The challenges of reproducibility for research based on geodata web services

Massimiliano Cannata[1], Maxime Collombin[2], Olivier Ertz[2], Gregory Giuliani[3], Jens Ingensand[2], Claudio Primerano[1], Daniele Strigaro[1]

[1] University of Applied Sciences Southern Switzerland (SUPSI), Switzerland, (massimiliano.cannata, claudio.primerano, daniele.strigaro)@supsi.ch
[2] School of Engineering and Management Vaud, HES-SO, University of Applied Sciences and Arts Western Switzerland, (maxime.collombin, olivier.ertz, jens.ingensand)@heig-vd.ch
[3] University of Geneva, Geneva, Switzerland, gregory.giuliani@unepgrid.ch

Keywords: FAIR; geospatial web services; interoperability; open science; reproducibility; transparency.

Abstract

Modern research applies the Open Science approach that fosters the production and sharing of Open Data according to the FAIR (Findable, Accessible, Interoperable, Reusable) principles. In the geospatial context this is generally achieved through the setup of OGC Web services that implements open standards that satisfies the FAIR requirements. Nevertheless, the requirement of Findability is not fully satisfied by those services since there's no use of persistent identifiers and no guarantee that the same dataset used for a study can be immutably accessed in a later period: a fact that hinders the replicability of research. This is particularly true in recent years where data-driven research and technological advances have boosted frequent updates of datasets. Here, we review needs and practices, supported by some real case examples, on frequent data or metadata updates in geo-datasets of different data types. Additionally, we assess the currently available tools that support data versioning for databases, files and log-structured tables. Finally, we discuss challenges and opportunities to enable geospatial web services that are fully FAIR: a fact that would provide, due to the massive use and increasing availability of geospatial data, a great push toward open science compliance with ultimately impacts on the science transparency and credibility.

1. Introduction

This article discusses the reproducibility of research that have been based on datasets offered by interoperable open geospatial Web services and that are subject of frequent modifications. We explore the current context and a few cases of frequent data changes of different geospatial data types and discuss some available technological solutions to support data versioning. The emphasis of this paper is on challenges and needs of practical solutions.

1.1 Open Science, Research Data, and Geospatial Services

Open Science represents a transformative approach to research, enabled by digital technologies and a collaborative ethos. It promotes the open sharing of data, information, and knowledge across the scientific community and society at large to accelerate discovery and understanding (Ramachandran, 2021). The core objectives of Open Science are to improve access to knowledge, enhance the efficiency of sharing research outputs, and evolve impact evaluation through new metrics.

At the heart of this paradigm is Open Research Data (ORD), which refers to data underlying scientific results that are freely accessible and reusable. ORD reduces duplication, supports interdisciplinary collaboration, and fosters innovation. To be effectively reused, data must comply with the FAIR principles (Findable, Accessible, Interoperable, Reusable), which has driven the creation of repositories like Zenodo, PANGAEA, ARCHE, and the UK Data Archive, indexed by platforms such as re3data. These services typically support persistent identifiers (DOIs), open licensing, standard metadata, and open file formats (e.g., CSV, JSON).

However, most repositories handle static datasets. To support advanced applications such as machine learning or big data analytics, there's a growing need for Analysis Ready Data (ARD) and platforms capable of regular data delivery and preprocessing. This demand aligns with the European Data Spaces initiative, which envisions secure and interoperable environments for data sharing across domains; integrating legal, ethical, and technical safeguards to drive innovation and digital transformation.

In the geospatial domain, data sharing has been facilitated through Spatial Data Infrastructures (SDIs) at multiple governance levels. These infrastructures are built on interoperable standards promoted by the Open Geospatial Consortium (OGC), which enables thousands of applications to access and integrate geospatial content globally. Traditionally, this was done through OGC Web Services (e.g., WMS, WFS, WCS), using XML and SOAP protocols.

In recent years, a new generation of standards, OGC APIs, has emerged, reflecting modern web development practices. Developed in collaboration with the World Wide Web Consortium (W3C), OGC APIs are RESTful, use JSON (or JSON-LD for linked data), support OpenAPI documentation, and are designed as modular "Building Blocks" to be flexibly combined. These standards simplify integration with non-geospatial systems, enhance search engine indexing, and better support Web-native data sharing—making geospatial data more accessible and actionable within broader open science and data space ecosystems.

Together, the evolution of ORD and interoperable geospatial services illustrates a shift toward open, machine-actionable, and integrated digital research environments—enabling new forms of collaboration, insight, and impact.

1.2 Time Varying Data

The technological growth in the last decades led to the explosive increment of time-varying data which dynamically change to represent phenomena that grows, persists and decline (Wang *et al.*, 2008), or that constantly vary due to data curation processes that periodically insert, update, or delete information related to data and metadata. As discussed by Saracco (2010), dealing with dynamic data has consistently played a pivotal role in facilitating a wide range of analyses, such as examining changes in client accounts for financial institution audits, assessing the clinical progression of patients for legal proceedings, evaluating insurance policy terms at the time of accidents, identifying disparities in travel itineraries involving car and hotel reservations, and adjusting interest rates for banks upon detecting errors. In addressing such cases Saracco identifies two different concepts of time: system-time and business-time. While business-time can be defined as the instant/period for which the data is meant to be used (often referred to as "valid-time" or "application-time"), system-time relates to the data state as stored in a specific instant/period (often also referred to as "transaction-time"). So, for example, in the case of an application tracking the position of a delivery track and the corresponding air temperature the system would record the coordinates and the air degrees Celsius. After a month, if the track undergoes a revision in which the temperature sensor was found to have a bias of 0.002 degrees and the raw GPS data were manually collected and reprocessed for improved position accuracy, data would be updated accordingly. Answering the questions "where was the track on August 30 at 11:30? What is the average temperature in May on London Street? What is the distance covered by the track in 2023?" requires the usage of the business-time. Answering questions "Which data did we use to compile the delivery report on August 25 at 17:15? How did the position of the track change after GPS data processing?" requires the usage of system-time. From a logical perspective business-time is characterized to be maintained by the user, future dates and times may make complete sense, its resolution (day, month, microseconds, etc.) is decided by the application. On the contrary, system-time is managed automatically, future dates are not permitted, and its resolution must be the finest possible.

2. Reproducibility in the Geospatial Sector

Based on the current trends and data availability, the ability to link Open Science concepts with interoperability and time-varying data management is paramount. In particular, the capability of obtaining results consistent with a prior study using the same materials, procedures, and conditions of analyses is very important since it increases scientific transparency, fosters a better understanding of the study, produces an increased impact of the research and ultimately reinforces the credibility of science (Konkol and Kray, 2019, Kedron *et al.*, 2021). In the Open Science paradigm this is indicated as Reproducible Research, and it can be guaranteed only if the same source code, dataset, and configuration used in the study is available. For geospatial data, while the presented OGC standards enable an almost FAIR (Giuliani *et al.*, 2021) and modern data sharing, they do not adequately support the reproducibility concept as pursued in Open Science. In other words, they do not offer any guarantee that the geodata accessed in a given instant can be persistently accessed, immutably, in the future. In fact, while business-time is considered in several aspects and standards like Part 3 from OGC API - Features including temporal filtering, Sensor Observation Service or OGC API Moving Features, at the best knowledge of the authors, none of them support the system-time. Releasing different dataset versions (e.g. storing exported datasets in FAIR repositories or offering a layer referencing a fixed time instant) can be a solution. Nevertheless, in many cases, where data (and metadata) are frequently updated and datasets have large size, this is not efficient nor applicable. Additionally, this approach may hinder the seamless capacity of analyses of data variations.
This is confirmed by Nüst and Pebesma (2021) that produced a comprehensive summary of the state of the art in reproducible research within the geospatial domain. In the manuscript they recognize that only a small body of work on reproducibility in the geospatial domain was available. They underlined that reproducibility might be achieved only when physical, logical and cultural components are available and identified that the main challenge is the general poor knowledge of reproducibility practices by researchers. Other barriers that they highlighted were related to: *(1)* the utilization of proprietary software which is often subject to licensing restrictions prevents reproduction; *(2)* the multitude of tools frequently employed in a single geospatial research project poses a challenge to replicability; *(3)* the reliance on geospatial infrastructure that depends on online services can lead to obstacles in accessing the original dataset due to potential changes; *(4)* analysing extensive datasets is often executed in proximity to the data source using online services, which would necessitate open accessibility to the server implementations; *(5)* while free platforms provide scripting capabilities for data processing, the environment may change and not ensure reproducibility.
Cerutti et al. (2021) in their study, which replicated and compared three studies conducted on disaster response using different geospatial algorithms, proposed the use of an analytics platform (KNIME, no date) which includes geospatial functions to create scientific workflows and enhance reproducibility. Similarly, registering the analysis workflow in computational notebooks enables the ordered re-execution of processing steps: this approach has been adopted for example by the GRASS GIS community (GRASS GIS Jupyter notebooks) and ESRI (ArcGIS notebooks). Nevertheless, the reproducibility of workflows does not guarantee that the code is executed using the same data and environment used during the study. For this reason, Yin et al. (2019) presented a cloud-based solution, named CyberGIS-Jupyter, that combines Jupyter notebook with docker technology to support computational reproducibility and scalability of geospatial analyses. While Jupyter notebook guarantees the reproducibility of the workflow, the docker technology permits to reproduce the environment, with exactly the same software and libraries versions, where the geospatial processes were executed. Kedron et al. (2021) highlighted in his review that even if the computational reproducibility is guaranteed, it does not include itself two essential components for the full reproducibility of research: the record of task coordination and of conceptual decision-making. For this reason, research notebooks and software like the Open Science Framework have been used to capture research provenance (steps and decision criteria) in addition to processing workflow including approaches and pre-analysis plans. The reported approaches present solutions that contribute to solving several reproducibility challenges, nevertheless they do not address the issue of accessing the original dataset when online services from a geospatial infrastructure are at the base of a study.
With the aim of understanding how system-time, and reproducibility as a consequence, could be addressed and managed in geospatial data services, this work aims at identifying the requirements, challenges and opportunities by reviewing:

1. how the geospatial domain is addressing reproducibility,

2. how updates of time varying spatial data happens in real case applications, and
3. which technological solutions exists to manage system-time for big-data spatio-temporal datasets.

It may be worth mentioning that the OGC issued a Call for participation for the Open Science Persistent Demonstrator (OSPD) Initiative. Although important for the replicability of science, this initiative focuses on platforms and workflows to demonstrate how a federation of OGC based services can offer FAIR Open Data in support of Open Research making use of the PROV-O (Groth and Moreau, 2013) provenance data model. In fact, while PROV-O provides an effective ontology to register and replicate processing chains to the best knowledge of the writers, there is currently no interoperable standard application that supports persistent retrieval of time-varying data from dynamic web services. In practice, PROV-O serves as a descriptive framework for provenance metadata, but it does not provide queryable interfaces or retrieval mechanisms for historical data states. The remainder of the paper presents the results of the literature review for each of these three aspects and finally presents a discussion highlighting possible solutions and approaches to be investigated in the future.

3. Updates And Changes of Time Varying Geospatial Data

In this section we evaluated the needs and practices supported by some real case examples related to common operations that update data or metadata of the different geospatial data types, specifically sensor observations, vector datasets and raster series.

3.1 Updates and Changes of Sensor Observations

In the geospatial domain, sensor observations are mainly addressed by two standard web services specifications from the OGC: the Sensor Observation Service (SOS) and the Sensor Things API (STA). While SOS is based on the SOAP and data are encoded in XML the STA is based on the use of RESTful services following the OData's specification (Kirchhoff and Geihs, 2013) and data are encoded in JSON format. Both specifications offer access to sensor data and metadata along with transactional capabilities. Apart from some specific differences in the data model (Blanc *et al.*, 2022) and requests the two standards can be considered conceptually comparable.
Based on the previously reported challenges of reproducibility in the geospatial sector these standards are exposed to potential changes of datasets in time, in fact an authorized user can change data using the offered transactional features. Best practices followed to serve ARD from sensor networks very often include post-processing (after the original acquisition from the sensor) to reduce uncertainties in further analyses which produce knowledge and wisdom. As discussed by Krishnamurthi et al. (2020) these processing include: (i) denoising to eliminate most of the noise signals in data, (ii) missing data imputation to deal with incomplete data that are not supported by several analyses techniques (e.g. ML models), (iii) data outlier detection to identify data which has been incorrectly sensed due to external unpredictable factors, (iv) data aggregation of heterogeneous observations (difference in time and property) to reduce data transmission size and complexity. Additionally, after the processing phase data may require (v) data fusion which integrates multiple data sources to improve accuracy. At this point data are available for analyses.
Strigaro et al. (2022) described a system collecting high frequency data of ecological and physical parameters from buoys that are located in lakes with the scope of monitoring and assessing lake ecosystem quality using WMO guidelines on quality control procedures (Zahumenskỳ, 2004). Reported cases of time series data management shows that data preprocessing and quality assurance procedures modify data and metadata values as a consequence of real-time analyses or post-processing elaborations or lately manual correction. It is particularly interesting that the WMO in its climate data management system specification includes as a required policy the "ability to reproduce specific data that were held in the climate database at a particular point in time". To better understand the entity of these changes in a real case study we have analyzed the transactional operations that have been executed on the hydro-met monitoring system in the Canton Ticino (2020) managed using the istSOS software (Cannata *et al.*, 2019). The datasets, at the time of writing, include observations ranging from the 1978 to the 2023 and related to 298 sensors observing different properties related to rivers and meteorology. From 2015, year of the activation of a transactional log features offered by istSOS and that permits to register transactional operations executed on the service, we can note that out of 117Mio observations there have been 15Mio updates of single observation data or metadata, value, which, not considering multiple updates for the same values, correspond to about the 13% of the data. Percentage that reaches 23%, if we consider only the measures related to river height and precipitation, which are the measures that actually undergo a systematic quality control process (6Mio out of 26Mio observations). These percentages highlight how important it could be to access specific dataset status in time. Additionally, since every year an annual hydrobook is produced in the following year, data undergo a specific process of re-analyses where, for example, stage-discharge relationship curves are updated or data gap filled, and the entire previous year aggregated data is re-calculated. As an example, in October 2023 hydrologists requested to re-calculates the daily aggregated data for all the discharge stations for the entire 2022, for a specific station for the entire 2021 and for another station for one week due to configuration changes.

3.2 Updates and Changes of Features

Vector layers which vary in time due to continuous data additions and updates are generally managed with database or file sources and are very often offered by means of standard OGC WMS or WFS and more recently via the OGC API Feature. As an example of geospatial datasets undergoing a continuous update, we can cite the cadastral data and the water protection zones. They are continuously updated to reflect changes that occur on land ownership or water policies. In particular, the groundwater protection zones identify different areas centered around groundwater collections or recharge plants. When new wells are established and/or new hydrogeological studies are conducted new protection zones are inserted or existing ones are edited and published as WMS on the official portal on the cadastre of public-law restrictions on land ownership of Canton Ticino (https://crdpp.geo.ti.ch/). Here the current situation is duly represented but there is no option to navigate and check how restrictions were represented in the past, which may be very relevant in case of disputes. Similarly, the cadastre of Canton Ticino is publicly available as WFS service (https://wfs.geo.ti.ch). This is the current version and while we can find in the attributes the date a feature entered in law, it does not offer any option to navigate in time the evolution of properties map.
A very popular geospatial vector dataset that is continuously updated is the OpenStreetMap (OSM). It is a crowdsourced geographic dataset with more than 10 million registered users.

OSM maintains timestamped historic changesets that are a group of modifications set by a single user in a short period. It is possible to access weekly snapshots and changesets of the full dataset or the full history by downloading them in XML or Planet PBF files (https://planet.openstreetmap.org/). To analyze the creation process of OSM data, some software were implemented by the are limited to examine and view a small portion of the database (Mocnik *et al.*, 2018). Martini et al. (2019) presented a methodology for analysis of changes in OSM. In this paper the authors analyzed the changed objects in the city area of Karlsruhe, Germany. The produced maps show areas with up to 2 thousand new objects per year and up to 13 thousand objects updates per year. These numbers and the full history file with its size of more than 200 Gb provide an order of magnitude of the high variability of the data in time due to new features, deletion or modification of either geometric or semantic information.

3.3 Updates and Changes of Rasters

Concerning raster data, the principal source of data is represented by satellite Earth Observations (EO) data. Indeed planet Earth is under continuous observation from many different types of satellites produce a continuous and increasing stream of observations from space. Among the different benefits of using satellite imagery for environmental monitoring, Merodio Gómez et al. (2019) listed (1) Temporal resolution: capacity to capture data at different frequency of revisit; and (2) Time-series: capacity to provide continuous data starting as early as 1972 (e.g., Landsat) as two important aspects.

To tackle the big data challenges related to EO data handling, the emergence of EO Data Cubes allowed to efficiently and effectively manage and analyze large amounts of EO data (Giuliani *et al.*, 2017, Lewis *et al.*, 2017), enabling spatio-temporal analysis of Analysis Ready Data (ARD) (Dwyer *et al.*, 2018). However, interoperability of Data Cubes is still a challenge, different existing and emerging standards can help deliver and leverage the power of EO data building, efficient discovery, access and processing services (Giuliani *et al.*, 2019). OGC WMS and WCS are common standards that already have demonstrated their ability to handle satellites for visualization and download purposes. While both standards have a time parameter that can be used to extract a specific time-slice, it remains limited to that single operation. It cannot do operations on a time interval or nearest values. Another restriction, related to the semantic of the time parameter, is ambiguous that could refer to acquisition - processing or publication time. The OGC WMS Earth Observation profile recommends using the time parameter only for the acquisition time (Lankester, 2009). Giuliani et al. (2019) discuss and demonstrate potential ways to properly handle the time dimension on existing OGC standards.

Among the new standards that have emerged in recent years, the Spatio-Temporal Assets Catalog (STAC) (STAC Contributors, 2024) provides a common structure for describing and cataloguing spatio-temporal assets (Ferreira *et al.*, 2020). It tackles most of the issues previously mentioned and facilitates the creation of flexible spatio-temporal analysis workflows, removing the burden of creating specific pipelines for each different data collection one consumes. Nevertheless, existing and emerging standards are not properly handling backward compatibility of raster-based products (e.g., guarantee that I can access the data as they were yesterday... not like they are today). Indeed, for example in data cubes, if one reprocessed a given product (e.g snow cover) or ARD satellite imagery, with an improved version of an algorithm, then the only solution for versioning them is to create a new data collection with a different version number that can then be queried/accessed through an OGC-compliant API or Web service.

4. Tools for Data Versioning

Traditional data management systems typically store only the current state of data, making changes irreversible as datasets evolve through acquisition (new features) or processing. However, reproducible research requires the ability to recreate the exact environment and dataset state used in a study. This makes data versioning essential, enabling access to historical versions of datasets. Many big data applications, particularly in Machine Learning (ML), benefit from tracking historical data changes, as models are continuously retrained and require traceability of both parameters and inputs over time. As data production grows, the shift from traditional databases to object storage and Data Lakes (Mathis, 2017) is accelerating. Consequently, most modern versioning tools focus on object storage (dataset-level) rather than database-level version control. Several open and collaborative solutions—often inspired by Git—now support dataset versioning by tracking file changes over time and enabling rollback. These tools facilitate transparency, collaboration, and reproducibility, aligning with Open Science values. The following sections briefly describe key open-source tools supporting different versioning approaches. Due to limited formal literature, references often include software documentation, websites, blogs, and community forums.

4.1 Data Versioning of Databases

Tracking the historical evolution of records or database version control mechanisms are generally based on the definition of a Slowly Changing Dimension (SCD) (Kimball, 2008) which is a dimension that registers, and permits to manage, the evolution in time of values in a database table. Three types of SDCs exist: (i) type 1 stores the latest valid values of a record and that is the standard database rule, (ii) type 2 stores all the versions of the record registering the period for which that value was active, and (iii) type 3 stores the current and previous values only of a record. In December 2011, ISO/IEC published an updated version of the SQL standard, SQL:2011 (Kulkarni and Michels, 2012) which introduced the capability of managing temporal tables. This includes the support of time period data type which can be declared as primary or foreign key and support a number of filtering operations (i.e. overlaps, equals, contains, precedes, succeeds, immediately precedes, immediately succeeds). A period column can have any name except SYS-TEM_TIME, which is a reserved name to enable system-time features as SDC type 2. Several databases implemented the SQL:2011 specification as described by Jungwirth (2019). In addition to relational databases, Soroush and Balazinska (2013) presented a methodology for extending column stores (array databases) with versioning.

MariaDB - In MariaDB you can enable the system-time versioning of a table using the syntax “WITH SYSTEM VERSIONING” which adds the “ROW_START” and “ROW_TO” pseudo-columns that do not appear in SELECT statements and are populated automatically. The ROW_START is populated with the insertion timestamp while ROW_END with an instant far in the future if the record is valid or the instant the row has been updated/deleted. Filtering using the system-time can be performed using the “AS OF” to extract a specific version of the data in a specific instant or the “FROM … TO” or “BETWEEN” statements to extract the record as they were valid in a provided period. Versioned tables with system-time can be partitioned so that historical rows and current valid

rows are separated, optionally users can set the historical partition to be partitioned every *n* records (MariaDB community, no date).

IBM DB2 - System-time is implemented similarly to MariaDB with a few syntax differences, like naming "ROW BEGIN/END" instead of START/TO. System-time pseudo-columns columns are not accessible in SELECT statements and filtering does not support BETWEEN statements [59].

Oracle - Oracle has its own implementation of historical values that do not comply with SQL:2011. It allows you to declare a PERIOD but not as a primary key and periods can have null values. Oracle Database supports Flashback Time Travel feature implementing SDC type 2 that can be configured with a retention time on a tablespace or identified tables making it easy to undo or query historical stored values (Deshpande, 2004, Gregg, 2023).

MS SQL Server - System-time tables are supported but with non fully standard syntax "WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.this))" where historical records are saved in an invisible table that can be named and queried as any normal table. Filtering supports options like MariaDB (rwestMSFT, 2023).

PostgreSQL - Postgresql does not support system-time. There are a few projects implementing such a feature but they are not official extensions and are under development. A few GitHub repositories implement a solution in Pl/PgSQL (Chiodi, 2023, Fearing, 2023). An ex-tension dated 2018 is proposed on the PostgreSQL Extension Network website. None are officially supported by postgreSQL.

OrpheusDB - It is a layer installed on top of relational databases and expose git-like command (Huang *et al.*, 2020). It stores data in tables following the SDC type 2. In OrpheusDB records are immutable and are archived in Collaborative Versioned Dataset (CVD) recording record id (rid) and version id (vid) with other associated metadata like creation time, commit time, committer and a commit message. OrpheusDB implements a CLI (Command Line Interface) with, among other, checkout and commit commands. SQL can be performed on a version with the run command that takes the SQL as an input, translates and executes it against the database for a specific version. In his paper Huang et al. (2020) demonstrated the solution on postgreSQL and presented a graphical interface to navigate through the version tree.

Dolt - While not found in scientific literature, Dolt is a MySQL git-like database versioning system. According to its documentation Dolt treats tables as files and registers modifications in a stage area (so called "working set") which is the current database version used when queries are executed. When a Dolt commit is performed a new version is persisted so that differences between versions, and metadata can be explored. It is possible to configure Dolt so that at each SQL commit a dolt commit is executed but according to Sehn (2022) in this case you lose the capability of annotating commits with messages and the complexity of the commit graph get hard to be used. Dolt supports branches, diffs and merges.

4.2 Data Versioning of Files

When data are not structured in relational databases but managed in files, changes can be recorded using Git. Unfortunately, Git has not been designed to manage large datasets, in fact it extracts the list of changes (diffs) from stored file snapshots, a fact that limits its performance (Low *et al.*, 2023). For this reason, some solutions have been implemented to overcome this issue and extend Git to support large files.

Git Large File Storage (Git LFS) - Kandpal (2023) proposed a tool for collaborative development of machine learning models, based on the Git LFS and described functioning and limitations. Its main feature is that it has been designed as a Git extension that permits tracking large binary files seamlessly in Git. It works similarly to ordinary Git solutions allowing users to add, commit, push, fetch and checkout file modifications, but instead of storing binary files in Git it replaces them with a text pointer to an external resource that hosts the actual file. When a file is tracked it is managed as a single object thus any modification of the file creates a new copy of the entire object in the storage. For this reason, its drawback is that storage size is proportional to the commits regardless of the size of the modification. Also, it is not possible to get meaningful diffs between versions but only get acknowledged that files are, or not, bitwise identical.

Data Version Control (DVC) - As discussed by Peuster et al. (2019) Data Version Control (DVC) is a software specifically implemented to facilitate management of Machine Learning models and data in Git fashion, using external storages to store binary files and Git as a reference. According to the DVC online documentation (https://dvc.org/doc/user-guide) DVC differs from Git-LFS mainly because it doesn't require specific servers but can use any cloud storage solution. Not much can be found on the mechanism for data versioning on the project documentation, but thanks to an answer from the co-founder of DVC on stack overflow (Shcheklein, 2020) we know that files are tracked as single objects and replicated in case of any part modification.

Lake FS - It is yet another version control system based on the Git approach that allows managing files stored in cloud storages. Like DVC its primary objective is to record Machine Learning models with its associated training dataset. Lake FS permits to branch, commit and merge data which could scale to petabytes allowing to manage data across different cloud storages. It can also revert changes in data. According to Park et al. (2008) it has been created specifically to improve performance on scalable systems.

Pachyderm - It is an open-source platform for managing data pipelines and the associated input/output data. It manages data versioning and lineage by using a combination of technologies: it leverages Git to manage version control using distributed file systems like Hadoop or S3 to store large datasets in addition to databases or key-value stores to record information on how data is generated, transformed, and consumed within the system. When data are committed file hash is produced and file recorded in data storage. When changes are committed it records the variations between the previous version and the new version so that any particular state of data can be then identified by commits (Novella *et al.*, 2019). The usage of docker allows to encapsulate processing and create portable, re-producible data pipelines.

Kart (https://docs.kartproject.org) - Kart is a distributed version-control built on Git specifically implemented for handling geospatial and tabular data. No scientific papers could be found on the software, but according to its documentation it supports different geospatial data types including raster, point cloud and vector datasets. In case of rasters or point-could due to the size of the data Kart uses Git LFS. Specific datasets are stored using defined data formats and folder structures in git, so for example rasters are stored as GeoTIFF in the folder .raster-dataset.v1 and point clouds are stored as LAZ files in the folder .point-cloud-dataset.v1 (using Git LFS), both have a nested structure with two folders: meta for the metadata and title with the actual data (stored in Git). Similarly, for vector and table data type Kart uses a .table-dataset folder with meta and feature subfolders storing all the information in Git. Vectors/tables data are stored as a single file per feature/row therefore modifications are versioned at row levels which permit, by using the metadata, to reconstruct a dataset at a specific commit.

4.3 Data versioning of Log-Structured Tables

Modern columnar data formats like Apache Parquet (Vohra and Vohra, 2016) and ORC (Liu *et al.*, 2023) due to their characteristics of being optimized for storage and retrieval, they became very popular (Cloud Native formats). Nevertheless, their characteristics of being immutable pose a limitation in their adoption in all the cases where frequent updates are required. To overcome this issue, while keeping the benefit of those formats, the Log-Structured Tables (LSTs) solution has been implemented. It adds on top of the immutable columnar data formats a metadata layer that records the versioning of tables and parameters to enable the interaction through the processing engine (Camacho-Rodríguez *et al.*, 2023). This solution is shifting the paradigm of data storage due to its capability of offering ACID transactions and supporting frequent table modification by creating a new immutable columnar file containing the changes. Additionally, it makes use of distributed cloud storage systems, and therefore with respect to traditional data warehouses, is simple and fast to scale. Nevertheless, if modifications are frequent, the metadata overly may slow down the process of data querying and retrieval. To overcome this issue several approaches have been adopted by different solutions. Popular LST solutions are herein reported.

Delta Lake - Delta Lake makes use of a transaction log along with Apache Parquet files to offer ACID properties over cloud object storages so that consistency and reliability are offered. It has been used to store Online Transaction Processing (OLTP) data, time series and logs. For querying data a fast query engine for lakehouse systems like Photon (Behm and Palkar, 2022) can be used for the integration of SQL operation.

Apache iceberg - Apache Iceberg is a LSTs format that has been designed for high performance and that connects with engines like Spark, Trino, Flink, Presto and object storages. This combined solution supports full SQL, schema evolution, time travel and optimization. It adds metadata layers to the existing files and exposes them as iceberg tables to the engines while maintaining traditional database features like ACID transaction and time travel. Every table change requires that the associated metadata file is replaced by a new one. The format requires that the data are immutable (not changed or moved after they are written), the files support seek and can be deleted or, if maintained, marked as deleted so that the capability of time travel is exposed. A specification for adding geometric data type in Apache Iceberg following the ISO-19107 standard and the OGC-Simple Feature Access specification has been presented (Badard, no date). The latest software release of 28th April 2025 (v1.9.0) added the geometry and geography type support with optional spatial statistics like bounding box calculation.

Apache Hudi - Apache Hudi (Hadoop Upserts Deleted Incrementals), like Apache Iceberg was created to support large data storage in distributed systems. It stores tables in folders and subfolders which comprise file groups sliced in partitions which ultimately contain data in parquet format. Depending on the configuration, changes on tables can be managed with the copy-on-write or merge-on-read: the first creates a copy of the parquet file on any changes and is optimized for read-intensive cases, the second store the updates in delta files that are then merged when the data are requested and is indicated for write-intensive situations (Hellman, 2023).

5. Discussion on Research Challenges and Opportunities

OGC open standards have been widely adopted in the geospatial domain to build Spatial Data Infrastructures (SDIs) that provide access to vast quantities of interoperable geospatial data. These infrastructures have enabled researchers across diverse scientific disciplines to consume and integrate geospatial datasets in their studies, supporting the principles of FAIR data (i.e. making data Findable, Accessible, Interoperable, and Reusable). However, while these standards and services offer robust capabilities for data discovery and access, they lack in supporting a fundamental requirement of Open Science: reproducibility.

To ensure reproducibility, researchers often need to rely on external archive services that duplicate the datasets used during experimentation. When combined with the computational environment, code, and metadata, these immutable snapshots guarantee that a study can be replicated. However, this duplication introduces significant challenges. Research face increased costs associated with storing data in managed FAIR repositories that provide long-term preservation, backups, and service availability. Limitations arise when attempting to archive large datasets, and burdens are introduced by the pre-processing and efforts to prepare datasets for reproducibility.

These challenges are exacerbated by the increasing volume and velocity of geospatial data. Advances in sensing technologies and the widespread deployment of IoT and remote sensing platforms have led to high-frequency, transactional data streams. In such dynamic contexts, duplicating datasets for each experiment becomes inefficient, costly, and often impractical. Moreover, static snapshots quickly become outdated, diverging from their live, continuously updated counterparts, which may compromise the relevance or accuracy of derived scientific results. Persistent identifiers like DOIs, while effective for static resources, are ill-suited for referencing evolving datasets. Although version numbers are common in software development to track changes, they are often unintuitive and difficult to use for meaningful data exploration or temporal analysis. To address this, SQL have introduced support for system-time queries, which enable users to access datasets as they existed at a specific point in time, or to explore how values have changed over defined intervals. Unlike version numbers, system-time attributes provide a more user-friendly and semantically meaningful approach to tracking data evolution.

Despite the potential of this model, current OGC web services only support business-time properties (i.e., user-defined timestamps related to data content) and lack system-time capabilities, which capture the actual time of data storage and modification. This gap significantly hinders the reproducibility of research that depends on interoperable web services.

From a technological standpoint, various data versioning solutions exist, but they vary in approach and supported data formats. Traditional data warehouses offer limited support for temporal SQL. PostgreSQL, for instance, which is a widely used in geospatial contexts due to the PostGIS extension, does not natively support system-time tables. For file-based datasets, Git and Git-like systems allow version control and embed valuable metadata (e.g., commit messages, authorship), but they lack transactional guarantees and can introduce consistency issues in collaborative environments with concurrent access.

Emerging solutions such as Lakehouse Storage Technologies (LSTs) address these limitations by combining ACID-compliant transactions with high-performance, column-oriented storage formats tailored for large-scale tabular data. These platforms offer "travel-time" queries that enable users to retrieve previous versions of data efficiently. However, they currently offer limited support for geospatial formats and indexing, which limits their applicability in the geospatial domain. The current state of the art indicates a clear and growing need for system-time support within the OGC standards ecosystem to accommodate reproducible research workflows based on dynamic datasets. While it has been formalized in SQL standard and implemented in modern data architectures, they are not yet widely integrated into mainstream open-source geospatial tools,

such as those supported by the OSGeo community. Furthermore, integrating Git-like metadata such as author, timestamp, and change motivation into interoperable transactional operations and data versioning systems would enhance not only reproducibility but also data lineage analysis. This would enable researchers to understand the provenance, ownership, and evolution of datasets, fostering transparency and strengthening the credibility of scientific results. In summary, addressing the reproducibility challenges in geospatial research requires a concerted effort to:

1. Integrate system-time capabilities into OGC web service standards, enabling users to reference and retrieve datasets as they existed at specific points in time (Persistent URL).
2. Extend emerging versioned data platforms (e.g., LSTs) to support geospatial data formats and indexing.
3. Promote metadata-rich versioning approaches that capture both technical and human elements of data changes, supporting Open Science principles.
4. Facilitate adoption within existing open-source ecosystems, minimizing the barrier to entry for research institutions and public bodies.

These efforts present a significant opportunity to align geospatial data infrastructures with the evolving needs of Open Science and reproducible research, ensuring that future geospatial analyses are not only FAIR but also credible, transparent, and verifiable.

Acknowledgements

This work was supported by swissuniversities, Programme Open Science: Measure A1, project "OSIReS".

References

Badard, T., no date. *GeoIceberg specification*. (online: https://geoiceberg.org/).

Behm, A., Palkar, S., 2022. Photon: A High-Performance Query Engine for the Lakehouse, *CIDR*. (online: http://cidrdb. org/cidr2022/papers/a100-behm.pdf).

Blanc, N., Cannata, M., Collombin, M., Ertz, O., Giuliani, G., Ingensand, J., 2022. OGC API state of play – a practical testbed for the national spatial data infrastructure in switzerland, *ISPRS-archives*, doi: 10.5194/isprs-archives-XLVIII-4-W1-2022-59-2022.

Camacho-Rodríguez, J. et al., 2023. LST-Bench: Benchmarking Log-Structured Tables in the Cloud, *arXiv*. (online: http://arxiv.org/abs/2305.01120).

Cannata, M., et al.,2019. Performance Testing of istSOS under High Load Scenarios, *ISPRS International Journal of Geo-Information*, Multidisciplinary Digital Publishing Institute, 8, 11, 467. doi: 10.3390/ijgi8110467.

Cerutti, V et al., 2021. Improving the reproducibility of geospatial scientific workflows: the use of geosocial media in facilitating disaster response, *Journal of Spatial Science*, Taylor & Francis, 66, 3, 383–400. doi: 10.1080/14498596.2019.1654944.

Chiodi, P., 2023. Temporal Tables, NearForm. (online: https://github.com/nearform/temporal_tables).

Deshpande, K., 2004. Oracle9i: Understanding Automatic Undo Management and Flashback Query, *SELECT Journal*, Independent Oracle Users Group, 11, 4, 22–30.

Dwyer, J. L et al., 2018. Analysis Ready Data: Enabling Analysis of the Landsat Archive, *Remote Sensing*, 10, 9, 1363. doi: 10.3390/rs10091363.

European Commission, no date. Facts and Figures for open research data, Researhc and Innovation. (online).

Fearing, V., 2023. Periods and system versioning for PostgreSQL. (https://github.com/xocolatl/periods).

Ferreira, K. R., Queiroz et al., 2020. Earth Observation Data Cubes for Brazil: Requirements, Methodology and Products, *Remote Sensing*, 12, 24, 4033. doi: 10.3390/rs12244033.

Giuliani, G. et al., 2021. SwissEnvEO: A FAIR National Environmental Data Repository for Earth Observation Open Science, *Data Science Journal*, 20, 22–22. doi: 10.5334/dsj-2021-022.

Giuliani, G et al., 2017. Building an Earth Observations Data Cube, *Big Earth Data*, 1, 1–2, 100–117. doi: 10.1080/20964471.2017.1398903.

Giuliani, G et al., 2019. Paving the Way to Increased Interoperability of Earth Observations Data Cubes, *Data*, 4, 3, 113. doi: 10.3390/data4030113.

Gregg, C., 2023. Familiar with Oracle Flashback Time Travel? (online: https://blogs.oracle.com/dbstorage/post/familiar-with-oracle-flashback-time-travel-if-not-keep-reading).

Groth, P., Moreau, L., 2013. PROV-overview, *W3C Working Group Note*, 1135, 881–906.

Hellman, F., 2023. Study and Comparsion of Data Lakehouse Systems. (https://www.doria.fi).

Huang, S et al., 2020. Orpheus db: bolt-on versioning for relational databases, *VLDB Journal*, Springer, 29, 1, 509–538.

Jungwirth, P. A., 2019. Temporal Databases: Theory And Postgres, *PGCon2019*, (https://github.com/pjungwir/postgres-temporal-talk/blob/master/slides.pdf).

Kandpal, N et al., 2023. Git-Theta: A Git Extension for Collaborative Development of Machine Learning Models, (online: http://arxiv.org/abs/2306.04529).

Kedron, P et al., 2021. Reproducibility and replicability: opportunities and challenges for geospatial research, *IJGIS*, Taylor & Francis, 35, 3, 427–445.

Kimball, R., 2008. Slowly changing dimensions, *Information Management*, SourceMedia, 18, 9, 29.

Kirchhoff, M., Geihs, K., 2013. Semantic description of OData services, in, *Proceedings of the 5th Workshop on SWIM*, NY, USA, ACM, pp. 1–8. doi: 10.1145/2484712.2484714.

KNIME, no date. *KNIME Analytics Platform*, *KNIME*. (online: https://www.knime.com/knime-analytics-platform).

Konkol, M., Kray, C., 2019. In-depth examination of spatiotemporal figures in open reproducible research, *Cartography and GI Science*, Taylor & Francis, 46, 5, 412–427

Krishnamurthi, R et al., 2020. An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques, *Sensors*, MDPI, 20, 21, 6076. doi: 10.3390/s20216076.

Kulkarni, K., Michels, J.-E., 2012. Temporal features in SQL:2011, *ACM SIGMOD Record*, 41, 3, 34–43.

Lankester, T. H., 2009. OpenGIS Web Map Services-Profile for EO Products, Version: 0.3. 3., OGC.

Lewis, A. et al., 2017. The Australian Geoscience Data Cube — Foundations and lessons learned, *Remote Sensing of Environment*, 202, 276–292. doi: 10.1016/j.rse.2017.03.015.

Liu, C. et al., 2023. A Deep Dive into Common Open Formats for AnalyticalDBMSs, *VLDB*, doi: 10.14778/3611479.3611507.

Low, Y. et al., 2023. Git is for data, in, *Conference on Innovative Data Systems Research*.

MariaDB, *System-Versioned Tables*, *MariaDB KnowledgeBase*. (https://mariadb.com/kb/en/system-versioned-tables/).

Martini, A. et al., 2019. Database-Supported Change Analysis And Quality Evaluation Of Openstreetmap Data, *ISPRS Annals, Volume IV-2/W5*, Copernicus GmbH, IV-2-W5, 535–541.

Mathis, C., 2017. Data Lakes, *Datenbank-Spektrum*, 17, 3, 289–293. doi: 10.1007/s13222-017-0272-7.

Merodio Gómez, P. et al., 2019. The Americas' Spatial Data Infrastructure, *ISPRS Int. Journal of Geo-Information*, MDPI, 8, 10, 432. doi: 10.3390/ijgi8100432.

Mocnik, F.-B. et al., 2018. Open source data mining infrastructure for exploring and analysing OpenStreetMap, *Open Geospatial Data, Software and Standards*, 3. doi: 10.1186/s40965-018-0047-6.

Novella, J. A. et al.,, 2019. Container-based bioinformatics with Pachyderm, *Bioinformatics*, 35, 5, 839–846. doi: 10.1093/bioinformatics/bty699.

Nüst, D., Pebesma, E., 2021. Practical Reproducibility in Geography and Geosciences, *Annals of the AAG*, Taylor & Francis, 111, 5, 1300–1310

Park, K.-T. et al., 2008. Lake: Towards Highly Manageable Cluster Storage for Extremely Scalable Services, in, *ICCSA*, pp. 122–131. doi: 10.1109/ICCSA.2008.37.

Peuster, M. et al., 2019. The Softwarised Network Data Zoo, arXiv. (online: http://arxiv.org/abs/1905.04962).

Pozzoni, M. et al., 2020. Retrospective and prospective of hydro-met monitoring system in the Canton Ticino, Switzerland, *HSJ*, Taylor & Francis, 0, 0, 1–15. doi: 10.1080/02626667.2020.1760280.

Ramachandran, R., Bugbee, K., Murphy, K., 2021. From Open Data to Open Science, *Earth and Space Science*, 8, 5, e2020EA001562. doi: 10.1029/2020EA001562.

rwestMSFT, 2023. *Create a system-versioned temporal table - SQL Server*. (https://learn.microsoft.com).

Saracco, C. M. et al, 2010. A matter of time: Temporal data management in DB2 for z, *IBM Corporation, New York*, 7.

Sehn, T., 2022. *When to make a Dolt Commit | DoltHub Blog*. (https://dolthub.com/blog/2022-09-28-when-to-dolt-commit/).

Shcheklein, 2020. Answer to 'By how much can i approx. reduce disk volume by using dvc?', *Stack Overflow*. (online: https://stackoverflow.com/a/60366262).

Soroush, E., Balazinska, M., 2013. Time travel in a scientific array database, in, *2013 IEEE 29th ICDE* pp. 98–109. doi: 10.1109/ICDE.2013.6544817.

STAC Contributors, 2024. SpatioTemporal Asset Catalog (STAC) specification. (online: https://stacspec.org).

Strigaro, D. et al, 2022. Open and Cost-Effective Digital Ecosystem for Lake Water Quality Monitoring, *Sensors*, 22, 17, 6684. doi: 10.3390/s22176684.

Vohra, D., Vohra, D., 2016. Apache parquet, *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools*, Springer, 325–335.

Wang, C. et al, 2008. Importance-Driven Time-Varying Data Visualization, *IEEE Transactions on VCG*, 14, 6, 1547–1554. doi: 10.1109/TVCG.2008.140.

Yin, D. et al, 2019. CyberGIS-Jupyter for reproducible and scalable geospatial analytics, *Concurrency and Computation: Practice and Experience*, 31, 11, e5040. doi: 10.1002/cpe.5040.

Zahumenskỳ, I., 2004. Guidelines on quality control procedures for data from automatic weather stations, *World Meteorological Organization, Switzerland*, 955, 2–6.