ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/362328790

System for Automated Quality Control (Saqc) to Enable Traceable and Reproducible Data Streams in Environmental Science

Article *in* SSRN Electronic Journal · January 2022

DOI: 10.2139/ssrn.4173698

CITATIONS
0

READS
12

7 authors, including:

Lennart Schmidt
Helmholtz-Zentrum für Umweltforschung
3 PUBLICATIONS **40** CITATIONS
SEE PROFILE

Peter Lünenschloß
Helmholtz-Zentrum für Umweltforschung GmbH - UFZ
2 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Karsten Rinke
Helmholtz-Zentrum für Umweltforschung
116 PUBLICATIONS **2,666** CITATIONS
SEE PROFILE

Jan Bumberger
Helmholtz-Zentrum für Umweltforschung
47 PUBLICATIONS **528** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Junior research group 'Environmental-health Interactions in Cities' (GreenEquityHEALTH) – Challenges for Human Well-being under Global Changes' View project

MARSOL (Demonstrating Managed Aquifer Recharge as a Solution to Water Scarcity and Drought) View project

All content following this page was uploaded by Lennart Schmidt on 10 October 2022.

The user has requested enhancement of the downloaded file.

Highlights

System for automated Quality Control (SaQC) to enable traceable and reproducible data streams in environmental science

Lennart Schmidt,David Schäfer,Juliane Geller,Peter Lünenschloss,Bert Palm,Karsten Rinke,Jan Bumberger

- We present the software framework System for automated Quality Control (SaQC)
- SaQC facilitates the implementation of workflows for automated quality control
- It is designed for domain scientists that manage environmental sensor networks
- It is universal and extensible, yet user-friendly
- It tackles crucial challenges for quality control and FAIRness of data streams

Electronic copy available at: https://ssrn.com/abstract=4173698

System for automated Quality Control (SaQC) to enable traceable and reproducible data streams in environmental science

Lennart Schmidt[a,b,*], David Schäfer[a,b,**], Juliane Geller[a,b], Peter Lünenschloss[a,b], Bert Palm[a,b], Karsten Rinke[d] and Jan Bumberger[a,b,e]

[a]Research Data Management - RDM, Helmholtz Centre for Environmental Research GmbH - UFZ, Permoserstraße 15, Leipzig, 04318, Germany
[b]Department of Monitoring and Exploration Technologies, Helmholtz Centre for Environmental Research GmbH - UFZ, Permoserstraße 15, Leipzig, 04318, Germany
[d]Department of Lake Research, Helmholtz Centre for Environmental Research GmbH - UFZ, Brückstraße 3a, Magdeburg, 39114, Germany
[e]German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena- Leipzig, Puschstraße 4, Leipzig, 04103, Germany

ARTICLE INFO

Keywords:
data management
quality control
quality assurance
anomaly detection
sensor data
FAIR

ABSTRACT

Environmental sensor networks produce ever-growing volumes of raw data that need to be transformed into actionable data for monitoring of ongoing environmental changes and decision-support. The crucial challenge is the data provisioning in real-time which requires rigorous automation of quality control (QC) workflows using suitable software tools. We present the System for automated Quality Control (SaQC), a software framework for automated quality control of time series data that is universal and extensible in its set of domain-agnostic QC and processing functionalities, yet user-friendly in its low-code configuration environment. Two use cases present the configuration of basic and advanced quality control applications using SaQC. Also, we elaborate on the explicit user controls over the handling of quality flags and how SaQC can be used to make QC-workflows traceable and reproducible, thus promoting FAIR data streams of high quality.

1. Software and data availability

The System for Automated Quality Control (SaQC) is available both on the Python Package Index (PyPI)[1] and via its Git repository[2]. It is open source under the GNU General Public License, version 3. At the time of writing, SaQC supports Python 3.7 to 3.9. The computations in this publication have been executed with SaQC, version 2.1 and Python 3.9. Multiple tutorials in form of so-called Cookbooks that support the user to get started can be found in the SaQC online documentation [3]. Among these, there are two Cookbooks that present the two use cases of section 5 in detail, along with the necessary data and configuration files to reproduce the results. *Note: A persistent version of SaQC 2.1 along with these cookbooks, data and configuration files will be made available via Zenodo-doi upon final acceptance of the manuscript.*

2. Introduction

Climate change, land use change and human interactions are putting the worlds' ecosystems under significant pressure. To promote sustainable resource use, a holistic quantification of these impacts is needed. Thus, interdisciplinary and large-scale observation networks are required as a means to leverage environmental sensor data across domain and terrain boundaries (Reid et al., 2010). To that respect, large scale observatories like ILTER[4], eLTER[5], NEON[6], ACTRIS[7], ICOS[8] or TERENO[9] have been set-up, delivering crucial data for long-term monitoring of environmental systems, the deduction of process understanding as well as for the parametrization, calibration and validation of earth system models. Given their significance, observation networks are continuously expanding in amount of sensors and geographical coverage. So does the number of newly-developed environmental sensors available to monitor additional environmental variables, e.g. air quality parameters in citizen science applications (Collier-Oxandale et al., 2022). Consequently, the amount of environmental data that are being collected is steadily increasing, posing infrastructural challenges with respect to the data processing routines handling these data streams. Adding to this, processing is required to run in real time, immediately transforming raw data into actionable data products or model results to support timely decision-making, e.g. for natural hazard management. The only way to enable this real time provisioning of ever-growing data volumes is rigorous automation (Sturtevant

*Corresponding author
**Corresponding software maintainer
lennart.schmidt@ufz.de (L. Schmidt); david.schaefer@ufz.de (D. Schäfer)
ORCID(s): 0000-0003-3270-8524 (L. Schmidt); 0000-0003-4517-6459 (D. Schäfer); 0000-0001-9602-1997 (P. Lünenschloss); 0000-0001-5106-9057 (B. Palm); 0000-0003-0864-6722 (K. Rinke); 0000-0003-3780-8663 (J. Bumberger)

[1]https://pypi.org/project/saqc/
[2]https://git.ufz.de/rdm-software/SaQC
[3]https://rdm-software.pages.ufz.de/saqc/index.html

[4]International Long-Term Ecosystem Research Network (Mirtl et al., 2018)
[5]European Long-Term Ecosystem Research Network (Mollenhauer et al., 2018)
[6]National Ecological Observatory Network (Loescher et al., 2017)
[7]Aerosols, Clouds and Trace gases Research Infrastructure Network (ACTRIS, 2021)
[8]Integrated Carbon Observation System (Heiskanen et al., 2021)
[9]Terrestrial Environmental Observatories Network (Zacharias et al., 2011)

Electronic copy available at: https://ssrn.com/abstract=4173698

et al., 2021), thus building on the FAIR principles to provide Findable, Accessible, Interoperable and Reusable data in machine-readable form (GoFair, 2021). Adding to this, it is essential to establish automated data pipelines that address one major challenge related to sensor data: The process of quality control (QC), i.e. the separation of erroneous from usable data based on expert knowledge (Koedel et al., 2022; Crystal-Ornelas et al., 2021). The use of data that has not undergone thorough quality control routines can lead to inaccurate model forecasts and inadequate decision-making, among others (Doraiswamy et al., 2000). Errors in environmental data can arise from multiple sources, among these are the suboptimal calibration or malfunctioning of sensors, technical failures during data recording and transmission or due to the influence of environmental conditions (e.g. sensor fouling, obstruction, freezing etc.) (Gandin, 1988; Wagner et al., 2006a). Ideally, the majority of these error sources are prevented by data pipeline design (*quality assurance*). In application, however, post-processing steps to separate erroneous from usable data are commonly required (*quality control*) (Campbell et al., 2013). Performing these manually, i.e. by visual inspection of the data by domain experts, is laborious, introduces subjective bias, requires expert-knowledge that is neither reproducible nor transferable and is simply unattainable for real-time provisioning (Fiebrich et al., 2010; Jones et al., 2018). Fully automated QC-workflows, on the other hand, are objective, reproducible and offer efficient handling of large data volumes while allowing for unlimited test specifications that can be adapted over time (WMO, 2018).

Figure 1 is a schematic representation of such an automated QC-workflow (left to right): *Raw data* flow from the sensor network to the database, run through QC back into the database as *QC-ed data*, which can then be published as a data set and be used for further analysis. The QC-part (dashed blue box) generally consists of QC-tests that are run on raw data in order to identify erroneous data points (e.g. outlier detection algorithms) and assign a so-called flag, e.g. "Bad", "Suspicious" or "Good" to each data point as a measure of its quality. These are then sent back into the database, along with the QC-ed data. In this process, raw data is always kept to allow for a potential reprocessing (Campbell et al., 2013).

There are documented set-ups of automated QC-pipelines in the environmental community, mostly by large and renown research infrastructures like NEON (Taylor and Loescher, 2012), ICOS (Vitale et al., 2020; Pastorello et al., 2020; Yver-Kwok et al., 2021), GHCN[10] (Durre et al., 2010) or IAGOS[11] (Petzold et al., 2015). Commonly, these infrastructures develop QA/QC protocols that are to be fulfilled by the members of the respective network. In many cases, the implementation of these protocols lies with the members who might be domain-experts with limited technical expertise, posing a challenge that can result in disadvantages with respect to data quality and standardization, requiring additional harmonization steps at the network level (Sturtevant et al., 2021). Similarly, small observation networks and single research institutes are challenged by the software implementation of their QC-workflows. To our knowledge, there are no publications that illustrate both the algorithmical set-up and the software implementation of such protocol to assist the domain-experts.

Thus, custom implementations of automated QC-workflows commonly exhibit the following shortcomings: Often, the implementations are use case specific and hard-coded, making them inflexible to new functionalities and changing requirements such as altered data pre-processing steps or input data structures. Also, aside from large networks, there are no standardized flagging schemata, these are usually defined as best-suited for the respective use case, inhibiting interoperability. Quality flags might be altered or overwritten implicitly, i.e. as an unnoticed result of data processing steps, or without further notice in the final dataset. Also, current implementations do not ensure the entire workflow to be versioned and reproducible. This might lead to different versions of QC-ed datasets that are published over time without specific notice in the metadata or without the possibility to reproduce previous versions, if needed. Additionally, current implementations pose a challenge to the users, commonly domain experts: finding a robust choice and parametrization of quality tests that identifies as many suspicious values as possible without removing valid data is usually a time-consuming endeavor, thus requiring an intuitive and efficient user interface. Current implementations, however, typically require programming knowledge in order to parametrize or make changes to the QC-tests, which can not be presumed to be available.

For a software framework to facilitate the process of setting up a QC-workflow for any kind of (environmental) sensor networks, we hereby define the following software requirements. A QC software framework should be:

- **Universal:** flexible input/output data structures and flagging schemata
- **Equipped:** provide a comprehensive set of generic pre-/post-processing methods and domain-agnostic QC-tests
- **Extensible:** enable integration of custom processing methods and QC-tests
- **Flag-centric**: accommodate explicit, user-defined flag handling inside data processing steps
- **Traceable:** keep provenance of processing and flagging operations along the pipeline
- **Reproducible:** enable versioning of QC-workflows
- **Accessible:** configuration using either a graphical or a low-code user interface
- **Open Source:** available to anyone free-of-charge

[10] Global Historical Climatology Network
[11] In-service Aircraft for a Global Observing System

Electronic copy available at: https://ssrn.com/abstract=4173698

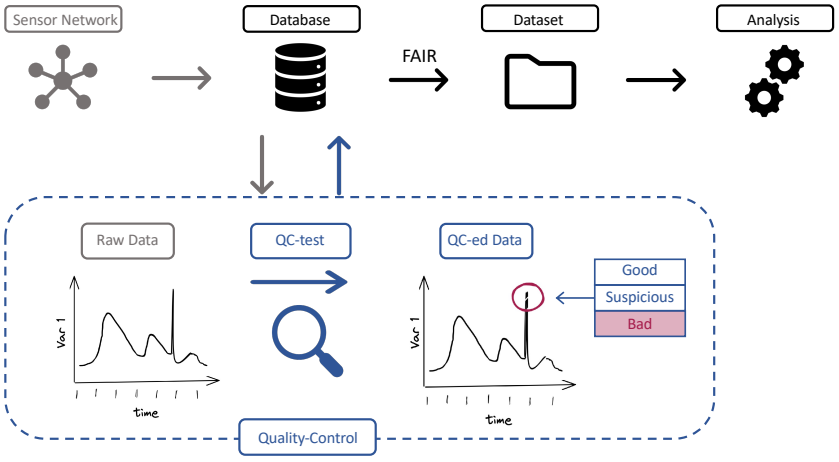

Figure 1: Illustration of a generalized data flow from sensor to analysis including automated quality control: *Raw data* flows from the sensors into the database and on to the QC-test. Here, a spike in the data (red circle) is identified as erroneous and flagged as "Bad". The resulting *QC-ed data*, along with a quality flag for each data point, is sent back into the database, from where it can be published as a dataset and later be used for analysis.

In the environmental community, there are about ten software tools available to assist the domain experts in setting up their QC-workflows. However, to our knowledge, none of the available frameworks meet all of the above requirements - see section 3.2 for a detailed screening.
For this reason, we present the newly developed **System for Automated Quality Control (SaQC)** that aims at facilitating the implementation of standardized QC-workflows for any environmental time series dataset, being user-friendly and flexible in all relevant interfaces while accounting for traceability and reproducibility. While all above software requirements are met, a unique focus of SaQC are its sophisticated mechanisms for the handling of quality flags. Following an overview of the current state of automated quality control in chapter 3, including a screening of available software tools, this paper provides a detailed software description of SaQC and its features in section 4. Next, two use case examples in section 5 provide an in-depth explanation of how SaQC is employed in practical application. These use cases present some of its basic and advanced functionalities and the results of the QC. Following this, we discuss current limitations of the software as well as future development and research directions.

3. Current state of automated quality control

Generally, current scientific literature on automated quality control in the environmental sciences is fragmented, characterized by many domain-specific applications that range from single-site to observatory network scale. To our knowledge, publications either present newly-developed QC-tests and set-ups of QC-pipelines, or they introduce software frameworks, never both in a joint manner.

3.1. Related literature

The topic of automated quality control has been promoted primarily by the meteorological community with large observation networks like GHCN, IAGOS etc. Consequently, the World Meteorological Organisation (WMO) has been providing the community with extensive and detailed baseline guidelines on QC-tests, along with parametrization and flagging guidelines from 1971 on (WMO, 2003, 2018, e.g.). Based on this, several national services have published similar guidelines for the operators of their observation networks, e.g. Wagner et al. (2006b).

While a multitude of deterministic and statistical QC-tests are available, there is no standard set-up of QC-tests to be executed for an arbitrary environmental dataset. However, several (domain-specific) publications are available that present the entire QC-workflow of known observatories and thus serve as orientation for researchers aiming at setting up their own QC-workflow. For instance, Durre et al. (2010), Estévez et al. (2011) and Fiebrich et al. (2010) present the set-ups of the NEON, the RIAA[12] and the Oklahoma Mesonet networks, respectively.

Within the meteorological community, the Eddy-Covariance and solar radiance communities have traditionally been faced with particularly challenging QC tasks, leading to the development of physics-based QC routines (e.g. Younes et al., 2005; Metzger et al., 2013; Mauder et al., 2013; Pastorello et al., 2020; Vitale et al., 2020). Next to meteorology, the oceanographic community has invested considerable efforts in setting up operational QC-systems that build on

[12] Agroclimatic Information Network of Andalusia

Electronic copy available at: https://ssrn.com/abstract=4173698

consistency checks to tackle high spatial variance in the measurements (e.g. Ingleby and Huddleston, 2007; Gourrion et al., 2020). Another challenging task is the QC of in-situ soil moisture data. Dorigo et al. (2013) and Liao et al. (2019) studied several spectrum-based approaches to detect error patterns specific to this application.

As the parametrization of QC-tests for a specific use case is a time-consuming task, statistical methods to automate this process have been developed (Taylor and Loescher, 2013; Gourrion et al., 2020). Moving beyond QC-tests at sensor level, several frameworks to check internal, temporal and spatial consistency have been presented (e.g. You et al., 2007; Durre et al., 2010) and subjected to sensitivity analysis (Hubbard and You, 2005). In order for a QC-workflow to assure good "quality of quality control", frequent auditing of QC-routines and its results is necessary to avoid introducing systematic bias or overflagging (Taylor and Loescher, 2013; Sturtevant et al., 2021). To this end, Durre et al. (2008) present a framework of systematic manual inspection and pattern analysis of the flags that result from a QC-workflow. Information on the performance of a QC-workflow including potential errors that may remain should be communicated to the user. For this purpose, Smith et al. (2014) propose a modular and flexible framework to aggregate flagging information at the level of both the sensor as well as the whole data product.
Generally, there is little standardization regarding implementation details of QC-workflows among networks or domains. There are only few guidelines regarding choice and order of QC-tests, flagging schemata or handling of flags that are agnostic as to the network or domain of application (see Campbell et al., 2013; Gouldman et al., 2017).

3.2. Available software tools

In order to accommodate the heterogeneous designs and requirements of QC-workflows, domain-agnostic, flexible and extensible software frameworks are needed. Many existing frameworks in the environmental sciences are, however, domain-specific and thus either limited with respect to the data sources that can be used or regarding the available set of processing functionalities and QC-tests. An example of this is the the R-package AirSensor (Collier-Oxandale et al., 2022) that includes basic QC and consistency tests along with time series processing and visualization functions. These are, however, taylored to the specific application of low-cost air quality sensors in the community-based "PurpleAir" network. Quite similar, Metzger et al. (2017) present the R-package eddy4R specifically for processing eddy-covariance data, including QC functionalities following Taylor and Loescher (2013) and Smith et al. (2014). For the solar irradiance community, the BSRN-Toolbox (Schmithüsen et al., 2019) includes all domain-relevant QC tests as proposed by Long and Dutton (2010). Going further, Urraca et al. (2020) present a free web-service of their "bias based" quality control method (BQC) to find low magnitude errors in data that are usually not captured by classical QC-methods. Addressing the current surge of machine learning in environmental sciences, Jones et al. (2022) provide multiple algorithms for quality control of aquatic sensor data via the Python package pyhydroqc.

As to frameworks that are domain-agnostic or flexible enough to be used for any sensor data there are only five software tools available and still maintained, at least to our knowledge. Of these, none meet all the requirements towards such a framework as formulated in chapter 2. For the three most suitable tools, capabilities are summarized in table 1. Horsburgh et al. (2015) present ODM Tools Python, a Python-based tool that enables users to query and export, visualize, and edit time series observations stored in, and thus limited to, an Observations Data Model (ODM) database. The main focus of the tool is to provide a GUI for manual QC, where provenance of all manual edits is kept as each step is recorded in a Python-script to keep the process traceable and reproducible. However, the tool is restricted in essential functionalities for setting up entirely automated QC workflows like data processing methods, flag handling during processing and regarding its set of available QC-tests. Another tool that is domain-agnostic and freely-available is the Great Expectations Framework (Gong and Campbell, 2022). It is available as a Python library and provides the user with a multitude of so-called "expectations", i.e. tests to check the validity of a given batch of data or an entire dataset. Due to an active user community, the range of available tests is large, ranging from simple checks to statistical methods. However, the aim of the framework is different from SaQC: Instead of identifying single erroneous values and assigning a single flag, the aim is "dataset validation", i.e. to validate the integrity of a whole dataset by checking its columns or entire data blocks for certain conditions. Consequently, it does not provide mechanisms to assign and handle flags at the data point level. Thus, its use for QC of sensor networks as envisioned in this paper is limited. There are two frameworks with a similar focus that are thus not included in table 1: The equivalent in the statistical programming language R is the assertR-package (Fischetti, 2022). Similarly, the tensorflow data validation framework (Caveness et al., 2020) provides the user with functionality for dataset validation in a machine learning setting.
Adding to the above, the GCE Data Toolbox (Sheldon, 2008) provides a vast array of functionalities to set up QC-workflows through a GUI, a CLI and a programming API (Matlab). It provides a set of processing functionalities, integration of custom functionalities and it ensures traceability and reproducibility - Thus, meeting most of the software requirements as formulated. However, the included set of QC-tests is limited and, using Matlab as the language of implementation, it can not be considered as open source software.

Proprietary solutions typically also provide the means to perform manual and automated QC, but only as part of a whole data management and alert system. Thus, the

Electronic copy available at: https://ssrn.com/abstract=4173698

Table 1
Capabilities of three domain-agnostic software frameworks for automated QC, based on the software requirements as formulated in chapter 2.

		Software Framework		
Aim	**Requirement**	ODM-Tools	Great Expectations	GCE Toolbox
Universal	Flexible I/O structure and flagging schemata	-	✓	✓
Equipped	Comprehensive set of processing methods and QC-tests included	-	✓	-
Extensible	Integration of custom processing methods and QC-tests	-	✓	✓
Flag-centric	Explicit flag handling during processing	-	-	-
Traceable	Metadata Enrichment	✓	-	✓
Reproducible	Versioning of QC-pipelines	✓	-	✓
Accessible	GUI or low-code configuration	✓	-	✓
Open Source	Entirely Open source	✓	✓	-
Note		Focus on manual QC	Validation at dataset level	Based on Matlab
Last Release		06/2017	05/2022	03/2019
Reference		Horsburgh et al. (2015)	Gong and Campbell (2022)	Sheldon (2008)

implementation inside custom infrastructures can be cumbersome. Additionally, these data management systems are typically designed for a specific domain, e.g. Kisters WISKI or Aquarius for the water sector. Other proprietary software frameworks are bound to specific hardware, raising the effort of integration of hardware from other manufacturers.

4. Materials and Methods

4.1. Software description

SaQC is a free and Open Source Software framework that aims to facilitate the implementation of standardized QC-workflows for any (environmental) time series dataset. It specifically addresses domain-scientists and data managers with limited IT expertise. In doing so, it aims to foster community-wide FAIR data streams by providing all essential functionalities to set up such workflows. SaQC can be controlled either via a command line application, a web-based Configuration App or a Python API that all address the exploratory nature of quality control by offering a continuously growing number of quality check routines through a flexible and simple configuration system. Adding to its QC functionalities, it provides extensive routines for data pre- and post-processing as well as handling and translation of quality flags. Thus, SaQC can be used as a general data processing tool to turn raw sensor data into secondary data products of controlled quality. Below its user interface, SaQC is highly customizable and extensible thanks to its modular structure with well-defined interfaces. The primary development goal was to meet the software requirements as formulated in chapter 2. Here, a particular focus was set to make SaQC flag-centric, i.e. to integrate explicit control over flag handling throughout the entire workflow - Making it unique in its versatility to meet complex requirements that can arise in practical application. Adding to this, the code base was developed such that automation of SaQC as part of an entire data pipeline from sensor to data product is as simple as possible: The parametrization of QC-tests is technically separated from the rest of the code so that the configuration of QC-workflows can be edited without changing the operational core code.

Based on the generalized data flow from sensor to analysis as shown in figure 1, figure 2 is a detailed presentation of the QC-component of such a data flow when using SaQC. Below, this exemplary SaQC-based QC-workflow is explained in more detail and in chronological order, with its elements corresponding to the numbers 1) - 9) as indicated in the figure. This way, the components of SaQC and their contribution to meet the *requirements of QC software* (ch. 2)

Electronic copy available at: https://ssrn.com/abstract=4173698

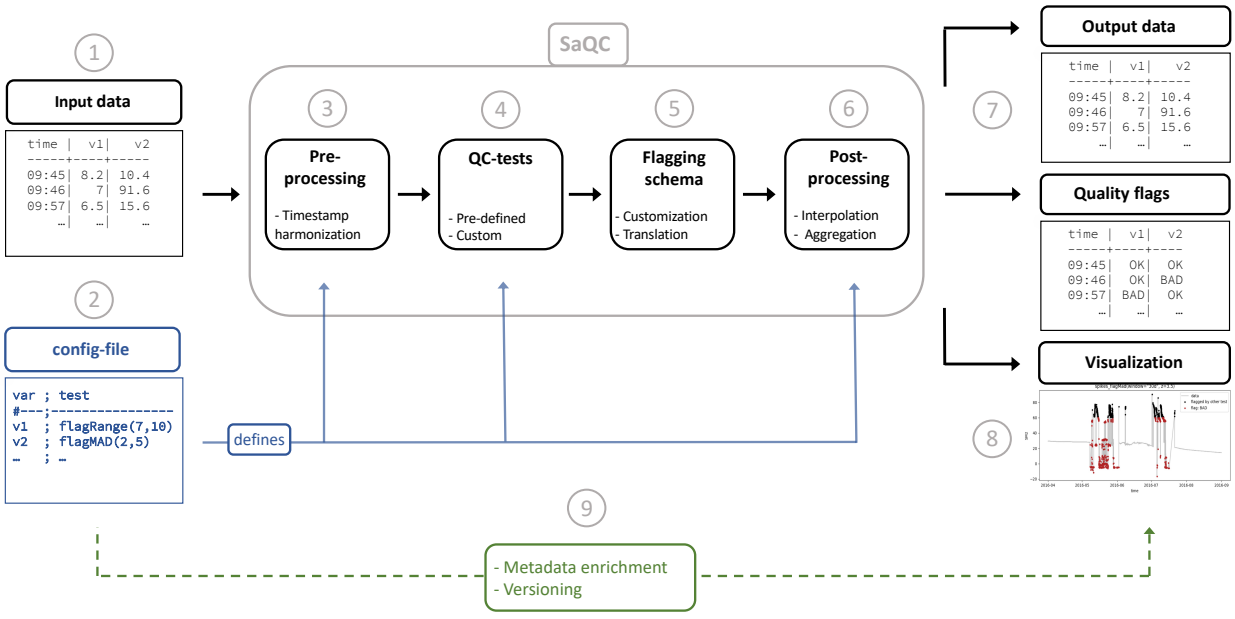

Figure 2: General QC-workflow using SaQC: Input data and a config-file are passed to SaQC which returns quality-controlled output data, along with quality flags and visualizations of the QC-results. The colors in the figure discriminate four categories: data flows (black), configuration (blue), metadata flows (green) and annotations (grey). The numbers correspond to the workflow elements as described in more detail in the text below.

are explained in more detail.

(1) Input data can be supplied as .csv or .parquet-files when using SaQC as a command line application, in any tabular structure when using it in Python, directly.

(2) Using a text-based configuration scheme that requires no knowledge of programming, users can then parametrize existing processing routines as well as QC-tests inside a *config-file*. Also, users can write custom routines and tests using a simplified, Python-like syntax. A detailed presentation of how to set up a config-file can be found in section 4.3

(3) The config-file also defines the pre-processing routines that might have to be executed prior to the QC-test. Both pre-processing- and QC-routines are tightly coupled as most QC-tests require data pre-processing such as timestamp harmonization.

(4) The selection of pre-defined QC-tests aims at being applicable across domains and ranges from standard tests such as constant value or spike detection to more advanced methods like pattern recognition and multivariate tests. More information on both processing and QC-test functionalities can be found in section 4.2. Users can also implement entirely new QC-test functionalities in the config-file or in the core Python code.

(5) Depending on the result of the tests, each data point gets assigned a flag following a user-defined flagging scheme that defines the number and hierarchy of the flags, e.g. *Good*, *Suspicious* and *Bad*. SaQC allows entirely user-defined flagging schemes and provides functionality to translate these between each other. See chapter 4.4 for details on the handling of flags.

(6) If desired, post-processing routines like interpolation or aggregation over time can be applied on the newly quality-controlled data.

(7) By default, the processed data, along with the assigned flags, is then returned as .csv or .parquet-files (command line application) or any other user-defined format (Python).

(8) Adding to the file output, SaQC provides various visualization functionalities that allow the user to investigate the effect of different test parametrizations on the QC-results.

(9) Across the whole workflow, SaQC ensures traceability and reproducibility by metadata enrichment and versioning of pipelines (see section 4.5 for details).

For documentation and tutorials, readers can refer to the online documentation of SaQC[13] which also contains the Getting Started Guide[14].

[13] https://rdm-software.pages.ufz.de/saqc/index.html

[14] https://rdm-software.pages.ufz.de/saqc/getting_started/TutorialCLI.html

Electronic copy available at: https://ssrn.com/abstract=4173698

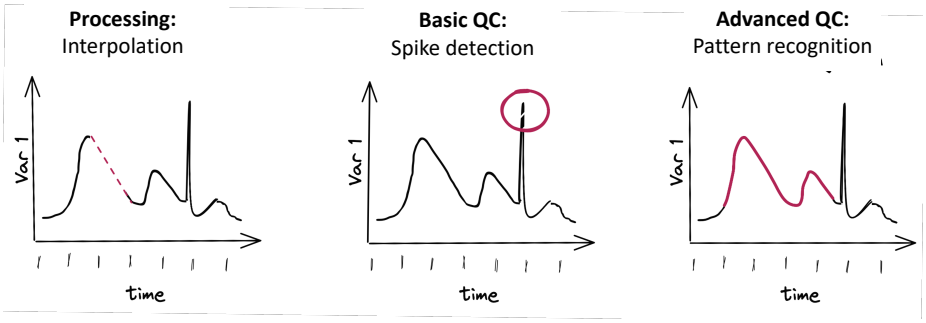

Figure 3: Schematic visualization the three usage categories of exemplary 1) processing, 2) basic QC-test, and 3) advanced QC-test (from left to right).

Table 2
The three usage categories as included in SaQC. Each category consists of functional groups which encompass a certain number (#) of functions that serve the same objective.

Group	Objective	#
Processing	*Processing steps prior to (pre-processing) or after QC-testing (post-processing)*	
Resampling	Aligning data to equi-distant timestamps by shifting, resampling or interpolation	3
Smoothing	Curve smoothing using parametric and non-parametric methods	2
Transformation	Derive new variables by transformation or using custom functions	2
Flag projection	Project flags from resampled data back onto original data	1
Basic QC	*Low algorithmic complexity and parametrization effort*	
Constants	Deterministic and variance-based detection of undesired stationary behaviour	2
Breaks	Detection of missing/isolated values or jumps in the data	2
Outliers	Detection of outliers and spikes, both deterministic and statistical methods	6
Manual flagging	Integration of precedent manual QC by experts from auxiliary files	1
Advanced QC	*Higher algorithmic complexity and parametrization effort*	
Noise	Separate noise from true signal using low pass filters	1
Changepoints	Detection of points of undesired system state transitions	2
Drift	Detection and correction of sensor drift based on deviation from reference system state	3
Pattern recognition	Detection of undesired patterns in the data based on Dynamic time Warping and Wavelets	2
Custom functions	Combination of existing/integration of custom QC-tests inside config-file or source code	1
Machine learning	Training of machine learning models for flagging and data imputation	4

4.2. Function overview

SaQC ships with extensive processing capabilities as well as various domain-independent QC-tests, giving a total of 34 primary functions that are continuously being extended. Figure 3 aims at giving an intuitive representation of how these functions are further categorized into **1) processing** which serves as either the preliminary (pre-processing) or finalizing (post-processing) step to both **2) basic QC-tests** and **3) advanced QC-tests**. In SaQC, each of these three categories is populated by multiple functional groups, which in turn encompass a number (#) of functions that serve the same objective. Table 2 provides a list and description of these functional groups, objectives and the respective number of functions. A more detailed table including the respective function names can be found in the appendix A.1. For an in-depth description of each of the functions, readers are referred to the software documentation.

Furthermore, the following facts should be kept in mind with respect to the use of the functions of SaQC:

- custom functions can be implemented either right in the config-file (simplified syntax, logical operators) or in the source code (Python API)
- The modular architecture of SaQC is designed to accommodate custom processing and QC functions of arbitrary complexity via predefined interfaces using the Python API
- processing functions can be run independently, i.e. without antecedent or subsequent QC, if desired

4.3. Configuring SaQC for use

As **command line application**, SaQC is controlled by a config-file (.csv) listing the variables of the dataset as well as both the processing routines and QC-tests along with respective parameters that are to be executed. The content of such a configuration could look like this:

```
varname     ; test
------------;-----------------------------------
SM2         ; shift(freq="15Min")
SM2         ; flagRange(min=10, max=60)
SM2         ; flagMAD(window="30d", z=3.5)
SM2         ; interpolateInvalid(method='linear')
```

Here, a timestamp shift to a regular interval of 15min is performed during pre-processing. This is followed by a range test that flags all values outside the range of [10,60] and a spike test based on median absolute deviation (MAD,

Electronic copy available at: https://ssrn.com/abstract=4173698

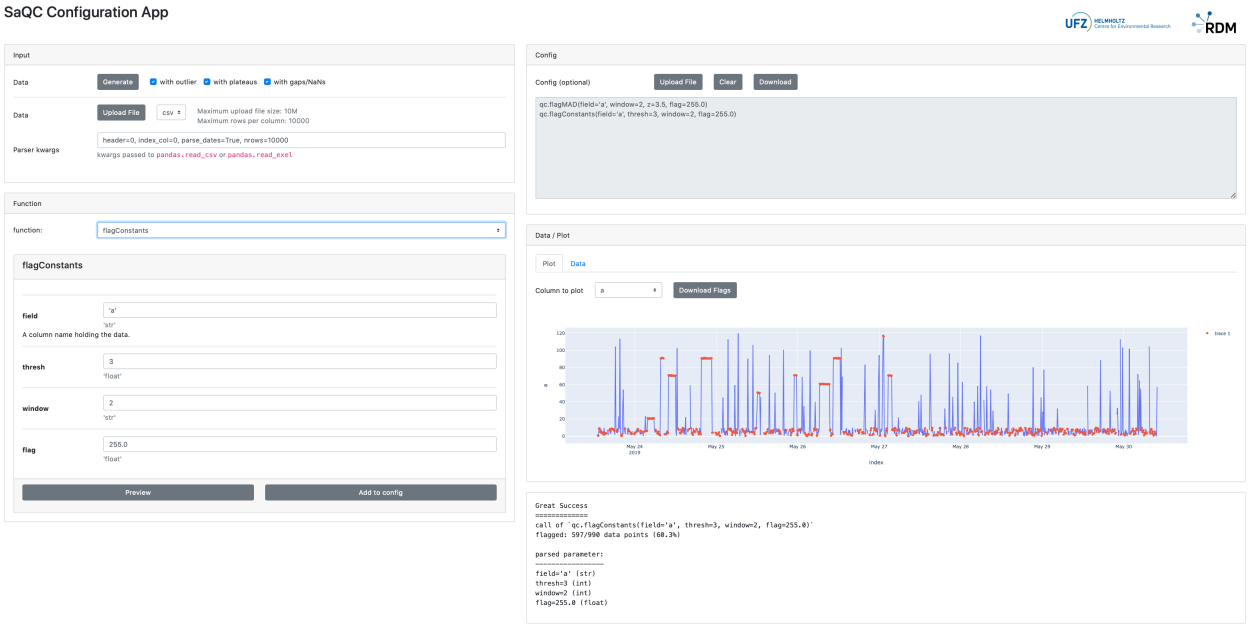

Figure 4: Graphical User Interface of the SaQC Configuration App.

see Iglewicz and Hoaglin, 1993). Lastly, data gaps are filled using linear interpolation. In fact, this is the config-file as used for use case 1 in section 5.1.
As soon as the basic inputs, a dataset and the config-file are prepared, running SaQC is as simple as:

```
saqc --config path_to_configuration.csv --data path_to_data.csv
```

where `path_to_configuration.csv` is the spaceholder for the configuration file described above, and `path_to_data.csv` acts as the spaceholder for the input dataset.
The same set-up can be achieved directly in a Python script by using SaQC's **Python API**, which is explained in more detail in the software documentation.

For users without a programming background, there is an alternative way to derive a suitable config-file using the web-based **SaQC Configuration App**[15] (see fig. 4). It offers a graphical user interface for users to upload their own data, parametrize the available preprocessing and QC-test functions and visualize the results. At the end, users can obtain the respective config-file to use in their own QC-pipeline.

4.4. Flag Handling inside SaQC

One of the software requirements as stated in chapter 2 is that a software tool should be *flag-centric*, i.e. all processing and QC-functionalities should accommodate explicit, user-defined flag handling. During the development of SaQC, considerable effort was invested into the handling and integration of quality flags into the QC workflow: From the support of custom flagging schemes and flag translation, their internal representation, the inherent rules for the transfer of flags from one processing step to the next one to a holistic output of the results. When chaining multiple processing steps and QC tests in practical application, flag handling raises challenges that can become quite complex. By means of five key challenges, the following section illustrates the potential complexity and how a flag-centric SaQC is equipped to tackle it.

Filtering of flags between QC-tests: Some QC-tests might require the data to be filtered according to the flags set by preceding QC-tests. For example, running a sparsity test on the sensor data $SM2$ in order to flag data points that are surrounded by erroneous data only, requires preceding QC tests to identify and flag these erroneous data. In SaQC, a user can control which data a QC-test should receive by setting a threshold for the flag level, $dfilter$, that must not be exceeded for data to be passed on to the respective test. This way, all data that was flagged as "BAD" by preceding tests, in this case by a range test, can be excluded when executing the sparsity test ($flagIsolated$):

```
varname     ; test
------------;-----------------------------------
SM2         ; flagRange(min=10, max=60)
SM2         ; flagIsolated(dfilter=BAD)
```

Conditioning of QC-tests based on previous flags: The execution of one QC-test might have to be conditioned on the results of the preceding QC-tests. In the exemplary configuration below, sensor data $SM2$ might be affected either by a drop in sensor battery voltage $BattV$ or due to maintenance work on the power supply. Thus, both a range test and manual flagging (based on an auxiliary file) are executed on $BattV$. Next, a custom function $flagGeneric$ is conditioned on the two preceding tests by making use of

[15] https://webapp.ufz.de/saqc-config-app/

Electronic copy available at: https://ssrn.com/abstract=4173698

Table 3
Illustration of an exemplary flagging history for sensor data $SM2$ after running three QC-tests. Traceability of each flag to the respective QC-test of origin is ensured by storing flags column-wise.

time	SM2	flagRange	flagMAD	flagByStray
2016-04-01 00:04:48	32.6	BAD	GOOD	GOOD
2016-04-01 00:20:42	32.7	GOOD	GOOD	GOOD
2016-04-01 00:36:05	44.8	GOOD	BAD	BAD
...				

the option to chain expressions via boolean operators (here: *or*): If one or both of the tests assigned a flag to $BattV$, the sensor data $SM2$ is flagged accordingly.

```
varname ; test
--------;---------------------------------------
BattV   ; flagRange(min=.5, label='voltage drop')
BattV   ; flagManual(mdata='maintData', label='maintenance')
SM2     ; flagGeneric(func=isFlagged(BattV, 'voltage drop')
          or isflagged(BattV, 'maintenance'))
```

Provenance of the QC-test for each flag: For each flag, information on the QC-test that produced it has to be traceable after the QC-process is finished. In SaQC, this is achieved by storing and returning the entire flagging history, i.e. the results of all QC-tests, for each data point (see table 3). This also allows a final aggregation of all assigned flags into a probabilistic metric of data quality as presented in e.g. Kaffashzadeh et al. (2019).

Keeping track of flags during resampling: For example, when aggregating a dataset of 5-minute temporal resolution to one of 15-minute resolution, three data points are reduced into one mean value. As to the flags that were already assigned, a decision has to be taken which flag should be assigned to that single value. SaQC allows the user to define a flag aggregation function $flag_{f}unc$. In this case, it takes the the worst flag of the three (see code below). Internally, the original data structure is retained, so that the flagging results on aggregated data can be projected back onto the original data.

```
varname     ; test
------------;-------------------------------------
SM2         ; resample(freq='15min', func=mean, flag_func=max)
```

Translation of flagging schemes: Datasets that exhibit different flagging schemes, originating from separate previous QC processes, might have to be joined for being processed by SaQC. Also, the output of the actual QC-workflow might need to match yet another flagging scheme. SaQC enables the translation between different input and output flagging schemes by representing any data points' flags history in a series of floats (not shown). This representation is sufficiently manifold in its structure to capture the majority of thinkable flagging schemata, ranging from simple, hierarchical flagging levels to more complex, cause-sensitive or probabilistic ones.

4.5. Enabling traceable and reproducible workflows

The FAIR-principles itself do not explicitly include quality control as a prerequisite (Koedel et al., 2022): "The FAIR guiding principles request that optimal care is taken to enable users to determine the 'usefulness' (for their purpose) of the data and other research objects they find, which includes rich, machine readable provenance" (GoFair, 2021). In other words, simply noting the absence of quality control in the metadata of a dataset would suffice to make a dataset FAIR. However, data does only actually become re-usable by going beyond that, i.e. by establishing traceable QC-workflows. This, in turn, requires more attention if FAIR guidelines are still to be met: R1.2., i.e. part of the definition of Reusability, states that "datasets are to be associated with detailed provenance (Wilkinson et al., 2016)". In the case of QC-workflows using SaQC, this is achieved by extensive metadata enrichment along the QA/QC-pipeline: Processing steps, QC-tests and their respective parametrization are accessible for every single quality flag produced by SaQC (*traceability*).
The ultimate goal of such successful metadata provenance is that "[datasets] can be replicated and/or combined in different settings" (GoFair, 2021). SaQC supports versioning of the entire pipeline including all parametrizations through version control systems (e.g. Git). As such, every single quality flag is not only traceable, but can be replicated if necessary (*reproducibility*).
The following figure 5 is a variation of the exemplary SaQC-workflow as previously depicted in figure 2 but with a focus on the metadata component (green colour). Figure 5 illustrates in detail how metadata enrichment and versioning can be achieved in SaQC-based workflows: The input data comes with source metadata, e.g. sensor location, ID or data author. As it runs through the SaQC-workflow, and as such through the QC-tests as defined in the config-file, metadata is being collected and merged with the source metadata. For instance, in the case of a data point identified as a spike (red circle in fig. 5), this includes the quality flag as well as information on the flagging schema it refers to. Also, the QC-test that assigned the flag (here: spike-test) is written, which becomes relevant when multiple QC-tests are executed on the same variable. To account for varying implementations across software versions, the version of SaQC is noted as well. In order to keep track of the version of the entire workflow, the config-file is versioned via Git and the respective Git Commit Hash can be written into the metadata. With this, a user receiving any dataset that was QC-ed using SaQC is able to **1) trace** and **2) reproduce** every single step of the pipeline.

5. Exemplary use cases

This section presents two use cases of SaQC in actual applications at the Helmholtz-Centre for Environmental Research (UFZ), more specifically at two observatories that are part of the TERENO Bode Hydrological Observatory (Wollschläger et al., 2017) which is located in the Harz

Electronic copy available at: https://ssrn.com/abstract=4173698

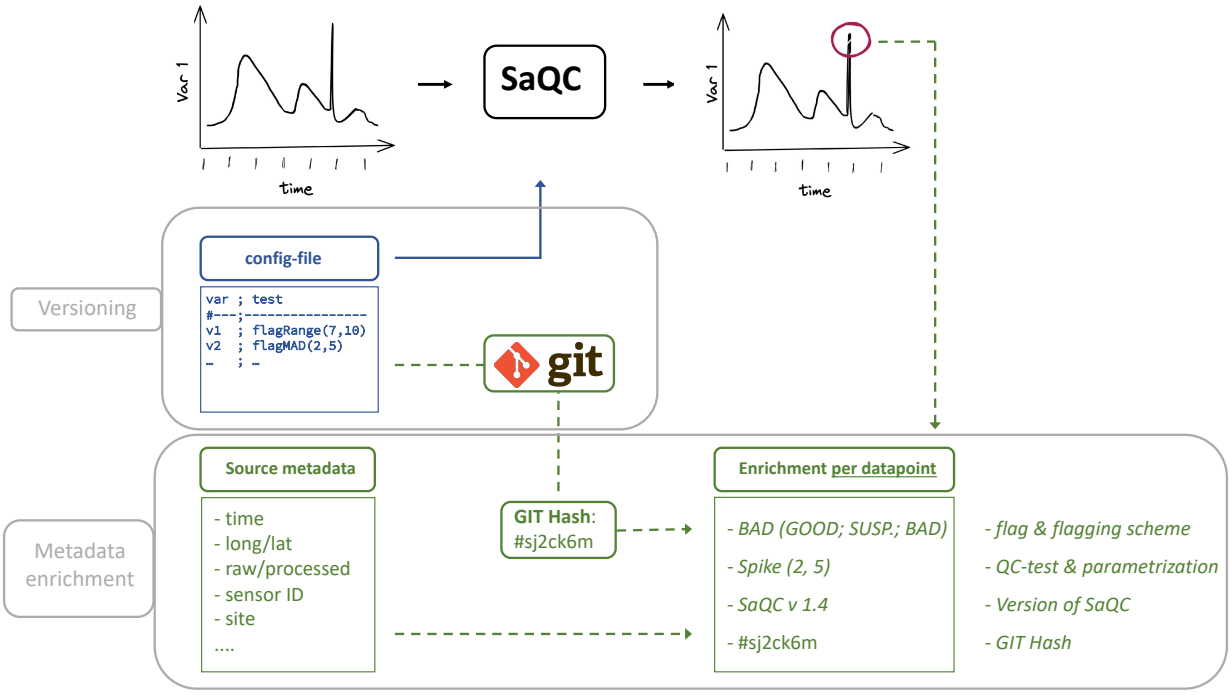

Figure 5: Schematic illustration of metadata enrichment (traceability) and versioning (reproducibility) when using SaQC. The red circle marks a data point identified as a spike by a QC-test. The other colors discriminate four categories: Data flows (black), configuration (blue), metadata flows (green) and annotations (grey)

region in central Germany. The first use case at the forest ecosystem observatory Hohes Holz (Rebmann et al., 2017) employs some of the functionalities listed as *Processing* and *Basic QC-test*. The second one at the hydrological Rappbode-observatory (Rinke et al., 2013) presents the use of *Processing* functionalities and both *Basic* and *Advanced QC-tests*. Both use cases are presented in more detail in our collection of Cookbooks in the online documentation[16], along with the necessary data and config-files to reproduce all results as presented here. For both use cases, the time series snippets that are shown were selected to be intuitive and illustrative with the aim of showcasing the functions of SaQC. The authors do not intend to evaluate the performance of the developed QC-workflows in detail, therefore the results are only discussed briefly.

5.1. Use case 1: Basic quality control of soil moisture data

At the Hohes Holz observatory (Central Germany), a patch of mixed beech forest, a huge set of environmental variables is measured continuously to improve the scientific understanding of carbon and water fluxes between the ecosystem and the atmosphere under the influence of environmental changes (see fig. 6). Various meteorological, hydrological and ecological variables are observed at high spatial and temporal resolution, among these distributed soil moisture content measurements using about 180 single sensors (sensor model: SMT100, TRUEBNER GmbH, Neustadt, Germany). In this use case, quality control of

Figure 6: Left: Birds-eye view of the forest ecosystem observatory Hohes Holz. Right: Measurement set-up of multiple soil moisture sensors at different soil depths as employed in use case 1. Image source: UFZ.

soil moisture data is performed in four steps. The dataset contains the variables *time*, *battery* (battery voltage of the data acquisition platine) and soil moisture *SM2* in vol.% at one sensor. The following description handles line by line of the actual config-file from section 4.3, also included in appendix A.2, along with illustrative figures. An in-depth presentation of the workflow, including all figures, algorithmic details and references, can be found in the respective Cookbook [17]. Figure 7 depicts the raw data of sensor *SM2* before quality control, exhibiting unrealistic values beyond a plausible value range.

[16] https://rdm-software.pages.ufz.de/saqc/index.html

[17] https://rdm-software.pages.ufz.de/saqc/getting_started/TutorialCLI.html

Electronic copy available at: https://ssrn.com/abstract=4173698

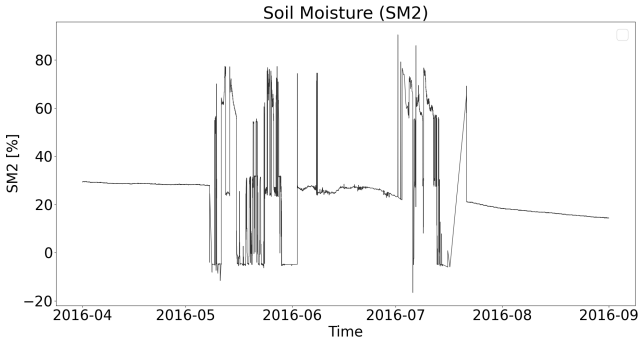

Figure 7: Raw soil moisture data of sensor *SM2* from a distributed soil moisture sensor network at the research site 'Hohes Holz'.

Table 4
Exemplary screenshot of the raw soil moisture data with irregular timestamps.

time	battery	SM2
2016-04-01 00:04:48	3573	32.6
2016-04-01 00:20:42	3572	32.7
2016-04-01 00:36:05	3572	32.6
...		

Table 5
Exemplary screenshot of the pre-processed soil moisture data after step 1, now with regular timestamps.

time	battery	SM2
2016-04-01 00:05:00	3573	32.6
2016-04-01 00:20:00	3572	32.7
2016-04-01 00:35:00	3572	32.6
...		

1. Timestamp harmonization via resampling: As a result of irregular data acquisition frequency of the wireless sensor network, the timestamps of the measurements are not equidistant. See table 4 for an exemplary screenshot of the data and the irregular timestamps that are supposed to be regular at 15min-interval.

To ensure regular time-steps for further processing, the data points are shifted to the closest timestamp of a 15min-grid, thereby also eliminating possible duplicates. The corresponding line in the config-file is this one:

```
varname ; test
#-------;-----------------------------------
SM2     ; shift(freq="15Min")
```

And the resulting output data is displayed in table 5: **2.**

Range test: Next, a range test to exclude values below/above a physically meaningful threshold is performed (here: [10, 60], vol.% soil moisture).

```
SM2     ; flagRange(min=10, max=60)
```

3. MAD outlier test: Following this, a Modified Z-Score (MAD) outlier detection test, as proposed by Iglewicz and Hoaglin (1993), is performed to detect and flag spikes in the data. The resulting flags of steps 2 and 3 are depicted in figure 8.

```
SM2     ; flagMAD(window="30d", z=3.5)
```

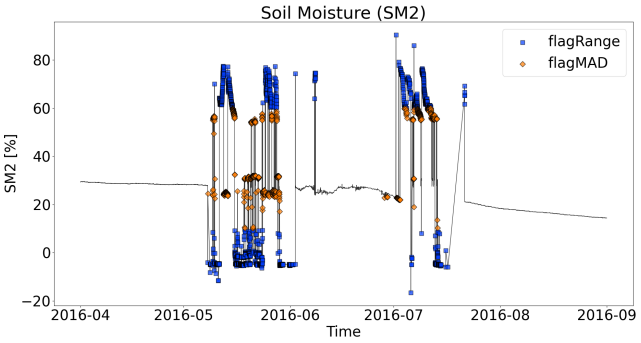

Figure 8: Quality flags of the variable *SM2* as a result of steps 2 and 3, i.e. a range-test (flagRange, blue markers) and a MAD outlier test (flagMAD, orange markers)

4. Linear interpolation of missing values As a final post-processing step, the missing values that result from identifying erroneous values during the preceding flagging are filled using linear interpolation:

```
SM2     ; interpolateInvalid(method='linear')
```

The final, clean results, i.e. a comparison of the time series of the variable *SM2* before and after the QC-process is displayed in figure 9. The major error pattern, fluctuations beyond a plausible value range, has been successfully corrected for. Given the amount of erroneous data points that were identified (fig. 8) and excluded, the resulting data gaps make up a considerable portion of the whole time series snippet. Therefore, the application of linear interpolation in step 4 is questionable and would have to be communicated to the end user in each data points' metadata as explained in section 4.5. Here, it is only applied with the aim of showcasing the functions of SaQC.

5.2. Use case 2: Advanced quality control of surface water body data

At the Rappbode observatory, Germany's largest drinking water reservoir with a maximum storage of about 100 million m^3, a multitude of hydrological variables are recorded to investigate the inflow dynamics of nutrients and dissolved organic carbon (DOC) from the surrounding catchment (see fig. 10). For more details on the observatory, please see Rinke et al. (2013). More specifically, the variables *water level, water temperature*, and *sac254* (spectral absorption coefficient at 254 nm, used as a proxy for DOC) are measured in all four major inflows using the sensors as presented in table 6. These measurements are highly sensitive and require regular maintenance as well as sophisticated quality control which is performed in five steps. In this section,

Electronic copy available at: https://ssrn.com/abstract=4173698

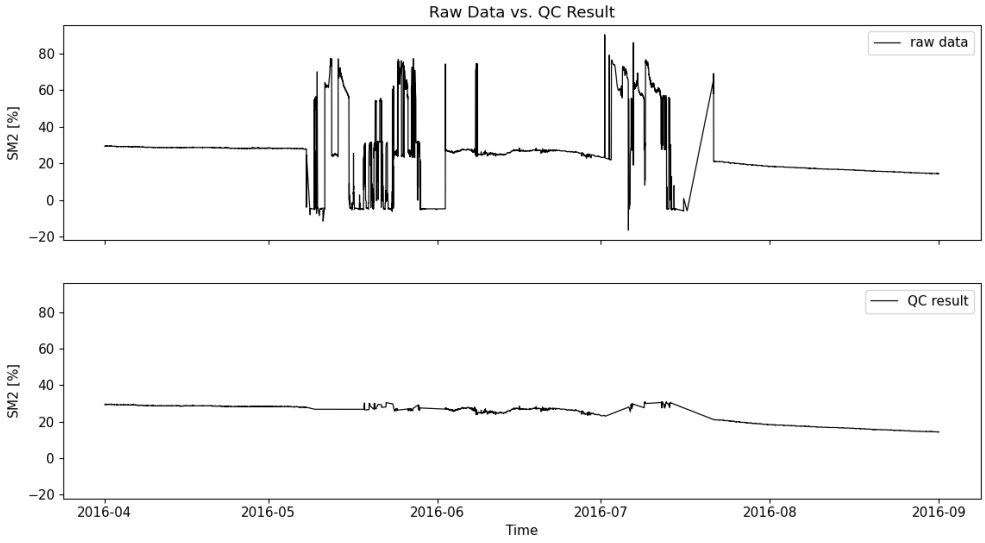

Figure 9: time series of the variable *SM2* before (top) and after (bottom) the QC-process.

Figure 10: Left: Birds-eye view of the Rappbode reservoir, the site of the corresponding hydrological observatory. Right: Measurement buoy carrying sensors as employed in use case 2. Image source: UFZ.

Table 6
Model and manufacturer of the sensors used to measure the variables of use case 2.

Variable	Unit	Sensor model
water level	m	Analog Submersible Level Sensor, STS AG
water temperature	°C	EXO2 Multiparameter Probe, Xylem Inc
sac254	m^{-1}	OPUS UV Spectral Sensor, TriOS, Germany GmbH

the first four steps are elaborated for one of the variables, *sac254*, only. In the last step, all three variables are used as covariates for multivariate flagging. This description handles line by line of the actual config-file needed for the final results (see appendix A.3), along with illustrative figures. An in-depth presentation of the entire workflow, including all figures, algorithmic details and references, can be found in the respective Cookbook [18].

Figure 11 depicts the raw time series of *sac254*, i.e. before quality control. It exhibits undesired outliers and cyclical drops in its data values that are physically implausible.

[18] https://rdm-software.pages.ufz.de/saqc/cook_books/MultivariateFlagging.html

This advanced use case touches on the potential complexity

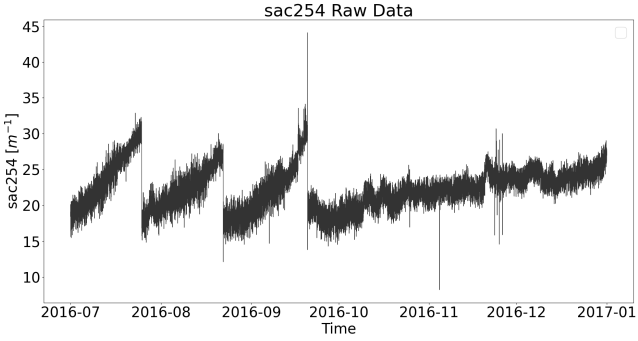

Figure 11: Raw data of the variable *sac254* before quality control.

of QC-workflows with multiple routines building on top of each other. Also, it shows some of the functionalities of SaQC that are indispensable in the practical application - e.g. the projection of flags in step 5.

1. Flagging of maintenance intervals: As a first step, data recorded during known maintenance operations is flagged as these operations affect the measurements. The respective information is encoded in a separate variable *maint* and used inside the manual flagging function.

```
varname ; test
#-------;----------------------------------------------
sac254  ; flagManual(mdata='maint')
```

2. Range test: Next, basic QC in form of a range test to exclude values below/above a physically meaningful threshold is performed (here: [0,70], m^{-1}). This is necessary to

Electronic copy available at: https://ssrn.com/abstract=4173698

avoid incorrect interpolation of data values in the next step. The resulting flags of step 1 and 2 are visualized in figure 12

```
sac254  ; flagRange(min=0, max=70)
```

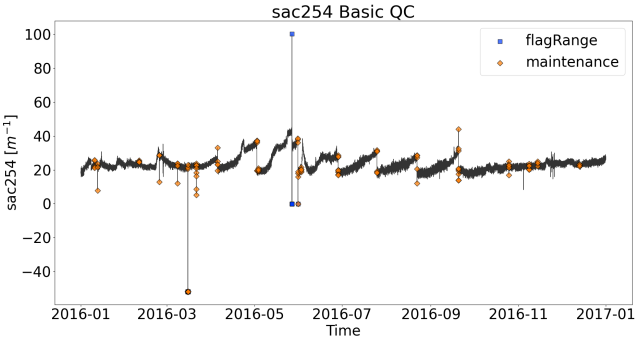

Figure 12: Flags for of the variable *sac254* as a result of steps 1 and 2, i.e. based on known maintenance intervals (flagManual, orange markers) and a range test (flagRange, blue markers)

.

3. Timestamp harmonization via resampling: As the sensors do not necessarily measure at the same interval or exact same time, the timestamps of all variables are resampled to an equi-distant, synchronous grid of 15min-intervals. This is needed for subsequent multivariate QC-tests. As the data values themselves have to be adapted accordingly, linear interpolation is performed on the data values where necessary, not touching on actual missing values. The interpolation time step has to be adjusted to a value that is fitting to the original temporal resolution and that does not lead to larger changes in the dynamics.

```
sac254  ; linear(freq='15min')
```

4. Drift correction: The variable *sac254* exhibits a short-term drift: Due to biofilm growth and dirt accumulation on the sensor lenses, the reference, or base value of the measurements increases over time as illustrated in figure 13. Based on expert knowledge, these increases are assumed to follow an exponential growth and are thus corrected for by subtraction of an exponential term that was parameterized manually in advance. Whenever maintenance, i.e. sensor cleaning or replacement is performed, the drift correction is reset to its new starting point.

```
sac254  ; correctDrift(target='sac254_corr', maintenance_field='maint',
                model=expDriftModel)
```

5. Multivariate spike detection using kNN and STRAY: Next, not only the information of *sac254*, but of all three variables (*water level, water temperature* and *sac254*) is used to identify multivariate outliers. The methodology is based on the *oddwater*-algorithm by Talagala et al. (2019). The aim is to employ the well-established unsupervised clustering algorithm k-Nearest Neighbors (kNN). This requires the variables to be normalized in a first step using z-score transformation. Next, the kNN-algorithm is used to assign

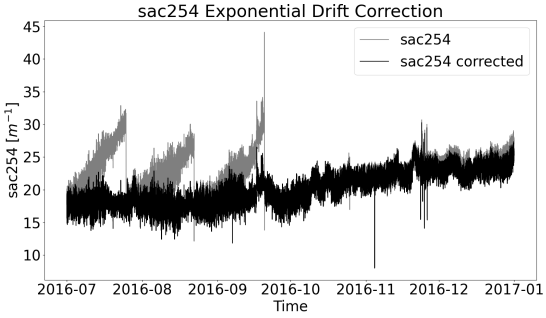

Figure 13: Illustration of the drift correction of the variable *sac254*. The grey line depicts the original data, the black line represents corrected data, both at 15min-resolution.

a multivariate outlier score (new variable: *kNN_scores*) to each timestep. Figure 14 (top) provides a visual intuition of how, during this dimensionality reduction, multivariate outliers become detectable when transformed into one variable. These *kNN_scores* are then processed further using the STRAY-algorithm (Talagala et al., 2021) to automatically define a threshold above which data points of the *kNN-scores* are considered to be actual outliers and can be flagged as such. As a last step, these flags are projected from the *kNN_scores*-variable back onto the actual data variables. Figure 14 (bottom) depicts the results after this projection is performed for *sac254*.

```
# 5: Multivariate spike detection using kNN and STRAY
# Normalization of variables for kNN
level      ; transform(target='level_norm', func='zScore', freq='20D')
water_temp ; transform(target='water_temp_norm', func='zScore', freq='20D')
sac254_corr ; transform(target='sac254_norm', func='zScore', freq='20D')

# Flagging by kNN and STRAY
level_norm,water_temp_norm,
sac254_norm ; assignKNNScore(target='kNN_scores', freq='20D')
kNN_scores  ; flagByStray(freq='20D')

# Project results of STRAY-algorithm onto variables
level      ; transferFlags(field=['kNN_scores'])
water_temp  ; transferFlags(field=['kNN_scores'])
sac254_corr ; transferFlags(field=['kNN_scores'])
```

The final, clean results, i.e. a comparison of the time series of the variable *sac254* before and after the QC-process, are presented in figure 15. In general, all major error patterns (major outliers, cyclical drops, drift) have been corrected for. A few spikes remain that should be analyzed in more detail by the domain expert and that could possibly be covered by the parametrization of one of the spike tests included in SaQC. However, it is to be noted that the figures shown here only depict snippets of the whole time series at one sensor while the observatory consists of numerous sensors that exhibit slightly varying error characteristics. As finding the right parametrization for a time snippet of one single sensor already requires substantial efforts by the domain experts, finding a robust parametrization for the whole network is always a trade-off between local performance

Electronic copy available at: https://ssrn.com/abstract=4173698

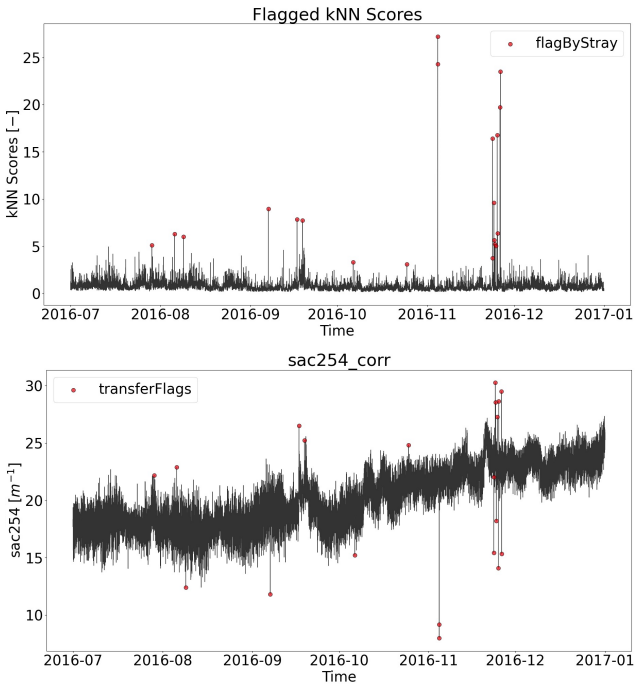

Figure 14: Top: Multivariate outliers (red markers) in the *kNN-scores* as identified by the STRAY-algorithm, i.e. based on all three variables (*water level*, *water temperature* and *sac254*). Bottom: The corresponding outliers projected onto and visualized for the variable *sac254*, only. Note that, for illustration, this plots depicts a shorter time snippet than the previous figures, i.e. is zoomed in.

(time series snippets of one sensor) and global performance (whole network).

6. Discussion

The only way to turn ever-increasing volumes of data from environmental sensor networks into actionable data products of high quality for monitoring, modeling and decision support is rigorous automation of the entire data workflow including its quality control component. Commonly, the latter has to be implemented by domain experts with limited IT expertise, requiring suitable and accessible software tools. The previous sections present SaQC as a possible solution that meets the key requirements for such software. Further, it is illustrated how domain experts are enabled to set up flexible and reproducible QC workflows for any time series data. Nonetheless, it is important to know about certain limitations of the software as well as future developments and research directions as presented in the following paragraph.

6.1. Limitations of the software

One challenge that still persists for users with limited IT skills is that, currently, it is required to install the programming language Python. Similarly, configuring the QC-tests via a .csv config-file might be unfamiliar to users that are used to employ software solely using a Graphical User Interface (GUI). To this end, the web-based SaQC Configuration App was developed that supports the process of QC-test parametrization via GUI. Regarding input file types, these are currently limited to .csv and .parquet if SaQC is used via the command line interface (CLI), here netCDF support is envisioned. If used via the Python API, any file format or database API can be implemented by the user. Concerning traceability and reproducibility, 1) versioning and 2) metadata enrichment are optional functionalities. If desired, the user is required to 1) host the QC-pipeline on the code versioning platform Git and 2) be able to store additional metadata (SaQC version, flagging schema, QC-test that was executed etc.) for each data point in the respective data repository. Depending on the system in use, the latter can potentially result in considerable computational load if the configuration of the pipeline is changed and all historical data points are to be re-processed. Lastly and naturally, setting up a QC-workflow still requires considerable parametrization effort, but SaQC is designed to make this as easy as possible while rendering labor-intensive manual quality control superfluous.

6.2. Future development and research directions

Future development of SaQC is envisioned in multiple ways: Firstly, it will be implemented in further real-world environmental monitoring workflows, both at the authors' institutions and beyond. While doing so, the existing set of QC-tests will be continuously extended by both domain-specific and -agnostic tests. Future data volumes are, however, expected to be ever growing, among others due to the use of mobile low-cost sensors inside the growing citizen science community Koedel et al. (2022). Sensor mobility will significantly raise the algorithmical complexity of the respective QC workflows to assure internal consistency, likely reaching a point where manual parametrization of QC tests becomes too complex. While we will evaluate semi-automated parametrization schemes as proposed by Durre et al. (2008) and Taylor and Loescher (2013), entirely automated, Machine Learning-based methods are likely to be required not only for QC, but also for data imputation. As a step in this direction, we currently evaluate the expansion of the available Machine Learning functions by Deep Learning algorithms for anomaly detection as these have recently been shown to achieve high classification accuracy by exploiting spatio-temporal patterns in the data of the entire sensor network (e.g. Erhan et al., 2021; Jones et al., 2022). Another potential extension of SaQC is the integration of external data sources to validate online sensor data. As an example, the soil moisture data of use case 1 could be validated using remote-sensing products like SMAP[19]. Another relevant topic is the auditing of QC-workflows: We will evaluate methods to monitor QC-workflows regarding spatial and temporal patterns to avoid introducing systematic bias or overflagging as a result of malparametrized tests. This could be achieved based on methods as presented in e.g. Durre

[19] NASA Soil Moisture Active Passive Mission (Entekhabi et al., 2010)

Electronic copy available at: https://ssrn.com/abstract=4173698

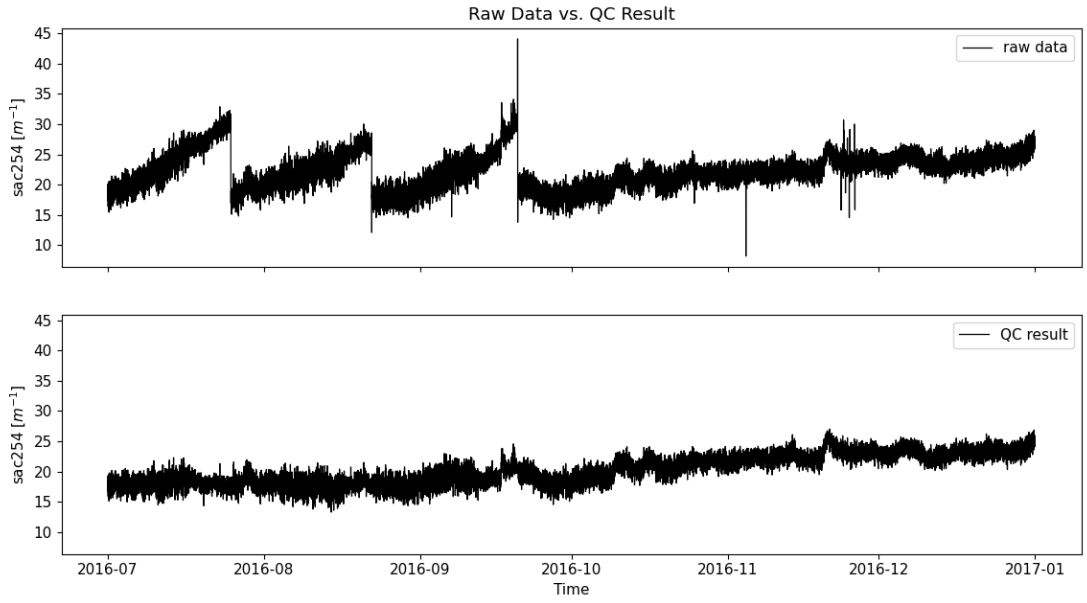

Figure 15: time series of the variable *sac254* before (top) and after (bottom) the QC-process.

et al. (2008) and Smith et al. (2014), along with a user-centred communication of the audit results, i.e. an estimate of the performance of a QC-workflow, as in Sturtevant et al. (2021).

SaQC will be disseminated in the scientific community via hands-on workshops to enable more domain scientists to use it in their specific application. With a growing user community, the authors envision an open source community that contributes specific QC-tests along with suitable parametrization schemes for future re-use. By providing a domain-agnostic and flexible software tool, the authors hope to contribute to an urgent and inevitable standardization of QC workflows and flagging schemata across domains, thus fostering FAIR data flows.

7. Conclusions

In order to monitor, model and investigate the drivers of ongoing environmental changes, large-scale sensor networks have been set up and are continuously expanding in number of sensors and geographical coverage. The corresponding ever-growing raw data volumes that need to be processed into actionable data products in real-time pose a significant challenge to the respective data workflows. Rigorous automation of data workflows and their quality-control component is essential. This in turn, demands for domain-agnostic software tools to establish automated QC-workflows, aiming at a user group that mostly consists of domain experts with limited IT-expertise. In this publication, we derive key requirements for such software and present SaQC, an Open Source framework for quality control of time series data that meets these requirements by providing the following features: It is universal as to its application domain and flexible with regard to both its input/output data structures as well as flagging schemata. While already providing a comprehensive set of methods, it is extensible in its pre- and post-processing as well as QC-testing functions. All these functions are designed such that the handling of quality flags between processing steps is controllable and explicit. Adding to this, it provides functionality to make QC-workflows traceable and reproducible, thus promoting FAIR workflows. Also, SaQC is designed to be accessible, particularly to users without programming knowledge as is often the case in practical application. By means of a detailed software description and the presentation of two real-world use cases, we showcase how SaQC can be used to process and perform quality control on one‘s own data both at basic and at advanced level. Thus, the scientific community is supplied with (instructions on) a software framework to automatically deliver environmental data that is of high quality, up-to-date, potentially large in volume and reusable - Principal prerequisites for data-driven and sustainable science to address urgent environmental challenges.

Electronic copy available at: https://ssrn.com/abstract=4173698

8. Acknowledgments and Funding

The Hohes Holz and the Rappbode observatories are financially supported by TERENO (Terrestrial Environmental Observatories, funded by the Helmholtz Association and the Federal Ministry of Education and Research, BMBF) and the Talsperrenbetrieb Sachsen-Anhalt.
Karsten Rinke received funding from the H2020-MSCA-ITN-2020 innovative training network InventWater (GrantNo 956623). We acknowledge previous work by Juliane Mai and Matthias Cuntz who established first data quality control procedures. Similarly, we acknowledge input by Corinna Rebmann who provided data, ideas and discussion on test procedures.

A. Appendix

A.1. Table of QC and processing functions

Please note that, for readability, the following table A.1 only lists the most relevant processing and QC-functions. There is a multitude of specific helper functions that are listed in the SaQC online documentation.

Electronic copy available at: https://ssrn.com/abstract=4173698

Table 7
Primary processing and QC functions of SaQC. The grouping is according to table 2 but giving name, descriptions and references, where applicable, for each single function. (*Note: Too small due to LaTeX class, will be full-page and horizontal*)

Group	Function Name	Objective
Processing	*Processing steps prior to (pre-processing) or after QC-testing (post-processing)*	
Harmonisation	shift	Shift data points and their flags to align them with regular frequency grids
	resample	Resample data points and flags with an aggregation to align them with regular frequency grids
	interpolate	Interpolate data and aggregate flags at regular frequency grids using parametric and non-parametric methods
Smoothing	rolling	Process data by applying or fitting a function in a rolling window manner
	fitPolynomial	Smooth data by chunkwise fitting polynomials
Transformation	transform	Process data by applying a custom function on data chunk, e.g. normalization
	processGeneric	Generate new data via custom function of (multiple) data variables and their flagging status
Flag projection	concatFlags	Project flags from (regularly sampled) data products back onto original data
Basic QC-tests	*Low algorithmic complexity and parametrization effort*	
Constants	flagConstants	Flag data where it is strictly constant
	flagByVariance	Flag data chunks that have too low a variance
Breaks	flagMissing	Assign flags to missing value markers
	flagIsolated	Flag groups of values, isolated by missing or invalid data
Outliers	flagRange	Flag values that lie below or exceed a given absolute limit
	flagOffset	Customizable deterministic outlier detection
	flagByStray	Use stray algorithm to flag values
	flagMAD	Use median statistics to flag outliers
	flagByGrubbs	Use Grubbs algorithm to find outliers in given window
	flagMVScores	Use K-nearest neighbor aggregation to flag multiple variables
Manual flagging	flagManual	Transfer pre-existing flags from auxiliary files into the pipeline
Advanced QC-tests	*Higher algorithmic complexity and parametrization effort*	
Noise	flagByStatLowPass	Flag data chunks based on tests in the frequency domain
Changepoints	flagChangePoints	Flag data points that represent a system state transition based on sample statistics
	flagRegimeAnomaly	Flag chunks of the data that behave abnormal based on agglomerative clustering
Drift	correctDrift	Remove drift component from data, according to a custom drift model
	flagDriftFromReference	Flag data where its deviation from a reference data set exceeds a threshold
	flagDriftFromNorm	Flag data chunk that gets assigned to a minority group by agglomerative time series clustering
Pattern recognition	flagPatternByWavelet	Detect and flag data chunks by evaluating their distance to a given pattern in the frequency domain
	flagPatternByDTW	Detect and flag data chunks by evaluating their distance to a given pattern using dynamic time warping
Custom functions	flagGeneric	Derive flags from a custom, boolean valued function of (multiple) data variables and their flagging status
Machine learning	trainModel	Train an Ensemble of tree-based boosting and bagging models with an Autotuner (MLjar)
	modelPredict	Predict data with a trained regressor
	modelFlag	Flag data with a trained classifier
	modelImpute	Impute data with a trained regressor

A.2. Config-file use case 1

This config-file includes all steps to perform QC as presented in use case 1 in section 5.1. For a more detailed description, the reader is referred to the respective Cookbook in the SaQC online documentation.

```
varname     ; test
#-----------;----------------------------------
SM2         ; shift(freq="15Min")
SM2         ; flagRange(min=10, max=60)
SM2         ; flagMAD(window="30d", z=3.5)
SM2         ; interpolateInvalid(method='linear')
```

A.3. Config-file use case 2

Please note that, while the description of use case 2 in section 5.2 focuses on the variable *sac254* for readability, this config-file lists the required steps for all three variables (*water level, water temperature* and *sac254*). For a more detailed description, the reader is referred to the respective Cookbook in the SaQC online documentation.

```
varname     ; test
#-----------;----------------------------------
# 1: Flagging of maintenance intervals
sac254      ; flagManual(mdata='maint')

# 2: Basic QC: Range-tests
level       ; flagRange(min=0)
water_temp  ; flagRange(min=-.5)
sac254      ; flagRange(min=0, max=70)

# 3: Pre-processing: Resampling via linear interpolation
level       ; linear(freq='15min')
water_temp  ; linear(freq='15min')
sac254      ; linear(freq='15min')

# 4: Drift correction
sac254    ; correctDrift(target='sac254_corr', maintenance_field='mai
                 model='exponential')

# 5: Multivariate spike detection using kNN and STRAY
# Normalization of variables for kNN
level     ; transform(target='level_norm', func='zScore', freq='20D')
water_temp ; transform(target='water_temp_norm', func='zScore', freq=
sac254_corr ; transform(target='sac254_norm', func='zScore', freq='20D

# Flagging by kNN and STRAY
level_norm,water_temp_norm,
sac254_norm ; assignKNNScore(target='kNN_scores', freq='20D')
kNN_scores  ; flagByStray(freq='20D')

# Project results of STRAY-algorithm onto variables
level       ; transferFlags(field=['kNN_scores'])
water_temp  ; transferFlags(field=['kNN_scores'])
sac254_corr ; transferFlags(field=['kNN_scores'])
```

CRediT authorship contribution statement

Lennart Schmidt: Conceptualization, Software Development, Writing. **David Schäfer:** Conceptualization, Software Development, Supervision. **Juliane Geller:** Conceptualization, Software Development, Writing. **Peter Lünenschloss:** Conceptualization, Software Development, Writing. **Bert Palm:** Conceptualization, Software Development, Writing. **Karsten Rinke:** Data collection and Curation,

Electronic copy available at: https://ssrn.com/abstract=4173698

Writing, Review. **Jan Bumberger:** Conceptualization, Writing, Supervision.

References

ACTRIS, 2021. Aerosols, clouds, and trace gases research infrastructure network. URL: https://www.actris.eu.

Campbell, J.L., Rustad, L.E., Porter, J.H., Taylor, J.R., Dereszynski, E.W., Shanley, J.B., Gries, C., Henshaw, D.L., Martin, M.E., Sheldon, W.M., et al., 2013. Quantity is nothing without quality: Automated qa/qc for streaming environmental sensor data. BioScience 63, 574–585. doi:10.1525/bio.2013.63.7.10.

Caveness, E., GC, P.S., Peng, Z., Polyzotis, N., Roy, S., Zinkevich, M., 2020. Tensorflow data validation: Data analysis and validation in continuous ml pipelines, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 2793–2796.

Collier-Oxandale, A., Feenstra, B., Papapostolou, V., Polidori, A., 2022. Airsensor v1. 0: Enhancements to the open-source r package to enable deep understanding of the long-term performance and reliability of purpleair sensors. Environmental Modelling & Software 148, 105256. doi:10.1016/j.envsoft.2021.105256.

Crystal-Ornelas, R., Varadharajan, C., Christianson, D., Damerow, J., Weierbach, H., Robles, E., Ramakrishnan, L., Faybishenko, B., Pastorello, G., 2021. A library of ai-assisted fair water cycle and related disturbance datasets to enable model training, parameterization and validation URL: https://www.osti.gov/biblio/1769646, doi:10.2172/1769646.

Doraiswamy, P., Pasteris, P., Jones, K., Motha, R., Nejedlik, P., 2000. Techniques for methods of collection, database management and distribution of agrometeorological data. Agricultural and Forest Meteorology 103, 83–97. doi:10.1016/S0168-1923(00)00120-9.

Dorigo, W., Xaver, A., Vreugdenhil, M., Gruber, A., Dostálová, A., Sanchis-Dufau, A., Zamojski, D., Cordes, C., Wagner, W., Drusch, M., 2013. Global automated quality control of in situ soil moisture data from the international soil moisture network. Vadose Zone Journal 12. doi:10.2136/vzj2012.0097.

Durre, I., Menne, M., Gleason, B., Houston, T., Vose, R., 2010. Comprehensive automated quality assurance of daily surface observations. Journal of Applied Meteorology and Climatology 49. doi:10.1175/2010JAMC2375.1.

Durre, I., Menne, M.J., Vose, R.S., 2008. Strategies for evaluating quality assurance procedures. Journal of Applied Meteorology and Climatology 47, 1785–1791. doi:10.1175/2007JAMC1706.1.

Entekhabi, D., Njoku, E.G., O'Neill, P.E., Kellogg, K.H., Crow, W.T., Edelstein, W.N., Entin, J.K., Goodman, S.D., Jackson, T.J., Johnson, J., et al., 2010. The soil moisture active passive (smap) mission. Proceedings of the IEEE 98, 704–716. doi:https://doi.org/10.1109/JPROC.2010.2043918.

Erhan, L., Ndubuaku, M., Di Mauro, M., Song, W., Chen, M., Fortino, G., Bagdasar, O., Liotta, A., 2021. Smart anomaly detection in sensor systems: a multi-perspective review. Information Fusion 67, 64–79. doi:10.1016/j.inffus.2020.10.001.

Estévez, J., Gavilán, P., Giráldez, J.V., 2011. Guidelines on validation procedures for meteorological data from automatic weather stations. Journal of Hydrology 402, 144–154. doi:10.1016/j.jhydrol.2011.02.031.

Fiebrich, C.A., Morgan, C.R., McCombs, A.G., Hall Jr, P.K., McPherson, R.A., 2010. Quality assurance procedures for mesoscale meteorological data. Journal of Atmospheric and Oceanic Technology 27, 1565–1582. doi:10.1175/2010JTECHA1433.1.

Fischetti, T., 2022. assertr: Assertive Programming for R Analysis Pipelines. Https://docs.ropensci.org/assertr/ (website) https://github.com/ropensci/assertr.

Gandin, L.S., 1988. Complex quality control of meteorological observations. Monthly Weather Review 116, 1137–1156.

GoFair, 2021. Fair principles. URL: https://www.go-fair.org/fair-principles/.

Gong, A., Campbell, J., 2022. Great expectations. doi:10.5281/zenodo.6403212.

Gouldman, C.C., Bailey, K., Thomas, J.O., 2017. Manual for real-time oceanographic data quality control flags.

Electronic copy available at: https://ssrn.com/abstract=4173698

Gourrion, J., Szekely, T., Killick, R., Owens, B., Reverdin, G., Chapron, B., 2020. Improved statistical method for quality control of hydrographic observations. Journal of Atmospheric and Oceanic Technology 37, 789–806. doi:10.1175/JTECH-D-18-0244.1.

Heiskanen, J., Brümmer, C., Buchmann, N., Calfapietra, C., Chen, H., Gielen, B., Gkritzalis, T., Hammer, S., Hartman, S., Herbst, M., Janssens, I.A., Jordan, A., Juurola, E., Karstens, U., Kasurinen, V., Kruijt, B., Lankreijer, H., Levin, I., Linderson, M.L., Loustau, D., Merbold, L., Myhre, C.L., Papale, D., Pavelka, M., Pilegaard, K., Ramonet, M., Rebmann, C., Rinne, J., Rivier, L., Saltikoff, E., Sanders, R., Steinbacher, M., Steinhoff, T., Watson, A., Vermeulen, A.T., Vesala, T., Vítková, G., Kutsch, W., 2021. The integrated carbon observation system in europe. Bulletin of the American Meteorological Society , 1 – 54URL: https://journals.ametsoc.org/view/journals/bams/aop/BAMS-D-19-0364.1/BAMS-D-19-0364.1.xml, doi:10.1175/BAMS-D-19-0364.1.

Horsburgh, J.S., Reeder, S.L., Jones, A.S., Meline, J., 2015. Open source software for visualization and quality control of continuous hydrologic and water quality sensor data. Environmental Modelling & Software 70, 32–44. doi:10.1016/j.envsoft.2015.04.002.

Hubbard, K.G., You, J., 2005. Sensitivity analysis of quality assurance using the spatial regression approach—a case study of the maximum/minimum air temperature. Journal of Atmospheric and Oceanic Technology 22, 1520–1530.

Iglewicz, B., Hoaglin, D., 1993. Volume 16: how to detect and handle outliers. The ASQC basic references in quality control: statistical techniques 16.

Ingleby, B., Huddleston, M., 2007. Quality control of ocean temperature and salinity profiles — historical and real-time data. Journal of Marine Systems 65, 158 – 175. doi:10.1016/j.jmarsys.2005.11.019.

Jones, A.S., Horsburgh, J.S., Eiriksson, D.P., 2018. Assessing subjectivity in environmental sensor data post processing via a controlled experiment. Ecological Informatics 46, 86–96. doi:10.1016/j.ecoinf.2018.05.001.

Jones, A.S., Jones, T.L., Horsburgh, J.S., 2022. Toward automating post processing of aquatic sensor data. Environmental Modelling and Software 151, 105364. URL: https://www.sciencedirect.com/science/article/pii/S1364815222000706, doi:10.1016/j.envsoft.2022.105364.

Kaffashzadeh, N., Kleinert, F., Schultz, M.G., 2019. A new tool for automated quality control of environmental time series (autoqc4env) in open web services, in: International Conference on Business Information Systems, Springer. pp. 513–518.

Koedel, U., Schuetze, C., Fischer, P., Bussmann, I., Sauer, P.K., Nixdorf, E., Kalbacher, T., Wichert, V., Rechid, D., Bouwer, L.M., Dietrich, P., 2022. Challenges in the evaluation of observational data trustworthiness from a data producers viewpoint (fair+). Frontiers in Environmental Science 9. doi:10.3389/fenvs.2021.772666.

Liao, W., Wang, D., Wang, G., Xia, Y., Liu, X., 2019. Quality control and evaluation of the observed daily data in the north american soil moisture database. Journal of Meteorological Research 33, 501–518. doi:10.1007/s13351-019-8121-2.

Loescher, H.W., Kelly, E.F., Lea, R., 2017. National ecological observatory network: Beginnings, programmatic and scientific challenges, and ecological forecasting, in: Terrestrial Ecosystem Research Infrastructures. CRC Press, pp. 27–52.

Long, C.N., Dutton, E.G., 2010. Baseline surface radiation network global network recommended qc tests, v2.x. doi:10013/epic.38770.d001.

Mauder, M., Cuntz, M., Drüe, C., Graf, A., Rebmann, C., Schmid, H.P., Schmidt, M., Steinbrecher, R., 2013. A strategy for quality and uncertainty assessment of long-term eddy-covariance measurements. Agricultural and Forest Meteorology 169, 122–135. doi:10.1016/j.agrformet.2012.09.006.

Metzger, S., Durden, D., Sturtevant, C., Luo, H., Pingintha-Durden, N., Sachs, T., Serafimovich, A., Hartmann, J., Li, J., Xu, K., et al., 2017. eddy4r 0.2. 0: a devops model for community-extensible processing and analysis of eddy-covariance data based on r, git, docker, and hdf5. Geoscientific Model Development 10, 3189–3206.

Metzger, S., Taylor, J., Luo, H., Loescher, H., 2013. Development of a quality assurance and quality control framework for neon's eddy-covariance flux measurements.

Mirtl, M., Borer, E., Djukic, I., Forsius, M., Haubold, H., Hugo, W., Jourdan, J., Lindenmayer, D., McDowell, W.H., Muraoka, H., et al., 2018. Genesis, goals and achievements of long-term ecological research at the global scale: a critical review of ilter and future directions. Science of the total Environment 626, 1439–1462. doi:10.1016/j.scitotenv.2017.12.001.

Mollenhauer, H., Kasner, M., Haase, P., Peterseil, J., Wohner, C., Frenzel, M., Mirtl, M., Schima, R., Bumberger, J., Zacharias, S., 2018. Long-term environmental monitoring infrastructures in europe: observations, measurements, scales, and socio-ecological representativeness. Science of The Total Environment 624, 968–978. doi:10.1016/j.scitotenv.2017.12.095.

Pastorello, G., Trotta, C., Canfora, E., Chu, H., Christianson, D., Cheah, Y.W., Poindexter, C., Chen, J., Elbashandy, A., Humphrey, M., et al., 2020. The fluxnet2015 dataset and the oneflux processing pipeline for eddy covariance data. Scientific data 7, 1–27. doi:10.1038/s41597-021-00851-9.

Petzold, A., Thouret, V., Gerbig, C., Zahn, A., Brenninkmeijer, C.A., Gallagher, M., Hermann, M., Pontaud, M., Ziereis, H., Boulanger, D., et al., 2015. Global-scale atmosphere monitoring by in-service aircraft–current achievements and future prospects of the european research infrastructure iagos. Tellus B: Chemical and Physical Meteorology 67, 28452. doi:10.3402/tellusb.v67.28452.

Rebmann, C., Claudia, S., Sara, M.J., Sebastian, G., Matthias, Z., Luis, S., Matthias, C., 2017. Integrative measurements focusing on carbon, energy and water fluxes at the forest site 'hohes holz' and the grassland 'am grossen bruch', in: EGU General Assembly Conference Abstracts, p. 9727.

Reid, W.V., Chen, D., Goldfarb, L., Hackmann, H., Lee, Y.T., Mokhele, K., Ostrom, E., Raivio, K., Rockström, J., Schellnhuber, H.J., et al., 2010. Earth system science for global sustainability: grand challenges. Science 330, 916–917. doi:10.1126/science.1196263.

Rinke, K., Kuehn, B., Bocaniov, S., Wendt-Potthoff, K., Büttner, O., Tittel, J., Schultze, M., Herzsprung, P., Rönicke, H., Rink, K., et al., 2013. Reservoirs as sentinels of catchments: the rappbode reservoir observatory (harz mountains, germany). Environmental earth sciences 69, 523–536. doi:10.1007/s12665-013-2464-2.

Schmithüsen, H., Koppe, R., Sieger, R., König-Langlo, G., 2019. BSRN Toolbox V2.5 - a tool to create quality checked output files from BSRN datasets and station-to-archive files. URL: https://doi.org/10.1594/PANGAEA.901332, doi:10.1594/PANGAEA.901332.

Sheldon, W.M., 2008. Dynamic, rule-based quality control framework for real-time sensor data, in: Proceedings of the Environmental Information Management Conference, Citeseer. pp. 145–150.

Smith, D.E., Metzger, S., Taylor, J.R., 2014. A transparent and transferable framework for tracking quality information in large datasets. PLoS One 9, e112249. doi:10.1371/journal.pone.0112249.

Sturtevant, C., Metzger, S., Nehr, S., Foken, T., 2021. Quality assurance and control, in: Springer Handbook of Atmospheric Measurements. Springer, pp. 47–90. doi:10.1007/978-3-030-52171-4_3.

Talagala, P.D., Hyndman, R.J., Leigh, C., Mengersen, K., Smith-Miles, K., 2019. A feature-based procedure for detecting technical outliers in water-quality fata from in situ sensors. Water Resources Research 55, 8547–8568. doi:10.1029/2019WR024906.

Talagala, P.D., Hyndman, R.J., Smith-Miles, K., 2021. Anomaly detection in high-dimensional data. Journal of Computational and Graphical Statistics 30, 360–374. doi:10.1080/10618600.2020.1807997.

Taylor, J., Loescher, H., 2012. Neon's fundamental instrument unit dataflow and quality assurance plan.

Taylor, J.R., Loescher, H.L., 2013. Automated quality control methods for sensor data: a novel observatory approach. Biogeosciences 10, 4957–4971. doi:10.5194/bg-10-4957-2013.

Urraca, R., Sanz García, A., Sanz-Garcia, I., 2020. Bqc: A free web service to quality control solar irradiance measurements across europe. Solar Energy 211, 1–10. doi:10.1016/j.solener.2020.09.055.

Electronic copy available at: https://ssrn.com/abstract=4173698

Vitale, D., Fratini, G., Bilancia, M., Nicolini, G., Sabbatini, S., Papale, D., 2020. A robust data cleaning procedure for eddy covariance flux measurements. Biogeosciences 17, 1367–1391. doi:10.5194/bg-17-1367-2020.

Wagner, R.J., Boulger Jr, R.W., Oblinger, C.J., Smith, B.A., 2006a. Guidelines and standard procedures for continuous water-quality monitors: station operation, record computation, and data reporting. Technical Report.

Wagner, R.J., Boulger Jr, R.W., Oblinger, C.J., Smith, B.A., 2006b. Guidelines and standard procedures for continuous water-quality monitors: station operation, record computation, and data reporting. Technical Report. U.S. Geological Survey. doi:https://doi.org/10.3133/tm1D3.

Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E., et al., 2016. The fair guiding principles for scientific data management and stewardship. Scientific data 3, 1–9. doi:10.1038/sdata.2016.18.

WMO, 2003. Guidelines on the global data-processing system (WMO-No.305). World Meteorological Organization Geneva, Switzerland.

WMO, 2018. Guide to the Global Observing System (WMO-No.488). World Meteorological Organization Geneva, Switzerland.

Wollschläger, U., Attinger, S., Borchardt, D., Brauns, M., Cuntz, M., Dietrich, P., Fleckenstein, J.H., Friese, K., Friesen, J., Harpke, A., et al., 2017. The bode hydrological observatory: a platform for integrated, interdisciplinary hydro-ecological research within the tereno harz/central german lowland observatory. Environmental Earth Sciences 76, 1–25. doi:10.1007/s12665-016-6327-5.

You, J., Hubbard, K.G., Nadarajah, S., Kunkel, K.E., 2007. Performance of quality assurance procedures on daily precipitation. Journal of Atmospheric and Oceanic Technology 24, 821–834. doi:10.1175/JTECH2002.1.

Younes, S., Claywell, R., Muneer, T., 2005. Quality control of solar radiation data: Present status and proposed new approaches. Energy 30, 1533–1549. doi:10.1016/j.energy.2004.04.031.

Yver-Kwok, C., Philippon, C., Bergamaschi, P., Biermann, T., Calzolari, F., Chen, H., Conil, S., Cristofanelli, P., Delmotte, M., Hatakka, J., et al., 2021. Evaluation and optimization of icos atmosphere station data as part of the labeling process. Atmospheric Measurement Techniques 14, 89–116. doi:10.5194/amt-14-89-2021.

Zacharias, S., Bogena, H., Samaniego, L., Mauder, M., Fuß, R., Pütz, T., Frenzel, M., Schwank, M., Baessler, C., Butterbach-Bahl, K., et al., 2011. A network of terrestrial environmental observatories in germany. Vadose zone journal 10, 955–973. doi:10.2136/vzj2010.0139.

Electronic copy available at: https://ssrn.com/abstract=4173698

View publication stats