Real-time automated quality control of atmospheric aerosol time series

Timo Houben, Thomas Müller, Peter Lünenschloss, Jens Voigtländer, Thomas Trabert, Ema Vosgerau, David Schäfer, and Jan Bumberger

Submitted to AMT

egusphere-2026-235

Response to Reviewer

29th of October 2026

Timo Houben, Thomas Müller, Peter Lünenschloss, Jens Voigtländer, Thomas Trabert, Ema Vosgerau, David Schäfer, and Jan Bumberger

RC1: 'Comment on egusphere-2026-235', Anonymous Referee #1

General comments

Houben and coauthors present a new data processing workflow for quality control (QC) of aethalometer (AE33) and nephelometer (Aurora 4000) measurements. This is an important and timely topic given the intense work on standardized, in situ aerosol data chains currently being performed by many research groups across the ACTRIS network. It is a relevant topic for AMT. However, there are major shortcomings in the work that should be addressed before publication.

We write this review as potential users of these tools. We have assessed the manuscript as well as the companion software package on which it is based.

Response:

Thank you for the careful and constructive assessment. We have revised the manuscript framing and substantially shortened the Introduction and the Background/State-of-the-Art section. The technical and implementation-related concerns are addressed individually below.

Additionally, we substantially improved the software (new name: saqc-atmo) and included almost all requested functionalities from the review. We further implemented a set of settings files which can be adjusted by the user and are tracked throughout the QC pipeline and saved afterwards for reproducibility. Especially the parameterization of the SaQC-tests, the resulting flags and the matching to the EBAS flagging scheme is not always straight forward and needs scientific discussion. We encourage the reviewers as well as the ACTRIS community to debate about appropriate settings and agree on these for further downstream implementation and application, which should continue even after the publication of this manuscript. The required flexibility now ships with the module.

The Python package saqc-atmo is hosted on Helmholtz GitLab

https://codebase.helmholtz.cloud/qc-of-atmospheric-time-series/saqc-for-atmospheric-data

and will be archived on Zenodo upon acceptance of this manuscript. The version used in this work is 0.1.0.dev. The user documentation can be found on GitLab as well and ships with the module:

https://codebase.helmholtz.cloud/qc-of-atmospheric-time-series/saqc-for-atmospheric-data/-/jobs/3630681/artifacts/external_file/public/index.html

The data sets and associated quality-control outputs presented in this work, including flags (CSV), logs (TXT) and diagnostic figures (PNG), will be available via Zenodo upon

acceptance. In order to inspect the results of the presented study based on the anomaly catalogue, the results were uploaded to the UFZ Nextcloud first

https://nc.ufz.de/s/kJk5CJzYRTc9n3G

pw: Kxs\tR82{W

RC1.1 – Scope of the workflow vs. what is claimed

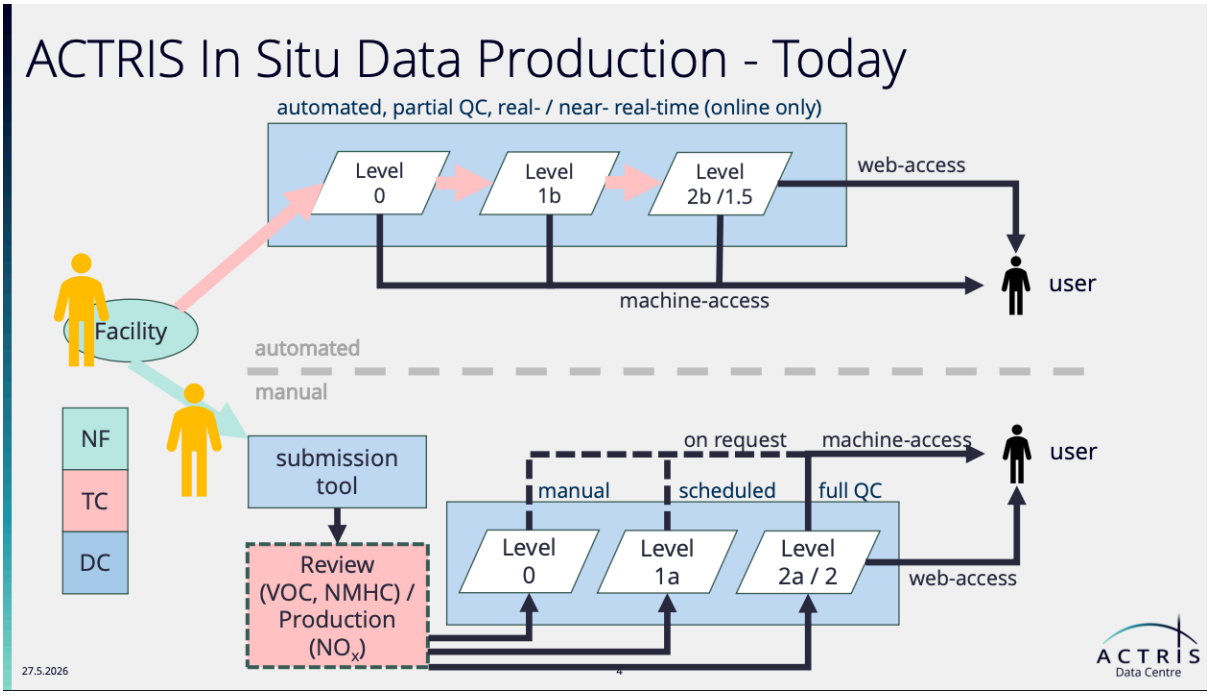
The first major issue is one of scope. The abstract and introduction promise an automated, real-time capable, FAIR-compliant, and AI-ready QC pipeline for AE33 and Aurora 4000 measurements. In reality, what is presented is a proof-of-concept of a potential first step that could be used on raw, level 0 data to partially take it towards level 1 data. It is still a long way from a true end-to-end data pipeline for these instruments. This is fine since the overall goal is probably too big to be dealt with in a single research article. But the limited scope of the work should be much more faithfully represented. We think it is important to clarify the steps that this workflow does not perform - ideally in specific relation to the ACTRIS requirements for level 0-2 AE33 and Aurora 4000 data - in order to help direct future work.

Response:

Thank you. We agree that the original wording did not sufficiently distinguish the automated QC workflow presented here from the complete ACTRIS production and expert-validation chain. We have therefore narrowed the central framing in the revised Introduction and Discussion/Outlook: the manuscript no longer uses "end-to-end" as the central claim for the full ACTRIS data chain and does not present the automated output as a final expert-validated ACTRIS product. ACTRIS is instead described as the reference application and validation context for the aerosol in situ use case.

This clarification is not intended to withdraw the real-time objective of the study. The operational demonstration of the real-time capability and the detailed relation of the workflow to ACTRIS data levels are addressed separately below.

We furthermore extended the workflow according to the ACTRIS Data Management plan, including EBAS flags and NASA-AMES export (experimental) for level 1b and 2b. The quality-controlled dataset after our QC stage 3 fulfills the requirements of the ACTRIS data level 1b and the final temporally aggregated dataset can be attributed as level 2b.



RC1.2 – Conciseness and manuscript length

The second major issue is conciseness. The manuscript reads more like a thesis or project report. Very substantial editing is required. For example, The introductory Sections 1 and 2 currently take up almost 7 pages. A lot of this is repeated content. To take only one example, the paragraph on line 30, page 4 simply repeats the same statements in the list above it. There are also many instances where the same references are repeatedly cited. In addition, there is a lot of irrelevant content. For example, Section 2.4 discusses low cost sensors in detail, even though these sensors are not part of the presented research results. By removing irrelevant details and repeated information it should be possible to drastically reduce the length of the manuscript without losing any of the essential details.

Response:

Thank you for this comment. We agree that the original Introduction and Section 2 were too extensive and contained repeated material. We have therefore substantially shortened and refocused both sections. The revised Introduction now moves more directly from the research-infrastructure and regulatory QA/QC context, in particular ACTRIS and GAW, to the concrete operational QC gap addressed by this study. Repeated positioning statements and material not required for this argument were removed or consolidated.

We also removed the former dedicated subsection on low-cost sensor networks from Section 2. The revised state-of-the-art section is now focused on relevant QA/QC frameworks, regulatory monitoring, existing automated QC tools, aerosol-instrument-specific QC and harmonisation, and the resulting research gaps. Transferability to

regulatory or low-cost monitoring systems is retained only briefly in the Discussion/Outlook as a future application of the modular architecture, with the explicit qualification that instrument-specific parameterisation and validation are required. The broader research-infrastructure context was retained, but condensed, because it provides the motivation for interoperable and operational QC.

RC1.3 – Basic flagging scheme vs. ACTRIS/EBAS flags

The actris_qc package defines a basic flagging scheme with only three categories (GOOD, DOUBTFUL, BAD). It is mentioned that users could supply their own flag dictionaries on top of this scheme for their own specific needs. However, it is not clear how these three basic categories could map onto the much more detailed flagging scheme used for ACTRIS/EBAS data submissions.

For example, the workflow provides no mechanism for distinguishing values that exceed the fixed DOUBTFUL and BAD thresholds due to instrument malfunction from values that exceed them due to genuine extreme atmospheric and/or contamination events. The means the workflow does not distinguish between data that is BAD because the instrument was malfunctioning and data that is BAD because it is measuring a non-representative signal - a distinction which is formally encoded in the ACTRIS/EBAS framework as the difference between invalid and contaminated data. Contaminated data may be scientifically valuable for various reasons even if it should be excluded from background climatology. Collapsing both situations into a single BAD flag discards potentially useful information and may produce AI training datasets with artificially truncated distributions that underrepresent genuine high-concentration events.

Response:

Thank you for this very valuable comment concerning the ACTRIS/EBAS flagging scheme. We agree that simply assigning BAD or DOUBTFUL to a data point discards important information which should be available for the user and further downstream tasks. Therefore, we introduced the ACTRIS/EBAS flagging scheme with an EBAS quality flag and quality description. Each test can now be configured in the qc_settings.yml file, where the user can further specify the corresponding EBAS flag and the mapping of BAD and DOUBTFUL to the them. A first mapping was accomplished according to our evaluation but should be discussed within the ACTRIS community.

Our matching and configuration ships with the package and can be examined in the user documentation under section 'Package settings'.

RC1.4 – Share of DOUBTFUL/BAD flagged data (24 %)

For the example datasets used in the study, it is reported that 24% of data was flagged as DOUBTFUL or BAD. It would be useful to know if this is expected behavior and standard for this type of dataset, and to discuss the implications for ACTRIS data reporting requirements.

Response:

The data do not represent the typical data quality expected under normal operating conditions. The European Centre for Aerosol Calibration and Characterization (CAIS-ECAC) requires an overall measurement data coverage of more than 90% (ACTRIS Standard Procedures). Experience shows that, under normal conditions, only a few percent of the data are typically excluded during quality control (QC). The relatively high proportion of data flagged as DOUBTFUL or BAD (24%) in this study is due to the deliberate selection of examples intended to demonstrate the capabilities of automated detection methods.

RC1.5 – Flag cause field always set to OTHER

The manuscript mentions that the workflow produces harmonized, FAIR-compliant QC flags with full provenance, but inspection of the actual CSV outputs reveals that the flag cause field is populated with OTHER for every flagged data point without exception. Specific cause codes used by the community are available, for example, in the ACTRIS vocabulary (INSTRUMENT_ERROR, MISSING_VALUE, ...). Is there a reason such codes have not been introduced and used?

Response:

The output files now contain full information about the quality control checks. Three different CSV files are exported per instrument:

- *A file after stage 2 containing flags which have been set by stage 1 and 2 with information about the quality flag (BAD, DOUBTFUL, NIL), quality cause in a short version (e.g. ISOLATED_SPIKE), the quality comment referring to the triggered SaQC test (e.g. {"test": "setFlags", "comment": "Local Outlier Factor (LOF) parameters for downward-spike detection on eBC channels (used together with a below-rolling-mean check)."}, the ebas flag (e.g. 459) and the ebas description (e.g. Extreme value, unspecified error).*
- *The second file is exported after stage 3 and temporal aggregation to 1min. It still contains flags and corresponding description originating from stage 3.*
- *The final exported file only contains the final cleaned data aggregated to 1h (default, user can specify in instrument.yml) without any flags or corresponding comments.*

The ACTRIS vocabulary has been introduced together with the EBAS flags. Each instrument directory in the package contains a thresholds.yml where the parameters

of the tests can be set including the quality flag and cause and ebas flag and description.

RC1.6 – Applying the QC flags back to the data

Also missing is discussion of how the how the generated QC flags written to CSV should flow back into the workflow and be applied to the data. This is a non-trivial step that requires thoughtful design.

Response:

The flags from stage 1 and 2 are applied within the workflow, which is configurable per instrument. Before the 1-minute means of Stage 3, values with the flag levels listed in pre_aggregation_exclude_flags in file instrument.yml (default DOUBTFUL and BAD) are excluded. If the user decides to keep the DOUBTFUL data points, this setting can be changed to DOUBTFUL only. The final aggregated product (e.g. hourly means, files <channel>_final_<freq>.csv) is computed only from the remaining values.

The EBAS export applies the flags following the EBAS rules: values with an invalid flag code are removed, values with a valid code are kept together with their flags.

RC1.7 – Missing operator feedback loop

One consequence of this is that there is no implemented feedback loop between the continuous QC output and an operator review workflow. The manuscript implies a seamless path from automated flagging to FAIR-compliant data products, but the actual system appears to have no tooling for an operator to query recent flags, confirm or dispute automated decisions, or trigger targeted reprocessing of a specific window without modifying source code. This gap between the described capability and the implemented reality should at the least be acknowledged as a significant limitation of the current software release

Response:

Thank you for pointing towards increased usability of the tool. We agree that feedback loops help to parameterize the QC tests and improves the evaluation of written flags. Therefore, we implemented a html-report which gets automatically written after each QC call, independent from which source (csv, database) or which period (continuous, time range or anomaly). The html-report includes information about the QC-run and configuration, the used software versions, and systems information. A summary table at the beginning provides an overview of how many data points have been declared as BAD or DOUBTFUL for each channel and instrument. In the next sections, this information is

visualized in several plots, each individually representing a stage in the QC workflow, while afterwards the full time range and corresponding flags are presented. Finally, information about the run time and performance is provided. If the user does not agree with the written flags, they can now change the parameterization in the qc_settings.yml file and rerun the QC workflow for specific time windows either from a query to a database or based on a CSV file import from disc.

An Example of the HTML report is attached at the end of this document.

RC1.8 – Empty comment field and missing flag legend

The comment field in the output CSV is always empty for flagged data points. The descriptive label strings defined in the code (for example "BAD:flagRange: flow_1 < 3.7 or flow_1 > 4.05") appear to be used internally by SaQC for plotting but are never written to the output file. As a consequence it is impossible to determine from the CSV output alone whether a flagRange entry was triggered by the BAD threshold or the DOUBTFUL threshold, or which source variable caused a transferFlags entry.

The QC flag values exported to CSV are numeric codes with no legend, mapping table, or reference specification saved alongside them. A downstream user cannot interpret the flags without reading the source code, which is inconsistent with the FAIR Reusable principle that data should be self-describing.

Response:

The comment filed is now populated with information. Please regard our answer to comment 1.5. Beside verbose information in the output files, the user can now modify the parameters of the QC run in the qc_settings.yml file. This file also contains the matching of BAD and DOUBTFUL to corresponding Ebas flags. The python package saqc-atmo ships with a SaQC-parameterization developed in our work which was matched with the current EBAS/ACTRIS guidelines. We recommend to use our parameterization, which is presented in the paper and also in the user documentation, and adapt if changes are required but we encourage the ACTRIS community to optimize and harmonize the parameterization of the SaQC tests in the future. We would like to emphasize that all tests can be repeated with a new parameterization and historic data can be flagged with new updated settings files making the whole workflow reproducible and transparent according to FAIR principles.

RC1.9 – Threshold documentation and configuration provenance

A key goal of this work is that the pipeline is transferable to other stations. However, we see several major barriers that will hinder this.

The decision to keep scientifically grounded QC thresholds out of the operational config.yml is defensible and arguably preferable to exposing them as freely editable

parameters. However, the current implementation scatters these values across hundreds of lines of processing code with no central documentation, no inline source references, and no mechanism for distinguishing universally applicable thresholds from those that were empirically derived for TROPOS and may require adjustment at other sites. This problem could be solved, for example, by consolidating all thresholds into a dedicated, well-documented module or parameter file where each value is annotated with its scientific source, units, and applicability scope.

The configuration file used to produce a given QC result is never saved alongside the output. If thresholds are adjusted between runs there is no mechanism to determine which parameter values produced which output file. Combined with the absence of a software version identifier anywhere in the output, this means the outputs cannot be unambiguously traced back to a specific code and configuration state, which is inconsistent with the FAIR Reusable principle of detailed provenance.

Response:

We agree that the manuscript should distinguish more clearly between transferability of the workflow architecture and transferability of individual, instrument- or site-specific thresholds. This distinction is now explicit in the revised Introduction and Discussion/Outlook: extension to other monitoring systems requires dedicated parameterisation and validation, and the thresholds used for the AE33 and Aurora 4000 are not presented as universally transferable.

In the first place, it was our decision to incorporate the parameterization of the QC run directly in the scripts and keep the number of settings files low but we agree that tracking individual flags and adaptations to the parameterization (=thresholds) was difficult. Therefore, as already mentioned before, we have introduced a qc_settings.yml file for each instrument which sets all relevant parameters of the SaQC-tests, a unique check number for each, flags, cause, description, EBAS flags and comments as well as a scope, differentiating between universal and tropos_empirical. An example block looks like this:

```yaml
flow_total:
  description: "Combined/control sample flow rate acceptable range"
  unit: "l/min"
  scope: universal
  source: "AE33 manufacturer manual, nominal flow specification"
  check: S1-03
  test: "flagRange"
  cause: BELOW_OR_ABOVE_MIN_MAX
  ebas_flag: "459"
  bad:
    min: 4.8
    max: 5
    flag: BAD
    label: "BAD:flagRange: {field} < {min} or {field} > {max}"
```

A file called provenance.yml written after each QC run summarizes the parameterization and the full configuration folder is optionally saved after each run, which can be switched off in the run.yml file.

RC1.10 – Noise-detection parameters hardcoded to serial numbers

The noise detection parameter optimization files are hardcoded to specific device serial numbers: noise_ebc_w1_ae33_sn706_OPT.csv for the aethalometer and aurora4000_sn213450_OPT.csv for the nephelometer. The Melpitz device (ae33_sns0700705) has no corresponding optimized parameter file, which presumably means the noise detection will fail or fall back silently for that instrument at runtime (see further discussion below).

Response:

Yes, this is true. The noise detection files where directly created from a manual flag-by-click workflow corresponding to the specific dataset and instrument. We did not transfer the resulting parameters from the TROPOS site to Melpitz instruments, since we observed a different noise level for both instruments. This is also mentioned in the paper in section 'Noise detection with calibrated parameters'. For the future, we propose finding specific noise parameters for each instrument type and afterwards transferring these to other instruments at other sites. The workflow for determining the noise parameters is now user friendly, directly incorporated in the tool so that the station operator can produce these values easily. Please find a detailed description further below and in the new user documentation shipped with the module, which is online available.

RC1.11 – Discrepancy in the lower eBC threshold

There is a discrepancy between the documented lower eBC threshold (-0.075 µg/m^3) and corresponding implemented value (-0.0075 µg/m^3). It is unclear which value was used in the results presented in the manuscript.

Response:

Thank you for identifying this error. We changed it to -0.075 in the new qc_settings.yml files.

RC1.12 – Only two measurement channels covered

One general limitation of the workflow is that it is applied to only two measurement channels, i.e., the shortest wavelength channels of each of the instruments (ebc_w1 at 370 nm; sca_450 at 450 nm). This is actually a quite major limitation that should be more clearly discussed (see general comment above), including its implications for multi-

wavelength scientific and regulatory use cases. In the context of the configuration issues, it is currently unclear whether and to what extent the developed parameterizations can be transferred to other channels, given that artefacts like LED degradation, detector drift, and filter inhomogeneity can be wavelength dependent.

Thank you for motivating the QC checks for additional channels! The package now checks all channels: the seven AE33 eBC channels (370–950 nm) and the six Aurora 4000 channels (sp and bsp at 450, 525, 635 nm).
Flags are routed depending on the respective data:

- *Stage-1 control checks (flow, status, temperature, humidity) are transferred to every channel.*
- *Stage-2 checks (range, constants, detection limit, spikes, noise, ATN of the channel's wavelength) flag only their own channel.*
- *Stage-3 Ångström-exponent checks are transferred back to the two channels of each pair.*

We agree that the transferability has not yet been demonstrated. The AE33 values of the other channels currently equal those derived at 370 nm, and noise calibrations exist only for eBC370 and the Aurora channels. We now discuss this limitation and its implications for multi-wavelength use in the revised manuscript in section 'The TROPOS calibrations'.
Details can be found user documentation in sections "Workflow in detail → Flag routing".

RC1.13 – Anomaly catalogue: circularity and construction

The anomaly catalogue plays a dual role as both training data for parameter optimization and evaluation benchmark, which creates a circularity that the Melpitz validation only partially resolves. The paper would benefit from a clearer description of the catalogue construction process - how many distinct anomaly periods were identified, what fraction of the total record they represent, and whether the taxonomy was developed prospectively or retrospectively. It would also strengthen the work to clarify whether the root causes assigned to each anomaly (e.g. filter change, LED instability) were confirmed from instrument event logs or inferred from the signal alone, since this affects how confidently the parameterization can be expected to transfer to sites without the same maintenance history.

Response:

Thank you for motivating further clarification on the development of the quality control parameterization. We have added a description of how the catalogue was built (Section Event Classification) and state its limitations explicitly.

The catalogue was built retrospectively as well as prospectively. For the retrospective part (mainly stage 2 and 3) experts at TROPOS screened the historical record of the AE33 and the Aurora 4000 and selected periods that each show a typical problem: constant values or drops, outliers, noise, or high inlet humidity, AAE relations. The anomaly types were

named after this screening. The catalogue contains 12 periods at TROPOS (7 AE33, 5 Aurora 4000) and 13 at Melpitz (5 AE33, 8 Aurora 4000). The development of the catalogue of the flags from the device control stage 1 were driven prospectively, mainly assigned based on literature such as the manuals of the devices.

The causes assigned to the periods were inferred from the signal and from the instrument's status and housekeeping channels. They were not confirmed from station logbooks or instrument event logs. We now state this in the manuscript and describe the causes as interpretations. Whether the parameterization transfers to other sites therefore has to be tested with their data. The catalogue is designed for this: every station can add its own periods to its configuration.

The circularity concerns only the noise check. Its parameters are the only ones fitted to labelled data, using hand-labelled patterns from two of the catalogue periods. All other checks use thresholds from the instrument manuals or thresholds set empirically by experts, and none of them was optimized on the catalogue. We added a respective section explaining the purpose of the anomalies under the Event 'Classification – Anomaly Catalogue' section.

RC1.14 – Anomaly catalogue in the operational workflow

It is also uncertain how it is envisioned that the anomaly catalogue fits into the intended operational workflow. Inspection of the code reveals that anomalies are hardcoded entries in a Python source file (examples/anomalies.py), which means that adding a new anomaly requires editing and redeploying source code. For an operational system this is not sustainable. More importantly, the anomaly system is a manual reprocessing mechanism, not automated anomaly detection: it requires a human to have already identified a problematic time window before the system can process it. This appears to be inconsistent with the aim of developing an automated, real-time QC system. These are acceptable (and perhaps necessary) limitations, but they should be explicitly noted.

Response:

Thank you for this hint. We agree that the role of the anomaly catalogue was not described clearly. In the current version, the anomalies are no longer defined in source code. They are a YAML file (anomalies.yml) in the station's configuration folder. Each entry has an id, a name, a description and a time range. New anomalies are added by editing this file; no code is changed or redeployed. A run selects them with --period anomalies --anomalies bc1,bc2, or with all which can be configured in the run.yml as well.

The catalogue is not part of the operational QC and does not detect anomalies. It is a collection of reference cases: periods that we identified manually and that each show a typical problem, such as constant values, drops, outliers, noise or high inlet RH. We used it to develop, parameterize and evaluate the automated checks. The new version of the

tool (saqc-atmo) performs a QC on each anomaly as its own run, with its own output folder, report and flag statistics.

In operation, the QC runs in the continuous mode (or over fixed time ranges) without any anomaly catalogue. The automated checks find problematic data without prior knowledge of where they are. The catalogue can also be used to reprocess a known period, for example after maintenance, but that is an optional manual use and not the automated workflow. We describe the catalogue as a benchmark that other stations can extend with their own cases to check whether the thresholds suit their site.

RC1.15 – Transparency of the parameter optimization

The parameter optimization procedure for noise detection algorithm is lacking methodological details. The algorithm is a multi-objective genetic algorithm that searches over four parameters of the flagByScatterLowpass function. The parameters of this algorithm were optimized against a manually labelled ground truth dataset that was obtained using this flagByClick functionality. This manually labelled ground truth is not included in the data repository. Furthermore, no details of the optimization procedure are provided. Therefore, it is not clear if or how this process could be reproduced for other sites. One can only assume with some difficulty, since the optimization workflow was not repeated for Melpitz or for any other device. These limitations should be discussed.

Response:

Thank you very much for pointing to this issue. We agree, that improvements were necessary in this section. Meanwhile, the noise calibration has become a documented, reproducible part of the SaQC software and is implemented in out saqc-atmo python tool. The user documentation section "Calibrating the noise check" describes the workflow. It uses the new calibration procedure built into saqc (≥ 2.10) and runs in four steps, each one command:

1. `saqc-atmo-make-pattern:` *select a period of a channel and mark noise and stretches to ignore by hand (*`flagByClick`*). The result is the labelled ground truth (pattern file).*
2. `saqc-atmo-calibrate:`
   fit window, thresh, sub_window and sub_thresh of flagByScatterLowpass with saqc's calibratePipeline (genetic algorithm, pymoo; population 100, 2000 evaluations by default). The fit minimizes false positives and false negatives against the marks; several runs can be made and the best is kept.
3. `saqc-atmo-inspect-pattern:` *compare the marks with the flags of the fit.*
4. *The QC run uses the fit of each device and channel automatically.*

The hand-labelled patterns and the fitted parameters of the TROPOS devices (AE33: eBC370 and flows; Aurora 4000: all six channels) are now included in the repository. A corresponding example with data, pattern and reference fit is available for reproduction (examples/optimization). The documentation also describes:

For a new site, these parameters can be transferred or new parameters can be found. If the user defines own patterns and runs the optimization, the fits are stored in its configuration folder and are available for the noise test. Without a fit, the noise check is skipped and the log says so; the other checks run as before. We agree that the procedure has not yet been repeated for Melpitz or other devices. We now discuss this, together with the dependence of the fit on the operator's marking, as a limitation in the revised manuscript in section 'Noise Detection with calibrated parameters'.

RC1.16 – Missing automated tests

The README file in the software package references a tests/ directory with conftest.py and test_data_manager.py, but these files are absent from the actual codebase. For a software package submitted as a companion to a scientific paper, intended for operational deployment, and claimed to produce reproducible and FAIR-compliant outputs, the complete absence of automated tests is a worry. We suggest that at the least, tests that run the pipeline against the bundled example data files and verify the output format and flag scheme should be added.

Response:

Thank you for motivating tests for the python package. Writing tests is a very complex and important task for each shipped software, which we accomplished with the help of the AI assistant (Anthropic, via Claude Code). The package now has an automated test suite (tests/, about 230 test cases), described in the README section "Tests" including a full CI pipeline including user documentation deployment.

We first developed and evaluated the scientific core manually: the selection, implementation and parameterization of the SaQC tests before the submission of the first manuscript. This produced a reference data set of input data and the corresponding quality flags. A regression test runs the complete pipeline on the bundled example data. It compares every output with this reference.

A contract test runs every instrument end to end. It checks the configuration files (saqc causes, EBAS codes, flag levels), unique check IDs, and the result files and report. Further tests cover the data sources, the EBAS export, chunked and parallel runs (same flags as a single run) and the evaluations.

GitLab CI runs the tests on every change, using the pinned dependency versions. It also builds the documentation with warnings treated as errors.

The tests, the CI, the documentation and further features were developed with the help of the AI assistant Claude (Anthropic, via Claude Code). We reference flags served as ground truth throughout. Every change had to reproduce them in the regression test, and any intended change of results was reviewed by the authors before the reference was updated. The authors checked all workflows and reviewed the code.

RC1.17 – Silently swallowed exceptions

It also appears that exceptions during anomaly processing are caught, printed to stdout, and silently swallowed - the pipeline continues to the next anomaly and run.py always

exits with code 0 regardless of whether any data was successfully processed. This means a silently failed QC run produces a data gap with no alert, no structured failure record, and no non-zero exit code for monitoring systems to detect. This limitation should be addressed or at least discussed.

Response:

Thank you for bringing up this issue. In our first attempt, this has been a workaround for unresolved coding issues. In the updated version, this has been accounted for: each error or occurring problem is caught by exceptions and reported in the stdout and saved to a log.txt.

The QC pipeline exits with status 1 when an instrument of a period failed, when no instrument had data, or when the configuration is invalid. Monitoring systems such as cron can therefore detect a failed run.

Structured record. Every period writes run_summary.json with status (OK/FAILED), the failed instruments, all warnings and errors, and the instruments without data. The traceback goes to the run's log file, and the HTML report lists warnings and errors with a status badge.

A failing instrument does not stop the others, but it is recorded as a failure, not skipped. An instrument without data while another has data (e.g. a device switched off) is reported as a warning. In chunked or parallel runs, a failed chunk marks its period as failed; a crashed worker process is rerun and its crash report logged.

RC1.18 – Undefined acronyms (P1, L31)

P1, L31: Many acronyms used without first being defined.

Response:

Thank you. We have systematically reviewed the manuscript for undefined and inconsistent abbreviations and revised the first occurrence of acronyms throughout the text. Central terms and institutions are now introduced explicitly before the abbreviation is used (e.g. SaQC, WMO/GAW, EPA, AERONET, NOAA, CAIS-ECAC, RH, SAE, CSV, UV and NIR), while unnecessary one-off acronyms were written out to reduce acronym load. We also harmonised repeated terminology, including the use of CAIS-ECAC, and corrected several local labels and abbreviation definitions in the instrument and QC-method sections.

RC1.19 – Details on data sources (P10, L21)

P10, L21: Please provide more details on the data sources. E.g., the data loggers used and whether any previous processing steps were applied to their outputs.

Response:

The data were collected using data loggers specifically developed for ACTRIS Near-Real-Time (NRT) data acquisition as part of a project funded by the Copernicus Atmosphere Monitoring Service (CAMS) [Reference]. These data loggers store the raw measurements locally and transmit Level 0 data to the EBAS database. For the present study, the locally stored raw data were used. These data are identical to the ACTRIS Level 0 data and have not undergone any corrections or further processing. We will add corresponding text passages to the manuscript.

RC1.20 – Meaning of "exceeding values" (Fig. 2)

P11, Fig. 2: It is not clear what 'exceeding values' mean.

Response:

We refer to data points exceeding the axis limits. We add respective explanation to the image.

RC1.21 – ACTRIS aerosol in situ vs. other ACTRIS components (P13, L5)

P13, L5: It should be clarified here and throughout the manuscript that this 'ACTRIS aerosol in situ' solution. ACTRIS has 5 other components that use their own QA/QC workflows, some of which are already quite developed and probably worthwhile discussing given their similarities to the present case (e.g., the gas phase in situ QC workflow).

Response:

We agree that the scope should be stated more precisely. The revised framing now makes clear that ACTRIS provides the reference application and validation context for the aerosol in situ use case considered here; the workflow is not presented as a QA/QC solution for ACTRIS as a whole.

ACTRIS comprises six observational components and corresponding Central Facilities: the Centre for Aerosol In Situ (CAIS), the Centre for Aerosol Remote Sensing (CARS), the Centre for Cloud In Situ (CIS), the Centre for Cloud Remote Sensing (CCRES), the Centre for Trace Gases In Situ (CiGas), and the Centre for Trace Gases Remote Sensing (CREGARS).

The workflow presented here is illustrated using selected variables within CAIS as examples. Other Central Facilities use different workflows, reflecting both historical developments and differences in data structures (e.g., dimensionality), quality assurance methods, and other requirements. To our knowledge, no publications describing the use of SaQC in other ACTRIS Central Facilities are currently available.

Please find according text adaptations at the beginning of section 'ACTRIS Specific QC Solution'.

RC1.22 – "Station-level" / "instrument-level" wording (P13, L11)

P13, L11: This workflow can't be described as 'station-level', or not even 'instrument-level' for that matter, given that currently it only operates on single measurement channels.

Response:

Thank you. We agree, that our first version did not cover all channels of an instrument. The new version comprises checks for all channels of both the aethalometer and nephelometer instruments. Therefore, we are confident that this workflow can be framed as 'instrument-level'. For the future, this tool is intended to be a 'station-level' solution which requires additional instruments to be added and corresponding SaQC-tests to be parameterized, consequently we remove the station-level claim.

RC1.23 – Resampling of 10 s flags to 1 min (P14, L42)

P14, L42: How are the 10 sec level flags resampled to 1 min?

Response:

Stage 3 works on 1-minute means of each channel, computed from the samples at their original time stamps, which carry the flags of Stages 1 and 2. The resampling uses SaQC's resample:

Samples which were flagged with one of the levels specified under the key pre_aggregation_exclude_flags in instrument.yml (default DOUBTFUL and BAD) are masked before averaging. The 1-minute mean is the mean of the remaining samples and mapped to the end timestamp of the 1-minute interval.

Each 1-minute value gets the highest flag of the samples in its minute (SaQC flag aggregation "max"). This is conservative: a minute that contained a flagged sample stays marked, i.e. if only BAD values are excluded by pre_aggregation_exlude_flags, minutes aggregates with doubtful values will keep the doubtful flag.

The Stage-3 checks (Ångström exponents) flag the 1-minute means. Their flags are projected back onto all original samples of that minute (EBAS level 1).

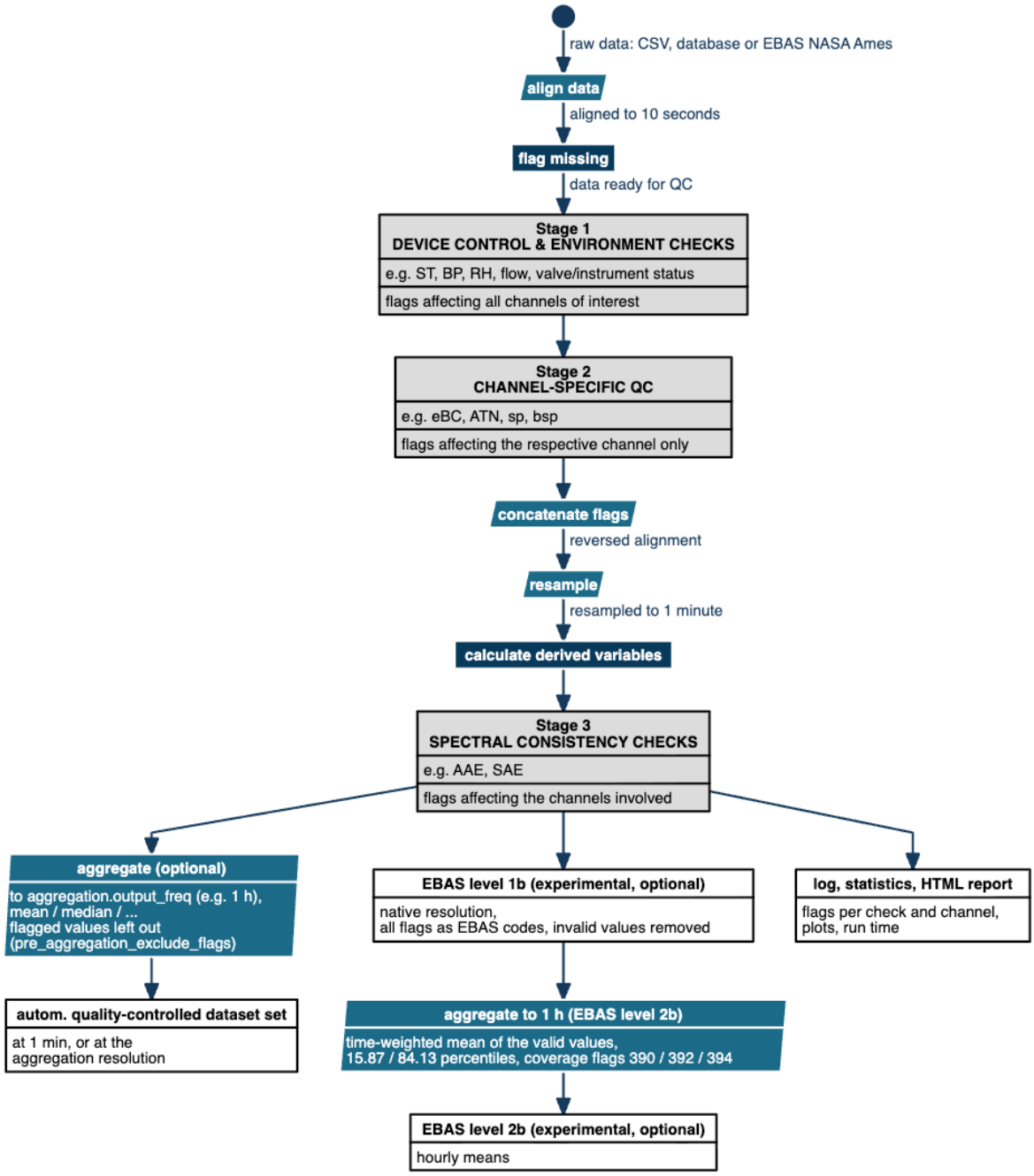
Details are described in the user documentation, "Workflow in detail" (Stage 3; "Flag routing", "Back to the original time stamps"), and in the comment on pre_aggregation_exclude_flags in instrument.yml as well as in the manuscript in section 'Flag Propagation and Finalization'.

RC1.24 – “Final quality assured dataset” (Fig. 5)

P15, Fig. 5: The output cannot be considered a 'final quality assured dataset' given the limitations discussed above.

Response:

We agree that the wording has to be adapted in this place and changed it to: autom. quality-controlled dataset. According to our response to comment 1.1, we added in the manuscript that according to ACTRIS in situ data production level 1b has been reached after Stage 3 and optionally level 2b can be reached if the user aggregates the quality assured data at the end of the pipeline by setting respective keys in instrument.yml. The new workflow diagram shows the following content:



RC1.25 – ATTN values between 100 and 120 (P17, L2)

P17, L2: What happens for ATTN values between 100 and 120?

Response:

Thank you for careful consideration. That was an error in the manuscript. We changed it to 80-120: DOUBTFUL. The settings can now be adjusted for each channel separately in the qc_settings.yml file.

RC1.26 – Determination of the AE33 threshold (P17, L9)

P17, L9: How was this threshold determined for this AE33 unit?

Response:

Flats and constant values were flagged with a threshold of 0.012 and a window size of 12 minutes, assuming that the maximum total change of a rolling window within 12 minutes does not exceed 0.012.

RC1.27 – Aurora 4000 zeroing periods (Fig. 7)

P21, Fig. 7: Panel a shows that the Aurora 4000 instrument zeroing periods are included in the input data. This requires further discussion. Such data should be flagged independently as it should be possible to isolate and analyze these data independently.

Response:

Thank you for this hint! We agree that the zeroing period is included and extended the QC to flag these data points. Fig. (now) 6a shows the input data, which still contain the zeroing periods. The QC now flags the zeroing periods of both instruments on their own, so they can be isolated by their flags. Both instruments report their operating state as status codes. For each code, its meaning, flag and EBAS code are set in the instrument's qc_settings.yml (following the manufacturer manuals) and listed in the user documentation (Configuration, "Status codes"). All zeroing periods are flagged BAD with EBAS code 686 ("invalid due to zero check") and a label naming the state. They are flagged in Stage 1 and transferred to all channels of the instrument.

Aurora 4000: zero calibrations, zero checks and zero offset adjusts (major_state 2, 4, 5) are flagged 686, and so is the zero pump in the pump and valve state (digital_io_state). Span checks and calibrations get 687. The settling time around each period is flagged separately: 5 min DOUBTFUL before and 20 min BAD after. Checks S1-02 and S1-03.

AE33: the clean-air cycles (valve position 1100) are flagged 686, and so are the stability and clean-air tests reported in the instrument status (codes 1024, 2048). They get the same settling-time handling: 5 min DOUBTFUL before and 30 min BAD. Checks S1-04 and S1-06. The clean-air cycles are also evaluated (check S1-07): each cycle is compared with the earlier ones, and a deviating cycle marks the following data DOUBTFUL (EBAS 559, contamination suspected). The cycles also give each channel's detection limit.

The zeroing periods can therefore be selected by EBAS code 686, by check ID or by label, both in the result files (<channel>_stage2_flags.csv) and in the EBAS level-1 files.

RC1.28 – Term "signal-to-noise" (P22, L2)

P22, L2: Signal-to-noise appears to be the wrong term to use here.

Response:

Corrected to noise.

Reviewer #2

General comments

Houben et al. presents a data processing workflow and software for aethalometer and nephelometer observations. Given the high number of sites (ie potential users) and the need for harmonized QA/QC, this work is crucial for the aerosol in-situ community, not only for ACTRIS but also at a global scale. As such, it is very relevant for AMT.

As for Reviewer#1, I am reviewing the paper as a potential user of the software. I do have some concerns regarding the article, which are listed below.

Response:

Thank you for the positive assessment of the relevance of the work and for the detailed comments. We have revised the manuscript for conciseness and clarified the scope and positioning of the workflow; the specific technical points are addressed individually below.

We substantially improved the software with its new name saqc-atmo and included almost all requested functionalities from the review. We further implemented a set of settings files which can be adjusted by the user and are tracked throughout the QC pipeline and saved afterwards for reproducibility. Especially the parameterization of the SaQC-tests, the resulting flags and the matching to the EBAS flagging scheme is not always straight forward and needs scientific discussion. We encourage the reviewers as well as the ACTRIS community to debate about appropriate settings and agree on these for further downstream implementation and application, which should continue even after the publication of this manuscript. The required flexibility now ships with the module.

The Python package saqc-atmo is hosted on Helmholtz GitLab

https://codebase.helmholtz.cloud/qc-of-atmospheric-time-series/saqc-for-atmospheric-data

and will be archived on Zenodo upon acceptance of this manuscript. The version used in this work is 0.1.0.dev. The user documentation can be found on GitLab as well and ships with the module:

https://codebase.helmholtz.cloud/qc-of-atmospheric-time-series/saqc-for-atmospheric-data/-/jobs/3630681/artifacts/external_file/public/index.html

The data sets and associated quality-control outputs presented in this work, including flags (CSV), logs (TXT) and diagnostic figures (PNG), will be available via Zenodo upon acceptance. In order to inspect the results of the presented study based on the anomaly catalogue, the results were uploaded to the UFZ Nextcloud first

https://nc.ufz.de/s/kJk5CJzYRTc9n3G

pw: Kxs\tR82{W

RC2.1 – Manuscript length and low-cost sensor background

the manuscript is long, some sections should be more concise such the introduction or Section 2. For instance, I don't think background is needed for low-cost sensors since AE33 and Aurora4000 don't fit in this category.

Response:

Thank you. We agree and have substantially shortened the Introduction and Section 2. In particular, the former dedicated background subsection on low-cost sensors has been removed because low-cost sensors are not part of the AE33/Aurora 4000 evaluation presented in this study.

The revised Introduction reaches the concrete operational QC gap more directly, while Section 2 is now focused on the QA/QC frameworks, automated tools and aerosol-specific work needed to position the present study. Transfer to regulatory or low-cost monitoring systems is retained only briefly in the Discussion/Outlook as a future application of the modular architecture and is explicitly stated to require instrument-specific parameterisation and validation.

RC2.2 – Scope, recommendations and ACTRIS harmonization

I think the scope of the paper should be clarified. The authors present a workflow and a software package, but also, through detailed examples, a way to configure the QA/QC: which parameter to use, which thresholds, which flag. Therefore it is not clear wether the authors also provide recommendations for these as well, and how they are in line with current ACTRIS recommendations provided by CAIS-ECAC. Harmonization comes from using common tools and common protocols. As such, it is not clear as well if previous data should be re-evaluated with this tool.

Response:

We agree that the status of the presented workflow and parameterisation needs to be distinguished more clearly. The revised Introduction and Discussion/Outlook now separate the reusable QC architecture from instrument- and site-specific rules and thresholds. ACTRIS is described as the reference application and validation context, while the specific parameterisation used in this study is not presented as a universal ACTRIS recommendation.

The scope of this paper is to provide a proof-of-concept python package for the ACTRIS community as potential starting point for further development and customization within the ACTRIS community. We demonstrate how harmonizing quality control workflows with SaQC could be achieved and provide a practical tool in order to reduce the time required by station operators to perform and evaluate the QC of their data. We

encourage the community to test our approach and discuss our proposed QC workflow, its parameterization and evaluation and are open for feedback and additional feature requests in the future.

As of today, the ACTRIS QC workflow consists of a combination of manual and automated quality control procedures that are not fully harmonized across stations. For some parameters, predefined criteria exist, such as detection limits derived from the technical specifications of the instruments, whereas other procedures, such as outlier detection, are more subjective.

Developing a SaQC package for individual instruments would require the ACTRIS community to agree on common SaQC presets that account for the wide range of measurement conditions at different stations. Therefore, this manuscript presents a pilot study demonstrating a method for automated QC that can be applied consistently across stations, rather than a fully operational solution. The need for reanalysis of existing data can only be assessed by the ACTRIS community itself.

RC2.3 – Fit with the ACTRIS DMP and NASA-Ames format

On a more general aspect, it is not clear how the paper fits in ACTRIS Data Management Plan and existing softwares already in place. Does it handle NASA-AMES format ? Can it be used (or how can it be implemented) for level0, 1, 2 generation ?

Response:

Thank you for motivating the integration of NASA-AMES formats and the consideration of the ACTRIS Data Management plan. In the new version of the python tool (saqc-atmo) we integrated EBAS flags for each test which can be adapted by the user in the qc_settings.yml file. Furthermore, we explicitly describe how level 1b and level 2b datasets can be achieved and implemented the NASA-AMES reader and writer. This, however, is currently in an experimental mode and needs thorough checking by colleagues, reviewers and users to ensure correct formatting of the exported files. We appreciate your feedback on this as well as to the entire tool as well.

RC2.4 – Compatibility with the ACTRIS flagging system

How the "BAD", "DOUBTFUL", "VALID" flags can be compatible with the ACTRIS flagging system ? For instance, the authors flagged the data with a "BAD" flag if the inlet RH is higher than 40%. In ACTRIS, it should be flagged with 640 ("Instrument internal relative humidity above 40%"), but remains valid. For AE33, status !=0 leads to a "BAD" flag; this is far too reductive. There are status numbers which are only warnings, and do not report a malfunction of the instrument. Plus, the status variable results from a binary combination of different individual status which can be decoded. I think this is important to take this into account.

Response:

Thank you very much for this hint. We agree that an explicit status codes handling and corresponding EBAS flags should be integrated in the tool. The new version provides a qc_settings.yml file which holds the parameters of the SaQC tests as well as all additional comments, flags and labels so the user can change and adapt the settings according to their needs. We propose to apply the matching developed in our work which ships with the package but we encourage the ACTRIS community to discuss and agree on a best fitting parameterization including EBAS code matching for the respective instruments. Furthermore, every status code for both instruments is attributed a specific flag, cause, EBAS code and description which will end up in the final exported files. Furthermore, the status variables of the instruments were decoded so that an individual flags for each code can be assigned in the qc_settings.yml of an instrument.

RC2.5 – Demonstrating the real-time aspect

the real-time aspect of the workflow / software needs to be better demonstrated. NRT is a potential application, but the automated QA/QC can also be applied on an "offline mode", right ? To this regard, I would mention NRT application in the text, but not in the title.

Response:

We agree that the real-time capability has to be demonstrated rather than inferred from the general workflow description. In revising the Introduction and Section 2, we have clarified that the same QC concept can be used for continuous station operation and for retrospective reprocessing. We retain the real-time objective of the study; the additional operational demonstration and the resulting wording of the title are addressed below.

Furthermore, we agree that the QC workflow is not restricted to near-real-time (NRT) use: the same configuration runs offline or continuously. Offline, it QCs a fixed time range, all data of a source, or a list of named test periods, from CSV files, a database or (experimentally) EBAS NASA Ames files. All results in the manuscript were produced in this mode, on a static data set that we used for developing and parameterizing the checks.

For NRT use, the workflow has a continuous mode. Each run QCs the most recent time window (configurable, e.g. the last day) up to the time of the run, reading from the station database. Any database that SQLAlchemy supports can be used; we tested PostgreSQL and SQLite. Table names, schema and time column are configurable. The downloaded data are kept in a local archive, so each run only fetches what is new since the previous run, plus a configurable overlap. State that has to persist between runs is stored outside the run folders. This includes the history of the AE33 zero/clean-air cycles that the contamination check and the detection limit are based on. Scheduled, for example, every 10 minutes or every hour with cron, the workflow therefore QCs the incoming data

continuously. Every run writes its log, a machine-readable summary and a report, and it ends with a non-zero exit status if the QC failed, so that it can be monitored. Automated tests check that continuous runs download only new data and that QC from the database gives the same flags as from the CSV files.

We have added a section on NRT application 'Near-real-time-application' that describes this mode, its configuration and the run times, and it states that the workflow does not yet upload the resulting data and flags to any architecture.

RC2.6 – Local NRT processing and the ACTRIS DMP

p13-14, NRT is a bit discussed, but it is not clear how producing qualified data locally (ie at the station) is compatible with ACTRIS Data Management Plan ?

Response:

Thank you. We adapted the text according to our answers from comment 2.3 and 2.4.

RC2.7 – Meaning of the DOUBTFUL flag

In NRT (but also in "offline" mode), what does the "DOUBTFUL" flag mean ? can the data still be considered valid, or not ?

Response:

DOUBTFUL means the same in NRT and in offline mode: the value is questionable but not shown to be wrong, and it needs to be reviewed by the operator. Examples are values below the detection limit, suspected contamination after a deviating zero cycle, elevated sample RH, or a calibrated noise pattern. By default, DOUBTFUL values are treated conservatively. Like BAD values, they are left out of the 1-min means, the derived quantities (AAE, SAE) and the final aggregated output. Users can include them by setting pre_aggregation_exclude_flags: ['BAD']. Whether a value counts as valid in the ACTRIS sense depends on the EBAS flag code that each check attaches, not on the DOUBTFUL level itself. In the level-1 NASA-AMES export, DOUBTFUL values with a code that EBAS classifies as valid (147, 559, 640) are kept, and those with an invalid code (459) are removed. We have clarified this in Sect. 'Flag Propagation and Finalization'.

RC2.8 – User-friendliness and step-by-step guidance

as a potential user, I tried to implement the workflow and the software for my site. I must admit that I miserably failed. Clear step-by-step guidance is critical. I sincerely suggest the authors to put themselves in the users' place; and potentially users with strong atmospheric science background, but little coding experience. This is also how this tool can be a success story or not. User-friendliness and guidance should always be an important aspect of the development process.

Response:

We thank you very much for trying the software and for this extensive and valuable feedback, and we agree that user guidance decides whether the tool is used. Since the first submission, we have reworked the software and tried to improve its usability:

The installation can be accomplished with a single command (uv sync) installs the exact versions of all dependencies, including Python. The setup with saqc_atmo init <folder> creates a complete, commented configuration for a new station. saqc_atmo <folder> --show-config shows every setting and the file it comes from, and warns about misspelled keys. Configuration errors name the setting to correct.

An extensive user documentation was built and is now publicly available. It contains step-by-step guides for installation, a first run and setting up one's own station (paths, devices, data source, period). It also includes a configuration reference and a guide for adding a new instrument type.

Several examples for every case are included and come with sample data for every workflow (CSV, database, continuous QC, parallel runs, noise calibration).

The result of every run writes an HTML report with all flags and plots and a log that explains every check.

We would very much welcome your feedback again on the new version and on where the setup failed, so that we can address exactly those steps.

RC2.9 – Uncorrected vs. corrected eBC

p8, it is not clear what is "uncorrected and corrected eBC". eBC refers to a framed concept with proper recommendations and not "raw" BC data. It is not clear wether the proposed software is able to provide eBC concentrations as defined in Savadkoohi et al. (2024).

Response:

By analysing the difference in attenuation between the two spots, the instrument corrects in real time for the nonlinear saturation signal caused by the accumulation of particles – a known source of measurement bias termed the "loading effect" (Drinovec et al., 2015). For simplicity, throughout the manuscript, we use the term eBC to refer to loading-effect-compensated eBC values.

RC2.10 – Evaluating the compensation algorithm at spot changes

As suggested by the Editor, it would be very valuable to implement the automatic evaluation of the compensation algorithm by comparing absorption (or BC) before and after each spot change.

Response:

We thank the Editor and the reviewer for this suggestion and have implemented it, however this is currently in an experimental stadium and requires additional user feedback.

Every QC run now detects the filter (spot) changes in its period from the AE33 tape counter. For each change and wavelength it stores:

- *the mean eBC of the last 5 min of valid level-1 data before the change,*
- *the mean eBC of the first 5 min of valid level-1 data after it,*
- *the compensation parameter k and the attenuation (ATN).*

These values are collected across runs in a history at the station, and each change is stored only once. The evaluation command (saqc-atmo evaluate, e.g. run weekly from cron) then reports, for each wavelength:

- *Before/after ratio: the ratio of the two means at every change. A ratio of 1 means the compensation is correct, below 1 means under-compensation and above 1 over-compensation. The trend over the filter changes is fitted with a least-squares line (with 95 % confidence band) and with a robust Theil–Sen line; a slope different from zero shows a drift of the compensation.*
- *Trend-extrapolated ratio: a line fitted to 30 min of data on each side of the change and evaluated at the time of the change. This removes the bias from concentration changes across the 35–45 min gap that stays invalid around each tape advance.*
- *eBC against ATN within each spot: the relative slope of hourly eBC against ATN within each spot. Correctly compensated eBC does not depend on spot loading.*
- *k over time: the time series of the instrument's compensation parameter.*

Only data not flagged by the QC are used. Changes with too few valid points or too low concentrations are excluded. All thresholds are configurable. The results are written as tables, plots and an HTML report.

Ambient variability makes single changes scatter strongly, so the evaluation is meant for long periods: months of data, judged by the median ratio and the slope with their confidence intervals. Because absorption is eBC multiplied by a constant factor at each wavelength (MAC/H), the ratios are the same for eBC and for absorption. The evaluation describes the performance of the compensation; it does not correct the data or set QC flags, and the eBC reported by the instrument is used unchanged. Further information on the loading compensation can be found only on the respective site of the user documentation.*

RC2.11 – AAE from the full spectral dependence

It would also interesting to use the complete spectral dependance of BC to calculate AAE from ln(abs) vs ln(lambda), which can enable to check the quality of the regression (r^2), as recommended in Savadkoohi et al. (2025).

Response:

Thank you very much for motivating the integration of this concept. Currently, this implementation would exceed our time resources, therefore, we were not able to start working on this evaluation step. We agree, that additional evaluations are very assistant in order to judge the QC results and the data overall. We encourage you and the ACTRIS community to request additional features via the GitLab Issue functionality after publication of the manuscript. Thank you for your understanding.

QC report: qc-anomalies-tropos 2022-10-01 00:00 - 2022-10-04 00:00 UTC, anomaly bc2 OK

saqc-atmo 0.1.0 · written 2026-10-28 21:01 UTC · log file

Run

run	qc-anomalies-tropos
period	2022-10-01 00:00 - 2022-10-04 00:00 UTC (3 days)
selected by	anomaly bc2: drop and constants 2 (extended constant drop in data)
config	anomalies-tropos/config
command	saqc-atmo anomalies-tropos/config
output	anomalies-tropos/runs/qc-anomalies-tropos_20221001_000000-20221004_000000_anomaly-bc2
log file	anomalies-tropos/runs/qc-anomalies-tropos_20221001_000000-20221004_000000_anomaly-bc2/2026-10-28_21-00-50_log.txt

Software

saqc-atmo	0.1.0 git adb7b5d on main (2026-10-28 21:54, uncommitted changes)
python	3.14.7 (CPython, /Users/houben/phd/studies/032-aiamo/tropos/saqc-bei-atmosphaerischen-daten/execution-for-paper/20261028-anomalies/saqc_tropos/saqc-atmospheric/.venv/bin/python3)
saqc	2.10.0
pandas	3.0.6
numpy	2.5.3
scipy	1.18.1
scikit-learn	1.9.1
matplotlib	3.11.2
ebas-io	4.7.1
omegaconf	2.3.1
pyyaml	6.0.3
sqlalchemy	2.0.54
psycopg2-binary	2.9.13

System

host	mac-houben-757.intranet.ufz.de
platform	macOS-15.8.1-arm64-arm-64bit-Mach-O (arm64, 12 CPUs)
user / pid	houben / 30306
local time	2026-10-28 22:01:31 CET (UTC+0100)
working dir	/Users/houben/phd/studies/032-aiamo/tropos/saqc-bei-atmosphaerischen-daten/execution-for-paper/20261028-anomalies/saqc_tropos/saqc-atmospheric

Flags per channel

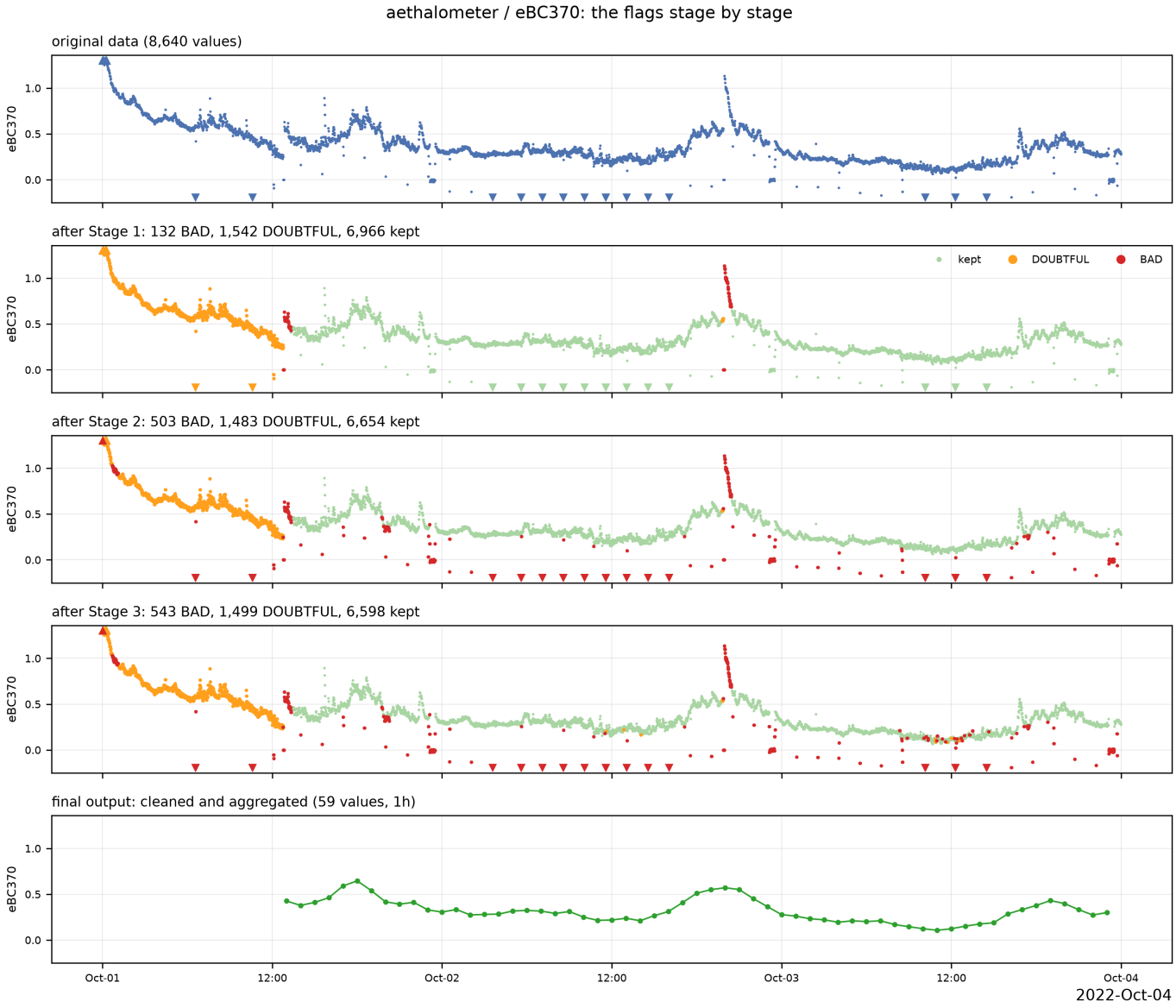
Final flags on the original time stamps (Stages 1-3); final = points of the final output (`_final_.csv`).

instrument	channel	points	BAD	DOUBTFUL	OK	NIL	flagged_percent	final_clear
aethalometer	eBC370 *	8,640	503	1,483	0	6,654	22.99	59
aethalometer	eBC470	8,640	933	12	0	7,695	10.94	72
aethalometer	eBC525	8,640	921	12	0	7,707	10.80	72
aethalometer	eBC590	8,640	484	13	0	8,143	5.75	72
aethalometer	eBC660	8,640	457	16	0	8,167	5.47	72
aethalometer	eBC880	8,640	490	14	0	8,136	5.83	72
aethalometer	eBC950	8,640	580	16	0	8,044	6.90	72
nephelometer	sp450 *	25,842	699	84	0	25,059	3.03	72
nephelometer	sp525	25,842	699	84	0	25,059	3.03	72
nephelometer	sp635	25,842	699	84	0	25,059	3.03	72
nephelometer	bsp450	25,842	880	104	0	24,858	3.81	72
nephelometer	bsp525	25,842	957	119	0	24,766	4.16	72
nephelometer	bsp635	25,842	1,144	154	0	24,544	5.02	72

aethalometer

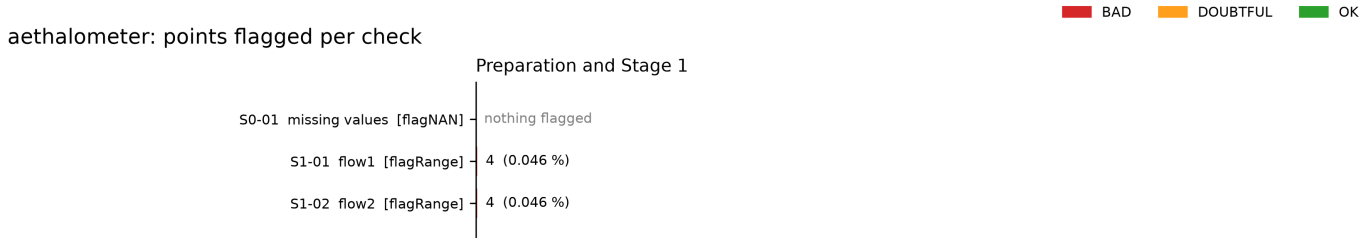
▸ configuration

eBC370 stage by stage



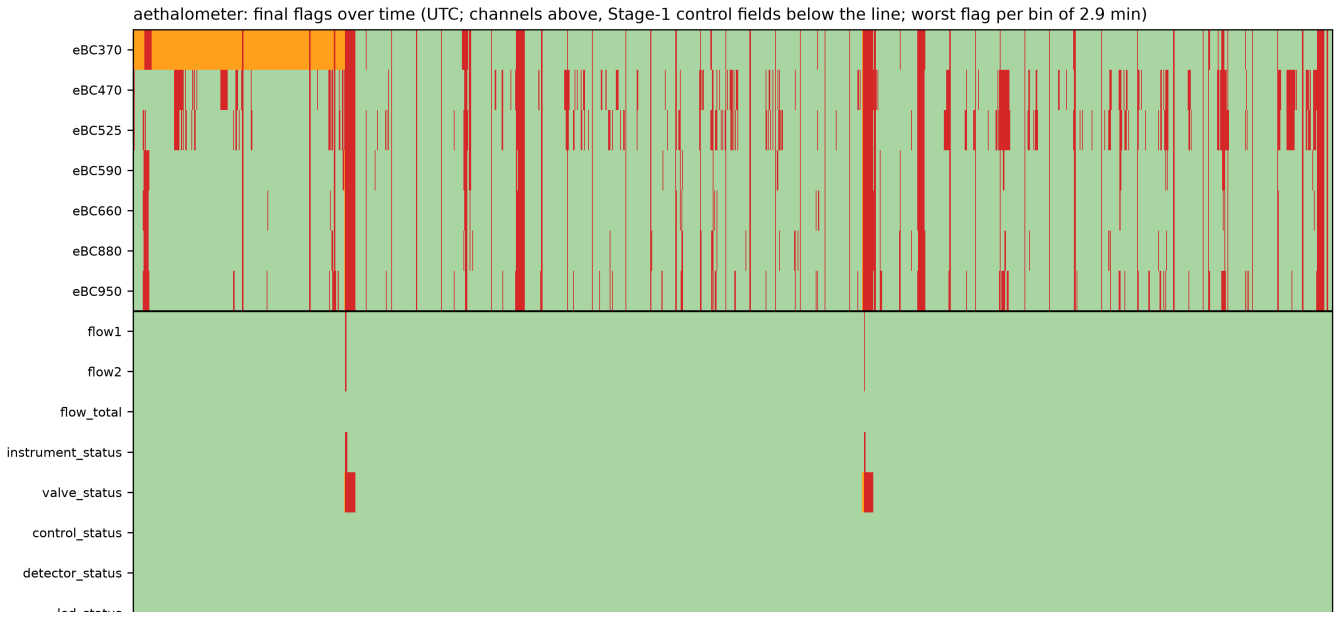
The flags stage by stage: original data, after Stage 1 (control data), Stage 2 (the channel's checks) and Stage 3 (derived quantities), each point coloured kept / DOUBTFUL / BAD; bottom: the final, cleaned and aggregated output. Values outside the 0.5-99.5 % range are triangles at the edge.

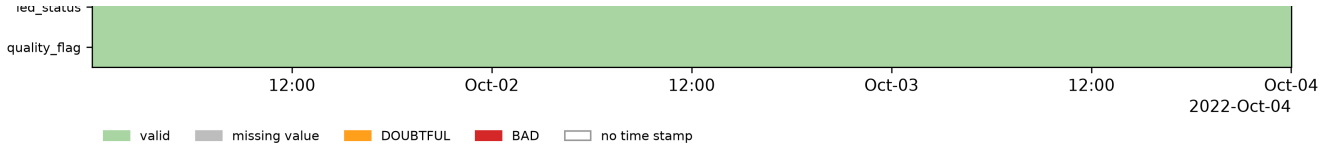
The other channels stage by stage (PNG files): eBC470, eBC525, eBC590, eBC660, eBC880, eBC950



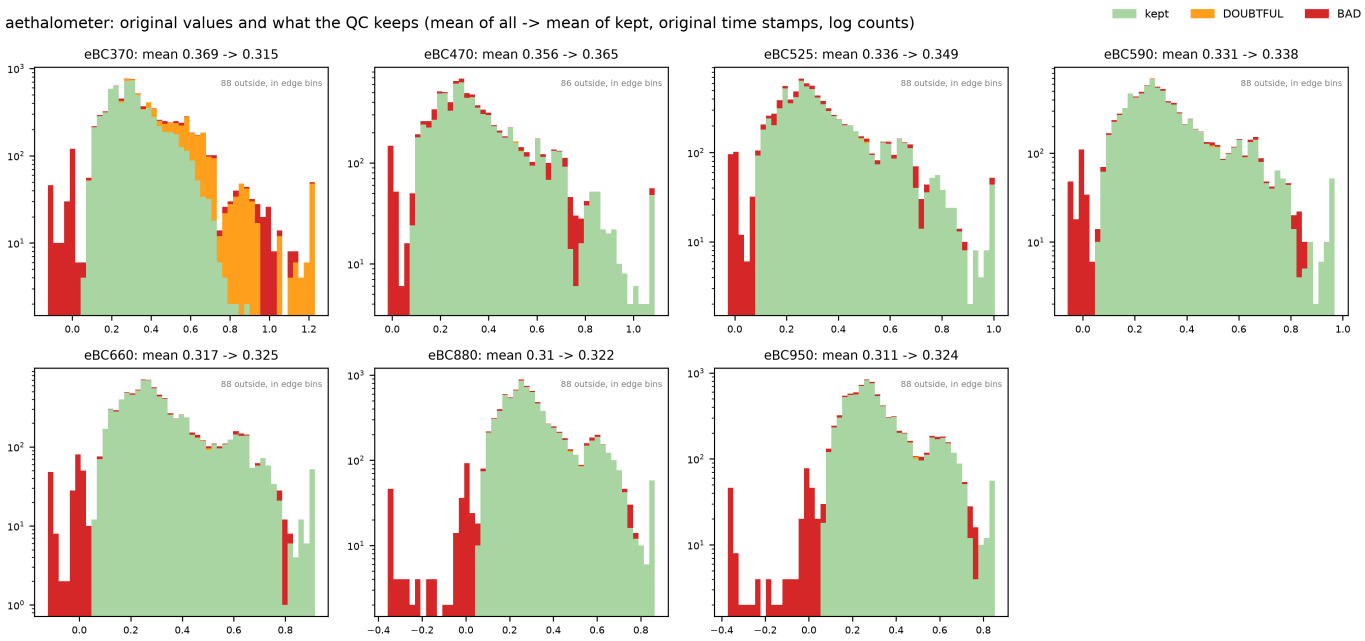


Points flagged by every check (Stage 2: B = BAD, D = DOUBTFUL, colour = % of the valid points of the checked field).





Final flags over time: worst flag per time bin; channels above the line, Stage-1 control fields below.



Distribution of the original values, split into kept / DOUBTFUL / BAD.

▸ all checks (table)

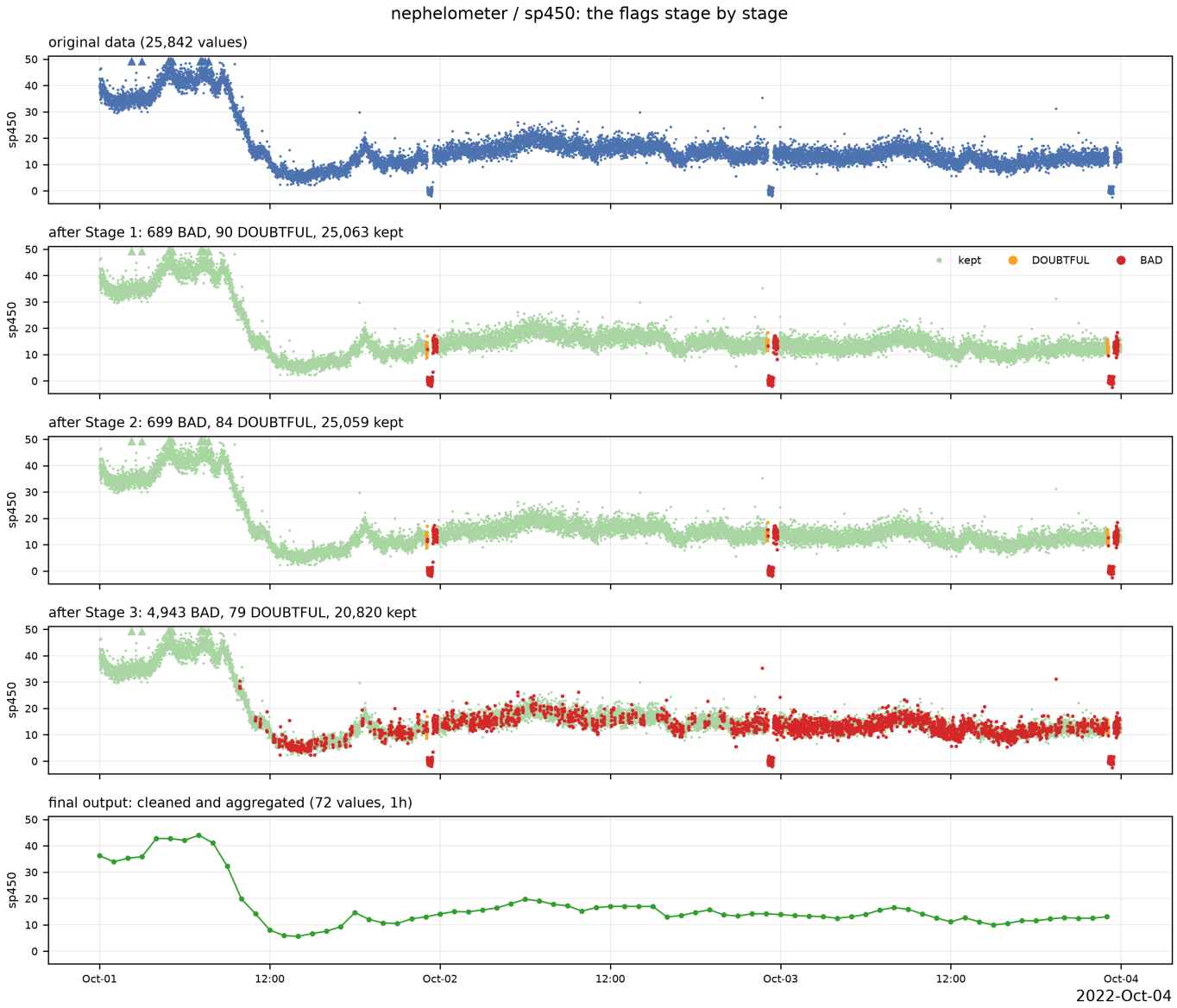
Files

- results (21 CSV): eBC370_final_1h.csv, eBC370_stage2_flags.csv, eBC370_stage3_1min.csv, eBC470_final_1h.csv, eBC470_stage2_flags.csv, eBC470_stage3_1min.csv, eBC525_final_1h.csv, eBC525_stage2_flags.csv, eBC525_stage3_1min.csv, eBC590_final_1h.csv, eBC590_stage2_flags.csv, eBC590_stage3_1min.csv, eBC660_final_1h.csv, eBC660_stage2_flags.csv, eBC660_stage3_1min.csv, eBC880_final_1h.csv, eBC880_stage2_flags.csv, eBC880_stage3_1min.csv, eBC950_final_1h.csv, eBC950_stage2_flags.csv, eBC950_stage3_1min.csv
- provenance.yml
- plots: AAE (4), control (9), eBC370 (8), eBC470 (6), eBC525 (7), eBC590 (6), eBC660 (6), eBC880 (7), eBC950 (6)

nephelometer

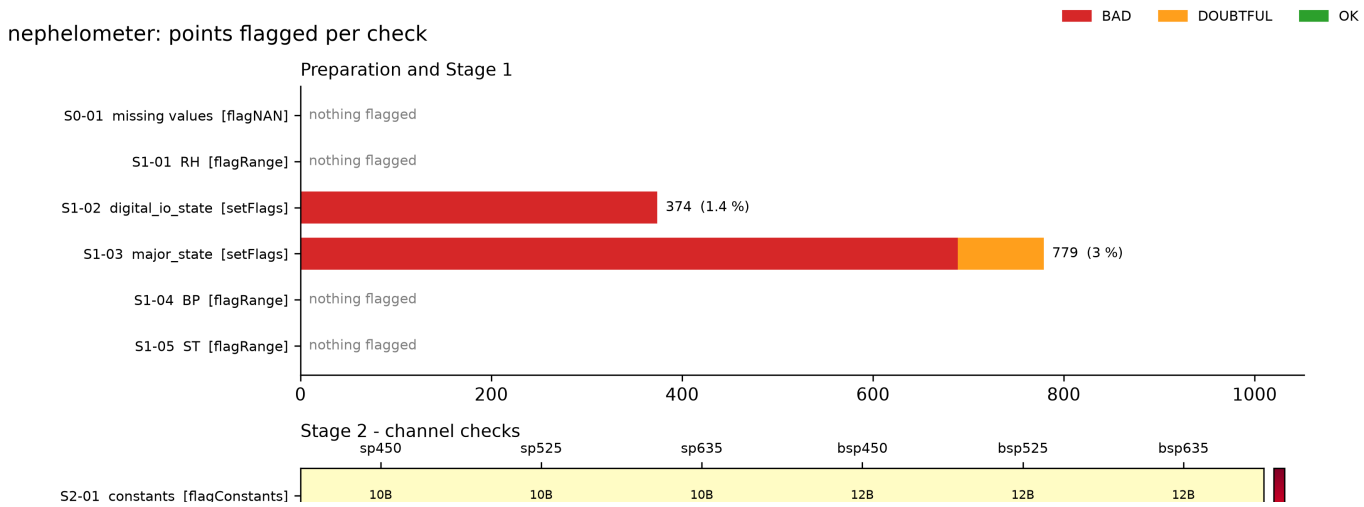
▸ configuration

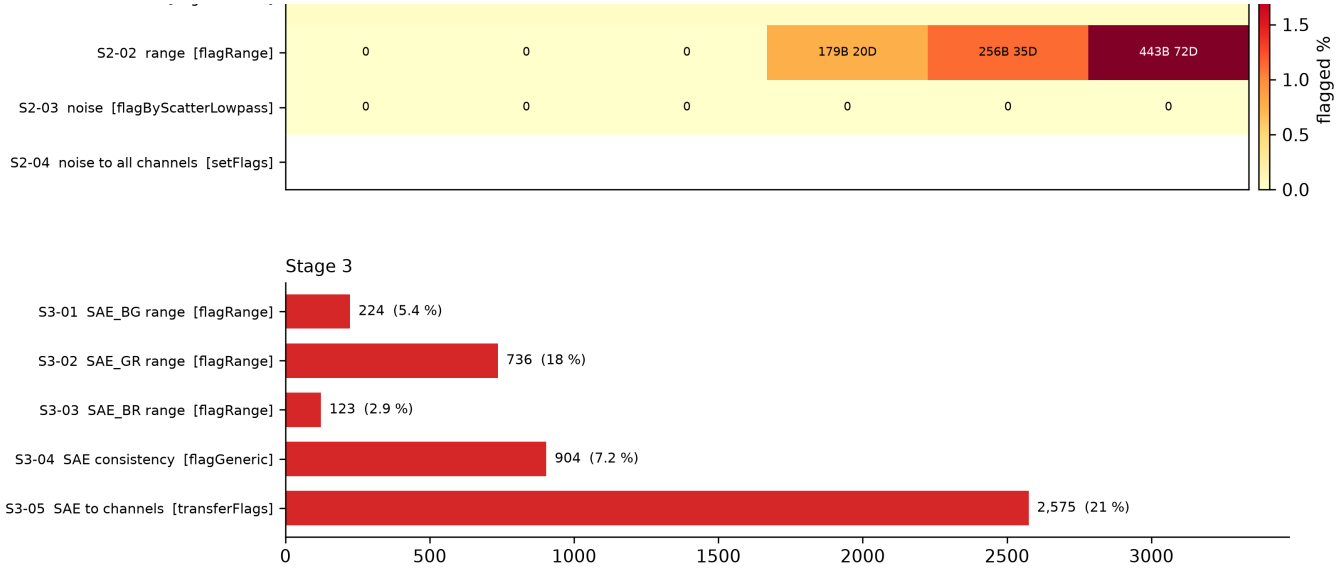
sp450 stage by stage



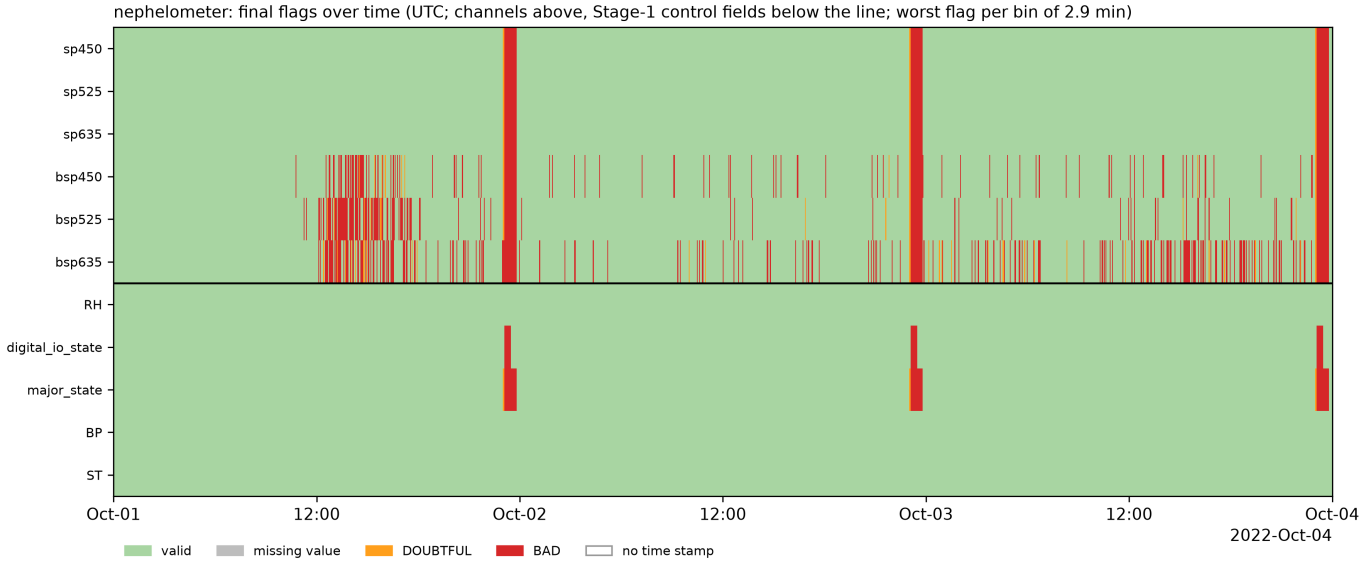
The flags stage by stage: original data, after Stage 1 (control data), Stage 2 (the channel's checks) and Stage 3 (derived quantities), each point coloured kept / DOUBTFUL / BAD; bottom: the final, cleaned and aggregated output. Values outside the 0.5-99.5 % range are triangles at the edge.

The other channels stage by stage (PNG files): sp525, sp635, bsp450, bsp525, bsp635

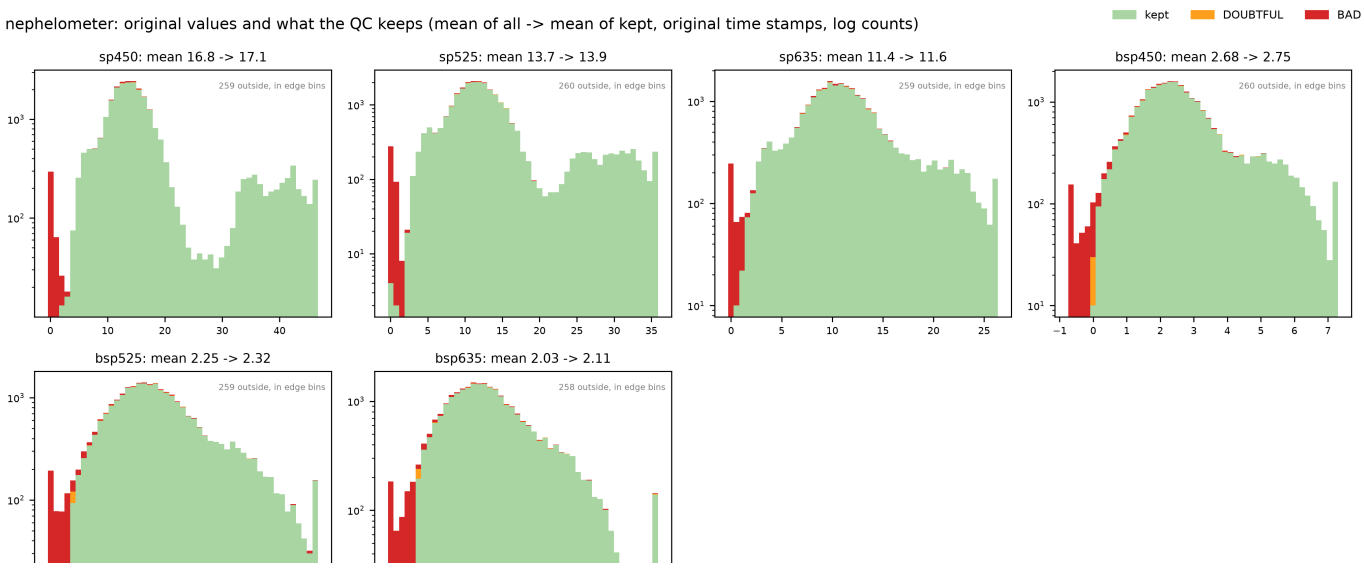


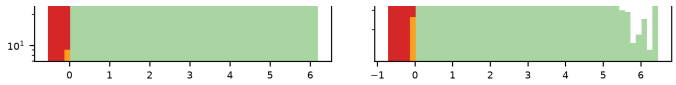


Points flagged by every check (Stage 2: B = BAD, D = DOUBTFUL, colour = % of the valid points of the checked field).



Final flags over time: worst flag per time bin; channels above the line, Stage-1 control fields below.





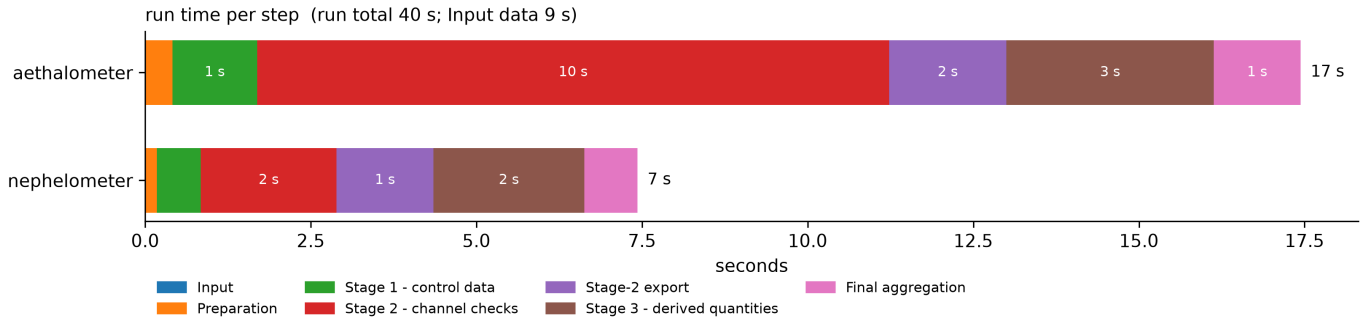
Distribution of the original values, split into kept / DOUBTFUL / BAD.

▸ all checks (table)

Files

- results (18 CSV): bsp450_final_1h.csv, bsp450_stage2_flags.csv, bsp450_stage3_1min.csv, bsp525_final_1h.csv, bsp525_stage2_flags.csv, bsp525_stage3_1min.csv, bsp635_final_1h.csv, bsp635_stage2_flags.csv, bsp635_stage3_1min.csv, sp450_final_1h.csv, sp450_stage2_flags.csv, sp450_stage3_1min.csv, sp525_final_1h.csv, sp525_stage2_flags.csv, sp525_stage3_1min.csv, sp635_final_1h.csv, sp635_stage2_flags.csv, sp635_stage3_1min.csv
- provenance.yml
- plots: SAE (4), bsp450 (6), bsp525 (6), bsp635 (6), control (5), sp450 (7), sp525 (7), sp635 (7)

Run time



Seconds per step and instrument.

▸ timing table

Warnings (1) and errors (0)

- aethalometer / Stage 1 - control data: inlet_rh: no data - check skipped

Made by saqc-atmo from the QC objects of the run; the tables are also in `stats/*.csv` and `run_summary.json`.