Rillor

METHODS NOTE

Versioning a dataset when its sources change

Dataset versioning for data whose sources revise and withdraw records: revision and release, two time axes, time travel retention, version numbers and change records, from published standards.



Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Dataset versioning lets a past state of a dataset be rebuilt and named after its sources revise records, withdraw them or change their terms.

- Sources change after capture. In February 2026 the U.S. Bureau of Labor Statistics revised its March 2025 estimate of seasonally adjusted total nonfarm employment by −898,000, or −0.6 percent, and Pew Research Center found that 38% of webpages that existed in 2013 were no longer accessible in October 2023 [31, 32].
- A revision and a release are separate acts. A CODATA paper defines a revision as any change to a dataset's bitstream and a release as an editorial choice to publish one revision, described by what changed and what that implies for use [1].
- Two time axes answer two questions. SQL:2011 separates application time, the period a fact is believed valid in the real world, from system time, the period the database held it, and a bitemporal table keeps both [13].
- Time travel keeps history only while retention settings allow. The documented defaults run from 1 day to 30 days once retention or expiry applies (derived), and an expired snapshot can no longer be queried [19, 21, 22, 23, 24].
- Version numbers for data need written rules. Croissant, a dataset format specification from MLCommons, uses PATCH when the data is unchanged and only packaged differently, MINOR when existing data can still be retrieved as it was, and MAJOR when existing records were edited, removed or moved across splits [10].
- Point-in-time correctness depends on knowing when each fact was known. The FISD Alternative Data Council defines look-ahead bias as the error from using data that became known only after the date being analyzed [29].
- A withdrawal needs a notice that travels with the data. Two datasets retracted in April 2019 were still used, with their derivatives, 54 and 73 times in papers published in 2020, in one study's 20% sample [35].

Background

Why one-time extracts go stale

A dataset is often changed "to correct errors, apply new algorithms, or add new data" [1]. When its sources change as well, an extract taken once drifts away from them.

Statistical agencies revise. The U.S. Bureau of Labor Statistics (BLS) benchmarks its establishment survey each year to employment counts "derived primarily from the administrative file of employees covered by Unemployment Insurance (UI)", published through the Quarterly Census of Employment and Wages (QCEW) [31]. With its January 2026 data, released on 11 February 2026, it revised March 2025 seasonally adjusted total nonfarm employment by −898,000, or −0.6 percent [31]. Over the prior decade the revision averaged 0.2 percent in absolute terms [31]. The Federal Reserve Bank of Philadelphia keeps vintages, or snapshots, of major macroeconomic series so that researchers can "verify empirical results", and updates them at the end of each month [17].

Web pages disappear. Pew Research Center found that a quarter of all webpages that existed at one point between 2013 and 2023 were no longer accessible as of October 2023 [32]. Some 38% of pages that existed in 2013 were gone, against 8% of pages from 2023, and 54% of Wikipedia pages had at least one reference link to a page that no longer exists [32].

Terms change as well as records. An audit of 14,000 web domains, reported in "Consent in Crisis: The Rapid Decline of the AI Data Commons", found that in a single year (2023 to 2024) about 5% or more of all tokens in the C4 web corpus, and "28%+ of the most actively maintained, critical sources in C4", became fully restricted from use [33]. Terms of service restricted crawling for 45% of C4 [33].

Some sources change continuously. The SEC's EDGAR APIs "are updated in real-time as filings are disseminated", with a typical processing delay of under a second for the submissions API and under a minute for the XBRL APIs [30]. Its bulk ZIP files are republished nightly at about 3:00 a.m. ET [30].

What versioning has to answer

The CODATA paper notes that a lack of best practice for data versioning "has been recognised as a contributing factor" in what has been called a reproducibility crisis [1]. It asks that users be able "to identify the exact version of the data and any data extract that was used" [1]. The Research Data Alliance (RDA) sets the goal of citing and retrieving data "as it existed at a certain point in time, whether the database is static or highly dynamic" [2]. Datasheets for Datasets asks dataset creators whether there is an erratum, whether the dataset is updated, how often, by whom and how consumers hear of updates, and whether older versions stay supported or how their obsolescence is announced [8]. A versioning scheme for data with changing sources has to answer those questions for every state the data passes through.

This note is for data engineers, machine learning engineers and quantitative researchers who keep datasets whose sources keep changing, and who need training runs and backtests that can be rebuilt.

Method

What was examined. The published rules for each part of dataset versioning: how a revision differs from a release, how two time axes are stored and queried, how table formats and cloud data warehouses keep and expire history, how version numbers apply to data, how point-in-time correctness and its biases are defined, how withdrawn data is handled and what a change record holds.

Sources. Thirty-five sources, each opened on 8 November 2026 and read for the passages cited: six journal and conference papers, one preprint, eight standards and specifications, twelve pages of tool and repository documentation, one dataset card, five pages from the BLS, two Federal Reserve Banks and the SEC, one industry note and one research center report.

How. Each rule is described from the document that sets it. Figures and mappings marked "derived" are this note's own arithmetic or reading of the sources. Tool defaults are as documented on 8 November 2026 and can change after that date. No tool or format is ranked, and none is described as Rillor's.

Where sources disagree. The BLS article gives the not seasonally adjusted revision as −862,000, or −0.5 percent, in its text and as −861 thousand in its Table 4, after a reconstruction adjustment [31]. This note uses the seasonally adjusted −898,000. SQL:2011 periods are closed-open [13], and FRED's real-time periods are closed-closed [15]. The W3C's bus-stop example says the removal of a stop might trigger a major version and an error fix would likely trigger only a minor one [3], while Croissant classes any edit of existing data as MAJOR [10]. Practice also differs on whether a metadata change makes a new version [1]: Zenodo edits metadata in place and makes a new version only when the files change [7].

Findings

1. Revision versus release

Two acts. The CODATA paper analyzed "39 use cases and current practices of data versioning across 33 organisations" and proposes six principles: revision, release, granularity, manifestation, provenance and citation [1]. It defines a revision as "identifying that a change is made to the bitstream of a dataset", and the release of a dataset as "an editorial process that designates a particular revision being published for reuse" [1]. "Not all revisions lead to a new release" [1].

Revision. A new instance that differs from its precursor "should be separately (uniquely) identified" [1]. The paper notes that the RDA's data citation group already holds that "any change to a dataset creates a new version of the dataset that needs to be identified" [1]. The RDA's solution is "ensuring that data is stored in a versioned and timestamped manner": R1 applies versioning so that "earlier states of data sets can be retrieved", and R2 marks "any additions, deletions" with a timestamp [2].

Release. A new release "must be accompanied by a description of the nature and the significance of the change, along with a description of possible implications for use that could result from the change" [1]. The paper names Semantic Versioning as "a commonly used practice to communicate the significance of a version change" [1, 9].

Granularity and manifestation. Entire time series should be identified as collections, as should time-stamped revisions of a series that is updated frequently [1]. The same data in another file format or encoding has a different checksum, yet the work does not differ: they are "manifestations of the same work" [1]. Croissant's PATCH level covers the same case [10].

Citing a state. The RDA's 14 recommendations, R1 to R14, attach a persistent identifier to a timestamped query that can be re-executed against the timestamped data store [2]. The result set is sorted stably (R5), verified with a checksum of the query result set (R6) and given a new identifier "if the result set returned from an earlier identical query is different due to changes in the data" (R8) [2]. When data moves to a new representation, the queries and their fixity information migrate with it (R13), and the migration is verified (R14) [2].

Who decides. DCAT, the W3C's catalog vocabulary, leaves the choice to providers: "It is up to data providers to decide when and why a new version should be released" [4]. The W3C's best practices note that "there is no consensus about when changes to a dataset should cause it to be considered a different dataset" [3]. The CODATA authors conclude that data publishers "must publish their data versioning policies and procedures" [1].

Identifiers in repositories. DataCite suggests tracking "major_version.minor_version" and registering "a new identifier for a major version change", with each steward deciding which changes are major [6]. In Zenodo, a new version "is a completely new record with separate metadata, files and persistent identifier", linked to all

previous and later versions, so a researcher who cites one version can be sure its files did not change [7]. The W3C notes that a generic URI "always resolves to the latest version" [3].

2. Two time axes: valid time and system time

Definitions. Jensen and Snodgrass define the valid time of a fact as the times when it is true in the modeled world, which may lie in the past, present or future, and its transaction time as "the time when the fact is current in the database" [14]. Transaction time "has a duration: from insertion to (logical) deletion" [14].

SQL:2011. The standard was "published in December of 2011", and its temporal features sit in Part 2, SQL/Foundation [13]. Application-time period tables capture "time periods during which the data is believed to be valid in the real world" [13]. System-versioned tables serve applications that "must maintain an accurate history of data changes either for business reasons, legal reasons, or both" [13]. In them, any update or delete "must automatically preserve the old state of the row", the system maintains the start and end times of each row's period, and users cannot modify historical rows [13].

Queries. A system-versioned table answers FOR SYSTEM_TIME AS OF a timestamp, FROM ... TO, which is closed-open, and BETWEEN ... AND, which is closed-closed [13]. A query without a system-time clause returns "only the current system rows" [13]. Periods follow "a closed-open period model" [13].

Bitemporal tables. A table with both periods captures when facts "were believed to be true in the real world" and when they "were recorded in the database" [13]. The paper's example is a legal name change entered in the database after it took effect. Successive updates "can journal complex twists and turns in the state of knowledge captured by the database" [13].

Real-time periods in public statistics. The FRED API gives sources, releases, series and observations a real-time period that "marks when facts were true or when information was known until it changed" [15]. Its realtime_start and realtime_end default to today's date, which shows "what information about the past is available today", and ALFRED users can move the period to see what was known at a past date [15]. ALFRED retrieves "each economic data release (vintage) that was available on a specific date in history" [16]. FRED's period is "(closed, closed)" [15]. The Philadelphia Fed's real-time data set offers the complete vintage history and, for selected variables, first-, second- and third-release values [17].

Conventions. A timestamp on a period's end boundary falls inside a closed-closed period and outside a closed-open one, so the two conventions give different answers at the end of a period (derived) [13, 15]. A versioning policy states which one its timestamps follow.

Provenance vocabulary. PROV-O's prov:generatedAtTime and prov:invalidatedAtTime "can be used to bound the starting and ending moments of an Entity's existence" [5].

3. Snapshots, time travel and retention limits

Snapshots. The Apache Iceberg spec defines a snapshot as "The state of a table at some point in time, including the set of all data files" [18]. Each snapshot records a timestamp, its parent snapshot ID when it has a parent, and, from format version 2, a summary in which the operation is a required field [18]. The spec lists versions 1, 2 and 3 as "complete and adopted by the community", and states that version 4 "has not been formally adopted" [18].

Operations. Iceberg's four operations say what a snapshot changed [18]:

Operation	What the spec says it records
append	Only data files were added, and none were removed
replace	Files were added and removed without changing table data, as in compaction, a file format change or relocated files
overwrite	Files were added and removed in a logical overwrite
delete	Data files were removed and their contents logically deleted, or delete files were added to delete rows

Source: [18].

Two histories. Iceberg keeps a snapshot log of timestamp and snapshot ID pairs, and the parent-child lineage of its snapshots, which "might indicate different snapshot IDs for a specific timestamp" [18]. Point-in-time queries "should use 'snapshot-log' metadata", and when no snapshot exists before the timestamp, systems "should raise an informative error message about the missing metadata" [18].

Expiry. "Table snapshots expire and are removed from metadata to allow removed or replaced data files to be physically deleted" [18]. Snapshots accumulate until they are expired by the expireSnapshots operation, and expired snapshots "are no longer available for time travel queries" [21]. Data files are kept while any snapshot that may be used for time travel or rollback still refers to them [21].

Delta Lake. Delta Lake lists "Re-creating analyses, reports, or outputs (for example, the output of a machine learning model)" and "Fixing mistakes in your data" among the uses of time travel, with reads by versionAsOf or timestampAsOf [22]. A DataFrame loaded from a Delta table "may return different results across invocations if the underlying data is updated", and time travel fixes the data returned [22]. Travel to a version needs both its log and its data files, and data files are deleted only when VACUUM runs [22]. DESCRIBE HISTORY gives the operation, user and metrics of each write, and "Table history is retained for 30 days" [22].

Documented defaults.

System	Setting	Default	Basis
Iceberg	Snapshot age kept when expiring	5 days	[19]
Iceberg	Snapshots kept when expiring	1	[19]
Iceberg	Branch and tag retention	Forever, and the main branch never expires	[19, 20]
Delta Lake	Log retention	30 days	[22]
Delta Lake	Deleted file retention	7 days	[22]
Snowflake	Time travel retention	1 day, settable from 0 to 90 days for permanent objects on Enterprise Edition and higher	[23]
BigQuery	Time travel window	7 days, settable from 2 to 7	[24]

As documented on 8 November 2026.

Retention is a recovery window. When Snowflake's retention period ends, historical data moves into Fail-safe and "is no longer available for querying" [23]. In Snowflake's worked example, a version of a permanent table then stays in Fail-safe for 7 days, during which only Snowflake can recover it [23]. With retention set to 0, a transient table's changed data is deleted rather than moved into Fail-safe [23]. BigQuery keeps deleted data for an additional seven days of fail-safe, a period that "isn't configurable and can't be extended" [24]. The defaults run from 1 day to 30 days, with a 90-day maximum for permanent objects on Snowflake's Enterprise Edition (derived) [19, 22, 23, 24]. A version that a published result cites outlives those windows only with its own retained reference: a tag, an export with a checksum or a stored query (derived) [2, 20, 22].

Tags. Iceberg branches and tags are "named references to snapshots with their own independent lifecycles" [20]. Its examples retain weekly tags for 7 days, monthly tags for 180 days and a yearly tag forever, and "Tags can be used for retaining important historical snapshots for auditing purposes" [20].

Content hashes and pinned revisions. Git is "a content-addressable filesystem" that names each object by a SHA-1 checksum of its content plus a header [25]. A .dvc file records a hash for each tracked file or directory, MD5 for most locations and an ETag for some remote outputs [26]. MLflow records a digest, a "Unique

hash/fingerprint for dataset identification", for each logged dataset, computed differently for different dataset types [27]. The Hugging Face Datasets library loads a dataset at a Git tag, branch or commit hash through its revision parameter [28]. Each one pins a state by its content or by a named reference.

4. Version numbers for data

Semantic Versioning. Given a version number MAJOR.MINOR.PATCH, Semantic Versioning increments MAJOR "when you make incompatible API changes", MINOR when adding functionality in a backward compatible manner and PATCH for backward compatible bug fixes [9]. Once a version is released, "the contents of that version MUST NOT be modified", and any change is released as a new version [9].

Croissant. The Croissant format specification from MLCommons states that "Datasets may change over time" and combines "a version, and files checksums" for reproducibility [10]. It recommends MAJOR.MINOR.PATCH following Semantic Versioning, with these meanings [10]:

Level	When it applies in Croissant 1.1
PATCH	The data remains the same, serialized differently or packaged in other file formats
MINOR	Existing data is the same and can be retrieved as is, with new fields, record sets or records added
MAJOR	Existing data was edited, removed or shuffled across splits, or extended so the previous version is hard to reach

Source: [10]. Version 1.0 of 1 March 2024 carries the same rules [11].

Checksums and live data. Each FileObject may carry a checksum in its sha256 property [10]. For a live dataset, isLiveDataset is set to true, files expected to be updated carry no checksum, and applications "filter data by dates if new data is added over time" [10]. In Croissant's stock-price example, changed files for prior years "should trigger a version update" because history was "rewritten", while appending to the current, incomplete year "should not trigger a version update" [10].

Other schemes. The W3C's bus-stop example treats adding a stop and deleting a removed one as a change that "might trigger a major version", and an error correction as one that "would likely trigger only a minor version" [3]. Producers in the CODATA use cases name versions with Version, Collection, Release or Edition numbers, or a vYYYYMMDD date [1]. For real-time data, the W3C accepts "an automated timestamp" as the version identifier [3].

A mapping from operations to levels (derived). Iceberg's append matches Croissant's MINOR, replace matches PATCH, and overwrite and delete match MAJOR [10, 18]. Neither source states this mapping. It is this note's reading of the two definitions.

A worked changelog. FineWeb, a public web-text training dataset, publishes a changelog on its dataset card [12]:

Version	Date	Change
v1.0.0	21 April 2024	Initial version
v1.1.0	31 May 2024	Reprocessed 11 dumps, CC-MAIN-2021-49 to CC-MAIN-2023-40, after a deduplication bug, and added the CC-MAIN-2024-18 dump
v1.2.0	3 January 2025	Added 8 snapshots covering May to December 2024
v1.3.0	31 January 2025	Processed documents some dumps had missed, about 400 billion more tokens, and removed specific domains in response to a cease-and-desist notice
v1.4.0	11 July 2025	Added 6 snapshots covering January to June 2025

Source: [12].

Under Croissant's rules, v1.1.0 and v1.3.0 changed existing data, which Croissant classes as MAJOR, while both were released as MINOR increments (derived) [10, 12]. v1.2.0 and v1.4.0 added data and fit MINOR (derived) [10, 12]. Each entry states what changed, as the W3C's version history practice asks, and the two entries that rewrote data also give the reason [3, 12].

5. Point-in-time correctness and look-ahead bias

Definitions. The FISD Alternative Data Council's note on point-in-time data says "the history described by the data needs to reflect, at each Point-In-Time, what was known and when it was known" [29]. Look-ahead bias is "caused by using data that was not yet known on the date being analyzed leading to inaccurate results" [29]. Survivorship bias occurs "when the universe of available companies only represents the present, skewing model results" [29].

Aggregates that move. The SEC's xbrl/frames API "aggregates one fact for each reporting entity that is last filed that most closely fits the calendrical period requested" [30]. An aggregate built on last-filed facts reflects filings made later, so a study of what was known on a past date needs its own dated copies (derived) [30].

Vintages. ALFRED and the Philadelphia Fed's real-time data set store each release as it was published, so a series can be read as it stood on a past date [16, 17]. In SQL:2011 terms, a query FOR SYSTEM_TIME AS OF the analysis date returns the rows the table held at that time [13]. That answers the point-in-time question only

when system time records when each fact was published or captured (derived) [13, 29].

Survivorship in collected data. A collection that drops withdrawn records keeps only the present universe, the condition FISD names for survivorship bias (derived) [29]. A system-versioned table keeps a deleted row's old state with the end of its period [13], so the record stays available to queries about earlier dates.

6. Withdrawn records and deprecation

Identifiers outlive data. The W3C's Best Practice 27 reads "When removing data from the Web, preserve the identifier and provide information about the archived resource" [3]. For data deleted entirely, "the server should be configured to respond with an HTTP Response code of 410 (Gone)", and for archived data it redirects to a page about the archive that holds the data [3].

Deletion under policy. The RDA's answer to "May data be deleted?" is "Yes, given appropriate policies" [2]. Queries may then no longer re-execute against the original timestamp, only against the current one, and "Landing pages should persist" [2].

What happens without a notice. Peng, Mathur and Narayanan analyzed "nearly 1000 papers" citing three datasets [35]. Two of them were retracted in April 2019, yet "their underlying data remained widely available and continued to be used in research papers" [35]. In the authors' 20% sample, one retracted dataset and its derivatives were used 54 times in papers published in 2020, and the other and its derivatives 73 times [35]. The authors write that "because the dataset creators did not clearly state that the datasets should no longer be used, they may have left users confused" [35].

A framework. Luccioni and colleagues write that "there are currently no standardized approaches" for deprecating datasets, and they identify cases of datasets that "continued to circulate despite having been deprecated" [34]. Their framework includes "risk, mitigation of impact, appeal mechanisms, timeline, post-deprecation protocols, and publication checks", and they propose a central repository for "archiving datasets, tracking dataset modifications or deprecations" [34].

Withdrawals inside a release. FineWeb's v1.3.0 removed specific domains after a notice, in a release that also added data, and its changelog says so [12].

7. Release notes and change records

Version history. The W3C's Best Practice 7 asks publishers to "Assign and indicate a version number or date for each dataset", and Best Practice 8 to "Provide a complete version history that explains the changes made in each version" [3]. Comparing versions is "typically very laborious unless a summary of the differences is provided", and publishers should "avoid making changes without incrementing the version indicator" [3].

Fields for a change record. DCAT 3 offers dcat:version "for the version name / identifier", adms:versionNotes "for a textual description of the changes, including backward compatibility issues with the previous version of the resource", dcterms:issued "for the version release date" and dcat:previousVersion for "a version chain,

consisting of snapshots of a resource" [4]. adms:status records a life-cycle status, and "DCAT does not prescribe the use of any specific set of life-cycle statuses" [4]. In PROV-O, prov:wasRevisionOf links a revision to the entity whose content it carries, as with "two editions of a book" [5]. DataCite lets a Description indicate "the nature and file/record range of version" [6].

Machine-readable change records. For data "dynamically being constantly updated, as in a time series", the CODATA authors ask providers for "machine readable records of where they have made any changes to the dataset", including hardware and software changes that "could also affect the data" [1]. Iceberg's snapshot summaries and Delta Lake's DESCRIBE HISTORY record each write by machine [18, 22], and Delta keeps that history for 30 days [22]. A change record meant to last longer than that keeps its own copy (derived) [22].

Change record field	What it holds	Basis
Version identifier	The version name or number, or a timestamp for real-time data	[3, 4]
Release date	When the version was issued	[4]
Previous version	The version it revises	[4, 5]
What changed	The nature and range of the change, including compatibility issues	[1, 4, 6]
Why	The significance of the change and its reason, such as a fixed bug or a removal notice	[1, 12]
Implications for use	What a user of the earlier version needs to know	[1]
Checksums	A checksum for each file, or for a query's result set	[2, 10]
Status	The life-cycle status, such as archived or deleted	[3, 4]

Implications

For data engineers. A versioning policy for data whose sources change:

1. Publishes what counts as a revision, a release and each version level before the first release [1, 3, 4].
2. Gives every revision an identifier and every release a change record with what changed, why it matters and the implications for use [1, 3].
3. Keeps valid time and system time on each record and states the period convention [13, 14, 15].
4. Keeps withdrawn and replaced records with the end of their period, so point-in-time queries still find them [13, 29].
5. Pins each version a result cites with a tag, a checksum or a stored query that outlives the time travel window [2, 19, 20, 22].
6. Sets version levels with written rules such as Croissant's, and checks each proposed level against the operation that made the change (derived mapping) [10, 18].
7. Keeps the identifiers of removed data with a notice of what happened to it [3, 35].
8. Checks retention settings against the longest period over which a result has to be rebuilt (derived) [19, 22, 23, 24].

For machine learning engineers and researchers. Record the dataset version, the checksum or digest and the query or revision used with every training run and evaluation [2, 10, 27, 28]. For a backtest or an evaluation on past dates, read the data as of those dates and include records that were later withdrawn [13, 29]. For a live dataset, filter by date, as Croissant asks [10].

For buyers of data. Ask for the versioning policy, the version history, the period convention, how withdrawals are announced and how long earlier versions stay retrievable, the points the maintenance questions of Datasheets for Datasets also cover [1, 3, 8, 34].

Rillor's data collection. Rillor's data collection for model training keeps each record's capture time, keeps a source's change as a new version and delivers the dataset with its capture log, its source list with terms and its version history. Each version comes with a change log and a datasheet that records sources, rights, lineage, quality results and stated limits.

Limits of this analysis

- This note describes versioning rules from standards, specifications and documentation. It ranks no tool or format and names none as Rillor's.
- Tool defaults are as documented on 8 November 2026 and can change after that date [19, 22, 23, 24].
- The SQL:2011 standard itself was not opened. Its temporal features are described through Kulkarni and Michels [13], and this note says nothing about which databases implement them.
- No source reviewed gives how often training datasets in general change, or what share of them publish a versioning policy. The examples of change come from economic statistics, web pages, web terms and one dataset changelog [12, 31, 32, 33].
- The mapping between Iceberg operations and Croissant levels, and the reading of FineWeb's changelog against Croissant's rules, are this note's own. Neither source states them [10, 12, 18].
- The retraction evidence comes from one study of papers citing three datasets [35].

Sources

1. Versioning Data Is About More than Revisions: A Conceptual Framework and Proposed Principles. Jens Klump, Lesley Wyborn, Mingfang Wu and others, Data Science Journal 20(1), Article 12, Ubiquity Press for CODATA, 23 March 2021, viewed 8 November 2026. https://datascience.codata.org/articles/10.5334/dsj-2021-012

2. Data Citation of Evolving Data: Recommendations of the Working Group on Data Citation (WGDC). Andreas Rauber, Ari Asmi, Dieter van Uytvanck and Stefan Pröll, Research Data Alliance, 20 October 2015, viewed 8 November 2026. https://doi.org/10.15497/RDA00016

3. Data on the Web Best Practices. W3C Recommendation, W3C, 31 January 2017, viewed 8 November 2026. https://www.w3.org/TR/dwbp/

4. Data Catalog Vocabulary (DCAT), Version 3. W3C Recommendation, W3C, 22 August 2024, viewed 8 November 2026. https://www.w3.org/TR/vocab-dcat-3/

5. PROV-O: The PROV Ontology. W3C Recommendation, W3C, 30 April 2013, viewed 8 November 2026. https://www.w3.org/TR/prov-o/

6. DataCite Metadata Schema 4.5, property 15: Version. DataCite, undated, viewed 8 November 2026. https://datacite-metadata-schema.readthedocs.io/en/4.5/properties/version/

7. Manage versions. Zenodo help, CERN, undated, viewed 8 November 2026. https://help.zenodo.org/docs/deposit/manage-versions/

8. Datasheets for Datasets. Timnit Gebru, Jamie Morgenstern, Briana Vecchione and others, Communications of the ACM, December 2021, arXiv 1803.09010, version 8, 1 December 2021, viewed 8 November 2026. https://arxiv.org/abs/1803.09010

9. Semantic Versioning 2.0.0. semver.org, originally authored by Tom Preston-Werner, undated, viewed 8 November 2026. https://semver.org/

10. Croissant Format Specification, Version 1.1. MLCommons, 29 January 2026, viewed 8 November 2026. https://docs.mlcommons.org/croissant/docs/croissant-spec-1.1.html

11. Croissant Format Specification, Version 1.0. MLCommons, 1 March 2024, viewed 8 November 2026. https://docs.mlcommons.org/croissant/docs/croissant-spec.html

12. FineWeb dataset card, Changelog. HuggingFaceFW on Hugging Face, undated, viewed 8 November 2026. https://huggingface.co/datasets/HuggingFaceFW/fineweb

13. Temporal features in SQL:2011. Krishna Kulkarni and Jan-Eike Michels, SIGMOD Record 41(3), September 2012, viewed 8 November 2026. https://sigmodrecord.org/publications/sigmodRecord/1209/pdfs/07.industry.kulkarni.pdf

14. Temporal Data Management. Christian S. Jensen and Richard T. Snodgrass, IEEE Transactions on Knowledge and Data Engineering 11(1), January/February 1999, viewed 8 November 2026.

https://www2.cs.arizona.edu/~rts/pubs/TKDEJan99.pdf

15. FRED API: Real-Time Periods. Federal Reserve Bank of St. Louis, undated, viewed 8 November 2026. https://fred.stlouisfed.org/docs/api/fred/realtime_period.html

16. ALFRED: Archival Federal Reserve Economic Data. Federal Reserve Bank of St. Louis, undated, viewed 8 November 2026. https://alfred.stlouisfed.org/

17. Real-Time Data Set for Macroeconomists. Federal Reserve Bank of Philadelphia, undated, viewed 8 November 2026. https://www.philadelphiafed.org/surveys-and-data/real-time-data-research/real-time-data-set-for-macroeconomists

18. Iceberg Table Spec. Apache Iceberg, Apache Software Foundation, undated, viewed 8 November 2026. https://iceberg.apache.org/spec/

19. Configuration: table properties. Apache Iceberg documentation, version 1.12.0, Apache Software Foundation, undated, viewed 8 November 2026. https://iceberg.apache.org/docs/latest/configuration/

20. Branching and Tagging. Apache Iceberg documentation, version 1.12.0, Apache Software Foundation, undated, viewed 8 November 2026. https://iceberg.apache.org/docs/latest/branching/

21. Maintenance. Apache Iceberg documentation, version 1.12.0, Apache Software Foundation, undated, viewed 8 November 2026. https://iceberg.apache.org/docs/latest/maintenance/

22. Table batch reads and writes. Delta Lake documentation, version 4.3.0, undated, viewed 8 November 2026. https://docs.delta.io/latest/delta-batch.html

23. Understanding and using Time Travel. Snowflake documentation, Snowflake, undated, viewed 8 November 2026. https://docs.snowflake.com/en/user-guide/data-time-travel

24. Data retention with time travel and fail-safe. BigQuery documentation, Google Cloud, last updated 27 October 2026, viewed 8 November 2026. https://cloud.google.com/bigquery/docs/time-travel

25. Git Internals: Git Objects. Pro Git, second edition, Scott Chacon and Ben Straub, git-scm.com, undated, viewed 8 November 2026. https://git-scm.com/book/en/v2/Git-Internals-Git-Objects

26. .dvc files. DVC documentation, undated, viewed 8 November 2026. https://doc.dvc.org/user-guide/project-structure/dvc-files

27. MLflow Dataset Tracking. MLflow documentation, undated, viewed 8 November 2026. https://mlflow.org/docs/latest/ml/dataset/

28. Load. Hugging Face Datasets documentation, Hugging Face, undated, viewed 8 November 2026. https://huggingface.co/docs/datasets/en/loading

29. Learning Point: Point in Time. FISD Alternative Data Council, SIIA, June 2021, viewed 8 November 2026. https://fisd.net/wp-content/uploads/2021/06/Learning-Point-Point-in-Time.pdf

30. EDGAR Application Programming Interfaces. U.S. Securities and Exchange Commission, last reviewed or updated 8 April 2025, viewed 8 November 2026. https://www.sec.gov/search-filings/edgar-application-programming-interfaces

31. CES National Benchmark Article: BLS Establishment Survey National Estimates Revised to Incorporate March 2025 Benchmarks. U.S. Bureau of Labor Statistics, 11 February 2026, viewed 8 November 2026. https://www.bls.gov/web/empsit/cesbmart.htm

32. When Online Content Disappears. Athena Chapekis, Samuel Bestvater, Emma Remy and Gonzalo Rivero, Pew Research Center, 17 May 2024, viewed 8 November 2026. https://www.pewresearch.org/data-labs/2024/05/17/when-online-content-disappears/

33. Consent in Crisis: The Rapid Decline of the AI Data Commons. Shayne Longpre and others, arXiv 2407.14933, version 2, 24 July 2024, viewed 8 November 2026. https://arxiv.org/abs/2407.14933

34. A Framework for Deprecating Datasets: Standardizing Documentation, Identification, and Communication. Alexandra Sasha Luccioni, Frances Corry, Hamsini Sridharan and others, ACM FAccT 2022, arXiv 2111.04424, version 2, 9 May 2022, viewed 8 November 2026. https://arxiv.org/abs/2111.04424

35. Mitigating Dataset Harms Requires Stewardship: Lessons from 1000 Papers. Kenny Peng, Arunesh Mathur and Arvind Narayanan, NeurIPS 2021 Datasets and Benchmarks Track, arXiv 2108.02922, version 2, 21 November 2021, viewed 8 November 2026. https://arxiv.org/abs/2108.02922

About this report

Published by Rillor on 9 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Versioning a dataset when its sources change. 9 November 2026. https://rillor.com/insights/versioning-datasets-when-sources-change

Work with Rillor

Send the question the data should answer, the sources you know of, the coverage you need and how often it should update. Contact us at rillor.com/contact?type=custom-data or contact@rillor.com. More on data services: rillor.com/data