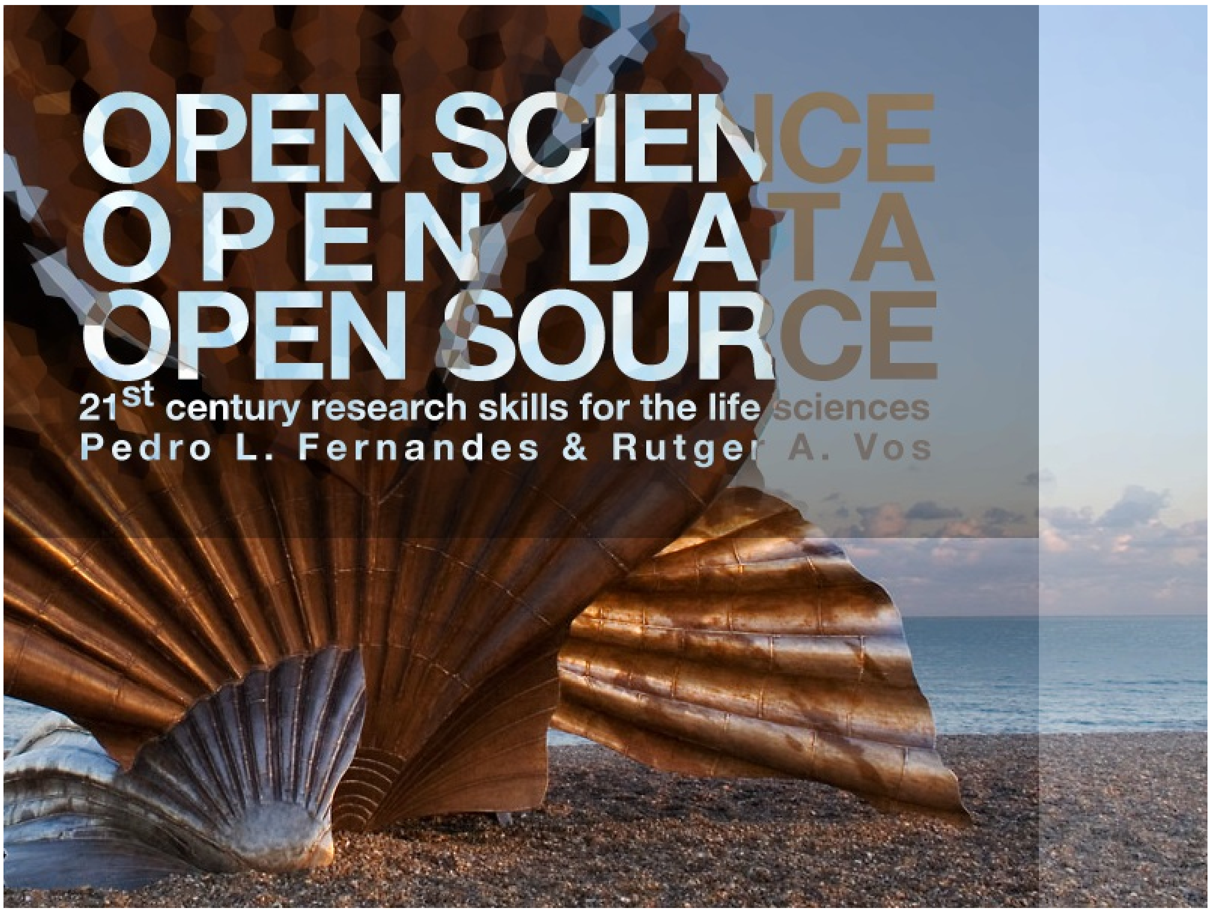
Table of Contents


OPEN SCIENCE
OPEN DATA
OPEN SOURCE
21st century research skills for the life sciences
Pedro L. Fernandes & Rutger A. Vos

Version history

DOI 10.5281/zenodo.1015288

Figure: 10.5281/zenodo.1015288

Version	Date	DOI
v1.0.0	2017-10-27	

Disclaimer

Naturalis Biodiversity Center

INSTITUTO GULBENKIAN DE CIÊNCIA

Who is this for?

The goal of these resources is to give a bird's eye view of the developments in open scientific research. That is, we cover both social developments (e.g. the culture in various communities) as well as technological ones. As such, no part of the contents are especially in-depth or geared towards advanced users of specific practices or tools. Nevertheless, certain sections are more relevant to some people than to others. Specifically:

- The most interesting sections for **Graduate students** will be about navigating the literature, managing evolving projects, and publishing and reviewing.
- **Lab technicians** may derive the most benefit from the sections about capturing data, working with reproducibility in mind and sharing data.
- For **data scientists**, the sections on organizing computational projects as workflows, managing versions of data and source code, open source software development, and data representation will be most relevant.
- **Principal investigators** may be most interested in the sections on data management, data sharing, and coping with evolving projects.
- **Scientific publishers** may be interested to know how scientists navigate the literature, what the expectations are for enhanced publications, and the needs for data publishing.
- **Science funders and policy makers** may easily find value in the capturing data, data management, data sharing and navigating the literature.
- **Science communicators** may be more interested in exploring the content by starting with navigating the literature, working with reproducibility in mind and sharing data.

How to navigate the scientific record

The number of scientific journals and the papers they publish keeps growing. Because a lot of this publishing only happens online, the limits to article lengths are also being adhered to less strictly. This has led to a situation where any normal human being can no longer keep abreast of the developments in their field simply by scanning "all" the relevant journals. Which ones are relevant? Which ones are nonsensical, or predatory? How do you find what you need? How do you get access to what you need? How do you organise your own knowledge base? And how so you do this collaboratively? Here we will address these questions and the principles behind them, illustrated by some current examples.

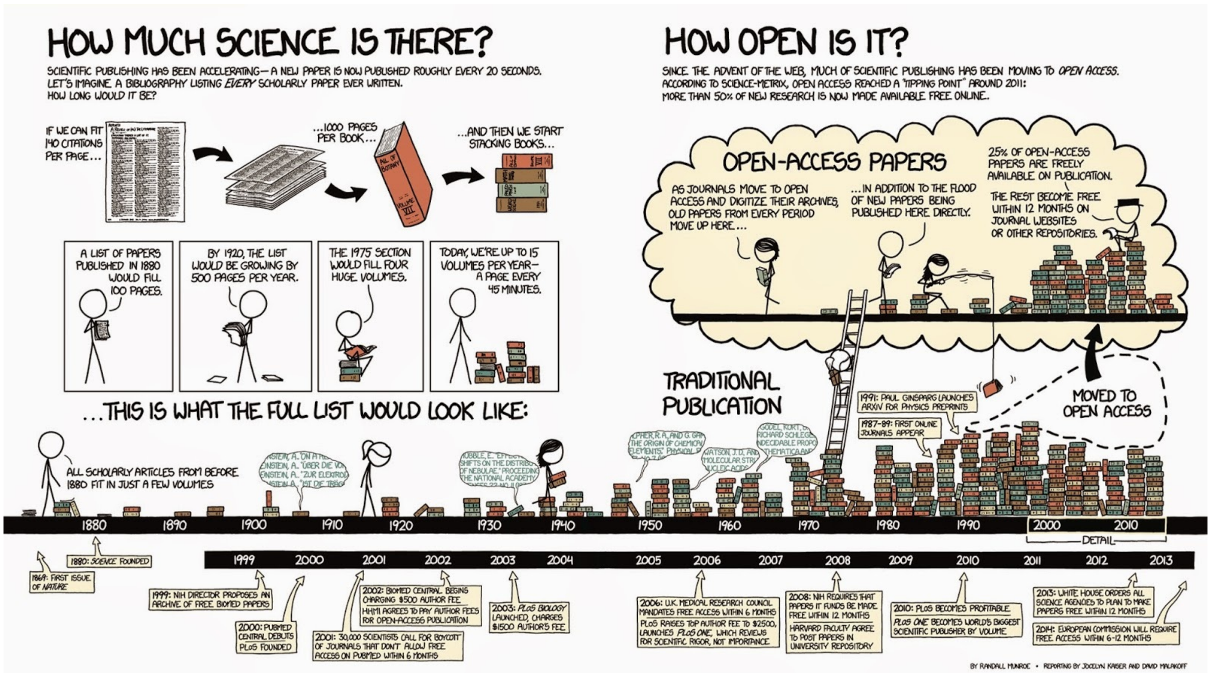
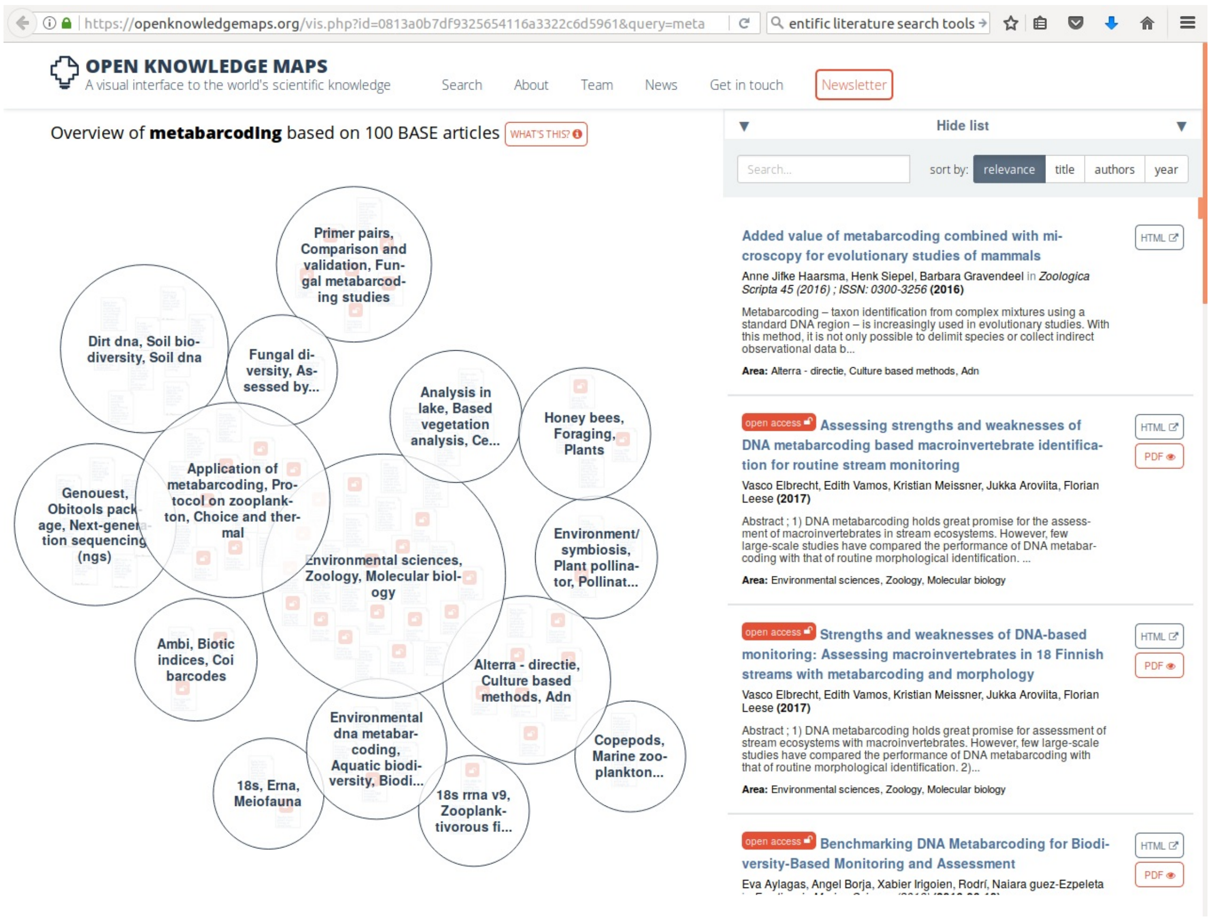

HOW MUCH SCIENCE IS THERE?
SCIENTIFIC PUBLISHING HAS BEEN ACCELERATING—A NEW PAPER IS NOW PUBLISHED ROUGHLY EVERY 20 SECONDS. LET'S IMAGINE A BIBLIOGRAPHY LISTING EVERY SCHOLARLY PAPER EVER WRITTEN. HOW LONG WOULD IT BE?
IF WE CAN FIT 140 CITATIONS PER PAGE...
...1000 PAGES PER BOOK...
...AND THEN WE START STACKING BOOKS...
A LIST OF PAPERS PUBLISHED IN 1880 WOULD FILL 100 PAGES.
BY 1920, THE LIST WOULD BE GROWING BY 500 PAGES PER YEAR.
THE 1975 SECTION WOULD FILL FOUR HUGE VOLUMES.
TODAY, WE'RE UP TO 15 VOLUMES PER YEAR— A PAGE EVERY 45 MINUTES.
...THIS IS WHAT THE FULL LIST WOULD LOOK LIKE:
ALL SCHOLARLY ARTICLES FROM BEFORE 1880 FIT IN JUST A FEW VOLUMES
HOW OPEN IS IT?
SINCE THE ADVENT OF THE WEB, MUCH OF SCIENTIFIC PUBLISHING HAS BEEN MOVING TO OPEN ACCESS. ACCORDING TO SCIENCE-METRIX, OPEN ACCESS REACHED A "TIPPING POINT" AROUND 2011: MORE THAN 50% OF NEW RESEARCH IS NOW MADE AVAILABLE FREE ONLINE.
OPEN-ACCESS PAPERS
AS JOURNALS MOVE TO OPEN ACCESS AND DIGITIZE THEIR ARCHIVES, OLD PAPERS FROM EVERY PERIOD MOVE UP HERE...
...IN ADDITION TO THE FLOOD OF NEW PAPERS BEING PUBLISHED HERE DIRECTLY.
25% OF OPEN-ACCESS PAPERS ARE FREELY AVAILABLE ON PUBLICATION.
THE REST BECOME FREE WITHIN 12 MONTHS ON JOURNAL WEBSITES OR OTHER REPOSITORIES.
TRADITIONAL PUBLICATION
1991: PAUL GINSPARG LAUNCHES ARXIV FOR PHYSICS PREPRINTS
1987-89: FIRST ONLINE JOURNALS APPEAR
MOVED TO OPEN ACCESS
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
DETAIL
1869: FIRST ISSUE OF NATURE
1880: SCIENCE FOUNDED
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
1999: NIH DIRECTOR PROPOSES AN ARCHIVE OF FREE BIOMED PAPERS
2000: PUBMED CENTRAL DEBUTS PLOS FOUNDED
2001: 30,000 SCIENTISTS CALL FOR BOYCOTT OF JOURNALS THAT DON'T ALLOW FREE ACCESS ON PUBMED WITHIN 6 MONTHS
2002: BIOMED CENTRAL BEGINS CHARGING $500 AUTHOR FEE
HHMI AGREES TO PAY AUTHOR FEES FOR OPEN-ACCESS PUBLICATION
2003: PLOS BIOLOGY LAUNCHED, CHARGES $1500 AUTHOR'S FEE
2006: U.K. MEDICAL RESEARCH COUNCIL MANDATES FREE ACCESS WITHIN 6 MONTHS
PLOS RAISES TOP AUTHOR FEE TO $2500, LAUNCHES PLOS ONE, WHICH REVIEWS FOR SCIENTIFIC RIGOR, NOT IMPORTANCE
2008: NIH REQUIRES THAT PAPERS IT FUNDS BE MADE FREE WITHIN 12 MONTHS
HARVARD FACULTY AGREE TO POST PAPERS IN UNIVERSITY REPOSITORY
2010: PLOS BECOMES PROFITABLE
PLOS ONE BECOMES WORLD'S BIGGEST SCIENTIFIC PUBLISHER BY VOLUME
2013: WHITE HOUSE ORDERS ALL SCIENCE AGENCIES TO PLAN TO MAKE PAPERS FREE WITHIN 12 MONTHS
2014: EUROPEAN COMMISSION WILL REQUIRE FREE ACCESS WITHIN 6-12 MONTHS
BY RANDALL MUNROE • REPORTING BY JOCELYN KAISER AND DAVID MALAKOFF

Figure: Randall Munroe on open access publishing

Searching the literature

When you are just getting into a new research topic, you will want to develop an overview of the different publications that speak to this topic and discover which of these are open access. One tool that you might like to know about, the open knowledge maps tool, shown in example 1 here gives an overview of the concepts related to the keyword "metabarcoding" and the 100 most relevant articles related to it, grouped by text similarity. The bubbles show the results of this clustering-by-similarity, and in each of these bubbles, the articles are shown (red icons are open access publications). As an exercise, now go to google and search with the same keyword. Notice how most of the results are not relevant papers, but any website that qualifies by google's proprietary algorithms, and no indication is given about openness.


https://openknowledgemaps.org/vis.php?id=0813a0b7df9325654116a3322c6d5961&query=meta
entific literature search tools
OPEN KNOWLEDGE MAPS
A visual interface to the world's scientific knowledge
Search
About
Team
News
Get in touch
Newsletter
Overview of **metabarcoding** based on 100 BASE articles
WHAT'S THIS?
Primer pairs, Comparison and validation, Fungal metabarcoding studies
Dirt dna, Soil biodiversity, Soil dna
Fungal diversity, Assessed by...
Analysis in lake, Based vegetation analysis, Ce...
Honey bees, Foraging, Plants
Genouest, Obitools package, Next-generation sequencing (ngs)
Application of metabarcoding, Protocol on zooplankton, Choice and thermal
Environment/ symbiosis, Plant pollinator, Pollinat...
Environmental sciences, Zoology, Molecular biology
Ambi, Biotic indices, Coi barcodes
Alterra - directie, Culture based methods, Adn
Environmental dna metabarcoding, Aquatic biodiversity, Biodi...
Copepods, Marine zooplankton...
18s, Erna, Meiofauna
18s rrna v9, Zooplanktivorous fi...
Hide list
Search...
sort by:
relevance
title
authors
year
HTML
PDF
open access

Added value of metabarcoding combined with microscopy for evolutionary studies of mammals
Anne Jifke Haarsma, Henk Siepel, Barbara Gravendeel in *Zoologica Scripta 45 (2016) ; ISSN: 0300-3256* **(2016)**
Metabarcoding – taxon identification from complex mixtures using a standard DNA region – is increasingly used in evolutionary studies. With this method, it is not only possible to delimit species or collect indirect observational data b...
Area: Alterra - directie, Culture based methods, Adn

open access **Assessing strengths and weaknesses of DNA metabarcoding based macroinvertebrate identification for routine stream monitoring**
Vasco Elbrecht, Edith Vamos, Kristian Meissner, Jukka Aroviita, Florian Leese **(2017)**
Abstract ; 1) DNA metabarcoding holds great promise for the assessment of macroinvertebrates in stream ecosystems. However, few large-scale studies have compared the performance of DNA metabarcoding with that of routine morphological identification. ...
Area: Environmental sciences, Zoology, Molecular biology

open access **Strengths and weaknesses of DNA-based monitoring: Assessing macroinvertebrates in 18 Finnish streams with metabarcoding and morphology**
Vasco Elbrecht, Edith Vamos, Kristian Meissner, Jukka Aroviita, Florian Leese **(2017)**
Abstract ; 1) DNA metabarcoding holds great promise for assessment of stream ecosystems with macroinvertebrates. However, few large-scale studies have compared the performance of DNA metabarcoding with that of routine morphological identification. 2)...
Area: Environmental sciences, Zoology, Molecular biology

open access **Benchmarking DNA Metabarcoding for Biodiversity-Based Monitoring and Assessment**
Eva Aylagas, Angel Borja, Xabier Irigoien, Rodrí, Naiara guez-Ezpeleta

Example 1 (ls1) - querying the open knowledge map

The example of the Open Knowledge Maps tool showed a very powerful technique that did not require you to have a great deal of rigour in composing your search terms: the tool did it for you. Under some circumstances, for example once you have learned more about the terminology used in a particular field, you may instead want to be more precise and specific and use commonly agreed-upon terms with an accepted meaning. One example of this is in the medical field, where so-called MeSH (Medical Subject Headings) terms are commonly used in composing search queries in the medical literature.

Medical Subject Headings (MeSH)

MeSH terms are defined in a controlled vocabulary and their meanings organised by the higher concepts they belong to. For example, by perusing the concept hierarchy for the MeSH term DNA Barcoding, Taxonomic, we discover there are actually two meanings:

1. a technique for typing organisms by matching against a reference database of marker sequences, or
2. the lab technique where such a specific marker (e.g. COI) is amplified and sequenced

If, for example, we only wanted to retrieve publications from pubmed.gov that match the first definition, we would compose the query as follows:

```
"dna barcoding, taxonomic"[MeSH Terms] "Sequence Analysis, DNA"[Mesh:NoExp]
```

This query gives, at time of writing, only 537 high-relevance results, whereas searching for `dna barcoding` yields 3278. Among these results is, for example, a paper about the application of DNA barcoding to faecal samples of birds. If we click on that hit we are taken to a page that displays the abstract.

Retrieving papers in PDF

Assuming that we are interested in the aforementioned paper, we might want to follow up and go to the publisher website, whose link is displayed in the top right of the page. However, depending on where you currently are (e.g. you might be at home, where you don't have institutional access) you might not be able to retrieve the PDF, which is reserved to subscribers, institutional or otherwise. To circumvent this issue, as of time of writing, a perfectly legal browser plugin has become very popular: unpaywall. This plugin exploits the fact that certain people (such as the authors of a publication) are generally acting within their rights if they make a PDF of their paper available elsewhere. The plugin discovers these alternative locations and re-directs you to it.

If you haven't already, please install the unpaywall plugin that matches your browser, and then click on the publishers link on the top right of the pubmed entry. You should now see a green, open padlock icon somewhere on the screen (precisely where differs a little between browsers). If you click on it, the PDF file will be downloaded. Magic!

Another browser add-on that may help you locate a PDF version of an article is the google scholar button, which lets you use a highlight phrase as a search term and then locate all the versions of a matching publication that are on the web. This often helps to find PDF versions that authors have posted, for example, on ResearchGate.

Managing your literature

Now that you have started locating papers and downloading PDFs you might be interested in organising a local repository. It is no good to simply dump all PDFs in a folder and write down literature references in a list by hand: there are much better ways to manage this information, that also give you the possibility of sharing. Various tools exist for this; some of the commonly used ones are:

- Mendeley
- Zotero
- Papers
- EndNote

At present, the most popular of these is probably Mendeley. It has several virtues: similar to the unpaywall bookmarklet, there is also a mendeley bookmarklet that allows you to import a citation entry from a variety of search engines (such as pubmed) as well as publisher webpages with a single button press; it allows you to make shared bibliographies that you can organise collaboratively; it has a plug-in for Microsoft Word that allows you to insert literature citations directly into the manuscript that you are working on, which then will be formatted according to the selected citation style.

To make this work across computers and within research collaborations, you will need to install the Mendeley desktop application on each of the participating computers (whether your own or of a collaborator). The application will then synchronise bibliographies and PDFs across these installations on request. This way, you and your collaborators all around the world can assemble a shared collection of relevant literature.

A real-world example of literature management

The pages you are reading now were written in a simple text file syntax (Markdown) that, although it is very useful for creating websites and e-books, does not by itself support in-text citations. Nevertheless, we did have a need for citing scientific literature in these pages here and there, and we did not want to do this 'by hand'. We therefore had to come up with a workaround, which was the following:

1. We collected our references using the Mendeley browser bookmarklet, which resulted in a public library.
2. We then made sure in Mendeley Desktop that every reference has at least an author, year, title, journal, and doi. We also made sure that all authors only had initials in their first names.

3. We exported the library to a bibtex file
4. We wrote a script that converts the bibtex file to a list of references in Markdown.
5. While writing the various pages, we cited the references by their bibtex key (a unique identifier that you can make up yourself, for which we used AuthorYear), creating a clickable link to the list of references, using syntax such as: `[Zhang2014](../REFERENCES/README.md#Zhang2014)` This might seem complicated at first, but it ensured consistent formatting and linking and once we had this workflow in place it was easy to add more references and cite them.

Expected outcomes

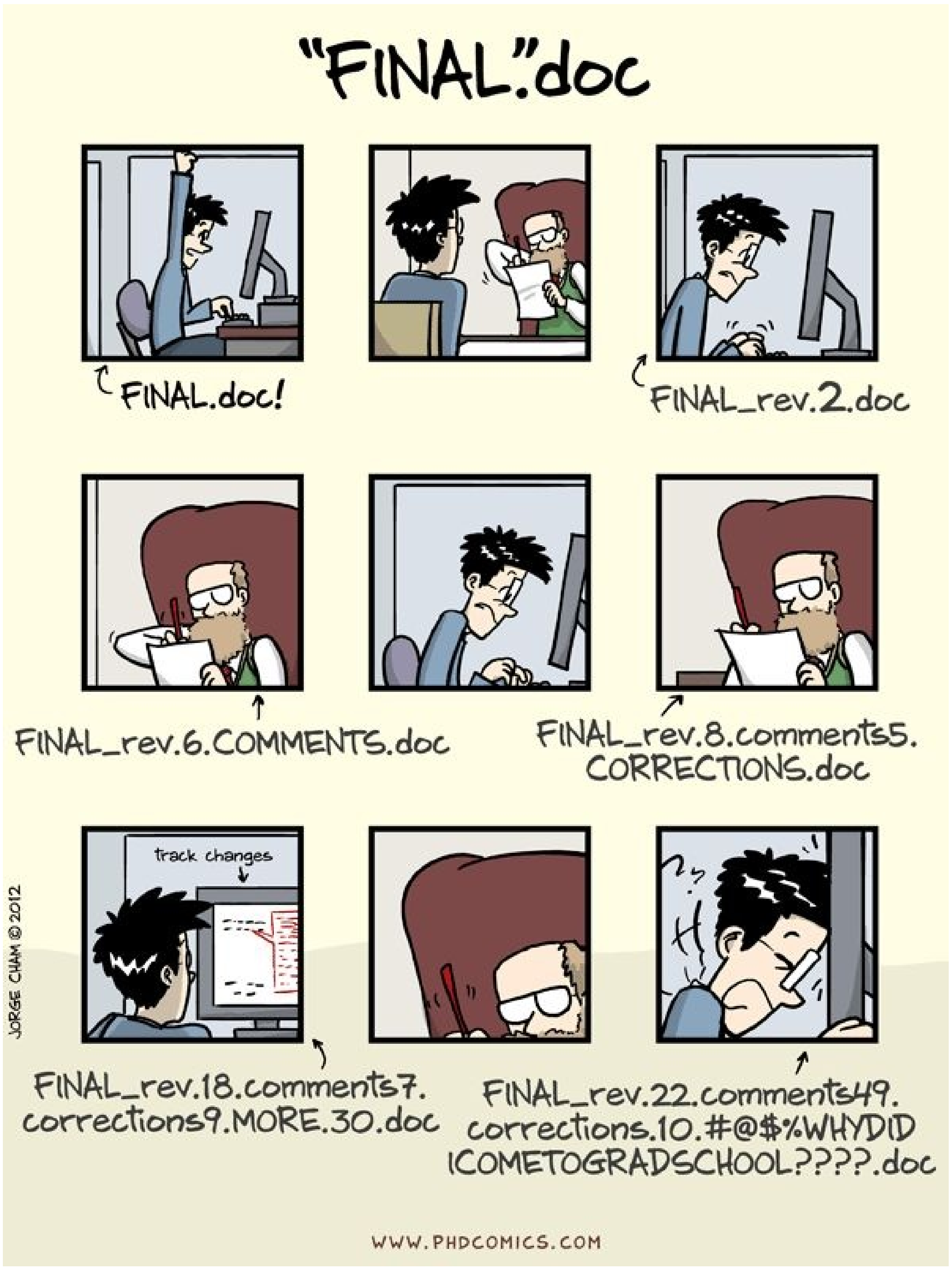
You have now had a chance to look at practical ways of exploring scientific literature. By now, you should be able to:

- Discover relevant literature in a topic, and explain how this is different from using unchecked terms
- Manage your literature in a platform
- Share literature with collaborators
- Insert citations in manuscripts
- Generate correctly formatted bibliographies
- Work around some of the limitations that emerge from mixing heterogeneous methods

How to cope with evolving research outputs

Every output of every project, whether a manuscript, some data, or an analytical workflow goes through an unpredictable number of versions. Whatever you call the "final" version never is - and you will be happiest accepting this and developing a way of working and collaborating that accommodates change at every stage while keeping specific versions of what you are working on uniquely identifiable. Here we will consider some general principles that apply to the process of version changes and identifiability in scholarly publishing, research data sets, and software code.

Manuscript versioning



Example 1 (v1) - Manuscript versioning (PhD comics)

Even before a manuscript has been submitted for its first round of peer review it has generally gone through a disheartening number of versions. If you are writing a manuscript using a word processiong program such as Microsoft Word or Open Office Write, you will want to come up with a sensible file naming scheme that is infinitely expandable and allows you to identify what the latest was that you and your collaborators were working - without temping fate by calling it `latest.doc` or `final.doc`. Instead, you probably want to adopt a system where the different versions of the manuscript are prefixed with the version's date in simple ISO8601 format (i.e. YYYY-MM-DD), so that the most recent version comes to the top of a folder when you sort file names numerically.

Subsequently, if you send a version around for review to your collaborator, it is more or less conventional to have them insert their changes using the "track changes" facility (which gives edits made by others a different colour and allows you to decide whether to accept or reject those changes) and return the file to you with a suffix to identify the collaborator by their initials. For example: `manuscript.doc` would then become `2017-06-22-manuscript.doc`, and the version returned by your contributor would be `2017-06-22-manuscript-RAV.doc`. Note that this returned version still matches the date of the version that you sent around, even though `RAV` probably made you wait for weeks. If you have just one collaborator you might then continue with the returned version, accept or reject the changes, and rename it with the date of the new version. If you have multiple collaborators, there is a facility for "merging" documents, which you will need to do iteratively starting from the version you sent out, merging the changes made by the collaborators one by one. With some luck, the merger of all collaborators still makes enough sense so that you can then work through it and accept or reject each of the suggested changes.

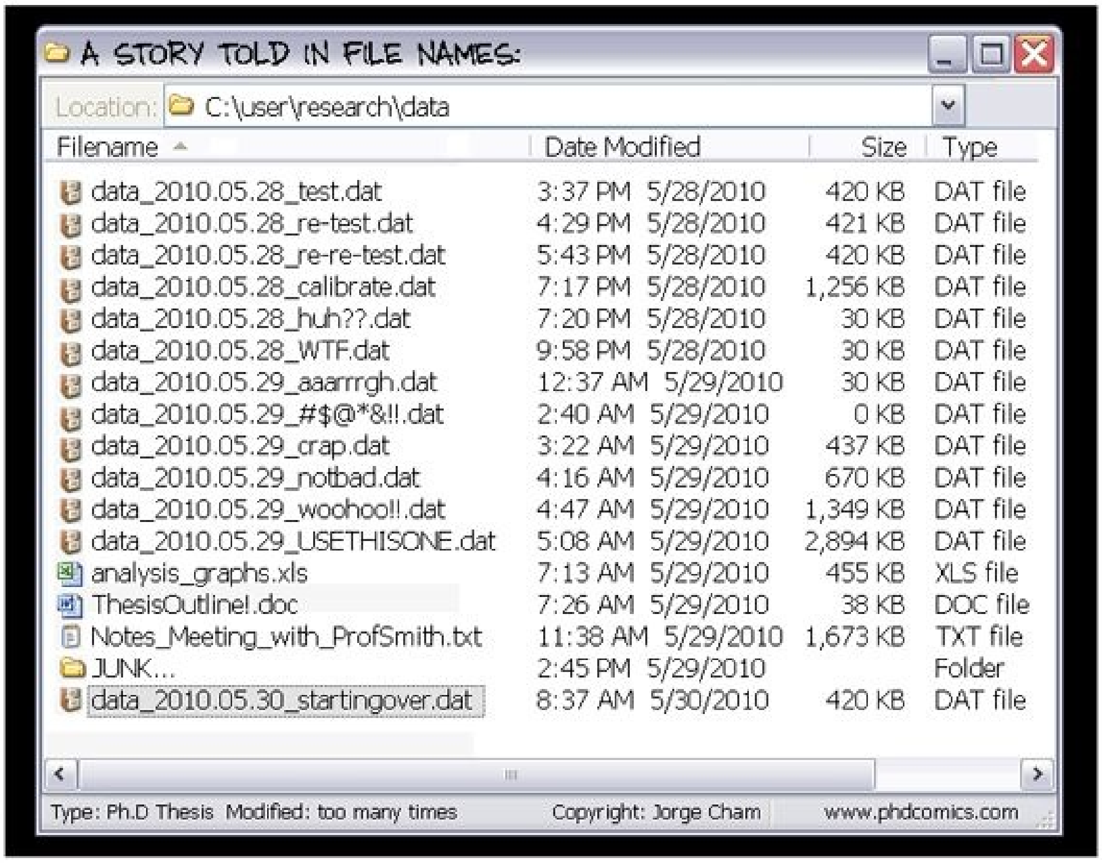
Cloud solutions for collaborative writing

Needless to say, the off-line way of managing change in a Word is not very satisfying. Numerous approaches exist for collaborating on manuscripts "in the cloud" instead, but each has its own advantages and drawbacks. At time of writing, the following approaches can be helpful at least in some stages of manuscript drafting:

- Google Docs - allows multiple people to collaborate on the same document at the same time. Each contributor can either edit directly or suggest changes that are then accepted or rejected by an editor. The main downside of Google Docs is that its manuscript editing capibilities are not sufficient for scholarly publications: there is no good way to insert citations and format bibliographies automatically (as we discuss in the section on literature study), and some of the facilities for formatting text and mathematical formulas are insufficient.
- DropBox - recently, Microsoft Word has got better at interacting with DropBox, so that manuscripts that are being edited simultaneously no longer tend to explode in a collection of mutually conflicting copies. That said, this approach still requires all collaborators to have roughly the same version of Word on their computer (a collaborator who uses OpenOffice to work on the manuscript instead will be disastrous) as well as the same reference management software.
- GitHub - the `git` protocol, discussed in more detail below, was developed for collaborative software development. The most popular website to facility this protocol is GitHub, which allows you to collaborate on any plain text file. Therefore, if you are able to agree with your collaborators on a text format that can be turned into a document format (like PDF) that may be acceptable to others, this may be a useful way of working. However, most plain text formats for editing and formatting text are either not suitable for scholarly manuscripts (for example, the MarkDown format, which was used to develop the text you are reading now, cannot handle in-text citations automatically, so we had to develop a workaround for this) or likely too complicated for some of your collaborators, like LaTeX.
- Overleaf - this is a web interface for editing LaTeX documents. It can do anything you need it to do to draft a scholarly manuscript, such as handling bibtex citations, mathematical formulas, vector drawings, complex tables, etc. The one downside is that many of your collaborators will find the syntax too daunting to deal with.

What all these systems have in common is that they have facilities for stepping through the revisions of a document, and the contributions made by collaborators, for the entire history of a manuscript. This is very hard to accomplish when you collaborate by emailing manuscript versions around.

Data versioning


A STORY TOLD IN FILE NAMES:

Location: C:\user\research\data

Filename	Date Modified	Size	Type
data_2010.05.28_test.dat	3:37 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_re-test.dat	4:29 PM 5/28/2010	421 KB	DAT file
data_2010.05.28_re-re-test.dat	5:43 PM 5/28/2010	420 KB	DAT file
data_2010.05.28_calibrate.dat	7:17 PM 5/28/2010	1,256 KB	DAT file
data_2010.05.28_huh??.dat	7:20 PM 5/28/2010	30 KB	DAT file
data_2010.05.28_WTF.dat	9:58 PM 5/28/2010	30 KB	DAT file
data_2010.05.29_aaarrrgh.dat	12:37 AM 5/29/2010	30 KB	DAT file
data_2010.05.29_#$@*&!!.dat	2:40 AM 5/29/2010	0 KB	DAT file
data_2010.05.29_crap.dat	3:22 AM 5/29/2010	437 KB	DAT file
data_2010.05.29_notbad.dat	4:16 AM 5/29/2010	670 KB	DAT file
data_2010.05.29_woohoo!!.dat	4:47 AM 5/29/2010	1,349 KB	DAT file
data_2010.05.29_USETHISONE.dat	5:08 AM 5/29/2010	2,894 KB	DAT file
analysis_graphs.xls	7:13 AM 5/29/2010	455 KB	XLS file
ThesisOutline!.doc	7:26 AM 5/29/2010	38 KB	DOC file
Notes_Meeting_with_ProfSmith.txt	11:38 AM 5/29/2010	1,673 KB	TXT file
JUNK...	2:45 PM 5/29/2010		Folder
data_2010.05.30_startingover.dat	8:37 AM 5/30/2010	420 KB	DAT file

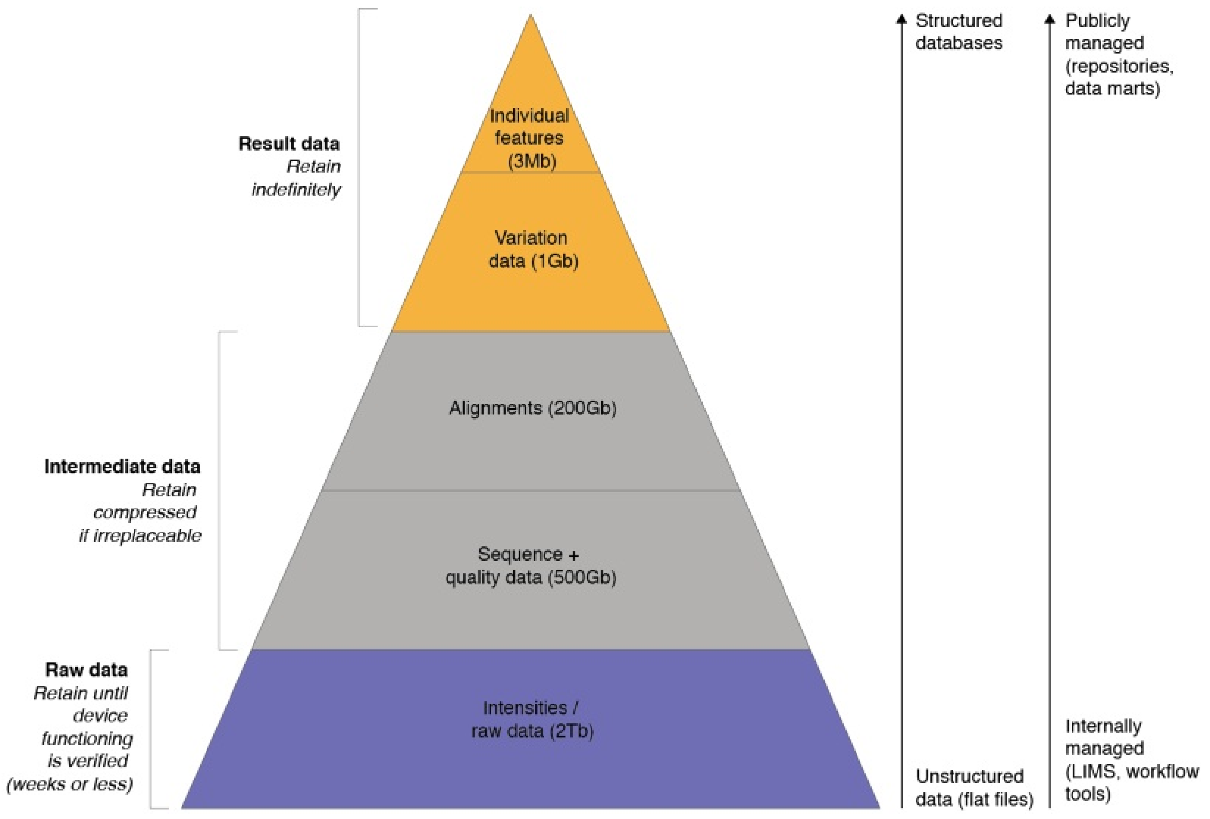
Type: Ph.D Thesis Modified: too many times Copyright: Jorge Cham www.phdcomics.com

Example 2 (v2) - Data versioning (PhD comics)

Through the course of a computational analysis, research data will go through a number of steps that might cause the data to be converted in different formats, reduced, filtered, and so on. Therefore, unlike what is shown in example 2, at least some of the principles for file naming discussed previously for manuscript versions ought to be used here as well. That said, data changes enacted by computational analytical steps are not (and should not) be acts of creation where something (like a well written, complete manuscript) grows out of nothing. In a sense, the information is already there, it just needs to be extracted out of the data.

Example 3 shows the volume reductions and types of information that are extracted and disseminated during the course of a "typical" analysis of next generation sequencing data. Here, too, a research product - in this case, sequencing reads - will go through many versions that will need to be tracked sensibly. However, these data versions should be the output of automated steps that can be re-run at will. As such, it is not the data changes themselves that are enacted by human hands, but rather, this is true of the analytical workflow, which will grow and improve over the course of the study (sometimes, a very creative process). If this is done dilligently, it should be possible to delete all but the initial, raw data to re-create everything else leading up to the result data. It should be obvious that this approach is virtuous in a number of ways:

- The reproducibility of the research output is improved and the provenance trail of the data is recorded automatically.
- The need to develop a versioning system for intermediate data is lessened. These data become, in a sense, ephemeral - because they can be re-generated.
- There is less storage space needed for different versions of intermediate data.


Result data
Retain indefinitely
Individual features (3Mb)
Variation data (1Gb)
Intermediate data
Retain compressed if irreplaceable
Alignments (200Gb)
Sequence + quality data (500Gb)
Raw data
Retain until device functioning is verified (weeks or less)
Intensities / raw data (2Tb)
Structured databases
Unstructured data (flat files)
Publicly managed (repositories, data marts)
Internally managed (LIMS, workflow tools)

Example 3 (v3) - NGS data reduction and conversion (gcoates)

Versioning public database records

Assuming sensible result data have been extracted, these will at some point be published in public repositories. Sometimes, this will be in a database for a specific research domain (like a DNA sequence database that issues public accession numbers), other times, this will be a more generic repository for research data, such as Dryad, Zenodo or FigShare. Once deposited in any of these places, data versioning becomes a much bigger issue: if something is changed about a public data record, this needs to be unambiguously clear to anyone else using these data. Therefore, all these repositories have policies for data versioning.

Ever since GenBank abandoned the GI for sequence data, their only identifier is the accession number, which is structured as a string of letters and numbers ending in a period followed by the version number. In the case of Dryad, Figshare and Zenodo, their respective versioning policies likewise state that for every change to deposited data a new identifier is minted (which, for all three of these repositories, is a DOI).

Software versioning

COMMENT
DATE
CREATED MAIN LOOP & TIMING CONTROL
14 HOURS AGO
ENABLED CONFIG FILE PARSING
9 HOURS AGO
MISC BUGFIXES
5 HOURS AGO
CODE ADDITIONS/EDITS
4 HOURS AGO
MORE CODE
4 HOURS AGO
HERE HAVE CODE
4 HOURS AGO
AAAAAAAA
3 HOURS AGO
ADKFJSLKDFJSDKLFJ
3 HOURS AGO
MY HANDS ARE TYPING WORDS
2 HOURS AGO
HAAAAAAAANDS
2 HOURS AGO
AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Example 4 (v4) - Software versioning (XKCD)

Software development is an exercise in managing complexity that expands gradually beyond the point where it still fits into any single person's head. Whereas it is unlikely that the introduction of a flawed turn of phrase in a manuscript can invalidate the whole text without being noticed, this is possible in software code. As such, changes need to be managed very carefully and need to be reversible - potentially even months down the line. To this end, numerous different version control systems have been developed.

What all version control systems have in common is that they provide an essentially unlimited undo for managed folder structures with text files in them. The, at present, most popular of these systems, `git`, was initially developed for the community process by which the kernel of the Linux operating system is modified. As such, this system has as an especially useful feature the ability to collaborate in a distributed fashion on multiple so-called "clones" of managed folder structures (often called "repositories") in such a way that the modifications made by different people can be mixed and matched intelligently.

Because `git` is an open source protocol, it can be freely adopted and facilitated by anyone that chooses to do so. The most popular service to do so is GitHub. In recent years it has gained great popularity, not just for collaboratively managing software source code versions, but also (as noted above), plain text manuscript files and small data files. In fact, the materials you are navigating now (text, images, PDFs, and so on) are also being assembled and developed collaboratively in a repository on GitHub.

As an exercise, have a look at this change to the repository of these materials. You can see the exact line difference between two versions of the document. You can also see a link to the "parent" of the current change, and if you click on that, the grandparent, and so on, all the way to the beginning of the file. These detailed, unlimited undo is just one of the advanges of using this system: the `git` protocol and the extra facilities provided by GitHub are very flexible and far-ranging. To learn more about the specific application of these tools in biology, you might find the guidelines provided in [Perez2016] useful.

Version numbers

Managing software source code (and other files) in a version control system such as `git`, and taking advantage of the additional facilities for continuous integration, automated testing, and collaborating in the open source spirit (as discussed in the section on software development) are good practices that will increase the likelihood of bringing a software project to a version that can be released. At this point, a similar need for the unambiguous and unique identification of versions as we saw in the versioning of manuscripts and data arises. It is certainly possible to use a `git` version string such as `d16e088` for this, but it will be hard to remember and not very informative.

Instead, it is more or less conventional in software development to release software with a shorter version number. This is perfectly compatible with systems such as `git`, because such version numbers can be used as aliases for the opaque version strings that `git` generates internally. One of the commonly used public version number systems for software is semantic versioning, which consists of three integers separated by periods (e.g. `1.8.23`) that are interpreted from left to right as:

1. MAJOR version number, which, when incremented, indicates that the new version is incompatible with the previous version.
2. MINOR version adds functionality in a backwards-compatible manner, and
3. PATCH version when backwards-compatible bug fixes are performed.

Whichever version numbering scheme for software is adopted, it is of vital importance in computational workflows and reproducibility that version numbers are issued consistently by software authors and recorded in sufficient detail by software users.

Expected outcomes

You have now learned about some of the considerations involved in managing changes in research output in such a way that every changed version can still be unambiguously identified. You should now be able to:

- Manage collaborative writing of manuscripts using a Word Processor and namimg files
- Make an informed choice between different cloud solutions for writing
- Explain the role of automation in managing change in data
- Know what happens to published data, and its identifier, if it is changed

How to make your research reproducible

Reproducibility is a basic requirement of any scientific endeavour. An experiment is simply invalid if another researcher can not produce (substantially) the same set of results from the same input. Anybody, in the same conditions, should be able to follow specifications and reproduce experiments and results. Likewise, experiments should be robust and perform equally well, independently of the observer. Note that this is distinct from *replication*, which might be defined as:

> The ability of a researcher to duplicate the results of a prior study if the same procedures are followed but new data are collected ([Goodman2016])

In other words, in the definitions that we adopt here (which are not necessarily the only ones out there), we *reproduce* the results of a *method*, and we *replicate* the effect of a *phenomenon*.

The reproducibility crisis and the aspects of addressing it

Currently (2016-2017) there is a declared reproducibility crisis. In the biomedical area, attempts to reproduce experiments with cancer cells, for example, have repeatedly failed (e.g. see [Mobley2013]). In consequence, some papers have had to be retracted. Several efforts have been put in place to provide systematic reproduction of experiments at various scales at the level of core facilities, laboratories, research institutes, universities and service providers. For example, since 2012, a PLoS initiative is in place to make this happen in a systematic way.

Quality assurance and control in laboratory data generation

In the scope of these materials, the concerns are naturally focused on digital data. Elsewhere, we discuss good principles and practices following data capture. However, in a laboratory that produces measurements one needs to deal with instruments that exhibit drifts and require a variety of standardisation interventions that we generically call calibratons and controls. A good laboratory service keeps a flawless track of those, and becomes capable of responding to auditing operations at any time. The whole procedure is often called quality assurance (QA).

Industrial production has needed it, in many aspects, ahead of the research world. It has requested very large contributions from statisticians and engineers. Quality assurance is strongly related to quality control (QC) but is not quite the same: QC refers to the detection and evaluation of faults or disturbances, while QA refers to the planning and execution of measures to prevent failures and to respond to their consequences. In many cases it relies on reports from QC. This why one often finds QA and QC together in the same process (QA/QC).

An interesting exploration of the meaning of the two terms can be found here. Briefly put, QA is concerned with the prevention of failures, while QC has to do with their detection.

Standardised QA/QC procedures allow for *inter-calibration*, a generic way of referring to experiments performed in reproducible circumstances in different laboratories or facilities. This is a common way of guaranteeing that quality is not assessed differently, therefore facilities can rely on quality to the point of being able to replace each other if needed, when for example there is an imbalance in measurement capacity that can be occasionally used to correct overloads of requests.

People concerned with data quality can find a lot of support from the accumulated knowledge and methodological developments in this field. Using QA/QC procedures to monitor data quality widens the comfort zone for the researcher, who needs to be concerned with the way in which experiments are planned, samples are collected and grouped, etc. Deciding on which (experimental and technical) and how many replicates are needed requires statistical skills that are very often below the required level.

Here are two tutorials that can be helpful:

- Design of Experiments (DOE)
- Promoting Responsible Scientific Research

General principles promoting reproducibility in computational research

The result of these developments is that scientists have become much more concerned about reproducibility and have tightened their controls. Scientific societies have studied ways of fighting the lack of reproducibility and issued recommendations (see, for example, the report produced by the American Academy of Microbiology). As well, teams of researchers have formulated their thoughts and documented their approaches for reproducible research. A good example to look at is the paper Ten Simple Rules for Reproducible Computational Research, which identifies rules that broadly agree with the points we raise in these materials:

1. **For Every Result, Keep Track of How It Was Produced** - a rule for which the authors emphasise the importance, as we do, of designing analyses as automated workflows
2. **Avoid Manual Data Manipulation Steps** a corollary to the first rule, i.e. automate everything. Manual data manipulation steps cloud the provenance of data
3. **Archive the Exact Versions of All External Programs Used** - as we discuss elsewhere, tracking of software versions is an important issue, including for reproducibility. As Sandve et al. also point out, this may be addressed by virtualisation
4. **Version Control All Custom Scripts** - indeed, the importance of versioning of all output can not be emphasised enough
5. **Record All Intermediate Results, When Possible in Standardized Formats** - adherence to open standards is vital in numerous contexts, as we discuss in relation to data capture, data sharing, semantics, and scientific software
6. **For Analyses That Include Randomness, Note Underlying Random Seeds** - wherein the authors again make the case for fully specified workflows, here in connection with the importance of recording all parameters
7. **Always Store Raw Data behind Plots** - another way of saying that manual data manipulation, including in the case of visualisations, must be avoided
8. **Generate Hierarchical Analysis Output, Allowing Layers of Increasing Detail to Be Inspected** - indeed, we note the need to develop an approach that allows you to drill down from the top of the data pyramid to its base in the section on publishing result data
9. **Connect Textual Statements to Underlying Results** - we suggest that one of the ways in which this can be done is by adopting an approach that encourages "literate programming"
10. **Provide Public Access to Scripts, Runs, and Results** - the time that publications could get away with vague statements that scripts are "available on request" (which are then not honoured) has passed. We strongly endorse an ethic of open, freely available source code

Example cases of reproducible research

In these pages we introduce the general principles and useful technologies for open source, open data, and open science. However, it is difficult to give one specific recipe to follow: different studies require different analytical tools, use different data types, and are performed by researchers with different ways of thinking, interests, and technological skills. In this context, an instructive collection of case studies is provided in the extensive e-book *The Practice of Reproducible Research*, which show the application of the technologies we introduce in these materials in a variety of research domains.

Expected outcomes

In this section we have discussed reproducibility in research, ranging from lab measurements to their analysis. We presented general principles and pointed you in the direction of example cases. You should now be able to:

- Articulate the difference between reproducibility and replicability
- Articulate the difference between quality assurance (QA) and quality control (QC)
- Describe the role of automated workflows in reproducibility of research

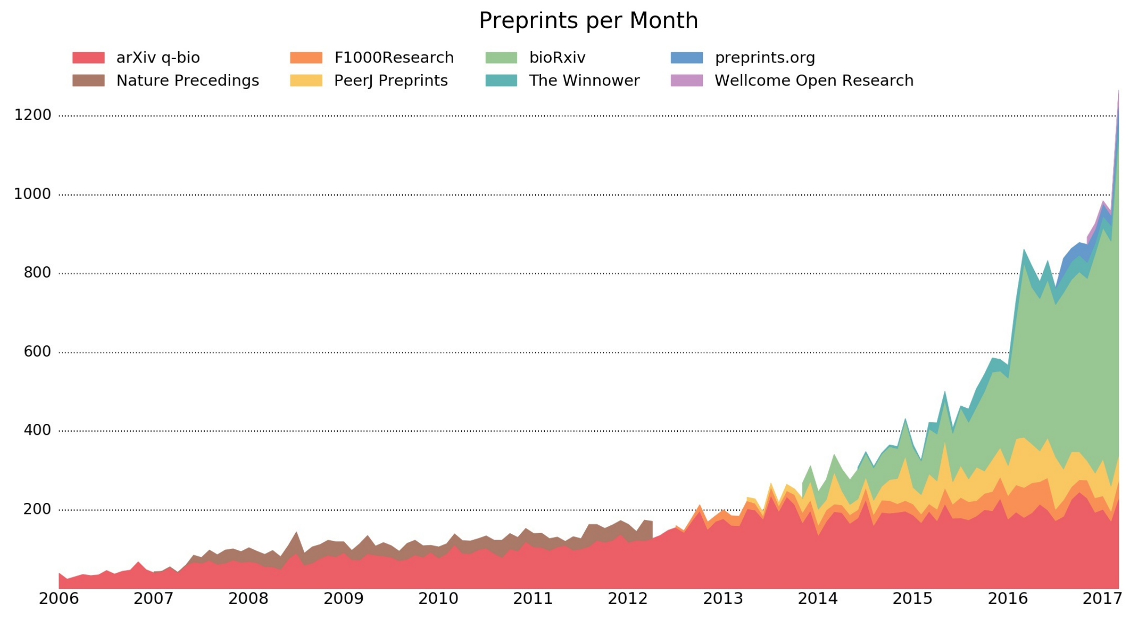
How to publish your research with impact

Elsewhere in these materials we discuss technological solutions and logical principles for how to study the scientific literature and how to edit and revise a manuscript collaboratively. The next challenges will be to write something good and publish it such that it is most likely to be read, cited, and otherwise recognised. Scientific writing is both a creative exercise in logical exposition and rhetoric as well as a highly rigid following of established rules for document structure and jargon usage. Practice makes perfect, but [Zhang2014] and [Weinberger2015] provide some useful guidelines.

The scholarly publishing cycle

Preprints

Assuming you have managed to draft a manuscript collaboratively into a state you all agree is ready to be sent out into the world, the next question is then where it will go. As the chart below shows, it is becoming increasingly common (and in more and more of the natural sciences) to send a manuscript to a preprint server, such as arXiv, PeerJ Preprints, and biorXiv.



Example 1 (p1) - Growth in preprint servers (source: Jordan Anaya)

When you upload your manuscript to such a server, it might not be type set - so this is your chance to make a pretty PDF, perhaps using an online authoring tool. What may happen is that a human on the other end will do some checks to see if what you uploaded is indeed a scholarly manuscript and that you have the right to post there (i.e. it is not copyrighted somehow) - and it will likely be assigned a digital object identifier - but it will not be peer reviewed. What, then, is the purpose of this? Here is an introductory video created by asapbio, *a scientist-driven initiative to promote the productive use of preprints in the life sciences.*

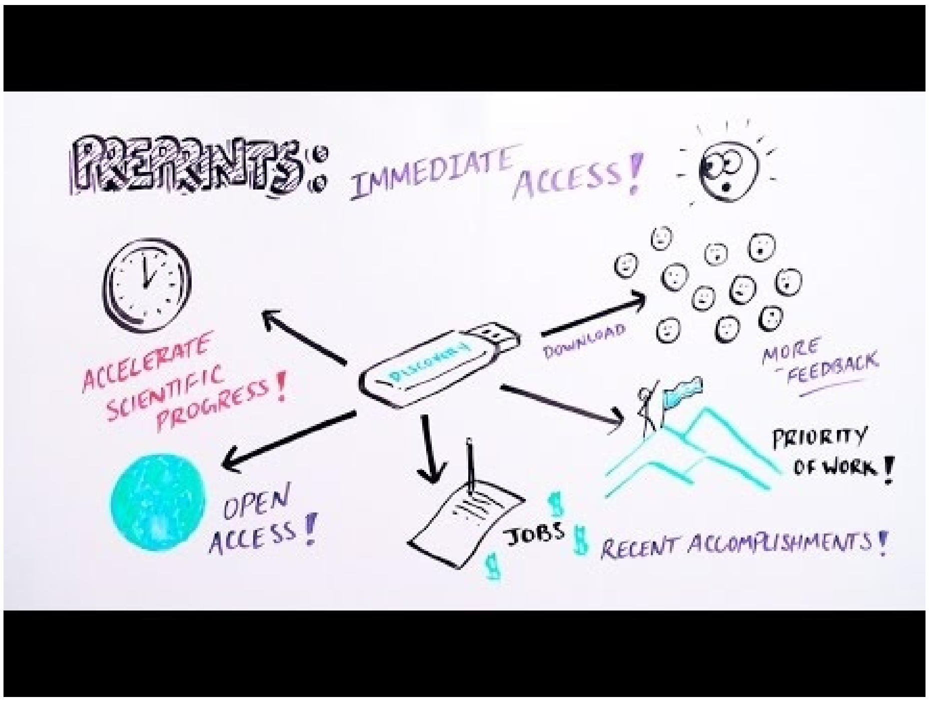


Figure: Preprints

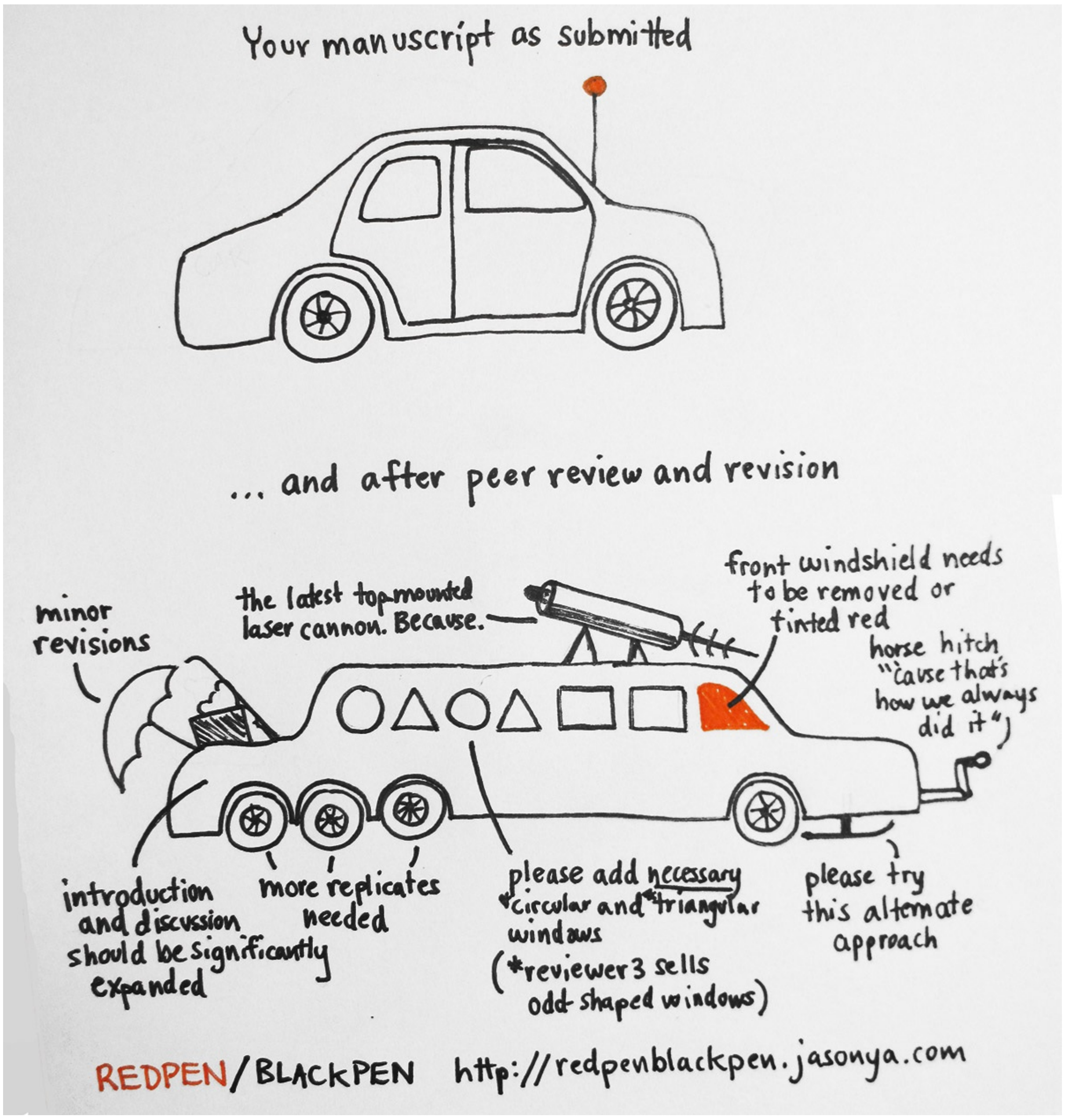
Your preprint will be the first version of your manuscript that is publicly accessible and uniquely identifiable to anyone on the web. As such, it is means by which you can circulate certain findings quickly and claim "scientific priority" while the manuscript is still working its way through peer review and various editorial steps.

Peer review

Subsequent to, perhaps, an upload to a preprint sever, you will submit your manuscript for peer review and publication in a journal. During this process you will most likely:

1. Prepare and upload a package that consists of your manuscript (double-spaced, with line numbers), the illustrations and any supplementary data in separate files, and a cover letter where you explain to the editor the importance of your manuscript for the readership of the journal you are targeting.
2. Receive reviews where two or more, usually anonymous, colleagues give feedback on your manuscript. This is usually at least one page per reviewer, consisting of remarks on the substance of your work as well as nitpicking about typos, phrasing, things you really ought to cite, and so on. These reviews will be accompanied by an editorial decision. The best you can reasonably hope for is *accept with minor revisions*, which means you probably will not have to do additional experiments or analysis, just rewriting. *Major revisions* means more work, which therefore means it will take a lot longer for the manuscript to be finally published. Because publishers want to avoid showing a long timespan between initial submission and publication (this is usually somewhere on the first page of an article) it appears to become more common to get *reject with resubmission* instead, which also means more work and a "reset" on the ticking clock from the publishers perspective. Most disheartening of all is a rejection where the editor does not want to see this manuscript resubmitted again. This is usually simply because your manuscript was not appropriate for that journal.
3. Draft your response to the reviewers and revise your manuscript. You should respond somehow to all of the remarks made by the reviewers. Sometimes the remarks will genuinely be helpful and improve your manuscript. Anything trivial requested by a reviewer you should just do outright so that you build up some credit with the final arbiter, the editor, for the parts where you may have to argue or refuse to do something: reviewers can be wrong, or unreasonable, so you might not be able to satisfy them in every respect.

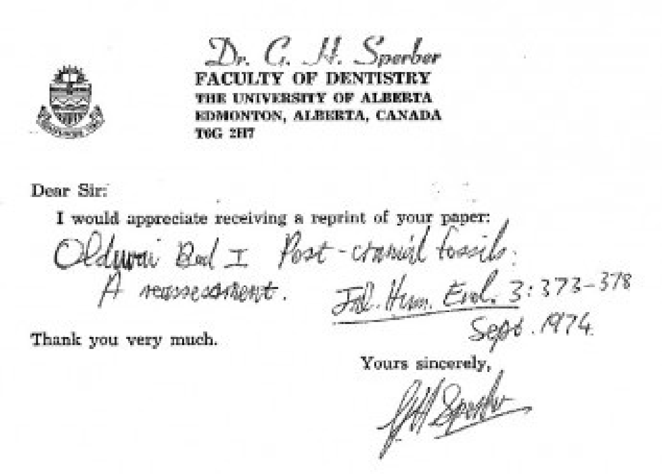
4. After one or more rounds of review, receive the final verdict. If it was accepted, you will then receive page proofs that you will need to check very carefully, because this is how your paper will look in print. You will return the annotated page proofs, and, perhaps, a signed statement where you transfer copyright to the publisher. Also, you will most likely have to pay: for colour figures and other expenses associated with making a "pretty" publication, but possibly also for *open access charges*, which are discussed in more detail in the next section.


Your manuscript as submitted
... and after peer review and revision
minor revisions
the latest top-mounted laser cannon. Because.
front windshield needs to be removed or tinted red
horse hitch "cause that's how we always did it")
introduction and discussion should be significantly expanded
more replicates needed
please add necessary *circular and *triangular windows
(*reviewer3 sells odd-shaped windows)
please try this alternate approach
REDPEN/BLACKPEN http://redpenblackpen.jasonya.com

Open access

The scholarly publishing model has undergone very little change over the last three hundred years, despite that fact that means of communication have changed enormously. Once upon a time, people had full jobs picking the letters (that is, pieces of lead) of an article one by one from wooden cases: the upper cases in front of them had the capital letters. Stacking the letters line by line and then page by page, this *type setting* was laborious and therefore costly. Then, the article was printed on paper, and this printed paper was shipped around the world. Needless to say, readers needed to pay for all this work and material. It is not so obvious why this is still the case in an age of desktop publishing and internet communication; nor is it particularly justifiable that the general public first finances the research at public institutions out of their taxes, and then has to pay again to read the results. Enter *open access* (OA) publishing.

The idea behind open access publishing is that everyone should be able to read scientific publications without hindrance. To accomplish this, there are, broadly speaking, two models. "Green" OA simply means that you, the author, often have the right to share your article with others. Even in the olden days, authors would get - or buy - a pile of "reprints" of their article that they could mail to interested readers (who would make their interest known by sending the author a special "reprint request card"). In the age of PDFs, authors can - and usually may, often after some period imposed by the publisher - upload these to their personal website, to an institutional repository, or to their web page on, for example, ResearchGate. The latter website is one of the locations where the Unpaywall plugin discussed in the section on literature study often finds free versions of articles.



Dr. G. H. Sperber
FACULTY OF DENTISTRY
THE UNIVERSITY OF ALBERTA
EDMONTON, ALBERTA, CANADA
T6G 2H7

Dear Sir:

I would appreciate receiving a reprint of your paper:
Olduvai Bed I Post-cranial fossils: A reassessment. Jnl. Hum. Evol. 3: 373-378 Sept. 1974

Thank you very much.

Yours sincerely,

[illegible]

Example 2 (p2) - Ye olden days, when scientists were always "Sirs", apparently

"Gold" OA means that the author pays the publisher to make the article immediately freely available on their website. Because publishers then lose the ability to charge readers for the article, they attempt to essentially recoup this from authors. Hence, *open access charges* usually exceed $1000.

Enhanced publications

The previous sections may have painted a somewhat negative picture of the scholarly publishing business. This does not mean that there is no useful place for publishers, and that there is nothing they can do to enhance scientific publications beyond "type setting". Here are some examples of the value that can be added to scientific articles by publishers:

- **Semantic annotations** - many scientific articles contain terms that originate in some sort of controlled vocabulary, such as gene names, species names, chemicals, and so on. Some publishers have started to provide services that automatically annotate articles with recognised terms. For example, the Biodiversity Data Journal does this for species names, as do other ARPHA journals.
- **Machine readable publications** - the language in some scientific articles is structured so rigidly that it is, or could be made, machine readable. An example of this lies in taxonomic descriptions of species (such as in floras)): in a recent study, such descriptions were read by scripts and converted to an ontology. An ongoing initiative to make taxonomic publications machine readable for such purposes is offered by Plazi.
- **Enhanced PDFs** - PDF files can contain more than just formatted text and figures: they can be enhanced with interactive visualisations (including 3D), clickable links, and so on. Some publishers (such as nature) are now offering enhanced PDFs with such interactive features.
- **Executable publications** - some articles contain a lot of computational analysis that could be presented in a re-runnable form. For example, here is a paper whose analysis can be re-run as a Galaxy workflow. In another

example, the infographic and its underlying data and analysis are available as Jupyter notebook.

Getting cited and making it count

The currency with which scientists recognise each other's work is citations. How to get cited? The best way to get cited is to do interesting, novel work. The next best way is to present your work in a clear and appealing way, using a good title, a clear abstract and the right keywords. Some content in your article may be discouraging, such as poor writing, or unnecessary but intimidating formulae. Other good ways to get cited are to make your article and any associated data and computational workflows freely available: a recent study showed that open access publications are cited more often.

A contentious issue in establishing the supposed quality of a scientist's work is the "impact factor" of the journals that the work gets published in. Although as a metric it is criticised, it does try to capture something meaningful: some journals are read and cited much more than others. Scientists want to publish in those journals, which forces the journal to become more selective, making it that much more of an accomplishment if you do make it through and get published in, say, *Nature*.

Apart from doing good, open science and publishing it in high-impact journals there are also some practical steps that you might consider to increase citations. For example, it is important that you make it clear which version of your article should be cited. The cumbersome nature of the scholarly publishing cycle, with potentially many versions of an article floating around on cloud services, preprint servers, indexing databases and publisher websites, creates the need for clear, unambiguous, globally unique identifiability of manuscripts. The DOI was invented for this purpose, and tools such as reference managers (e.g. Mendeley) make use of it to locate the "correct" metadata associated with this article. You should therefore make sure that the right DOI is used. For example, if you uploaded an earlier version to a preprint server, you should make sure that people using the DOI issued for that preprint version will be able to locate the "final" version. This might mean that the record on the preprint server needs to be updated to point to the "final" DOI.

The next step towards being credited correctly for your work is being identifiable. Citations are tracked by automated processes (such as Web of Science and Google Scholar) that scan the literature for citations to your work and compute metrics such as the H index. If these automated processes encounter multiple, different versions of your name, e.g. because you got married and took on your spouse's name, or you have middle names, then some citations might not flow towards you but to your alias. This is one of the reasons why systems such as the ORCID, which identifies researchers with an identifier that can be attached to publications, have come into usage.

Expected outcomes

In this section we discussed scientific publishing. After reading this, you should be able to:

- Outline the main steps of the scholarly publishing cycle
- Explain the purpose of preprints
- Explain the difference between "green" and "gold" open access
- Discuss "impact" in the context of scholarly publishing, and various metrics to measure it

How to capture data to produce reliable records

Capturing data in a laboratory is a daily operation that easily falls in routine chores. Something that can be a way to pile-up habits, both good and bad. Good data capture habits result in organised datasets in which one can find the means to reconstruct the capture itself, identify the instruments that were used, the reactants, the people intervening, etc. For a scientist, even if working alone, good datasets are an assurance that his/her attitude towards a rigorous relationship with the experiments started correctly. To the questions who, where, when and how (WWWH) did you get your data, one can expect to be able to answer unequivocally. That information needs to feed the annotation that is recorded together with the dataset at or near the data capture time.

It is relatively easy to produce and accumulate bad records and store bad datasets that have little or zero scientific value. Apparently good practices like keeping a lab-book, usually mandatory nowadays, are NOT a guarantee that experiments are properly recorded into datasets that allow anyone, including the original researcher, to fully address the WWWH questions, to start with. For example, a bad habit that apparently looks good is to record data in spreadsheets, occasionally print parts of spreadsheets on paper and glue tables in the lab-book. Easily the WWWH questions become unanswerable, the content of such a table becomes artificially dependent from other recorded information in the lab-book but it easily detaches from it with a simple annotation mistake. Highly dangerous, yet quite common practice. Worse than just being bad, it can "look" good, misleadingly!

Data provenance

In Data Science, the the WWWH question is usually referred to as **provenance**.

Provenance in data is tied not only to the Data Collection act but also to the its further movements between databases. Provenance may or not entail ownership and licensing of its use. If the appropriate steps, as will be explained below, are taken, authenticated provenance implies authorship. To dig further, [Buneman2000] is an interesting document about Data Provenance.

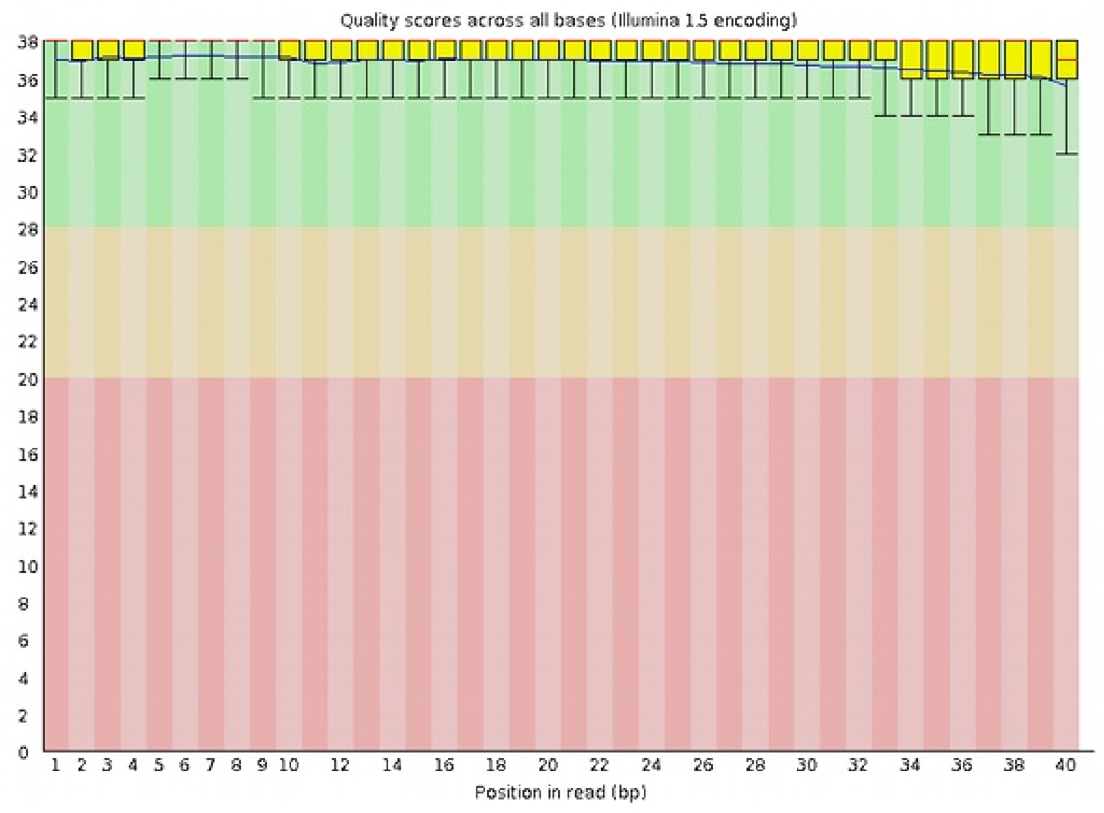
When Data Collection is performed via a service of some kind, such as the work from a Core Facility or Analytics provider, even more care must be put into fully describing the provenance. The reliability of a good provider is measurable, as described in the reproducibility part of this course. This should not be confused with the reliability of a specific set of measurements. Here the evaluation of data quality should be intrinsic to each dataset, and taken into consideration in the analytical steps that follow.

For example, in a sequencing project, the final preparation of the samples and the actual running of the sequencing job is not usually in the hands of the experimentalist, and is handled by service personnel in a facility. The results of a sequencing run are usually handed to the requester via a usually large amount of files, either in a folder deposited in a server or in a hard disk that can be shipped. Either way care is taken to use standardised formats and keep data integrity. The fist thing that the experimenter should do is assess data quality and decide on a strategy to accept or reject records and to clean up the artifacts.

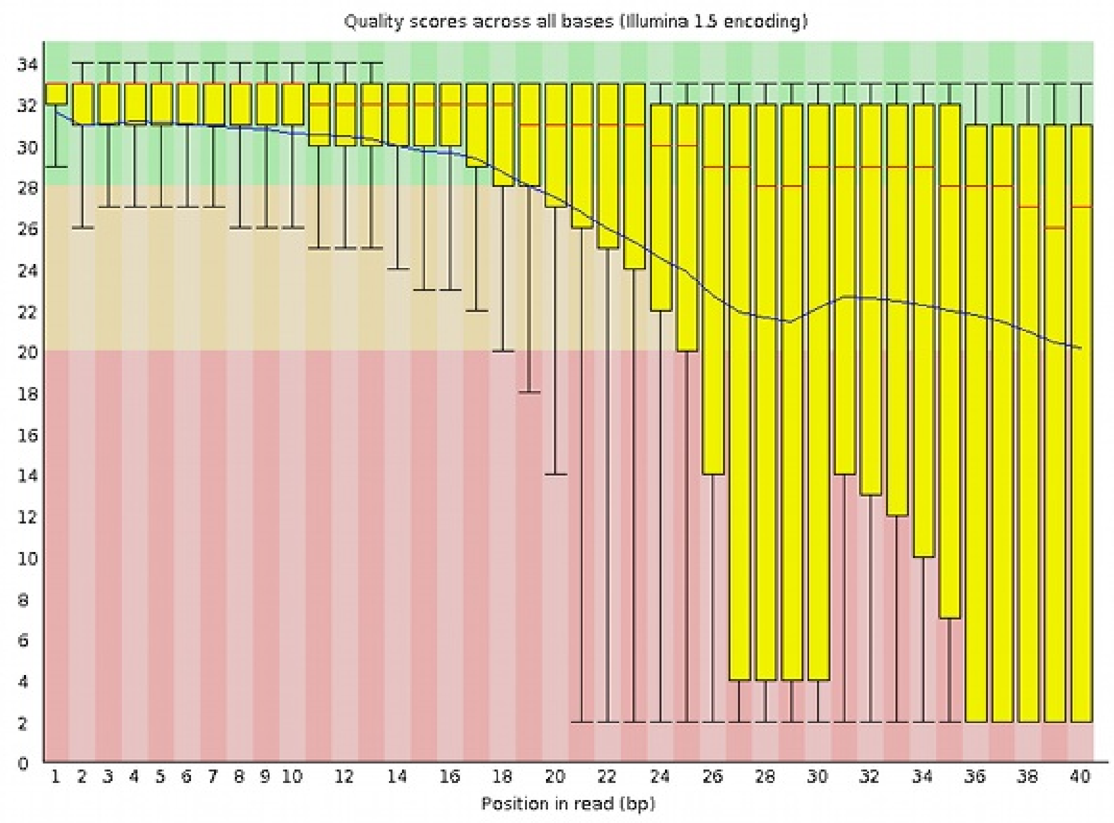
Data quality in DNA sequencing

In sequencing data, whichever instrumental platform is used, data quality will not be uniform and tends to be worse at least at the end of each read. This is the result of a natural phenomenon that is tied to the measurement process. So, removing the tails of the reads, as much as removing traces from sequencing adaptors, for example, is a normal task that lies in the hands of the experimentalist. Data from a sequencing run can be inspected with a widely used public tool called `FastQC` produced at the Babraham Institute, Cambridge, UK. A very useful tutorial is provided at: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

A very good dataset can look like this:



Whereas a very bad dataset may look like this:



A program like FastQC can look at the dataset from a variety of different perspectives, other than the above quality score and allows the experimenter to find support for accepting or rejecting a dataset and also to diagnose possible sources of disturbance, figuring out strategies to clean it up and checking for the effectiveness of such clean-up processes.

The above example shows an instance of data capture where, because we have large data sets in hands, a statistical assessment of quality may be comfortable or easy to perform. Also, this step in the provenance comes from a single machine, run by an identifiable team of professionals that, in principle, checks the entire service provision regularly for the quality of the service itself. That is obviously not the case if we are dealing with field work, for example, where samples are collected by human agents that we barely know, over a large geographical area in a long period. It is very different if samples travel at different times and via different routes. It is very different if samples are measured with different instruments with different checking and calibration routines. In such cases, finding commonality is an issue, uncontrollable sources of variation in measurements can occur, even if we are only measuring weight with a scale or length with a ruler!

Minimal information standards

In cooperative work, the need for adoption of standards and formats is even higher and simple things like the reliable time stamping of the datasets can seriously compromise the accreditation of datasets. Likewise, as the result of serious efforts in standardisation, consortia have worked on the definition of standardised ways of describing experiments, and managed to reduce their summarised information to minimum sets of descriptors. [Brazma2001] and [Taylor2007] are some examples of such specifications.

This is only the tip of the iceberg. If you continue to dig into this subject you will find that much more can be done to enhance not only the reliability, but also the traceability, for example. In any case, always keep original data and invest on its correct annotation and storage.

Negative data

It is quite common that data that is collected from experiments is not usable to prove working hypotheses. In fact, in some cases it would just disprove them. By convention we call it **negative data**. Unfortunately there is no common agreement on what to do with it, and in most cases it gets hidden from everybody else. Yet, it is quite obvious that it could play an important role in avoiding useless experimentation and even bad hypothesis formulation. For the time being, responsible researchers should always keep this kind of data in repositories where it should be labelled as having been considered negative, and annotated just as any other dataset. Possibly awaiting reuse, negative data should not be discarded.

Expected outcomes

After exploring this module, you should be able to:

- Figure out how to collect and annotate datasets so that WWWH (who, where, when and how) questions can always be unequivocally answered.
- Document datasets so that provenance is uniquely assigned
- Assess data quality early, while it is being collected
- Locate and respect standards for minimal information about experiments
- Think about a strategy to deal with your own negative data

How to manage data and plan for it

Anyone handling data should be aware of the need to manage it. There are risks of loss or corruption, there is growth, there is the possible need to share, there is attribution, etc.

The need for data management and the need to create plans

There are several reasons why one should think that data from one or many projects can easily be at risk if not managed properly. Naturally the need to manage data is reduced if the volume of data is low, but it is also true that even small volumes of data that go unmanaged cause serious problems when there is a loss of any kind. Research Funding agencies began to ask for a data management plan in the grant applications. This has obliged researchers to at least get informed, but it is quite clear that training provision in this area is far below the present and foreseeable levels of demand.

Data from research projects

Research projects generate enormous quantities of data, every single day. In many cases it is clear that raw, unprocessed data should be kept, but in many others it is not adequate at all. In large scale sequencing, but also in radio astronomy and particle physics, for example, the volumes of raw data are often enormous and their permanent storage is unreasonable. Performing the sequence again may be a lot more adequate, if the need arises.

Permanent, long term data storage requires tailored strategical decision-making. What to store, when, where and how it is backed-up. The availability of cloud resources, where there is a physical distribution of the storage servers has brought very considerable benefits in this area. But that is the technical side of it. Often harder is the management of a team of people involved in the progress of the same project. The need for a frequently updated guided process, that takes into considerations that people exhibit very different ways of thinking, not always in the best structured way, calls for plans that are goal-directed yet very adaptable to a wide range of circumstances.

Getting to know how to write a Data Management Plan

One possible way to make some progress in increasing one's capabilities for writing data management plans is to look at well built examples to get inspired. No single plan will suit everyone's needs, but it is common practice to use a good one as a seed and modify it to one's needs. You can find some good examples here:

http://data.library.arizona.edu/data-management-plans/data-management-plan-examples

Browsing though several of these you will discover ideas that may fit your requirements, here and there.

Examples to illustrate Data Management and sharing (from NIH)

> The precise content and level of detail to be included in a data-sharing plan depends on several factors, such as whether or not the investigator is planning to share data, the size and complexity of the dataset, and the like. Below are several examples of data-sharing plans.
>
> Example 1

> The proposed research will involve a small sample (less than 20 subjects) recruited from clinical facilities in the New York City area with Williams syndrome. This rare craniofacial disorder is associated with distinguishing facial features, as well as mental retardation. Even with the removal of all identifiers, we believe that it would be difficult if not impossible to protect the identities of subjects given the physical characteristics of subjects, the type of clinical data (including imaging) that we will be collecting, and the relatively restricted area from which we are recruiting subjects. Therefore, we are not planning to share the data.
>
> Example 2
>
> The proposed research will include data from approximately 500 subjects being screened for three bacterial sexually transmitted diseases (STDs) at an inner city STD clinic. The final dataset will include self-reported demographic and behavioral data from interviews with the subjects and laboratory data from urine specimens provided. Because the STDs being studied are reportable diseases, we will be collecting identifying information. Even though the final dataset will be stripped of identifiers prior to release for sharing, we believe that there remains the possibility of deductive disclosure of subjects with unusual characteristics. Thus, we will make the data and associated documentation available to users only under a data-sharing agreement that provides for: (1) a commitment to using the data only for research purposes and not to identify any individual participant; (2) a commitment to securing the data using appropriate computer technology; and (3) a commitment to destroying or returning the data after analyses are completed.
>
> Example 3
>
> This application requests support to collect public-use data from a survey of more than 22,000 Americans over the age of 50 every 2 years. Data products from this study will be made available without cost to researchers and analysts. https://ssl.isr.umich.edu/hrs/ Link to Non-U.S. Government Site
>
> Disclaimer
>
> User registration is required in order to access or download files. As part of the registration process, users must agree to the conditions of use governing access to the public release data, including restrictions against attempting to identify study participants, destruction of the data after analyses are completed, reporting responsibilities, restrictions on redistribution of the data to third parties, and proper acknowledgement of the data resource. Registered users will receive user support, as well as information related to errors in the data, future releases, workshops, and publication lists. The information provided to users will not be used for commercial purposes, and will not be redistributed to third parties.

Useful resources about data management

- Teachers with the Data Carpentry initiative have been motivated by the above referenced concerns have worked on a set of recommendations (best practices) in a scientific paper **"Ten Simple Rules for Digital Data Storage"**.
- William K. Michener has prepared a set of recommendations in a scientific paper **"Ten Simple Rules for Creating a Good Data Management Plan"**.
- The Digital Curation Center has published online resources that provide guidance and examples on Data Management Plans for a variety of purposes on this website.
- The purpose of LEARN is to take the LERU Roadmap for Research Data produced by the League of European Research Universities (LERU) and to develop this in order to build a coordinated e-infrastructure across Europe and beyond. The "Toolkit of Best Practice for Research Data Management" associated with this initiative can be downloaded from http://learn-rdm.eu/en/dissemination/
- The FAIR principles, which we discuss elsewhere, have been adopted by the Horizon 2020 framework program. Hence, Guidelines on FAIR Data Management in Horizon 2020 have been established.
- A suitable alternative can be acquiring some online training via a simple web-based tutorial on Data Management Plans (DMP), such as this one from PennState University.

Specific issues with sensitive data

Particularly acute cases arise when we speak about sensitive data, as for example in data management plans that involve patient data, even when there is informed consent. It is not always obvious but human genetic data consisting only of stretches of sequence from a single human sample may be sufficient to uniquely identify a person. This could even be considered good, were it not for the fact that it may be used to violate individual privacy in ways that do not protect citizens.

To a large extent, the problems of lack of protection arise from the inadequacy of legislation. In general terms, there is very limited or no punishment for perpetrators that abuse the fundamental rights of individuals using sensitive data, for example non-anonimised health data used for businesses such as insurance companies, employers, etc.

Expected outcomes

After exploring this module you will:

- Be aware of the factors that justify the need for a data management plan.
- Know where to get guidelines or inspiration to create a data management plan
- Be confident about what best practices to pick from the Research Data Management resource in http://learn-rdm.eu/en/dissemination/
- Be aware of the specific difficulties that arise when sensitive data is used

How to share research data fairly and sensibly

Collaboration is a natural, essential element in research. However, sharing of resources amongst scientists should be a lot easier than it is, as it finds expected and unexpected barriers everywhere. In recent years, issues surrounding scientific data sharing have received a lot of attention (e.g. see [Gewin2016]), and this has led both to a better understanding of the right principles and practices that should surround such sharing as well as to better infrastructure.

The FAIR Guiding Principles

The principles that (should) guide scientific data sharing are abbreviated as **FAIR**, which stands for *Findable, Accessible, Interoperable, Reusable*. What is meant by this is outlined below, and discussed in much greater detail in [Wilkinson2016].

> To be **Findable**:
>
> - F1. (meta)data are assigned a globally unique and persistent identifier
> - F2. data are described with rich metadata (defined by R1 below)
> - F3. metadata clearly and explicitly include the identifier of the data it describes
> - F4. (meta)data are registered or indexed in a searchable resource
>
> To be **Accessible**:
>
> - A1. (meta)data are retrievable by their identifier using a standardised communications protocol
> - A1.1 the protocol is open, free, and universally implementable
> - A1.2 the protocol allows for an authentication and authorisation procedure, where necessary
> - A2. metadata are accessible, even when the data are no longer available
>
> To be **Interoperable**:
>
> - I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
> - I2. (meta)data use vocabularies that follow FAIR principles
> - I3. (meta)data include qualified references to other (meta)data
>
> To be **Reusable**:
>
> - R1. meta(data) are richly described with a plurality of accurate and relevant attributes
> - R1.1. (meta)data are released with a clear and accessible data usage license
> - R1.2. (meta)data are associated with detailed provenance
> - R1.3. (meta)data meet domain-relevant community standards

Many different standards, databases, and policies can be adopted and combined to develop practices that comply with these general principles. These are collected on FAIRSharing, which also contains an extensive educational section.

Assuming that we are persuaded by the wisdom of these FAIR principles, we may want to adopt them in our own research when it comes to sharing our data in various contexts. Among these contexts we can at least distinguish between the sharing of raw and intermediate data (for example, within a collaborative network) and the publishing of our "final", result data, i.e. when we finished with our analyses and want to share our conclusions with the world.

In either case, many of the FAIR principles can be implemented following guidelines we establish elsewhere on these pages. For example, to be properly **Findable**, our data should treated at least according to our suggestions in the section on versioning; and to be **Interoperable** and **Reusable** we should describe our data collection process following reporting standards such as described in the section on data capture, we should express the meaning of our data and metadata using clear, unambiguous semantics, and we should adopt open, community standards for representing our data, as we elaborate on in the section on open source.

What remains to be discussed are the options for making our data **Accessible**, and for this there are different technologies to choose from depending on whether we are sharing raw and intermediate data or whether we are publishing result data. In the following sections we will discuss these options.

Sharing raw and intermediate data

In the more private context of a research collaboration before publication it might seem that it does not matter that much how data are shared: the collaborative network is probably small enough that most people know each other and can jointly reconstruct where the data are, what has been done to it so far, and so on. However, research projects usually take longer to arrive at the final results than planned, and meanwhile some people might leave, others come in, and everyone forgets what they were exactly doing or thinking a few years ago. Even in this setting there are therefore good and bad ways to share data. Here are some of the main ways in which small research groups might share data, assessed within the context of the FAIR principles:

- **email** - For small data sets (spreadsheets, for example) it might seem easy and quick to just email them as attachments. This is a bad idea for numerous reasons: 1) an email is uniquely *not* findable. Within a group of people you would have to refer to it as, for example, "that email that I sent a few weeks ago", and this of course gets worse when other people start replying with other versions of data files, 2) there is no way to link to an email such that everyone has access to it, 3) email attachments have a size limit. **Sending important files (such as data, manuscripts, or source code) by email within a research project should be discouraged under nearly all circumstances.**
- **sneakernet** - Large data sets (such as produced by an external DNA sequencing facility) are often shipped on external hard drives, and subsequently carried around the halls of a research institution, a practice referred to as "sneakernet". This has obvious access problems (you somehow need to get access to the actual drive), but also a deeper "findability" problem: unless the team is disciplined in taking checksums of the data on the drive, there is no good way to tell whether the data have been changed or corrupted, and what the "original" data set was. **Data from portable drives need to be copied onto a networked system, have checksums verified, and then shared using one of the methods below in such a way that the drive can subsequently be discarded without problems.**
- **peer-to-peer** - Numerous technologies exist for sending data directly "peer-to-peer". An example of this that works for large blocks of data is WeTransfer. This has similar problems as with email attachments: it is practically impossible to refer unambiguously to a specific transmission (and anyone that was not among the recipients in the first place will not be able to access it). This will make it hard to track versions. In addition, peer-to-peer transfers are ephemeral (i.e. a service such as WeTransfer will remove the transferred file from its system after a certain amount of time). **Taken together, peer-to-peer file transfer of important research data is not a good approach and should be discouraged.**
- **HTTP servers** - Online sharing systems such as DropBox and Google Drive have the great virtue that there is a unique *location* that all collaborators that have been granted access can refer to when talking about the data. For most of these systems, this location is a HTTP URL, which means that the uniqueness of the location is guaranteed by how the internet (i.e. DNS) works, that analytical workflows can directly access the location (HTTP access is available in all common scripting languages and workflow systems), and that semantic web applications can make machine readable statements about the resource at the URL location. There are two main downsides: 1) given a URL for a file, it is not obvious how neighbouring files can be discovered (for example, there is no standard folder structure to navigate), 2) there is no standard way to track versions. Different commercial systems

(such as the aforementioned DropBox and Drive) have their own versioning systems, however, versions created by these systems are simply periodic snapshots without descriptive metadata (i.e. unlike a "commit" using version control systems such as `git` or `svn`). An open protocol that extends the functionality of HTTP, WebDAV, forms the basis for how version control systems communicate with remote servers. Such version control systems are nearly perfect for FAIR data sharing, except for the bandwidth and storage limitations that these systems (may) have. Two optional extensions to git, git-lfs and git-annex, have been developed to address these issues. **Data sharing based on HTTP servers, especially when enhanced by open extensions for version and access control, is well worth considering when files meaningfully and notably change through time.**
- **FTP servers** - Data sharing using FTP has the same advantage as HTTP in that it results in unambiguous locations based on DNS. In addition, it has the virtue that neighbouring locations on a server (e.g. the other files in a folder, and so on) can be navigated using a standard protocol supported by many tools. Bandwidth challenges are better addressed than in HTTP because downloads can be parallellised (trivially by opening different connections for different files, and by fetching multiple chunks of the same file in parallel). FTP is therefore one of the preferred methods for downloading large genomic data sets from NCBI. A downside is that there are no standards built on top of FTP for tracking file versions. Because FTP allows for navigating through file systems, this downside is commonly addressed by placing checksum files next to the data files. This allows for a quick check to see whether a remote file is the same version as a local copy without having to download the remote file. However, this is not sufficient for tracking entire file histories. **Data sharing based on FTP servers is well worth considering for navigating static libraries of large files.**
- **rsync servers** - RSync is an open source utility for synchronisation and incremental transfers between file systems. The advantage of this system is that libraries of large files can be kept in sync between systems, so if you are keeping sets of files where occasionally some change (such as a library of reference genomes that go through releases), this system is more efficient with bandwidth than FTP, which is why NCBI prefers it over FTP. Using RSync means that changed files will be updated (which is why a number of backup systems are based on it) but it is not a versioning system. **Data sharing using RSync is appropriate for synchronising file structures across systems, assuming that the changes in the file structures are managed by some other systems (such as release notes).**

The above solutions are well within reach of every researcher: the server technologies that are recommended are either already installed on some operating systems, or are freely available as very popular, well-documented open source software with large user communities. For larger institutions that have the ICT staffing resources to maintain more advanced data sharing infrastructures, two open source systems are worth mentioning within the context of bioinformatics applications: iRODS and Globus. What these systems add is that they can track (meta)data, and thus file versions, across distributed architectures and protocols. For example, such systems can figure out that the raw data file you are trying to fetch exists as a local copy near you so as to use bandwidth more efficiently, or allow for the discovery of related files by custom metadata that researchers attach to their project files, integrate file management in data processing workflows, and a lot more. The downside of these systems lies in the complexity of installing, configuring and maintaining these systems.

Publishing result data

The final outcomes of a data-intensive research project in terms of data are referred to as "result data". For example, for a genome re-sequencing project, these might be the specific gene variants detected in the individuals that were sequenced (e.g. see this pyramid). These are the data that scientific conclusions - such as presented in a scholarly publication - will probably be most intimately based on. (However, these cannot be decoupled from data nearer to the base of the pyramid, so developing an approach that allows you to "drill down" from the top to these lower levels is vital.) and The pressures to share result data come from more sides than in the case of raw or intermediate data:

1. Funding agencies more and more require that result data are shared. Research projects funded by such agencies probably need to submit a data management plan for approval to the agency that describes how this will be handled. (The NIH, the US funding agency for medical research, has published a list of frequently asked

questions surrounding data sharing that researchers might ask a funding agency.)
2. Many journals require that data discussed in a paper are deposited in a data repository such that the paper refers to a public, clearly identifiable, data record. (In addition to policies of individual journals, community initiatives to establish author guidelines to this end are being developed, see [Nosek2015].)
3. A publication based on open, reusable data is more likely to be cited (e.g. by 9% in the case of microarray data [Piwowar2013]), so there is also an aspect of enlightened self interest to depositing result data. Along this vein, a growing trend is towards "data publications", where the paper is mostly just an advertisement for a re-usable data set.
4. Data sets themselves might be citable, which is a vision that is being advanced, for example, by the Joint Declaration of Data Citation Principles and by infrastructural initiatives such as DataCite and the Data Citation Index.

These different pressures have created a need for online data repositories, and in response a bewildering array of choices has arisen. Broadly speaking, these can be subdivided in generic data repositories that will accept many different data types but will not process (e.g. validate, index, visualise) these in great detail, and domain-specific repositories that do a lot more with the data. The choices are discussed below.

Generic data repositories

In what concerns platforms to share various types of data, a researcher has access to a reasonable amount of choice. At present (August 2017), we invite you to explore a number of repositories. Often they are complemented with a series of services that offer some specific advantages, such as linking to publications, providing licensing options, offering digital object identifiers (DOI). They differ quite considerably in their policies, level of curation and organization methods. Specificity also occurs in the case of repositiories orientated towards certain communities of users. Such is the case of digital libraries, for example.

- **Dryad Digital Repository** "The Dryad Digital Repository is a curated resource that makes the data underlying scientific publications discoverable, freely reusable, and citable. Dryad provides a general-purpose home for a wide diversity of datatypes." http://datadryad.org/

- **FigShare** "As governments and funders of research see the benefit of open content, the creation of recommendations, mandates and enforcement of mandates are coming thick and fast. figshare has always led the way in enabling academics, publishers and institutions to easily adhere to these principles in the most intuitive and efficient manner." Mark Hahnel, Founder and CEO, figshare https://figshare.com/

- **Zenodo** "Zenodo helps researchers receive credit by making the research results citable and through OpenAIRE integrates them into existing reporting lines to funding agencies like the European Commission. Citation information is also passed to DataCite and onto the scholarly aggregators." https://zenodo.org/

- **Dataverse** "Dataverse is an open source web application to share, preserve, cite, explore, and analyze research data. It facilitates making data available to others, and allows you to replicate others' work more easily. Researchers, data authors, publishers, data distributors, and affiliated institutions all receive academic credit and web visibility." https://dataverse.org/

- **EUDAT** "EUDAT offers heterogeneous research data management services and storage resources, supporting multiple research communities as well as individuals, through a geographically distributed, resilient network distributed across 15 European nations and data is stored alongside some of Europe's most powerful supercomputers." https://eudat.eu/

- **Mendeley Data** "Mendeley Data is a secure cloud-based repository where you can store your data, ensuring it is easy to share, access and cite, wherever you are." https://data.mendeley.com/

Domain-specific repositories

In addition to the generic data repositories listed above, a very large number of domain-specific data repositories and databases exists. Such repositories accept only a limited number of data types - for example, DNA sequences - that need to be provided in specific formats and with specific metadata. This means that, probably, not all different data and results generated by a research project can be uploaded to the same repository, and that uploading is sometimes cumbersome and complicated. On the other hand domain-specific repositories can provide more services tailored to a specific data type (such as BLAST searching of uploaded DNA sequences), and perhaps do things with the data such as synthesizing it in a meta-analysis or a computed consensus. Nature publishes an online list of a variety of data repositories, while re3data provides a searchable database of repositories.

Licensing, Attribution and Openness in Data Repositories

Data made available via open repositories enables a series of attractive opportunities for researchers. First, the opportunity for attaching an attribution to the work of capturing, storing and curating data. Credit is assigned to the depositor following well established and understood rules, in general this is ensure by assigning one of the variants of the Creative Commons licensing scheme. (Refer to the equivalent session on this topic for software development to see what is done with source code.)

Data sharing in this way also enables reuse and, in particular, reprocessing. This exposure of data to other resarchers, students and the public is gaining popularity. It is obvious that others may find more from the same data by using different tools with different parameter settings and options, and it is interesting to observe that it can be used preserving attribution. On the other hand, the opportunities that arise when combining heterogeneous datasets may allow for important steps that would be much more difficult otherwise.

Expected outcomes

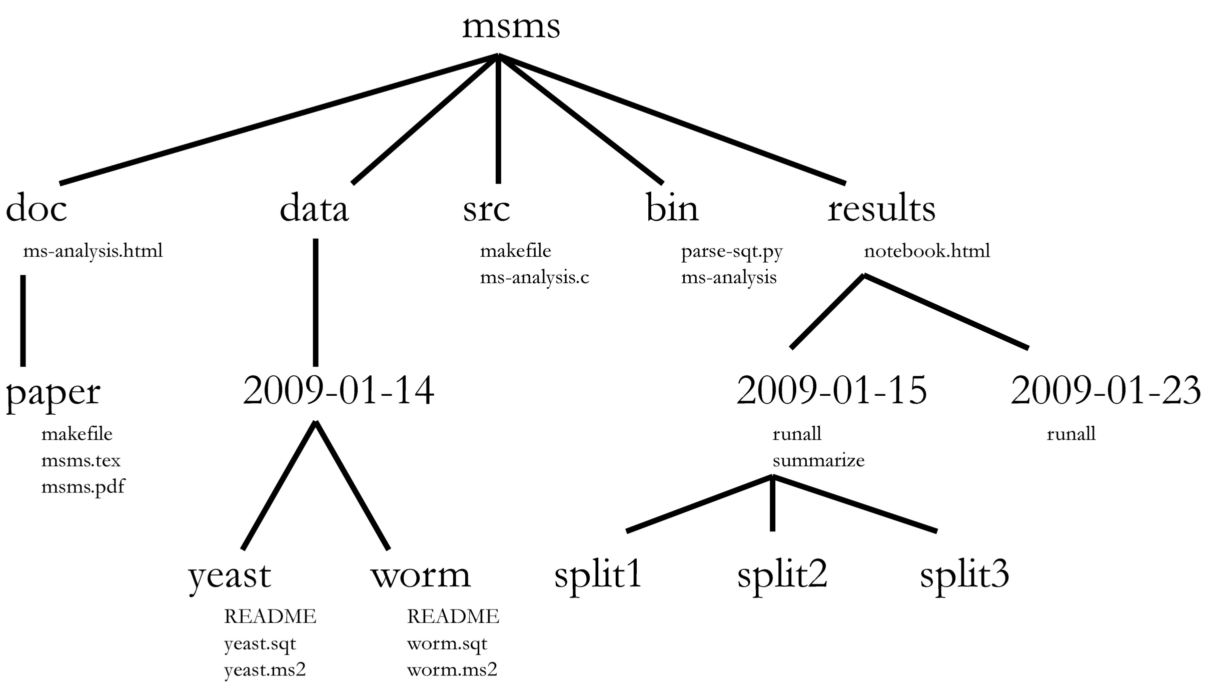
In this section we have discussed the different infrastructural solutions for sharing research data and how these relate to the developing principles for data sharing. You should now be able to:

- Explain what the terms of the **FAIR** acronym stand for in relation to data sharing
- Explain the difference between sharing raw and intermediate data, and result data
- Assess the advantages and drawbacks of different data sharing approaches within teams
- Be able to locate the appropriate domain-specific repository for a given, common research data type (e.g. DNA sequence)
- Be able to locate generic repositories for opaque research data and weigh their advantages and drawbacks against domain-specific ones

How to do analyses in a workflow-oriented manner

Virtually no research project of interest that includes *in silico* analysis only uses a single software tool, operating on a single data set, under a single set of parameters. There will be iterations over data sets, sweeps over parameter ranges, and multiple software tools will be used. These are all operations whose reproducibility is every bit as vital as that of a wet lab procedure. Therefore, the haphazard and manual chaining together of computational analyses is a recipe for failure. In addition, because research is usually exploratory such that the right approach only becomes clear after many failed attempts ("everything you do, you will probably have to do over again"), manual repetition becomes boring and error prone. Hence, computational analysis should be viewed and organised as a workflow that is automated as much as possible so that it can be re-run at will and shared with collaborators as well as the wides research community. Here we will consider some of the practical approaches and considerations in developing computational analysis workflows.

Organising a computational analysis



A computational analysis workflow chains software tools together in a series of steps that operate on data. Although each analysis will be different, some common file types (source code, compiled executables, data files, configuration files, etc.) are usually involved. Hence, a common project organisation such as shown in example 1 can probably be applied. Adopting such a scheme will result in a predictable, self-documenting structure that you can easily pick back up even if you return to a project months later. In this example, the basic layout is as follows:

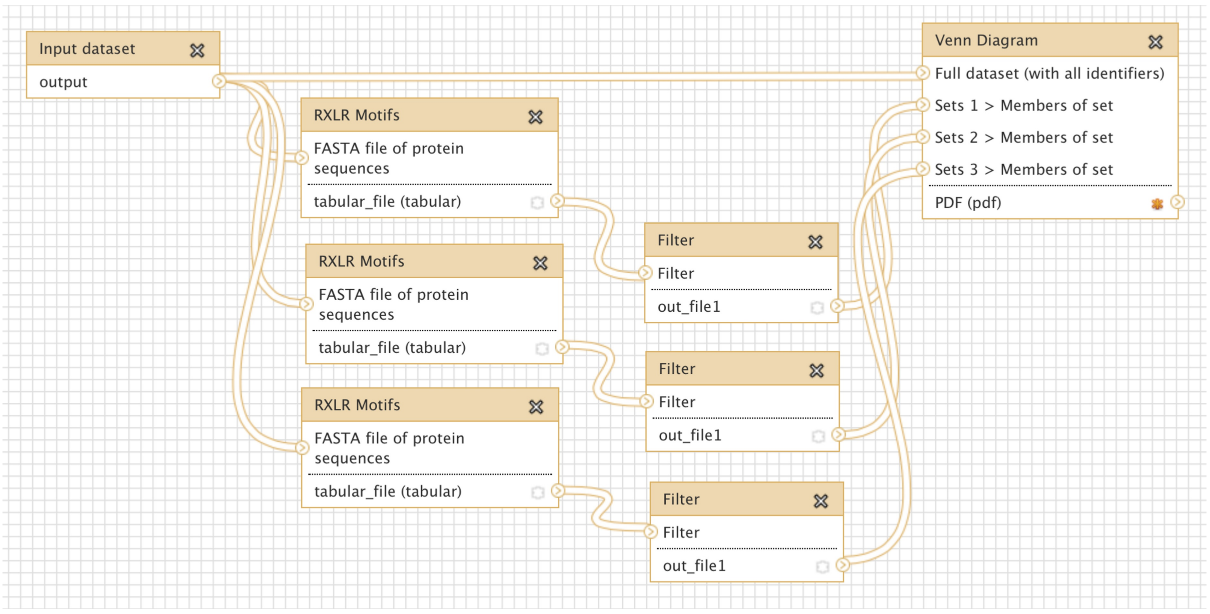
- **doc** - contains documentation files leading up to a scholarly manuscript
- **data** - data files and metadata (i.e. data about data, here as README text files)
- **src** - source code for compiled executables
- **bin** - compiled executables and scripts (i.e. the $PATH) for the workflow)
- **results** - outcomes of analysis steps

Some of these folder names match those in UNIX-like operating systems (such as Linux) and play roughly the same role. This is, of course, no coincidence - but rather a mnemonic aid. To read more about the reasoning underlying this project structure, consult [Noble2009] (where this layout came from).

Workflow tools

As we noted above, a computational analysis workflow chains tools together. Such chaining is best not done by hand because manual operations increase the chances that commands will be executed subtly different from one workflow execution to the next. For example, you might forget to set a certain parameter to the right value, or go with program defaults that make the output unpredictable from one execution to the next. The latter is for example the case with the "random number seed", i.e. the initial value for the random number generator (which is used for algorithms such as proposal mechanisms in certain Bayesian statistical analyses), which typically uses the computer's clock to generate an input value if it is not set explicitly. Hence, it is best to specify this yourself, where possible. To automate such parameter specification and chaining of workflow steps, numerous options exist. Here are some of the obvious, commonly used ones:

- **shell scripting** - on most operating systems (also on Windows), programs can be invoked on the command line shell, and these same invocations can be stored in a text file that can thus execute a series of program operations automatically. On many operating systems, shell programming is flexible enough to accommodate simple conditionals ("do this if that is true") and loops ("do this for all files"), which could easily be all you need if all the complex logic is encapsulated with the software tools you are using. A very good guide for shell scripting on OSX and UNIX-like operating systems is here.
- **make tools** - one step up in terms of syntax is to use tools like make). Originally developed for compiling software, `make` (and tools like it) also issue invocations on the command line, but they are more intelligent in handling dependencies between steps (i.e. step B can only be invoked if step A has completed successfully) and allow you to label intermediate steps so that you can run a specific command defined somewhere in a larger input file for `make`, called a `Makefile`. This allows you to work incrementally, so that the output of one step can be checked before moving on to the next.
- **scripting languages** - a number of open source, general purpose scripting languages exist that can both invoke other programs on the command line, and do complex operations on data themselves. These languages include Python, Perl and Ruby. More domain specific and originally geared towards statistics is the R programming language, which can also be used to chain software tools together. Although these languages are a departure from familiar command line syntax, they are extremely flexible, well documented, and have large, helpful user bases. It is a very useful skill to pick up at least one of them.
- **literate programming environments** - as discussed in more detail in the section on scientific software, several environments exist that allow you to mix prose, visualisations and programming logic. Although these are more specialised tools as opposed to general scripting languages, the ability to document workflow logic in prose, operate incrementally and include visualisations is extremely useful.
- **visual workflow tools** - touted as a way to build complex computational analysis workflows without having to write any code, several visual workflow tools are also available. Example 2 shows the visual workflow interface of Galaxy. The advantage of these tools is that, indeed, analysis steps can be enacted with mouse clicks. The disadvantage is that to enable this kind of "button-press bioinformatics", the underlying code to make the right invocations still needs to be written by *someone*, so these tools are not very flexible or innovative. They are very good for standard operating procedures in labs, but not as much for novel research.



Sharing and re-use

Like all the text files that you invest a lot of time in to develop, the files associated with a computational analysis workflow should be versioned. But, is a version control system such as GitHub also a useful way to share and distributed workflows? Perhaps not: workflows can have so many different dependencies that a faithful reproduction of the entire workflow environment on a new computer may be a daunting task. Consider, for example, the SUPERSMART pipeline. It depends on a multitude of tools for DNA sequence analysis and phylogenetic inference (`muscle` , `blast+` , `phyml` , `examl` , `exabayes` , `raxml` , `treepl` , `mafft` , `beast` , and a variety of packages for R and Perl) as well as more generic Linux tools (`sqlite3` , `wget` , `curl` , various build tools, and so on). Installing all of these by hand is prohibitive for most potential users. Luckily, several solutions exist to package workflows and all their dependencies for sharing with others:

- **virtualisation** - computers have grown powerful enough, and software emulation clever enough, that one or more operating systems can be run inside another: virtualisation. Using such technology, workflow developers can start from a fresh, virtualised operating system, install the entire workflow and all its dependencies on it, and then package and distributed that virtualised operating system as a disk image. The result is a rather large file (because it is an entire OS) that users can launch on their local computers as a virtual machine using tools such as VirtualBox or VMWare, or "in the cloud", e.g. on an installation of openstack maintained by an institution or at a commercial provider of cloud computing such as AWS.
- **containerisation** - a more lightweight solution for bundling tools that are otherwise difficult to install is offered by "containerisation". Here, the idea is that a lot of the standard components bundled in a virtual machine are actually redundant because the host operating system within which the virtual machine is run has these components as well. Hence, a more lightweight solution can be arrived at by only packaging the non-standard components of a workflow environment into a "container", e.g. as implemented in Docker. Note, however, that there has to be tighter relationship between the host operating system and the container (because they share more components), so this approach works best on Linux hosts. Also, there is more potential for security risks than with virtual machines, so providers of compute power (such as, for example, your institutional system administrator) might be less keen to support containers.
- **provisioning** - another solution to dependency management is to define all the dependencies and the steps to their installation in a script. For example, the SUPERSMART pipeline uses a puppet script to install all the dependencies.

Expected outcomes

You have now learned about computational analysis workflows and should be able to:

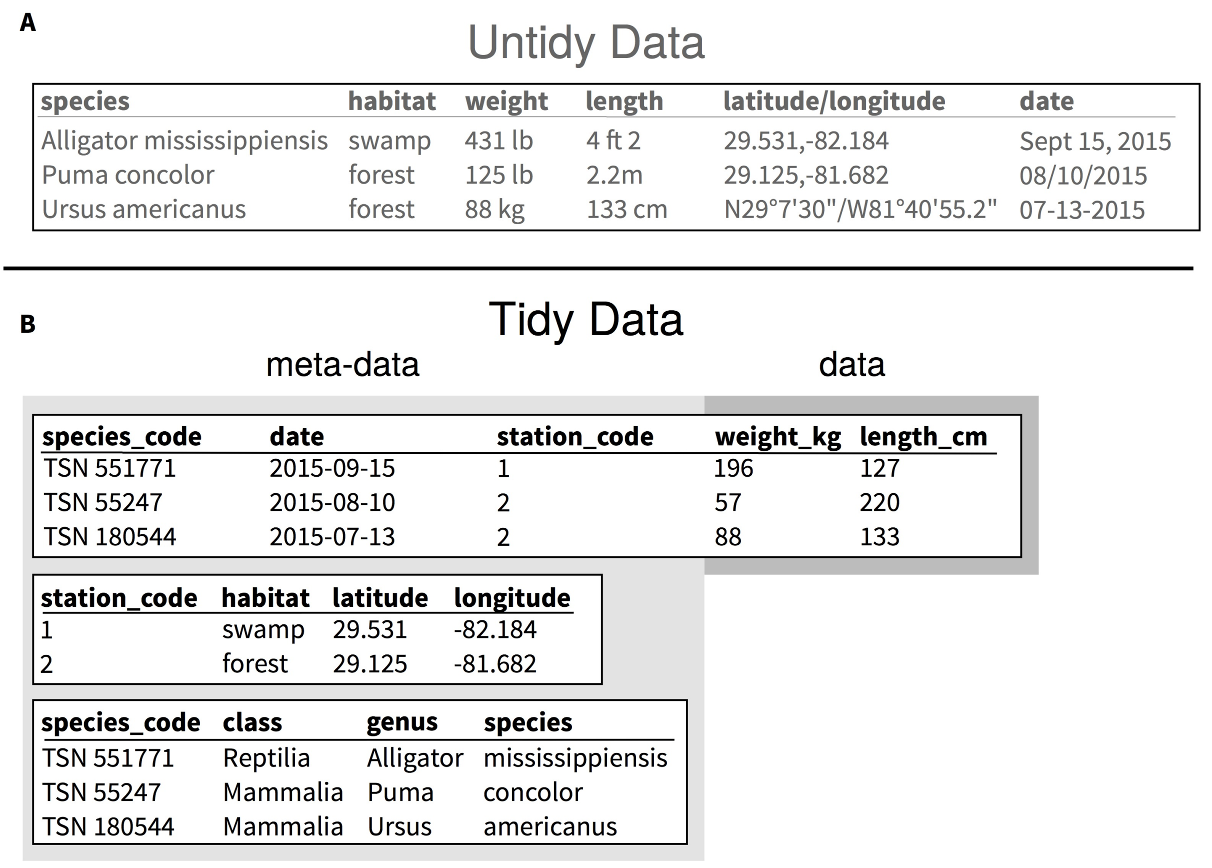
- Understand the value of consistent project organisation
- Know some of the options for automating steps in a workflow
- Be aware of the challenges and available solutions for sharing workflows

How to be formally explicit about data and concepts

In a number of locations in these pages we refer to cases where there is a common understanding among stakeholders of the meaning of certain terms. For example, in discussing community conventions in open source software development, certain specific keywords are used in package names and metadata files describing packages; in the issuing of version numbers for software, where the different parts of a version number contain embedded meaning about how one version differs from the next; in literature search, where search terms might be anchored to specific interpretations, as in the case of MeSH terms; in the description of experiments, where minimal information standards define checklists of what to report about an assay. These are all instances where a community has defined the semantics of terms, usually for the purpose of integrating information (articles, software) from different sources. In an open world where we want to share our research output with others and others to use our data in turn, similar challenges (and solutions) arise when integrating research data. Here we will discuss some of the general principles to guide is in making our data more amenable to integration.

How to structure data

Sometimes your data is captured by a machine in such a way that you might never edit it "by hand" and it is automatically structured according to community standards (such as FASTQ in the case of high-throughput DNA sequencing). In other cases, you might collect and collate data by the hand, for example in the field, or while scanning through the literature or perusing online resources. A common approach is then to enter data in a spreadsheet. In this, there are certainly tidy and untidy ways to do this.



A

Untidy Data

species	habitat	weight	length	latitude/longitude	date
Alligator mississippiensis	swamp	431 lb	4 ft 2	29.531,-82.184	Sept 15, 2015
Puma concolor	forest	125 lb	2.2m	29.125,-81.682	08/10/2015
Ursus americanus	forest	88 kg	133 cm	N29°7'30"/W81°40'55.2"	07-13-2015

B

Tidy Data

meta-data

data

species_code	date	station_code	weight_kg	length_cm
TSN 551771	2015-09-15	1	196	127
TSN 55247	2015-08-10	2	57	220
TSN 180544	2015-07-13	2	88	133

station_code	habitat	latitude	longitude
1	swamp	29.531	-82.184
2	forest	29.125	-81.682

species_code	class	genus	species
TSN 551771	Reptilia	Alligator	mississippiensis
TSN 55247	Mammalia	Puma	concolor
TSN 180544	Mammalia	Ursus	americanus

Consider example 1. *Dataset A is untidy because it mixes observational units (species, location of observations, measurements about individuals), the units are mixed and listed with the observations, more than one variable is listed (both latitude and longitude for the coordinates, and genus and species for the species names), and several* *formats are used in the same column for dates and geographic coordinates. Dataset B is an example of a tidy version of dataset A that reduces the amount of information that is duplicated in each row, limiting chances of introducing mistakes in the data. By having species in a separate table, they can be identified uniquely using the Taxonomic Serial Number (TSN) from the Integrated Taxonomic Information System (ITIS), and it makes it easy to add information about the classification of these species. It also allows researchers to edit the taxonomic information independently from the table that holds the measurements about the individuals. Unique values for each observational unit facilitate the programmatic combination of information using "join" operations. With this example, if the focus of the study for which these data were collected is based upon the size measurements of the individuals (weight and length), information about "where", "when", and "what" animals were measured can be considered metadata. Using the tidy format makes this distinction clearer.* (reproduced from [Hart2016])

Assuming you follow the advice from Hart et al. faithfully, your next decision will be what file format to store your data in. Although you might be very productive in a spreadsheet editor such as Microsoft Excel, resist the temptation to store your data in Excel's proprietary file format (`.xls` or `.xlsx`). Not only is Excel format readable by fewer programs than a plain text format such as tab-separated values, which reduces the ways in which the data can be re-used without further conversion, the risk that your data will be unintentionally altered (corrupted) by Excel is very real: gene name errors (because of this) are widespread in the scientific literature.

The point of structuring your data in "tidy" form as tabular, plain text will be immediately obvious once you (or your collaborators, or people using your data) try to analyse the data by including it, for example, in a computational analysis workflow: software such as the R environment for statistical computing, will simply refuse to read untidy data correctly, and your chosen platform for data versioning or sharing will likely also not display the data correctly.

But what does it all mean?

The advice to tidy your data and store it in an open file format is good, but it is incomplete. Whereas the change from `weight` and `length` to `weight_kg` and `length_cm` is an improvement, the encoding of the latitude and longitude coordinates in the "tidy" version still omits crucial information: geographic coordinates can be expressed in different units (e.g. in minutes and seconds, or in decimal values), and it is not made explicit which one is used, i.e. the semantics are still somewhat unclear. To address this, we might pick something like `longitude_decimal` and `latitude_decimal` , but this, like the other column headers, is actually an arbitrary choice: maybe my colleague uses `decimal_longitude` or `longitude_dec` , which would be a nightmare if we try to merge our tables automatically in a script that is not programmed to "know" that these are equivalent. We need to agree on a common terminology for this.

Common terminologies expressed as structured data in open file formats are called ontologies) and controlled vocabularies. In the life sciences, numerous of these ontologies and controlled vocabularies have been and are being developed for the jargon of different subdomains. For example, a suitable controlled vocabulary for the terms involved in recording species occurrences is the DarwinCore, which has terms for geographical coordinates in decimal form. If we choose to use these terms, we specify the vocabulary that they came from by a prefix (`dwc:` , which identifies the DarwinCore vocabulary), so that the column headers then become dwc:decimalLongitude and dwc:decimalLatitude.

Useful ontologies and controlled vocabularies

Many databases have started to adopt terms from ontologies with which to annotate records. We have already seen how databases with species occurrences such as GBIF uses DarwinCore terms. Likewise, GenBank uses the Sequence Ontology as the terminology for sequence features such as the CDS, and terms from the Gene Ontology for gene functions, such as GO:0006118 *electron transport* and GO:0005507 *copper ion binding*, in the record below. (Note, also, the versioned accession number `BT022039.1` and the way in which the taxon is anchored to an online taxonomy as taxon:3702 along similar lines as was done in the "tidy" data example.)

```
LOCUS       BT022039               536 bp    mRNA    linear   PLN 03-MAY-2005
```

```
DEFINITION  Arabidopsis thaliana At1g22480 gene, complete cds.
ACCESSION   BT022039
VERSION     BT022039.1
KEYWORDS    FLI_CDNA.
SOURCE      Arabidopsis thaliana (thale cress)
  ORGANISM  Arabidopsis thaliana
            Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
            Spermatophyta; Magnoliophyta; eudicotyledons; Gunneridae;
            Pentapetalae; rosids; malvids; Brassicales; Brassicaceae;
            Camelineae; Arabidopsis.
REFERENCE   1  (bases 1 to 536)
  AUTHORS   Cheuk,R., Chen,H., Kim,C.J., Shinn,P. and Ecker,J.R.
  TITLE     Arabidopsis ORF clones
  JOURNAL   Unpublished
REFERENCE   2  (bases 1 to 536)
  AUTHORS   Cheuk,R., Chen,H., Kim,C.J., Shinn,P. and Ecker,J.R.
  TITLE     Direct Submission
  JOURNAL   Submitted (03-MAY-2005) Salk Institute Genomic Analysis Laboratory
            (SIGnAL), Plant Biology Laboratory, The Salk Institute for
            Biological Studies, 10010 N. Torrey Pines Road, La Jolla, CA 92037,
            USA
FEATURES             Location/Qualifiers
     source          1..536
                     /organism="Arabidopsis thaliana"
                     /mol_type="mRNA"
                     /db_xref="taxon:3702"
                     /chromosome="1"
                     /clone="S80220"
                     /ecotype="Columbia"
                     /note="This clone is in pUNI 51"
     CDS             6..530
                     /note="plastocyanin-like domain-containing protein
                     go_function: copper ion binding [goid 0005507];
                     go_process: electron transport [goid 0006118]"
                     /codon_start=1
                     /product="At1g22480"
                     /protein_id="AAY25451.1"
                     /translation="MSTLLGCLVLIFSMVAQASSASLTVNWSLGTDYTPLTTGKTFSV
                     GDTIVFNYGAGHTVDEVSENDYKSCTLGNSITSDSSGTTTIALTTTGPRYFICGIPGH
                     CAAGMKLAVTVASNSSNGVAGGTTTPTPFTGGGGGYNPTTTQAIPCAAWAVSCPLRAL
                     VATWAVVFYALALS"
ORIGIN
        1 aaaatatgag cacacttctt ggttgtcttg tcctcatatt ctctatggtc gcacaggcct
       61 catccgccag tcttacggtg aactggtccc ttggcaccga ctacactccg ctcaccactg
      121 gaaagacctt ctctgtcggc gataccatag tgttcaatta tggtgcgggt cacacggtgg
      181 atgaagtgag cgagaacgac tacaagagtt gcactctagg gaactccatt acgtccgaca
      241 gcagcggaac cacgaccata gctctcacga ccactggtcc tcgctacttc atctgtggaa
      301 tccccggcca ttgcgctgcc ggtatgaagc tcgcagtcac cgtcgcgtcg aactcttcaa
      361 acggtgtagc tggtggcacc actacaccaa ccccattcac cggaggtggt ggtggttaca
      421 atcccacgac cacacaggcc attccttgtg cggcttgggc cgtgtcctgt ccattacggg
      481 ctttggttgc tacttgggcc gttgtttttt atgctttggc tttgtcttag ttgaaa
//
```

As we have seen, ontology terms can be embedded in data files in a variety of ways. Beyond the examples shown here, there are also open data formats that are explicitly designed for linking ontology terms (and data values) together in statements about the world. This is a technique that is intended to link distributed data together on the "semantic web", which, although very powerful, is not necessarily something you have to aim for if your goal is simply to adopt common terminology within existing data files. Irrespective of your specific reasons for wanting to use ontology terms, the growing number of ontologies complicates choosing the right one. It may therefore be useful to read the guidelines by [Malone2016] for this.

Expected outcomes

In this section we have discussed some of the pitfalls in assembling tidy data sets that can be shared and integrated with others. You should now:

- Be persuaded to tidily structure your data
- Know to use open, simple data formats
- Understand the purpose of ontologies and controlled vocabularies
- Know some common techniques for how ontology terms are included in data files

How to improve scientific source code development

Research in the life sciences is increasingly computational (or so says [Markowetz2017] in a somewhat controversial paper), which, because all research is about expanding what is known, means that the development and application of new computational methods is part of the field. Even if you are primarily a bench scientist or a field worker, you should have some awareness of scientific computing. How is software code written and how can you do this collaboratively? How does one use the code of others? How do you share your own? How to improve your code, and make it verifiable and testable? Here we will address these questions and some of the approaches and community standards that are in current usage.

Picking the right tools

All source code, and a lot of research data (molecular sequence data, tabular data, analysis logs, etc.), consists of plain text files. Programs that are intended for composing prose and tables for human readers (such as Microsoft Word and Excel) are wholely unsuitable for operating on such text files: they might do things such as automatically convert simple quotes to inverted ones, which might invalidate your data; convert between different, local conventions for decimal points, which are commas in some countries; attempt to run spell checks on data, which clutters the screen with useless information; attempt to export to proprietary file formats.

Text editors

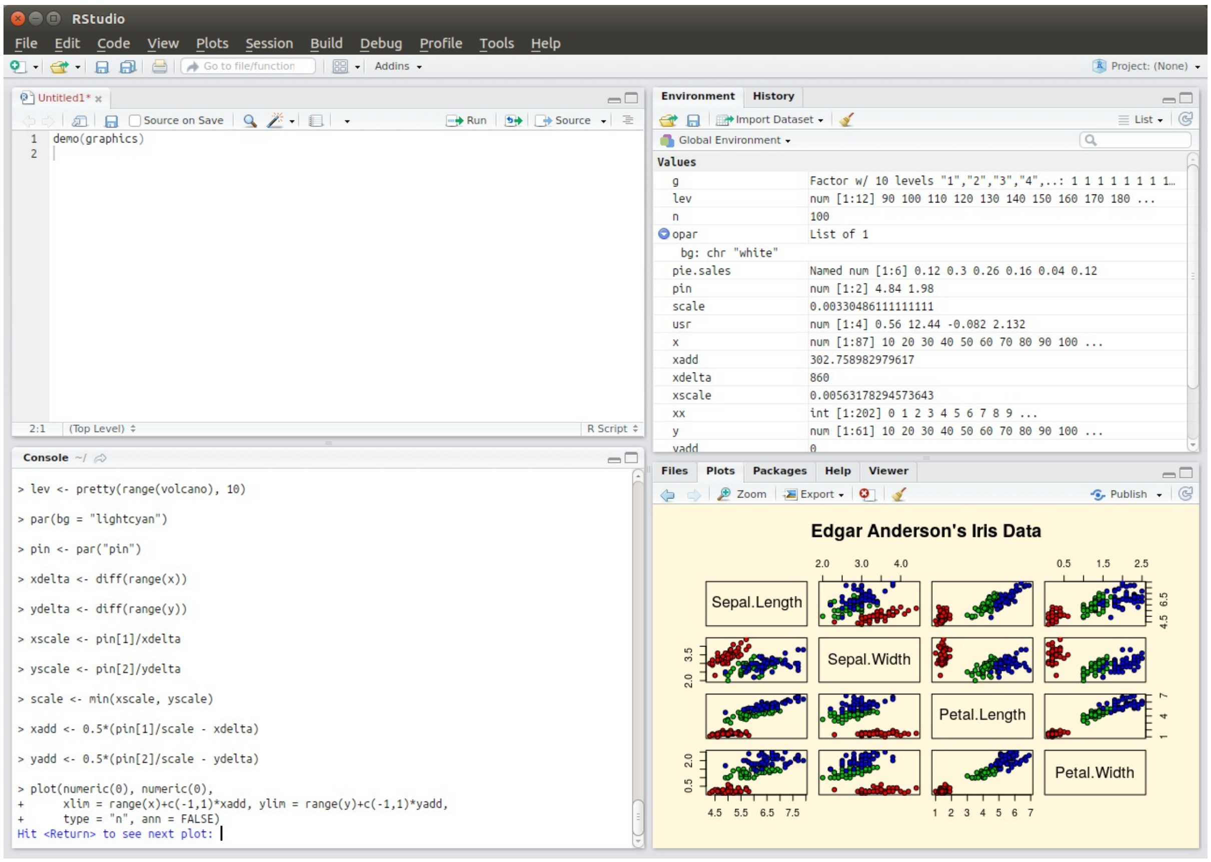
Because of the aforementioned problems with using word processors to edit plain text, the first, right tool to either locate on your computer or install if it isn't there, is a text editor. There are good, free, lightweight editors for every operating system, for example:

- On Windows there is notepad++ and editpad
- On OSX there is BBedit
- On Linux there are numerous options, gedit being a common one on GNOME, for example

Aside from plain text data files, text editors are also useful for working on source code. In many cases, a text editor will recognise the programming language (for example by the file extension of the source code file, e.g. `*.py` for python and `*.R` for R) and will colorise the syntax accordingly and allow blocks of code to be collapsed or expanded. However, for any project that comprises multiple files - and this is nearly always the case, if we consider input and output data files, configuration files, as well as source code - a text editor will not suffice. Hence, the next, right tool for the job will be an integrated development environment or IDE.

IDEs

An IDE allows you to organise sets of files into projects such that the dependencies between the files are managed. An IDE will typically have a deeper understanding of the programming language you are using, so that it may spot problematic syntax and logic errors, and may suggest functions and variables for code completion. Also, an IDE will allow you to execute your code line by line, which helps in localising problems and in stepping through an analysis workflow. Lastly, an IDE will be able to visualise different things, such as complex data structures.



```
demo(graphics)
```

```
Values
  g            Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1…
  lev          num [1:12] 90 100 110 120 130 140 150 160 170 180 ...
  n            100
  opar         List of 1
    bg: chr "white"
  pie.sales    Named num [1:6] 0.12 0.3 0.26 0.16 0.04 0.12
  pin          num [1:2] 4.84 1.98
  scale        0.00330486111111111
  usr          num [1:4] 0.56 12.44 -0.082 2.132
  x            num [1:87] 10 20 30 40 50 60 70 80 90 100 ...
  xadd         302.758982979617
  xdelta       860
  xscale       0.00563178294573643
  xx           int [1:202] 0 1 2 3 4 5 6 7 8 9 ...
  y            num [1:61] 10 20 30 40 50 60 70 80 90 100 ...
  yadd         0
```

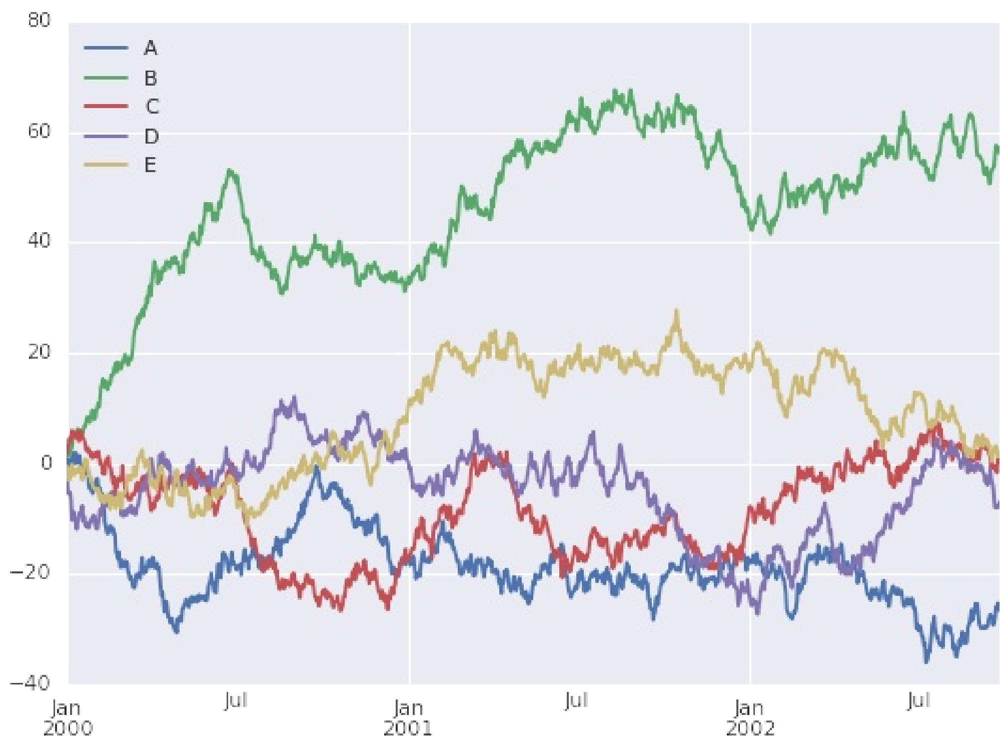
```
> lev <- pretty(range(volcano), 10)

> par(bg = "lightcyan")

> pin <- par("pin")

> xdelta <- diff(range(x))

> ydelta <- diff(range(y))

> xscale <- pin[1]/xdelta

> yscale <- pin[2]/ydelta

> scale <- min(xscale, yscale)

> xadd <- 0.5*(pin[1]/scale - xdelta)

> yadd <- 0.5*(pin[2]/scale - ydelta)

> plot(numeric(0), numeric(0),
+      xlim = range(x)+c(-1,1)*xadd, ylim = range(y)+c(-1,1)*yadd,
+      type = "n", ann = FALSE)
Hit <Return> to see next plot: 
```

Example 1 (ss1) - the graphical user interface of the most popular integrated development environment for R, RStudio. The top left pane organises files, the bottom left pane evaluates R statements line by line (e.g. to test out commands), the top right pane visualises complex data structures, and the bottom right pane allows for viewing various things such as help documentation or statistical plots.

Just like text editors, numerous IDEs exist. For most programming languages there are very good, free options. For example, for the R language for statistics there is RStudio (shown in example 1), for Python there is pycharm, for Java there is eclipse, and so on.

Literate programming

A slightly different take on source code development that is more geared towards analysis workflows than to application development is provided by the literate programming paradigm. In this way of working, source code is primarily a prose document, interspersed with bits of executable code and dynamic visualisations. This is found in R programming in the guise of RMarkdown (an example of this is the working document that formed the basis of the publication for the RNeXML library). The Jupyter system facilitates the same way of working but accommodates more programming languages, while ActivePapers has a facility for (recursive) inclusion of data from other ActivePapers, i.e. a form of citation.



Example 2 (ss2) - example output of the modified "Welcome to Python" notebook.

As an exercise of literate programming, try the Welcome to Python notebook. Modify the code to draw five (instead of four) curves, labeling the additional one `E`. An example of what the expected output might look like is shown in example 2, but keep in mind that these curves are randomly generated so they will look different every time.

Numerous learning materials for this method of programming exist on the web. As applied specifically to data science, we found the following potentially worthwhile:

- LearnDataScience
- Scikit-learn

Working with others

Like most aspects of scientific research, scientific software development is becoming increasingly collaborative, which means that developers of software code and analytical workflows are increasingly participating in open source development. There are numerous idealistic reasons for why this ought to be done (for example, because computational analysis is a research method and so should be transparent in order to be reproducible; or, because scientific software is typically funded publicly, it should be freely available) but there are also very practical, self-interested reasons for adopting open source. The main ones of these are that it allows you to build on the shoulders of others, e.g. by re-using software components developed and published by others, and that it allows you, in turn, to have greater impact with your work, because others will use it (and cite it) in turn. To participate in open source developments, here we discuss some of the main aspects to consider.

Community conventions

Every community of open source developers, whether it's a community centred around a programming language or a problem domain (like bioinformatics), has its own conventions. Some of these may be well-considered and useful, such as documentation standards, while others may be somewhat arbitrary, such as debates about what is or is not "pythonic" - while some community conventions might even be actively harmful, like the perverse pleasure in writing deliberately cryptic, obfuscated code. If you want to start contributing to a community, learn about the conventions that have been adopted, especially insofar as they affect collaboration. For example, learn what is expected of a software package that you plan to contribute: how do the files need to be organised? How does the code need to be structured? Are there specific design patterns that ought to be followed? Conventions for package names and their meanings or for the keywords to use in package descriptions?

Rights and licenses

Open source software is also referred to as "free software". This does not just mean in the sense of "free beer", i.e. at no cost, but also - more importantly - in the sense of "free speech". In other words, open source has to do with the rights of people to intellectual property. These rights are defined in software licenses, and they are relevant to developers because they both concern what you can do with the software developed by others (e.g., under what conditions, commercial or otherwise, can you re-use somebody else's source code) and what others can do with the source code that you write. Whereas the creative commons licenses are typically used for works such as images, text (including scholarly publications) and data sets, open source software is usually released under one of the licenses recognised by the open source initiative.

Responsive communication

Collaborative development and participation in a community also means to respond to feedback from others at every stage of the development cycle. When you are first planning a software tool or a computational workflow you will need to learn what your collaborators think the requirements are; when you have a prototype or an early version you may need to adjust your approach in response to early user testing; once you have released something you may need to manage and address issues reported by users.

Some of the challenges of working collaboratively can at least partly be addressed (or facilitated) by technology. Specifically, collaborating on anything that changes over time, whether a manuscript, data, or source code, can be facilitated by technologies that track version changes, a topic that is dealt with in more detail in another section.

Developing robust, verifiable software

Most scientific software is not developed by professional software engineers but by researchers. In general, such software is highly innovative in terms of the application of new analytical techniques, but also very fragile and difficult to use. Numerous specific recommendations to address these issues can be made (below, we link to two documents each with ten simple rules regarding this), but one of the key principles on all of this is the need for a structured approach to software testing using (valid and invalid) data.

In every programming language in common usage in scientific computing there are helpful tools to automate testing. What these do, in general, is to run small programs or commands that you developed in addition to the main software to test its functioning. By adopting one of these conventional testing tools, you will gain numerous advantages:

- Users can see the software in live action. This will aid them in verifying that the installation succeeded, and they will see what the inputs and outputs (in terms of data, commands, and parameters) should look like, i.e. this will make the software self-documenting.
- Sets of tests can be integrated automatically in systems that run periodically (for example, every time a version change is recorded) to verify that the system still functions as intended.
- If you make major changes in the architecture of the software you can verify automatically that this change did not break anything.

For further reading on these and related topics, you may be interested in the guidelines provided by [Taschuk2017] and [List2017].

Expected outcomes

You have now had an encounter with some of the principles, tools and techniques that play a role in scientific software development. You should now be able to:

- Understand why to use a text editor for plain text files
- Understand what the purpose is of an IDE
- Modify and execute a simple workflow
- Know some of the principles of open source development
- Know the purpose of software testing in scientific computing

References

Boland, M. R., Karczewski, K. J., Tatonetti, N. P. 2017. Ten Simple Rules to Enable Multi-site Collaborations through Data Sharing. *PLoS Computational Biology*. **13** (1): e1005278. [10.1371/journal.pcbi.1005278] `[id:Boland2017]`

Brazma, A., Hingamp, P., Quackenbush, J., Sherlock, G., Spellman, P., Stoeckert, C., Aach, J., Ansorge, W., Ball, C. A., Causton, H. C., Gaasterland, T., Glenisson, P., Holstege, F. C. P., Kim, I. F., Markowitz, V., Matese, J. C., Parkinson, H., Robinson, A., Sarkans, U., Schulze-Kremer, S., Stewart, J., Taylor, R., Vilo, J., Vingron, M. 2001. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data.. *Nature genetics*. **29** (4): 365--71. [10.1038/ng1201-365] `[id:Brazma2001]`

Buneman, P., Khanna, S., Tan, W.-C. 2000. Data Provenance: Some Basic Issues. *Lecture Notes in Computer Science: Foundations of Software Technology and Theoretical Computer Science*. **1974** : 87--93. [10.1007/3-540-44450-5_6] `[id:Buneman2000]`

Chavan, V., Penev, L. 2011. The data paper: a mechanism to incentivize data publishing in biodiversity science. *BMC Bioinformatics*. **12** (Suppl 15): S2. [10.1186/1471-2105-12-S15-S2] `[id:Chavan2011]`

Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L., Rice, P. M. 2009. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*. **38** (6): 1767--1771. [10.1093/nar/gkp1137] `[id:Cock2009]`

Gewin, V. 2016. Data sharing: An open mind on open data. *Nature*. **529** (7584): 117--119. [10.1038/nj7584-117a] `[id:Gewin2016]`

Goodman, S. N., Fanelli, D., Ioannidis, J. P. A. 2016. What does research reproducibility mean?. *Science Translational Medicine*. **8** (341): 341ps12--341ps12. [10.1126/scitranslmed.aaf5027] `[id:Goodman2016]`

Hart, E. M., Barmby, P., LeBauer, D., Michonneau, F., Mount, S., Mulrooney, P., Poisot, T., Woo, K. H., Zimmerman, N. B., Hollister, J. W. 2016. Ten Simple Rules for Digital Data Storage. *PLoS Computational Biology*. **12** (10): e1005097. [10.1371/journal.pcbi.1005097] `[id:Hart2016]`

Hoehndorf, R., Alshahrani, M., Gkoutos, G. V., Gosline, G., Groom, Q., Hamann, T., Kattge, J., de Oliveira, S. M., Schmidt, M., Sierra, S., Smets, E., Vos, R. A., Weiland, C. 2016. The flora phenotype ontology (FLOPO): tool for integrating morphological traits and phenotypes of vascular plants. *Journal of Biomedical Semantics*. **7** (1): 65. [10.1186/s13326-016-0107-8] `[id:Hoehndorf2016]`

Lariviere, V., Kiermer, V., MacCallum, C. J., McNutt, M., Patterson, M., Pulverer, B., Swaminathan, S., Taylor, S., Curry, S. 2016. A simple proposal for the publication of journal citation distributions. *bioRxiv*. 062109. [10.1101/062109] `[id:Lariviere2016]`

List, M., Ebert, P., Albrecht, F. 2017. Ten Simple Rules for Developing Usable Software in Computational Biology. *PLoS Computational Biology*. **13** (1): e1005265. [10.1371/journal.pcbi.1005265] `[id:List2017]`

Malone, J., Stevens, R., Jupp, S., Hancocks, T., Parkinson, H., Brooksbank, C. 2016. Ten Simple Rules for Selecting a Bio-ontology. *PLoS Computational Biology*. **12** (2): e1004743. [10.1371/journal.pcbi.1004743] `[id:Malone2016]`

Markowetz, F. 2017. All biology is computational biology. *PLoS Biology*. **15** (3): e2002050. [10.1371/journal.pbio.2002050] `[id:Markowetz2017]`

Michener, W. K. 2015. Ten Simple Rules for Creating a Good Data Management Plan. *PLoS Computational Biology*. **11** (10): e1004525. [10.1371/journal.pcbi.1004525] `[id:Michener2015]`

Mobley, A., Linder, S. K., Braeuer, R., Ellis, L. M., Zwelling, L. 2013. A Survey on Data Reproducibility in Cancer Research Provides Insights into Our Limited Ability to Translate Findings from the Laboratory to the Clinic. *PLoS ONE*. **8** (5): e63221. [10.1371/journal.pone.0063221] `[id:Mobley2013]`

Noble, W. S. 2009. A quick guide to organizing computational biology projects. *PLoS Computational Biology*. **5** (7): e1000424. [10.1371/journal.pcbi.1000424] `[id:Noble2009]`

Nosek, B. A., Alter, G., Banks, G. C., Borsboom, D., Bowman, S. D., Breckler, S. J., Buck, S., Chambers, C. D., Chin, G., Christensen, G., Contestabile, M., Dafoe, A., Eich, E., Freese, J., Glennerster, R., Goroff, D., Green, D. P., Hesse, B., Humphreys, M., Ishiyama, J., Karlan, D., Kraut, A., Lupia, A., Mabry, P., Madon, T., Malhotra, N., Mayo-Wilson, E., McNutt, M., Miguel, E., Paluck, E. L., Simonsohn, U., Soderberg, C., Spellman, B. A., Turitto, J., VandenBos, G., Vazire, S., Wagenmakers, E. J., Wilson, R., Yarkoni, T. 2015. Promoting an open research culture. *Science*. **348** (6242): 1422--1425. [10.1126/science.aab2374] `[id:Nosek2015]`

Pepe, A., Cantiello, M., Nicholson, J. 2017. The arXiv of the future will not look like the arXiv. *Authorea preprint*. [10.22541/au.149693987.70506124] `[id:Pepe2017]`

Perez-Riverol, Y., Gatto, L., Wang, R., Sachsenberg, T., Uszkoreit, J., Leprevost, F., Fufezan, C., Ternent, T., Eglen, S. J., Katz, D. S., Pollard, T. J., Konovalov, A., Flight, R. M., Blin, K., Vizcaino, J. A. 2016. Ten Simple Rules for Taking Advantage of Git and GitHub. *PLoS Computational Biology*. **12** (7): e1004947. [10.1371/journal.pcbi.1004947] `[id:Perez2016]`

Piwowar, H. A., Vision, T. J. 2013. Data reuse and the open data citation advantage. *PeerJ*. **1** : e175. [10.7717/peerj.175] `[id:Piwowar2013]`

Pond, S. K., Wadhawan, S., Chiaromonte, F., Ananda, G., Chung, W. Y., Taylor, J., Nekrutenko, A. 2009. Windshield splatter analysis with the Galaxy metagenomic pipeline. *Genome Research*. **19** (11): 2144--2153. [10.1101/gr.094508.109] `[id:Pond2009]`

Sandve, G. K., Nekrutenko, A., Taylor, J., Hovig, E. 2013. Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*. **9** (10): e1003285. [10.1371/journal.pcbi.1003285] `[id:Sandve2013]`

Taschuk, M., Wilson, G. 2017. Ten simple rules for making research software more robust. *PLoS Computational Biology*. **13** (4): e1005412. [10.1371/journal.pcbi.1005412] `[id:Taschuk2017]`

Taylor, C. F., Paton, N. W., Lilley, K. S., Binz, P.-A., Julian, R. K., Jones, A. R., Zhu, W., Apweiler, R., Aebersold, R., Deutsch, E. W., Dunn, M. J., Heck, A. J. R., Leitner, A., Macht, M., Mann, M., Martens, L., Neubert, A. A., Patterson, S. D., Ping, P., Seymour, S. L., Souda, P., Tsugita, A., Vandekerckhove, J., Vondriska, T. M., Whitelegge, J. P., Wilkins, M. R., Xenarios, I., Yates, J. R., Hermjakob, H. 2007. The minimum information about a proteomics experiment (MIAPE).. *Nature biotechnology*. **25** (8): 887--93. [10.1038/nbt1329] `[id:Taylor2007]`

Weinberger, C. J., Evans, J. A., Allesina, S. 2015. Ten simple (Empirical) rules for writing science. *PLoS Computational Biology*. **11** (4): e1004205. [10.1371/journal.pcbi.1004205] `[id:Weinberger2015]`

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., {da Silva Santos}, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M, Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., {'t Hoen}, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., Mons, B. 2016. The FAIR Guiding Principles for scientific data management and stewardship.. *Scientific data*. **3** : 160018. [10.1038/sdata.2016.18] `[id:Wilkinson2016]`

Zhang, W. 2014. Ten Simple Rules for Writing Research Papers. *PLoS Computational Biology*. **10** (1): e1003453. [10.1371/journal.pcbi.1003453] `[id:Zhang2014]`

Ziemann, M., Eren, Y., El-Osta, A. 2016. Gene name errors are widespread in the scientific literature. *Genome Biology*. **17** (1): 177. [10.1186/s13059-016-1044-7] `[id:Ziemann2016]`