Ten Simple Rules for Data Storage

Edmund Hart [1,*], Pauline Barmby [2], Jeffrey Hollister [3], David LeBauer [4], Naupaka Zimmerman [5], Kara H. Woo [6], Sarah Mount [7], Timothée Poisot [8], François Michonneau [9],

1 Univeristy of Vermont, Department of Biology, Burlington
2 University of Western Ontario, Department of Physics and Astronomy
3 US Environmental Protection Agency, Atlantic Ecology Division
4 University of Illinois at Urbana-Champaign, National Center for Supercomputing Applications and Institute for Genomic Biology
5 University of Arizona, School of Plant Sciences
6 Washington State University, Center for Environmental Research, Education, and Outreach
7 University of Wolverhampton, School of Mathematics and Computer Science
8 Université de Montréal, Département de Sciences Biologiques
9 University of Florida, iDigBio, Florida Museum of Natural History, Gainesville, FL 32611-7800

∗ **E-mail: edmund.m.hart@gmail.com**

Abstract

Introduction

- Data is the lifeblood of science.
- Yet the how, when and where of storing the data is not often given much thought.
- This can have some unforeseen and sometimes unfortunate consequences
  - examples of where poor data management/storage caused big problems?
- Best practices for storing data are different than best practices for sharing/publishing data.
  - Often publishing data best practices are more standard, (e.g. [1])
  - Best practices can sometimes be discipline specific
- Avoiding these potential problems is possible if scientist and their research collaborators practice some simple rules.
- A discussion about this topic took place on the SWC mailing list
- In this mansucript we have distilled the essence of that discussion into 10 simple rules, if followed, will help facilitate quick, robust analysis, allow others to re-use your data for new insights, and serve as a record of the work that lives beyond a single publication.

Rule 1: Know what to expect

Most of the troubles encountered during the analysis, management, and release of data can be avoided by having a clear roadmap of what to expect *before* the data acquisition starts. How will the raw data be presented? In what format should they be for analysis? Does the study involves simulations, and what is the model output? Is there a community standard on the format for release? This can range from simple cases (sequencing data in the fasta format, that can be used as is throughout the analysis), to experimental designs involving several instruments, each with its own output format. Knowing the state in which the data needs to be at each step can help (i) create converters from these data, (ii) orient technological choices about how and where these data should be stored, and (ii) rationalizes the analysis pipeline, and make it more amenable to re-use.

Another side of preparedness is the ability to estimate the volume needed to store these data at each step. The strategy to apply is not the same when the total amount of data is in the order of a few Mb, than when it reaches the Gb or Tb sizes. Although (and we do not condone this practice) lighter datasets can be managed without much of a data management plan, larger ones require careful planning and preparation (see Rule 9).

Rule 2: Know your use case

Researchers should know their use case and store data appropriately. This involves answering the following questions. Should the raw data be archived (see rule 3)? Should the data used for analysis be prepared once, or re-generated from the raw data (and what difference does it means for storage and computing requirements)? Should the final data be released, and in what format? How do you track the changes made to the data, and where are they logged? Do you anticipate to make manual corrections, and why? Are there restrictions on the data, and how can you make them (*e.g.* for survey results) anonymous? Do you need validation from within your institution to release the data? Does your funding agency requires data deposition, and are there some specific platforms? Does the journal in which you plan to publish requires data deposition? None of these questions have universal answers, nor are they the only questions one should ask before starting data acquisition. But similarly to Rule 1, knowing what *you* will do with the data, when, and how, will bring you close to a very detailed roadmap on how to handle these data fron their acquisition to their publication.

Rule 3: Keep raw data raw

Since analytical and data processing procedures may improve or otherwise change over time, having access to the 'raw' or unprocessed data helps to facilitate future re-analysis and analytical reproducibility. Data should always be kept in a raw format whenever possible, within the constraints of technical limitations. In addition to being the most appropriate way to ensure transparency in analysis, having the data stored and archived in its original state given a common point of reference for derivative analyses. There's clearly some discussion about what constitutes "raw" (ohms off a temperature sensor?). However the focus here is really on the spirit of the rule. Data should be as "pure" as possible when it's stored. If derivations occur, they should include storage of relevant code and subsequent data sets. NEON handles this with a schema for various "levels" of data products that pertain to the amount of processing

that happens, here's a brief overview. We defined raw data as things like voltage, or unprocessed lidar returns. The issue of course is that this is a tremendous amount of data that NEON still hasn't quite figured out how to share and document (e.g. if you ask for L0 they'll ship you an HDD). The way around it is to write detailed "Algorithm Theoretical Basis Documents" (ATBD's) detailing the different processing "levels". This is mostly borrowed from the NASA EOSDIS program.

Rule 4: Store data in open formats

To maximize accessibility and long-term value, data should be stored in file formats whose specifications are freely-available. The appropriate file type will depend on the type of data being stored (e.g. numeric measurements, text, images, video) but the key idea is that data should not require proprietary software or hardware, or a license, to be accessed. Proprietary formats can change, maintaining organizations can go out of business, and changes in license fees could make access to data in proprietary formats simply unaffordable. Examples of open data formats include comma-separated values (CSV) for tabular data, hierarchical data format (HDF) for scientific data, portable network graphics (PNG) for images and extensible markup language (XML) for documents. Examples of closed formats include DWG (for AutoCAD drawings), Photoshop document (PSD), and Windows Media Audio (WMA) (need refs?). At a minimum, data being stored for archival purposes should be stored in open formats, even if day-to-day processing uses closed formats, for example due to software requirements.

Rule 5: Data should be uniquely identifiable

To aid reproducibility, the data used in a scientific publication should be uniquely identifiable. Ideally, datasets should have a unique identifier such as a Document Object Identifier (DOI), Archival Resource Key (ARK), or a Persistant URL (PURL). An increasing number of online services, such as Figshare, Zenodo or DataOne are able to provide these.

Datasets may evolve over time. In order to distinguish between different versions of the same data, each dataset should have a distinct name, which includes a version identifier.

A simple way to do this is to use date stamps as part of the dataset name. To avoid regional ambiguities, it is wise to use the ISO 8601 standard, which mandates the date format `YYYY-MM-DD` (i.e. from largest time unit to smallest). For example, the date `01-02-2015` could be in January (US format) or February (UK format), but in ISO 8601 format it has the canonical form `2015-02-01`.

Semantic versioning, as described in [2], is a richer approach to solving the same problem. An example of this can be seen in the CellPack datasets [3].

A semantic version number takes the form: `Major.Minor.Patch`, e.g. `0.2.7`. The **major version** numbers should incremented (or *bumped*) when a dataset scheme has been updated, or some other change is made that is not compatible with previous versions of the data with the same major version number. This might mean that an experiment using version `1.0.0` of the dataset could not be run on version `2.0.0` without making some changes to the data analysis. The **minor version** should be bumped when a change has been made which is compatible with older versions of the data with the same Major version. This means that any analysis that can be performed on version `1.0.0` of the data should be repeatable with version `1.1.0` of

the data. The **patch version** number should be bumped when typos or bugs have been fixed. For example version `1.0.1` of a dataset may fix a typo in version `1.0.0`.

Rule 6: Link relevant metadata and license files

- Importance of metadata (citations of MD best practices)
- Data formats with embedded metadata (HDF5)
- Linkage in a database
- How to best link flat files to metadata
- how to mention or include license file

Rule 7: Rule

Rule 8: Have a systematic backup scheme

Every storage medium can fail, and every failure can result in loss of data. Researchers should therefore ensure that data is backed up at all stages of the research process. Data stored on local computers or institutional servers during the collection and analysis phase should be backed up to other locations and formats to protect against data loss. No backup system is failsafe (see the stories of the Dedoose crash and the near deletion of Toy Story 2), so more than one backup system should be used. Kristin Briney advocates the Rule of 3 for backing up data: two onsite copies (such as on a computer, an external hard drive, or tape) and one offsite copy (e.g. in cloud storage). Keeping backups in multiple locations protects against data loss due to theft, natural disasters, etc.

Researchers should also test their backups regularly to ensure that they are functioning properly. Reasons they might not include:

- faulty backup software
- incorrect configuration (e.g., not backing up sub-directories)
- encryption (e.g., someone has encrypted the backups but lost the password)
- media errors

and many others.

Consider the backup plans of data repositories before publishing your data. Many repositories mirror the data they host on multiple machines. If possible, find out about the long-term storage plans of the repository. Are there plans in place to keep data available if the organization that manages the repository dissolves?

Rule 9: Data size matters / requires special considerations

- #39 and related GH issues #16, #19, #25
- Size classes:
  - larger than RAM
  - larger than HD space
  - larger than data storage server

- Storage method depends on the size of data; storage costs, transfer time, and computing costs can become substantial.
  - data generated by simulation and derived data should consider cost of storage vs. the cost of re-generating output.
  - For analyses of large data sets, the speed of reading and writing data can limt the speed of computation.
- Larger data sets that are actively used in analysis should be stored on a disk that is attached to a computer rather than being moved around between analysis and storage.
  - inactive data can be put in longer-term storage; this is less expensive, but can be slow to retrieve. Archiving of 'stale' files can be automated (and is at HPC centers).
- Data that is larger than memory can handle,
  - can be handled by 'big memory' nodes.
  - Computing can also be done 'in the database'
- Don't move (large data) around more than you have to - it can become inefficient, and make storage slower than necessary.
  - New tools make it easier to find and download data combined with reproducible scripts can lead to excessive and careless abuse of resources.
  - subset and compute on the server, in the database where possible. The dplyr R package does lazy eval; SQL can perform a wide range of data summaries, by groups, etc. On the other hand, it may be quicker to transfer normalized (e.g. 'flattening' a relational database can increase the size of data by orders of magnitude)
  - Use tools to store local 'cached' copies, instead of writing scripts that always download archived data. Only update data if there are changes. * knitr has a cache argument that saves time in re-computing and in re-downloading.
- For data larger than a single hard drive disk, up to multiple servers
  - requires a meta-data server to allow fast access to distributed across many disks
- For very large data
  - it is not practical to store data
  - there are trade offs among cost, information content, and accessibility.

Rule 10: Data should be stored in a machine readable format

Not only data should be stored in an open format to ensure that data will be easily and widely accessible (see Rule #4), they should also be stored in a format that allows computers to make sense of it.

As datasets become increasingly larger, it is crucial that they can be parsed efficiently. This is best achieved by using standard data formats that have clear specifications (e.g., CSV, XML, JSON, HDF5). Such data formats can be handled by a variety of programming languages as efficient and well-tested libraries for parsing them are typically available. These standard data formats also ensure interoperability,

facilitate re-use, and reduce the chances of data loss or mistakes being introduced during conversion between formats.

Because a computer will be able to import your data directly (i.e., without the need for manual manipulation of your data), the script used to import and modify the data can be made available and the origin of the data used in the analysis will be evident. In turn, it will make the analysis more robust as there will be no opportunity to introduce mistakes in the data, and it will make the analysis reproducible.

To take full advantage of the data, it is important that it is structured such that the data can be manipulated and analyzed easily, in other words that the data is tidy (Wickham2014tidy). With tidy data, each variable is a column, each observation is a row, and each type of observational unit is a table. When data is organized like this, it reduces the duplication of information and it is easier to subset or summarize the dataset to include the variables or observations of interest.

To facilitate interoperability, it is best to use variable that can be mapped to existing data standards. For instance, for biodiversity data, the Darwin Core Standard provides a set of terms that describe observations, specimens, samples, and related information for a taxa. Because each term is clearly defined and documented, each dataset can use the terms consistently facilitating data sharing across applications and disciplines.

With machine readable data, it is also easier to build an Application Programming Interface (API) to query the dataset to retrieve a subset of interest.

1 Acknowledgements

National Center for Supercomputing Applications. Software Carpentry Foundation. iDigBio/NSF.

Figure Legends

Figures here: Will need to figure out numbering...

Tables

Tables here: Will need to figure out numbering...

References

1. White E, Baldridge E, Brym Z, Locey K, McGlinn D, Supp S. Nine simple ways to make it easier to (re)use your data. Ideas in Ecology and Evolution. 2013;6: 1–10. doi:10.4033/iee.2013.6b.6.f

2. Preston-Werner T. Semantic Versioning 2.0.0. `http://semver.org`;

3. Johnson GT, Goodsell DS, Autin L, Forli S, Sanner MF, Olson AJ. 3D molecular models of whole HIV-1 virions generated with cellPACK. Faraday Discuss. The Royal Society of Chemistry; 2014;169: 23–44. doi:10.1039/C4FD00017J