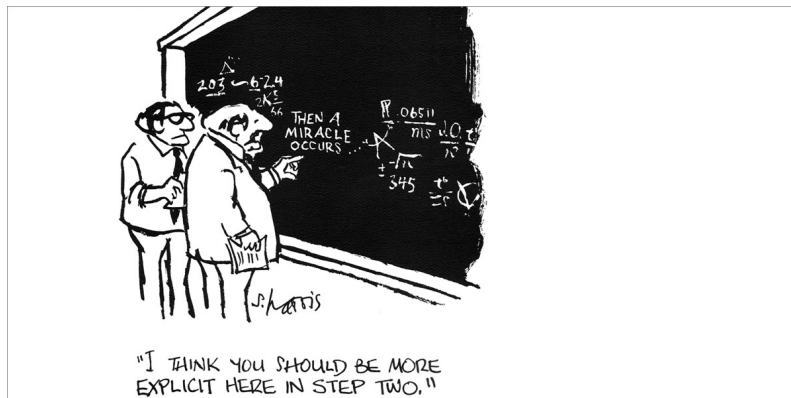


Figure 4.1: A miracle occurs - Sidney Harris



To reproduce, or indeed to easily collaborate on a data analysis project you need shared access to three things:

- **Code / Documentation**
  - The source code and its dependencies that spell out the steps taken in an analysis. The

“In science consensus is irrelevant. What is relevant is reproducible results.”

- Michael Crichton

<sup>2</sup>Jérôme's CNRS page as ORCID is a bit sparse

<sup>3</sup>inkscape-imagej-panel resources:

- [git repo](#)
- [slide deck](#)
- [Flashtalk](#)
- [Tutorial Video](#)

comments, context and motivation for writing the code that you did, how and why it works the way it does.

- **Data**
  - The inputs to that code both larger datasets and configuration options / parameters used.
- **Compute Environment**
  - The computational context in which the code was run. Operating system, Package versions, Configuration, etc.

We will cover a number of technologies in the following sections each of which solves a different aspect of the problems associated with performing, collaborating on and sharing reproducible computational analyses. Then we will look at a tool which brings many of these technologies together into a single relatively easy to use platform, [Renku](#). If you are in a hurry you can skip directly to the [Renku section](#) Section 4.8 and revisit the intervening sections as needed though I'd suggest at least skimming them to get a little context.

### 4.3.1 Source Management

*AKA version control or source control*

When working with code at any scale beyond a few small scripts (and sometimes even then) it is highly advisable to use a **tool to keep track of the changes** that you have made to your code. This is especially true if you are collaborating with others as such tools usually also feature utilities to help you to merge code developed by **multiple people working on the same project asynchronously**. The *de facto* standard tool for this is [git](#), it is widely used and there is much tooling built around the core git software.

Using **git in a data analysis project is also a bit like using a lab notebook**. Whenever you take a snapshot of your

**git** - Track changes but OP<sup>4</sup> (*and a bit more complicated*)

- Richard J. Acton

project by making a ‘commit’ you accompany it with a ‘commit message’ giving a brief description of why you did what you did. **A digital file is not necessarily like a lab notebook in that a physical notebook has a chronological order where you can see the history of what you did, when and why.** In contrast a digital file that you change over time just has its current form and does not retain a history of its changes. **git adds this chronological dimension back to digital projects** letting you time travel through the history of your projects this can be very valuable. For example if you want to be able to get back a result exactly as you generated it before you updated your code.

It is simple to learn basic git operations but its underlying structure can be a bit conceptually difficult grasp. I recommend taking the time to form a good mental model of git’s workings if you are going to use it regularly. If you want to understand more and perform more advanced operations or indeed just fully understand the simple ones. See the learning resource below for some more in depth material on git.



Figure 4.2: git

Before we delve a little more into git I'm going to introduce another concept - Literate programming. Source code is after all just text and many of these same concepts translate well to collaborating on prose.

### 4.3.1.1 Literate programming

**Literate programming** is a paradigm for writing code interspersed with natural language prose, or *vice versa*. The concept was introduced by Donald Knuth in 1984, an early example of Literate Programming was *LaTeX* the document authoring and typesetting language still popular with anyone writing mathematical notation on computers. **Literate programming is now very popular in ‘computational notebooks’ used by data scientists**, these are in many ways **the computational equivalent of a lab notebook**. *This* is a literate programming document, it leans heavily towards prose and does not contain much code but I can easily include some, check it out:

```
2 + 2
```

```
[1] 4
```

The literate programming tool I’m using is called [Quarto](#). With it I can write plain text documents which include snippets of code in R or a variety of other languages and format my text with a simple markup syntax called markdown. As I mentioned above **I’m currently editing this document in a WYSIWYG editor much like the word processors with which you are likely familiar** that generates the Quarto/markdown formatted text. **Markdown is however remarkably simple and very easy to learn** and I regularly switch between source and visual modes with minimal friction.

This is an extremely powerful tool for generating and properly documenting my work and indeed for outputting it for different publication formats, a concept called [single source publishing](#). This document for example is automatically published as a [website](#), a [pdf](#) & and [epub](#) every time I commit and push changes to the [gitlab repository where it is hosted](#). You can even get your markdown formatted according to the requirements of many journals with `{rticles}` or [Quarto Journals](#). Thus the published output from this source document is tightly coupled to the code in it. Any code I write here is re-run when this document is built (unless I cache the results).

“Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.”

- Donald Knuth

## **i** Quick markdown syntax

Here's is a quick markdown syntax rundown (~90% of all the markdown syntax you'll ever need):

```
# Heading 1
```

```
## Heading 2
```

```
### Heading 3 etc. {#sec-h3}
```

```
[hyperlink text](url)
```

```
footnote [^1]
```

```
Inline references were used by @smith2021, it has been claimed [@jones2022] (not inline)  
cross refernces @sec-h3 using the h3 short alias
```

```
**Bold**
```

```
*italic*
```

```
***Bold & Italic***
```

```
`inline code`
```

```
- Bullet point
```

```
  - nested
```

```
- point
```

```
1. Numbered list
```

```
2. another thing ...
```

```
> Quotation - unattributed
```

```
![alt text](/path/to/an/image.png)
```

```
...
```

```
generic code chunk
```

```
...
```

```
 $inline~math~x^y$ 
```

```
$$
```

```
math~chunk~\frac{x}{y}
```

```
$$
```

A

Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions.

- John Gruber



```
[^1]: callback!
```



Markdown comes in a number of ‘flavors’ usually a superset of the [commonmark](#) specification / reference implementation which extend it with additional features so there is some variation in syntax, many tools have built in [linters](#) to check/auto-correct any syntax not supported in a given flavor.

#### 4.3.1.1.1 Literate Programming Learning resources

[Quarto](#) is a scientific and technical publishing system that uses markdown

Whilst still from the Posit (formerly RStudio) team it is more language agnostic than Rmarkdown which may be familiar to R users and can be installed as a separate command-line utility without R dependencies. It can use jupyter notebooks as a source document format and integrates well with [vs-code](#) as well as RStudio. It also unifies the variety of different pre-processing steps for different output formats previously performed by a family of R packages bringing us closer to true [single source publishing](#).

**if you are starting in 2023 begin with Quarto** it’s basically the same as Rmarkdown but better and highly backwards compatible with R markdown. These texts are still relevant but you can now do a lot of cool new stuff in Quarto in particular ‘fenced divs’ are awesome.

- **Text** (*documentation*) [R Markdown: The Definitive Guide](#)
- **Text** (*cheatsheet*) [Rmarkdown cheat-sheet](#)

#### 4.3.1.1.2 Jupyter Notebooks (Python)

Jupyter notebooks are another major player in the scientific computational notebook space and originate in the python community with the iPython interactive shell. They are run from within your web browser either locally or on a remote jupyter hub. They also make use of markdown syntax for literate programming.

Unlike Quarto/Rmarkdown they are not tightly integrated with an IDE (integrated development environment) meaning they sometimes lack some of the features that this can provide. Though tighter integrations with Microsoft's featureful open source text editor [vscode](#) are changing this.

I am primarily an R user I've used jupyter notebooks for Python and [Raku](#) projects but much prefer the experience of Rmarkdown style notebooks over jupyter. Mostly as you do not generally see or edit the actual source document you only generate it from the interface. This makes working with version control tools like git more challenging. Thankfully [MyST](#) makes markdown style notebooks possible if you don't like Quarto.

There are trade-offs between the Rmarkdown & Jupyter Notebook ways of working (see: [The First Notebook War](#)) but [Quarto](#) and [jupyter book](#) in conjunction with [MyST](#) go a long way to resolving some of these issues.

If you are primarily a python person looking to get started with a literate programming workflow I would suggest that you avoid classic jupyter notebook files in favor of those [written entirely in Markdown](#).


You could use [Quarto with VSCode](#) and or [Python in RStudio](#) with Quarto and `{reticulate}` over [jupyterhub](#) or the [jupyter extension for vscode](#), but this may not be the best fit for you established workflow, it is a matter of taste.

One of the nice features of vscode for working collaboratively is the [live share extension](#) which gives you real-time google-docs-like collaboration tools, though of course you can still use [git in vscode](#) for asynchronous collaboration. JupyterHub now also has [support for real-time collaboration](#).

## 4.4 Jupyter/Quarto & vscode learning resources

- **Text/Video** (quickstart) [getting started with jupyter notebooks](#)
- **Text** (*quickstart*) [Quarto in vscode](#)
- **Video** (*longform*) [Jupyter Notebooks in VS Code Extension NEW in 2022 - Tutorial Introducing Kernels, Markdown, & Cells](#)
- **Text** (*documentation*) [jupyter notebooks in vscode](#)

### 4.4.0.1 Using git

 A short practical git glossary

I'm including command line examples here but the concepts should map well onto a number of different GUI front-ends to git. The glossary should help with both git's terminology and understanding some of the key concepts that make it up.

- **repository/repo** - a object in git, they are just a directory (a folder) with the right git configuration files in it.
  - `git init` initializes a new repo, you can also create one on a git hosting service like gitlab github then clone a local copy. After this you'll find a hidden `.git` folder in your repo.
- **commit** - A single point in the git history, like a save point. Commits have authors, short messages, optional longer descriptions, and a reference to the previous commit. When you have made changes in your repo you can pick those changes that make sense to group together under a single commit and add this state to the tracked history of your project by committing these changes.

- `git commit <file that I changed>`
  - `-m"Short informative message"`
  - commit all changes `git commit -am"Short informative message"`
- **add** - To include a file in tracking by git. Often used to add multiple files, if there are some items in your project that you explicitly want to be tracked by git you can specify these in a `.gitignore` file.
  - `git add new_file.txt`
  - `git add -A` add all files - *handle with care!* you don't want to accidentally commit a large binary file or things that should be kept secret.
  - *[advanced note]* an untracked file will remain in your working directory even if you change branches and by default will not be stashed if you run `git stash` to get a clean working tree to perform other git actions. They must be added or stashed with the `-u` option.
- **branch** - commits form a tree with each commit referencing the previous one. This tree can branch and allow two different commits to have the same parent. Thus a branch represents a particular lineage of development. The default name for the primary branch or 'trunk' of a repo is usually `master` or `main`.
  - list branches `git branch -l`
  - create a branch `git branch <new branch name>`
  - switch to branch `git checkout <new branch name>`
  - delete a branch `git branch -d <branch name>`
- **head** - The one referring to the commit at the tip of a branch, i.e. the latest commit made to that branch.
- **HEAD** - 'HEAD' is the 'head' of your current active branch. Unless you have a 'detached HEAD' in which case it references the arbitrary commit that

you have checked out (you can checkout commits not just branches if you want to see the state of the project at a point of a given commit). I find it helpful to think of HEAD as a window that I can drag over the tree of commits. Usually it slides along the tip of the branch I'm committing changes to. If I want to see another branch or point in history I can slide HEAD there and now the working directory will reflect the state of the project at that point in the tree.

- **checkout** - to look at the state of the repo on a given branch or at a given commit. 'Checking-out' the repo can be scary as it can look as though your work has disappeared from the repo. All *committed* states are kept in the `.git` directory, your working directory merely points to em here. By changing which commit your HEAD is pointing to you can move the window of your working directory around the history and branches of your project.
- **remote** - A repository tracking the same code base t somewhere else. Often used in the context of a git server from which you might pull and push changes. The default name of a repository's main remote is usually `origin`.

- list remotes and their urls `git remote -v`
  - add a new remote `git remote add <remote name> <remote url>`

- **pull** - To retrieve changes from a remote repo. If you are collaborating on a project you would generally pull any changes from remote before starting work so that you have the latest version of your colleagues changes.

- `git pull <branch name> <remote name>`, commonly `git pull origin master`

- **push** - To send the changes that you have made locally a remote. The commit at the tip of the remote branch must be an ancestor of the tree that you are trying to push to the remote. If it is not you will

have to perform a pull and resolve conflicts between your changes and those your colleagues.

- `git push <branch name> <remote name>`, commonly `git push origin master`

- **log** - The history of git commits. This shows the commit hash, message, author, author email, and date or previous commits.

- `git log` shows a list of commits adding the `--graph` flag shows a text based graphic of the branch structure

- **commit hash** - Every commit has an identifier associated with it this is a seemingly random string of letters and numbers. This is called the commit hash (because it is a [SHA-1](#) hash the commit). It is a function of the contents of the commit and an (approximately) globally unique identifier for it. **You can refer to a commit by a unique portion of its commit hash**, this is why you often see the first few characters of a commit used to reference it. *(Remember that a commit contains the hash of the previous commit making git a hash tree data structure from which you can infer a directed acyclic graph)*

- **staging area** - The git staging area houses the contents of proposed commits. It lets you decide which of the changes you have made you would like you to commit.

- The `git status` command reveals the contents of your staging area under **Changes to be committed**: if you change a file it will appear in this section. If you create a new file it will not be in the staging area until you add it with `git add`.

- The staging area can be manipulated interactively from the command line with `git add -i`

- **stash** - The stash is generally used to get your committed changes out of the way so that you have a clean

working directory. Many git operations are easier or only possible once you have a clean working directory. For example to do things like pull from upstream then re-applying your changes to the updated tree. This way you can update whilst avoiding committing or deleting unfinished changes.

- `git stash` stash your current changes
- `git stash list` list the stashes
- `git stash pop` apply the first item in the stash list to working directory. This is like rebasing in that it will apply your stash to the tip of the current branch.
- **merge** - Replaying the changes made on one branch on top those made on another in order to combine them. There are different approaches to combining branches than merging this may not always be the best action, we will discuss branching and merging strategies later.
  - If I am on the `master` branch and I want to merge in the `feature` branch I can use the command: `git merge feature` to merge the branches. This will, if there are no conflicts, create a merge commit.
  - Resolving merge conflicts can be fiddly and is best avoided. Try and get your branch into a state where it can be merged without conflicts before attempting merge. You can backout of a merge with `git merge --abort` If you weren't expecting a conflict, do so before making any attempts at resolving the conflict as if you make changes during a merge you may not be able to revert cleanly
- **rebase** - Reapply commits from your branch from the tip another. If your branch has diverged from `master` but has no conflicts with it's current tip then instead merging you can **rebase on master**. 'snip' your branch off from it's current parent and auto-

matically generate new commits where the current tip of `master` is the parent instead.

- Rebase your current branch on master `git rebase master`.
  - Rebase a specific branch on master `git rebase master <branch name>`
  - You may also encounter the concept of a ‘fast-forward merge’ this is equivalent to rebasing
- **diff** - Get the difference between file(s) at different rates. This might be between two commits, various special cases of this. A common diff is between the staging area and unstaged changes. (If you use git on the CLI checkout `delta` for improved visual diffs)
  - **status** - What is the condition of the working tree? Which changes have been made since the last commit? Which of the changes has or has not been staged yet.
  - **mv / rm** - using the base UNIX commands rather than the git versions may not have the desired effect. If you `mv` a file to re-name it it will appear to git as though you deleted add re-added it less you use `git mv`. If you `rm` a file you will counter-intuitively need to add the action of removing it to your staging area to let git know you’ve removed it is simpler to `git rm` a file which both removes it and stages the action of removing it.
  - **submodule** - *gitception: a git repo inside another git repo!*
  - **Pull Request (PR)** - A request that the stewards of the upstream project pull changes from your tree into theirs.

#### Getting started with git on the command line

*unfinished!*

Installing git...links

**Tell git who you are**

```
git config --global user.email
```



```
"youremail@yourdomain.com"  git config --global
user.name "your name"
(Dropping --global will only set these values for the cur-
rent project)
Initialize a git repository (turn a folder into a git repo)
git init
Add files to be tracked by git:
echo "# README" > README.md # an example file
git add README.md
set-up a remote
(Send your changes to a git server)
pull from a remote (Get the latest changes from a git
server)
Get the status of your git repository with git status this
will show you ...
diff
staging chunks
```

#### 4.4.0.1.1 git hosting & UIs

There are **numerous GUI** (e.g. [gitkraken](#)) / **TUI** (terminal/text user interface) (e.g. [gitui](#), [lazygit](#)) interfaces to **git** which provide convenient interfaces to git beyond the core command line application. **RStudio provides a built-in git UI** in which you can commit changes, see diffs, explore history, manage branches etc. By default it is located in a tab in the top right pane of the RStudio interface in projects which use git.

Git and the platforms built around it such as [github](#) and [gitlab](#) solve the problem of sharing and collaborating on your code, and in the context of literate programming your prose as well.

You can also explore the history of the changes made to project in the history view of project in github or gitlab for example here is the [gitlab history of this document](#).

**Another useful feature of git is for attribution. Every git commit has an author** so when collaborating on a project managed in git credit can go to the people who wrote

particular parts of the document. (git also distinguishes between an author and a committer, so a committer can commit changes from an author who is not themselves directly using git if desired, though this is not entirely the intended use case)

#### 💡 Setting git commit Author & Committer separately

You can temporarily override the default author / committer values set in the global or local git config files by setting these

```
export GIT_AUTHOR_NAME="John Smith"
export GIT_AUTHOR_EMAIL="jsmith@example.com"
export GIT_COMMITTER_NAME="Jane Doe"
export GIT_COMMITTER_EMAIL="jdoe@example.com"
```



Note that truly **deleting things from a git history** once that history has been pushed to a repo used by others **can be quite difficult**. (It can take a long time because git is based on hash tree if you delete something from the history you have to re-write all subsequent commits. This is part of what makes it such a good system for provenance of code.) So **never commit secrets such as passwords or API keys even to private repos** if these might ever be made public. Storing sensitive values in environment variables is a common solution to this problem.

#### 4.4.0.1.2 git branching strategies for collaborative document editing

*feature branches for internal collaborators, forking for external collaborators.*

There are a variety of workflow patterns which can be followed when it comes to collaborating on git projects. For a solo project you might be able to get away with just committing directly to the primary branch (often called main or master) almost all of the time. When collaborating however it can be a good idea to switch to working in 'feature branches'. You

have some small feature that you want to implement or issue to address so you make a branch and work on it there. Once you are done you can check on the status of the master branch. If master is ahead of where you branched off you might want to rebase on the new master, resolve and conflicts and perform a fast forward merge appending your new commits to the end of the master branch. It is best to keep the scope of these as small as possible so there are minimal issues when merging back into the master branch.

A collaborator with access and permissions on your repository can work on feature branches in your repo, but an external collaborator without these permissions cannot. So to achieve the same thing they can fork the repo i.e. make their own copy and submit pull requests (PR) from there. PRs are best for specific suggested changes. If there is a problem or query around what changes need to be made an issue should be opened in the issue tracker of the project and once plans for specific changes are agreed then a PR can be generated with the proposed changes. PRs can be reviewed and revised before being accepted and merged into the master branch.

This process is generally how ‘peer review’ usually just referred to as ‘code review’ tends to happen in software projects. An issue becomes a proposed set of changes, becomes a specific implementation of those changes, becomes a pull request. Any alterations to the specifics are worked out in the PR before the agreed changes are merged.

**In literate programming it is advisable to follow the convention of one sentence per line in the source document when using git.** This makes it easier to manage git diffs as git focuses on linewise not character-wise differences. You can get this behavior in RStudio with the options below in the YAML header of an Rmarkdown document, or at of project of global level in the RStudio settings.

```
---
editor_options:
  markdown:
    wrap: sentence
    canonical: true
```

---

## 4.5 Git learning resources

- **Video** (*quickstart*) [Collaborating on R code in Rstudio with Git - quick demo](#)
- **Video** (*quickstart*) [Easy ways to go back in your git commit history with RStudio](#)
- **Video** (*quickstart*) [Learn Git In 15 Minutes](#)
- **Video** (*longform*) [Git Tutorial for Beginners: Learn Git in 1 Hour](#)
- **Text** (*longform*) [Pro Git book](#)
- **Text** (*documentation*) [Git Docs](#)
- **Video** (*longform*) [Deep-dive into git internals to deepen your understanding Git Internals by John Britton](#)
- **Visual** If you are visual thinker who appreciates a good explanatory diagram checkout this [visual guide to git](#)
- **Visual** (*interactive*) [Visualizing git concepts with D3](#)
- **Game** If you are seeking to grok git a bit more deeply and have some fun doing it then I recommend the [oh my git!](#) game
- git troubleshooting and common problems / mistakes
  - **Video** [Fixing Common Mistakes and Undoing Bad Commits](#)
  - **Video** [How to Undo Mistakes With Git Using the Command Line](#)
- Alternative git interfaces
  - [gitless \(gl\)](#)
  - [design theory / critique](#)
  - [Jujutsu VCS \(jj\)](#)
  - [legit](#)
  - [sapling](#) - not yet fully available worth watching

### 4.5.1 Environment management

When you write data analysis code in a language like R or Python chances are that you are going to be depending on some

other packages to do your work. You may have noticed that **updates to the packages sometimes break your code**. A function that used to exist has been deprecated and is no longer in a package, or the arguments to a function have changed. More worrying still **sometimes such changes won't stop your code running but produce an output that is wrong but not in an obvious fashion**. Thus in order to reproduce your analysis exactly we would need **not just your code but the versions of the language and the packages that your code depends on**. This way it is possible run your code with the confidence that it is functioning the same way for us as it was for you.

#### 4.5.1.1 Package & Environment management tools

In the R programming language the best package management solution for reproducible environments is `{renv}`.

`{renv}` provides `renv::install()` which is a replacement for the base `install.packages()`, as well as the `BiocManger::install()` & `remotes::install_github()` functions used to install R packages. The `renv::snapshot()` function is used to create a project specific manifest file `renv.lock` which documents all the packages used and their versions. `renv::restore()` can then be used to make the installed packages and their versions match those specified in the lock file. `{renv}` has a central package cache and uses symbolic links to project libraries to ensure that there is only one copy of a given version of a package installed on your system improving it's performance over previous attempts at project specific package management in R.

In Python there are two main tools for managing package environments.

`venv` is a python specific environment manager for isolated project specific python package management and part of the python standard libraries. **A virtual environment can be created with by running `python3 -m venv venv/` in your project directory**. This command uses the `venv`

module (`-m venv`) to create a virtual environment called `venv` (`venv/`) but the name is arbitrary and a sub-directory with the environment's name will be created. **To use the environment it must be activated with: `source venv/bin/activate`, `deactivate` exits the environment.** The `pip` package manager can be used as normal within the environment and will only affect the local environment while it is active. **To capture a snapshot of the environment from which it could be restored later use `pip freeze > requirements.txt`** A virtual environment can be restored from a `requirements.txt` file with: `pip install -r requirements.txt` This [guide to python virtual environments](#) goes into some additional details of how to use `venv` and how it works. For management of the version of python itself `pyenv` is a good tool. For the management of python packaging the tool `poetry` is a good choice for it's good dependency management.

`conda` is both a package and environment manager and is **language agnostic** however tends to be used in predominantly python settings. Whilst `conda` can be used to manage R packages I would not recommend it for a predominantly R project. **By default `conda` does not take the approach of storing the specification of your environment within the project directory**, unlike `{renv}` & `venv`, I would avoid this default behavior. **Keeping the environment specification in the directory is obviously preferable if you want to be able to share the project along with it's environment.** This [guide to conda projects](#) provides a nice overview of getting started with `conda` environments using an `environment.yml` file and this demonstrates [how you can set the location of the conda environments to be within a project directory](#).

Beyond **language specific packages** many of the packages in a given language will **depend on system libraries in your operating system**. For instance **an R package which parses XML files might rely on a fast system library written in C which is used by many other packages** in other languages rather than re-implement it's own XML parsing package and duplicate the effort. **Now a language specific package or environment management solution will no longer be sufficient alone.** Solutions to this problem in-

clude more advanced system level package management tools such as [NIX](#) based package management (see also [GNU GUIX](#)). Alternatively, and more popularly at the moment, system dependencies can be managed by the operating system's package manager and **containers** can be used to create portable and isolated system environments with different system dependencies. Nix-like package management solves more and different problems than containers and can be used to more reproducibly build container images but still currently has a bit of a 'early adopter tax'.

#### 4.5.1.2 Containers

**A container provides an isolated self-contained computing environment** similar in practice to that of a virtual machine (VM) whilst not having nearly the same performance deficits associated with virtualization (for the technically inclined, a simplification is containers share a kernel but provide a different user-land). This lets you **package up your code along with all it's dependencies and configuration in a standard 'box' that can run exactly the same way on** essentially **any Linux back-end** (as well as on mac and windows through what amounts to a wrapper around a linux VM).

The most popular containerization technology is [Docker](#) though others exist ([podman](#) & [Apptainer/Singularity](#) for example). You specify the environment you want inside a Docker container using a **Dockerfile** and building a container image which can run things in the environment specified in the **Dockerfile**. Whilst **running something with the exact container image is fully reproducible building a container image from a specification is not necessarily so**. The **Dockerfile** starts from a 'base image' usually of the operating system you'd like to setup your environment in. You might use for example `ubuntu:latest` the second part of this text specifying the operating system `latest` is called a tag. **The latest tag obviously depends on what happened to be the latest version when the build command was run** thus you cannot rebuild an identical image to the original one built from this **Dockerfile** unless you know what version

of Ubuntu was the latest when the build command was run. **To avoid this ambiguity it is best to specify the version more explicitly e.g. `ubuntu:jammy`**, `jammy` is the code name for Ubuntu 22.04 the current (as of writing) LTS (long term support) release of the Ubuntu operating system.

## 4.5.2 Dataset Management

When datasets are small their day-to-day management is often relatively un-complicated. We can just make a copy of our original raw data and work with that in our analysis. As datasets get larger and simply making a copy of them becomes an expensive operation we often have to get a bit more creative with their management.

### 4.5.2.1 Your Own Raw Data

**Raw data is conceptually ‘read-only’ it can be a good idea to make this literal.** Keep the raw data for your project in a place that you cannot accidentally modify or delete it. An easy way to do this is to make your raw data files read only and keep them in specific location which you can back-up with a little extra thoroughness.



On UNIX like systems you might want to follow a pattern like this:

```
# A central directory to store all your raw data files
# with subdirectories by project
mkdir -p ~/tank/test-project-data

# Make an example data file
touch ~/tank/test-project-data/data.file

# Change the mode of all the files in `test-project-data`
# and all sub-directories with `-R`
# remove the write permission with `-w`
```



```
chmod -R -w ~/tank/test-project-data

# Make a directory in your project folder to link to your raw data
mkdir -p ~/projects/test-project/data
ln -s ~/tank/test-project-data/data.file ~/projects/data/data.file

# Links to files in ~/projects/data/data.file can now be deleted
# The files they are linked to will not be affected
rm ~/projects/data/data.file

# If you run this you'll find it's still there
ls -l ~/tank/test-project-data/data.file
```

Need Help understanding any of these shell commands?  
Checkout [explainshell.com](http://explainshell.com) paste in any shell command to  
get a breakdown of its component parts and what they  
mean.

**This approach lets you keep your datasets within your project directory without actually having to keep the files there.** For example you might have your data directory on a secondary higher capacity storage device than your projects folder.

You may even want to make a dedicated user account who is the only one with write permissions to your raw data files as extra protection against their accidentally being changed. A dedicated account able only to read the raw data files that is used to perform backups is also a potentially sensible strategy.

#### 4.5.2.2 Raw Vs. Processed Data

**Raw data is generally data directly from whatever your instrument is. There may be some degree of pre-processing applied** by that instrument on its own raw data from its sensors prior to outputting this pre-processed data to the end user. For example in DNA sequencing machines base calls are generally made on the machine from the raw sensor output e.g. the fluorescence intensity before being output as a fastq file with the call and an indicator of its quality.

Once you have your raw data you process it yielding (surprise!) processed data. Some of this processed data will be ‘end points’ and other parts may be ‘intermediate data products’. Whether your data is an endpoint or an intermediate product is context dependent. You might for example consider the count matrix from a RNA-seq experiment as an endpoint as it is a common product of analysis used in further downstream analyses. It’s the sort of data product that it is useful to others if you include it when you deposit your data in a public repository. But you might discard the alignments in the form of BAM files as these are very large. BAM files are however computationally expensive to generate so you might keep them around for the active duration of the project but not archive them.

**In theory all processed data should be dispensable if your raw data, analysis code, and computational environment are properly documented. It should be possible to exactly regenerate your results from your raw data and your computational methods.**

#### 4.5.2.3 Public Data

When working with data that you did not generate and thus do not need to ensure the preservation of. You might want to keep this somewhere separate from your own raw data. Somewhere without your own backups where you can cache the data. Always be sure to capture the metadata about how you acquired your copy though, accession numbers, when you did so, and any version numbers available.

Domain specific data repositories may have their own download tools and approaches to locally caching data which you can use.

As we will discuss in section [Chapter 5 When to Publish Data](#) **it can be a good idea to publish your own data to a public repository before publishing your main analysis.** This way you can access your data from the public resource as other researchers would. This is a good practice as it **permits you to validate if your data are indeed FAIR.** It shows that you were able to find you own data, refer to it with it’s

unique identifiers and retrieve it in a appropriate format. This improves the documentation of the provenance of your data as it's shared accessions and metadata annotation are used in the original work leaving less opportunity for errors of labeling etc. in the data repository.

#### 4.5.2.4 Large files in git repos

Within a project managed by `git` large binary files, such as images, can be a problem as they will cause a repo to quickly grow to an unmanageable size if they are included by `git`. A solution to this problem if you want to remain within the git paradigm is `git-lfs` (git large file storage) though this approach is not without it's drawbacks. Every committed version of a large file is still kept just on the `git-lfs` server not in everyone's local repos where only the needed version in synchronized. When `git-lfs` is available at a git hosting service it is often, understandably, a paid feature or has limited capacity. It is also a non-trivial effort to configure and host your own `git-lfs` server.

The tool [data version control \(DVC\)](#) provides git compatible `git-lfs` like functionality with different storage back-end options including consumer cloud storage options like google drive drop-box etc. Other alternatives include [lakefs](#), and you can also use ZFS to version data if you are using it directly and not just as a storage back-end behind other abstractions.

## 4.6 git-lfs learning resources

- [git-lfs](#) Documentation
- **Video** (*quickstart*) for those already quite familiar with git this [Introduction to Git LFS](#) is a good starting point
- **Video** (*deepdive*) [Managing huge files on the right storage with Git LFS](#)
- **Text** (*quickstart*) [git-lfs tutorial](#)
- **Text** (*longform*) [git-lfs tutorial](#)

#### 4.6.0.1 Imaging datasets

*Note: I am not an image analysis specialist, so this section would likely benefit from the contributions of someone with more experience in this area.*

Working with imaging data can pose a number of substantial practical challenges. Imaging datasets are often quite complex to administer they are often comprised of many files which need to be structured and accompanied by both experimental design and technical metadata. Many microscopes produce images in proprietary formats which attempt to address some of these organisational issues by bundling together metadata and images from individual planes or channels into single files that are interpretable to their software, unfortunately these formats are often proprietary which can present issues when trying to use them with software other than that provided by the manufacture of your imaging equipment or their partners.

Thanks to the open microscopy environment's (OME) [bio-formats](#) project and its hard work reverse engineering many of these formats it is now possible to work with many of them interoperably and with open software tools. There are ongoing efforts to have commercial imaging providers make use of open standards and open up their imaging formats<sup>5</sup>.

As alluded to in Chapter 3 [How To Store Your Data](#) imaging data can in some cases be very large. 3 dimensional multi-channel and time course (aka 5D) datasets at high resolution from imaging techniques such as light sheet microscopy can rapidly balloon in size. Extremely high resolution electron microscopy images are another example. When datasets reach multiple terabytes we start running up against the limits of the current generation of readily available computing technology to make use of datasets of this unwieldy size. Fortunately there is much work underway to make this a more manageable

---

<sup>5</sup>Dear Funders, please get together and adopt a blanket policy of refusing to fund the purchase of any scientific equipment which outputs data in a proprietary format, ideally eventually moving on to refusing to fund the purchase of any equipment with proprietary embedded software. This would really save everyone a lot of time and money in the long run.  
*Pretty Please*

problem including the development of next generation file formats [OME-NGFF](#) which facilitate parallel processing and the streaming of only needed portions of large datasets to users remotely accessing data from central repository(s) ([Moore et al. 2021](#) [cito:citesAsAuthority] [cito:credits] [cito:agreesWith]).

**Organizing your imaging data benefits from software tools which permit you to store, view, annotate, share, search, and programatically explore your imaging datasets.** A simple file and folder directory structure with some standard operating procedures for where to put files and what to call them is slow, manual, cumbersome, and error prone. **The Open Microscopy Environment’s OMERO tool is probably the best available software tool to solve your image data organisation woes.** It operates a standard client-server approach with a central server on which the data is indexed and stored which can be accessed by various clients. There is a general web client, and additional web based viewers and figure creation tools as well as a desktop client to speed up larger image uploads & downloads. The OMERO server can be accessed via an application programming interface (API) which permits you to interact with your data from applications like [Fiji](#), [cellprofiler](#), [QuPath](#), or [napari](#). There are libraries for [Python](#), [R](#), [Java](#), & [MATLAB](#) to facilitate using the API in your own custom analysis code.

One of the advantages to deploying an OMERO instance and using it to store and analyse your data is that it is the same software stack which underpins public image databases such as the [Image Data Repository \(IDR\)](#) a highly curated ‘added-value database’ for image data-sets that are community resources. You can interact with your own data in the same way you interact with publicly available datasets and when you make your own data public others can access it the same way you do internally

#### Image Data Learning Resources

- [Fiji / ImageJ](#)
  - **Text** (*documentation*) [Fiji/ImageJ documen-](#)

tation

- **Text** (*tutorial*) [ImageJ tutorials](#)

- **OMERO**

- **Text** (*documentation*) [Open Microscopy Environment \(OME\) documentation](#)
- **Text** (*tutorial*) [OMERO guides](#)
- **Video** (*longform*) [Image data: management, sharing & re-use - IDR workshop](#)

- **CellProfiler**

- **Text** (*documentation*) [Cellprofiler manual](#)
- **Text** (*tutorial*) [Cellprofiler tutorials](#)

- **QuPath**

- **Text** (*documentation*) [QuPath documentation](#)
- **Text** (*publication*) [QuPath: Open source software for digital pathology image analysis \(Bankhead et al. 2017 \[cito:citesAsAuthority\]\)](#)
- **Video** (*YouTube Channel*) [QuPath YouTube channel](#)
- **Video** (*quickstart*) [QuPath OMERO access](#)
- **Video** (*longform*) [QuPath tutorial #1 - Getting started](#)

- **Human Help** (*community forum*) [image.sc](#) is a **discourse** forum for support and advice on image analysis and image data management.

## 4.6.1 Pipeline/Workflow Management

### 4.6.1.1 Pipeline management tools

#### 4.6.1.1.1 Why pipeline/workflow management tools?

One of the significant practical issues addressed by using a pipeline management system for developing a new analysis or

iterating on an existing one is results caching. If your long fairly complex pipeline with some slow computationally expensive steps is just a script that has to be re-run from scratch because you changed how a graph looks you're not going to use a single framework for your whole analysis. You are, quite sensibly, going to break it up into separate steps. You are however now at risk of ending up with your analysis in an inconsistent state if, for example, you forget to re-run a step downstream of your change. You have introduced a semi-manual stepping through each of the separate sections of your analysis to get the final result.

Many pipeline managers are designed to be (mostly) idempotent, that is to say running the same pipeline repeatedly will get you the same result, subsequent runs will not be affected by previous runs. Also running it repeatedly is a save operation in that it won't affect the outcome. Whilst you can manage to get the same result with an ordinary script it can be very cumbersome and time consuming to do so. One of the tricks generally employed by pipeline managers to make idempotency practical is caching. If you can cache the computationally expensive parts of an analysis you can feel safe running the pipeline command again. This way you can make a minor downstream modification to a plot safe in the knowledge that you won't have to wait hours to see the results of your change, as the same pipeline command only runs steps that need to be run to apply the changes.

This lets you keep your long and complex analyses properly connected together and re-run-able from scratch with a single command but means that you don't have to re-run the bits you have not changed to ensure everything stays consistent. Despite this it is almost always advisable to re-run any pipeline from scratch with a clean cache once you think you have the final version ready to make sure there are no hitches. Cache invalidation is after all a legendarily hard thing to get consistently right. Because pipeline managers generally understand the dependency relationships between the steps of your analysis it is usually simple to automatically parallelise independent tasks and get better run times.

Pipeline management tools are most advantageous for longer

more complex or more computationally expensive analyses, especially those intended to be reused by others. Their design tends to favor workloads which require large batch processing with little to no user interaction needed during a run. So they won't be applicable for everyone's use-case.

#### 4.6.1.1.2 Which pipeline tool?

There are a number of language specific pipeline tools which may be easier to learn if you are already proficient with a particular language and benefit from language specific integrations. R's `{targets}` pipeline manager for instance has nice integrations with R's literate programming tools, which can be useful when writing a pipeline with nicely formatted outputs. In Python there is the `snakemake` pipeline manager.

A common reason for using a pipeline manager however is not writing a new pipeline from scratch but making use of an existing one. A good example of this is the `nf-core` project which uses `nextflow` a domain specific language (DSL) for pipeline management which excels in portability of pipelines between different systems. `nf-core` has a number of pre-built pipelines for common bioinformatic analyses which can be used by anyone and make it easy for others to reproduce your analysis. `nf-core` is an open source project so anyone can contribute updates, extensions, bug fixes or entirely new pipelines to the project which may be incorporated into the upstream versions used by the community. If you have a novel analysis method creating such a community pipeline is one of the best ways to make it easy for other researchers to use your work. (Publications which accompany tools which become popular tend to attract out-sized numbers of citations.)

A project that is worth being aware of in the workflow management space is the `common workflow language (CWL)`. Many other pipeline management frameworks have a least partial compatibility to import/export CWL which is very valuable when migrating a pipeline between systems. If you are working with a pipeline management tool other than `nextflow` or even if you are using `nextflow` you can also deposit workflows in `WorkflowHub` which supports workflows of any type.



The [Renku](#) platform also has a built-in workflow management system which takes a slightly different approach to constructing pipelines step-wise which can be exported to CWL and which we will cover more in the [renku section](#) Section 4.8.

## 4.7 Pipeline tools learning resources

- Nextflow
  - **Text** (*documentation*) [Nextflow Docs](#)
  - **Video** (*longform*) [Nextflow Training Workshop \(2022\)](#)
- {targets} (R)
  - **Text** (*documentetation*) [The {targets} user manual](#)
  - **Video** (*longform*) [Reproducible Computation at Scale in R with {targets}](#)
- snakemake (Python)
  - **Text** (*documentation*) [Snakemake Docs](#)
  - **Video** (*quickstart*) [An introduction to Snakemake - tutorial for beginners](#)

### 4.7.0.0.1 Continuous integration & deployment (CI/CD)

CI/CD are concepts popularized by the software development industry for testing and deploying applications. If you are developing a software package to share your code learning some of this tooling can be very useful to automate many of the steps involved in testing and distributing software. However this same tooling can be very useful for checking that you analyses are indeed reproducible and for publishing documentation associated with any workflows that you share.

This text is making use of a CI/CD pipeline in its publication process. It is built from its markdown source files into a website, epub & pdf on a gitlab CI/CD ‘runner’ every time I push commits to the remote repository. The static website for this document is only updated if the build completes correctly with

no errors. [This .gitlab-ci.yml file](#) details the steps taken when building this document from source. (I'm doing some extra things so it would not normally be as complicated as it might appear in this file to build a document like this.)

- CI/CD for R package development
  - `{usethis}` provides functions to automate R package testing with `{testthat}` using [github actions](#)

## 4.8 Renku - Bringing it all together

### *Keeping your code, data, & compute environment together*

In the last few sections we covered a number of powerful, complex and configurable technologies, it may feel a bit **overwhelming** as there is a lot to learn and a lot of choices to make. **Fortunately, the Renku platform combines many of these technologies and has chosen some sensible defaults to make it simpler to get started using them.** It is about as easy as picking up a project that is a jupyter notebook or RStudio project if you are already familiar with these and getting much of the rest (almost) for free. Renku's flexible template system makes it possible for people with more experience of the platform to set up easy to use environments specialized for particular tasks for other collaborators on a project.

**Renku** ( “linked verses”), is a Japanese form of popular collaborative linked verse poetry, written by more than one author working together.

- Wikipedia

### 4.8.1 Why Renku?

Whilst there are other solutions to the reproducible compute problem which make it fairly straightforward to reproduce environments, e.g. [binder](#) they lack data integrations. There are proprietary cloud based solutions to having the trifecta of data, code and compute environment in the same place such as [Google's Colab](#). However, given [google's graveyard full of dead projects](#) it may be unwise to depend on them if you want your work to be around and accessible in the medium to long term future. There is also [Code Ocean](#) & [DagsHub](#) but these are also a paid closed source solutions even if many of their internals

and integrations are based on open source tooling. [Stencila](#) is an ambitious but still early stage open source project notably they have an integration with [eLife](#).

Fundamentally adoption of a proprietary platform as a standard for computational reproducibility is an oxymoron, full transparency is not possible with this approach. I can't verifiably reproduce your analysis if I'm using a black box to do it or am missing key features to create new analyses or interact with their results. Paid services in support of open source tools is the only transparent, ethical and sustainable approach to solving this problem. A project worth watching as a source of publicly funded could infrastructure for hosting such open open platforms is the European [Open Science Cloud \(EOSC\)](#), though this remains at a relatively early stage of development at time of writing.

Renku provides all the needed features of the above projects but is an open-source project developed at the Swiss Data Science Center based at EPFL and ETH Zurich. A project which you can host yourself and which has a public instance at [renkulab.io](#). Similar considerations apply to the choice of a reproducible computational analysis platform as to [the choice of electronic lab notebooks](#) (section Section 4.1) this is because they have semi-overlapping functions. A platform like Renku can serve as the lab notebook to your more computationally focused researchers.

## 4.8.2 Getting started with Renku

### 4.8.2.1 Account setup

You can signup for an account at [Renkulab.io](#) at this [registration page](#) ORCID & github are supported as single single sign-on providers.

### 4.8.2.2 Your first project

### 4.8.2.3 Templates

When you start a new project in renku you generally do so from a template. [Renku has a templating system](#) which permits users to create their own templates for projects. There are a [core set of default templates](#) as well as [community contributed ones](#). I'm also developing some [templates for HDBI](#).

### 4.8.2.4 Running Renku Sessions Locally

If you have [docker](#) and the [renku CLI client](#) installed on your system you can run an interactive renku session on your local system by running: `renku session start` and navigating to the link that it returns in your web browser the link will look something like this: `http://0.0.0.0:49153/?token=998dasdf...`

#### Remote access to a local session

If you are running a renku session on a local workstation or server with a lot of compute resources but still want to access this session remotely from your laptop or even phone there are a couple of ways of doing this.

If you can ssh (secure shell) into the machine running your container you can access your session by ssh port forwarding using a command structured like the following:

```
ssh -nNT -L <local port>:<host>:<remote port>
<host>. let's say renku session start on my workstation returns: http://0.0.0.0:49153/?token=998dasdf
I can run ssh -nNT -L 49153:me@host:49153 host on my laptop where me is my username on my workstation and host is my workstation's IP/url then I can navigate to http://0.0.0.0:49153/?token=998dasdf on my laptop and remotely access the session.
```

[tailscale](#) will create a secure wireguard mesh VPN between the clients on which it is installed so if you can install tailscale on your workstation and laptop

you can connect to your workstation irrespective of any firewalls or NAT normally blocking your path. Simply navigate to your workstation's tailscale IP address and enter the port/token for the session. e.g. `http://100.10.10.10:49153/?token=998dasdf` where 100.10.10.10 is your workstation's IP from tailscale status

### 4.8.3 Renku Learning Resources

- **Text** (*documentation*) [Renku Docs](#)
- **Human Help** (*forum*) [Renku Forum](#)
- **Video** (...) [Renku YouTube Channel](#)

## 5 When To Publish Data

There are varying norms in the academic community about when in the research cycle to publish your data. In the biological sciences today the most prevalent practice in my experience is data release at time of publication. At least for individual research projects. For large consortia producing sequencing data it is more common for data to be released shortly after it is generated as laid out in [The Fort Lauderdale Principles](#) which we will be covering and critiquing in section Section 5.1.

There are a number of good reasons to publish data (*almost*) as soon as it is generated instead of waiting until time of publication that we will cover below.

We will be addressing considerations of privacy & consent in Chapter 7 [What Data to Publish](#).

### 5.1 The Fort Lauderdale Principles

The Fort Lauderdale principles arose from a meeting of ~40 people organised by the Wellcome trust to discuss **pre-publication release of data in the field of genomics**. These principles have become the **basis for many data release policies in consortia in the biological sciences** since. [The Fort Lauderdale Report](#), a 4 page summary of the conclusions of this meeting, was **produced approximately 20 years ago at time of writing in January of 2003**.

The report lays out the responsibilities of 3 groups: **Funders, Producers & Users of data**.

Here is a summary of these responsibilities:

- **Funders** are tasked with ensuring that:
  - Project descriptions are published

- Requiring that data be publicly released as a condition of funding
  - Encouraging participation in community resource projects
  - Providing a centralized index of community resource projects
  - Providing centralized repositories for the data
- **Resource producers** with:
    - Publishing a project description
    - Producing data of consistently high quality
    - Making the data immediately available without restrictions
    - Recognize that there may be violations of the norm of not publishing ‘global’ analyses prior to those generated by the data generators
- **Resource users** with:
    - Citing the project description
    - Not publishing ‘global’ analyses of the data before its producers do
    - Ensuring that these norms are adhered to by the community

These guidelines contain **mostly very sound recommendations and have largely served the community very well, indeed data sharing in genomics is ahead of many other domains, imaging for example is only now beginning to catch up.** There are, however, some distinctly problematic aspects of these recommendations, and there is much room to continue to improve our best practices.

Specifically the section on the recommendations to ‘resource users’ contains language which could not too uncharitably be interpreted as advising ‘resource users’ to punish those who violate the norm of allowing ‘resource produces’ to be the first to publish ‘global’ analyses of the datasets that they generated. Implying means such as holding up review of publications and/or grants to do so. This is of course totally unacceptable.

One of the issues with this approach to the enforcement of these norms is that more senior researchers can more readily enforce

them on Junior researchers than *vice versa*. Senior researchers are more likely to sit on grant awarding bodies and be requested as reviewers on papers giving them more access to these tools. Junior researchers are hesitant to release data prior to publication for fear that larger and more well resourced groups may be able to analyse their data and shepherd a manuscript through the publication pipeline before they are able to do so. This potentially reduces the ‘novelty’ of their findings as prior work has now been published with their datasets. This need not, and likely rarely is, malicious on the part of other researchers who have simply identified suitable datasets to use in their analyses. Even if in reality this is an infrequent occurrence, concern over the possibility suppresses researchers willingness to share data.

Since the Fort Lauderdale report was authored changes in the technology have meant that many more researchers are producing datasets at a scale which could reasonably be considered a ‘resource’. Whole international consortia in 2003 may have produced less data than some individual PhD projects in 2023. We are almost all ‘resource producers’ now, thus to preserve the practice of pre-publication release of data resources, and the benefits this has for the FAIRness of data and the pace of scientific investigations, we must fix the incentives issues around individual pre-publication data sharing.

I would suggest that **20 years on these principles are in need of a bit of an update** to reflect a more constructive attitude to the use of public data. The updated advice that I would recommend is to follow a different workflow which **preserves the publication precedence of the ‘resource producers’ without restricting general access to data shortly after its generation**. This approach is called **registered reports**, and they address more problems than just being hesitant to release your data prior to publication for fear of being ‘scooped’<sup>1</sup>.

---

<sup>1</sup>‘Scooped’ is a term we should excise from the academic lexicon, we are scientists not journalists we are ‘corroborated’ not ‘scooped’.



## 5.2 Registered Reports

The [centre for open science \(COS\)](#) describes [Registered Reports](#) as “Peer review before results are known to align scientific values and practices”. There are over 300 [participating journals](#) where this format is accepted (including for example [Nature](#) (“[Nature Welcomes Registered Reports](#)” 2023 [cito:citesAsAuthority]), [Nature Communications](#), [PLOS Biology](#) & [BMC Biology](#)). If a Journal that you would like to publish with does not yet support registered reports I would suggest writing to an editor there and requesting that they consider doing so, maybe get together with some colleagues and write a joint letter.

The workflow for a registered report differs somewhat from the conventional publication process. You effectively write a version of your introduction and methods sections prior to performing your main experimental work, submit this for review and secure an agreement in principle to publish your results irrespective of the outcome of your experiment.

To be clear this does not limit exploratory analysis just makes clear what is planned and what is post-hoc. This Prevents **HARKING** (**H**ypothesizing **A**fter the **R**esults are **K**now)



Figure 5.1: Registered Reports Flow - centre for open science

One of the advantages to in-principle acceptance at stage 1 review is that it allows researchers to list publications much sooner than they could for a conventional manuscript, especially useful for early career researchers in a context where conventional publication can drag out over years.

For a computational analysis the study design phase would ideally include writing your analysis code using mock or example data and submitting running code as a part of the review,

as was discussed in section Chapter 2 when to generate data. Once your data has been generated you are then ready to perform your planned analysis almost immediately using reviewed and tested code. Data would ideally be placed into a public repository **before** you run your main analysis prior to the 2nd review stage. Your data would also ideally be accessed from the repository by your analysis pipeline using its globally unique identifier(s). This serves to test the *accessible* and *interoperable* components of the FAIRness of your dataset. **You don't know if data is FAIR until you actually try and (re-)use it**, if its first use follows the same pattern as anyone who might subsequently want to re-use it we have a clear demonstration of how the data can be re-used.

Protocols you plan to use in your analysis perhaps developed and refined whilst generating some preliminary data can be shared on a protocol sharing platform such as [protocols.io](#) or via a tool like the [OSF](#). These can be cited in your registered report, and because such platforms permit versioning if any further refinements are made in the course of the main experiment, can be updated and the new version cited in the final manuscript permitting people to see the revisions.

Whilst not applicable in every context the approach taken by registered reports needn't apply only to the more narrowly hypothesis driven work, they can apply in contexts where more exploratory questions are being asked. For example if you are carrying out a screen like question looking for genes or pathways that differ between some experimental conditions it is still possible to lay out in advance your analysis plan and criteria for deeming a change of sufficient magnitude or significance. Though the acceptability of this approach may vary depending on the criteria for registration at specific publication venues.

This workflow puts thinking about the analysis and experimental design strictly before data collection. This helps to address common issues faced by data analysts with being consulted too late in the process to address issues in the experimental design, about which I opined in Chapter 2 [When To Generate Data](#).

If you retroactively discover unanticipated problems with your design and think that different analytic methods than you planned may yield results that more accurately reflect the

“In a world where research can now circulate rapidly on the Internet, we need to develop new ways to do science in public.”

[Saloni Dattani](#)

phenomenon that you are working on, then present these alongside your planned analyses along with your reasoning about the improvements. Your entire analysis need not be constrained by your pre-registered plan. The plan merely serves as an accountability mechanism and as an exercise to ensure that you have thought deeply enough about how you are going to analyse your results before you embark on generating them. This workflow provides a much more robust mechanism against unintentionally performing experiments that would be obviously bad, unnecessarily expensive and/or wasteful with the benefit of hindsight, something that occurs more often than we'd probably like to admit.

If you are considering doing a registered report I suggest that you refer to the [section on the center for open science's website](#) and this [practical guide to navigating registered reports](#) from [Anastasia Kiyonaga, Jason M. Scimeca](#) that is summarized in the figure below.



Figure 5.2: Practical considerations for navigating Registered Reports

## 5.3 Counterpoints to publication on data generation

### 5.3.1 Poor Quality data

A point against publishing data *immediately* on generation is potential quality issues and I would recommend performing basic quality control (QC) analyses before publishing a dataset

along with your QC analysis. If your dataset fails QC so badly it's a total write-off then you shouldn't bother to publish but marginal samples should generally be included even if you exclude them from your analysis with the QC info that you used to make this call. Alternative methods may make data that fails QC for you still useful for someone else's question.

Publishing QC information can be valuable data for anyone trying to perform meta-scientific analyses that might inform decision making of future researchers. for example if a particular instrument or preparation technique has a high rate of QC failure this is a consideration that could be important in picking sample numbers in a future experiment.

# 6 Where To Publish Data

## 6.1 Searching for Research data repositories

[re3data](#) is a global registry of research data repositories with over 3,000 entries where you can search for an appropriate place to deposit your data ([Pampel et al. 2023](#) [cito:citesAsAuthority]). So if you don't know of a suitable repository then searching for one that is a good fit for your data in re3data is a good place to start.

[FAIRsharing.org](#) is a curated resource of educational material on databases, standards, and policies for data sharing.

## 6.2 Selected Public Data Repositories & Data Sharing Platforms

### 6.2.1 Sequencing Data

#### 6.2.1.1 [GEO](#) (Gene Expression Omnibus)

- Accepts data from methods which measure some property of genomic features e.g. expression micro-arrays, RNA-seq, ChIP-seq, ATAC-seq but not genomic sequence data.
- [GEO submission](#)  
GEO has quite a flexible model for metadata you can (and should) include a good deal of additional data along with any sequencing data that you deposit here. They explicitly provide for secondary data (i.e. data derived from your sequencing data) to be included. It is also a good idea, wherever possible, to include the process (or appropriate links to) the process by which you got from

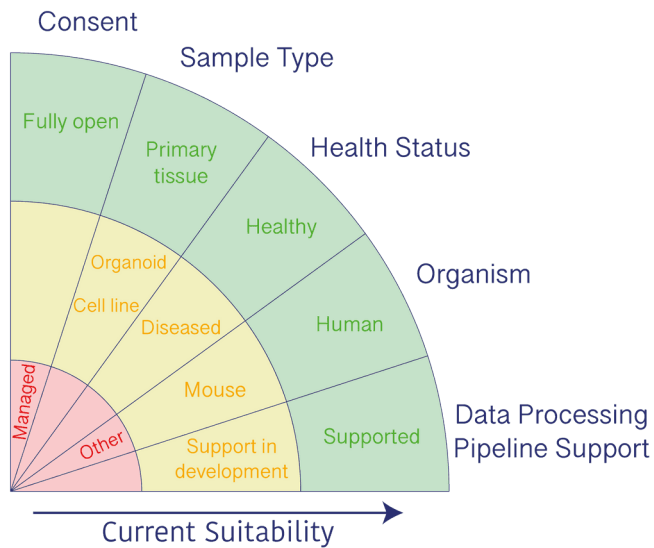
the data/metadata to the secondary results. For example in previous submissions to GEO of RNA-seq data that I processed with a standard [nf-core](#) I have included the count matrices and some other pipeline outputs as secondary results. I also included command run to kick off the pipeline that generated these results, the design matrix input file needed in addition to the sequencing files and the version of the pipeline that I used. This way someone downloading my data could re-capitulate my results exactly by re-running that same version of the pipeline on my raw data. This also means anyone wanting to use my results can interrogate the code in the nf-core pipeline to see exactly how the analysis was performed.

#### 6.2.1.2 **SRA (Sequence Read Archive)**

- Store the raw sequencing data underlying GEO, and other data including genomic sequencing data

#### 6.2.1.3 **HCA (Human Cell Atlas) data portal**

- The Human Cell Atlas project has a [data portal](#)
- Much of HDBI's data is likely to meet the [data suitability criteria for the HCA data portal](#) which are summarized in the image below.



- Submitting data to HCA
  - To submit your data with the metadata in spreadsheet form check out the [spreadsheet guide](#)
  - If you prefer code checkout the guide to [submitting data programmatically](#) and the [metadata standards](#).
  - You can email the HCA data portal’s data wrangling team with any questions: [wrangler-team@data.humancellatlas.org](mailto:wrangler-team@data.humancellatlas.org)
- HCA also has an [analysis tools registry](#) for any tools that you produce for analyzing this data and to which [you can contribute](#).

#### 6.2.1.4 GenBank

- Stores whole genome sequencing data and assembled genomes

#### 6.2.1.5 ENA (European Nucleotide Archive)

- Sequencing data



## 6.2.2 Imaging Data

‘Global Bioimaging’ (Swedlow et al. 2021 [cito:agreesWith] [cito:citesAsAuthority] [cito:citesAsRecommendedReading] [cito:discusses]) is a group founded to: “to disseminate best practices, develop common imaging and data standards that promote data sharing”. They describe a distinction between ‘image data archives/repositories’ and ‘added-value databases (AVDBs)’. The Image Data Resource (IDR) is an added-value database whereas the Bioimage Archive is, as the name suggests, an archive/repository. (see the dedicated sections on these below).

The envisioned workflow for image generation, storage and sharing is outlined at a high level in three steps:

1. Local Data Storage (pre-publication)
2. Archive/Repository
3. Added-value Database

This might for example go:

1. a local OMERO instance
2. Bioimage Archive
3. IDR

Only data with sufficient value to be archived should make it from local data storage into public data repositories. If it is data that underpins a published result then it should be archived. Once in a public archive that data should ideally only be referenced by AVDBs rather than replicated to them to avoid unnecessary duplication. Work is underway at the Bioimage Archive to implement APIs which should make it easier to submit data to the archive directly from a local OMERO instance. This should also make further curation/annotation efforts in the AVDBs like IDR & EMPIAR easier.






When publishing results based on image data their value and the effectiveness of the communication of your results can be increased by following community developed standards such as those described here: (Schmied et al. 2023 [cito:citesAsRecommendedReading]). These recommendations

come with simple checklists to follow for publishing images and image analysis workflows.










## Image Publication Checklist

### Checklist for image publishing






#### Image format

-  Focus on relevant image content (e.g., crop, rotate, resize)  Minimal
-  Separate individual images
-  Show example image used for quantifications
-  Indicate position of zoom view/inset in full-view/original image
-  Show images of the range of the described phenotype

#### Image colors and channels

-  Annotation of channels (staining, marker, etc.) visible  Minimal
-  Adjust brightness/contrast, report adjustments, use uniform color scales
-  Image comparison: use the same adjustments
-  Channel colors: high visibility on the background  
Best visibility: grayscale
-  Multicolors: provide grayscale for each color channel
-  Multicolor: if channels are merged, make accessible to color-blind individuals
-  Provide intensity scales (calibration bar) for grayscale, color, pseudocolor etc.  Recommended
-  Pseudocolored images: additionally provide grayscale version for comparison  Ideal
-  Gamma adjustments: additionally provide linear-adjusted image for comparison  Ideal

#### Image annotation

-  Add scale information (scale bar, image length in figure/figure legend)  Minimal
-  Explain all annotations (in figure/figure legend)
-  Annotations should be legible (line width, size/point size, color)
-  Annotations should not obscure key data
-  Annotate imaging details important for interpreting the figure (depending on the main message and imaging technique, this may be, e.g., image pixel size, imaging intervals (time-lapse in movies), exposure time or anatomical section)  Recommended

#### Image availability

-  Images are shared (lossless compression/microscope images)
-  Image files are freely downloadable (public database)  Recommended
-  Image files are in dedicated image database (added-value database or image archive)  Ideal

**i** Image Analysis Workflow Publication Checklist

## Checklists for publication of image-analysis workflows

### Established workflows

	Cite workflow and platform	<input type="checkbox"/>	Minimal
	Key settings	<input type="checkbox"/>	
	Example data	<input type="checkbox"/>	
	Manual ROI	<input type="checkbox"/>	
<b>193</b>	Exact version	<input type="checkbox"/>	
	All settings	<input checked="" type="checkbox"/>	Recommended
	Public example	<input checked="" type="checkbox"/>	
	Document usage (e.g., screen recording or tutorial)	<input checked="" type="checkbox"/>	Ideal
	Cloud hosted or container	<input checked="" type="checkbox"/>	

### New workflows

	Cite components and platform	<input type="checkbox"/>	Minimal
	Describe sequence	<input type="checkbox"/>	
	Key settings	<input type="checkbox"/>	
	Example data and code	<input type="checkbox"/>	
	Manual ROI	<input type="checkbox"/>	
<b>193</b>	Exact versions	<input type="checkbox"/>	
	All settings	<input checked="" type="checkbox"/>	Recommended
	Public example data and code	<input checked="" type="checkbox"/>	
	Rationale	<input checked="" type="checkbox"/>	
	Limitations	<input checked="" type="checkbox"/>	
	Screen recording or tutorial	<input checked="" type="checkbox"/>	Ideal
	Easy install and usage, container	<input checked="" type="checkbox"/>	

### Machine learning workflows

	Cite original method	<input type="checkbox"/>	Minimal (all models)
	Access to model	<input type="checkbox"/>	
	Example or validation data	<input type="checkbox"/>	
	Training and testing data and metadata	<input checked="" type="checkbox"/>	Recommended (pretrained and new models)
	Code available	<input checked="" type="checkbox"/>	
	Limitations	<input checked="" type="checkbox"/>	
	Cloud hosted or container	<input checked="" type="checkbox"/>	
	Standardized format	<input checked="" type="checkbox"/>	Ideal (new models)

When depositing images in repositories such as the BIA then the REMBI (Recommended Metadata for Biological Images) standard provides excellent guidance for making image data FAIR ([Sarkans et al. 2021](#) [cito:citesAsRecommendedReading]).

#### 6.2.2.1 **IDR (Image Data Resource)**

- An added value database of high quality well annotated bio-image data of cells and tissues
- Quite extensive manual curation, currently only accepting ‘reference’ collections with high potential for re-use.
- IDR is an instance of OMERO, thus managing your metadata in a local instance of OMERO should make it easier to release result here.

#### 6.2.2.2 **Bioimage Archive (EBI)**

- An Archival repository of biological images
- Broad scope, any scale or modality
- Associated with a publication or resource of general interest.

#### 6.2.2.3 **EMPIAR (Electron Microscopy Public Image Archive)**

- An Added value database for Electron microscopy data

#### 6.2.2.4 **figshare**

- Sharing of other images / figures. This can include things like raw blots and gels underlying more highly edited figures in a paper.

### 6.2.3 **Protocols**

Useful article in Nature about [writing reproducible lab protocols](#)([Baker 2021](#) [cito:citesAsRecommendedReading]).

### 6.2.3.1 [Protocols.io](#)

- Publish details of your laboratory protocols. Step-by-step procedures optionally supplemented them with images and other media as a supplement to the textual descriptions of the methods. (Unfortunately [protocol.io](#) is proprietary platform operated by a private company (Springer Nature) not a publicly owned archive or open source tool but I'm not aware of any good alternatives at the moment.)

### 6.2.3.2 [JOVE](#)

- (Journal of Visualized Experiments) Publish videos of how you perform your experimental work. This makes it easier share intricate experimental details not readily captured in text.

## 6.2.4 Code & Computational Environments

This [Flow Diagram](#) is intended to guide you through the steps of sharing, publishing and distributing different kinds of research software outputs. In addition to the the diagram there is further information in the sub-sections below.

### 6.2.4.1 Software packages

- In language specific package repositories
  - **R:**  
[bioconductor](#), [CRAN](#) (Comprehensive R Archive Network), both have review processes for submitting packages to their repositories.
  - **Python:**  
[PyPi](#), anyone can upload

- Software publications

If you have written a piece of open source software as a part of your research that stands alone as a substantial scientific output the you might want to turn it into an academic publication with peer review. These slightly alternative journals facilitate that.

- [JOSS](#) (Journal of open source software)
- [rOpenSci](#) specifically for R packages
- [PyOpenSci](#) specifically for Python packages

#### 6.2.4.2 Bioinformatic analysis pipelines

If you have constructed a robust bioinformatic analysis pipeline that does the sort of data processing that other might want to do as well, then as long as you have used the appropriate tools to build your pipeline there are options to share them with a wider community of researchers.

- [WorkflowHub](#) for any type of workflow
- [nf-core](#) for [nextflow](#) pipelines
- [targetopia](#) for R {[targets](#)} pipelines (via [rOpenSci](#)) - more focused on composable components of pipelines that can be connected together to perform certain types of analysis that necessarily complete pipelines

#### 6.2.4.3 Scripts, Notebooks and project specific workflows can be shared as [git](#) repositories.

- on git Hosting platforms: [Renku](#) Section 4.8, [Gitlab](#), [Github](#)
- You can generate DOI's for your projects with [Zenodo](#)

**i** HDBI Zenodo URLs

**Collection URL:**

<https://zenodo.org/communities/hdbi/>



Above address links directly to your community collection.

**Upload URL:**

<https://zenodo.org/deposit/new?c=hdbi>

Above address will automatically ensure people who use it will have their record added to your community collection.

**Curation URL:**

<https://zenodo.org/communities/hdbi/curate/>

Above address links to your private curation URL. You will find all uploads pending your curation.

**Harvesting URL:**

[https://zenodo.org/oai2d?verb=ListRecords&set=user-hdbi&metadataPrefix=oai\\_dc](https://zenodo.org/oai2d?verb=ListRecords&set=user-hdbi&metadataPrefix=oai_dc)

Above address links to a OAI-PMH feed, which can be used by other digital repositories to harvest this community.

- Docker images - pre-built reproducible computational environments
  - [Docker Hub](#)

## 6.2.5 Biological Materials access / Sharing

### 6.2.5.1 HDBR (Human Developmental Biology Resource)

- “[HDBR] is organised from two sites: the Institute of Genetic Medicine, Newcastle, and the Institute of Child Health, London. The HDBR is an ongoing collection of human embryonic and fetal material ranging from 3 to 20 weeks of development.”
- See also the [HDBR Atlas](#): “a digital atlas comprising 3D reconstructions from Carnegie Stage 12 to 23, generated using Optical Projection Tomography (OPT), and annotations of the 3D models linked to an anatomical database”

## 6.2.6 Spatial transcriptomics

A consensus has yet to emerge in this area and different technologies have different underlying datatypes, some use sequencing and some are more array like.

The [SpatialData](#) format is emerging as an open format for processed spatial data. It is most similar to that of other image data as it is built on [zarr](#) data structures and has been developed in coordination with the OME-NGFF efforts. It is possible therefore that BIA might take submissions in this form, one could consider the expression matrices as rather extensive image metadata.

The [Haniffa lab's webatlas](#) is a good tool for viewing this data especially if it is integrated with single cell transcriptomics.

## 6.2.7 Flow Cytometry

- [flowrepository](#) the International Society for Advancement of Cytometry (ISAC) FCS File Repository
  - Data deposited in the flow repository should meet the MIFlowCyt (minimum information about a Flow Cytometry Experiment) standard ([Lee et al. 2008](#) [cito:citesAsAuthority]), This paper Guide to preparing data that meets the MIFlowCyt standard ([Spidlen, Breuer, and Brinkman 2012](#) [cito:citesAsRecommendedReading]).

## 6.2.8 Proteomics

[PRIDE](#) is the primary repository for proteomics data. To submit data to them you need an account and to download their Java based submission tool, the process is [well documented](#) on their website.

### 6.2.9 None of the above

If your data does not fit into any of the above categories and you can't find a public repository that will host it for you then there are number of generalist repositories like [Zenodo](#) see [this Zenodo publication on choosing a generalist repository](#), [OSF](#), & [Dyrad](#)

If for some reason even these generalist repositories don't work for you you might consider hosting your own instance of [data-verse](#), [DataHub](#) or [iRODS](#) which are software projects that provide tooling for managing your own data repository.

## 6.3 Integrated publishing - a possible future

*Data, analysis, prose, collaboration, pre-print, review and publication in one place with literate programming and single source publishing*

You begin your project on an instance of a platform like [Renku](#) (section Section 4.8), Start by uploading your raw data to a domain specific data repository. You get a DOI or accession for your dataset. You import this into your project. You perform your computational analyses in the reproducible computational environment. Potentially documenting your analysis as a workflow that could be used by others with a pipeline management tool. You write your manuscript in a literate programming format like Quarto. You work with your collaborators on the manuscript using a git hosting tool like gitlab where you raise and discuss issues, and share revised versions. You generate your statistics and graphics for inclusion in the manuscript with code from your data in a reproducible computational environment. You publish a pre-print by making use of a static site generator like the one built into gitlab and simply setting the project to public. You tag this version 0.0.0 and associated it with a DOI from zenodo. To manage reviews of your work you make use of gitlab issues in a manner similar to the review processes of [JOSS](#), [rOpenSci](#) and [f1000](#) but potentially independent of a particular publication venue through community peer review projects like [Peer Community In \(PCI\)](#) & [Review](#)

**Commons.** This approach permits author led updates, errata, & corrections whilst preserving a version of record (Kane and Amin 2023 [cito:agreesWith]). Once Reviewed and published you have the 1.0.0 version of your manuscript, for future minor corrections you increment the patch version 1.0.1 and your change-log reflects that you fixed a typo. If you add a new dataset or fix an error that changes an outcome you increment the minor version number. If the journal updates the version of record you increment the major version number.

**In this Fashion the complete history of the project is documented start to finish and you never had to change medium from scripts to manuscripts in word processors to emailing pdfs, to publisher websites etc. Review is handled with the same set of tools as was your internal collaboration with co-authors. pre-print publication is creating a version tag and setting the repo to public. Anyone can pick up your project in it's entirety and play around with their own variants of your analysis at the click of a button (specifically the 'fork' button).**

## 7 What Data To Publish

### *Data Sharing policies and privacy protection considerations*

In the general case when sharing research data as much detail as possible should be shared to maximize the utility of the data to the scientific community. **When dealing with human data however considerations of consent and privacy are paramount.**

When considering what data should or should not be deposited publicly you should consult the consent agreements signed by the tissue donors of samples used in your study. When these samples come from a tissue bank you should consult the tissue bank where you should be able to find details of how participants were consented. There will also generally be guidance on what data they are able to share with you as a researcher and which portions of that data should not be made generally available. Many of the tissue samples used by HDBI researchers are from the [Human Developmental Biology Resource \(HDBR\)](#). For additional details on HDBR policy about the sharing of data relating to their sample see the [HDBR's data sharing policy](#)

Data which pertains to technical or biological properties of the sample are generally suitable to be shared. Data about the donor of the sample will generally not be suitable to be shared. This will depend on the specific consent agreements associated with a given sample and should not be assumed.

Data points like sample collection dates can be a problematic, where time between two points in a sample's history might be relevant e.g. time from collection to processing this is better expressed in relative time rather than absolute dates.

## 7.1 Additional Ethical & Legal Considerations

*Context to be aware of when making data-sharing policy decisions*

Where samples have been donated under the condition that any published data be pseudonymised care should be taken to minimize re-identification risk. i.e. the risk that the identify of the donor could be re-associated with the data. One of the challenges in this space is that **any genetic data now carries inherent re-identification risk.**

Re-identification attacks are generally carried out by joining together disparate pieces of information which are not separately identifying but which when combined can uniquely or probabilistically identify an individual. These attacks can take forms which are difficult to anticipate.

For example cross-referencing a list of names and clinic appointment times with sample collection dates could substantially narrow the field of possible donors of a given sample, potentially even uniquely identifying the donor. This only requires two mappings, name to appointment data and sample to collection date. If genetic information is tied to sample collection date and there are still more than one possible person who could be the donor this could be employed to attempt to narrow down identities further. Genetic data can be used to estimate ancestry and ethnic origin which could probabilistically re-identify someone with a name of characteristic ethnic or geographic origin. **Even sequencing data which is not primarily intended for genotyping such as RNA-seq can be used for this purpose.** This paper for example performs a comparison of SNP (Single Nucleotide Polymorphism) calling methods using RNA-seq data (Quinn et al. 2013 [cito:citesAsAuthority]).

Unfortunately there are numerous examples of cybercriminals using open source intelligence (OSINT) techniques and leaked medical data to target and exploit victims. Whilst not yet a popular attack vector with cybercriminals tying an individual's identity to their (or close relatives') genetic profile(s) is a potential avenue of attack. Related methods have already been

used by law enforcement to catch criminals, a topic which has entered the public discourse with prominent science communication content creators like [Veritasium covering the subject](#).

In the EU the sharing of genetic data is governed by the GDPR. This also largely applies to the UK post brexit but attention should be paid to any divergences of the UK GDPR from the EU GDPR. Genetic data is defined in the EU GDPR by [Recital 34](#) (see also [article 4 definition 13](#)) and its processing permitted for specified purposes where the data subject has given consent (see [article 7](#) & [article 4 definition 11](#)) for said processing under [article 9](#). **The Public Health Genomics Foundation has produced an extensive report on GDPR and genomics data which aids significantly with the interpretation of this legislation in the context of genomic data.** Genetic data in particular presents challenges around who the data subject is as your genetic data is also in part your relatives genetic data.

[a nice talk giving a practical look at health data anonymisation under GDPR with an example from Prof. Dr. Fabian Prasser of the Berlin Institute of Health.](#)

In the USA GINA (Genetic information non-discrimination act) 2008 & Health Insurance Portability and Accountability Act (HIPAA) 1996 are the primary pieces of legislation governing genomic data sharing. This talk: [De-identification Standards: What Works, What Doesn't, and What Fails Miserably](#) by [Bradley Malin](#) of [Vanderbilt University](#) provides an interesting compare and contrast of the the EU and US approaches to this question.

Genetic data sharing in the consumer space is currently something of a regulatory 'wild-west' with consumers able to 'consent' to genetic data sharing practices that might be prohibited or closely scrutinized for public entities though accepting end user licence agreements (EULAs). These agreements are unilaterally modifiable by the entity collecting the data and it is widely accepted that almost no one reads them.

This has implications for research data sharing as databases of direct to consumer genetic testing results can potentially be

used in attempts to re-identify data subjects. Secondary genetic analysis platforms which take data generated by direct to consumer genetic test providers and provide additional analysis results are also becoming more prevalent. These include sites with significant repositories of user submitted data often for the purposes of performing genealogical analysis. These developments represent a lowering of the technical barriers to potential abuses of this data for purposes such as re-identification. A fact that is relevant for the interpretation of whether or not data can be considered anonymous under GDPR [recital 26](#).

#### **i** Additional Data Ethics Resources

##### [HDBI Research Tissues & Ethics page](#)

The [Global Alliance for Genomics & Health \(GA4GH\)](#) an independent, non-governmental, not-for-profit, international association provides a number of toolkits with useful resources for those working with genomic data.

- [Genomic Data Toolkit](#)
- [Regulatory & Ethics Toolkit](#)
- [Data Security Toolkit](#)



# 8 How To License Your Data

*Optimize for breadth, FAIRness, & ethical re-use*

*If you do not suitably license your work by default others are restricted from re-using it without asking for your permission first.* Under most jurisdictions you retain the copyright to your work by default so if you want to permit full re-use you must explicitly allow this, the best way to do this is usually by using an existing open license. Here is how to pick and use one.

“The world’s most valuable resource is no longer oil, but data”

- Kiran Bhageshpur, the Economist, 2017

## 8.1 Data

It is fairly common to release research data publicly under the CC-0 ‘public domain’ license in which no rights are reserved by the author and anyone may use the data for any purpose. Alternatively CC-BY can be an option, under this license anyone using the data is obligated to attribute the data to its author through e.g. citation in order to be in compliance with the copyright on this work. It is the norm to provide attribution of some kind even for CC-0 licensed data in the scientific community but this is not a legal obligation under the terms of the license, unlike CC-BY. These licences are those recommended in [Wellcome’s data guidelines for authors](#).

It is commonly stated that research data is a public good and should be released with as few restrictions as possible, whilst requiring attribution is an additional restriction it increases the provenance of data by making its source known. Research data is a public good, research data with good metadata and provenance is a still greater good.

### 8.1.1 Images

**Like software; images, figures and diagrams should come with licenses.** If an image lacks a license the default is their creator reserves all rights (in most copyright regimes). Thus if you make a figure or diagram people will not be able to use it without exposing themselves to potential copyright claims unless you use a licence which explicitly permits the use of your work by others.

If you are in need of some freely licensed images to use in your own scientific diagrams, illustrations or figures [bioicons](#) is an excellent source.



The [creative commons licenses](#) are probably the best choice here. The base creative commons licence has a number of modifiers that can be applied in combination to it.

- **BY** - by attribution, you must attribute the work to it's originator in order to reuse it
- **SA** - Share alike, if you redistribute the work or a derivative of it you must do so under the same licence as the original
- **NC** - Non-commercial, you may not redistribute the work for commercial purposes
- **ND** - No derivatives you may redistribute the work but only in unmodified form

For example the most restrictive combination would be CC-BY-SA-NC-ND, this work can only be redistributed if the author is credited, the license used is the same as the original, it is unchanged and it is not for commercial purposes.

The image shows a guide to Creative Commons licenses. It features the Creative Commons logo and the word 'LICENSES'. Below this is a table with seven rows representing different license types and five columns representing permissions. Each cell in the table contains a checkmark (indicating the permission is allowed) or an 'X' (indicating it is not allowed). The license types are: Public Domain, BY (Attribution), BY-SA (Attribution ShareAlike), BY-ND (Attribution NoDerivs), BY-NC (Attribution NonCommercial), BY-NC-SA (Attrib NonComm ShareAlike), and BY-NC-ND (Attrib NonComm NoDerivs). The permissions are: Copy & Publish, Attribution Required, Commercial Use, Modify & Adapt, and Change License.

	Copy & Publish	Attribution Required	Commercial Use	Modify & Adapt	Change License
Public Domain	✓	✗	✓	✓	✓
BY Attribution	✓	✓	✓	✓	✓
BY-SA Attribution ShareAlike	✓	✓	✓	✓	✗
BY-ND Attribution NoDerivs	✓	✓	✓	✗	✓
BY-NC Attribution NonCommercial	✓	✓	✗	✓	✓
BY-NC-SA Attrib NonComm ShareAlike	✓	✓	✗	✓	✗
BY-NC-ND Attrib NonComm NoDerivs	✓	✓	✗	✗	✓

Figure 8.1: Guide to Creative Commons Licenses

You can use the [Creative Commons License Chooser](#) to find a suitable license and/or generate suitable attributions to creative commons content which appropriately link to the original work, and to the license text.

These licenses can apply to any multimedia, audio, video or other digital files that you produce and research products.

## 8.2 Software

- **You should avoid publishing code without any accompanying license as the author reserves all rights by default in most copyright regimes. Consequently anyone using code with no associated licence is opening themselves up to copyright liability.**
- Software produced by members of the HDBI should be licensed with a license approved by the [Open Source Initiative \(OSI\)](#), (or the more opinionated [Free Software Foun-](#)

dition (FSF)), in accordance with the guidelines from the Wellcome trust.

### 8.2.1 Quick Primer on choosing a software license

Software licences can be placed into three broad categories, **proprietary** or copy right, **permissive** and **copy left**.

- In **proprietary** software the source code is not generally available, though some is ‘source available’ (this is not the same as ‘open source’). Thus it’s internal operations are not generally transparent to the end user, a state of affairs problematic for the transparency of the scientific process. Users of proprietary software lease or ‘buy’ permission to use the software under the terms of a license leases are not always paid monetarily, they are often paid in access to user data which can be monetized through services to the software’s customers.
- In **permissively** licensed software the source code is available and the user is free to do more or less whatever they like with it.
- ‘**Copy left**’ licensed software requires that if you distribute any derivatives of the original software you must publish the source code under the same, or a compatible, license.

#### 8.2.1.1 Recommendations

- **Copy left**
  - GPL  $v \geq 3.0$  (General Public Licence) [*My preferred license for non-network software*]
    - \* Use a GPL licence when you want to ensure that your software and any derivatives of it remain freely available to the community and cannot be re-packaged extended and re-sold under proprietary licenses. Paid services are still possible with GPL code e.g. hosting or additional development work under contract.

- AGPL (Affero General Public Licence) [*My preferred license for software used over a network*]
  - \* AGPL is essentially identical to the GPL with an extra stipulation especially for software that runs on a server that others might use as a service. This licence requires that the source code must be available to anyone using the software over a network.
- LGPL (Lesser General Public Licence)
  - \* The lesser GPL license permits a software library to be used in a proprietary application whilst keeping the library itself copy left

- **Permissive**

- Apache 2.0 permissive [*My preferred permissive License*]
  - \* Patent and copyright are distinct areas of law and patents exist on software processes in some jurisdictions thus permissively licensed code for copyright purposes can still be in violation of patents. The Apache license grants a perpetual royalty free license to use any patents held by the licener that are used in the software. This only applies to patents held by the organisation licensing the software and thus cannot protect you from infringing on patents held by 3rd parties.
- ‘MIT’ (aka Expat)
  - \* A short simple permissive license covering the software and it’s documentation it permits the use of the software essentially without restriction but with no warranty.

I default to the use of the ‘copy-left’ or ‘share alike’ licenses as I regard these as the most ethical choice in most contexts. There are however reasons why **you might not want to use these licenses. They can be an impediment to working with certain commercial partners whose business models make use of proprietary licensing which is not**

**always compatible with copy left / share alike licensing.**

3rd parties that offer proprietary commercial software products may avoid using your code if it has a copy left license like the GPL. This might be a problem if you for example you wrote a library that reads a particular type of file and a company wants to use your library to read that file type in a proprietary analysis tool that they sell licenses to. Thus if you are writing tools that you would like companies to be willing to include in software products with paid licenses you may want to opt instead for a permissive licence like the Apache 2.0.

**i** A note on patents.

Patents function somewhat differently to copyright despite the common conflation of these two distinct areas of law under the term IP (intellectual property). Prior disclosure of an invention or process that you wish to patent in any venue including a conference paper or online post can be an impediment to being granted a patent in prominent patent jurisdictions like the USA. Some other jurisdictions have less strict criteria for prior disclosure. This is in contrast to copyright where default presumption is that all rights are retained by the author. Patenting is an affirmative process there are generally fees associated with asserting a patent and they are subject to approval by the patent office in your jurisdiction. Not all jurisdictions have a concept of software patents as exists in the USA under the rubric of business method patents. If you are working on something that you are interested in patenting you should not publish anything revealing its patent-able aspects.

I would advise reviewing Wellcome's guide on '[intellectual property](#)' for relevant guidance from HDBI's funder.

**i** A note on Contributor License Agreements (CLAs).

CLAs are agreements made by contributors to a open source software project. They cede the copyright claim of contributors on the code that they write to the organisation that is the steward of the project. Alternatively they

may more narrowly provide for an agreement to permit the dual-licencing of code over which the original author retains copyright. This can facilitate the organisation's ability to dual-license the code for commercial purposes. It also makes it easier for them to take a code base closed source and make future development proprietary as they not longer need the consent of all contributors to make changes to the licensing but can act unilaterally.

### 8.3 Retaining Rights

Prior to the submission of a work for publication authors should apply a suitable license such as a CC BY license to the work. This permits the retention of the rights by the authors to that work so that 'author accepted manuscript' can be freely redistributed by the authors under the terms of this license. For example to update pre-prints to match the 'version of record' including changes during peer-review. Taking this approach permits works to be released immediately, not following a potentially lengthy embargo period, as is now required by many funders.

This text, or similar, should be included in submitted manuscripts and alluded to in submission cover letters:

For the purpose of open access, the author(s) has(have) applied a Creative Commons Attribution

To find out more about rights retention see the [Plan S Rights Retention Strategy](#) and [Rights retention: A Primer from UKRN](#)

#### Disclaimer

I am not a lawyer this is not legal advice. If you have any questions about how any of these consideration apply to your work please consult with a suitable professional legal expert.\*

## 8.4 Resources

- For a deeper dive into licensing checkout the [chapter in the turing way on licensing](#).
- [Creative Commons License Chooser](#)
- Freely licensed images to use in your own scientific diagrams, illustrations or figures [bioicons](#).
- The [REUSE](#) initiative started by the [free software foundation europe \(FSFe\)](#) provides some useful tooling to ensure that licences of your code are clearly denoted. This is more useful in larger project that also ships code with other licences but is a useful reference for practical licencing best practices.

### A Short video guide to software licences



# 9 How To Manage References

Use [Zotero](#)

## 9.1 Zotero

Why Zotero? There are a number of other reference managers available perhaps the most popular is [Mendeley](#) so why am I recommending Zotero? A number of years ago Mendeley might have been a better choice than Zotero in terms of features, this gap has now largely closed with Zotero providing superior features in a number of areas. The primary area of disparity where Mendeley is still ahead is mobile app support. Predictably Mendeley's development stagnated after an [initial bump in activity arising from the cash infusion following its acquisition in 2013 by Elsevier](#). It has become increasingly closed and as of 2018 prevents complete export of user data [by encrypting its local database](#).

**A reference management system is something that you will ideally have with you for your entire academic career.** Building a comprehensive personal database of references and personal notes/annotations represents a **massive investment of time and energy**. **Longevity, openness and interoperability are therefore features that should be prioritized in selecting a reference manager.** If a particular tool ceases to exist you want to be able to migrate your library to new ones as easily and completely as possible. It is also advantageous to have a library which readily interoperates with additional tools for exploring analyzing and processing that library. Zotero being an open source software project it is much more likely to have and retain these characteristics than any proprietary solution which are incentivised to lock-in users to their platform by reducing data portability.

### 9.1.1 RSS feeds in Zotero

#### *Really Simple Syndication*

One of my favorite features of Zotero for keeping up to date with the latest developments in the literature is the RSS feed reader.

You can import a new RSS feed by clicking: **New Library (Icon, top left) > New Feed > From URL**

Some of the best ways to populate your feed with interesting papers/manuscripts are:

1. Creating RSS feeds from NCBI/PubMed searches is trivially easy  
Define any custom search you want with PubMed's advanced search tool and simply click the 'Create RSS' link under the search box. Any new results for these search terms will now appear in this feed.



Figure 9.1: PubMed RSS

2. [BioRxiv's RSS feeds](#) are useful for staying ahead of the curve.
3. Most blogs have or can be adapted to an RSS feed so if you have a favorite academic blogger you can get their latest posts directly in Zotero

4. Are you a compulsive follower of academic twitter? Get your favorite follows' feeds directly in your reference manager with [fetchrss](#) to more quickly import their recommended reading. If Elon's completely trashed twitter by the time you're reading this every mastadon account is also an RSS feed by default so you come join us in the [fediverse](#) [genomic.social](#), [scholar.social](#) and [fediscience.org](#) are instances with an academic bent and an alternative platform governance model (W. Gehl and Zulli 2022 [cito:citesAsRecommendedReading]). You can find me [@RichardJActon@fosstodon.org](#).

### 9.1.2 Zotero Web

Zotero has a web-client through which references can be accessed, shared and synchronized between devices. There is a cap to the amount of data you can store on their servers for free, after this you can either pay a subscription for more storage or host your own synchronization server (though this is not easy).

The web-client is the way I would recommend using Zotero on mobile. You can pin the app to your phone's home screen using the Firefox mobile browser by [tapping install in the menu](#) once you have navigated to the page for the Zotero web app. Depending on your mobile platform you may be able to install Zotero Web as as progressive web app with other browsers.

You can create groups in Zotero Web to share references with others I recommend that labs maintain Zotero groups with reference collections for:

- The labs own publications
- A curated list of recommended reading for new lab members
- The papers from any Journal Club sessions that lab holds

Checkout the [Zotero library of HDBI publications](#), if yours are not there [email me](#) and I'll get them added as long as they acknowledge the HDBI Wellcome grant (ref: 215116/Z/18/Z).

### 9.1.3 Extensions & Integrations

Zotero has an [ecosystem of plugins](#) which extend its functionality.

Zotero integrates with Microsoft Word, [LibreOffice](#), [OnlyOffice](#), and Google Docs as well as RStudio's Visual document editor mode.

If you are a user of note taking applications such as [LogSeq](#) or [Obsidian](#) Zotero has nice integrations with many of these as well.

To enable native Zotero RStudio Integration generate and API key from the Zotero web interface. (I recommend that you store it securely in your bitwarden vault as [custom hidden field](#) associated with your Zotero web login credentials.)

Settings > Feeds/API > Create new private key

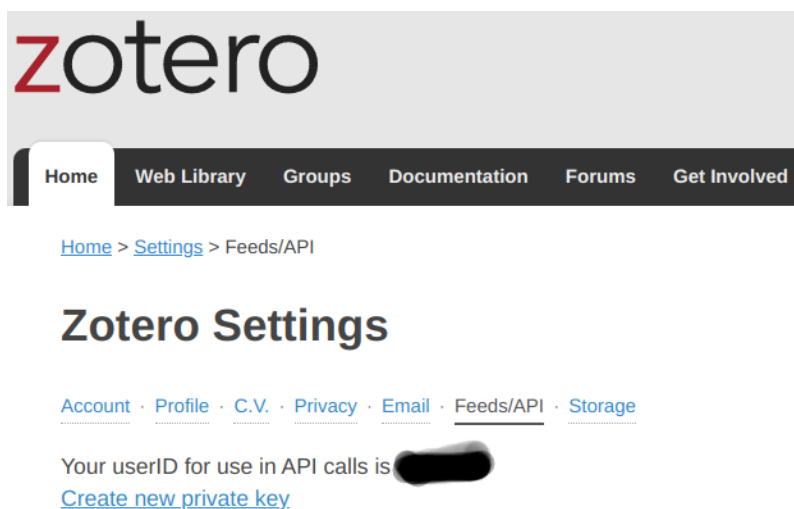


Figure 9.2: Create Zotero API Key

You can enter this API key in RStudio at Tools > Global Options > Rmarkdown > Citations.

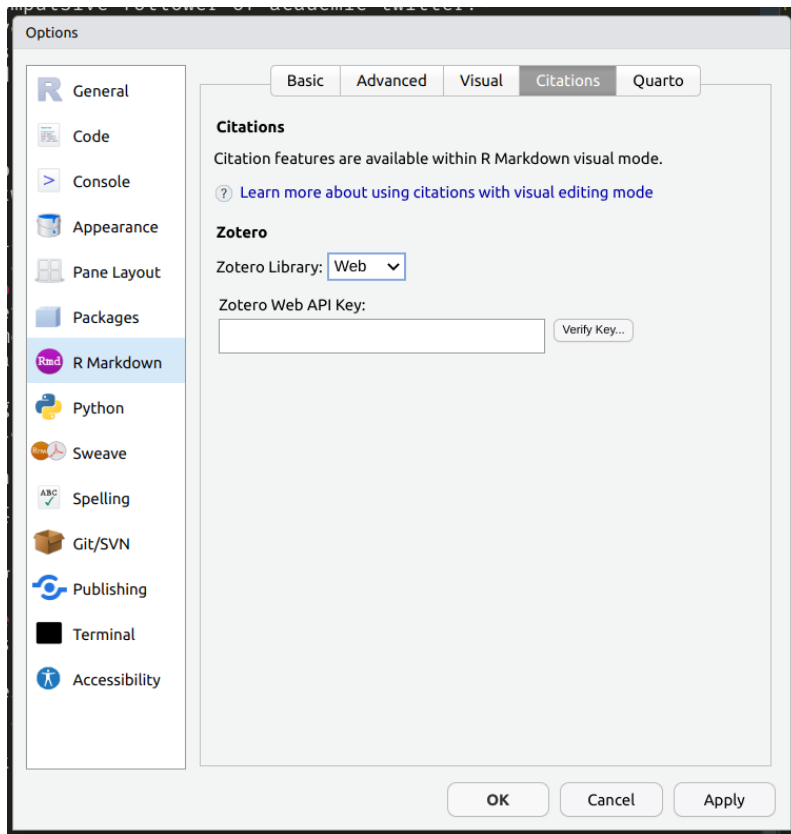


Figure 9.3: Enter Zotero API Key in RStudio

#### 💡 Zotero Learning Resources

- **Video** (*quickstart*) [Zotero for Beginners: A guided walkthrough](#)
- **Video** (*longform*) This is a playlist of an extended Zotero Course [Mastering Zotero](#)
- **Video** (*shortform*) Using Zotero Groups for collaborative projects [Zotero Tutorial: Group Collaboration](#) like [the one I setup for HDBI papers](#).
- **Text** (*documentation*) [Official Zotero Documentation](#)

## 9.2 Personal Knowledge Management

*Bullet Journaling, Notes, Task Management, Exobrain, Zettelkasten, spaced repetition & productivity*

**For things that don't fall under the umbrella of bibliographies or lab notes you may need a different approach and set of tools to their electronic organisation.** Personal Knowledge Management can be a serious rabbit hole of exploring all the many and varied tools and organisational philosophies that exist out there. **Whatever approach you choose to take to this I would advise: "keep it simple or you won't keep it up".**

It can be **incredibly useful to have an dedicated electronic notes tool to keep track of notes**, thoughts ideas, sources less formal bibliographic references, tasks and the relationships between them. **One of the biggest value adds over paper is keyword search.** Remember a seminar you attended three years ago on topic X or by person Y? just search for it/them and bang! you've got your notes from that session. Another useful feature in many modern note taking apps can be representations of relationships between notes as graphs so that you can explore related notes containing related concepts.

When it comes to choosing a personal notes tools the same advice that I offered for selecting an ELN in Section 4.1 and a reference manager in Chapter 9 largely holds true.

"[W]e are all chimeras, theorized and fabricated hybrids of machine and organism; in short, we are cyborgs."

- Donna Haraway  
(1991)

**i** General Note Taking with simple task management apps

### 9.2.0.1 Open Source Options

#### 9.2.0.1.1 Personal favorites

[logseq](#)

- Simple text notes in markdown/[emacs org-mode](#) like syntax
- Strong Daily log feature
- Graph of note relationships

- Extensions / plugins system
- Zotero Integration
- Spaced repetition / flashcards tool with [anki](#) integration
- Optional syncing with git or basic file sync through tools like (Syncthing, Nextcloud, dropbox, Google drive)

### Trilium

- Simple text based notes in markdown with full support for extensions like *LaTeX* Maths and [mermaid diagrams](#)
- freehand drawing with canvas notes
- Very Feature Rich
- Highly extensible / hackable / customisable with scripting
- Graph of note relationships
- Split view for looking at multiple notes
- Optional syncing through a server, self-hosted or paid hosting available
- Built-in encryption option

#### 9.2.0.1.2 Others

##### [zettlr](#)

- Simple markdown based notes
- Zotero Integration
- Specific support for [Zettelkasten](#) style information organisation
- pandoc based document export to pdf and other formats

## Joplin

- Dead simple markdown based notes, & tasks
- Well established and Reliable
- Many solid syncing options
- Built-in encryption option
- Good mobile app support
- Browser plugins to capture web content

## Focalboard

- Simple text based notes in markdown like syntax
- Nice to those who like the [kanban](#) style of task management
- Features very similar to a minimal version of Notion
- Tabular data view
- optional sync, focal board has a paid hosted option

### 9.2.0.1.3 Up & coming

[AFFiNE](#) has some cool UI innovations but is still in Alpha  
[appflowy](#) Similar to Focalboard / Notion

### 9.2.0.2 Some popular proprietary `emo::ji("disappointed")` solutions

## Obsidian

- Simple text notes in markdown (paid closed-source servers but open notes file format)
- Graph of note relationships

## Notion

- General markdown like text notes
- Tabular data view
- More team focused features



For the sake of brevity I'm not going to delve into separate todo/task management solutions independent of tools whose primary function is note taking for now.

**Your notes and/or task management tool, your password vault, your electronic lab notebook and your potentially your Renku projects come together to form an external extension of mind an exobrain if you will, a prosthetic that extends your information management, integration & recall capabilities.**

# HDBI Logos, Links, & Communities

[HDBI website](#)

## Code Repositories

- [github](#)
- [gitlab.com](#)
- [renkulab.io](#)

## Protocols

- [protocols.io](#)

## Publications

- [Zotero Library](#)

Baker, Monya. 2021. “Five Keys to Writing a Reproducible Lab Protocol.” *Nature* 597 (7875): 293–94. <https://doi.org/10.1038/d41586-021-02428-3>.

Bankhead, Peter, Maurice B. Loughrey, José A. Fernández, Yvonne Dombrowski, Darragh G. McArt, Philip D. Dunne, Stephen McQuaid, et al. 2017. “QuPath: Open Source Software for Digital Pathology Image Analysis.” *Scientific Reports* 7 (1). <https://doi.org/10.1038/s41598-017-17204-5>.

- Higgins, Stuart G., Akemi A. Nogiwa-Valdez, and Molly M. Stevens. 2022. “Considerations for Implementing Electronic Laboratory Notebooks in an Academic Research Environment.” *Nature Protocols* 17 (2): 179–89. <https://doi.org/10.1038/s41596-021-00645-8>.
- Hinsen, Konrad. 2018. “Verifiability in Computer-Aided Research: The Role of Digital Scientific Notations at the Human-Computer Interface.” *PeerJ Computer Science* 4 (July): e158. <https://doi.org/10.7717/peerj-cs.158>.
- Kane, Adam, and Bawan Amin. 2023. “Amending the Literature Through Version Control.” *Biology Letters* 19 (1). <https://doi.org/10.1098/rsbl.2022.0463>.
- Lee, Jamie A., Josef Spidlen, Keith Boyce, Jennifer Cai, Nicholas Crosbie, Mark Dalphin, Jeff Furlong, et al. 2008. “MIFlowCyt: The Minimum Information about a Flow Cytometry Experiment.” *Cytometry Part A* 73A (10): 926–30. <https://doi.org/10.1002/cyto.a.20623>.
- Markowetz, Florian. 2015. “Five Selfish Reasons to Work Reproducibly.” *Genome Biology* 16 (1). <https://doi.org/10.1186/s13059-015-0850-7>.
- McInturff, Stephen, and Victor Adenis. 2022. “It Takes a Laboratory to Avoid Data Loss.” *Nature*, September. <https://doi.org/10.1038/d41586-022-02967-3>.
- Moore, Josh, Chris Allan, Sébastien Besson, Jean-Marie Burel, Erin Diel, David Gault, Kevin Kozłowski, et al. 2021. “OME-NGFF: A Next-Generation File Format for Expanding Bioimaging Data-Access Strategies.” *Nature Methods* 18 (12): 1496–98. <https://doi.org/10.1038/s41592-021-01326-w>.
- “Nature Welcomes Registered Reports.” 2023. *Nature* 614 (7949): 594–94. <https://doi.org/10.1038/d41586-023-00506-2>.
- Pampel, Heinz, Nina Leonie Weisweiler, Dorothea Strecker, Michael Witt, Paul Vierkant, Kirsten Elger, Roland Bertelmann, et al. 2023. “Re3data – Indexing the Global Research Data Repository Landscape Since 2012.” *Scientific Data* 10 (1). <https://doi.org/10.1038/s41597-023-02462-y>.
- Quinn, Emma M., Paul Cormican, Elaine M. Kenny, Matthew Hill, Richard Anney, Michael Gill, Aiden P. Corvin, and Derek W. Morris. 2013. “Development of Strategies for SNP Detection in RNA-Seq Data: Application to Lym-

- phoblastoid Cell Lines and Evaluation Using 1000 Genomes Data.” Edited by Bernard W. Futscher. *PLoS ONE* 8 (3): e58815. <https://doi.org/10.1371/journal.pone.0058815>.
- Sansone, Susanna-Assunta, Philippe Rocca-Serra, Dawn Field, Eamonn Maguire, Chris Taylor, Oliver Hofmann, Hong Fang, et al. 2012. “Toward Interoperable Bioscience Data.” *Nature Genetics* 44 (2): 121–26. <https://doi.org/10.1038/ng.1054>.
- Sarkans, Ugis, Wah Chiu, Lucy Collinson, Michele C. Darrow, Jan Ellenberg, David Grunwald, Jean-Karim Hériché, et al. 2021. “REMBI: Recommended Metadata for Biological Images—enabling Reuse of Microscopy Data in Biology.” *Nature Methods* 18 (12): 1418–22. <https://doi.org/10.1038/s41592-021-01166-8>.
- Schmied, Christopher, Michael S. Nelson, Sergiy Avilov, Gert-Jan Bakker, Cristina Bertocchi, Johanna Bischof, Ulrike Boehm, et al. 2023. “Community-Developed Checklists for Publishing Images and Image Analyses.” *Nature Methods*, September. <https://doi.org/10.1038/s41592-023-01987-9>.
- Shotton, David. 2010. “CiTO, the Citation Typing Ontology.” *Journal of Biomedical Semantics* 1 (Suppl 1): S6. <https://doi.org/10.1186/2041-1480-1-s1-s6>.
- Spidlen, Josef, Karin Breuer, and Ryan Brinkman. 2012. “Preparing a Minimum Information about a Flow Cytometry Experiment (MIFlowCyt) Compliant Manuscript Using the International Society for Advancement of Cytometry (ISAC) FCS File Repository (FlowRepository.org).” *Current Protocols in Cytometry* 61 (1). <https://doi.org/10.1002/0471142956.cy1018s61>.
- Swedlow, Jason R., Pasi Kankaanpää, Ugis Sarkans, Wojtek Goscinski, Graham Galloway, Leonel Malacrida, Ryan P. Sullivan, et al. 2021. “A Global View of Standards for Open Image Data Formats and Repositories.” *Nature Methods* 18 (12): 1440–46. <https://doi.org/10.1038/s41592-021-01113-7>.
- W. Gehl, Robert, and Diana Zulli. 2022. “The Digital Covenant: Non-Centralized Platform Governance on the Mastodon Social Network.” *Hcommons*. <https://doi.org/10.17613/1B0D-KB17>.
- Willighagen, Egon. 2020. “Adoption of the Citation Typing Ontology by the Journal of Cheminformatics.” *Journal of*

*Cheminformatics* 12 (1). <https://doi.org/10.1186/s13321-020-00448-1>.