Research Data Alliance
Data Versioning Working Group
Updated Compilation of Data Versioning Use Cases

Version 1.3

Jens Klump, Lesley Wyborn, Kirsten Elger, Mingfang Wu (Editors)

Version information		
Release	Date	Description
V1.0	2020-01-16	The version was submitted to RDA for community review. This version has 38 use cases. Version 1.0. **DOI:** 10.15497/RDA00041 The final report from the WG is based on the analysis of the 38 use cases. **Citation:** Klump, J., Wyborn, L., Downs, R., Asmi, A., Wu, M., Ryder, G., & Martin, J. (2020). Compilation of Data Versioning Use cases from the RDA Data Versioning Working Group. Version 1.0. *Research Data Alliance*. DOI: 10.15497/RDA00041
V1.1	2020-04-06	Add the google use case (#Google, No. 39) Add figure captures Revised the ASTER (#ASTER) use case **Recommended citation:** Klump, J., Wyborn, L., Downs, R., Asmi, A., Wu, M., Ryder, G., & Martin, J. (2020). Compilation of Data Versioning Use cases from the RDA Data Versioning Working Group. Version 1.1. *Research Data Alliance*. DOI: 10.15497/RDA00041
V1.2	2022-06-08	Add TOC, new use cases to be added
V1.3	2025-01-06	New use cases added (No. 40-55). Release of Version 1.3

Table of Contents

Foreword

Data versioning is a fundamental element to ensuring the reproducibility of research. Work in other Research Data Alliance (RDA) groups on data provenance and data citation, as well as the W3C Dataset Exchange Working Group (DXWG), have highlighted that definitions of data versioning concepts and recommended practices are still missing.

An important driver to more closely examine data versioning practices came from the work of the RDA Working Group (WG) on Data Citation, whose final report recognised the need for systematic data versioning practices.

However, while the recommendations put forward by the RDA WG on Data Citation are well suited for relational databases that are accessed using database queries, the recommendations sparked a debate that highlighted the need for more general principles on data versioning and a clarification of the terminology used to describe versioning of data. This led to the formation of the RDA Working Group on Data Versioning. An early requirement for the new WG was to capture use cases where versioning requirements could not be met by the RDA WG on Data Citation recommendations. Numerous organisations and individuals were approached, or offered to contribute use cases.

In the course of the active phase of the RDA Data Versioning Interest Group and then RDA Data Versioning Working Group, 39 use cases from about 33 organisations representing different domains and data types were documented. There were three main sources for these 39 use cases:

A) Web sources (use cases 1-10);
B) RDA Sources (use cases 11-12); and
C) Data Repositories (use cases 13-39).

These 39 use cases were presented in Version 1.1, which was published in April 2020 along with contextual information including definitions, workflows and 'best practices' for versioning. Analysing the collected use cases and other resources on data versioning we were able to extract versioning patterns. These versioning patterns form the basis of the data versioning principles presented in the Final Report of the RDA Data Versioning Working Group.

For Version 1.3, an additional 16 use cases have been added, either from individual contributions or from literature reviews.

Acknowledgements

The chairs of the RDA Data Versioning WG would like to thank all who contributed use cases to the WG and joined the discussions at the plenary sessions and along the way.

Use cases were contributed to the first version of the use case collection by Natalia Atkins (IMOS, Australia), Catherine Brady (Australian Research Data Commons (ARDC)), Jeff Christiansen (QCIF, Australia), Martin Capobianco, Andrew Marshall and Margie Smith (Geoscience Australia), Bob Downs (Columbia University, USA), Kirsten

Elger and Damian Ulbricht (GFZ Potsdam, Germany), Ben Evans, Nigel Rees, Kate Snow and Lesley Wyborn (National Computational Infrastructure, Australia), Siddeswara Guru (TERN, Australia), Julia Hickie (National Library of Australia), Dominic Hogan (CSIRO, Australia), Leslie Hsu (USGS, USA), Paul Jessop (International DOI Foundation), Dave Jones (StormCenter Communications Inc., USA), Danie Kinkaide (BCO-DMO, USA), Heather Leasor (ADA, ANU), Benno Lee (Rensselaer Polytechnic Institute, USA), , Simon Oliver (Digital Earth Australia), Simon O'Toole (Australian Astronomy Observatory (AAO)), Andreas Rauber (Vienna University of Technology), Martin Schweitzer (Bureau of Meteorology, Australia).

For version 1.3, further use cases were kindly contributed by Melinda Hodkiewicz (University of Western Australia), Adina Wagner (Forschungszentrum Jülich, Germany), Martin Fenner (Front Matter, Germany).

Special thanks go to the Australian Research Data Commons for their support.

We also like to thank our RDA Secretariat and TAB Liaisons, Stefanie Kethers and Tobias Weigel, for their guidance and support for version 1.0 and version 1.1, and Bridget Walker for Version 1.3.

A) Web Sources

1 W3C Data on the Web Best Practices (#W3C-BP[1])

https://www.w3.org/TR/dwbp/#dataVersioning

Datasets published on the Web may change over time. Some datasets are updated on a scheduled basis, and other datasets are changed as improvements in collecting the data make updates worthwhile. In order to deal with these changes, new versions of a dataset may be created. Unfortunately, there is no consensus about when changes to a dataset should cause it to be considered a different dataset altogether rather than a new version. In the following, we present some scenarios where most publishers would agree that the revision should be considered a new version of the existing dataset:

- Scenario 1: a new bus stop is created and it should be added to the dataset;
- Scenario 2: an existing bus stop is removed and it should be deleted from the dataset; and
- Scenario 3: an error was identified in one of the existing bus stops stored in the dataset and this error must be corrected.

In general, multiple datasets that represent time series or spatial series, e.g. the same kind of data for different regions or for different years, are not considered multiple versions of the same dataset. In this case, each dataset covers a different set of observations about the world and should be treated as a new dataset. This is also the case with a dataset that collects data about weekly weather forecasts for a given city, where every week a new dataset is created to store data about that specific week.

[1] This hash tag is used to reference corresponding use cases.

Scenarios 1 and 2 might trigger a major version, whereas Scenario 3 would likely trigger only a minor version. But how you decide whether versions are minor or major is less important than that you avoid making changes without incrementing the version indicator. Even for small changes, it is important to keep track of the different dataset versions to make the dataset trustworthy. Publishers should remember that a given dataset may be in use by one or more data consumers, and they should take reasonable steps to inform those consumers when a new version is released. For real-time data, an automated timestamp can serve as a version identifier. For each dataset, the publisher should take a consistent, informative approach to versioning, so data consumers can understand and work with the changing data.

2 W3C Dataset Exchange Use Cases and Requirements (#W3C-UCR)

https://w3c.github.io/dxwg/ucr/#ID4

Most datasets that are maintained long-term and evolve over time have distributions of multiple versions. However, the current DCAT model does not cover versioning with sufficient details. Being able to publish dataset version information in a standard way will help both producers publishing their data on data catalogues or archiving data and dataset consumers who want to discover new versions of a given dataset, etc. We can also see some similarities with software versioning and dataset versioning, for instance, some data projects release daily dataset distributions, major/minor releases etc. Probably, we can use some of the lessons learned from software versioning. There are several existing dataset description models that extend DCAT to provide versioning information, for example, HCLS Community Profile.

Links:

- https://www.w3.org/TR/hcls-dataset/#datasetdescriptionlevels
- https://www.w3.org/TR/dwbp/#dataVersioning
- https://www.w3.org/TR/dwbp-ucr/#R-DataVersion
- http://db.csail.mit.edu/pubs/datahubcidr.pdf
- https://lists.w3.org/Archives/Public/public-dxwg-wg/2017Jun/thread.html#msg6
- https://github.com/w3c/dxwg/issues?q=label%3Aversion

Related use cases:

- 5.32 Relationships between Datasets [ID32]

Related requirements:

- 6.5 Define version
- 6.6 Version identifiers
- 6.7 Version release dates
- 6.8 Version changes
- 6.9 Version discovery

3 Wikipedia Page on Software Versioning (#Wiki-SV)

https://en.wikipedia.org/wiki/Software_versioning

Software versioning is the process of assigning either unique *version names* or unique *version numbers* to unique states of computer software. Within a given version number category (major, minor), these numbers are generally assigned in increasing order and correspond to new developments in the software. At a fine-grained level, revision control is often used for keeping track of incrementally different versions of electronic information, whether or not this information is computer software.

Modern computer software is often tracked using two different software versioning schemes—an internal version number that may be incremented many times in a single day, such as a revision control number, and a *released version* that typically changes far less often, such as *semantic versioning*[1] or a project code name.

4 Semantic Versioning (#SV)

Preston-Werner, T. (2013). Semantic Versioning 2.0.0. https://semver.org/spec/v2.0.0.html

In the world of software management there exists a dread place called “dependency hell.” The bigger your system grows and the more packages you integrate into your software, the more likely you are to find yourself, one day, in this pit of despair.

In systems with many dependencies, releasing new package versions can quickly become a nightmare. If the dependency specifications are too tight, you are in danger of version lock (the inability to upgrade a package without having to release new versions of every dependent package). If dependencies are specified too loosely, you will inevitably be bitten by version promiscuity (assuming compatibility with more future versions than is reasonable). Dependency hell is where you are when version lock and/or version promiscuity prevent you from easily and safely moving your project forward.

As a solution to this problem, I propose a simple set of rules and requirements that dictate how version numbers are assigned and incremented. These rules are based on but not necessarily limited to pre-existing widespread common practices in use in both closed and open-source software. For this system to work, you first need to declare a public API. This may consist of documentation or be enforced by the code itself. Regardless, it is important that this API be clear and precise. Once you identify your public API, you communicate changes to it with specific increments to your version number. Consider a version format of X.Y.Z (Major.Minor.Patch). Bug fixes not affecting the API increment the patch version, backwards compatible API additions/changes increment the minor version, and backwards incompatible API changes increment the major version.

I call this system “Semantic Versioning.” Under this scheme, version numbers and the way they change convey meaning about the underlying code and what has been modified from one version to the next.

5 DataCite on data versioning (#DataCite)

DataCite recommends to include version information in data citations. The DataCite metadata kernel has an optional element “version” to record the version of a dataset. DataCite recommends to use semantic versioning (#SV): major_version.minor_version. Register a new identifier for a major version change. Data stewards need to determine which are major vs. minor versions. https://schema.datacite.org/meta/kernel-4.0/doc/DataCite-MetadataKernel_v4.0.pdf

6 Recommended Practice for Statisticians (#AS)

Bryan, J. (2018). Excuse Me, Do You Have a Moment to Talk About Version Control? *The American Statistician*, 72(1), 20–27. https://doi.org/10.1080/00031305.2017.1399928

Data analysis, statistical research, and teaching statistics have at least one thing in common: these activities all produce many files! There are data files, source code, figures, tables, prepared reports, and much more. Most of these files evolve over the course of a project and often need to be shared with others, for reading or edits, as a project unfolds. Without explicit and structured management, project organization can easily descend into chaos, taking time away from the primary work and reducing the quality of the final product. This unhappy result can be avoided by repurposing tools and workflows from the software development world, namely, distributed version control. This article describes the use of the version control system Git and the hosting site GitHub for statistical and data scientific workflows. Special attention is given to projects that use the statistical language R and, optionally, R Markdown documents. Supplementary materials include an annotated set of links to step-by-step tutorials, real world examples, and other useful learning resources. Supplementary materials for this article are available online.

7 Git Workflows (#Git-workflows)

From Marcel Jurtz “A Software Developer’s Blog”, with permission. https://blog.mjurtz.com/2018/09/git-workflows/

Almost all programming projects work with some kind of version control. When I started to work with Git, I used the tool also directly for my private projects. But especially at the beginning, I found it hard to structure my commits and branches in a practical way. For this reason I would like to show you some common strategies today, the so-called Git Workflows.

Simple Workflow

The simple workflow consists of a single master branch. There is only this one branch to which changes are pushed. This workflow is only suitable for very small projects, e.g. private ones, where only you work on yourself. As the team grows, this workflow becomes very messy and you’re going to have to deal with a lot of merge conflicts.

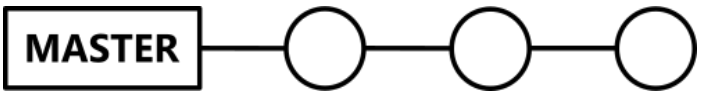


Figure 1. Git workflow: Single master branch

Feature Branches

This second level adds feature branches to the simple workflow. These branches are used to develop new functionalities separately from the rest of the project. After a feature is completed, the branch is merged. Unlike the master branch, the feature branches are therefore short-lived and only exist until their merge. Depending on their complexity, feature branches can often be further subdivided. Just make sure you don't exaggerate, which could again affect the overall structure.

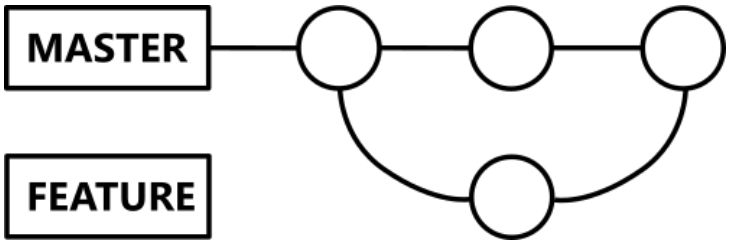


Figure 2.Git workflow: Feature branch

Developer Branch

With the Developer Branch, a second, long-lived branch is created next to the Master Branch. This is the only place where development takes place, so that the master branch always remains in a release-ready state. Here, however, similar problems arise as with the simple workflow, which is why it should only be used for very small teams.

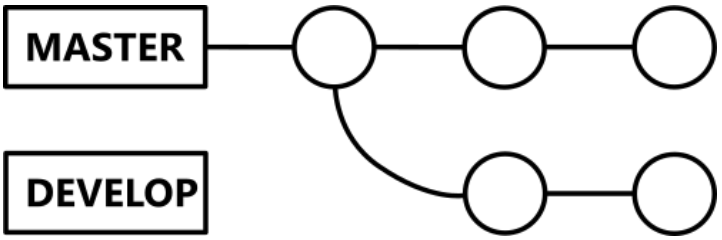


Figure 3.Git workflow: Developer branch

Developer and Feature Branches

The previous two strategies can be combined very well. Again, the master branch must always be ready for release, feature branches are only ever merged with the developer branch. After successful testing of the functionalities on the developer branch, this branch is merged to master, which then can be released.

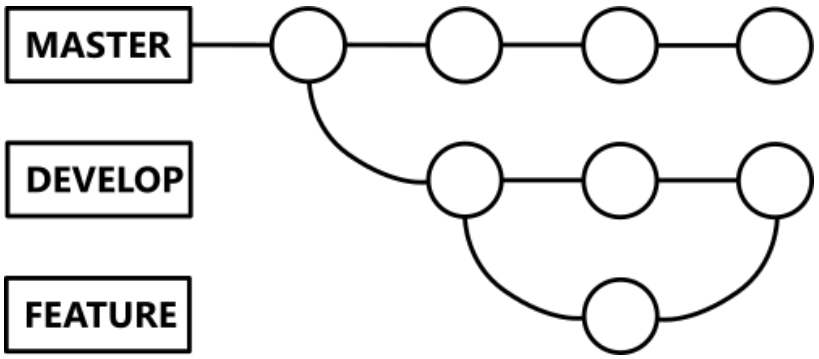


Figure 4.Git workflow: Developer and feature branches

Release Branches

This extension of the developer and feature branch workflow is often used for large projects that are planning frequent releases. For a new release, a new release branch is created from the developer branch. This only is used for final bug fixes, no new features are developed here. As soon as the release can be shipped, the branch will be merged into both the master and the developer branch. The fixes in the release branches allow other teams to work on new features without disturbing the work on the release.

The model is often complemented by another branch: the hotfix branch which allows direct bug fixing from the master branch.

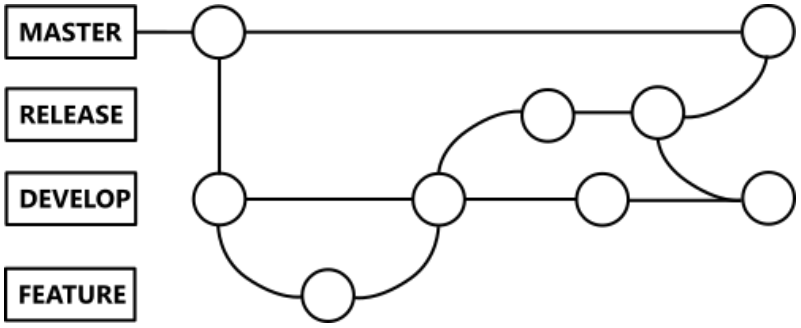


Figure 5.Git workflow: Release, developer and feature branches

But which concept is right for me?

Basically, the more complex your project, the more complex the workflow should be. But also for one-man projects it often makes sense not to use the simple workflow and to use a branching strategy already here. For my own projects, for example, I currently use the Developer Branch concept. But whatever you decide to do: Make sure you have a consistent naming strategy for branches (and commits, of course) and you're good to go.

8 DVC Data Version Control (#DVC)

https://dvc.org/

DVC is built to make ML models shareable and reproducible. It is designed to handle large files, data sets, machine learning models, and metrics as well as code.

DVC allows storing and versioning source data files, ML models, intermediate results with Git, without checking the file contents into Git. It is useful when dealing with files that are too large for Git to handle. DVC stores information about your data file in a special DVC-file, that has a description of a file that can be used for versioning. DVC supports various types of remote locations for your data files and allows you to easily store and share your data alongside your code.

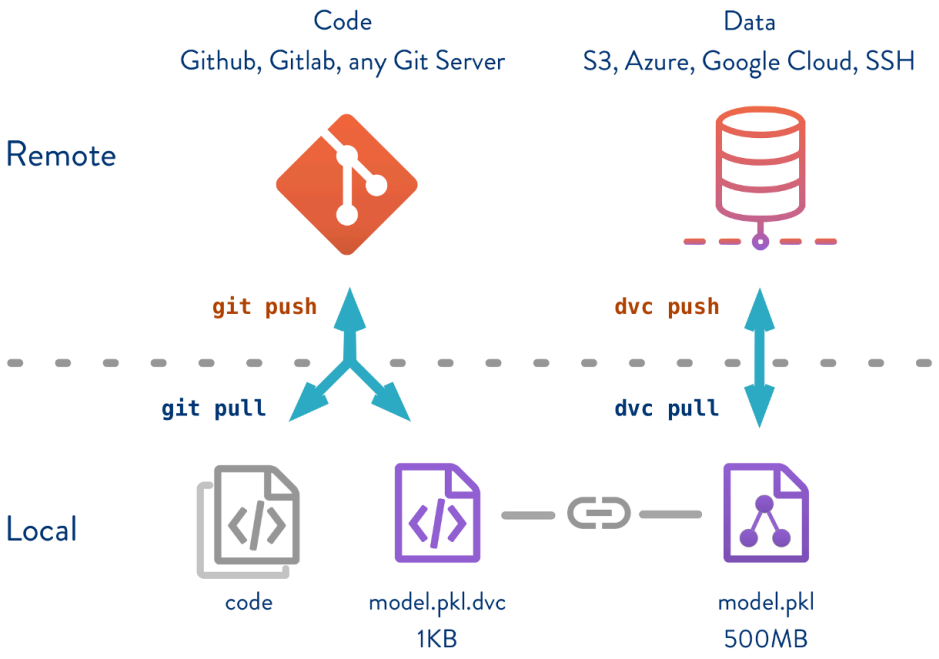


Figure 6. Data version control workflow

In this very basic scenario, DVC is a better replacement for git-lfs (check the Related Technologies to get a better sense why) and ad-hoc scripts on top of Amazon S3 (or name-it cloud) that are usually used to manage ML artifacts like model files, data files, etc. Unlike git-lfs, DVC doesn't require installing a server; it can be used on-premises (NAS, SSH, for example) or with any major cloud provider (S3, Google Cloud, Azure).

There are two ways to get to the previous version of the dataset or model - a full workspace checkout or checkout of a specific data or model file. Let's consider the full checkout first. It's quite straightforward:

> v1.0 is a Git tag that should be created in advance to identify the data set version you are interested in, it can be just a Git commit hash instead.
>
> ```
> $ git checkout v1.0
> ```
>
> ```
> $ dvc checkout
> ```

These commands will restore the working tree to the first snapshot we made - code, dataset and model files. DVC optimizes this operation internally to avoid copying

dataset or model files each time. So dvc checkout is quick even if you have a large dataset or model files.

On the other hand, if we want to keep the current version of code and go back to the previous dataset only, we can do something like this (make sure that you don't have uncommitted changes in the data.dvc):

```
$ git checkout v1.0 data.dvc
$ dvc checkout data.dvc
```

If you run git status you will see that data.dvc is modified and currently points to the v1.0 of the data set. While code and model files are from the v2.0 version.

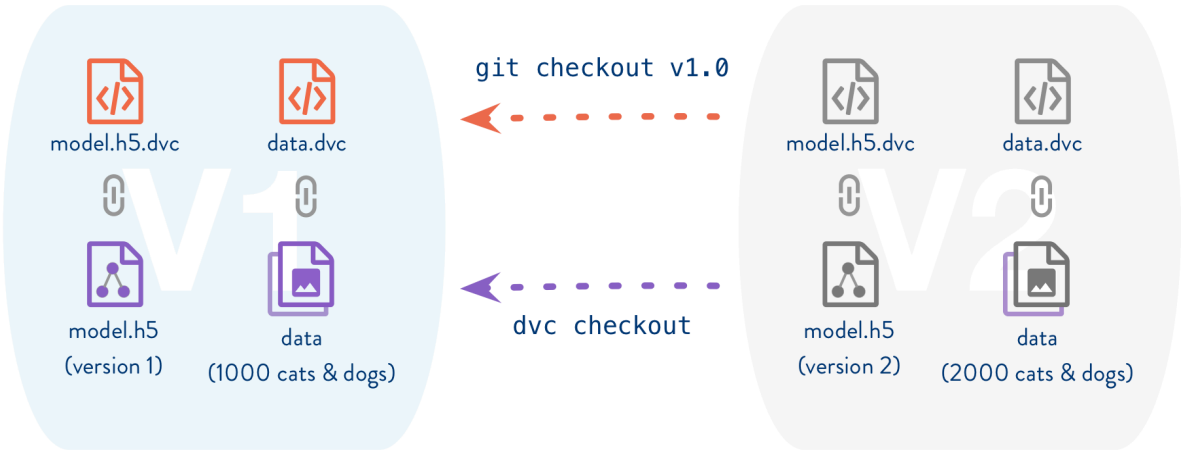


Figure 7. Data version control checkouts

To share your data with others you need to setup a remote repository. Check the Share Data And Model Files use case to get a high level overview on how to setup it and use dvc pull and dvc push commands to collaborate. Please, don't forget to check the versioning get started example to get a hands-on experience with datasets and models versioning.

9 OASIS Naming Guidelines Part 2: Metadata and Versioning (#OASIS)

Cover, R., & McRae, M. (2008). OASIS Naming Guidelines: Metadata and Versioning (Specification) (p. 18). Burlington, MA, USA: OASIS Technical Committee. Retrieved from http://docs.oasis-open.org/specGuidelines/namingGuidelines/metadata.html

Informally, we use the term "version" in casual reference to any instance (expression, manifestation) of a specification, or parts of a specification, having some genetic relationship to other instances in its lineage: "let's create a new version" or "an earlier version". In our metadata model, formally, a Version of an OASIS specification refers to a significant body of work that is chartered to take place (typically) over several months, often leading to the creation of an OASIS Standard. A Version is thus a "specification development stage [identified] for purposes of distinguishing levels of implementation and conformance by a public community of developers. An OASIS

Standard is associated with a single version throughout its development and approval..."

A specification Version is represented textually by a numeric string composed of digits [0-9] and period (".") corresponding to any of the following lexical models provided below (as examples), as may be relevant to the Technical Committee's (TC) work activity and preference for major/minor version notation. Formally, using parentheses to indicate optionality and "#" to represent a digit, the allowable pattern is: #(#).#(#)(.#(#)). Use of any other pattern for version number must be negotiated with the TC Administration.

10 Operational Readiness Levels Model (#ORLM)

Contributed by Dave Jones (StormCenter Communications Inc.) for ESIP

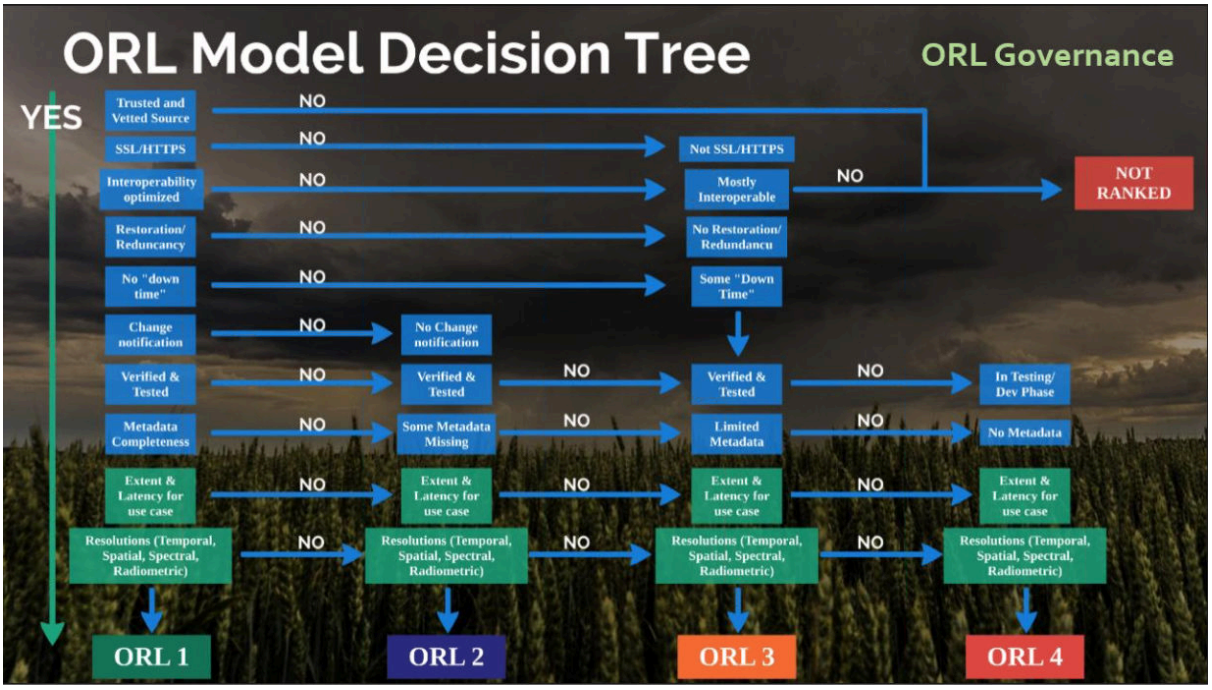


Figure 8. ORL model decision tree

B) RDA Sources

11 RDA Data Citation Recommendation (#RDA-DDC-R)

Contributed by Andreas Rauber (Co-chair of the RDA Data Citation WG)

https://www.rd-alliance.org/group/data-citation-wg/outcomes/data-citation-recommendation.html

Digitally driven research is dependent on quickly evolving technology. As a result, many existing tools and collections of data were not developed with a focus on long term sustainability. Researchers strive for fast results and promotion of those results, but without a consistent and long term record of the validation of their data, evaluation and verification of research experiments and business processes is not possible.

There is a strong need for data identification and citation mechanisms that identify arbitrary subsets of large data sets with precision in a machine-actionable way. These mechanisms need to be user-friendly, transparent, machine-actionable, scalable and applicable to various static and dynamic data types. As changes to data can affect anything ranging from an individual value to entire subsets of data and happen at any time interval, ranging from milliseconds to annual batch updates, we would like to have a single mechanism that applies to all kinds of data, data representations, and amounts of changes. Imposing any kind of semantic structure on such versions turns out to be difficult and not applicable generally and across different data sources. For example, the same update to data that does not impact its interpretation for a specific use case might lead to a different interpretation on another use case (such as exact reproducibility), making the difference between a major and minor version number update confusing. It also violates the principle of not embedding semantics in an identifier. Any semantic interpretation of the impact of updates should thus be performed separately as provenance metadata (update documentation) and not be included in any identifier creation.

The RDA Recommendations on data citation thus recommend not applying version numbers to entire data sets (and, specifically, not to pre-defined subsets of such data), but to (1) version and timestamp individual updates to data items on an element/record level (i.e. marking each addition of a record with a timestamp when it became available in the data set, marking deleted records as deleted with the according timestamp, and marking updates to values as deleted and re-inserted with the new value at a specific timestamp); and to (2) assign identifiers to timestamped queries which allow to retrieve the specific subsets at any given point in time. Instead of discrete version numbers, a version of a dataset thus is indicated by the status of the data set at a given point in time. This allows any state of a data set to be retrieved, and allows the current version of any data set to be used at any point in time. The principle is applicable to all types of data, ranging from numeric data to software code or document editing systems, with versioning systems allowing to retrieve the state of any code document as it existed at a specific point in time). Optimizations specifically for high-frequency updates to data may include not maintaining/keeping the update states of the dataset that were never read/accessed, i.e. states that were never observed.

12 RDA Data Foundations and Terminology IG (#RDA-DFT)

https://smw-rda.esc.rzg.mpg.de/index.php?title=Versioning

Definition: Generate a (changed) copy of a data object that is uniquely labeled with a version number. The intent is to enable access to prior versions.

Explanation: Note that a version is different from a backup copy, which is typically a copy made at a specific point in time, or a replica, which is a copy of a data object that can be periodically updated.

Related term – version, replication

Example:

Scope: RDA Term Collection Core

C) Data Repository Sources

13 da|ra Registration agency for social and economic data (#da|ra)

da|ra (Registration agency for social and economic data) provides recommendations on versioning:
https://www.da-ra.de/fileadmin/media/da-ra.de/PDFs/TechnicalReport_2014-18.pdf

In general the following aspects should be considered regarding versioning (p.16)

- An object with an assigned DOI name should not be changed.
- Each change must be saved as a new version and a new DOI name must be assigned.
- The publication agent is responsible for versioning.

The GESIS Leibniz Institute for the Social Sciences (www.gesis.org) is compliant with this recommendation and is using a three-digit-versioning.

Major.Minor.Revision

Major number starts with „1“, Minor and Revision number start with „0“ separate with „.“

First version of a data file is „1.0.0“.

1. Increase of the first digit if new data is added (e. g. waves, samples etc.)
2. Change of the second digit if corrections are made, which influence the analysis (e. g. change of values of respondents)
3. If the documentation is changed or amended (typing error or more detailed text added etc.) only the third digit will be increased

This versioning is based on the recommendations of the Data Documentation Initiative (DDI). DDI-Lifecycle 3.2

http://www.ddialliance.org/Specification/DDI-Lifecycle/3.2/drafts/IVMR_DRAFT.pdf ; page 2/3 “Versioning”

In the GESIS Data catalogue (DBK) the versioning and corresponding errata are documented. Description (in German only) please see here:
http://www.gesis.org/fileadmin/upload/forschung/publikationen/gesis_reihen/gesis_methodenberichte/2012/TechnicalReport_2012-01.pdf on page 13/14

14 DIACHRON project (#DIACHRON)

https://cordis.europa.eu/project/rcn/108537_en.html

DIACHRON was an integration project that addressed certain issues arising from the evolution of the data such as:

- Detect the changes that happen to datasets (tracking the evolution)
- Archive multiple versions of data and cite them accordingly to make the reference of previous data feasible (archiving and citation)

- Retrieve and query previous versions (time traveling queries)
- Validate and repair various data deficiencies (curation problem)
- Identify the cause of the evolution of the datasets in respect with the real world evolution of the entities the datasets describe (provenance problem)
- Provide various quality metrics so as to enable quality assessment of the harvested datasets and determination of the datasets versions that need to be preserved (appraisal)

The DIACHRON solution aims not only to store previous versions for preservation in case of future need of them, but to create a live repository of the data that captures and highlights data evolution by keeping all data (current and previous) accessible, combined with a toolset that handles the full life cycle of the Data Web.

15 United States Geological Survey (#USGS)

Contributed by Leslie Hsu

https://www.usgs.gov/about/organization/science-support/office-science-quality-and-integrity/guidance-documenting

Guidance on Documenting Revisions to USGS Scientific Digital Data Releases *(*Updated October 4, 2019)

Purpose

This guidance describes a formal revision process for scientific digital data and associated metadata that have been released as USGS information products. This guidance supplements U.S. Geological Survey (USGS) Fundamental Science Practices (FSP) requirements in SM 502.7 and SM 502.8.

Data release revisions are characterized as Level 1, Level 2, Level 3, or Level 4, similar to the characterization of levels for revisions to USGS publication series products. The procedures for documenting data release revisions vary depending on the level of revision.

This guidance covers individual USGS datasets. Not covered in this guidance are USGS approved databases and data services as defined in SM 502.8 because they have other approved processes in place for making revisions, including data quality evaluation, prior to data being uploaded. Examples of these systems or services include National Water Information System (NWIS-Web), USA National Phenology Network (USA-NPN), and Biodiversity Information Serving Our Nation (BISON).

Reasons for Revisions

The reason for revising a data release will guide the process of review and approval. The revision level (1, 2, 3, or 4) depends upon whether the changes could affect outcomes of future data use and on the proportion of the data that needs to be corrected.

- Level 1 revisions are changes to the metadata record that do not affect the understanding of the data, changes to data files that do not involve modifying the data itself, and changes to a landing page.
- Level 2 revisions are changes that are not expected to have a significant impact on the use of the data, and apply to a small number of data values. Examples include adding negative signs to one or two values in the data; adding five values that were missing from the original data release; or making corrections to transposed latitude and longitude values in the metadata record.
- Level 3 revisions are data-appending revisions, that is, adding new data records without changing the data structure. A primary example is the release of data in stages to meet project timelines and increase the amount of data provided in an information product.
- Level 4 revisions are changes that are expected to have a significant impact on the use of the data, including changing a large number of data values, such as correcting an error in the formula for calibrating the data. Changes to the data structure are also Level 4 revisions. These revisions might add new tables to a data release that is structured as a database, or add new variables to a table. These revisions are appropriate for data releases that are standalone research products, rather than for data that are foundations of associated or companion scientific publications, or a policy decision.

Level 1 Revision

A Level 1 revision does not change the dataset. The following are examples of Level 1 revisions:

- Changes in the metadata record to add new keywords, contact information, or a link to a new publication.
- Changes in a data file to correct a misspelling in a data header or in a site location name.
- Changes in a data landing page to correct a misspelt word in the title or abstract, or to revise one of the contacts listed.

These revisions can be done by replacing or updating the erroneous file or text and updating the metadata record and any additional supporting documentation. Ensure that the updated metadata record replaces the previous version provided to the USGS Science Data Catalog.

Although it is a good practice to have an independent reviewer check to ensure that no errors were introduced during the revision process, review and approval for Level 1 revisions do not need to be documented in the internal USGS Information Product Data System (IPDS).

Level 2 Revision

A Level 2 revision creates a new version of the data release that will normally be used instead of the previous version. The changes for a Level 2 revision, however, should not significantly impact the use of the data. The following are examples of Level 2 revisions:

- Adding negative signs that were omitted from one or two data values in the original data release.
- Adding five data values that were missing in the original data release.
- Correcting latitude and longitude values for geospatial locations that were transposed in the metadata record.
- Modifying a polygon shapefile by slightly shifting a line, so that a boundary is consistent with the boundary in another polygon shapefile that was subsequently released.

Science Center approving officials for data releases should be consulted if help is needed to distinguish between Level 2 and Level 4 error corrections, in recognition of the differences in methods among scientific disciplines. Level 2 review and approval not only focus on the sections of the data release that are corrected but also identify any inadvertent changes made to other sections as a consequence of the corrections.

When a Level 2 revision is needed, the following actions are required:

1. Create a new data release record in the IPDS and complete the review and approval steps for the new data release version. Review and approval should focus on the new or corrected sections but also identify any inadvertent changes made to other sections as a consequence of the modification. The new IPDS record is used to ensure that the requirements of SM 502.7 and SM 502.8 have been met.

2. Do not create a new Digital Object Identifier (DOI). The existing DOI should be used for the revised data release. If the data must be removed from public access for any period of time during the revision process, the DOI should be directed to a "temporary tombstone page," explaining that the release is being revised and will be available again soon. Most Trusted Digital Repositories should be able to provide this messaging on the existing landing page of the data release, without needing to change the Location URL of the DOI.

3. Assign a version number to the revised data release product or update the existing version number, for example change version 1.1 to version 1.2, and revise the title of the data release in the recommended citation and the metadata file to include the new version number. Refer to the "Examples" section.

4. Revise the metadata record as follows:
   a. Add processing steps that describe the changes.
   b. Insert the version number and version release date into the title and recommended citation.
   c. Update the metadata revision date.
   d. Add instructions for obtaining prior versions.
   e. Provide the revised metadata record to the USGS Science Data Catalog.

5. Modify the landing page as follows:
   a. Point users to the new version of the data and metadata.
   b. Include a list of version numbers and version release dates.

c. Link to a revision history text file that provides a detailed description of the changes and a justification for making the changes.

6. Once the new version is published, update the DOI in the DOI Tool as follows:
   a. Login to the DOI Tool, open the DOI, and go to the Supplemental Information tab.
   b. In the section “Dates Relevant to the Data,” add a Date Type of ‘updated’ and pair it to a Date that denotes the mm/dd/yyyy of the published update.
   c. Click the ‘Add’ button.
   d. On the ‘Manage Record’ tab, update the Title to include the version number (refer to action 3 above).
   e. Click ‘Update Published Record in DataCite’ in the left menu.

7. Preserve the previous version of the data in accordance with records management and litigation holds requirements in case that version is needed to understand any information that was based on it. Refer to the "Archiving Prior Versions of Data" section for additional guidance.

8. If the revision could affect scientific conclusions in an existing USGS publication, consult your assigned Bureau Approving Official (BAO) in the Office of Science Quality and Integrity (OSQI) for guidance.

Level 3 Revision

For a Level 3 revision, the data are updated to include additional data, which might be from a new time period, place, or field activity. Level 3 review and approval focus on the new data that are added, but also identify any inadvertent changes made to other sections as a consequence of the appended data.

When a Level 3 revision is needed, the following actions are required:

1. Create a new data release record in the IPDS and complete the review and approval steps for the new data release version. Review and approval should focus on the new sections but also identify any inadvertent changes made to other sections. The new IPDS record is used to ensure requirements in SM 502.7 and SM 502.8 have been met.

2. Do not create a new Digital Object Identifier (DOI). If the data must be removed from public access for any period of time during the revision process, the DOI should be directed to a “temporary tombstone page,” explaining that the release is being revised and will be available again soon. Most Trusted Digital Repositories should be able to provide this messaging on the *existing* landing page of the data release, without needing to change the Location URL of the DOI.

3. Assign a version number to the revised data product and revise the title of the data release in the recommended citation and the metadata file to include the new version number. The change in the version number for Level 3 revisions is

usually done by changing the number before the decimal point, for example, changing version 1.1 to version 2.0. Refer to the "Examples" section.

4. Once the new version is published, update the DOI in the DOI Tool as follows:
   a. Login to the DOI Tool, open the DOI, and go to the Supplemental Information tab.
   b. In the section “Dates Relevant to the Data,” add a Date Type of ‘updated’ and pair it to a Date that denotes the mm/dd/yyyy of the published update.
   c. Click the ‘Add’ button.
   d. On the ‘Manage Record’ tab, update the Title to include the version number (refer to action 3 above).
   e. Click ‘Update Published Record in DataCite’ in the left menu.

5. Revise the metadata record as follows:
   a. Add processing steps that describe the changes.
   b. Insert the version number and version release date into the title and recommended citation.
   c. Update the time period information to address the dates of the newly appended data.
   d. Update the metadata revision date.
   e. Add instructions for obtaining prior versions.
   f. Provide the revised metadata record to the USGS Science Data Catalog.

6. Modify the landing page as follows:
   a. Point users to the new version of the data and metadata.
   b. Include a list of version numbers and version release dates.
   c. Link to a revision history text file that provides a detailed description of the changes and a justification for making the changes.

7. Preserve the previous version of the data in accordance with records management and litigation holds requirements in case that version is needed to understand any information that was based on it. Refer to the "Archiving Prior Versions of Data" section for additional guidance.

8. If the revision could affect scientific conclusions in an existing USGS publication, consult your assigned BAO for guidance.

Level 4 Revision

For a Level 4 revision, the data structure is modified, or data are significantly and substantially changed. Review and approval focus on the new structure and the new data, but also identify any inadvertent changes made to other sections as a consequence of the revisions. The following are examples of Level 4 revisions:

- Modifying a data structure to allow inclusion of a new table or column of values.
- Correcting a large number of data values when an error is discovered in an algorithm used for calculating a column of numbers.

- Correcting an error in a processing step. For example, a new data release of a bathymetry grid is prepared after an error is detected in the processing step that applied tide corrections.
- Updating or changing the underlying authoritative data source.

When a Level 4 revision is needed to address a modification to the data structure, the following actions are required:

1. Create a new data release record in the IPDS and complete the review and approval steps for the new data release version. Review and approval should focus on the new or corrected sections but also identify any inadvertent changes made to other sections. The new IPDS record is used to ensure that the requirements of SM 502.7 and SM 502.8 have been met.

2. Create a new DOI for this new version.

3. Update the status of the DOI for the previous version in the USGS DOI Tool as follows:

a. Change the URL associated with the previous DOI to a web page (a 'tombstone URL') that explains the reason for the new version and provides the new DOI.
b. Update the Date information on the Supplemental Information tab of the DOI Tool as follows: change Date Type to 'withdrawn' and enter or update the date (YYYY-MM-DD) to designate the date that the data were removed from public access.
c. On the Supplemental Information tab, create a related identifier within the DOI records for the previous DOI and the new DOI, using the Relationship Type pair "Obsoletes/isObsoletedBy." In the record for the previous DOI, assign the relationship 'isObsoletedBy' and enter the URL for the new DOI. In the record for the new DOI, assign the relationship 'Obsoletes' and enter the URL for the previous DOI.

Note: there may be cases when it is appropriate to leave a previous version of the dataset accessible online, thus eliminating step 3. Consult your assigned BAO if you have questions.

When a Level 4 revision is needed to correct significant and substantial errors in the dataset the following actions are required:

1. Remove access to the data and metadata from the public landing page (for example, in a repository) and provide notice on the page to users that the data have been withdrawn.

2. Preserve the previous version of the data in accordance with records management and litigation holds requirements in case that version is needed to understand any information that was based on it. Refer to the "Archiving Prior Versions of Data" section for additional guidance.

3. Login to the USGS DOI Tool, and update the DOI for the original data release. Update the Date information on the Supplemental Information tab of the DOI Tool

as follows: add a Date Type ‘withdrawn’ and the date YYYY-MM-DD to designate the date upon which the data were removed from public access.

4. Create a new data release record in the IPDS and complete the review and approval steps for the new data release version. Review and approval should focus on the new or corrected sections but also identify any inadvertent changes made to other sections. The new IPDS record is used to ensure that the requirements of SM 502.7 and SM 502.8 have been met.

5. Create a new DOI for this new version. On the Supplemental Information tab, establish a Related Identifier linkage between this new DOI and the DOI for the withdrawn previous version. Assign the relationship ‘Obsoletes’ and enter the URL for the previous DOI.

6. Reopen the DOI of the withdrawn version of the data release. On the Supplemental Information tab, assign the relationship ‘isObsoletedBy’ and enter the URL for the new DOI.

7. Determine the version number for the revised data product. The change in the version number for Level 4 revisions is usually done by changing the number before the decimal point, for example, changing version 1.1 to version 2.0. Refer to the "Examples" section.

8. Revise the metadata record as follows:
   a. Add processing steps that describe the changes.
   b. Insert the version number and version release date into the title and recommended data citation.
   c. Update the metadata revision date.
   d. Add instructions for obtaining prior versions.
   e. Provide the revised metadata record to the USGS Science Data Catalog.

9. Create a new landing page for the new version of the data release:
   a. Include a list of version numbers and version release dates.
   b. Link to a revision history text file that provides a detailed description (for example, see ‘Version History 2.0’ link for data release https://doi.org/10.5066/F7542MHG) of the changes and a justification for making the changes.

10. Complete the new DOI with the Location URL of the new landing page, and publish the DOI.

11. Return to the landing page of the withdrawn data release. Provide a detailed description that gives information on the reason for the revision and uses the new DOI to point the user to the landing page of the new version of the data release.

12. If the revision could affect scientific conclusions in an existing USGS publication, consult your assigned BAO for guidance.

More About Version Numbering

Version numbers consist of two parts--a major component and a minor component, separated by a period. The original release is considered version 1.0, although the version annotation is not used if no subsequent versions are released. Either the major component or the minor component of the version number will be incremented when a new version is released.

In the example “version 1.2,” the number to the left of the period, “1,” is the major component and the number to the right of the period, “2,” is the minor component and represents the number of separate Level 2 revisions. Level 2 revisions, regardless of how many there are, do not initiate a change in the major component of the version number. For example, if the data release was revised on seven separate occasions for Level 2 revisions, the new version will be numbered “version 1.7.”

In the example “version 2.0,” a Level 3 revision was completed, and thus the major component number (“2”) was increased by one number and the minor component was reset to zero (“0”).

Preserving Prior Versions of Data

When data releases are replaced with a new version, the previous versions are not publicly offered but may be made available to users on request. Because previous versions may have been used to support scientific conclusions in a publication or a policy decision, it is essential to preserve them, for example in a dark archive (an offline location for preservation) or on an inaccessible page in a repository. The file name and accompanying documentation for previous versions should make clear that the data have been superseded. If frequent small revisions of large data files are anticipated, the science center or program should consider investing in an automated version management system that can automatically recreate each prior version by processing a standard revision history file, rather than manually archiving each version.

Examples

The following examples show various notations for documenting data revision changes on the data release landing page.

1. Examples of citation changes:

   Original citation:
   Klunk, O.T., 2012, Bathymetry of the Bermuda Triangle: U.S. Geological Survey data release, https://doi.org/10.5066/XXXXXXXX.

   Revised citations:
   Klunk, O.T., 2012, Bathymetry of the Bermuda Triangle (ver. 1.1, July 2012): U.S. Geological Survey data release, https://doi.org/10.5066/XXXXXXXX.

Klunk, O.T., 2012, Bathymetry of the Bermuda Triangle (ver. 2.0, May 2013): U.S. Geological Survey data release, https://doi.org/10.5066/XXXXXXXX.

Note that the data product title and DOI do not change but that version information is added. Additionally, the publication year should reflect the year that the original version was released. Include the new version number and version year in parentheses in the citation.

2. Examples of version release dates and version numbers:
   First release: 2012

   Revised: July 2012 (ver. 1.1)

   Revised: May 2013 (ver. 2.0)

3. Example of revision history:
   A revision history text file that concisely describes what changed in each revision is needed. For an example, refer to Pendleton, E.A., Ackerman, S.D., Baldwin, W.E., Danforth, W.W., Foster, D.S., Thieler, E.R., and Brothers, L.L., 2014, High-resolution geophysical data collected along the Delmarva Peninsula, 2014, USGS Field Activity 2014-002-FA (ver. 4.0, October 2016): U.S. Geological Survey data release.

16 BCO-DMO (#BCO-DMO)

Contributed by Danie Kinkaide

The Biological and Chemical Oceanography Data Management Office (BCO-DMO) works with investigators to serve data online from research projects funded by the Biological and Chemical Oceanography Sections, the Division of Polar Programs Arctic Sciences and Antarctic Organisms & Ecosystems Program at the U.S. National Science Foundation.

The BCO-DMO system is a data server plus a DSpace archive for data publication where data packages (timestamped, checksummed copy of the data, plus ISO metadata record and supplemental docs) are deposited.

To summarise, BCO-DMO curated data are:

- Served: http://bco-dmo.org (URLs, URIs)
- Published: at an Institutional Repository (WHOAS) http://dx.doi.org/10.1575/1912/4847
- Archived: at NCEI, a US National Data Center http://data.nodc.noaa.gov/cgi-bin/iso?id=gov.noaa.nodc:0078575

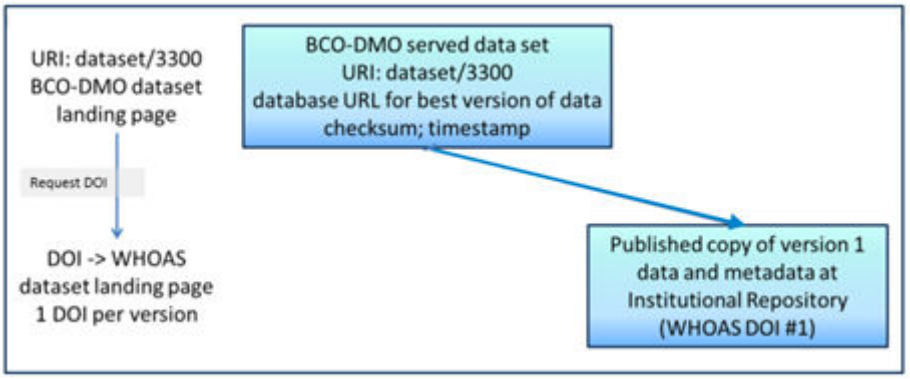


Figure 9: BCO-DMO data publication system components. Source: Chandler, C. *et al* (2016)

In 2016, BCO-DMO received funding from RD-A to implement the recommendations of the RD-A Data Citation Working Group. The 14 WG recommendations can be summarised as:

- ensure that data are stored in a versioned and timestamped manner
- identify data sets by storing and assigning persistent identifiers (PIDs) to timestamped queries that can be re-executed against the timestamped data store

The BCO-DMO evaluation of the recommendations found that R1-11 of the WG were a good fit with BCO-DMO architecture, R12 was regarded as doable, and R13-14 were consistent with Linked Data approach to data publication and sharing.

A primary driver for the BCO-DMO to implement the WG recommendations was to support citation of published data. As a result of the RD-A funded project, the following procedure is now invoked when a BCO-DMO data set is updated:

- A copy of the previous version is preserved
- Request a DOI for the new version of data
- Publish data, and create new landing page for new version of data, with new DOI assigned
- BCO-DMO database has links to all versions of the data (archived and published) Both archive and published dataset landing pages have links back to best version of full dataset at BCO-DMO
- BCO-DMO data set landing page displays links to all archived and published versions

The BCO-DMO identified five use cases for managing dataset DOIs for data versions.

- Use case 1: when new dataset is published with status = final
    - assign a DOI

  Noting that changes to the dataset that might result in different conclusions require a new version (timestamped, checksum) and a new DOI

- Use case 2: dataset is modified (columns added or removed)
    - mint/assign a new DOI in this case
    - create a new landing page for the new DOI, and link dc.related old and new one
- Use case 3: routine dataset extension over time (ie active time-series)

    - when adding new time range to dataset, inherit existing DOI
    - data replacement is not permitted, only extension in time
    - metadata temporal range is updated
- Use case 4: update to metadata only (eg typos corrected)
    - handle is appended with .1 in the local repository; DOI does not change
- Use case 5: minor replacement (fixes, adjustments, format) within a dataset (# sig digits)
    - data object modified as needed; small changes
    - internal version control (new version date), update metadata to clearly reflect changes
    - DOI remains the same
    - version 1.0 gets a DOI
    - new version declared if different science result
    - new columns etc., different conclusions – new landing page

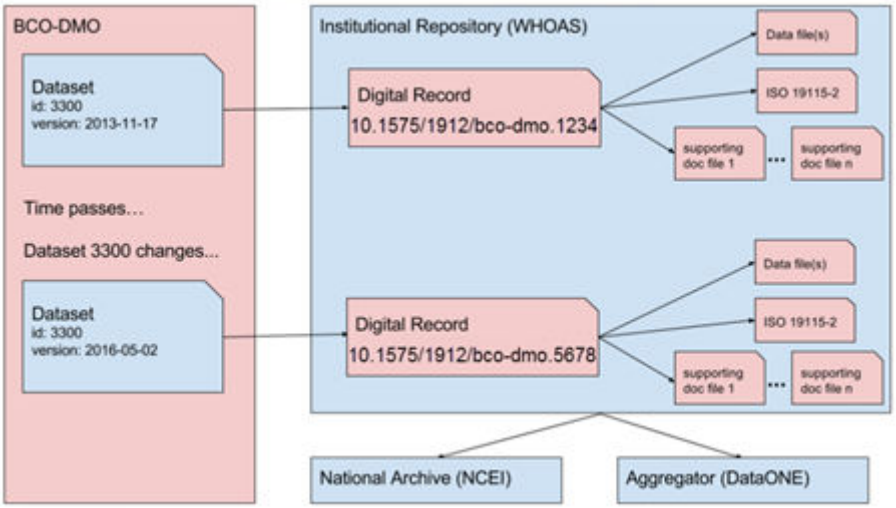


Figure 10: BCO-DMO data citation system components. Source: Chandler, C. *et al* (2016)

References

Chandler, C., Shepherd, A., Bassendine, D. (2016) *Adoption of Data Citation Outcomes by BCO-DMO.* Presentation to RDA 8th Plenary Data Versioning BoF meeting, Denver Colorado.

17 NASA: EOSDIS and SEDAC (#NASA)

Contributed by Bob Downs

The Earth Observing System Data and Information System (EOSDIS) is a core capability in NASA's Earth Science Data Systems (ESDS) Program. It provides end-to-end capabilities for managing NASA's Earth science data from various sources – satellites, aircraft, field measurements, and various other programs.

EOSDIS does not use a common standard to number versions

- Diverse data producers (instrument or science teams) contribute data to the NASA Distributed Active Archive Centers (DAACs)
    - Designate versions in their own ways
    - Use various terms: Version 1, 2; Collection 1, 2; Release 1, 2; Edition 1, 2
- Number of versions vary, depending on volumes from instrument
- Absence of standard numbering scheme for versions is not problematic
    - each data set title is identified by a particular version
    - software version that generated data is identified, when applicable
    - provenance is tracked independent of a standard numbering scheme
- Version information is recorded in the data metadata
- New DOIs are assigned as new versions are generated
    - Landing pages reference older versions if they exist
    - Landing pages for superseded versions will persist and refer to newer versions

The Socioeconomic Data and Applications Center (SEDAC) is a data centre within EOSDIS. It has established the following practice for versioning data products

- Establish titles with consistent designations for version or year
    - Uniquely identify each edition of collection, dataset, or product
    - Unique aspects can include year or range of dates for observations
- Default version number is 1; implied if not stated explicitly
    - New version number is assigned if a dataset is changed
    - Subsequent versions are Version 1.01, Version 2, or Revision 01
- New collection version reflects new stage of development
    - Collection versions are assigned as integers
    - New collection supersedes previous collection

PIDS at SEDAC

SEDAC uses the following guidelines for assigning global persistent identifiers:

- assigned to the landing page for each dataset disseminated by SEDAC
- included in the recommended citation for each dataset
- recorded and maintained to identify current location and optimize discovery

Their procedure for assigning global persistent identifiers:

- DOIs assigned to datasets and documentation; software and services may be next
- assigned using EZID and the DataCite Metadata Schema
- DOIs also recorded in the FGDC CSDGM (Federal Geographic Data Committee Content Standard for Digital Geospatial Metadata)
- Related Identifier field is used to link to other data, documentation, publications
- DOI record is modified when location of landing page changes

18 Australian Bureau of Meteorology (#BoM)

Contributed by Martin Schweitzer, edited by Ben Evans

The Australian Bureau of Meteorology (BoM) is Australia's national weather, climate and water agency. BoM collects observational data and produces a lot of downstream data products. There are requirements to go back to the point of time a data product (as a version) was created, be able to link this data product all the say back to the relevant raw datasets and processing pipelines.

The BoM uses a relational database to manage some observational data. The database for climate data is called ADAM (Australian Data Archive for Meteorology). ADAM stores observations by daily, hourly and minute. Data comes in two types: points of data and gridded data. Data comes from various sources such as stations - about 600 stations that records data per minute (temperature, rain, humanity, wind, etc. about 12 variables), paper records from farmers - registered farmers send rain gauge to BoM, BoM then deposit the data into ADAM. Currently, the ADAM holds over 120 years of records.

The BoM applies QC consistently for any inconsistency by station and time series etc. All changes are recorded and time-stamped in an audit table. Data products made available from the BoM web portal are identified by Product Code, State Date (date to be forecasted, or time where an observation was made) and Product Issue Date. The combination of the three “IDs” defines a “version”, it can be traced back in time where the data product was produced and can be reproduced if required.

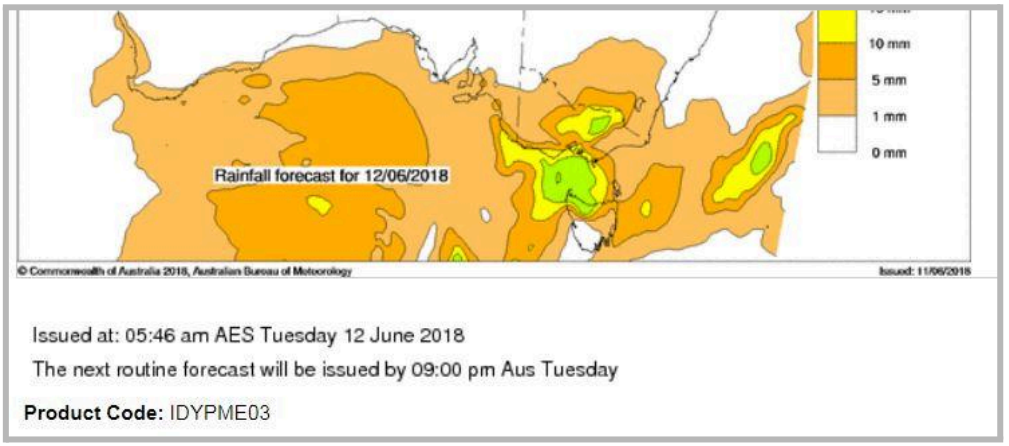
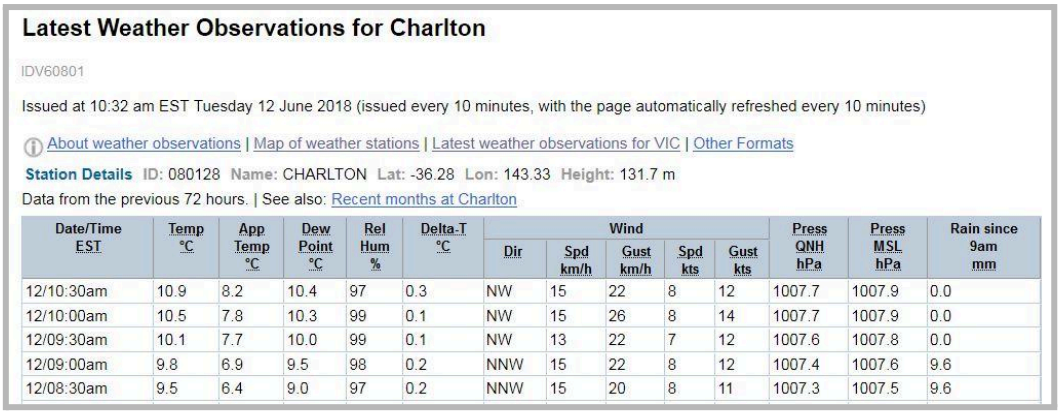


Figure 11. BoM forecast data product



Latest Weather Observations for Charlton

IDV60801

Issued at 10:32 am EST Tuesday 12 June 2018 (issued every 10 minutes, with the page automatically refreshed every 10 minutes)

About weather observations | Map of weather stations | Latest weather observations for VIC | Other Formats

Station Details ID: 080128 Name: CHARLTON Lat: -36.28 Lon: 143.33 Height: 131.7 m

Data from the previous 72 hours. | See also: Recent months at Charlton

Date/Time EST	Temp °C	App Temp °C	Dew Point °C	Rel Hum %	Delta-T °C	Wind					Press QNH hPa	Press MSL hPa	Rain since 9am mm
						Dir	Spd km/h	Gust km/h	Spd kts	Gust kts			
12/10:30am	10.9	8.2	10.4	97	0.3	NW	15	22	8	12	1007.7	1007.9	0.0
12/10:00am	10.5	7.8	10.3	99	0.1	NW	15	26	8	14	1007.7	1007.9	0.0
12/09:30am	10.1	7.7	10.0	99	0.1	NW	13	22	7	12	1007.6	1007.8	0.0
12/09:00am	9.8	6.9	9.5	98	0.2	NNW	15	22	8	12	1007.4	1007.6	9.6
12/08:30am	9.5	6.4	9.0	97	0.2	NNW	15	20	8	11	1007.3	1007.5	9.6

Figure 12: BoM observational data product

19 Australian Integrated Marine Observing System (#IMOS)

Contributed by Natalia Atkins

Since 2006, IMOS has been routinely operating a wide range of observing equipment throughout Australia's coastal and open oceans, making all of its data accessible to the marine and climate science community, other stakeholders and users, and international collaborators. There are five major research themes that unify IMOS science plans and related observations:

- Long-term ocean change;
- Climate variability and weather extremes;
- Boundary currents;
- Continental shelf and coastal processes; and
- Ecosystem responses.

Most of IMOS data are dynamic: new data is continuously added, existing data can be modified or updated. Data are file based, e.g., in netCDF, stored in databases, and there are also data types such as AUV images and acoustic recordings. Datasets vary in size from a few 1000 rows in a database to 20TB of satellite data.

Data is quality checked before being sent to IMOS but may be corrected or reprocessed several times. When new versions are published, the previous version is archived (except for satellite data). Therefore it is possible that a disparity may occur between data previously accessed and cited, and the data that is currently available.

For data stored on Amazon S3 (object storage), they use the versioning feature from the S3 object storage to keep all previous versions (except for satellite data). The version is identified by date and time, and if "versioning" is enabled, assigned a randomly generated Version ID (versions are "linked" by virtue of having the same file name). This version information is not publicly viewable, and users have to contact IMOS for access to. For data in netCDF file format, (change) history is captured with a file.

IMOS advises their data consumer to cite their data as follows:

> IMOS [year-of-data-download], [Title], [data-access-URL], accessed [date-of-access].

20 Australia Astronomy Observatory Data Centre (#AAO)

Contributed by Simon O'Toole

The All-Sky Virtual Observatory (ASVO) is a federated system of astronomical data nodes. There are five nodes of the ASVO: the AAO stores optical astronomy data at the AAO; ANU/Mt Stromlo stores images from the SkyMapper survey at NCI; MWA-ASVO stores data from the Murchison Widefield Array (MWA) and CSIRO/CASDA stores data from the ASKAP telescope. The last two are radio telescope pathfinders for the SKA, and their data are stored at the Pawsey Supercomputing Centre. There is also the Theoretical Astrophysics Observatory, which generates and stores numerical simulations at Swinburne University.

Data types: images, spectra, image-spectral data cubes, raw visibility data (radio interferometry)
Data size: ~2 petabytes of optical, >12 petabytes radio, 100s terabytes theory
Data formats/models: FITS, HDF5, PostgreSQL, Hadoop/Spark
Data access: 1) web UI, 2) third-party VO apps, 3) APIs
Data ingest: mostly dynamic, but only released publicly at discrete time points, e.g. yearly observation

The ASVO stores optical imaging data at the National Computational Infrastructure (NCI), and other types of optical data at the AAO. Every version is stored, but only published versions are publicly accessible. DOIs will be attached to data on an ongoing basis soon.

Radio data from ASKAP (Australian Square Kilometre Array Pathfinder) and the MWA are stored at Pawsey, each version is kept. Data process is an ongoing activity, old versions may be rewritten. Improvement of data quality is driving data versioning. New versions may come from different calibrations, with improved calibration pipelines.

General approach to versioning AAO:

- Data Release: new DOIs for each public data release (manual validation & release process). E.g. _v01 to first version, _v02 to 2nd version.
- Old versions of data/branches remain available, but the current version is the default.

21 Digital Earth Australia (#DEA)

Contributed by Simon Oliver

Digital Earth Australia (DEA) is a platform that uses spatial data and images recorded by satellites orbiting the planet to detect physical changes across Australia in unprecedented detail. Using high performance computing power provided by the National Computational Infrastructure (NCI) and commercial cloud computing platforms, DEA organises and prepares satellite data into stacks of consistent, time-stamped observations that can be quickly manipulated and analysed to provide information about a range of environmental factors such as water availability, crop health and ground cover.

Satellite earth observation data are highly structured and stored as large to very large binary data files, each of which may contain gigabytes or even terabytes of data. The data are diverse and dynamic with new data being continuously added and/or existing data being updated (e.g., as calibration algorithms improve over time, errors are found in existing data, etc.)

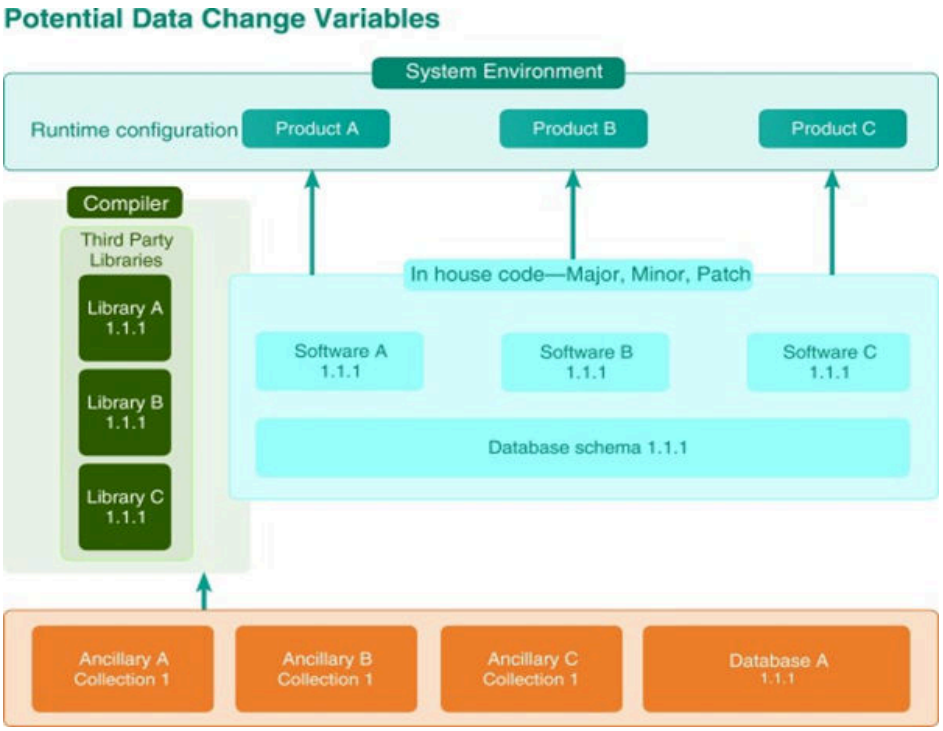


Figure 13. Inputs to processing of Earth observation raw archives to produce products. Source: Lewis, A. *et al* (2017)

Figure 1 shows how multiple ancillary inputs are used which change from time to time as quality is improved. Software code libraries, which may be internal to the processing agency or from third party providers, have specific versions. The hardware and operating system environments are also significant and are frequently upgraded. Finally, products often are chained, with one product being an input to the next, for example as higher levels of correction are produced.

DEA uses the concept of Managed Collections to address these challenges. The approach differs significantly from other current models of processing which often use

common systems of differing versions across multiple platforms to produce “like” products.

The concept of a managed collection includes:

- Software versioning, and governed production software upgrades;
- Ancillary input collection versioning and update control;
- Assessment of the scope and significance of a proposed change for the collection. Scope refers to the proportion of the collection affected, whereas significance is established through comparison with benchmarks and acceptable deviations from these in regard to radiometric and geometric changes, for example;
- Business processes, which determine a course of action based on the scope and significance of the change, for example to:
    - Add to a backlog until significance and scope reach thresholds;
    - Upgrade the collection;
    - Update components of the collection; and
    - Patch components of the collection.

Underpinning the approach is a three level hierarchy major.minor.patch convention in line with the semantic versioning scheme commonly in use in the IT domain. Data collections and their subcomponents are attributed via this convention to enable patch and repair of various components of the collection. The scheme allows for variation within a data collection but enables management practices to enact a virtual self healing methodology. Decisions on the major/minor/patch attribution to a dataset are made via a Change Control Board assessment of the scope and significance of a proposed change to the collection (whether it be to software, systems, data or metadata).

Code	Assessment of change in data	Software collection	Data collection
From v1.0.0	Significant	v1.0.0	v1.0.0
To v2.0.0		v2.0.0	v2.0.0
From v1.0.0	Insignificant/ minor	v1.0.0	v1.0.0
To v2.0.0		v1.1.0	v1.1.0
From v1.0.0	Significant	v1.0.0	v1.0.0
To v1.1.0		v2.0.0	v2.0.0

Ancillary Data	Assessment of change in data	Software collection	Data collection
From v1.0.0	Significant	v1.0.0	v1.0.0
To v2.0.0		v2.0.0	v2.0.0
From v1.0.0	Insignificant/ minor	v1.0.0	v1.0.0
To v2.0.0		v1.1.0	v1.1.0

Figure 14. Collection numbering approaches adapted from software versioning. The numbering of versions depends upon the scope and significance of each change. Source: Lewis, A. *et al* (2017)

References

Lewis, A., Oliver, S., Lymburner, L. et al (2017) *The Australian Geoscience Data Cube : Foundations and lessons learned*. Remote Sensing of Environment, vol 202, pp.276-292. https://doi.org/10.1016/j.rse.2017.03.015

22 Geoscience Australia: Enterprise metadata catalogue (#GA-EMC)

Contributed by Martin Capobianco and Andy Marshall

Geoscience Australia (GA) is Australia's pre-eminent public sector geoscience organisation. GA is the government's technical advisor on all aspects of geoscience and custodian of the geographical and geological data and knowledge of the nation. GA manages hundreds of thousands of datasets, collected over many years. Many of the datasets are publicly accessible via the eCAT catalogue. While there is no formal, documented approach to versioning within GA, there are standard approaches that are applied.

Example 1: a database snapshot is captured and published.
SHRIMP U-Pb Geochronology Interim Data Release July 2007
http://pid.geoscience.gov.au/dataset/ga/65358

The data published here are a snapshot of the database at the "Ending-Date", although entry into the database is continuous. The snapshot was created as an interim form of data release while web-based accessibility to the continually updated database was explored.

The Lineage statement in the catalogue record explains: *These data are derived directly from Geoscience Australia's corporate Oracle OZCHRON database for U-Pb ages derived using the SHRIMP method. An ASCII extraction of the database is generated as ASCII comma separated values (CSV)*

The metadata record offers a File Download link to a zip file that includes:

- Copyright txt
- Data dictionary pdf
- Geochron data extract as at 27 July 2007 xls
- Metadata txt

http://pid.geoscience.gov.au/dataset/ga/79134 (14th ed)
http://pid.geoscience.gov.au/dataset/ga/74888 (13th ed)
http://pid.geoscience.gov.au/dataset/ga/72767 (12th ed)
This form of snapshot data release is not commonly practiced within GA.

Example 2**:** a new metadata record is created for each version. Public access to older versions is retained.

Index of airborne geophysical surveys

A new edition or version of the Index is published annually. A new metadata record is created for each new edition. The version is indicated by the edition number.

The Lineage statement in the catalogue record for the 14th edition explains: *This Record is published as the thirteenth in a series of GA Records which contain regularly updated information as the specifications of surveys already completed are incorporated and as new surveys are added to the National Airborne Geophysical Database (ARGUS Oracle database). This version of the Index includes details of surveys completed since the previous edition in January 2013 as well metadata for new surveys.*

The catalogue record for the 14th edition provides 3 links to related products.
One link leads to this dataset:
Digital files for the Index of airborne geophysical surveys, Fourteenth edition, 2014.

The Lineage statement for the Digital files dataset explains:
This dataset supercedes the previous version of the product released in January 2013 (Geocat #74888) and earlier versions released in October 2011 (Geocat #73075), May 2004 (#61337), May 2003 (#47656), June 2002 (#40757), June 2001 (#36834) & October 2000 (#35181).

No similar explanation is provided in the Lineage statement for the Index itself.

A DOI has been assigned to the 14th edition, but is not visible in the default metadata record view in the catalogue.

Separate catalogue records exist for the 12th and 13th editions of the Index with information provided in Description and Lineage fields updated accordingly. No link to a later version of the Index is provided in the catalogue records for 12th and 13th editions. DOI have not been retrospectively assigned to 12th and 13th editions.

An eCAT search for “Index of airborne geophysical surveys” returns results for several versions of the Index. There appears to be no weighting to the search results that elevates the most recent version.

Example 3: a single metadata record, and DOI, for a dataset subject to update.
Electricity Transmission Lines
http://pid.geoscience.gov.au/dataset/ga/83105

The National Electricity Transmission Lines dataset presents the spatial location, in line format, of all known high voltage electricity transmission lines that make up the electricity transmission network within Australia.

A DOI has been assigned to the catalogue metadata record for the dataset, not the dataset being described. No version number appears in the catalogue record.

The Lineage statement in the catalogue record explains:
The electricity transmission lines were digitized in 2011 from the library of imagery held within Geoscience Australia. Imagery used ranged from 0.15m resolution aerial photos to 2.5m resolution satellite images. The database was revised in January 2014 to

reflect the most current version of the Australian Energy Market Operator (AEMO) Transmission Network Diagrams dated 14 February 2013.

More detail is provided in the Metadata Statement pdf which can be downloaded from the catalogue record.

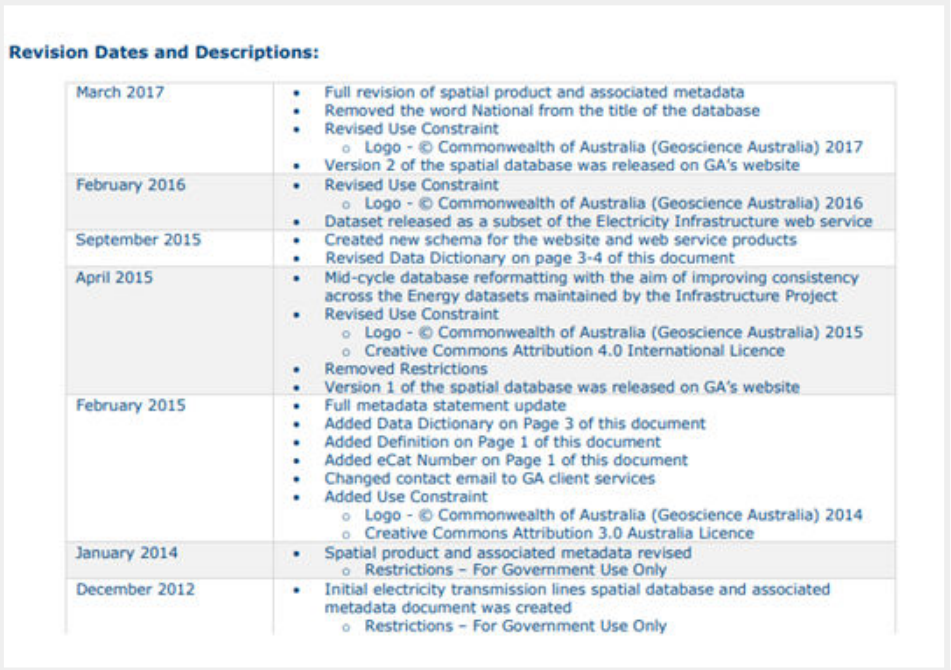
The Metadata Statement pdf is titled:
Electricity Transmission Lines Database Metadata Statement
Version 2 Last updated in 2017 eCatID: 83105

The lineage statement in the Metadata Statement pdf states:

> *The electricity transmission lines were digitized in 2011 from the library of imagery held within Geoscience Australia. Imagery used ranged from 0.15m to 2.5m resolution. The electricity transmission lines dataset was revised (Version 2) in March 2017 using Esri World Imagery. Version 1 of the database was first released publicly on Geoscience Australia's website in April 2015 and the updated revision re-released as Version 2 in March 2017 The electricity transmission lines web service – Version 1 was released as a subset of the Electricity Infrastructure web service in February 2016.*

A full revision history is also provided in the Metadata Statement pdf

The catalogue record links to the current version of the data. A user must read the Metadata Statement pdf to determine the version number of the dataset and when the dataset was last updated.



Revision Dates and Descriptions:

March 2017	• Full revision of spatial product and associated metadata • Removed the word National from the title of the database • Revised Use Constraint ○ Logo - © Commonwealth of Australia (Geoscience Australia) 2017 • Version 2 of the spatial database was released on GA's website
February 2016	• Revised Use Constraint ○ Logo - © Commonwealth of Australia (Geoscience Australia) 2016 • Dataset released as a subset of the Electricity Infrastructure web service
September 2015	• Created new schema for the website and web service products • Revised Data Dictionary on page 3-4 of this document

April 2015	• Mid-cycle database reformatting with the aim of improving consistency across the Energy datasets maintained by the Infrastructure Project
	• Revised Use Constraint
	  ○ Logo - © Commonwealth of Australia (Geoscience Australia) 2015
	  ○ Creative Commons Attribution 4.0 International Licence
	• Removed Restrictions
	• Version 1 of the spatial database was released on GA's website

February 2015	• Full metadata statement update
	• Added Data Dictionary on Page 3 of this document
	• Added Definition on Page 1 of this document
	• Added eCat Number on Page 1 of this document
	• Changed contact email to GA client services
	• Added Use Constraint
	  ○ Logo - © Commonwealth of Australia (Geoscience Australia) 2014
	  ○ Creative Commons Attribution 3.0 Australia Licence

January 2014	• Spatial product and associated metadata revised ○ Restrictions – For Government Use Only
December 2012	• Initial electricity transmission lines spatial database and associated metadata document was created ○ Restrictions – For Government Use Only

Figure 15. GA revision history in metadata statement

23 Geoscience Australia: Earthquake Seismic Data (#GA-ESD)

Contributed by Margie Smith

Geoscience Australia (GA) receives real-time data from over 60 seismic stations in Australia and more than 130 international seismic stations. The seismic information is automatically analysed by Geoscience Australia's seismic monitoring and analysis systems that form part of the 24 hours a day, seven days a week operations centre.

According to the *National Archives of Australia Records Disposal Authority for Geoscience Australia* (2005), GA has a legal requirement to retain:

- Records documenting advice/technical advice provided to government agencies on potentially damaging earthquakes and tsunamigenic events.
- Records documenting advice/technical advice provided by the agency on earthquakes and engineering seismology to standards bodies, insurance industry, public.

In order to satisfy this requirement, at the time such advice is provided, GA must capture a snapshot of the entire seismic dataset, package it with related inputs and store it in the Corporate Data Store for retention in line with NAA requirements. This snapshot is regarded as a version of the database.

The models below shows the process developed by GA to meet NAA requirements. A proof-of-concept implementation has successfully been undertaken to test the process.

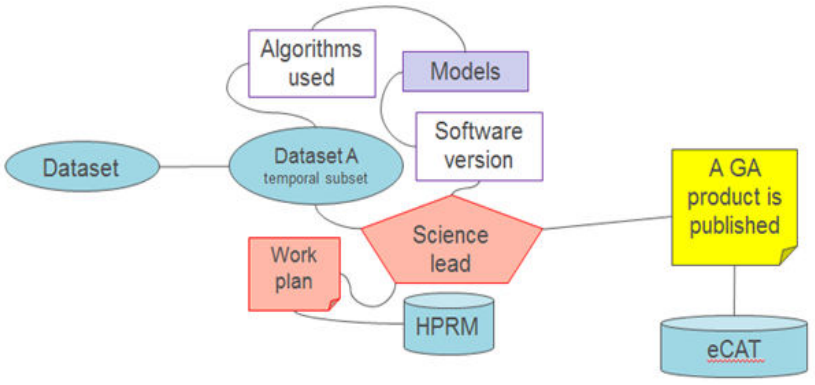


Figure 16. The problem was how to capture all the associated information related to a packaged product

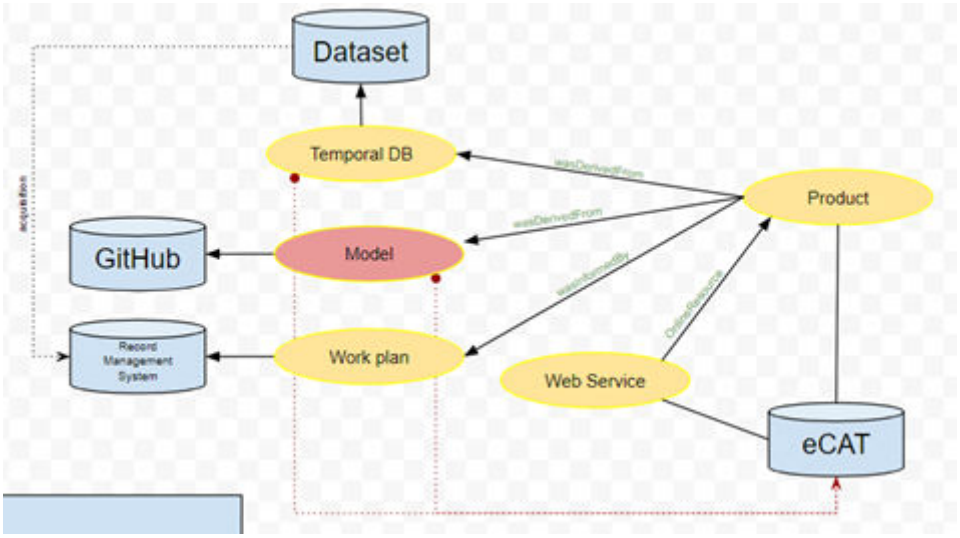


Figure 17. A model was developed and successfully tested

24 ESGF Climate Model Data (#CMIP6)
Contributed by Kate Snow

The purpose of CMIP (Coupled Model Intercomparison Project) is to provide state-of-the-art multi-model advancements coordinated at an international level to improve our understanding of past, present and future climate change. The CMIP data forms an important component of the high-end climate research that is assessed as part of the Intergovernmental Panel on Climate Change (IPCC) and to inform policy makers in making evidence-based decisions in relation to current and future climate change. The CMIP data is managed through the Earth Systems Grid Federation (ESGF).

For CMIP6 data, version labels are standardized to be vYYYYMMDD and the version date is the publication date.

An already published version of a publication unit must not be changed. This means no addition, deletion or replacement of files which are part of a publication unit. Any change must lead to a new version.

A new version can only be created on discovery of an errata and if justification is given for the requirement of a new version. Publication of a newer version of a dataset needs to have a valid motivation, which is referred to as an issue.

The ESdoc-errata project created an issue tracker platform for CMIP6 to keep track of the issues affecting specific versions of datasets/files. It enables users to resolve the history tree of each dataset/file.

It is recommended that the unit of versioning be an atomic dataset: a complete time-series of one variable from one experiment and one model. The implication is that

other variables need not be republished, if the error is found in a single variable. If an entire experiment is retracted and republished, all variables will get a consistent version number.

Old versions do not need to be published and PIDs are important in the version management. For example, the data might be no longer available in a certain version as it gets revised and published under a new version, but the information on its previous version remains (a PID on such a file should point to a tombstone page). Ideally, the PID target page for the old and unpublished data version should include errata information and provide a link to the latest (revised) data version.

There is a requirement that at least one instance of each submitted dataset be stored at an ESGF Tier 1 node (in addition to its primary residence) within a reasonably short time period following submission. Most Tier1 nodes will maintain a replica of the most highly used datasets by climate researchers.

The description below of the CMIP abstract versioning workflow and version domains is drawn from this discussion document that has not yet been officially ratified, but is reproduced here as an example of one approach.

DRAFT Abstract versioning workflow:

Initial version publication:

pre-condition: unversioned, complete CMIP6 publication units in agreed CMIP6 directory structure

pre-condition: version-string in agreed format

pre-condition: map file for publication units (including version information, generated by esgscan_directory command or locally developed software - but same format of map files, format definition at ...site…)

- (major storage sites: add version-info in directory tree of the new CMIP6 publication units)
- merge with existing “ESGF accessible” (thredds accessible) storage pool
- publish in ESGF

post-condition: local CMIP6 data pool including versioning information reflected in directory structure

post-condition: ESGF published thredds catalogs reflecting version

post-condition: ESGF solr index with latest etc. version information

post-condition: PID metadata reflecting published units as “latest” version

New Version:

pre-condition: complete new CMIP6 publication units in agreed CMIP6 directory structure

pre-condition: version string in agreed format, indicating “newer” version

pre-condition: separate “annotation” describing the background of the version change:

e.g. reason, reference to additional info, changes made etc.

- check version (existence, version format, greater than previous) and check checksums in case of “same version publication actions” (broken publication activities requiring “same version” publications)
- add version-info in directory tree(s) of CMIP6 publication units
- merge with existing “ESGF accessible” storage pool
- publish in ESGF

post-condition: local CMIP6 data pool including versioning information reflected in directory structure

post-condition: ESGF published thredds catalogs reflecting version

post-condition: ESGF solr index with latest etc. version information

post-condition: PID metadata including link to previously published version

post-condition: versioning related annotation (optionally “basic stub”) published alongside new version

Addition of an old version of publication unit:

pre-condition: an old version of a publication unit is to be published. (E.g. re-publishing after a fault or upgrade).

pre-condition: a more recent version of the publication is already published to ESGF.

pre-condition: complete new CMIP6 publication units in agreed CMIP6 directory structure

pre-condition: version string in agreed format, indicating “newer” version

pre-condition: separate “annotation” describing the background of the version change:

e.g. this might include information about the newer version.

- check version (existence, version format, greater than previous) and check checksums in case of “same version publication actions” (broken publication activities requiring “same version” publications)
- add version-info in directory tree(s) of CMIP6 publication units
- merge with existing “ESGF accessible” storage pool
- publish in ESGF

post-condition: local CMIP6 data pool including versioning information reflected in directory structure

post-condition: ESGF published thredds catalogs reflecting version

post-condition: ESGF solr index with latest etc. version information

post-condition: PID metadata including link to previously published version AND newer versions

post-condition: versioning related annotation (optionally “basic stub”) published alongside new version

Version retraction:

pre-condition: identification of CMIP6 publication units to be retracted

(including version info)

pre-condition: separate “annotation” describing the background of the version retraction:

e.g. reason, reference to additional info, changes made etc.

- un-publish in ESGF, including publication of “annotation”
- adapt the “latest” link in the storage pool (if relevant)

post-condition: unchanged local CMIP6 data pool

post-condition: updated ESGF published thredds catalogs removing version inf

post-condition: updated ESGF solr index removing involved versions

post-condition: PID metadata updated reflecting new version chain

(re-publication of retracted data = new version + annotation !)

post-condition: versioning related annotation (optionally “basic stub”) published alongside new version

Version removal:

pre-condition: complete set of CMIP6 publication unites to be unpublished

pre-condition: separate “annotation” describing the background of the removal

e.g. reason, reference to additional info, changes made etc.

- remove involved CMIP6 publication units in storage pool (e.g. adapting “latest” link”)
- un-publish in ESGF (including publication of annotation)

post-condition: changed publication units from ESGF storage pool

- adapt versioning information CMIP6 data pool with involved publication units removed

post-condition: new ESGF published thredds catalogs reflecting removal

post-condition: new ESGF solr index reflecting removal

post-condition: PID metadata updated with indication that version was removed

(object permanently unavailable)

post-condition: versioning related annotation (optionally “basic stub”) published alongside new version

Versioning domains:

Thus different “domains” or levels of versioning can be separated:

A. Versioning of datasets (and individual files) at the storage level (reflecting versioning info on the file system level e.g. by consistently maintaining soft/hard links)
B. Versioning of ESGF published datasets at the ESGF infrastructure level (ESGF metadata in thredds and solr consistently searchable etc.)
C. Versioning of datasets (and individual files) at the PID infrastructure level: PID metadata associated to PID based tracking ids (and collection ids) contains versioning information (links to predecessor/successor PIDs)

Sources (and with thanks to Kate Snow, NCI)

CMIP6 versioning requirements collection

CMIP6 Data Citation and Long-Term Archival
Requirements for a global data infrastructure in support of CMIP6 (Section 7)

25 CSIRO Data Access Portal (#CSIRO)

Contributed by Dominic Hogan

The Commonwealth Scientific and Industrial Research Organisation (CSIRO) is Australia's national science agency and one of the largest and most diverse research agencies in the world. The CSIRO Data Access Portal (DAP) provides access to research data, software and other digital assets published by CSIRO across a range of disciplines.

In the DAP, a new version is created whenever a dataset or its metadata record are changed. A new landing page is created, and a new DOI may be assigned. The criteria to mint a new DOI involves a substantial change to the metadata fields that make up the attribution statement and/or adding or deleting files. Metadata updates that do not substantially affect the attribution statement, and where no change is made to the data, will not receive a new DOI. Instead, the last DOI for that exact version of the data will redirect to the most recent metadata record. In all cases, the version number is automatically included in the citation statement.

Any alteration to a DAP collection is recorded accurately through the use of version control. Changes to metadata and/or files in the DAP create a new version. The previous file(s), Archival Information Packages (AIPs) and Dissemination Information Packages (DIPs) are retained. The current version is returned in query results.

Software published via the DAP is assigned a DOI. Depositors are advised that best practice is to use the DAP to publish major releases and to make their code repository accessible and linked if they wish to make minor releases available.

To create a new release for software already in the DAP, users can update the record with the new files and the DAP will automatically create a new DAP version and assign a new DOI, with access to the previous version retained.

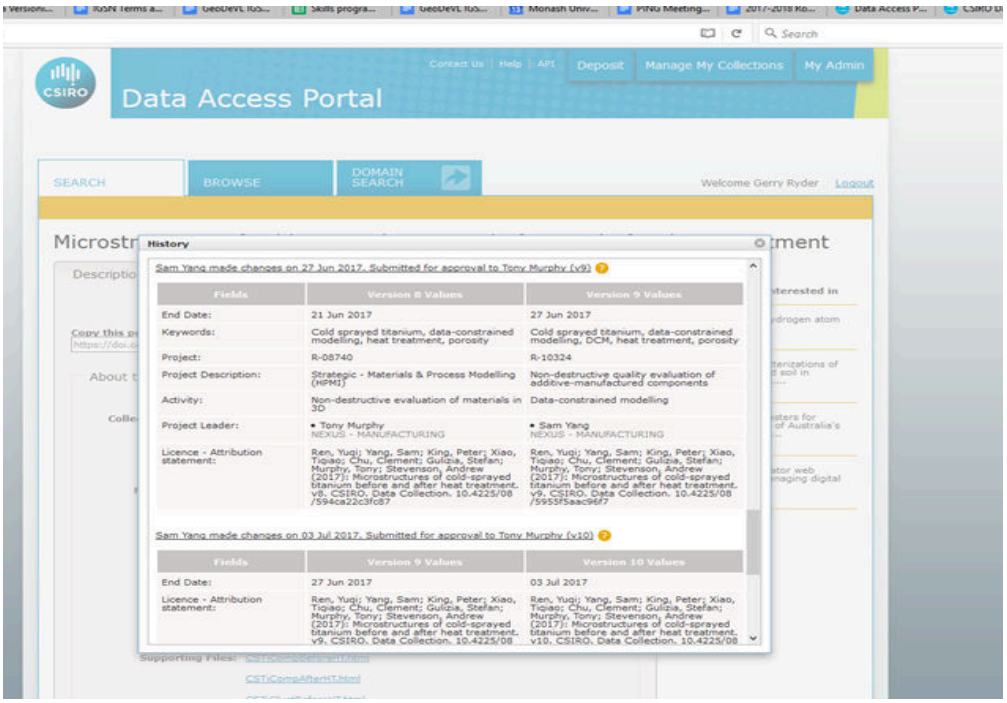


Figure 18. A logged in user can view the full version history of a collection

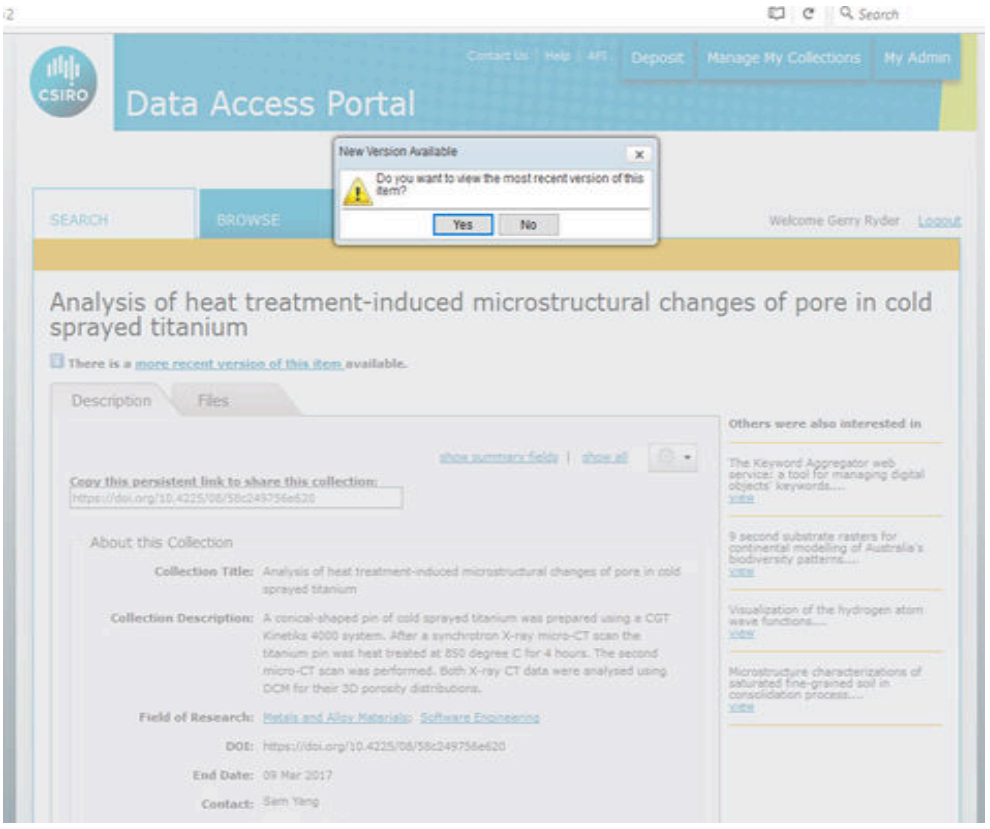


Figure 19. A user that resolves the DOI for a ‘previous’ version is alerted to the most current version

26 NLA TROVE Government Gazettes Collection on CloudStor (#NLA)

Contributed by Catherine Brady & Julia Hickie

A gazette is an official publication for the purpose of notifying the public of government business. All Australian governments (Commonwealth, State and Territory) publish official gazettes.

Notices published in government gazettes cover all aspects of government concern and regulation, and most are published because of a requirement of law. Acts, regulations and other subordinate legislation are notified in all gazettes, with some states publishing regulations in full as part of the notification.

Background

The National Library of Australia provides access to a digitised collection of gazettes via the TROVE service. Broadly, the digitisation process involves a 6 step process represented in the diagram below.

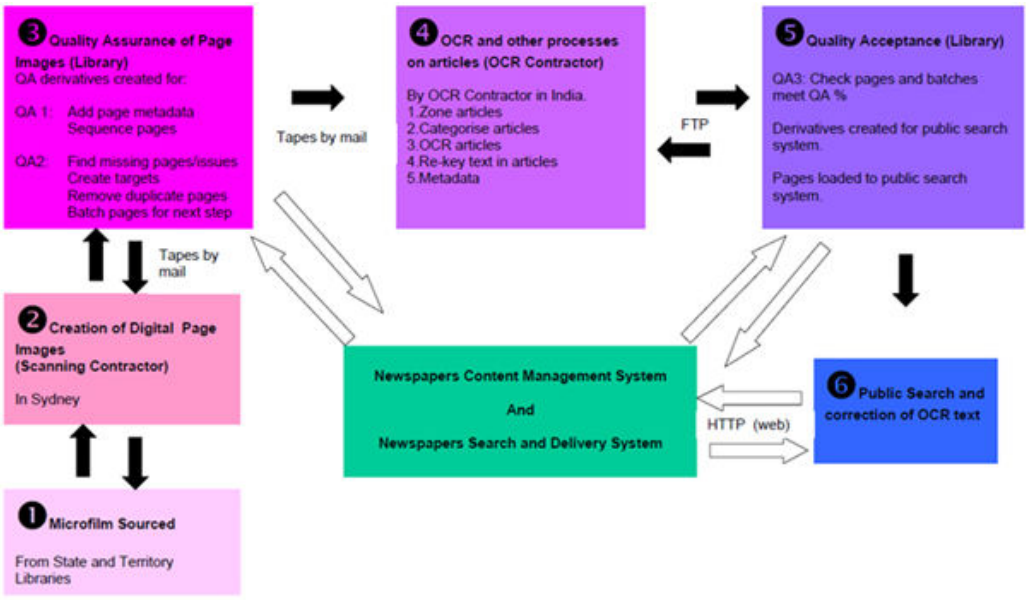


Figure 20. Trove digitalisation process[2]

The dataset published via TROVE is comprised of scanned page images which are displayed alongside the OCR text for the page. Each article in the dataset has metadata added to indicate Title, Issue number, Pagination and more.

Once published on TROVE, members of the public can correct errors in the OCR text and may add comments and tags such as keywords that can be viewed by opening a dialogue box in the user interface. These changes are saved in the database and available to others. There are already millions of lines of text corrected representing millions of changes to the original published dataset.

Gazettes on Cloudstor

[2] http://help.nla.gov.au/trove/for-digitisation-partners/digitisation-workflow-process-overview

A current project, run under the auspices of the HASS DeVL project, is to upload a snapshot of the OCR text from a subset of the TROVE Government Gazettes database to Cloudstor. The intent is to make the OCR text more accessible and interoperable for research purposes to enable for example, text mining and analysis. As of March 2018, the Cloudstor dataset provides access to:

- NSW Government Gazette 1832-1900;
- Government Gazette of NSW 1901-1968;
- Commonwealth of Australia Gazette 1901-1969

The workflow for achieving this is illustrated below.

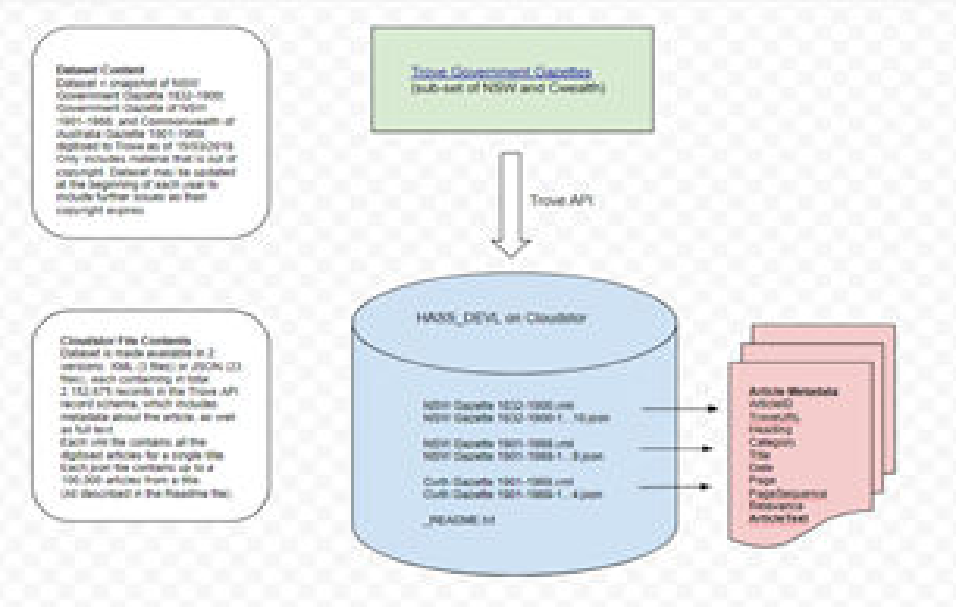


Figure 21. The workflow of archiving OCRed data

On Cloudstor, the dataset is made available in 2 file formats

1. XML (3 files) containing all the digitised articles for a single (Gazette) title
2. JSON (23 files) containing up to 100,000 articles from a (Gazette) title

All are described in the readme file.

Each format provides access to more than 2 million records in the TROVE API record schema which includes metadata about the article, as well as the full text.

The screenshot below from Cloudstor shows the file structure of the dataset.

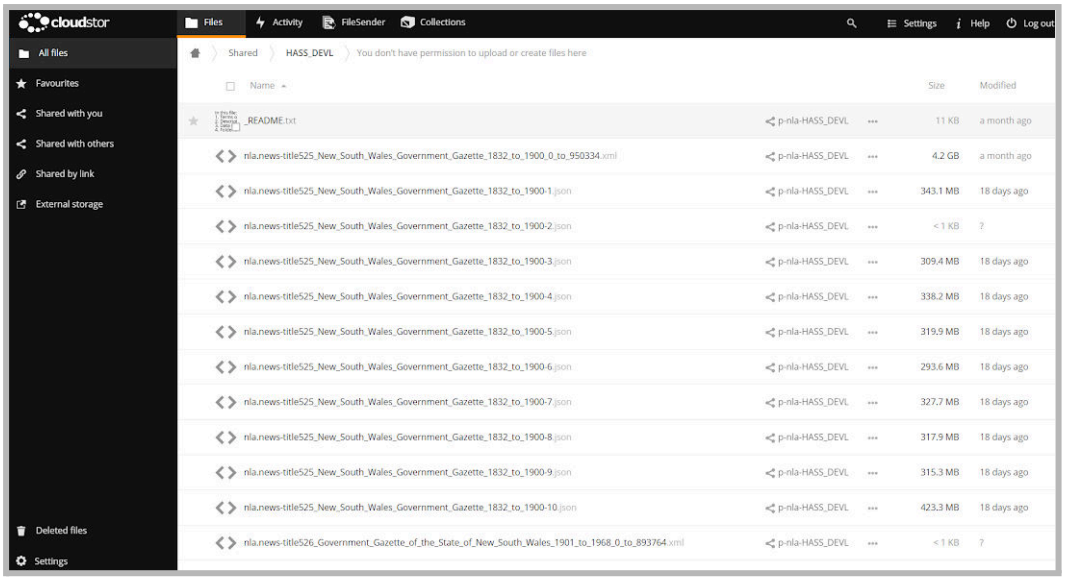


Figure 22. TROVE data file explorer

Versioning challenges in the Cloudstor collections

The Cloudstor collections are a snapshot of the evolving dataset on TROVE. The dataset evolves in two ways:

1. Frequent updates to the OCR text in TROVE are made as members of the public annotate and correct the OCR text.
2. The size and time coverage of the dataset will increase each year as later issues of the Gazettes are digitised as their copyright expires.

There is likely to be demand from the research community to periodically update the snapshot on Cloudstor so the content available there is aligned to content in the TROVE dataset. This presents a number of challenges for managing versions in the Cloudstor collections.

1. Currently, there is no formal version management procedure in the TROVE dataset that could be mirrored in the Cloudstor dataset.
2. The dataset in Cloudstor is structured differently to the dataset in TROVE with Cloudstor providing access via 26 discrete collections as described earlier.
3. User annotations, and information about them (who, when, what, etc) are captured in TROVE, but the Cloudstor collections contain the most corrected version of the OCR text as it stood at the time the snapshot was taken ie. OCR text plus user corrections. Users of the Cloudstor dataset wishing to roll-back to an earlier version of OCR text, or view information about user annotations, would currently need to do so via the TROVE interface.
4. A further issue associated with 3. is that in some cases, the user annotations may themselves be the object of study. Hence, the project team is currently exploring how the provenance associated with OCR text could be made available via the Cloudstor dataset.

The TROVE team are currently discussing approaches to versioning the Cloudstor collections. They are also keen to assign DOI to the collections. Current thinking is to:

- Create an annual snapshot or version of the data that will be stored on CloudStor using the existing folder or collections structure
- Assign a DOI to each annual snapshot or version
- Create a landing page on Drupal for each snapshot or version that the DOI will resolve to.

27 Molecular Bioscience (#Molecular)

Contributed by Jeff Christiansen

This use case outlines the data management practices for nucleotide sequence data (i.e. genomic (DNA) or transcriptomic (mRNA/cDNA)). It is a community recognised requirement that prior to peer-reviewed manuscript publication, biological nucleotide sequence data generated to underpin any study findings are deposited into the global nucleotide sequence archives. These archives are managed primarily by the NCBI (US) or EBI (Europe), and sequences are submitted through a number of submission portals, and then exchanged between the two resources. At the point of submission, sequence data are assigned an accession number (i.e. A persistent identifier) and a version number (i.e. #1) (e.g. see example K03160.1)

Once housed within NCBI or EBI, curation actions are undertaken on the sequences submitted (for example, removal of artifacts such as short linking sequences from the ends). After such activities, the curated sequences (which are still identified with the original accession ID) are assigned a new version to identify the change, and appropriate notes are added in the associated metadata to describe the change.

Additionally, much work is undertaken to align multiple observed raw sequences deposited into the archives to generate evidence-based (yet artificial) ideal ‘reference sequences’. Reference sequences are assigned a new and unique accession ID, and will contain metadata indicating the raw sequences that have been used to contribute to the generation of the ideal reference sequence.

All nucleotide sequence data distributed by NCBI-GenBank is in flat file format. As shown in the figure below. The area within the red rectangle is head file, the fourth line contains the current GenBank file format release number. The release number consists of three numbers separated by a decimal point. The number to the left of the decimal is the major release number. The digit to the right of the decimal indicates the version of the major release; it is zero for the first version. Note the format has been stable since 1992. [Copied from NCBI-GenBank Flat File Distribution Release Notes]

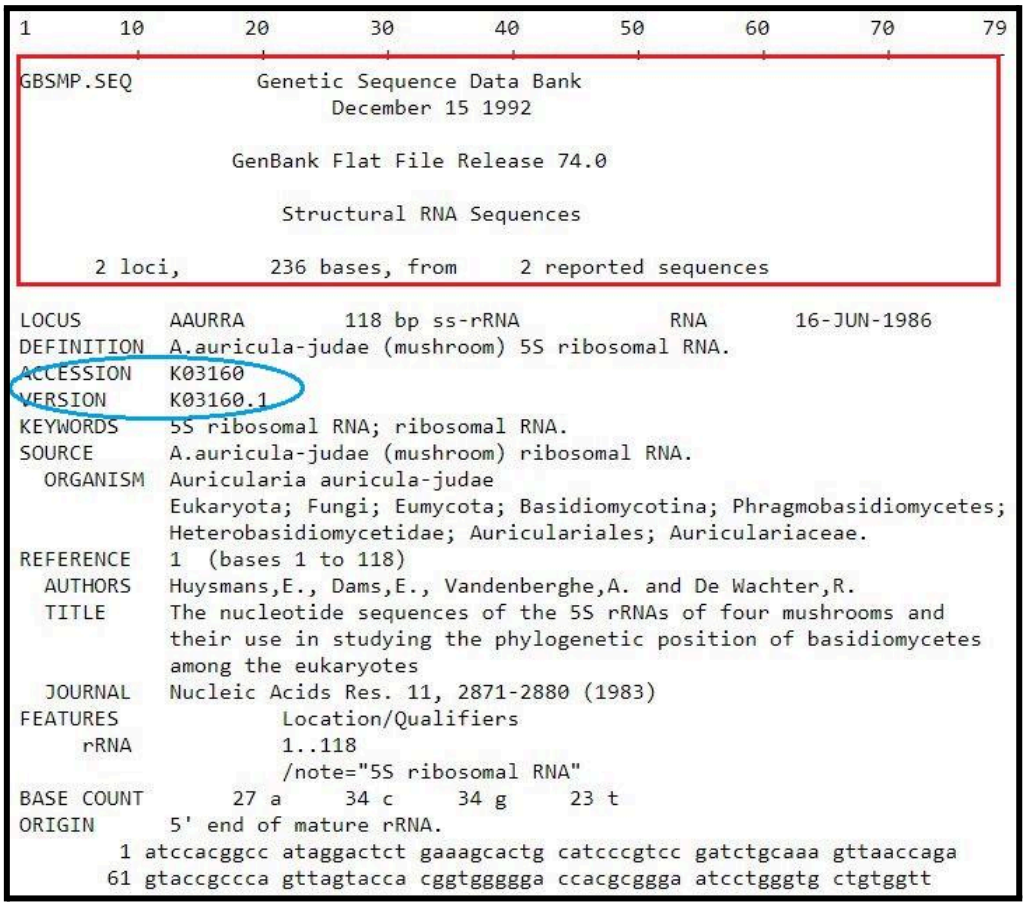
The release refers to the quarterly EMBL (Europe equivalent of NBCI) release in which a flat file appeared, or was expected to appear.

Below the red rectangle is the data entry, including a global persistent identifier (ACCESSION) and Version.

ACCESSION - The primary accession number is a unique, unchanging identifier assigned to each GenBank sequence record. (Please use this identifier when citing information from GenBank.)

VERSION - A compound identifier consisting of the primary accession number and a numeric version number associated with the current version of the sequence data in the record. This is optionally followed by an integer identifier (a "GI") assigned to the sequence by NCBI.

This example data file is displayed at the NCBI web portal as this.



```
1       10        20        30        40        50        60        70        79
GBSMP.SEQ          Genetic Sequence Data Bank
                         December 15 1992

                  GenBank Flat File Release 74.0

                      Structural RNA Sequences

     2 loci,       236 bases, from     2 reported sequences

LOCUS       AAURRA        118 bp ss-rRNA            RNA       16-JUN-1986
DEFINITION  A.auricula-judae (mushroom) 5S ribosomal RNA.
ACCESSION   K03160
VERSION     K03160.1
KEYWORDS    5S ribosomal RNA; ribosomal RNA.
SOURCE      A.auricula-judae (mushroom) ribosomal RNA.
  ORGANISM  Auricularia auricula-judae
            Eukaryota; Fungi; Eumycota; Basidiomycotina; Phragmobasidiomycetes;
            Heterobasidiomycetidae; Auriculariales; Auriculariaceae.
REFERENCE   1  (bases 1 to 118)
  AUTHORS   Huysmans,E., Dams,E., Vandenberghe,A. and De Wachter,R.
  TITLE     The nucleotide sequences of the 5S rRNAs of four mushrooms and
            their use in studying the phylogenetic position of basidiomycetes
            among the eukaryotes
  JOURNAL   Nucleic Acids Res. 11, 2871-2880 (1983)
FEATURES             Location/Qualifiers
     rRNA            1..118
                     /note="5S ribosomal RNA"
BASE COUNT       27 a     34 c     34 g     23 t
ORIGIN      5' end of mature rRNA.
        1 atccacggcc ataggactct gaaagcactg catcccgtcc gatctgcaaa gttaaccaga
       61 gtaccgccca gttagtacca cggtgggggа ccacgcggga atcctgggtg ctgtggtt
```

Figure 23. A sample sequence data file from NCBI-GenBank Distribution Release Notes

Like other PIDs, Accession number consists of a prefix and numerals. The GenBank data submission guide states: GenBank will provide accession numbers for submitted sequences, usually within two working days. This accession number serves as an identifier for your submitted your data, and allows the community to retrieve the sequence upon reading the journal article. The accession number should be included in your manuscript, preferably in a footnote on the first page of the article, or as required by individual journal procedures.

Versions of synthetic data objects (i.e. synthetic sequences) are created through computational methods (eg assembling short sequences into longer sequences), or adding annotations (e.g. where does a gene start or end) as well as curation activities

done by the database (e.g. Genbank at NCBI) either via computational pipelines or manual / semi-manual processes. Anyone who made a change to a previous version can submit a new version to NCBI DB. This guideline explains how to submit a revised or updated a sequence, revision includes editing source information, updating publication information, updating nucleotide sequence, adding features, and updating features. There is a format and encoding for each type of updates.

VERSION is made of the accession number followed by a dot and a version number (and is therefore sometimes referred to as the “accession.versionNo”). Example of a sequence in its 5th version https://www.ncbi.nlm.nih.gov/nuccore/NM_182700.5. NM_182700.5 is the pid for the specific 5th version. An accession number without a version suffix always refers to the latest version of the sequence data.

To see the revision history of a sequence, append report=girevhist to the record's URL. For example, accession U46667's revision history's URL is http://www.ncbi.nlm.nih.gov/nuccore/U46667 ?report=girevhist, or Query an Accession Number will enable access of all version of the ID.

EMBL-EBI offers a Version Checker, which highlights what changes have been made between two selected versions. Here is an example. This Version Checker compares two flat files and uses different coloured lines to represent whether a line has remained unchanged (white), whether it has been deleted (orange), or whether it has been inserted (green).

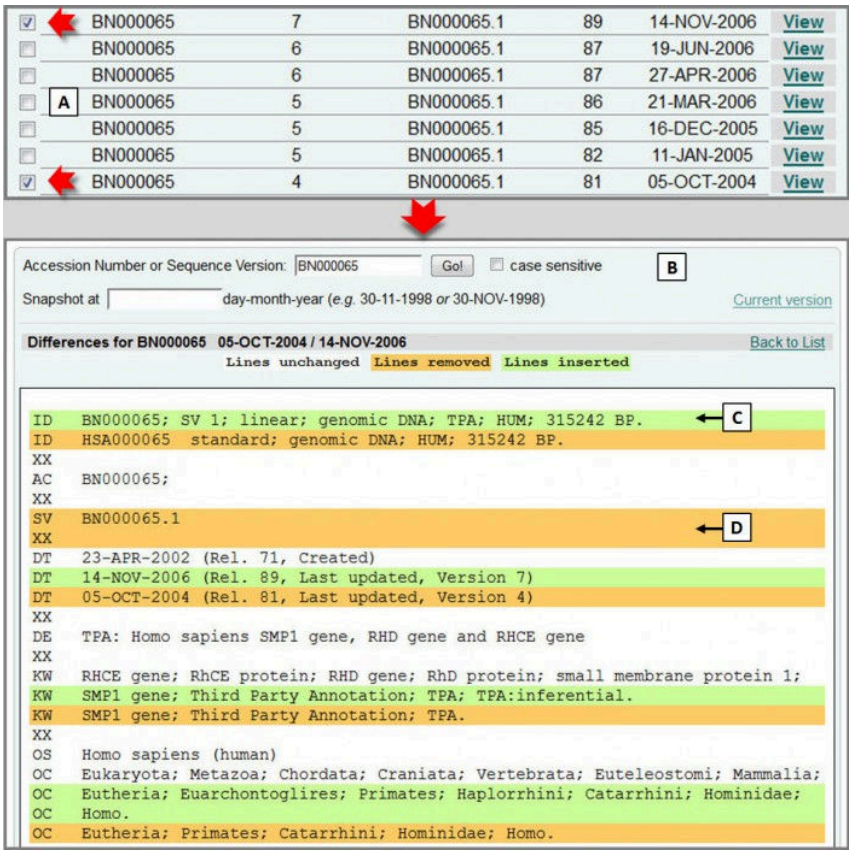


Figure 24. An example of showing comparison of two versions of EMBL-Bank entry BN000065 (14-Nov-2006 and 05-Oct-2004), where the lines inserted or removed are highlighted.

28 Versioning Ontology (#VersOn)

Contributed by Benno Lee

A major tradition in versioning practice is the use of dot-decimal identifiers. The identifiers categorize data objects as a major, minor, or smaller difference from the previous iteration. One major issue in using data identifiers to indicate or measure the amount of change between versions is illustrated in the following diagram.

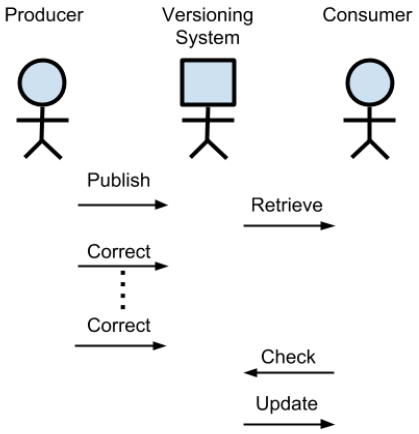


Figure 25. Versioning ontology workflow

Between the Data Producer and Data Consumer, only the Producer supplies information into the versioning system. Leaving authority solely with the Producer means that the impact introduced by changes is not assessed using the Consumer's context.

The Versioning Ontology (VersOn) is a linked-data ontology (currently located at http://orion.tw.rpi.edu/~blee/VersionOntology.owl) which captures individual changes between data objects as linked data. It organizes changes into three classes: Additions, Invalidation, and Modifications. The resulting versioning graph forms a ladder-like structure where the rungs can be counted as a method to assess change to a greater precision than the broad categories of dot-decimal identifiers.

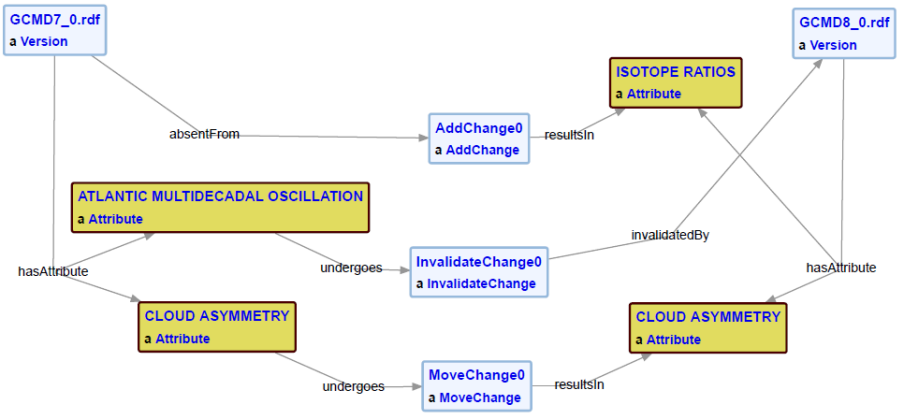


Figure 26. A linked-data ontology

The ontology was used to capture the changes within NASA's Global Change Master Directory (GCMD) Keyword taxonomy.

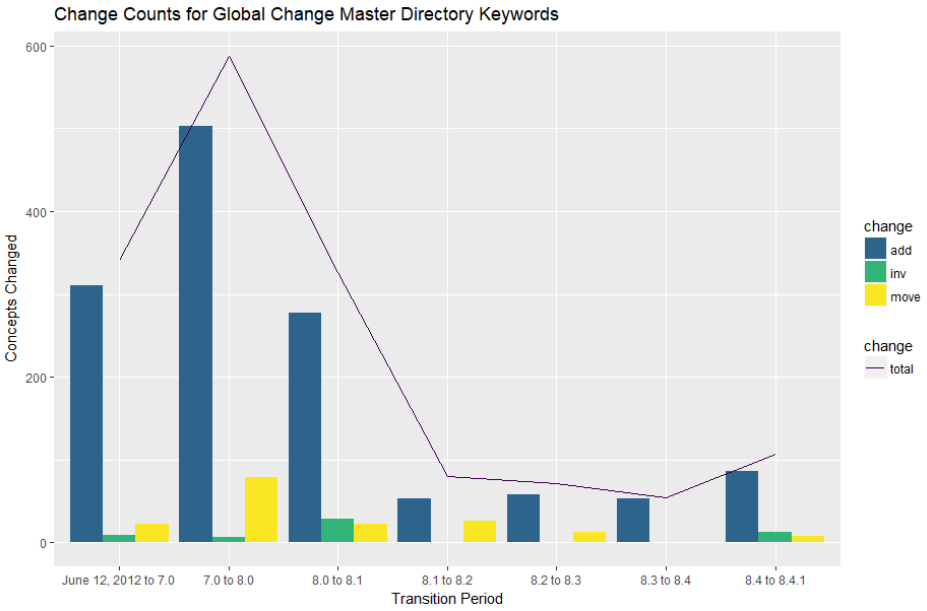


Figure 27. Changes within NASA's Global Change Master Directory (GCMD) Keyword taxonomy

VersOn breaks the total change into component parts so that trends based on operations can be seen. Here, GCMD Keywords experience steady growth with many additions in blue as compared to more stable data sets which may primarily feature modifications. The VersOn method also illustrates an interesting dynamic between Producer and Consumers with the publication of GCMD Keywords Version 8.5.

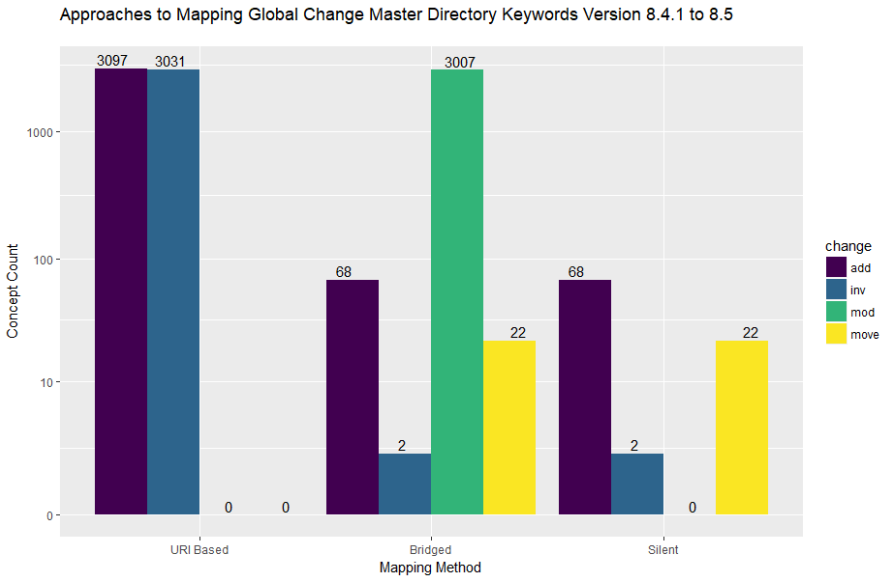


Figure 28. Changes in keywords from 8.4.1 to 8.5

In Version 8.5, the protocol of the keyword namespace was changed from http to https, but the change is non-trivial to web URIs. As a result, using URI best practices results in the assessment that the entire taxonomy has been invalidated and a new set of keywords are added. The GCMD Keywords group directed some consumers to adjust their software to manually map between the new keywords. From the informed consumer's perspective, a Bridged assessment can be made where around 3000 keywords have been modified in green. Finally, the Silent method ignores the changed namespace. We know the GCMD Keywords group used the Silent method because it assigned a minor version dot-decimal identifier to the release, indicating an incremental release. The URI-Based and Bridged methods both have changes on the order of the entire data set, constituting a full release and necessitated a major version identifiers.

VersOn enables detailed change assessment to be conducted regardless of the major or minor identifier assigned to the version. The assessment can be performed after the data set's publication, that is, after the version identifier has been assigned. Additionally, VersOn enables an assessment to be conducted by the consumer in the context of the consumer's application.

29 Closed and Open Manifestations of the same Work (#C-O-M)

Contributed by Paul Jessop (International DOI Foundation)

Cases exist where the authoritative version of a work is behind a paywall while an Open version of the work is accessible, too. Are these two manifestations of the same work?

30 Changes in the File Headers (#C-F-H)

What happens if the data stays the same but the file headers change?

This case could be seen as a new manifestation of the same work. If the new structure impacts the workflows in the designated user community, this new manifestation might need a new identifier.

31 ESIP Data Citation Guidelines for Earth Science Data, Version 2 (#ESIP)

ESIP Data Preservation and Stewardship Committee. (2019). Data Citation Guidelines for Earth Science Data, Version 2. ESIP. Retrieved from https://doi.org/10.6084/m9.figshare.8441816.v1

In all cases, it is very important to carefully track and document versions of the data set. Individual stewards and data centers will need to develop and follow their own practices, but there are some suggestions on how to handle different data set versions relative to an assigned identifier.

For relatively static data sets, a simple approach is to assign a new identifier every time there is any change to the data or metadata. For changing data, the Digital Curation Centre (DCC) Data Citation Guidelines (Ball & Duke, 2015) suggest that DOIs be assigned to different data snapshots taken at regular intervals or as needed. This

would work well for infrequently changed data sets. DCC also suggests a “time slice” approach where “the citable entity becomes the set of updates made to a data set during a particular time period rather than the full data set itself (e.g. the 2008 data from a series running since 1950).” Similarly, the Zenodo repository and DataONE support the ability to cite the “Concept” of a data set with one DOI and the specific version of a data set with another DOI: http://help.zenodo.org/#versioning.

These approaches may be workable in some situations, but they are often unwieldy for the frequently updated time-series common in Earth science. Many repositories with such highly dynamic data only assign a new PID when there are major changes to the data (i.e. a major version). They then rely on documentation and timestamps to identify when minor changes have occurred (minor versions). Individual stewards need to determine which are “major” vs. “minor” versions and describe the nature and range of every change. Typically, something that affects the whole data set, like a reprocessing with an improved algorithm, would be considered a major version. Ongoing additions to an existing time series need not constitute a new version.

This is one reason for capturing the date accessed when citing the data. Small corrections or changes may constitute minor versions and should be explained in the documentation, ideally in file-level metadata. This general approach, while workable, relies heavily on human interpretation. The RDA Recommendation provides better specificity and verifiability.

32 Zenodo DOI versioning (#Zenodo)

What is DOI versioning?

DOI versioning allows you to:

- edit/update the record’s files after they have been published.
- cite a specific version of a record.
- cite all of versions of a record.

How does DOI versioning work?

When you publish an upload on Zenodo for the first time, we register two DOIs:

- a DOI representing the specific version of your record.
- a DOI representing all of the versions of your record.

Afterwards, we register a DOI for every new version of your upload.

This is best illustrated by an example of a software package. If the software has been released in two versions (v1.0 and v1.1) on Zenodo, then the following DOIs would have been registered:

- v1.0 (specific version): 10.5281/zenodo.60943
- v1.1 (specific version): 10.5281/zenodo.800648
- Concept (all versions): 10.5281/zenodo.705645

The first two DOIs for versions v1.0 and v.1.1 represent the specific versions of the software. The last DOI represents all the versions of the given software package, i.e. the concept of the software package and the ensemble of versions. We therefore also call the them Version DOIs and Concept DOIs (note, technically both are just normal DOIs).

You may notice that the version DOIs do not include a ".v1"-suffix. Read below to find out why.

Which DOI should I use in citations?

You should normally always use the DOI for the specific version of your record in citations. This is to ensure that other researchers can access the exact research artefact you used for reproducibility. By default, Zenodo uses the specific version to generate citations.

You can use the Concept DOI representing all versions in citations when it is desirable to cite an evolving research artifact, without being specific about the version.

Where does the Concept DOI resolve to?

Currently the Concept DOI resolves to the landing page of the latest version of your record. This is not fully correct, and in the future we will change this to create a landing page specifically representing the concept behind the record and all of its versions.

Do you support versioning for already existing records?

Yes. However, for uploads published before the 30th of May 2017, you have to first upgrade your record to support versioning. This is done by clicking the "Upgrade to versioned record" button on the record page.

IMPORTANT If you have previously uploaded multiple versions of an upload as individual records on Zenodo, then DO NOT click the button to upgrade your record with versioning support. Please contact us so we can link the records under one versioning scheme.

Clicking the "Upgrade to versioned record" button on any of the records you would like to link, will irreversibly register them as individually-versioned records.

If you used the GitHub integration to archive your software on Zenodo, then we have already migrated and linked your records to support versioning.

I only want to change the title of my upload, do I still get a new DOI?

No, as before you can continue to edit the metadata of your upload without creating a new version of a record. You should only create a new version if you want to update the files of your record.

Why don't the DOIs have a version number suffix like ".v1"?

Including semantic information such as the version number in a DOI is bad practice, because this information may change over time, while DOIs must remain persistent and should not change.

Moreover, Zenodo DOI versioning is linear, which means that the Zenodo version number may in fact not be the real version number of the resource. Take for instance software, where it is common practice to have dot versions and make new releases in a non-linear order (e.g. first v1.0, then v1.1, then v2.0, then v1.2).

The versioning suffix is also not a functionality of the DOI system, i.e. adding .v2 to DOI will not resolve to version 2 of a resource for any DOI from any provider. Different providers also use different patterns such as e.g. .v2, .2, /2.

Most importantly, version suffixes are not machine readable. A discovery system that understands DOIs, will not know that .v1 and .v2 of a DOI are in fact two versions of the same resource.

A better solution to this problem is to semantically link two DOIs in the metadata of a DOI. This ensures that discovery systems have a machine-readable way to discover that two DOIs are versions of the same resource.

See also Cool DOIs blog post by Martin Fenner, DataCite Technical Director.

Why do you include "zenodo" in the DOI?

Currently DOIs registered by Zenodo follows the pattern "10.5281/zenodo." where 10.5281 is the Zenodo DOI prefix and is a sequentially assigned integer. The word "zenodo" is semantic information, and as mentioned in the previous question it is a bad idea to include semantic information in DOIs as it may change over time. The current practice was introduced when Zenodo was launched in May 2013, and while it is not ideal we did not want to change the existing practice.

Do you duplicate all the files for every new version of a record?

No, if you change a 10kb README file in 50GB dataset we do not duplicate the entire 50GB dataset. Invenio v3, the underlying digital repository platform that powers Zenodo, efficiently handles the file storage so we only store the new extra 10kb.

33 Adopters of the RDA Recommendations on Dynamic Data Citation (#RDA-DDC-A)

Contributed by Andreas Rauber (Co-chair of the RDA Dynamic Data Citation WG)

The following data centers have abandoned semantic versioning as a versioning strategy and are adopting a time-stamping based approach to document changes to a data with historization allowing to go back to any early state of the data at any arbitrary point in time. Detailed documentation of these adoptions including webinar recordings, slide sets as well as additional materials are available at the Webinar Series page of the RDA Working Group at:

- **Implementing of the RDA Data Citation Recommendations by the Climate Change Centre Austria (CCCA) for a repository of netCDF files**
    - Presenter: **Chris Schubert**, Head of the CCCA Data Center, Vienna, Austria

    - **Recording is available at**: https://www.rd-alliance.org/implementing%C2%A0-rda-data-citation-recommendations-climate-change-centre-austria-ccca-repository-netcdf
    - **Slides are available at**: https://www.rd-alliance.org/rda-wgdc-webinar-slides-chris-schubert-climate-change-centre-austria-ccca
    - **Supporting Paper is available at:**
      Chris Schubert, Harald Bamberger: Handling Continuous Streams for Meteorological Mapping. Service-Oriented Mapping, Lecture Notes in Geoinformation and Cartography book series (LNGC), pp 251-268, Springer, 2018. https://doi.org/10.1007/978-3-319-72434-8_13

- **Implementing the RDA Data Citation Recommendations for Long-Tail Research Data / CSV files**
    - Presenter: **Stefan Pröll**
    - **Recording is available at:** https://www.rd-alliance.org/implementing-rda-data-citation-recommendations-long-tail-research-data-csv-files
    - **Slides are available at**: https://www.rd-alliance.org/system/files/documents/20170518-RDA-StefanProell.pdf
    - **Supporting papers are available at**:
        - Stefan Pröll, Kristof Meixner and Andreas Rauber. Precise Data Identification Services for Long Tail Research Data. 13th International Conference on Digital Preservation (iPRES). 2016. https://www.rd-alliance.org/ipres2016-paper-implementing-wgdc-recommendations-long-tail-research-data-csv-files
        - *Stefan Pröll. Enabling Reproducibility for Small and Large Scale Research Data Sets. D-Lib Magazine, January/February 2017, Volume 23, Number 1/2*
        - *http://www.dlib.org/dlib/january17/proell/01proell.html*
        - *Stefan Proell and Andreas Rauber. A Scalable Framework for Dynamic Data Citation of Arbitrary Structured Data," in 3rd International Conference on Data Management Technologies and Applications (DATA2014), 2014*

- **Implementing the RDA Data Citation Recommendations in the Distributed Infrastructure of the Virtual and Atomic Molecular Data Center (VAMDC)**
    - Presenter: **Carlo Maria Zwölf**, VAMDC, Observatoire de Paris, France
    - **Recording is available at:** https://www.rd-alliance.org/dynamic-data-citation-within-distributed-infrastructure-virtual-and-atomic-molecular-data-center
    - **Slides are available at**: https://www.rd-alliance.org/webinar-slides-carlo-maria-zw%C3%B6lf-implementing-rda-data-citation-recommendations-distributed
    - Supporting paper is available at: *C.M. Zwölf, N.Moreau, M-.L. Dubernet, New Model for dataset citation and extraction reproducibility*

in VAMDC, Journal of Molecular Spectroscopy, doi:10.1016/j.jms.2016.04.009, (arXiv version at http://arxiv.org/abs/1606.00405)
    - https://www.rd-alliance.org/journal-molecular-spectroscopy-paper-implementing-wgdc-recommendations-vamdc-infrastructure

- **Implementation of Dynamic Data Citation at the Vermont Monitoring Cooperative**
    - Presenter: **James Duncan**, VMC, University of Vermont, Burlington, VT
    - **Recording is available at**: https://www.rd-alliance.org/implementation-dynamic-data-citation-vermont-monitoring-cooperative
    - **Slides are available at**: https://www.rd-alliance.org/webinar-slides-james-duncan-implementation-dynamic-data-citation-vermont-monitoring-cooperative

- **Adoption of the RDA Data Citation of Evolving Data Recommendation to Electronic Health Records**
    - Presenter: **Leslie McIntosh**, PHD, MPH, Director Center for Biomedical Informatics, Washington University in St.Louis
    - **Recording is available at**: https://www.rd-alliance.org/adoption-rda-data-citation-evolving-data-recommendation-electronic-health-records
    - **Slides are available at**: https://www.rd-alliance.org/webinar-slides-leslie-mcintosh-adoption-rda-data-citation-evolving-data-recommendation-electronic
    - **Supporting paper is available at**: https://www.rd-alliance.org/amia-joint-summits-2017-paper-implementation-wgdc-recommendation-biomedical-data-wustl (AMIA Joint Summits 2017)

34 ASTER (#ASTER)
Contributed by Lesley Wyborn, NCI.

Background

ASTER (Advanced Spaceborne Thermal Emission and Reflectance Radiometer, http://asterweb.jpl.nasa.gov) is a Japanese Space Systems (JSS) imaging instrument that was launched in December 1999 onboard the United States Terra satellite (http://terra.nasa.gov NASA's Earth Observing System, 2011). ASTER is a multispectral satellite system that has 14 spectral bands (Abrams et al., 2002, https://lpdaac.usgs.gov/documents/262/ASTER_User_Handbook_v2.pdf) including:

i. The Visible and Near-Infrared (VNIR - 500-1000 nm – 3 bands @ 15 m pixel resolution);
ii. Short Wave-Infrared (SWIR – 1000-2500 nm range – 6 bands @ 30 m pixel resolution); and

iii. Thermal Infrared (TIR 8000-12000 nm - 90 m pixel resolution) atmospheric windows in a polar-orbiting, 60 km swath.

These 14 spectral bands span wavelengths sensitive to the identification of important rock forming minerals such as Iron oxides; clays; carbonates; quartz; muscovite and chlorite, and are ideal for developing specific mineral distribution maps of the surface of the Earth.

From the raw ASTER instrument data a series of derivative versions have been produced and can, to some extent, be aligned with the defined NASA processing levels which range from Level 0 (L0) to Level 4 (L4), with L0 products being the raw data at full instrument resolution, whilst at higher levels, the data are converted into more useful parameters and formats and released as additional versions. (https://earthdata.nasa.gov/collaborate/open-data-services-and-software/data-information-policy/data-levels)

The reduction of the ASTER raw L0 instrument data to the derivative L1B ((radiance@sensor) and L2 (reflectance and emissivity) products by JSS's Ground Data Segment (GDS - www.gds.aster.ersdac.or.jp) involves the correction for instrument, illumination, atmospheric and geometric effects as described in the ASTER Science Team's Algorithm Theoretical Basis Documents (www.science.aster.ersdac.or.jp/en/documnts/atbd.html).

In late 2009, a Australian initiative led by CSIRO and supported by State, Territory and Federal government geoscience agencies across Australia, as well as the ASTER Science Team (http://www.science.aster.ersdac.or.jp/en/science_info/index.html), JSS, NASA-JPL, United States Geological Survey (USGS), AuScope and the National Computational Infrastructure (NCI) aimed to produce National, public, web-accessible, ASTER National mosaic maps of the Earth's surface mineralogy of Australia which were to generated from the JSS ASTER L1B and L2 products. A goal of the project was to ensure that any final data products could be used from continental scale down to 1:50,000 prospect scale (Cudahy, 2012: https://publications.csiro.au/rpr/pub?pid=csiro:EP125895).

The original Australian mosaic was sourced from ~35,000 JSS ASTER L1B and L2 scenes, with approximately 3500 scenes selected and then rectified and re-projected to geodetic coordinates to produce an Australian Mosaic (Level 3). This mosaic was then used to generate 17 mineral map L4 data products in Band Sequential (BSQ) image format and involved applying a series of product masks/thresholds to generate a suite of geoscience mineral maps that included fourteen ASTER VNIR/SWIR Geoscience products and three ASTER TIR products.

Next, from these L4 BSQ files, contrast stretching and colour rendering was applied to generate products in GeoTIFF format for use in GIS packages and online mapping systems. As well, the BSQ files were also converted into self describing netCDF files to optimise use in HPC and other scientific analyses. To reduce the file sizes downloads, the national coverages were also broken up into 1:1,000,000 map tiles.

In 2012 multiple organisations then proceeded to release the various versions of the 17 mineral maps from their own data repositories and/or portals and online GIS mapping systems at varying scales from National to State to individual1:1,000,000 map tiles.

The FRBR hierarchy of versions created from the original work.

The simple NASA Level 0 to Level 4 classification of the processing levels did not reflect the complexity of the multiple versions of the ASTER products released by the various organisations involved in the project, nor did it take into account any subsequent revisions over time by each individual organisation. Although the same data product was originally simultaneously released by multiple organisations in 2012, at each site there have been subsequent modifications to both the data and/or to the methods used to make the data accessible. Since the initial release in 2012, there has been no systematic inter-organisational process to ensure that where the same product is released from multiple sites that product is still exactly the same. Indeed, in a quick comparison of the same GeoTIFF product from several sites, the bitstreams were found to be different for supposedly equivalent files.

Ensuring scientific **reproducibility** (knowing the source of each version that was published and/or used in subsequent reanalysis), **provenance** (knowing the sequential history of any evolved data product) and **attribution** (knowing which organisation/individual had produced and/or funded and/or was sustaining) each 'version' released led to the use of the Functional Requirements for Bibliographic Records (FRBR) to help develop an understanding of the Full-path of data use from the original collection of the source data by the ASTER mission to the release of multiple data products at various levels of processing and refinement on numerous disconnected websites. FRBR was developed in 1992-1995 in the library community as a conceptual, generalized view of the bibliographic universe, intended to be independent of any cataloging code or implementation (https://www.loc.gov/cds/downloads/FRBR.PDF). The FRBR approach argues that any products of any individual intellectual endeavor can be expressed around four entities: work (the original distinct intellectual creation), expression (the specific intellectual form a specific work can take), manifestation (the physical embodiment of each individual expression of any work) and item (a single exemplar of any single manifestation, i.e., an individual concrete entity that is made available for distribution).

Based on FRBR, the equivalent and derivative versions of the ASTER use case can be divided into these four entities (work ⇒expression ⇒manifestation ⇒ item) as is illustrated in the following figure. Note that the expression and manifestation entities can have multiple derivative 'versions' and in addition there can also be multiple subsequent revisions released over time as new versions of the last three (expression, manifestation, item). Recording provenance is critical to understanding both where the processing workflow each released version is originally from, as well as any subsequent revisions that are produced of each version.

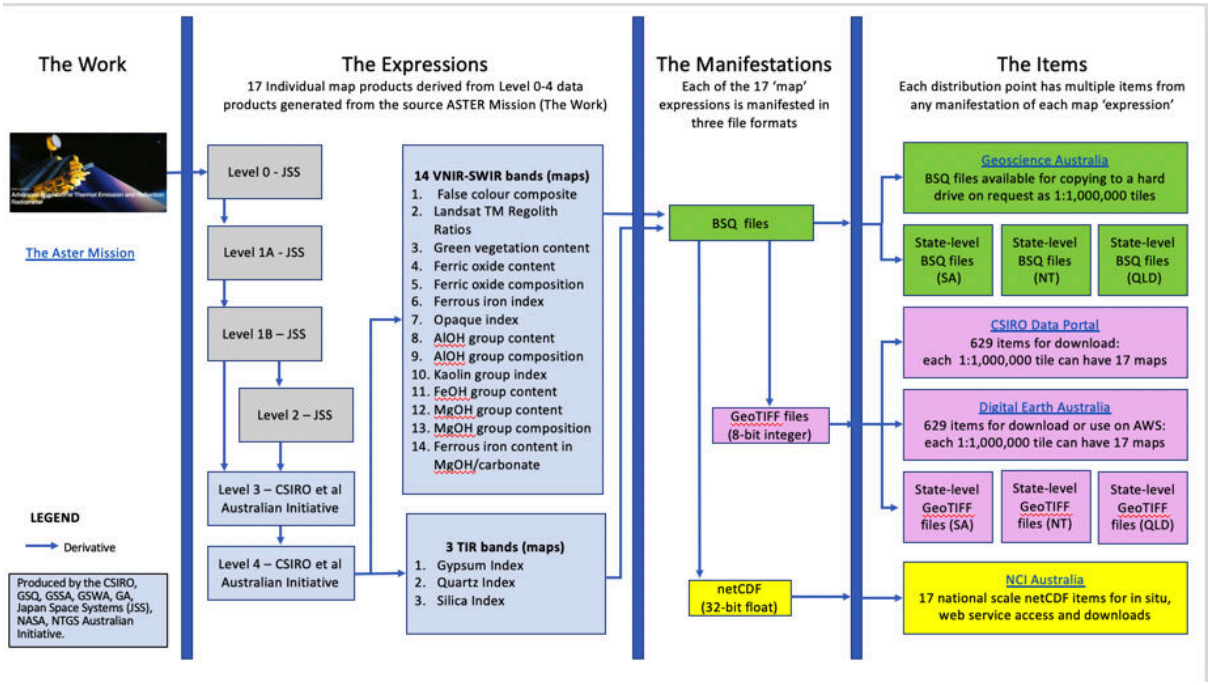


Figure 29. Schematic overview of the FRBR model applied to Full-path of data use from the ASTER Mission.

The ASTER Full-path of Data Processing and Release as FRBR entities.

i. The Work

In FRBR terminology, the work is defined as the original ASTER mission.

ii. The Expressions

The reduction of the ASTER Level 0 instrument data to Level 1B (radiance@sensor) or Level 2 (reflectance and emissivity) products by JSS produced 4 initial versions of the ASTER 'work' (the grey boxes in the above figure). The Australian initiative then generated the national mosaic (L3) from the JSS L1B and L2 scenes and from this L3 mosaic, the set of L4 17 mineral map data products were derived (the blue boxes in the above figure).

*In FRBR terminology, the JSS Level 1B and Level 2 products as well as the Australian L3 mosaic and each of the 17 L4 maps are considered an '**expression'** of the '**work**'.*

iii. The Three Manifestations

Each of these 17 L4 mineral maps was then made available in three different formats that relate to different user requirements and/or delivery infrastructure as follows:

i. Band sequential image (BSQ) files that can be restretched/ processed),
ii. GeoTIFF files that were generated by contrast stretching and colour rendering to national standards to generate more user friendly GIS-compatible products. Note that these are optimized for usage in GIS packages or as online image files and are 8-bit integer files.
iii. Self-describing netCDF files that were produced at NCI for analysis at full resolution at continental scale: using standardised data services, these could also be subsetted down to very small bounding boxes for local analysis at the

prospect or locational file. Note that the netCDF files are optimised for HPC and scientific analysis and are 32-bit float.

*In FRBR terminology, each of these three formats is considered to be a **'manifestation'** of each of the seventeen L4 '**expressions'** of the '**work'.***

iv. The Individual Items

The file sizes of some of the national coverages was very large, e.g., the BSQ are ~60 GB each. In 2012 the BSQ and even the GeoTIFF manifestations were too large to deliver as online file downloads and 37 individual 1:1,000,000 tiles (up to 350 MB each) were generated for each of the 17 expressions in BSQ or GeoTIFF (629 files in total). All three manifestations are made accessible as items from various organisational websites, and in some cases, multiple organisations are releasing an equivalent version of the same data product.

Some known items available include:

a. CSIRO Data Access Portal (DAP, https://data.csiro.au/dap/search?q=ASTER):
   i. Each of the 17 maps is available as thirty seven, 1:1,000,000 map sheet tiles in GeoTIFF format (629 files in total).
   ii. A manual that describes the Australian ASTER National Initiative and how the 17 mineral maps were produced from the original JSS L1B/L2 data products.

b. Geoscience Australia (GA) (https://ecat.ga.gov.au/geonetwork/srv/eng/catalog.search#/metadata/74347)
   i. National-scale images available as WMS from the Exploring For The Future (EFTF) Portal (https://portal.ga.gov.au/)
   ii. Digital Earth Australia in GeoTIFF format as thirty seven individual 1:1,000,000 map sheet tiles ((https://data.dea.ga.gov.au/?prefix=ASTER_Geoscience_Map_of_Australia/ and made accessible on Amazon Web Services (AWS)
   iii. IBSQ files available on an external hard drive from the GA Sales Centre (http://d28rz98at9flks.cloudfront.net/74427/National_ASTER_Geoscience_Maps_Flyer.pdf)

c. National Computational Infrastructure (NCI)
   i. Available as national coverages as follows (http://dap.nci.org.au/thredds/remoteCatalogService?catalog=http://dapds00.nci.org.au/thredds/catalogs/wx7/catalog.xml)

      - THREDDS Server
      - OPeNDAP
      - WMS
      - WCS
      - NetCDF Subset service
      - File download
      - GSKY service (both WMS and WCS).

ii. Each of the national netCDF files can be further subsetted using OPeNDAP protocols to any bounding box defined by the user. That is, the user can create their own unique subset (version) of the data.

d. Geoscience Maps at the State/Territory level

i. Versions of the ASTER datasets have also been released as statewide BSQ and GeoTIFF files from the NT, SA, QLD and WA State Geological Survey websites.
ii. The QLD data are also accessible from the CSIRO DAP.

*In FRBR terminology, each of these multiple versions are considered to be an **'item'** of each **'manifestation'** of each '**expression'** of the '**work'.***

35 Magnetotelluric Geophysics Workflow (#MT)

Contributed by Nigel Rees, NCI

For the Magnetotelluiring (MT) Geophysics workflow, the various data versions released were aligned with the defined NASA processing Levels to facilitate both reproducibility, provenance and also enable attribution to the organisations/individuals credited with producing and/or making available each version.

Table 1. The different Magnetotelluric data processing levels (Rees et al., 2019) based on NASA's Earth Observing System Data and Information System (EOSDIS) data products[3]

Processing levels	Name	Description	Collected / Processed by	Typical Volumes
Packed Raw Data	Raw Time Series	Telemetry data streamed from site loggers.	Single researcher or Research Team	GBs to TBs
Level 0	Edited Time Series	Time ordered instrument recorded data (e.g., raw voltages, counts) at full resolution.	Single researcher or Research Team	GBs to TBs
Level 1A	Calibrated Time Series	Level 0 data that have been calibrated in a reversible manner and packaged with associated calibration equations.	Single researcher or Research Team	GBs to TBs
Level 1B	Resampled Time Series	Level 0 or 1A data that have been irreversibly transformed (e.g., resampled, noisy data removed, filters applied).	Can be processed by anyone with access to L1A	GBs to TBs
Level 2	Derived frequency domain processed data (e.g., EDI)	Geophysical parameters (e.g., impedance tensors) derived from frequency domain time series processing of Level 1A or 1B data.	Can be processed by anyone with access to L1A or L1B	MBs
Level 3A	Derived modelling inputs	Level 2 parameters converted into input files for modelling and inversion algorithms.	Can be processed by anyone with access to L2	MBs
Level 3B	Derived modelling outputs	Level 2 parameters mapped onto space-time grids.	Can be processed by anyone with access to L2 or L3A	MBs

3 https://earthdata.nasa.gov/collaborate/open-data-services-and-software/data-information-policy/data-levels

The processing of raw MT time series data (Level 0, Level 1A, 1B) to the MT transfer functions (EDI files / EMTF XML) (Level 2) to the inverted conductivity model outputs (Level 3A, 3B) is quite complex (Table 1). There are many different processing and inversion codes (and methods) available, which all have different user specific choices of parameters that can be made along the full path of data processing and modeling. For example, Figure 1 shows the different processing steps involved to get from the raw time series data to the MT transfer functions, and each step can be done differently depending on various factors (e.g., the code being used, the objective of the experiment being conducted, the instruments being used, processing parameters, the geophysicist running the processing, etc.). Likewise, inversions of the transfer functions to produce models is also highly subjective. As a result, it is likely that each version of the MT transfer functions and/or MT conductivity models produced by individual geophysicists from a single raw dataset will be different.

To enable comparisons between the processing of individual geophysicists requires knowledge of the individual versions produced at each level of processing: transparency of processing will require a provenance chain that links each subsequent version. Not all Level 3 products need to be developed from the same Level 0/Level 1 products: these can be produced from any published Level 2 product. Likewise, each level could be revised and as a flow-on effect there would then be different versions of each subsequent level.

This high degree of variability of processing makes it essential that there are clear statements of which earlier versions the higher level products are derived from. This is important not just in achieving transparency, but also in being able to give attribution to those that were involved in the collection and curation of the original data in the field (including researchers, institutions, funders, etc). Whether each of the individual intermediate levels can also be stored and made accessible is debatable: storage costs may prohibit this. Hence, clear statements of how each intermediate version is created is essential.

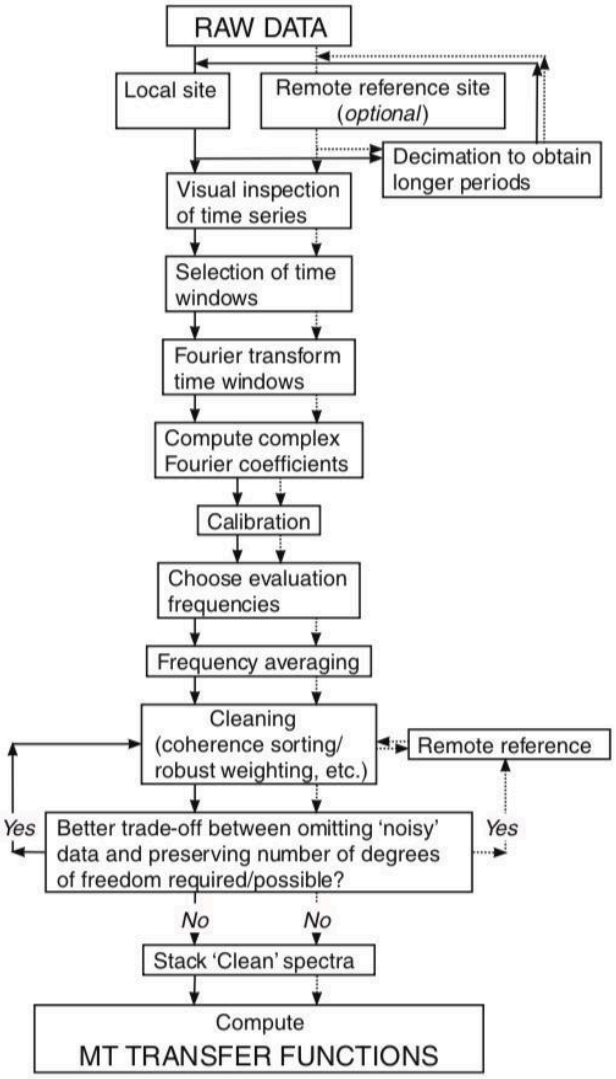


Figure 30. Flowchart of the steps involved in processing MT data from the raw time series to the MT transfer functions (taken from Simpson and Bahr (2005)). By having MT time series available online with compute, a MT scientist now has the ability to reprocess transfer functions to their desired standard and specific use-case without having to rely on what another data provider had produced.

Supporting paper:

Rees, N., Evans, B., Heinson, G., Conway, D., Yang, R., Theil, S., Robertson, K., Druken, K., Goleby, B., Wang, J., and Wyborn, L., 2019. The Geosciences DeVL Experiment: new information generated from old magnetotelluric data of The University of Adelaide on the NCI High Performance Computing Platform. AEGC 2019 Data to Discovery, September, 2019, Perth, https://2019.aegc.com.au/programdirectory/docs/138.pdf

36. TERN (Terrestrial Ecosystems Research Network) Eco-informatics Facility (#TERN)

Contributed by Siddeswara Guru

The TERN Data Services Platform works with governments, researchers, educators and students to make ecological “plot” data (including quadrats, transects, pitfall

traplines, cage trap arrays, and other systematic collection methods) discoverable and freely accessible.

The Facility is underpinned by a data submission service known as SHaRED and the AEKOS portal that enables discovery of, and access to, ecological datasets. AEKOS provides access to primary data, provided to the Facility by government agencies and research organisations, as well as derived data, submitted by researchers via the SHaRED service.

The Facility offers a DOI service for data published by researchers via SHaRED.
In cases where data and corresponding metadata is updated, the repository supports version control via the DOI. New versions are linked to older versions via metadata using the old DOI. In the case of annual and periodic data, new data is appended to previous published version (e.g., 2000-2015, add 2016 to give 2000-2016) and published as a full dataset with a new DOI to maintain the integrity of the data collected. This means all versions of a particular dataset are accessible.

A search in AEKOS on “Ausplots Forest Monitoring Network - Forest Fuel Survey” returns records for both v1 and v2 of the dataset.

TERN consider data as a new version if the changes are more than 10%. For a file-based datasets each of the dataset version will have a different filenames. For large datasets (e.g. remote sensing data), we don’t keep each and every version of data due to space constraints but keep and make the latest version available.

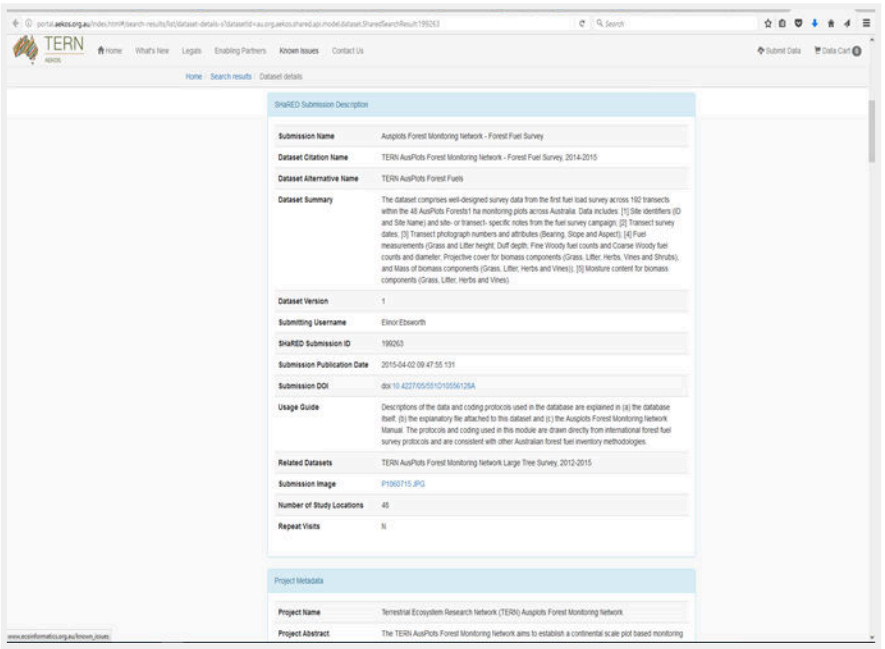


Figure 31. The landing page for version 1 of the dataset does not reference v.2

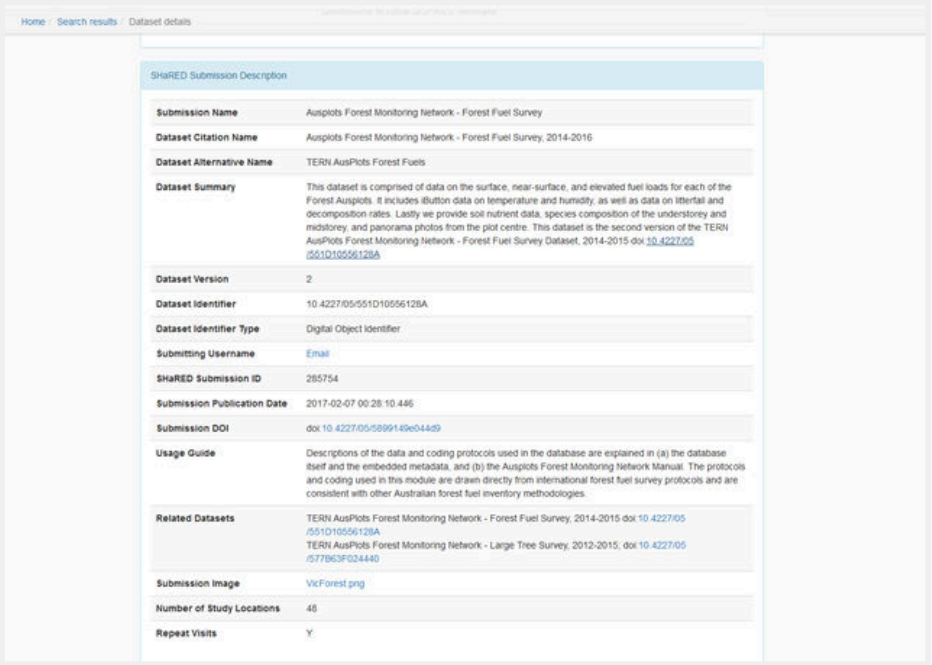


Figure 32. The landing page for version 2 references version 1 and and provides a link to it via the DOI

37 Australian Data Archive (#ADA)

Contributed by Heather Leasor

The Australian Data Archive provides a national service for the collection and preservation of social science digital research data. ADA holds over 6000 datasets from more than 1500 projects and studies which range from 1838 through until the present day. ADA data cover longitudinal studies, social attitudes surveys, health data, elections & political studies and public opinion polls. ADA utilizes Dataverse platform for managing data deposit and access. This platform has inherent version controls in built which are detailed below.

Version Numbering

ADA current versioning approach has been in place since around 2009, versioning numbering system at decimal level was implemented in 2010. Versioning is done at the collection/project level, about 10% of ADA data have been versioned. Versions are not restricted to longitudinal data but are more common in this data type. The general rule applied at the ADA is (V major.minor):

- Major version change= full new wave of collection, information alteration that will alter the previous outcome of the data = numeric before the decimal place number change (V1.# to V2.#)
- Minor version change = minor changes between full waves, changes to spelling or minor alterations or additions to metadata = numeric after the decimal place number change (V#.1 to V#.2)

Longitudinal studies have variety of versioning dependant on agreements with depositing groups. Each agency and research community has different protocols and

specification for versioning of data and the ADA attempts to work within their parameters, with examples provided herein after the general ADA Dataverse versioning explanation below. All versioning is detailed in the citation block.

Dataverse Automatic Version Numbering

The ADA utilizes dataverse as a distribution platform for data it distributes. This platform also incorporates functions for version control which cannot be altered as it occurs programmatically. These are detailed in the dataverse user guides (http://guides.dataverse.org/en/4.6.1/user/dataset-management.html#dataset-versions). Datasets cannot be deleted in dataverse but it can be deaccessioned. All versions of the data are saved in the ADA long term storage archive, these are in the forms of SIP, AIP and all versions of superceded data and current DIP with processing syntax where applicable. Earlier versions are available by request if necessary but not always searchable on dataverse.

Dataverse has inbuilt versioning capabilities. The graphic below (Graph 1) details the versioning automated by dataverse. This is documented in the versioning tab of the dataset and in the data citation (Table 1 and Table 2 below). If you add a data file the study will automatically be updated to a major version (v1.2 moved to v2.0) after the publish button is actioned. Minor metadata alterations will result in minor version change (v1.1 move to v1.2) after the publish button is actioned.

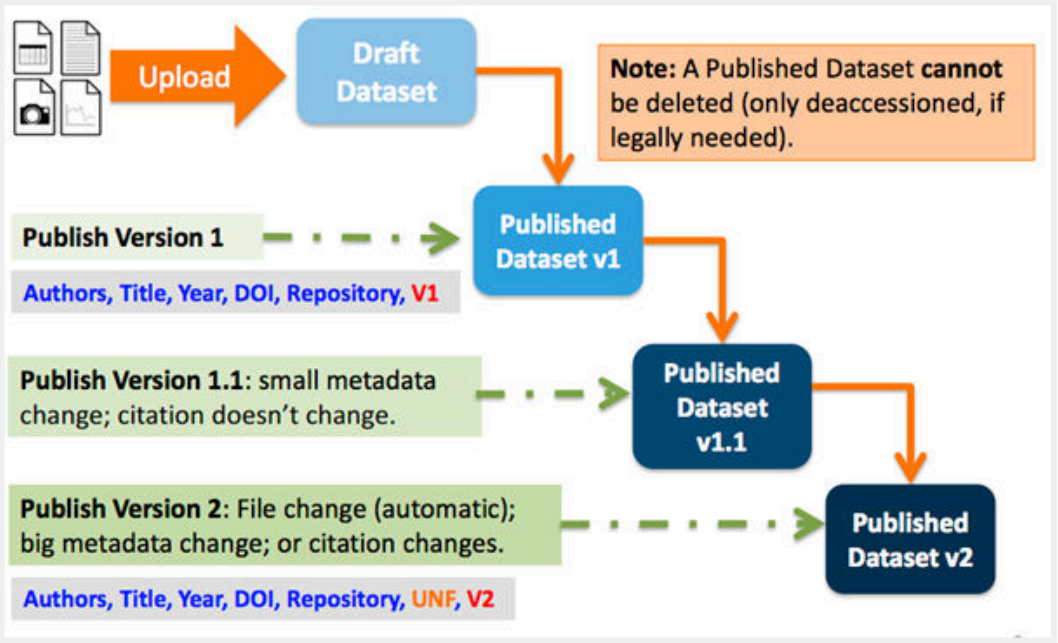


Figure 33. Graphic from Dataset Versions in the Dataverse User Guide 4.6.1 accessed September 24 2019 at link provided in text.

Table 2: An example of details in the version tab of dataverse are below:

	Dataset	Summary	Contributors	Published
	5.0	Files (Added: 1; Removed: 1); View Details	Sebastian Kocar	September 18, 2019
	4.1	Files (Changed File Metadata: 4); View Details	Steven McEachern	July 24, 2019
	4.0	Citation Metadata: Title (Changed); Additional Citation Metadata: (4 Added, 6 Changed); Files (Added: 5; Removed: 5; Changed File Metadata: 12); View Details	Steven McEachern	July 24, 2019
	3.1	Files (Changed File Metadata: 17); View Details	Steven McEachern	June 7, 2019
	3.0	Citation Metadata: Title (Changed); Additional Citation Metadata: (1 Changed); Files (Added: 17; Removed: 15); View Details	Steven McEachern, Sebastian Kocar	March 1, 2019
	2.0	Citation Metadata: Title (Changed); Description (1 Changed); Contact (1 Changed); Author (1 Added); Keyword (18 Added, 4 Changed); Social Science and Humanities Metadata: (7 Added, 1 Removed, 4 Changed); Additional Citation Metadata: (17 Added, 1 Removed, 11 Changed); Geospatial Metadata:(1 Added); Files (Added: 15; Removed: 9); View Details	Steven McEachern, Sebastian Kocar	January 30, 2019
	1.0	This is the first published version.	Steven McEachern, Marina McGale	September 24, 2018

Department of Social Services; Australian Institute of Family Studies; Australian Bureau of Statistics, 2018, "Growing Up in Australia: Longitudinal Study of Australian Children (LSAC) Release 7.2 (Waves 1-7)", doi:10.26193/F2YRL5, ADA Dataverse, V5

Longitudinal Studies Versioning

Longitudinal studies have data which is usually changed annually. Most Longitudinal data have a new version every year. Actual versioning is not done at ADA - ADA

receives updated version from agencies, e.g., Australia Bureau of Statistics, AIFS, NCLD and others. Longitudinal studies have a variety of versioning dependant on agreements with depositing groups. Each agency and research community has different protocols and specification for versioning of data and the ADA attempts to work within their parameters. All versioning is documented in the title or citation block as well as in the version tab of Dataverse and potentially in the notes section of metadata pertaining to the study. All older versions are stored with the ADA but made available only by request (i.e. not be available and searchable online), some remain as a separate study with specified DOI release wave and versioning of the latest. This is negotiated with depositors.

- Release= A data release is a compilation of data, typically from several surveys or other data sources. In longitudinal studies, a data release refers to a compilation of all waves of data collected until a particular point in time, and can include data from other surveys or administrative sources as well.
- Wave= Within the context of survey research, a wave refers to each separate survey in a series of related surveys. If a survey is conducted only once, i.e. is a cross-sectional survey, then the concept of a "wave" does not apply.
- Major version=Alterations to data that alter the results, analysis or outcomes would result in a major version altering the first number to the left of the decimal in the version (V1.0 to V2.0)
- Minor version=Alterations to data spelling not altering results of data, alterations in metadata or addition of documentation result in minor version altering the second number after the decimal in the version (V1.1 to V1.2)

Most longitudinal studies are disseminated as releases and are consequently versioned as releases (even though there might be changes to only one wave in a new data version). Versioning of longitudinal studies is separate from Dataverse versioning for two main reasons:

1. longitudinal data releases are often published as separate Datasets,
2. any minor changes, i.e. metadata updates or additional documentation disseminated, result in a Dataverse version change, but not in a Release version change (no changes to the data files)

Some studies require a separate DOI per release/wave thus have more than one dataverse created for the study. The title will indicate wave/release number and version of the release. The metadata field for related studies will link to other release/waves of the data. Since longitudinal data releases already have their version number to start with, e.g. Release 7, any changes between releases result in a minor version change, e.g. Release 7.0 → Release 7.1. That type of versioning is a result of data updates between releases (to fix errors, etc.) Longitudinal study version/release number can differ from the number of waves, since there could be more than one wave of data collection between data releases (e.g. LSAY). Versioning in longitudinal studies can be documented at the filename level as well, e.g. 7_2_2. LSAC General Release_R7-2.zip. An example of a citation with release and waves: The following study is the 7th release which has been altered twice containing waves 1-7 of corrected data and the dataverse has been altered 5 times since first published.

Department of Social Services; Australian Institute of Family Studies; Australian Bureau of Statistics, 2018, "Growing Up in Australia: Longitudinal Study of Australian Children (LSAC) Release 7.2 (Waves 1-7)", doi:10.26193/F2YRL5, ADA Dataverse, V5

38 Data Versioning Workflow of GFZ Data Services (#GFZ)

Contributed by Kirsten Elger, Damian Ulbricht (GFZ German Research Centre for Geosciences, Potsdam, Germany)

We are currently using two different ways for versions of DOI-referenced data (1) new DOI for the new version and (2) new version number for the same DOI

New DOI number for each version

This was our first approach and is applied for occasional updates of data, e.g. error correction or model update:

- The new version is identified with a new DOI and is cross-linked with the old version via the DataCite relatedIdentifiers ("IsNewVersion" and "IsPreviousVersionOf")
- Both versions have the same title. Version numbers are either added to the title or populate the DataCite version field and the abstract(e.g. http://doi.org/10.5880/ICDP.5054.002 http://doi.org/10.5880/icgem.2016.008)
- When the DOI metadata of the old version links to the new version through a related identifier (e.g. "IsPreviousVersionOf") the DOI landing page creates a prominent link to the new version(see example http://doi.org/10.1594/GFZ.SDDB.ICDP.5054.2015
- In addition, we add a version history to the abstract describing the purpose of the new release and the changes, e.g. http://doi.org/10.5880/ICDP.5054.002
- In most cases, we don't include the version number in the title, but are using the "Version" field of DataCite. This means that the version appears in the citation without changing the title

Figure 34. A example of data record with version information

New version for the same DOI number

The procedure described here is applied when the processing of the data did not change significantly. For instance, when additional time frames are added to time series or when additional information is added. In these cases new DOIs would populate metadata catalogues with identical information, making the discovery of datasets in overarching catalogues (e.g. DataCite or Google Dataset Search) difficult.

At GFZ we have a small number of DOIs (e.g. http://doi.org/10.5880/FIDGEO.2019.010) where authors created additional information after publication. Furthermore, we assigned DOIs to >100 micrometeorological stations that are part of the TERENO network and plan to use this version scheme for future updates.

Therefore we have developed a workflow with the same DOI and provide the previous versions via the data download folder:

- For the old version, we combine the data, data description and XML metadata (ideally plus checksum) in one zip folder
- This zip folder is accessible in a subfolder named “previous versions”
- The updated metadata includes the new version and a version history with release date
- The DOI, title and publication year (as defined in the DataCite Metadata Schema) remain the same
- Example: http://doi.org/10.5880/FIDGEO.2019.010
- Download folder of the DOI above:

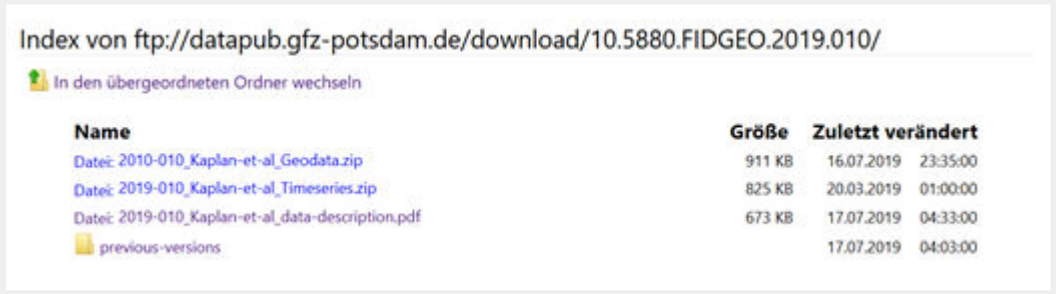


Figure 35. Use file folder to organise versioned data files

Folder previous-versions:

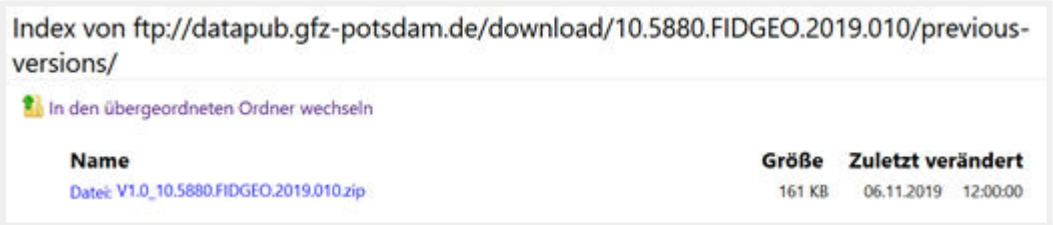


Figure 36. All older versions are made available through the “previous-versions” folder

- In the example shown, only one part of the data are updated (the geodata). The time series remain the same and are not included in the V1.0 folder
- The version history (in the abstract and the data description) “17. July 2019: release of Version 2.0. This version includes additionally the catchment boundaries provided as subfolder of geodata.zip. The version 1.0 is available in

the "previous-versions" subfolder via the Data Download link. The time series did not change and are not included in the V1.0 zip folder."

Some comments:

- For regular updates of data (e.g. additional months of time series TOGETHER with an update of the previously published data, the second approach is preferred, mainly to avoid flooding of data catalogues. Of course, we could implement mechanisms avoiding it for GFZ Data Services, but this won't happen in other catalogues (DataCite, Google Dataset Search, EPOS…)
- We don't assign new DOIs for dynamic data if the time series are only growing (only if there are changes in the already published data).

D) Use Cases Contributed After Publication of Version 1 of This Document

39 Google dataset search recommendations for developers (#Google)
https://developers.google.com/search/docs/data-types/dataset

Source and provenance best practices

It is common for open datasets to be republished, aggregated, and to be based on other datasets. This is an initial outline of our approach to representing situations in which a dataset is a copy of, or otherwise based upon, another dataset.

- Use the `sameAs` property to indicate the most canonical URLs for the original in cases when the dataset or description is a simple republication of materials published elsewhere. The value of `sameAs` needs to unambiguously indicate the dataset's identity - in other words two different datasets should not use the same URL as `sameAs` value.

- Use the `isBasedOn` property in cases where the republished dataset (including its metadata) has been changed significantly.
- When a dataset derives from or aggregates several originals, use the `isBasedOn` property.
- Use the `identifier` property to attach any relevant Digital Object identifiers (DOIs) or Compact Identifiers. If the dataset has more than one identifier, repeat the `identifier` property. If using JSON-LD, this is represented using JSON list syntax.

40 Figshare Versioning Guidelines

https://help.figshare.com/article/can-i-edit-or-delete-my-research-after-it-has-been-made-public

Figshare supports versioning for both items and collections. Because projects are a continuous piece of work with a start date and finish date they are not candidates for versioning.

There are a couple of changes that would trigger versioning. These rules are a bit different between items and collections.

Items

Provided it is a public Figshare item, the following actions will trigger versioning when saving publicly:

- Modifying the title
- Add/edit/remove author/s
- Changing the licence
- Adding new files
- Removing files
- Replacing files
- Removing permanent embargoes from files
- Removing the metadata only flag and uploading files
- Replacing the link associated with a linked item

Collections

Provided it is a public collection, the following actions will trigger versioning, when saving publicly:

- Modifying the title
- Changing the licence
- Adding new items
- Removing items
- Replacing items
- Add/edit/remove author/s
- Upgrading the item's version linked to the collection

These actions can be done from the website, the API, or any submission method used by the user. The user can modify the title, for example, but also another field that would not trigger a new version. In this situation, since the title has been changed, we would generate a new version that would have both the new title and the other metadata changed.

The same rules apply to individual Figshare accounts and to institutional accounts or publisher accounts.

For accounts owned by publishers, Figshare is able to remove the automatic versioning process and maintain only one public version.

Versions are listed and accessible in the drop-down menu under the item title and each is timestamped.

41 OSF Version Control

https://help.osf.io/article/149-version-control

Selecting Systems that Work in Theory and Practice

Version control systems vary widely in capabilities and complexities, from automatic tools that sync sequential versions of your files with various cloud storage offerings, to tools like git (see also gitlab, github) that allow for active management of multiple branching and merging versions of the same file. Both the theoretical capabilities of the tool and how those capabilities are to be utilized in practice should be considered.

When selecting a version control system for your research, the primary consideration should be what can be most consistently implemented in your current environment. If your chosen tools are too challenging to use or are not available on all platforms where research materials need to be managed, individual researchers may begin use of their own tools and systems, resulting in the same kind of confusion that you hoped to avoid. Existing familiarity with a particular tool, availability of tools, and ease of configuring those tools on new devices are all important considerations when selecting a version control system.

Once you have identified what systems can be implemented consistently in your research environment, some additional considerations are worth including in your evaluation. In addition to general software selection considerations (price, size of the community using and supporting the tool, amount of community control, Free and Open Source (FOSS) licensing terms, etc), some specific questions to ask follow.

Questions to Ask

- How much complexity is functionally required by your particular task and workflows? For instance collaborative software development may benefit from more complex tools than are needed to track sequential versions of a data file during analysis.
- How many versions do you need to keep and for how long? Some tools may only allow you to keep a certain number of versions, to only keep them for a certain period of time, or may limit whether you can migrate those versions to other tools. Ensure that any limitations are in line with your research and archival needs or select tools without such limits.

- How much of the metadata and provenance tracking data specified in your Data Management Plan will this tool record as it is actually implemented in your workflows? If your team delegates all interactions with the version management system to a small number of team members, your version control system may end up with less complete or less accurate information than if everyone interacted with it directly. For instance, if you have a single data manager who enters all the files into the group version control tool, it may be unclear who authored particular changes and when, whereas if each researcher had entered their own changes directly, that information would have been automatically captured by the version control tool.

42 Versioning of Vocabularies

Cox SJD, Gonzalez-Beltran AN, Magagna B, Marinescu M-C (2021) Ten simple rules for making a vocabulary FAIR. PLoS Comput Biol 17(6): e1009041. https://doi.org/10.1371/journal.pcbi.1009041

Rule 10. Implement a process for publishing revisions of the FAIR vocabulary

The FAIR vocabulary should be created and maintained so that it reflects the content and updates agreed and issued by the content custodian, so it is important to obtain the maintenance schedule and versioning strategy for the vocabulary from the content custodian (Rule 1).

We recommend updating the FAIR vocabulary as soon as practical after the content custodian updates the legacy vocabulary. If the content custodian wishes to maintain the content in its original form (i.e. the legacy vocabulary), then try to arrange for alerts advising you of changes to be issued by the custodian, in order to trigger the process of update of the FAIR vocabulary. However, it may be possible to transition to an arrangement in which the FAIR vocabulary becomes the primary version or 'point of truth' for the content, in which case individual revisions should be proposed and tracked in a traceable maintenance environment (see Rule 4). However, this should only be done with the consent of the content custodian. Note that as well as improved tracking of revisions, some kinds of improvement may be supported better in the FAIR representation (see Rule 6) than on the legacy (print-based) platform, including specific relationships between terms, mappings to other vocabularies, and detailed axiomatization of definitions.

If revision of the vocabulary is by new releases of the vocabulary-as-a-whole, then status and version information will be in the vocabulary metadata (see Rule 7). If maintenance is continuous, then the per-term metadata should capture its status and version information (see Rule 6). Standard Dublin Core, SKOS and OWL properties that may be useful in versioning include:

dcterms:created—date or date-time that the vocabulary or term was initially created

dcterms:modified—date or date-time that the vocabulary or term was last updated

dcterms:isReplacedBy—to point to a superseding vocabulary or term

dcterms:replaces—to point to a prior version of a vocabulary or term

owl:deprecated = 'true' if the vocabulary or term is no longer valid

owl:priorVersion—to point to a previous version of a vocabulary

owl:versionInfo—general annotations relating to versioning

skos:changeNote—modifications to a term relative to prior versions

skos:historyNote—past state/use/meaning of a term

Do not re-assign or remove identifiers; they are persistently associated with the term to which they were originally assigned (Rule 5). If necessary, you can deprecate or retire an identifier. However, the IRI for every retired and superseded term must remain de-referenceable, as well as for previous versions of the vocabulary, so that references to them still return a result, annotated with the status.

Terms that carry over between releases without the definition changing must retain the same IRI. If the IRI were changed, then datasets that use different versions of the same vocabulary cannot interoperate. Consult with the content custodian to clarify the 'identity-determining' characteristics of terms, but note that changing relationships (e.g. position in a hierarchy) or the textual definition do not necessarily require changing the identifier (i.e. minting a new IRI) provided that the intention for the concept is still the same.

43 A Guide to Using GitHub for Developing and Versioning Data Standards and Reporting Formats

Crystal-Ornelas, R., Varadharajan, C., Bond-Lamberty, B., Boye, K., Burrus, M., Cholia, S., et al. (2021). A Guide to Using GitHub for Developing and Versioning Data Standards and Reporting Formats. Earth and Space Science, 8(8), e2021EA001797. https://doi.org/10.1029/2021EA001797

4.1.4 Versioning

We recommend semantic versioning (e.g., v1.0.1) be used to track updates to version-controlled data standard documents (Preston-Werner, 2020). By using the built-in GitHub "Release" feature, repository managers can save a snapshot of their GitHub repository at a point in time and assign a version number that aligns with semantic version conventions (Figure 3). Clear version numbers enable users to identify when they need to migrate their data to an updated standard or locate and download previous versions of the documentation. When repository managers choose to publish a "Release," we also recommend that they archive their data standard documents in a long-term data repository. If no domain-specific archive exists, then GitHub's integration with Zenodo can be used to instantly archive GitHub repository content and also generate a recommended citation. Some data standard developers may choose to version terms and vocabularies used within their data standard, separately from the standard itself (e.g.,

https://github.com/tdwg/vocab/blob/master/vms/maintenance-specification.md or https://cfconventions.org/standard_name_rules.html). Decoupling vocabulary and data standard versioning can be an effective way to communicate with users when different aspects of the standard change (e.g., specific vocabulary terms vs. supporting documentation). As community data standards are used, tested, and feedback is generated, developers should be prepared for standards to be updated and changed over time. In addition to the semantic versioning described above, we recommend that changes to data standards be documented using one or more of the following approaches: listing the latest updates in the repository's README file, describing changes in local commits, providing pull request descriptions, referencing issue numbers in commit or pull request messages, or creating a GitHub changelog to provide details on data standard updates.

44 Data diffs: Algorithms for explaining what changed in a dataset

https://blog.marcua.net/2022/02/20/data-diffs-algorithms-for-explaining-what-changed-in-a-dataset.html

tl;dr: part 1 explains what an explanation algorithm is, and part 2 describes an open source SQL data differ.

"Why did this happen?" "What changed?"

In the data world, most reporting starts by asking *how much?*: "how many new customers purchase each week?" or "what is the monthly cost of medical care for this group?"

Inevitably the initial reports result in questions about *why?*: "why did we see less purchases last week?" and "why are the medical costs for this group increasing?"

The academic community has an answer to such *why?* questions: explanation algorithms. An explanation algorithm looks at columns/properties of your dataset and identifies high-likelihood explanations (called "predicates" in database-speak). For example, the algorithms might find that you got less customers in the segment of people who saw a new marketing campaign, or that the medical costs for the group you're studying can largely be attributed to costly treatments in a subgroup.

The academic interest is founded in real pain. When a journalist, researcher, or organization asks *why?*, the resulting data anlysis largely goes into issuing ad hoc GROUP BY queries or unscientifically creating pivot tables to try to slice and dice datasets to explain some change in a dataset over time. Companies like Sisu (founded by Peter Bailis, one of the authors of the DIFF paper discussed below) are built on the premise that data consumers are increasingly asking *why?*

You can rephrase lots of different questions in the form of an explanation question. This is an area I've been interested in for a while, especially as it might help people like

journalists and social scientists better identify interesting trends. In A data differ to help journalists (2015), I said:

> It would be nice to have a utility that, given two datasets (e.g., two csv files) that are schema-aligned, returns a report of how they differ from one-another in various ways. The utility could take hints of interesting grouping or aggregate columns, or just randomly explore the pairwise combinations of (grouping, aggregate) and sort them by various measures like largest deviation from their own group/across groups.

At the time of that post, I hadn't yet connected the dots between the desire for such a system and the active work going on in the research world. Thanks to database researchers, that connection now exists! In this post, I'll first cover two approaches to explanation algorithms, and then introduce an open source implementation of one of them in my datools library.

Two ways to ask for explanations

In 2013, Eugene Wu and Sam Madden introduced Scorpion, a system that explains why an aggregate (e.g., the customer count last week) is higher or lower than other example data. Figure 1 in their paper explains the problem quite nicely. They imagine a user looking at a chart, in this case of aggregate temperatures from a collection of sensors, and highlighting some outliers to ask "compared to the other points on this chart, why are these points so high?"

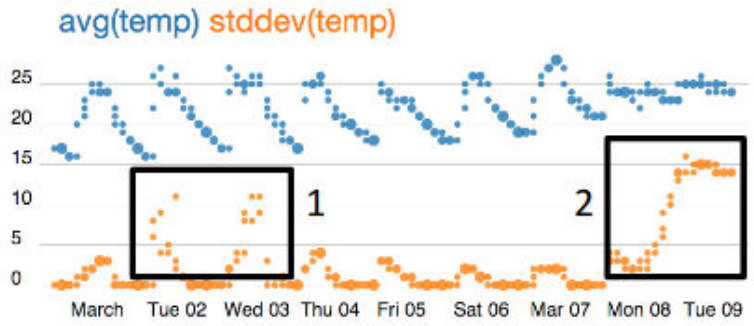


Figure 1: Mean and standard deviation of temperature readings from Intel sensor dataset.

A figure that shows how a user might highlight outliers on a chart (source: Scorpion paper)

Scorpion has two nice properties. First, it operates on aggregates: it's not until you look at some weekly or monthly statistics that you notice that something is off and search for an explanation. Second, it's performant on a pretty wide variety of aggregates, with optimizations for the most common ones (e.g., sums, averages, counts, standard

deviations). I believe that of all the explanation algorithms, Scorpion pairs the most intuitive phrasing of the question (“why so high/low?”) with the most intuitive experience (highlighting questionable results on a visualization).

The challenge in implementing Scorpion is that, as presented, it does its processing outside of the database that stores the data. Specifically, the way Scorpion partitions and merges subsets of the data to identify an explanation requires decision trees and clustering algorithms that traditionally execute outside of the database[1]. It is also specific to aggregates, which are commonly the source of *why* questions, but aren’t the only places that question arises.

This is where DIFF comes in[2]. In 2019, Firas Abuzaid, Peter Kraft, Sahaana Suri, Edward Gan, Eric Xu, Atul Shenoy, Asvin Ananthanarayan, John Sheu, Erik Meijer, Xi Wu, Jeff Naughton, Peter Bailis, and Matei Zaharia introduced an explanation algorithm in the form of a database operator called DIFF that can be expressed in SQL. If you’re so inclined, here’s the syntax for the DIFF operator:

```
diff_query = table_ref "DIFF" table_ref
             "ON" {attrs}
             "COMPARE BY" {diff_metric([args]) > threshol
             ["MAX ORDER" integer] ;

diff_metric = "support" | "odds_ratio" | "risk_ratio"
            | "mean_shift" | udf
```

Figure 1: DIFF syntax in extended Backus-Naur F

The syntax for the DIFF operator (source: DIFF paper)

An example with SQL might help in understanding how it works:

```
SELECT * FROM
  (SELECT * FROM logs WHERE crash = true and timestamp BETWEEN
       08-28-18 AND 09-04-18) this_week
DIFF
  (SELECT * FROM logs WHERE crash = true and timestamp BETWEEN
       08-21-18 AND 08-28-18) last_week
ON app_version, device_type, os
COMPARE BY risk_ratio >= 2.0, support >= 0.05 MAX ORDER 3;
```

which yields the following result:

app_version	device_type	os	risk_ratio	support
v3	Galaxy S9	11.0	20.0	75%

A simple example of the DIFF operator in action (source: DIFF paper)

In this example, the DIFF operator compares the crash logs of an application from this week to those of last week, considering columns like application version, device, and operating system for an explanation. The most likely explanation happened 20x more

this week than last week (risk_ratio = 20.0), and explains 75% of this week's crashes (support = 75%).

DIFF requires that we do some mental gymnastics to transform "why was X so high?" into "how are these two groups different?". It also requires the user to wrap their head around statistics like risk ratios and support. In exchange for that mental overhead, DIFF is exciting for its praticality. As the example shows, DIFF's authors envision it being expressed in SQL, which means it could be implemented on top of most relational databases. While a contribution of the paper is a specialized and efficient implementation of DIFF that databases don't have today, it can also be implemented entirely in the database as a series of SQL GROUP BY/JOIN/WHERE operators.

If you have a relational database, love SQL, and want to run an explanation algorithim, DIFF is exciting because those three things are all you need. Luckily for you, dear reader, I had a relational database, loved SQL, and wanted to run an explanation algorithm.

An open source implementation of DIFF

Over the past few months, I've been implementing DIFF as a thin Python wrapper that generates the SQL necessary to compute the difference between two schema-aligned queries. The core of the implementation to do this, including comments, requires a little under 300 lines of code. To see a full example of the tool in action, you can check out this Jupyter Notebook, but I'll show snippets below to give you a sense of how it works.

First, we need a dataset. For that, I took inspiration from the Scorpion paper's experiments, one of which relied on sensor data from Intel collected by my grad school advisor Sam Madden (and a few collaborators). Using Simon Willison's excellent sqlite-utils library, I load the data into SQLite and inspect it:

```
# Retrieve and slightly transform the data

wget http://db.csail.mit.edu/labdata/data.txt.gz

gunzip data.txt.gz

sed -i '1s/^/day time_of_day epoch moteid temperature humidity
light voltage\n/' data.txt

head data.txt


# Get it in SQLite

pip install sqlite-utils

sqlite-utils insert intel-sensor.sqlite readings data.txt
--csv --sniff --detect-types

sqlite-utils schema intel-sensor.sqlite
```

That last sqlite-utils schema shows us what the newly generated readings table looks like:

```
CREATE TABLE "readings" (
   [day] TEXT,
   [time_of_day] TEXT,
   [epoch] INTEGER,
   [moteid] INTEGER,
   [temperature] FLOAT,
   [humidity] FLOAT,
   [light] FLOAT,
   [voltage] FLOAT
);
```

OK! So we have a row for each sensor reading, with the day and time_of_day it happened, an epoch to time-align readings from different sensors, a moteid (the ID of the sensor, otherwise known as a mote), and then the types of things that sensors tend to sense: temperature, humidity, light, and voltage.

In the Scorpion paper (Sections 8.1 and 8.4), a user notices that various sensors placed throughout a lab detect too-high temperature values (reading the experiment code, this happens in the days between 2004-03-01 and 2004-03-10). A natural question is why this happened. The Scorpion algorithm discovers that moteid = 15 (a sensor with ID 15) was having a bad few days.

Can we replicate this result with DIFF? Let's see! The DIFF implementation is part of a library I've been building called datools, which is a collection of tools I use for various data analyses. Let's install datools:

```
pip install datools
```

Now let's use it!

```
from sqlalchemy import create_engine
from datools.explanations import diff
from datools.models import Column
```

```
engine = create_engine('sqlite:///intel-sensor.sqlite')


candidates = diff(
        engine=engine,
          test_relation='SELECT moteid, temperature, humidity,
light, voltage FROM readings WHERE temperature > 100 AND day >
"2004-03-01" and day < "2004-03-10"',
                control_relation='SELECT moteid, temperature,
humidity, light, voltage FROM readings WHERE temperature <=
100 AND day > "2004-03-01" and day < "2004-03-10"',
        on_column_values={Column('moteid'),},
        on_column_ranges={},
        min_support=0.05,
        min_risk_ratio=2.0,
        max_order=1)
for candidate in candidates:
    print(candidate)
```

What's diff have to say?

```
Explanation(predicates=(Predicate(moteid        =        15),),
risk_ratio=404.8320855614973)
Explanation(predicates=(Predicate(moteid        =        18),),
risk_ratio=200.5765335449176)
```

Wow! moteid = 15 is the top predicate that datools.diff identified as being the difference between the test_relation and control_relation! With a risk_ratio = 404.83, we learn that sensor 15 is about 400 times more likely to appear in the set of records with high temperature readings than in the set of records with low temperature readings. Hooray for replicating the Scorpion result! Poor sensor 15!

Let's break that call to diff down a bit so we understand what's going on:

- engine: a SQLAlchemy engine that's connected to some database, in this case the SQLite database.
- test_relation: the "test set," which is a query with records that show a particular condition. In our case, it's the higher-temperature records during the period of interest. This could alternatively be a SQL query for "patients with high medical costs" or "customers who purchased."

- control_relation: the “control set,” which is a query with records that don’t show that particular condition. In our case, it’s the lower-temperature records during the period of interest. This could alternatively be a SQL query for “patients who don’t have high medical costs” or “leads who haven’t purchased.”
- on_column_values: these are set-valued columns you want to consider as explanations. In our case, we’re considering the moteid column, so we can identify a specific sensor thats misbehaving.
- on_column_ranges: these are range-valued columns you want to consider as explanations. diff will bucket these columns into 15 equi-sized buckets, which works well for continuous variables like {Column('humidity'), Column('light'), Column('voltage'),}. In this example, we don’t provide any (more on why later), but in the Jupyter Notebook, you can see this in action.
- min_support: The smallest fraction ([0, 1]) of the test set that the explanation should explain. For example, min_support=0.05 says that if an explanation doesn’t include at least 5% of the test set, we don’t want to know about it.
- min_risk_ratio: The smallest risk ratio that the explanation should cover. For example, min_risk_ratio=2.0 says that if an explanation isn’t at least 2 times as likely to appear in the test set than in the control set, we don’t want to know about it.
- max_order: How many columns to consider for a joint explanation. For example, in the Scorpion paper, the authors find that not just sensor 15 (one-column explanation), but sensor 15 under certain light and voltage conditions (three column-explanation), is the best explanation for outlier readings. To analyze three-column explanations, you’d set max_order=3. Sadly and hopefully temporarily, while max_order is the most fun, interesting, and challenging-to-implement parameter of the DIFF paper, datools.diff only supports max_order=1 for now.

An astute reader will note that I coaxed the results in my example a bit by asking DIFF to consider only moteid explanations (on_column_values={Column('moteid'),}). The Scorpion paper considers the other columns as well and still gets the strongest signal from moteid. In the Jupyter Notebook, we dive into this more deeply and run into an issue replicating the Scorpion results with diff. I offer some hypotheses for this in the notebook, but to have a more informed opinion, we’ll have to wait until datools.diff supports max_order > 1.

Where to go from here?

Before we go off and celebrate the replication of the Scorpion paper’s findings with the DIFF paper’s algorithm, you should know that it’s not all roses. Luckily, I’m just as excited about improving datools.diff as I was when I first wrote it, so consider the list below to be both limitations of the current version and a roadmap for the library. If you’re curious, this project board tracks the things I’m working on most actively.

- **Make diff work on more than just SQLite**. diff generates SQL, and I’d love for that SQL to run on any database. This is largely a matter of improving the test

harness to provision other databases and fixing whatever breaks. The next few databases I'm targeting are DuckDB, Postgres, and Redshift, but if you're interested in collaborating on something else, I'd love to help.

- **Support max_order > 1**. One of the DIFF paper's contributions is in how to spar with the combinatorial explosion you encounter in looking for multi-column explanations. I'd love to support at least 2- or 3-column explanations.
- **Use diff on more datasets**. If you've got a dataset (especially a public one) you're hoping to try this on, let me know!
- **Replicate diff on Scorpion's analysis after implementing higher-order explanations**. The full Jupyter Notebook shows that diff can't yet replicate Scorpion's results when we ask it to consider more columns than moteid. The notebook offers explanations ranging from "DIFF and Scorpion are different algorithms and have different tradeoffs" to "Why are we considering an output measure as an explanation?" I think it's worth revisiting this after implementing max_order > 1, so that we can see how datools.diff handles more complex explanations.
- **Share more about datools**. diff is part of the datools package, but I haven't told you much about datools. Countless words have been spilled about how SQL, despite being here to stay, also has its rough edges. datools smooths some of these rough edges out[3].

Footnotes

1. Strictly speaking, it doesn't have to be the case that more complex analytics or machine learning algorithms have to be run outside the database. MADlib speaks to this nicely, although in practice the approach hasn't taken off as widely as I wish it did. ↩

2. This blog post is not a literature review of explanation algorithms. Across the statistics and databases communities, several bodies of work related to explanation algorithms have predated or proceeded in parallel with Scorpion and DIFF. These two algorithms just happen to have shaped my understanding of the space the most. ↩

3. As an example, not every database (I'm looking at you, SQLite and Redshift) supports things like grouping sets and data cubes, but these operators are critical for making tools like DIFF-in-SQL work effectively. datools offers wrappers that, if a database supports grouping sets, will use the native functionality, but if the database doesn't, will do the next best thing. ↩

45 Explaining Dataset Changes for Semantic Data Versioning with Explain-Da-V

Shraga, R., & Miller, R. J. (2023, January 30). Explaining Dataset Changes for Semantic Data Versioning with Explain-Da-V (Technical Report). arXiv. https://doi.org/10.48550/arXiv.2301.13095

In multi-user environments in which data science and analysis is collaborative, multiple versions of the same datasets are generated. While managing and storing data versions has received some attention in the research literature, the semantic nature of such changes has remained under-explored. In this work, we introduce \texttt{Explain-Da-V}, a framework aiming to explain changes between two given dataset versions. \texttt{Explain-Da-V} generates \emph{explanations} that use \emph{data transformations} to explain changes. We further introduce a set of measures that evaluate the validity, generalizability, and explainability of these explanations. We empirically show, using an adapted existing benchmark and a newly created benchmark, that \texttt{Explain-Da-V} generates better explanations than existing data transformation synthesis methods.

46 Helmholtz Metadata Collaboration, Guidance on Versioning of Digital Assets (HMC Paper 3)

Buttigieg, P. L., Cristiano, L., Curdt, C., Ihsan, A. Z., Jejkal, T., Koch, C., et al. (2022). Guidance on Versioning of Digital Assets (HMC Paper 3) (20 pp.). Kiel, Germany: HMC Office, GEOMAR Helmholtz Centre for Ocean Research. https://doi.org/10.3289/HMC_publ_04

Versioning of data and metadata is a crucial - but often overlooked - topic in scientific work. Using the wrong version of a (meta)data set can lead to drastically difference outcomes in interpretation, and lead to substantial, propagating downstream errors. At the same time, past versions of (meta)data sets are valuable records of the research process which should be preserved for transparency and complete reproducibility. Further, the final version of (meta)data sets may actually include errors that previous versions did not. Thus, careful version control is the foundation for trust in and broad reusability of research and operational (meta)data. This document provides an introduction to the principles of versioning, technical recommendations on how to manage version histories, and discusses some pitfalls and possible solutions. In the first part of this document, we present examples of change processes that require proper management and introduce popular versioning schemes. Finally, the document presents recommended practices for researchers as well as for infrastructure developers.

47 PAV - Provenance, Authoring and Versioning

Contributed by Melinda Hodkiewicz (University of Western Australia, Australia)

Paolo Ciccarese, Stian Soiland-Reyes, Khalid Belhajjame, Alasdair JG Gray, Carole Goble, Tim Clark (2013): **PAV ontology: provenance, authoring and versioning**. *Journal of biomedical semantics* **4**:37. doi:10.1186/2041-1480-4-37

Provenance is a critical ingredient for establishing trust of published scientific content. This is true whether we are considering a data set, a computational workflow, a

peer-reviewed publication or a simple scientific claim with supportive evidence. Existing vocabularies such as Dublin Core Terms (DC Terms) and the W3C Provenance Ontology (PROV-O) are domain-independent and general-purpose and they allow and encourage for extensions to cover more specific needs. In particular, to track authoring and versioning information of web resources, PROV-O provides a basic methodology but not any specific classes and properties for identifying or distinguishing between the various roles assumed by agents manipulating digital artifacts, such as author, contributor and curator.

PAV is a lightweight ontology for tracking Provenance, Authoring and Versioning. PAV specializes the W3C provenance ontology PROV-O in order to describe authorship, curation and digital creation of online resources.

This ontology describes the defined PAV properties and their usage. Note that PAV does not define any explicit classes or domain/ranges, as every property is meant to be used directly on the described online resource.

48 Keeptachangelog.org

Lacan, O. (2023). Keep a Changelog. Retrieved June 19, 2024, from https://keepachangelog.com/en/1.1.0/

What is a changelog?

A changelog is a file which contains a curated, chronologically ordered list of notable changes for each version of a project.

Why keep a changelog?

To make it easier for users and contributors to see precisely what notable changes have been made between each release (or version) of the project.

Who needs a changelog?

People do. Whether consumers or developers, the end users of software are human beings who care about what's in the software. When the software changes, people want to know why and how.

https://keepachangelog.com/en/1.1.0/

49 Oxford Common File Layout (OCFL)

Contributed by Martin Fenner (Front Matter, Germany).

Woods, A., & Jeffries, N. (2023). OCFL · GitHub (Version 1.1). Retrieved from https://github.com/OCFL

The Oxford Common File Layout (OCFL) specification describes an application-independent approach to the storage of digital information in a structured,

transparent, and predictable manner. It is designed to promote long-term object management best practices within digital repositories.

A need expressed by the community was the need to update and change objects, either the content itself or the metadata associated with the object. The OCFL relies heavily on the prior art in the Moab Design for Digital Object Versioning which utilizes forward deltas to track the history of the object. Utilizing this schema allows implementers of the OCFL to easily recreate past versions of an OCFL object. Like with objects, the OCFL remains silent on when versioning should occur recognizing this may differ from implementation to implementation.

50 The Moab Design for Digital Object Versioning

Anderson, R. (2013). The Moab Design for Digital Object Versioning. The Code4Lib Journal, (21). Retrieved from https://journal.code4lib.org/articles/8482

The Moab Design for digital object versioning was born out of an urgent need to figure out an efficient, practical, and robust mechanism for making changes to digital objects that had already been ingested. The need for a versioning capability was holding us back from fixing some problems that had already occurred, or opening up the repository door to new sources of documents that we anticipated might get modified in a relatively short time.

The nature of a preservation repository turns out to be different enough from that of a software change management system or file backup system that some technologies which initially appeared promising (such as Git and Boar) turned out to have design aspects or behaviors that ruled them out as solutions that we could adopt off of the shelf. The CDL Micro-Services approach had the best promise and well-written specifications, but we were disappointed when we realized that the ReDD reverse-delta mechanism would likely prove too resource consumptive for us. Inspiration was derived, however, from all of those designs and the result of assembling pieces from each approach into a new framework has yielded rewarding results.

Recapping the criteria used in section 3.6 [of the paper] to evaluate those other technologies, here is how the Moab design measures up:

• File Fixity: Good

Moab does not alter any content of object files so that the fixity remains consistent throughout the system. Manifests record file sizes and checksums for use in verification.

• No File Duplicates: Good

By using file signatures as the key to file identity, Moab guarantees that only one copy of any given file manifestation is stored at any given storage or replication location.

• Replication Efficiency: Good

For the same reason, the transmission, processing, and storage of a new version's files only needs to involve the files that were newly introduced in that version.

• Version Reconstruction: Good

Any version of a digital object can be reconstructed and disseminated with equal ease to any other version.

51 Dryad's data versioning feature

Hirst, M. (2024, July 9). For authors: Keep your data current with Dryad's data versioning feature. https://blog.datadryad.org/2024/07/09/for-authors-keep-your-data-current-with-dryads-data-versioning-feature/

Learn when, why, and how to version your dataset on Dryad. Versioning is crucial for proper data management. First, it ensures reproducibility and transparency in research by documenting all changes to the data, thus maintaining trust in the scientific process. This comprehensive documentation allows other researchers to accurately understand and replicate studies. Secondly, compliance with funders and publishers is greatly facilitated by versioning. Many funding bodies and publishers mandate that data be regularly updated and made accessible, and versioning helps meet these requirements by enabling efficient data sharing and preservation. Lastly, versioning enhances the discoverability and citability of datasets. By assigning a unique, permanent DOI to each dataset, regardless of its version, both current and previous versions remain accessible and are linked to a single DOI. This ensures that data remains discoverable and usable over time, making it easier for researchers to locate and cite the datasets they need.

Researchers are continually refining, updating, and building upon their work – this is the goal of research at its core. Versioning datasets, and including an adequate change log, is a way to capture this continual learning process and allow for better collaboration throughout the research community. Taking the time to update your data and its associated metadata enhances the usability, reliability, and impact of your research. Dryad's dataset versioning feature is easy to use and enables you to adopt best practices for reproducibility and transparency in your research.

52 ConVer-G: Concurrent versioning of knowledge graphs

Gil, J. P., Coquery, E., Samuel, J., & Gesquiere, G. (2024, September 6). ConVer-G: Concurrent versioning of knowledge graphs. Databases, arXiv. https://doi.org/10.48550/arXiv.2409.04499

The multiplication of platforms offering open data has facilitated access to information that can be used for research, innovation, and decision-making. Providing transparency and availability, open data is regularly updated, allowing us to observe their evolution over time.

We are particularly interested in the evolution of urban data that allows stakeholders to better understand dynamics and propose solutions to improve the quality of life of citizens. In this context, we are interested in the management of evolving data, especially urban data and the ability to query these data across the available versions. In order to have the ability to understand our urban heritage and propose new scenarios, we must be able to search for knowledge through concurrent versions of urban knowledge graphs.

This work presents the ConVer-G (Concurrent Versioning of knowledge Graphs) system for storage and querying through multiple concurrent versions of graphs.

53 DataLad: distributed system for joint management of code, data, and their relationship

***Contributed by Adina Wagner* (Forschungszentrum Jülich, Germany)**

Halchenko, Y. O., Meyer, K., Poldrack, B., Solanky, D. S., Wagner, A. S., Gors, J., et al. (2021). DataLad: distributed system for joint management of code, data, and their relationship. Journal of Open Source Software, 6(63), 3262, https://doi.org/10.21105/JOSS.03262

DataLad is a Python-based tool for the joint management of code, data, and their relationship, built on top of a versatile system for data logistics (git-annex) and the most popular distributed version control system (Git). It adapts principles of open-source software development and distribution to address the technical challenges of data management, data sharing, and digital provenance collection across the life cycle of digital objects. DataLad aims to make data management as easy as managing code. It streamlines procedures to consume, publish, and update data, for data of any size or type, and to link them as precisely versioned, lightweight dependencies. DataLad helps to make science more reproducible and FAIR. It can capture complete and actionable process provenance of data transformations to enable automatic re-computation. The DataLad project (datalad.org) delivers a completely open, pioneering platform for flexible decentralized research data management (RDM). It features a Python and a command-line interface, an extensible architecture, and does not depend on any centralized services but facilitates interoperability with a plurality of existing tools and services. In order to maximize its utility and target audience, DataLad is available for all major operating systems, and can be integrated into established workflows and environments with minimal friction.

54 git-annex

Contributed by Adina Wagner (Forschungszentrum Jülich, Germany)

Hess, J. (2025). git-annex (Version 10.20250102).

https://git-annex.branchable.com/

git-annex allows managing large files with git, without storing the file contents in git. It can sync, backup, and archive your data, offline and online. Checksums and encryption keep your data safe and secure. Bring the power and distributed nature of git to bear on your large files with git-annex.

git-annex is designed for git users who love the command line. For everyone else, the git-annex assistant turns git-annex into an easy to use folder synchroniser.

55 TIB Blog: ***The role of PIDs for secondary publications: More than a technical identifier***

https://blog.tib.eu/2024/10/25/the-role-of-pids-for-secondary-publications-more-than-a-technical-identifier/ *(by Frauke Ziedorn)*

Frauke Ziedorn argued in this blogpost that a secondary publication should be allocated a PID (such as DOI) in order to make the secondary publication clearly identifiable and permanently accessible. An example of a secondary publication could be the re-publication of a previously published work, for example, authors publish a copy of a work on their own website or their institutional repository often differ in layout and format from the publisher's primary version.

In the dataset context, a secondary republication can be the same expression but with different manifestation (format) from its primary publication, or be a different expression (with content revision) from its primary publication.