ICA2025OSLO
14th IWA Specialised Conference on
Instrumentation, Control and Automation

# met*EAU*data: A framework for automatic metadata generation for environmental time series pre-processing

J.D. Therrien, P.A. Vanrolleghem

model*EAU*, Département de génie civil et de génie des eaux, Université Laval, 1065 Avenue de la Médecine, Québec, Québec, G1V 0A6 Canada. E-mail: jean-david.therrien.1@ulaval.ca; peter.vanrolleghem@gci.ulaval.ca

**Abstract:** Water Resource Recovery Facilities (WRRFs) generate tremendous amounts of process data, most of it in the form of time series. This data can be found in Supervisory Control and Data Acquisition (SCADA) systems, process historians or relational databases where metadata is (hopefully) abundant. Removal of data artifacts and sensor faults, as well as smoothing and aggregation, are all essential steps in transforming raw data into a form fit for modelling, analysis and decision-making. These transformations typically ignore metadata, meaning metadata-rich datasets can quickly lose all context, traceability and reusability. The context at risk of being lost includes both the original metadata and descriptions of the processing itself. Failure to report these metadata is due to the difficulty of manual reporting and to the sometimes highly iterative nature of data pre-processing. This paper presents met*EAU*data, a framework that automates metadata creation and preservation during pre-processing operations. Unlike existing solutions that bundle limited preprocessing algorithms, met*EAU*data provides a lightweight interface enabling any algorithm to become metadata-aware through simple wrapper functions, extending metadata recording to existing scientific libraries with minimal effort. By automating the creation of structured, extensive metadata for arbitrary pre-processing pipelines, met*EAU*data supports the transition towards FAIR-compliant time series data.

**Keywords**: data pipelines; digitalization; metadata; pre-processing; open-source software

## INTRODUCTION

Water Resource Recovery Facilities (WRRFs) are becoming increasingly complex as utilities adapt to stricter effluent standards and strive to extract useful resources from wastewater. At the same time, urbanisation puts WRRFs under increased load, and economic constraints push utilities to cut costs wherever possible. Meeting those challenges requires WRRFs to run more reliably and to control their processes more accurately (Vanrolleghem and Vaneeckhaute, 2014). Collecting data is a necessary component of achieving these goals. By adding data to pre-existing process understanding, mechanistic and data-driven modelling tools, and control algorithms, WRRFs can transform the data they collect into new process insights (Makropoulos and Savić, 2019).

However, interpreting raw WRRF process data is complicated by the harsh operating conditions and process upsets that affect the plant and its sensing equipment (Alferes et al., 2014). Sensor technologies are varied, and with each, operational challenges can arise, potentially leading to sensor faults (Newhart et al., 2019). Some faults are evident from the data itself (e.g., a constant zero value of total suspended solids (TSS) in a sludge return line) while others can be difficult to distinguish from real dynamics (e.g., a slow drift in influent COD). In addition to plain faults, interpretation of process data also depends on the plant context, i.e., what operational events (e.g., tank maintenance, process upsets) coincide with observations.

In either case, an analyst inspecting raw data must have access to context to understand not only the data but what the data means (Aguado et al., 2024). Metadata is the collective name for all pieces of context that are recorded alongside data. The presence of metadata can be the deciding factor that determines whether a dataset is reused. Models, controllers and decision support systems are only as reliable and effective as the data used to create them. Thus, using data without context can lead to misunderstandings and false assumptions that might prove counterproductive or harmful. The risk associated with using poorly documented data slows the development and adoption of data-driven tools, leading to loss of value, data graveyards, and wasted opportunity. It is therefore critical to consider metadata to be as precious as data itself for the creation of high-value datasets.

Simple plants with few sensors and resources may perform all their data and metadata tracking by visually analysing time series inside their SCADA system or via spreadsheet software. However, as plants scale and data sources multiply, simple methods quickly become unwieldy. More sophisticated preprocessing tools, such as automatic data cleaning, fault detection algorithms, and data mining schemes, become advantageous for surfacing the most relevant information from large datasets (Therrien et al., 2020). A flow thus emerges by which data is first collected, then stored, retrieved, pre-processed and mined, used in models, and finally presented to an agent (whether automated or human) who can then take data-driven action. However, trust regulates the flow of data from one stage to the next. If one does not trust that the data that went into a model truly reflects reality, the model will be discarded, destroying any possibility of data-driven action (Therrien, 2024a).

Incorrectly calibrated or deployed pre-processing pipelines may mistake relevant process information as errors and real errors as valid data, discarding critical information. Transparency and explainability are therefore essential to establish the trust required by later uses of the data. One way to achieve this transparency is for the tools to generate metadata that can be audited. This metadata would consist of describing what each tool does, how it is applied, and for what purpose.

Pre-processing consists of two closely related activities: bringing a signal into a form that fits the requirements of another data system (*e.g.,* setting the time increment, the data types, or imputing missing values in preparation for modelling) and increasing its quality (*e.g.,* outlier detection, or sensor fault detection). Data pre-processing and modelling pipelines can apply many algorithms in sequence, and parameter tuning is an inherently iterative process. Expecting analysts to maintain accurate records over entire pipelines is thus time-consuming and error-prone.

However, even if metadata is generated, it is likely to be overlooked if it is inconvenient to browse or challenging to parse. Consistent structures and automatic generation tools are, thus, both essential to ensuring that metadata is available and usable (Aguado et al., 2024).

Researchers have previously proposed frameworks for metadata collection in WRRFs. For instance, both Plana et al.'s dat*EAU*base (2019) and ETH Zurich's WaterHubDataPool (2019) offer relational database schemas that allow users to store WRRF process data along with its metadata (e.g., what water quality parameter it represents, the sensor that produced it, the sensor location, and the overarching project). However, no data standard currently exists in the field. Thus, the current situation is that each data producer creates bespoke data organisation schemes, leading to challenges in the interoperability of data tools for the wastewater field.

Fewer groups have ventured beyond data collection to create and organise pre-processing metadata. For instance, dat*EAU*base does not currently accept pre-processing metadata (beyond acknowledging that pre-processing occurred). The *wwdata* Python package proposed by de Mulder et al. (2018) is one notable example of a software library for processing time series data that also automatically generates pre-processing metadata. However, its degree of specificity is limited. The package creates and stores information on whether a data point was imputed, rejected or accepted. However, it does not record how those decisions were made, or which processing function was used to arrive at a given decision. Moreover, the package does not support carrying metadata from previous steps of the data's lifetime (i.e., collection metadata) forward with the new data it produces, increasing the risk that existing metadata will be lost before the dataset reaches its end user.

Another example found in the literature is that of Daneshgar et al. (2024)'s, *wwtwin* library, which underpins a fully automated data pipeline between data capture from WRRF sensors, up to its use in digital twin models. The package also creates pre-processing metadata for a fully traceable data transformation pipeline. However, the precise nature of the metadata generated by *wwtwin* is not

yet reported in the literature. Moreover, the package is not yet publicly released, making its inspection or reuse impossible.

The Open Geospatial Consortium's WaterML2.0 standard (2011), on the other hand, does include specifications for reporting so-called "procedures" applied to time series data. These procedures include pre-processing operations such as resampling and the application of algorithms. However, the standard is complex, and few tools exist to translate data transformations into WaterML2.0-compliant representations.

The scientific community is increasingly appreciating the importance of FAIR data (Wilkinson et al., 2016), meaning that data (and metadata) should be findable, accessible, interoperable and reusable. The structure and searchability that relational schemas such as dat*EAU*base and WaterHubDataPool help make data findable. Publishing datasets online with open licenses in public repositories makes them accessible. Metadata-aware exchange formats, such as WaterML2.0, also make the data interoperable. However, reusability suffers if datasets do not report how data has been transformed between collection and publishing. Having schemas to report this information (as in WaterML2.0) is helpful, however, these schemas will only contain empty fields if processing tools fail to record metadata.

Consequently, there exists a need for metadata generation tools that provide rich details about each pre-processing operation while maintaining previous metadata. Such a tool should be flexible enough to work with any pre-processing operation, be openly accessible and reusable, and support the creation of FAIR-compliant datasets. In this work, these needs are addressed through a lightweight, open Python package named met*EAU*data. The package, currently available in open-source (Therrien, 2024b), consists of two main components:

1. A collection of structured objects that can hold and organise metadata concerning the data's origin and the transformations it undergoes with minimal direct intervention from the user.
2. A programming interface to apply transformations to data while automatically creating relevant metadata.

In the following sections, the architecture of met*EAU*data is described in detail, the design choices are justified, and the extent to which met*EAU*data addresses these issues is discussed.

## METHODS

The two basic assumptions made in the design of met*EAU*data are the following:

1. The fundamental unit of WRRF data is the time series, *i.e.*, a group of values tracking the same quantity over time, and
2. Metadata accumulates through incremental operations (*e.g.,* collection, then outlier removal may follow, then interpolation may be done on the outlier-free data, and so on).

These assumptions reflect both the nature of environmental monitoring (continuous sensor measurements) and typical preprocessing workflows (sequential application of algorithms). The design of metEAUdata relies on these assumptions to create a hierarchy of objects. Each object contains attributes that store information specific to itself and lower-order objects, as shown in Figure 1 and explained below.

Datasets contain attributes that identify the creator and their purpose in extracting, grouping and processing the data (e.g., for reporting, or as inputs to a model). Quantities that are measured and tracked over time (e.g., the stream of pH measurements that come out of a pH sensor) are represented as Signal objects. Signals contain each version of their data generated during pre-processing. In other words, if a raw pH signal was later smoothed, the raw and smoothed versions of the data would belong to the same Signal. Signal objects have metadata attributes akin to those

used by Plana et al.'s datEAUbase (2019) to describe the signal's provenance, i.e., the context of its collection.

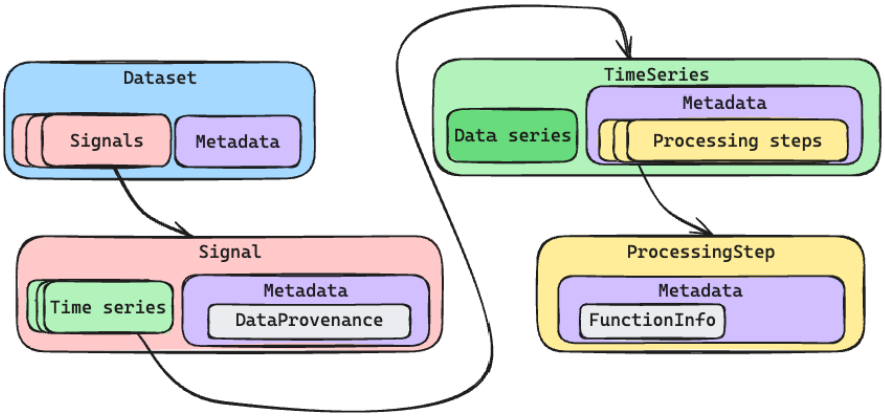


**FIGURE 1** The met*EAU*data object hierarchy. Dataset objects (blue) contain one or many Signals (pink), each representing a single measured quantity. Signal objects contain one or many TimeSeries (green), representing different versions of the same signal issued from processing operations. Each TimeSeries contains ProcessingStep objects (yellow) that document individual operations.

Each version of a Signal's data is encapsulated in a distinct TimeSeries object. These objects hold not only the (raw or processed) data and timestamps, but also an ordered list of ProcessingStep objects. ProcessingSteps detail the nature and sequence of each pre-processing operation that led to a particular version of the data. ProcessingStep objects describe which algorithm was applied, which TimeSeries served as input, and what parameter values were used. ProcessingSteps contain a metadata object (FunctionInfo) that describes the specific algorithm (e.g., Python function) that was used to carry out the processing step (including its version number and source code) to ensure complete transparency and repeatability of the processing workflow.

This object hierarchy results in a nested, append-only data structure that allows metadata to follow data throughout pre-processing operations. The three-tiered structure of Dataset, Signals and TimeSeries separates metadata about the analytical purpose (Dataset), the origin of the data (Signals), and individual operations they undergo (TimeSeries). The metadata structure and non-destructive accumulation of metadata thus lead to auditable and repeatable data pipelines, as argued for by Rieger and Vanrolleghem (2008).

Having now described the object hierarchy, the following section explains how users interact with it during a pre-processing session.

## METADATA MANAGEMENT DURING PRE-PROCESSING

Dataset and Signal are the objects users interact with the most. At the instantiation of an object, users provide their raw data, and they are prompted to supply values for metadata attributes, much like one would be while filling out a short form. All metadata except the name attribute is optional, meaning that objects will instantiate and work normally even if some attributes are left blank. Allowing users to select the level of fidelity they require from their metadata allows metEAUdata to be exhaustive if desired, and non-obtrusive otherwise.

The bulk of user interactions with metEAUdata occur after instantiation, while applying pre-processing operations to data. Contrary to other metadata-aware libraries, the core functionality of metEAUdata does not rely on pre-bundled processing algorithms. Instead, the package provides interfaces through which any processing function can be applied and ensures metadata is automatically added as these algorithms perform their work. The interfaces consist of two object-

oriented methods named "process" found in the Signal and Dataset objects. These methods act as a centralised meeting point for inputs, algorithms, and parameters. Algorithms may be univariate, in which case the "Signal.process" method is used, or multivariate, in which case "Dataset.process" is available. In either case, the user provides the following three pieces of data to the method as arguments:

1. The names of the TimeSeries that should be provided as inputs to the processing algorithm.
2. The processing algorithm itself, presented in the form of a Python function or callable class.
3. An arbitrary long list of arguments and keyword arguments (parameters) required by the processing algorithm.

With these three pieces of information, both the Dataset and Signal objects execute the following tasks:

1. Extract the specified input series from the Signal or Dataset's internal structure.
2. Execute the algorithm on the inputs, with the appropriate parameters.
3. Encapsulate the algorithm's output in new Time Series objects (univariate) or Signals (multivariate).
4. Add to the outputs any ProcessingStep objects found in the input TimeSeries objects.
5. Append to the list of preexisting ProcessingStep objects a new one that describes the newly applied algorithm.

A diagram illustrating the use of the "Signal.process" method is provided in Figure 2. By executing processing operations via the process methods instead of their algorithm's original interface, users gain access to complete metadata records for each step of their pre-processing pipeline without any additional record-keeping action on their part.

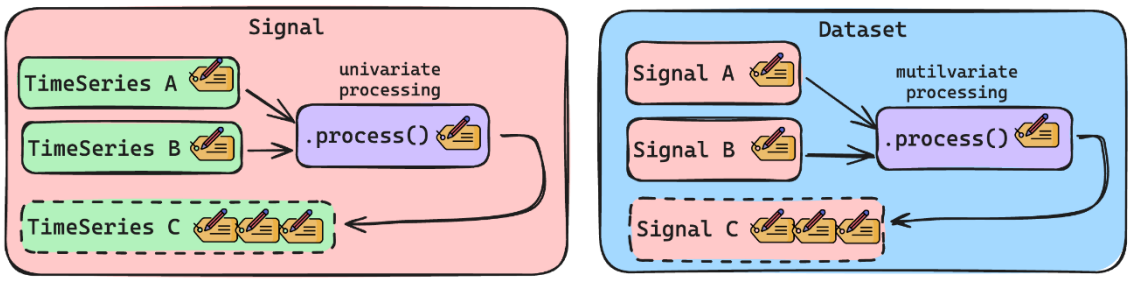


**FIGURE 2**: met*EAU*data processing interfaces. Dataset.process() enables multivariate processing across multiple Signals, while Signal.process() handles univariate operations on TimeSeries within a single Signal. Both methods create new objects while preserving accumulated metadata.

The interface offered by metEAUdata can be made compatible with virtually any processing algorithm, provided the algorithm can be accessed via a function that adheres to the protocol specified by metEAUdata. That is, the algorithm should accept as arguments the list of arrays it is operating on, followed by any other parameter, and it should return each output array with a list of ProcessingStep objects representing what the algorithm does, how it does it, and what parameters were used.

By asking for this protocol to be followed by compatible algorithms, metEAUdata effectively splits the burden of specifying metadata between algorithm users (who enter information about the data's origin and purpose) and algorithm implementers (who best understand the inner workings of their algorithm).

In practice, the protocol requirement means that algorithms must be adapted before they can be used with metEAUdata. However, this requirement can be fulfilled with a simple function that

wraps the original algorithm and outputs its results with the required metadata objects. A "wrapper" consists of a function that performs the following tasks:

1. Create a FunctionInfo metadata object to describe what the algorithm does.
2. Execute the algorithm according to its original calling convention.
3. Return, alongside the algorithm's output, a ProcessingStep object that includes the FunctionInfo object, the parameter values used with the algorithm, and the names of the input time series.

A simple wrapper example is presented in Figure 3. For simple algorithms acting on a single data array at a time, wrappers can be implemented in approximately 20 lines of code.

```
def meteaudata_interpolate(input_data, method="linear"):
    # 1. Create FunctionInfo
    func_info = FunctionInfo(name="pandas.interpolate", ...)

    # 2. Execute algorithm
    result = pd.Series(input_data).interpolate(method=method)

    # 3. Return with ProcessingStep(s)
    return (
        result,
        [ProcessingStep(func_info, Parameters(method=method), ...)]
    )
```

**FIGURE 3**: Pseudo-code example of a wrapper function that turns a third-party pre-processing algorithm into a met*EAU*data-compatible algorithm.

## ENSURING INTEROPERABILITY AND REUSABILITY

Metadata that cannot be easily accessed, shared, or reused provides no practical value beyond the original analysis session. The design of metEAUdata aims to improve the reusability of both data and the structured metadata it produces. This concern is addressed by making each object fully serializable and deserializable into a text-based representation. This functionality is provided by simple save and load methods found in all metEAUdata objects. The ability to serialise the objects into structured text means that both data and metadata are human-readable outside of a Python session, enabling users of other tools and computing environments to still benefit from the information metEAUdata produces after the pre-processing step of their workflow. It also enables integration with version control systems and facilitates long-term data preservation independent of the availability of specific software.

TimeSeries, Signals and Datasets are split into two types of files during serialisation. Data is stored in a plain CSV file, making it ready for use in other computing environments such as simulators, R or MATLAB. The metadata is saved to a separate YAML file that mirrors the nested structure of metEAUdata objects. YAML was chosen for its human-friendly syntax and its ability to represent complex, nested data structures. Besides making the data interoperable across computing environments, serialisation and de-serialisation mean that the state of data processed using metEAUdata can be preserved across computing sessions and between users of the same dataset, increasing reusability inside metEAUdata itself.

## APPLYING TO REAL-LIFE PRE-PROCESSING WORKFLOWS

The core of metEAUdata consists of the metadata objects and programming interfaces it provides. However, the package also includes a selection of simple pre-processing functions for both univariate and multivariate processing, including time series resampling and interpolation. These consist of simple wrappers around functions from McKinney's pandas library (2011).

The interoperability of metEAUdata with a variety of pre-processing workflows was tested by applying it to multiple independently developed processing suites. In addition to pandas, metEAUdata wrapper functions were created for a Python implementation of the data validation process proposed by Alferes and Vanrolleghem (2016), in addition to the Principal Component Analysis (PCA)-based multivariate fault detection methods of Alferes et al. (2012) and the data-driven Long Short-Term Memory (LSTM) outlier detection of (Covre, 2024). Though the original interfaces of the algorithms varied significantly, creating wrapping functions was always straightforward, following the blueprint shown in Figure 3. Moreover, once wrappers were available, the uniform interface of the "process" methods meant that lay users (e.g., students and new staff) could apply algorithms from any wrapped library using the identical three-argument pattern (input data, processing function, parameters), regardless of the underlying algorithm complexity. Figure 4 represents an example of a full data pre-processing workflow using processing algorithms from multiple libraries. The figure shows how the algorithm origin (green or red boxes) is fully transparent to users because of the uniform calling convention.

This direct experience with metEAUdata demonstrated its general compatibility across diverse algorithm types, such as machine learning approaches (LSTM), statistical methods (PCA), autoregressive filtering and rules-based filters (Alferes and Vanrolleghem, 2016). Moreover, low development overhead (the wrappers required an average of 20 lines of Python code per output TimeSeries) suggests that metEAUdata could serve as a unifying framework for the growing ecosystem of WRRF data processing tools without overburdening developers within the ecosystem. The unified interface also encourages the adoption of new pre-processing techniques in the WRRF community by lowering implementation complexity.

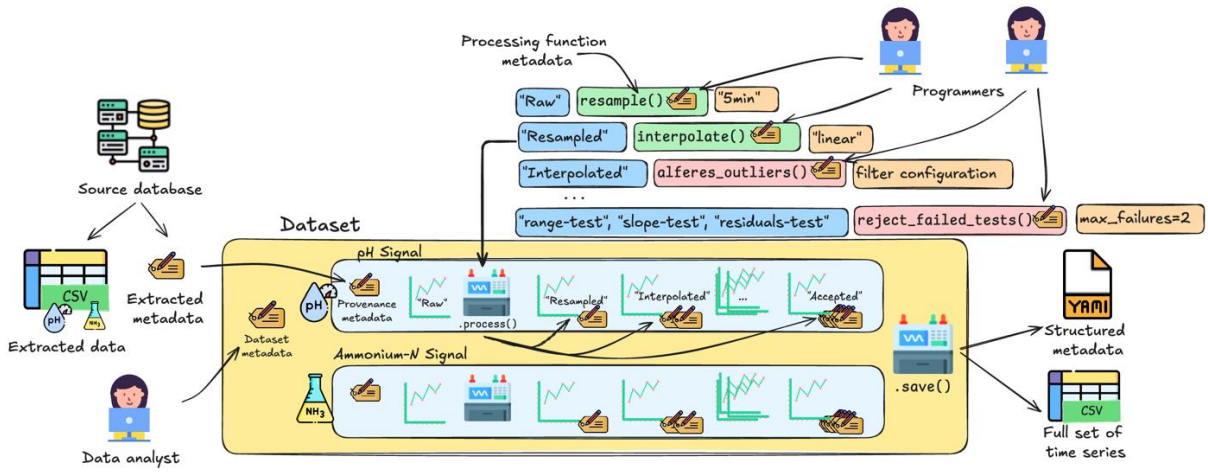


**Figure 1:** Complete met*EAU*data preprocessing workflow showing dual metadata streams. Data analysts contribute extracted data and contextual metadata describing the analysis they are performing. Meanwhile, programmers provide processing functions with embedded algorithmic metadata. met*EAU*data then orchestrates the processing and accumulates structured metadata until it is exported for use in other tools.

## DISCUSSION

The preceding sections highlighted the challenges faced by users of environmental time series as they interact with their data and proposed a framework, metEAUdata, for addressing these concerns. One of the issues outlined is the difficulty in preserving existing metadata while executing pre-processing operations and during later use of the data. By making metadata an integral component of the objects that orchestrate pre-processing operations, metEAUdata ensures that pre-existing metadata is not lost during processing. The difficulty in reporting pre-processing operations is also addressed in two different ways: by making metadata accumulation passive on

the user side, and by providing straightforward protocols for library developers to describe their algorithms.

Splitting the responsibility of metadata creation among two parties—users and developers—lowers the burden on users and makes the task of metadata tracking more palatable. While this approach introduces dependency on developer participation, the minimal wrapper requirements and clear protocols reduce this risk. Moreover, the alternative—relying solely on users to maintain comprehensive metadata—has proven consistently unreliable in practice.

The tools presented in this work advance FAIR compliance in multiple ways.

- **Findability**: The package creates findable datasets by keeping metadata close to data and presenting metadata in a searchable structure. The metadata it contains is extensive (e.g., processing steps with algorithm and parameter values, signal origins, dataset purpose). Exporting metadata to YAML also means that metadata can be accessed even if the original data is no longer available.
- **Accessibility**: The package's open-source status and free license make it highly accessible. Moreover, the package increases accessibility of the data it produces by making the output metadata machine-actionable (structured) while remaining human-readable. It also removes a barrier to entry by decoupling the metadata creation process from specific algorithms. This means that a group seeking to use metEAUdata but reticent to release their algorithm's source could choose to use metEAUdata as an internal dependency, rather than having to contribute their algorithm as a dependency to metEAUdata itself.
- **Interoperability**: By proposing a unified interface for applying pre-processing algorithms to time series data, it provides a path for integrating pre-processing methodologies across the sector. The selected output formats (CSV and YAML) are text-based, making metEAUdata outputs interoperable with virtually all computing platforms, as well as with version control tools such as Git. However, interoperability could be significantly improved by providing methods to map the metadata schema to an established standard such as WaterML2.0. Though mapping to various standards is technically feasible, no such functionality is currently implemented in metEAUdata. Future versions should therefore focus on standards compliance to increase interoperability with existing datasets.
- **Reusability** is enhanced for the pre-processed data by enriching it with extensive metadata. By collecting precise details concerning algorithms, input time series, parameter values, and the sequence of operations, metEAUdata provides enough information to potentially re-execute pre-processing pipelines. This ability not only creates transparency but also allows later users to repurpose pre-processing pipelines documented with metEAUdata with other datasets. The export to CSV also makes the data directly reusable with tools further along in a plant's data pipeline, such as models, controllers, or Key Process Indicator (KPI) reports.

In terms of user experience, one of the pathways to simplifying the use of metEAUdata would be to develop mapping tools that automatically transfer data and metadata to metEAUdata objects, rather than requiring users to fill in this metadata manually at the beginning of a session. However, given the diversity of WRRF data storage approaches—from proprietary SCADA historians to custom relational schemas like datEAUbase and WaterHubDataPool—developing universal mappers is impractical. Still, including mapper base classes within the metEAUdata package would help users implement their own, reducing both developer burden and, in the long run, user burden.

The functional scope of metEAUdata can also be expanded. For instance, while metEAUdata handles time series processing reliably, the current objects are not optimally suited to store quantities derived from time series that are not time series themselves. For example, computing a correlation coefficient between two time series or generating summary statistics would require an extension to the current Signal (for cross-time series statistics) and TimeSeries (for single time series) objects.

On the performance side, the current implementation of metEAUdata holds its data in memory during pre-processing sessions, which may lead to performance concerns for high-frequency, multi-year datasets. Thus, workflows with many steps or long time series would benefit if metEAUdata were extended to communicate with data stores such as relational databases when desired. The modular structure of the package would allow this feature to be included without modifying the user workflow.

Following Therrien et al. (2020)'s framework, metEAUdata is positioned between the data collection and modelling stages, on its way to informing data-driven decisions. This positioning means that whatever enriched metadata is produced at this stage can only deliver value if it successfully informs downstream modelling, control decisions, and operational insights. Future work could further enrich this metadata foundation by developing tools that support modelling workflows (tracking model versions, training data, performance metrics) and decision support systems (documenting control logic, or operational context). Visual tools that facilitate the inspection and discovery of metadata would also be impactful in improving and facilitating metadata use. By facilitating the passage of rich metadata through an additional step of the data pipeline, this work paves the way for fully traceable, metadata-aware WRRF data systems where every operational decision can be traced back through validated data and documented transformations to original sensor measurements.

## CONCLUSION

metEAUdata proposes open-source tools–a hierarchy of objects, a programming interface and helper methods–to facilitate the automatic creation of extensive metadata during pre-processing, while maintaining existing metadata about its origins. By splitting metadata creation between programmers and users, it lowers the burden of metadata creation to manageable levels for both groups. Users provide information on their data's origin and purpose, whereas programmers do the same for their algorithms–without needing to completely change their implementations (~20-line wrapper functions are sufficient). As such, it bridges the gap between raw data storage and modelling in WRRF data pipelines. With its programming interface and open-source license, the package contributes to building an open, composable and metadata-aware computing ecosystem where diverse tools can work together while maintaining data traceability. This work paves the way for auditable and transparent data-driven WRRF operations while remaining general enough for use in any other environment where time series are abundant.

## DATA AVAILABILITY STATEMENT

The open-source package met*EAU*data can be found on GitHub at the link: https://github.com/modelEAU/meteaudata

**ACKNOWLEDGEMENTS**

**This work was supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant RGPIN-202104347**

## REFERENCES

Aguado, D., Alferes, J., Plana, Q., Ruano, M. V., Samuelsson, O., and Villez, K. (2024) *Metadata Collection and Organization in Wastewater Treatment and Wastewater Resource Recovery Systems*, London, UK, IWA Publishing. DOI: 10.2166/9781789061154.

Alferes, J., Lamaire-Chad, C., Chetri, R. K., Thirsing, C., Sharma, A. K., Mikkelsen, P. S., and Vanrolleghem, P. A. (2014) Advanced monitoring of wastewater quality: Data collection and data quality assurance. In: Proceedings 13th International Conference on Urban Drainage. Sarawak, Malaysia, September 7-12 2014.

Alferes, J., Tik, S., Copp, J., and Vanrolleghem, P. A. (2012) Advanced monitoring of water systems using in situ measurement stations: Data validation and fault detection. In: Proceedings New Developments in IT & Water Conference. Amsterdam, The Netherlands, November 4-6 2012.

Alferes, J. and Vanrolleghem, P. A. (2016) Efficient automated quality assessment: Dealing with faulty on-line water quality sensors. AI Communications, **29**, 701–709.

Covre, N. (2024) Automatic validation of high-frequency water quality data without the need of expert tuning. M. Sc. Thesis, Université Laval, Québec, QC, Canada. [online] https://hdl.handle.net/20.500.11794/153490 (Accessed June 16, 2025).

Daneshgar, S., Polesel, F., Borzooei, S., Sørensen, H. R., Peeters, R., Weijers, S., Nopens, I., and Torfs, E. (2024) A full-scale operational digital twin for a water resource recovery facility-A case study of Eindhoven Water Resource Recovery Facility. Water Environment Research, **96**(3), e11016. DOI: 10.1002/wer.11016.

De Mulder, C., Flameling, T., Weijers, S., Amerlinck, Y., and Nopens, I. (2018) An open software package for data reconciliation and gap filling in preparation of Water and Resource Recovery Facility modeling. Environmental Modelling and Software, **107**, 186–198. DOI: 10.1016/j.envsoft.2018.05.015.

ETH Zurich (2019) WHDP documentation. [online] https://whdp.readthedocs.io/en/latest/index.html (Accessed June 3, 2025).

Makropoulos, D. A. and Savić, C. (2019) Urban hydroinformatics: past, present and future. Water (Switzerland), **11**(10), 1959. DOI: 10.3390/w11101959.

McKinney, W. (2011) Pandas: A foundational Python library for data analysis and statistics. [online] http://pandas.pydata.org.

Newhart, K. B., Holloway, R. W., Hering, A. S., and Cath, T. Y. (2019) Data-driven performance analyses of wastewater treatment plants: A review. Water Research, **157**, 498–513. DOI: 10.1016/j.watres.2019.03.030.

Open Geospatial Consortium (2011) OGC WaterML 2.0: Part 1- Timeseries. Measurement, 1–151. DOI: 10.25607/OBP-611.

Plana, Q., Alferes, J., Fuks, K., Kraft, T., Maruéjouls, T., Torfs, E., and Vanrolleghem, P. A. (2019) Towards a water quality database for raw and validated data with emphasis on structured metadata. Water Quality Research Journal, **54**(1), 1–9. DOI: 10.2166/wqrj.2018.013.

Rieger, L. and Vanrolleghem, P. A. (2008) monEAU: A platform for water quality monitoring networks. Water Science and Technology, **57**(7), 1079–1086. DOI: 10.2166/wst.2008.135.

Therrien, J.-D. (2024a) Traversing the wastewater data pipeline: increasing the utility of data for WRRF and public health decision-making. Ph. D. Thesis, Université Laval, Québec, QC, Canada. [online] https://hdl.handle.net/20.500.11794/155814 (Accessed June 16, 2025).

Therrien J.-D. (2024b) metEAUdata. GitHub. [online] https://github.com/modelEAU/meteaudata (Accessed June 16, 2025).

Therrien, J. D., Nicolaï, N., and Vanrolleghem, P. A. (2020) A critical review of the data pipeline: How wastewater system operation flows from data to intelligence. Water Science and Technology, **82**(12), 2613-2634. DOI: 10.2166/wst.2020.393.

Vanrolleghem, P. A. and Vaneeckhaute, C. (2014) Resource recovery from wastewater and sludge: modelling and control challenges. In: Global Challenges: Sustainable Wastewater Treatment and Resource Recovery, IWA Specialist Conference. Kathmandu, Nepal, October 26-30 2014. [online] http://hdl.handle.net/1854/LU-5715330 (Accessed June 16, 2025).

Wilkinson, M. D., Dumontier, M., Aalbersberg, I., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J. W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., 't Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S. A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., and Mons, B. (2016) The FAIR guiding principles for scientific data management and stewardship. Scientific Data, **3**(1), 160018. DOI: 10.1038/sdata.2016.18.