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],

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

∗ **E-mail: emh@emhart.info**

Abstract

Introduction

Some example text with a citation [1]

Rule 1: Rule

1 Rule 2: Rule {Know your use case}

Researchers should know their use case and store data appropriately. Is this data collected and just being archived? Will it change regularly? How will those changes be logged (e.g. provenance if any)? Will this be shared via an API? Linked to a paper? What are the institutional restrictions? Can you use a commercial service like Dropbox or use a personally maintained system? Knowing the reason why you're sharing your data will constrain your choices here.

2 Rule 3: Rule {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.

3 Rule 4: Rule {Store data in open formats}

To maximize accessibility and long-term value, data should be stored in file formats whose specificiations are freely-available. The exact 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 to access. 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 * CSV for tabular data * HDF5 for (??) * ?? for images * (help me out here folks, it's been a long week) and examples of closed formats include XLSX, DICOM, (again need more examples). At a minimum, data being stored for archival purposes should be stored in open formats, even if day-to-day processing uses closed formats.

4 Rule 5: Rule {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 are able to provide these.

Datasets may evolve over time and in order to distinguish between different versions of the same data *Semantic versioning* should be used. An example of this can be seen in the CellPack datasets [2].

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`.

A common, but less semantically dense, alternative to semantic versioning is the use of date stamps as file or directory names to describe when the data was produced. This is problematic, as formats for date stamps vary regionally, in ways that can render the date stamp ambiguous. For example, the date `01-02-2015` can mean 02 Jan 2015 (US format) or 01 Feb 2015 (UK format), without metadata to say how the date stamp is formatted the reader cannot tell which date this is.

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). In this format, the ambiguous example above would have the canonical form `2015-02-01`.

Rule 6: Rule

Rule 7: Rule

5 Rule 8: Rule {What is your back-up scheme?}

Researchers should consider the safety of their data and ensure that it 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: always keep three copies of your 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 test their backups regularly to ensure that they are functioning properly.

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?

6 Rule 9: Rule {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: Rule

7 Acknowledgements

National Center for Supercomputing Applications. Software Carpentry Foundation.

Figure Legends

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

Tables

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

References

1. Goodman A, Pepe A, Blocker AW, Borgman CL, Cranmer K, Crosas M, et al. Ten simple rules for the care and feeding of scientific data. PLoS computational biology. Public Library of Science; 2014;10: e1003542.

2. 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