Deliverable D4 (D1.4)

NRT source apportionment service tools for submicron carbonaceous matter

RI-URBANS

Research Infrastructures Services Reinforcing Air Quality Monitoring Capacities in European Urban & Industrial AreaS (GA n. 101036245)

By

INERIS & co-workers

22nd December 2022

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

Deliverable D4 (D1.4): NRT source apportionment service tools for submicron carbonaceous matter

Authors: Olivier Favez (INERIS), Jean-Eudes Petit (CNRS/LSCE), Aurélien Chauvigné (CNRS/AERIS-Icare), Nicolas Pascal (CNRS/AERIS-Icare), Mohamed Gherras (INERIS), Anna Tobler (Datalystica), Francesco Canonaco (Datalystica)

Work package (WP)	WP4 Novel AQ metrics and advanced source apportionment STs for PM, and nanoparticles
Deliverable	D4 (D1.4)
Lead beneficiary	INERIS
Deliverable type	■ R (document, report) ☐ DEC (websites, patent filings, videos….) ☐ Other: ORDP (open research data pilot)
Dissemination level	■ PU (public) ☐ CO (confidential, only members of consortium and European Commission))
Estimated delivery deadline	M15 (31/12/2022)
Actual delivery deadline	22/12/2022
Version	Final
Reviewed by	WP1 Leaders
Accepted by	Project Coordination Team
Comments	This deliverable describes and provides both submicron organic aerosols and black carbon NRT source apportionment service tools (STs) which have been set-up as part of RI-URBANS WP1 Task 1.2 outputs. These STs are now ready to be tested online and updated in the frame of RI-URBANS WP4 Task 4.1 pilot activities.

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

Table of Contents

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

1. About this document

This Deliverable D4 (D1.4) summarizes the work done during the first 15 months of the RI- URBANS project to establish and implement near real time (NRT) source apportionment (SA) service tools (STs) for fine carbonaceous particles.

This Deliverable is produced in WP1, T1.2 on developing and implementing advanced SA-STs. This task aims at providing STs, based on best procedures and methodologies, to apportion novel health-related AQ metrics. We will evaluate and apply the most suited SA receptor models for operational applications, considering previous work in FAIRMODE, EMEP and COLOSSAL (COST Action: Chemical On-Line cOmpoSition and Source Apportionment of fine aerosol). This T1.2 provides pilot NRT-SA functionalities (harmonised with CAMS21a development and outcome) (D1.4-D1.5), with operational requirements of the SA software and data transfer/formatting STs for the novel NRT-SA of non-refractory aerosols (ACSM) and BC measurements data products, for modelling STs (WP3), pilot applications (WP4) and upscaling activities (WP5).

Carbonaceous particles, including black carbon (BC) and organic aerosols (OA), are representing a substantial part (typically, in the range 40-80%) of fine particulate matter (PM) in urban environment. At ACTRIS national facilities, their in-situ high-time resolution monitoring is usually conducted using aerosol chemical speciation monitors (ACSM) and multi-wavelength aethalometer (AE33), for OA and BC respectively. In the last decades, research activities allowed to develop novel methodologies to identify and quantify the main sources of carbonaceous aerosols measured using these two types of instruments. The aim of the present task within RI- URBANS is to implement such methodologies at a centralized server to demonstrate their ability to be operated - and thus to gain knowledge on these sources - in NRT. This demonstration will be conducted over the year 2023, in various European pilot cities, as part of WP4 Task1 activities.

Please note that ACSM are commonly measuring submicron (PM_1) aerosols while AE33 are generally installed with a $PM_{2.5}$ sampling head within air quality monitoring networks (AQMN). However, as BC particles are known to be overwhelmingly present in the smaller aerosol fraction (< 1µm), it is assumed that the STs established for this RI-URBANS activities are addressing submicron particulate matter.

This is a public document, available in the RI-URBANS website (https://riurbans.eu/work-package-1/#deliverables-wp1). The document will be distributed to all RI-URBANS partners for their use and submitted to European Commission as the RI-URBANS deliverable D4 (D1.4).

2. Data workflow defined for both service tools

As AE33 and ACSM instruments have their own data format and data treatment procedure specificities, different software solutions have been set-up to establish BC and OA SA-STs. Figure 1 presents the dataflow designed for each of them in the context of RI-URBANS.

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

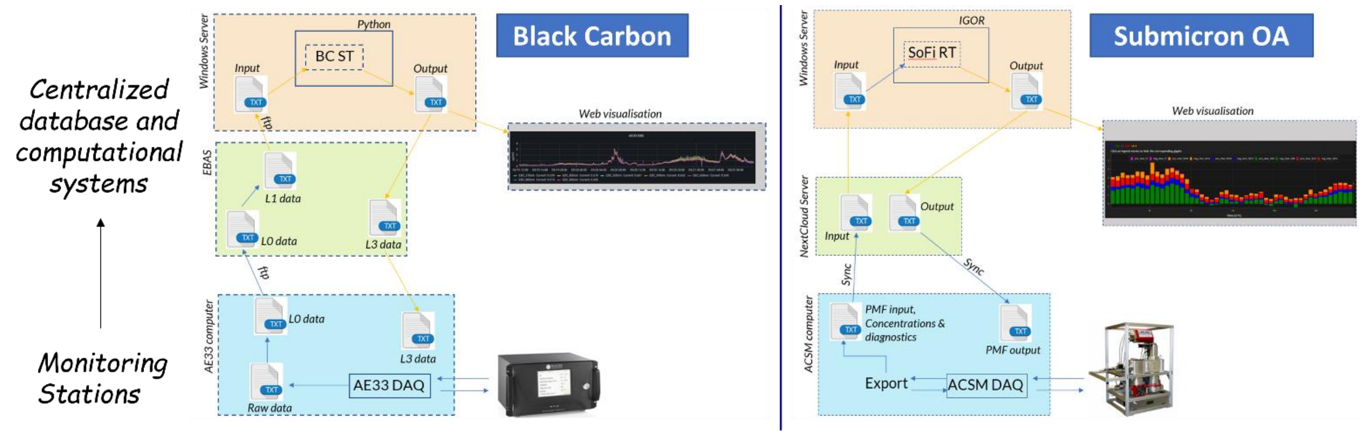
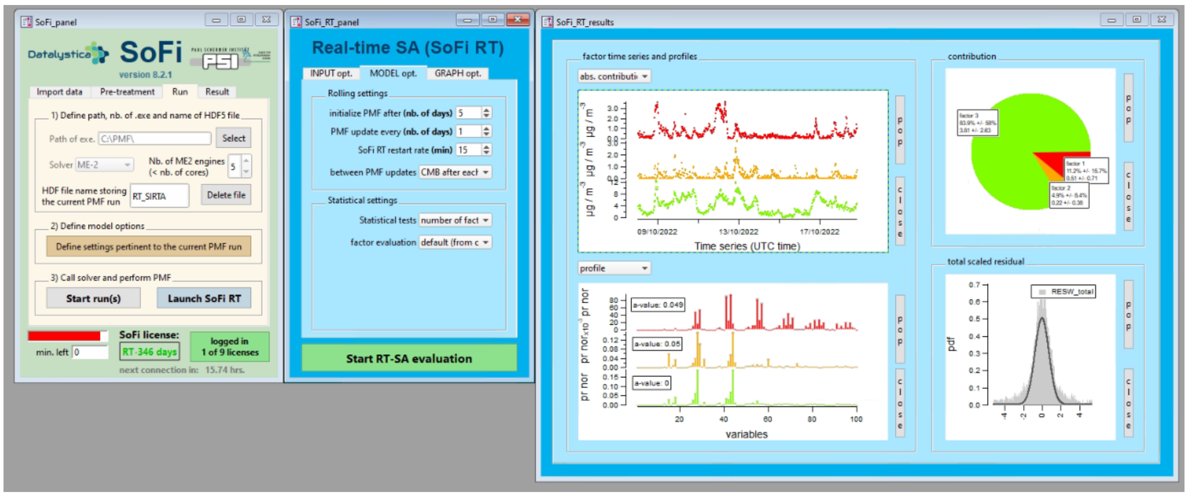


Figure 1. *Dataflow to be used in RI-URBANS for BC and OA NRT source apportionment.*

For BC, raw data are firstly stored as a text/ascii-format file on a computer installed at the station. These raw data files can then be transformed into ACTRIS Level0 datafile using a Python procedure developed in collaboration with ECAC-WCCAP (Topical centre unit coordinating in-situ measurements of aerosol microphysical properties within ACTRIS). These Level0 datafiles contain meta information about the measurements (e.g., instrument information, measurement conditions, AE33 parameters) and results for the most recent measurements (previous last hour), which can then be gathered, through EBAS, on a centralized server, located at AERIS-ICARE (CNRS) as part of ECAC-ACMCC (Topical centre unit coordinating in-situ online measurements of aerosol chemical speciation within ACTRIS). This server is hosting the BC SA software (BC ST) configured for the aim of RI-URBANS activities, which is described in section 4.

For OA, an Igor-based data export tool has been created to generate a data package that is subsequently sent to a NextCloud server maintained by AERIS-ICARE. This NextCloud can then be used for input data collection on the ACMCC/AERIS-ICARE centralized server hosting the OA SA software provided by Datalystica (SoFi-RT). Calculation principles and operation of this software is described in section 3.

Finally, the NRT-SA outputs are archived on the centralized server (for future submission to EBAS) and sent back to data provider. They can also be visualized through a dedicated interface, available in the internet page: https://dataviz.icare.univ-lille.fr/acsm_dataviz, as presented in Figure 2 for OA SA outputs.



1) Define path, nb. of .exe and name of HDF5 file
Path of exe. C:\PMF\
Solver ME-2
Nb. of ME2 engines (< nb. of cores) 5
HDF file name storing the current PMF run RT_SIRTA
2) Define model options
3) Call solver and perform PMF
min. left 0
SoFi license: RT-346 days
logged in 1 of 9 licenses
next connection in: 15.74 hrs.

Rolling settings
initialize PMF after (nb. of days) 5
PMF update every (nb. of days) 1
SoFi RT restart rate (min) 15
between PMF updates CMB after eact
Statistical settings
Statistical tests number of fact
factor evaluation default (from c

a-value: 0.049
a-value: 0.05
a-value: 0

Figure 2. *Snapshot of the SoFI-RT results from the Paris SIRTA site in September 2022.*

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

3. Description of the OA Source Apportionment Software (SoFi-RT)

During the project preparation, it has been decided that the OA SA software to be used within RI-URBANS is based on the state-of-the-art SoFi toolkit developed by Datalystica. The latter company is based in Switzerland and has been recently created as spin-off of the Paul Scherrer Institute (PSI, Villigen) to develop and commercialize computational solutions for SA of atmospheric pollutants.

For the aim of RI-URBANS, Datalystica is providing a NRT software (SoFi-RT) which is fully described in Chen et al. (2022a) and references therein. This software is commercially available from Datalystica, Igor-based and proprietary (i.e., non-open source).

The operating principle of SoFi-RT is presented in Figure 3. Briefly, NRT input data obtained from the ACSM measurement and subsequent data transfer, validation and formatting procedures are computed online using a Chemical Mass Balance (CMB) receptor model, allowing to apportion the total OA signal into various OA fractions (source factors). These OA factors are indicative of specific emission sources (such as traffic exhaust, wood burning, etc ..) or of secondary oxidation processes. To enable ACSM NRT-SA, the source profiles (i.e., mass spectra) of constrained primary sources such as HOA (Hydrocarbon-Like Organic Aerool) and BBOA (Biomass Burning Organic Aerosol) and exact number of factors to be searched, are being currently collected in the pilot cities. This is based on pre-existing information obtained through prior Positive Matrix Factorization (PMF) analyses conducted for each of the investigated site.

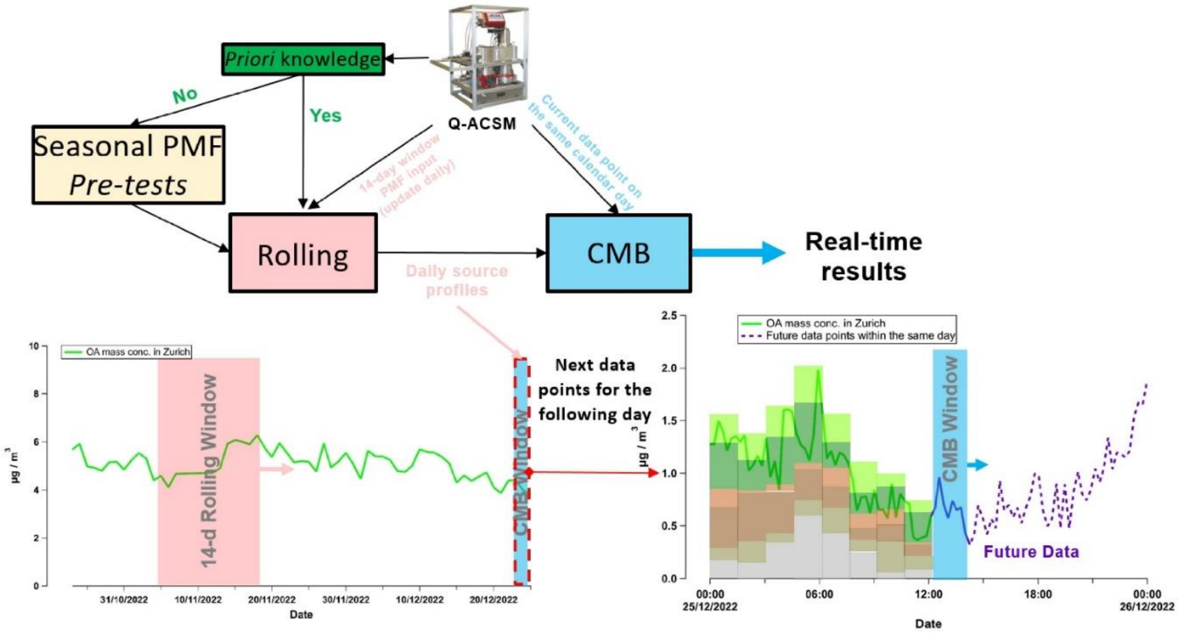


Figure 3. *SoFi-RT operating principles (taken from Chen et al., 2022a).*

For WP4 Task 1 piloting activities, most of the stations participated in a previous study by Chen et al. (2022b), where the sources were studied offline using similar SA methods. For the new stations, with no such existing SA outputs, the manual SA needs to be done first in order to establish the factors. Moreover, rolling PMF analyses are further achieved routinely by SoFi-RT (see Figure 4) to ensure the correct factor number and chemical profiles all along the online CMB-based computations in NRT.

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

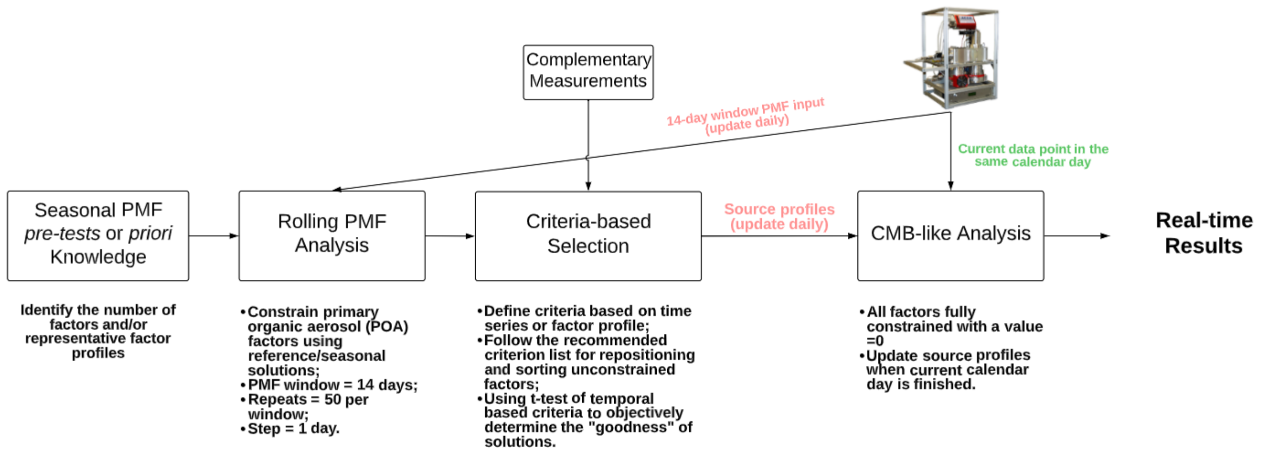

Complementary Measurements
14-day window PMF input (update daily)
Current data point in the same calendar day
Seasonal PMF pre-tests or priori Knowledge
Rolling PMF Analysis
Criteria-based Selection
Source profiles (update daily)
CMB-like Analysis
Real-time Results
Identify the number of factors and/or representative factor profiles
•Constrain primary organic aerosol (POA) factors using reference/seasonal solutions;
•PMF window = 14 days;
•Repeats = 50 per window;
•Step = 1 day.
•Define criteria based on time series or factor profile;
•Follow the recommended criterion list for repositioning and sorting unconstrained factors;
•Using t-test of temporal based criteria to objectively determine the "goodness" of solutions.
•All factors fully constrained with a value =0
•Update source profiles when current calendar day is finished.

Figure 4. *Step-by-step instruction flow chart within SoFi-RT.*

SoFi-RT can now be downloaded from Datalystica website https://datalystica.com/downloads/ (a password is needed).

It can be installed on Windows 7 or later or macOS, including the following steps:

- Initial configuration:
  a. Set a configuration file (RT_config_[STATION_ID].txt) following the structure presented Appendix A.
  b. Set a profile file (profile_[STATION_ID].ipf) following the structure presented Appendix B.
- Running process: A compiled procedure is detailed below based on the use of a configuration file for a specific station.
  a. Place the SoFi software under [hard drive]:\ME2_engine\ME2_General\SoFi
     Note that the ME2.exe file is supposed to reside under [hard drive]:\ME2_engine
  b. Open a new Igor experiment
  c. Load the main software: SoFi- (Version nb.).ipf

  Load the key: SoFi_key.ipf

  d. Compile the software (button at the end of the software SoFi sheet).

  The tab "SoFi" should appear in the menu bar on top in Igor.
  Select "Initialize SoFi".

  e. On the “Run” tab, select “Launch SoFi-RT”
  f. Select “CONFIG file” and enter the configuration file name.
  g. Select the corresponding configuration file.
  h. Select ”input files” and select the input folder.
  i. Select “Start RT-SA evaluation” and save the experiment. The process will be running until you select “Stop RT-SA evaluation”.
- Outputs: The code generates a text file for every input file.

SoFi-RT version 9.2.0.2 allows to set a configuration file as input (not available online and can be requested to the Datalystica team). A SoFi key will be needed. It should also be noted that Igor Pro 9 is recommended for the best use of SoFi-RT.

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

4. Description of the BC Source Apportionment Software (BC RT)

The BC RT software package is composed of four parts, two main scripts and two supporting scripts:

- Import level 0 NASA Ames files (Appendix C).
- Convert level 0 NASA Ames files to level 1 data (Appendix D).
- NRT_SA of averaged BC (15 minutes) based on user-defined α-values (Appendix E).
- Offline determination of site-specific Angström Absorption coefficients (so-called α-values), based on data recorded previously at that station (optional but highly recommended, Appendix F).

Level 0 NASA Ames files (.nas) should be directly generated by the different stations and should contain the measured data in the format as defined by EBAS (https://ebas-submit.nilu.no/templates/Filter-Absorption-Photometer/AE33_lev0) as well as the site-specific metadata. The metadata contains information about the site location (not needed for the SA) as well as some crucial parameters such as the filter type and the leakage factor.

The script is set up so that when it is started, it firstly checks whether there is data from previous files available. A history file is automatically generated/updated each time the output files are created and this history filet is created in the same folder where the level 0 data can be found.

Afterwards, the code continuously checks the indicated file folder (i.e., synchronized cloud folder such as Nextcloud) for newly added files. If a new file is added, the algorithm first waits until the file is fully created. This is important, as the code will crash otherwise when it wants to import data, which is not fully generated in the first place. Afterwards, it will check for file extensions. Only if new NASA Ames files (.nas) are newly added, the script will continue. The newly created data will be imported, treated, averaged, recalculated and flagged before SA will be performed and the results will be stored.

For the site-specific configuration, variables are split into separate classes. The class "*quality_control*", "*SA_values*" "*path_settings*" should be changed depending on the site, while the class "*instrument_settings*" contains general AE33 parameters.

The general operating principles of BC RT are provided in Figure 5 and each of the data treatment steps are detailed hereafter.

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

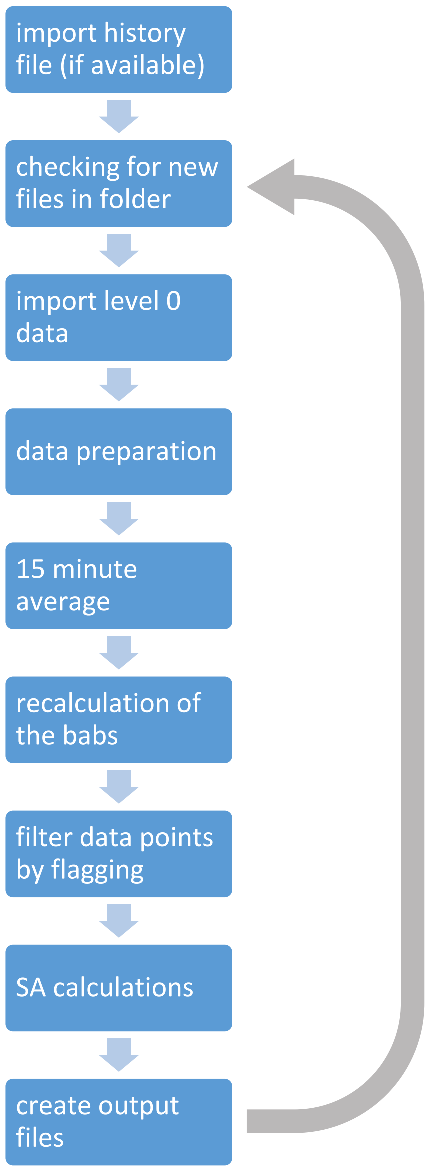

import history file (if available)
checking for new files in folder
import level 0 data
data preparation
15 minute average
recalculation of the babs
filter data points by flagging
SA calculations
create output files

Figure 5. Schematic overview on the different steps for NRT-SA of BC data.

Data quality control

The *quality control* class contains information on what should be regarded as valid data. Therefore, the user needs to define a detection limit (DL), an upper and a lower limit for the acceptable AAE values (upper AAE and lower_AAE), the limit for the correlation coefficient for the curve fit (r2), an upper and lower limit for the allowed ratio of BC before and after the filter change (upper ratio and lower ratio), the ACTRIS harmonization coefficient (hfactor) and the default filter tape type (default tape). As described in the for the **Lvl0_to_Lvl1.py** script, this default filter tape type is currently needed as some level 0 data is based on a previous template. For the future, this can probably be dropped and currently it should only be regarded as a back-up for those level 0 files based on a previous template. If the level 0 data is based on a newer template and the values are at the proper place in the metadata, the default filter tape type will be disregarded and the one in the file will be used.

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

SA values

The *SA_values* class contains the α-value that will be used for the SA. By default, they are set to 0.9 for liquid fuel (alpha_lf) and 1.68 for solid fuel (alpha_sf). Site-specific α-values are encouraged though. For this, the user is referred to the previous chapter "Determination of site-specific α-values" (Appendix F), where the python script **AAE_dist_past.py** is described in more details.

Path_settings

In the *path_settings*, the user needs to define the path, where the NASA Ames files are stored (folder_path). Furthermore, the path for where the result files should be stored (output_path) as well as the base name for the result files (file_name) needs to be defined.

Instrument_settings

There is one more class "*instrument_settings*", which contains some instrument parameters. These are universal AE33 parameters and there is no need for the user to change this as long as the instrument is operated under normal conditions. The instrument parameters include the wavelengths (wavelengths) and the corresponding MAC values as described in the AE33 manual:

Table 1. *Wavelenghts and corresponding mass absorption cross-section (MAC) values as described in the AE33 manual (https://gml.noaa.gov/AE33_UsersManual_Rev154).*

Channel	**Measurement wavelength (nm)**	**Mass absorption cross-section σ_{air} (m²/g)**
1	370	18.47
2	470	14.54
3	520	13.14
4	590	11.58
5	660	10.35
6	880	7.77
7	950	7.19

Furthermore, this class also includes the filter surface area ($S = 0.785e\text{-}4\ m^2$), as it is needed for the calculation of the 15-minute average.

Data preparation

After importing the data, it is converted into level 1 data using the **Lvl0_to_Lvl1.py** script. For further steps, certain parameters are required. The enhancement factor (C-value) and the leakage factor (Z-factor) are extracted from the metadata file. Depending on the filter tape type, the C-value is extracted. The code currently knows three different filter tape types: Magee AE33-FT, Magee M8050 or Magee M8060. Furthermore, the flow 1 (since all further calculations will be done on channel 1), the k parameter for all wavelengths as well as the attenuation (ATN) for all wavelengths on channel 1, are extracted from the level 0 data.

Averaging and further calculations

For increase the signal-to-noise ratio of BC RT outputs, it has been decided that the average should not be just "normal" averaging over the 15 minutes, but rather should replicate the algorithm that also the AE33 uses for the 1-minute BC average. Calculation of the average in this way might be less noisy, since it is calculated from the difference in attenuation signals obtained for the end vs. the beginning of the 15-minute period. The calculation of the BC average is based on the filter surface area (*S*), the difference in attenuation on spot 1 (ΔATN_{i_1}), the volumetric flow on spot 1 (*F_1*), the leakage factor (*ζ*), the wavelength-specific mass absorption cross-section (*MAC_i*), the enhancement factor (*C*), the loading effect parameter (*k*) and a time interval of 15 minutes (*Δt* = 15 min):

$$eBC_i = \frac{S * \frac{\Delta ATN_{i_1}}{100}}{F_1 * (1-\zeta) * MAC_i * C * \left(1 - k * ATN_{i\_1}\right) * \Delta t}$$

Since the ATN is extracted from level 0 data, it is already ATN_0 corrected. The ATN_0 correction is needed, as it can be assumed that the first measurement point after the tape advance has a ATN of zero (since it is a fresh filter spot point). However, the instrument typically measures a small ATN value and with the ATN_0 correction, this small ATN value is subtracted from all following points until the next tape advance.

If a filter tape advance is happening during the 15 min over which is averaged, the total concentration of BC will be negative. The tape advance is induced once a certain ATN is reached, typically an ATN of 120. Since the recalculation of the BC is done over the difference in ATN, which will end up negative in this case, the total BC concentration will be (strongly) negative as well. There is not really a way to avoid this. One possibility would be to instead of using the difference, to add the maximum ATN, at which the tape advances, to the ATN measurement after the tape advance. But this would induce unquantifiable uncertainties as the tape advances typically lasts for about roughly 4 minutes and during this time no measurements are performed. Therefore, it was decided in accordance with INERIS to tolerate a loss of one 15 minutes average point every few days during the tape advance. If the code should be adapted to averaging longer time periods, this point might need to be discussed again, also depending on how often such tape advances happen.

Note that the very first BC concentration will come back as NaN since the difference in ATN can only be calculated from the second point on.

After the recalculation of the BC concentrations, the absorption coefficients b_{abs} for each wavelength are recalculated. Furthermore, the AAE at 470 vs 950 nm is calculated for each data point. The wavelengths of 470 nm and 950 nm are chosen since the SA is performed also using these wavelengths. 470 nm is chosen of 370 nm because less interferences of brown carbon (BrC) can be expected at this higher wavelength which is still in the UV range.

Data filtering/flagging

In a next step the data is undergoing another data quality checks (detection limit, AAE outside range and correlation criterion) and data that fails this check will be flagged. If the data is already flagged with some this other that 000, these flags will not be overwritten, it will be assumed that the flags from earlier quality checks have higher priority. The python script is prepared to filter out the data properly and 3-digit flags are used. However, these flags (just the 3-digit code) might need to be adjusted to fit the EBAS requirements.

Detection limit

First of all, the data is filtered for data below the detection limit. For data below the detection limit (user-defined in the user-defined definitions), the data is flagged with “147”. Assuming that all data below 0 are due to problem of the recalculation of BC at the filter tape advance, this data is flagged with “456”.

AAE outside range

In a next step, the data is filtered/flagged for data points that have AAE outside the user-defined limits. The flag “147” is used. Typical limits are 0.7 and 3, but in the end, these can be defined by the user.

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

Correlation criterion

In a last step, the data is filtered for data points that do not match the criterion of having a high enough correlation coefficient over the seven wavelengths. It is assumed that absorption coefficient over the wavelengths should exhibit an exponential relationship with a high correlation coefficient. For this, the logarithmic and normalized (to the value at 950 nm) b_{abs} coefficient as well as the logarithmic and to 950 nm normalized wavelengths are calculated. Afterwards, a linear fit is performed over the calculated coefficients versus wavelengths. Data points that fall below the user-defined limit (typically $r^2 < 0.9$) are flagged with “147”.

Final source apportionment calculation

Finally, SA based on Sandradewi et al. (2008) and the user-defined α-values is performed. For the SA, the wavelengths of 470 nm and 950 nm are chosen. The calculation of the BC_{lf} and BC_{sf}, is based on the b_{abs} at the two chosen wavelengths, the wavelengths themselves and the respective α-values. Furthermore, it is assumed that only two combustion sources are present, so that the sum of BC_{lf} and BC_{sf} is the total BC concentration.

$$sf_{@950nm} = \frac{b_{abs,470nm} - \left(\frac{950}{470}\right)^{\alpha,lf} * b_{abs,950nm}}{\left(\frac{950}{470}\right)^{\alpha,sf} - \left(\frac{950}{470}\right)^{\alpha,lf}}$$

$$lf_{@950nm} = b_{abs,950nm} - sf_{@950nm}$$

$$fraction_{lf} = \frac{lf_{@950nm}}{b_{abs,950nm}}, only\ allowing\ values\ between\ 0\ and\ 1$$

$$eBC_{lf} = eBC_{total} * fraction_{lf}$$
$$eBC_{sf} = eBC_{total} - eBC_{lf}$$

In the following, we described the procedure to operate BC RT such as this will be done within RI-URBANS WP4 Task 1 piloting activities:

- Installation of:
  a. Required dependencias:

```
pip install -r requirements.txt
```

  b. EBAS python package:
     Install EBAS python packages available at https://git.nilu.no/ebas/ebas-io
- Initial configuration: Set a new configuration file([STATION_ID].cfg) on the cfg directory following the structure presented below. Add your own station ID (in green) and set every configuration value to fit with your instrument. It is recommended to use the EBAS_STATION_NAME as STATION_ID.

```
[ID]
STATION_ID = “TEST”

[QUALITY_CONTROL]
DL = 0
upper_AAE = 3
lower_AAE = 0.7
r2 = 0.9
upper_ratio = 1.5
lower_ratio = 0.5
hfactor = 2
default_tape = "Magee AE33-FT"
good_flags = 0
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
[SA_VALUES]
alpha_lf = 0.9
alpha_sf = 1.68

[ALGO_TYPE]
avg_type = 0

   [INSTRUMENT_SETTINGS]

MAC = [18.47, 14.54, 13.14, 11.58, 10.35, 7.77, 7.19]
wavelengths = [370, 470, 520, 590, 660, 880, 950]
S = 0.785e-4
sigma = 7.77
```

- Running process

The AE33-SA code can be run using two modes:

a. Watchdog mode: new process will be running as soon as new level 0.nas data will be created on the INPUT/[STATION_NAME] folder.

Run the command line:

```
python3 NR-BC.py [configuration_file_name.cfg]
```

b. One file mode: Only one process will be running

Run the command line:

```
python3 NR-BC.py [configuration_file_name.cfg] [full_input_path.nas]
```

- Outputs: The code generates two files:
  a. A text file AE33-SA_[STATION-ID]_lastData.txt containing last processed input data at full time resolution.
  b. A text file AE33-SA_[STATION-ID]_[start_time].txt containing outputs at 15 minutes resolution.

Since there is no pre-defined EBAS format for BC SA results, the output is kept very simple. After each new NASA Ames file that successfully undergoes all calculations, a new text file is generated. The file will be created in the user-defined output path. The new file is named based on the file name that was defined as well as the date and time the file is created. Currently, the file contains the time stamp, the total BC concentration, the BC_{lf} and BC_{sf} calculated concentration, and the flags.

5. Conclusion and perspectives

The present report describes both submicron (PM_1) organic aerosols and black carbon (BC) near real time (NRT) source apportionment (SA) service tools (STs) which have been set-up as part of RI-URBANS WP1 Task 1.2 outputs.

For submicron OA SA (based on ACSM measurements), the ST has been designed on the use of SoFi-RT which is an advanced commercially available and closed-source software. Upscaling activities within RI-URBANS WP5 should define in which conditions such a solution may be envisaged in the long-term for ACTRIS Level3 data provision.

For BC, an open-source software has been set-up in Python, in accordance with the FAIR principles. It should be noted that related codes which are provided in the present report will probably be updated in the frame of upcoming RI-URBANS activities. Some of the data qualification procedures introduced here may also be used to optimize the production of ACTRIS absorption coefficient Level2 data.

Both SA-STs are now ready to be tested online and updated in the frame of RI-URBANS WP4 Task 4.1 pilot activities (see Milestone M18 (M4.2)).

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

6. References

Chen, G., Canonaco, F., Slowik, J. G., Daellenbach, K. R., Tobler, A., Petit, J.-E., Favez, O., Stavroulas, I., Mihalopoulos, N., Gerasopoulos, E., El Haddad, I., Baltensperger, U., and Prévôt, A. S. H.: Real-Time Source Apportionment of Organic Aerosols in Three European Cities, Environ. Sci. Technol., 56, 15290–15297, https://doi.org/10.1021/acs.est.2c02509, 2022a.

Chen, G., Canonaco, F., Tobler, A., Aas, W., Alastuey, A., Allan, J., Atabakhsh, S., Aurela, M., Baltensperger, U., Bougiatioti, A., De Brito, J. F., Ceburnis, D., Chazeau, B., Chebaicheb, H., Daellenbach, K. R., Ehn, M., El Haddad, I., Eleftheriadis, K., Favez, O., Flentje, H., Font, A., Fossum, K., Freney, E., Gini, M., Green, D. C., Heikkinen, L., Herrmann, H., Kalogridis, A.-C., Keernik, H., Lhotka, R., Lin, C., Lunder, C., Maasikmets, M., Manousakas, M. I., Marchand, N., Marin, C., Marmureanu, L., Mihalopoulos, N., Močnik, G., Nęcki, J., O'Dowd, C., Ovadnevaite, J., Peter, T., Petit, J.-E., Pikridas, M., Matthew Platt, S., Pokorná, P., Poulain, L., Priestman, M., Riffault, V., Rinaldi, M., Różański, K., Schwarz, J., Sciare, J., Simon, L., Skiba, A., Slowik, J. G., Sosedova, Y., Stavroulas, I., Styszko, K., Teinemaa, E., Timonen, H., Tremper, A., Vasilescu, J., Via, M., Vodička, P., Wiedensohler, A., Zografou, O., Cruz Minguillón, M., and Prévôt, A. S. H.: European aerosol phenomenology − 8: Harmonised source apportionment of organic aerosol using 22 Year-long ACSM/AMS datasets, Environment International, 166, 107325, https://doi.org/10.1016/j.envint.2022.107325, 2022b.

Sandradewi, J., Prévôt, A. S. H., Szidat, S., Perron, N., Alfarra, M. R., Lanz, V. A., Weingartner, E., and Baltensperger, U.: Using Aerosol Light Absorption Measurements for the Quantitative Determination of Wood Burning and Traffic Emission Contributions to Particulate Matter, Environ. Sci. Technol., 42, 3316–3323, https://doi.org/10.1021/es702253m, 2008.

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

7. Annexes

7.1. Appendix A: SoFi-RT configuration file structure

```
PATHS
path of executable: C:\ME2_engine
path of input files: C:\Users\icare-exploit\Nextcloud\SoFi
path for scan-wise results (CMB): C:\Users\icare-exploit\Nextcloud\SoFi\CMB_results
path of factor profiles from file: C:\Users\icare-exploit\Nextcloud\SoFi\profiles.itx

NAMES
name of HDF result files (large files): SIRTA_ACSM_140113_CMB_hdf
name of txt result file (single scans): SIRTA_ACSM_140113_CMB
name of prefix for input filename: SIRTA_ACSM-140113_RTdata

DATA TYPE
Data type: AMS UMR / Q-ACSM (without m/z 19 and 20)
AMS specific: exclude CO2 related variables
instrument type: ACSM

MODEL OPTIONS
restart RT: 0.1
factor start: 3
factor end: 3
factor names: HOA,BBOA,OOA
nb. of iterations: 2
resampling: enabled
rolling: enabled
window length: 5
window shift: 1
between PMF: CMB after each scan
stats tests (corr): disabled
stats tests (comp.): disabled
a value constraints: enabled
name of constraints over pr: HOA,,0.1;BBOA,,0.3
name of constraints over ts:
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

7.2. Appendix B: OA-profile file structure used within SoFi-RT

```
IGOR
WAVES/D amus    BBOA    HOA
BEGIN
       12      0.0195816      0.00435548
       13      0.00177887     0.00123129
       15      0.0578156      0.000323669
       16      0.00661979     0.000159817
       17      0.0413707      0.000993411
       18      0.152803       0.0176608
       24      0.000954616    0.000598064999999999
       25      0.00417111     0.00168077
       26      0.0188817      0.0101653
       27      0.0394386      0.0467035
       29      0.105552       0.0575171
       30      0.00235511     0.00115438
       31      0.0345104      0.00225392
       37      0.00344756     0.00281497
       38      0.0063574      0.00482307
       41      0.00357835     0.0871881
       42      0.0260754      0.0231025
       43      0.0589952      0.0957982
       44      0.152803       0.0176608
       45      0.0212448      0.00212237
       48      0.00127729     0.000586638
       49      0.00114331     0.000470311
       50      0.00525251     0.00359498
       51      0.00665134     0.00579356
       52      0.00547677     0.00290547
       53      0.0105094      0.00971986
       54      0.00449332     0.0107367
       55      0.0113834      0.0786988
       56      0.00699615     0.025329
       57      0.0105343      0.0732141
       58      0.00442504     0.00791393
       59      0.00437367     0.00103431
       60      0.0364833      0.000490688
       61      0.00580087     0.000704403
       62      0.00249367     0.00128073
       63      0.00440234     0.00285842
       64      0.00399333     0.00485254
       65      0.00530497     0.0108396
       66      0.00376887     0.00285353
       67      0.000269255    0.0314539
       68      0.00532577     0.0119253
       69      0.00718298     0.046239
       70      0.00367162     0.0142537
       71      0.00417081     0.0339731
       72      0.00279796     0.0027035
       73      0.0186776      0.00141897
       74      0.00491129     0.00204456
       75      0.00292925     0.00155402
       76      0.00322449     0.00176308
       77      0.00697773     0.0102862
       78      0.00311967     0.00505601
       79      0.00335329     0.0171883
       80      0.0023346      0.00271679
       81      0.00204644     0.027883
       82      0.00318945     0.00954215999999999
       83      0.000123758    0.0292814
       84      0.000334244    0.0114569
       85      0.00271962     0.0210689
       86      0.00237462     0.00249624
       87      0.00406878     0.00142766
       88      0.00192141     0.000482247
       89      0.00438034     0.00150617
       90      0.00184626     0.000613516999999999
       91      0.00565266     0.0162853
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
       92     0.00219963     0.00406125
       93     0.00373682     0.0105739
       94     0.00311281     0.00362235
       95     0.00541655     0.0221047
       96     0.00395541     0.00504482
       97     0.00341162     0.0205734
       98     0.00122703     0.00670492
       99     0.00383785     0.00399512
       100    0.00198064     0.000870173
END
X SetScale/P x 0,1,"", amus; SetScale y 0,0,"", amus
X SetScale/P x 0,1,"", BBOA; SetScale y 0,0,"", BBOA
X SetScale/P x 0,1,"", HOA; SetScale y 0,0,"", HOA

WAVES/T amus_txt
BEGIN
       "12"
       "13"
       "15"
       "16"
       "17"
       "18"
       "24"
       "25"
       "26"
       "27"
       "29"
       "30"
       "31"
       "37"
       "38"
       "41"
       "42"
       "43"
       "44"
       "45"
       "48"
       "49"
       "50"
       "51"
       "52"
       "53"
       "54"
       "55"
       "56"
       "57"
       "58"
       "59"
       "60"
       "61"
       "62"
       "63"
       "64"
       "65"
       "66"
       "67"
       "68"
       "69"
       "70"
       "71"
       "72"
       "73"
       "74"
       "75"
       "76"
       "77"
       "78"
       "79"
       "80"
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
	"81"
	"82"
	"83"
	"84"
	"85"
	"86"
	"87"
	"88"
	"89"
	"90"
	"91"
	"92"
	"93"
	"94"
	"95"
	"96"
	"97"
	"98"
	"99"
	"100"
END
X SetScale/P x 0,1,"", amus_txt; SetScale y 0,0,"", amus_txt
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

7.3. Appendix C: Procedure to import Level0 NASA Ames data files (for BC RT usage)

```
from ebas.io.file.nasa_ames import EbasNasaAmes
import pandas as pd


def nasa_ames_read (path):
     nas=EbasNasaAmes()
     nas.read(path)
     return (nas)



def ebasobj_ae33_lv0 (nas):
    """


    Parameters
    ----------
    nas : nasa_ames object
        DESCRIPTION.

    Returns
    ebasobject AE33 Data Lv0
    buf : TYPE
        DESCRIPTION.

    """
    buf= pd.DataFrame()
    for i in range (78):
        vbuf=nas.variables[i].values_
        fbuf=nas.variables[i].flags
        buf[nas.vname(i)[0]]=vbuf
        buf['numflag']=fbuf
    return buf
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

7.4. Appendix D: Procedure to convert Level0 NASA Ames into Level1 data files (for BC RT usage)

```
# -*- coding: utf-8 -*-

from math import ceil
import re
from ebas.io.file.nasa_ames import EbasNasaAmes
from nilutility.datatypes import DataObject
from nilutility.datetime_helper import DatetimeInterval
import pandas as pd
import sys
import warnings
import numpy as np


def ae33_proclvl1 (ebasobj_ae33_lvl0, nas, hfactor, default_tape):
    """
    Parameters
    ----------
    ebasobj_ae33_lvl0 : ebasobject lv0 AE33 Data
        DESCRIPTION.
    hfactor :
        DESCRIPTION. The default is 2.
    default_tape : string
        DESCRIPTION. user defined default tape, in case no tape is mentioned in lvl0
data

    Returns
    ebasobject lvl1 AE33 Data
    df_buf1 : TYPE
        DESCRIPTION.

    """

    #check if ebasobj is ae33 and lvl 0
    buf = nas.metadata['instr_type']
    if (buf != 'filter_absorption_photometer'):
        sys.exit("wrong data format: entry of 'Instrument type' is not
'filter_absorption_photometer'")

    buf = nas.metadata ['datalevel']
    if (buf != '0'):
        sys.exit("wrong data format: entry of 'Data level' is not '0'")
    ebasobj_ae33_lvl1 = ebasobj_ae33_lvl0


    # data set creation
    df_ae33 = ebasobj_ae33_lvl1.copy()
    df_buf = pd.DataFrame({'start_time':Time(nas)['starttime'],
                           'end_time': Time(nas)['endtime'],
                           'temp': df_ae33.iloc[:,2],
                           'pres': df_ae33.iloc[:,0],
                           'status': df_ae33.iloc[:,9],
                           'ebc370': df_ae33.iloc[:,21],
                           'ebc470': df_ae33.iloc[:,30],
                           'ebc525': df_ae33.iloc[:,39],
                           'ebc590': df_ae33.iloc[:,48],
                           'ebc660': df_ae33.iloc[:,57],
                           'ebc880': df_ae33.iloc[:,66],
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
                           'ebc950': df_ae33.iloc[:,75],
                           'numf':df_ae33['numflag']})
    #converting ebc values from decimal to float
    df_buf['ebc370']=df_buf['ebc370'].astype(float)
    df_buf['ebc470']=df_buf['ebc470'].astype(float)
    df_buf['ebc525']=df_buf['ebc525'].astype(float)
    df_buf['ebc590']=df_buf['ebc590'].astype(float)
    df_buf['ebc660']=df_buf['ebc660'].astype(float)
    df_buf['ebc880']=df_buf['ebc880'].astype(float)
    df_buf['ebc950']=df_buf['ebc950'].astype(float)


    #---
    #stp cond
    #---
    stpcond=extrstpcond(ebasobj_ae33_lvl1, nas)
    df_buf=ae33_stpcprr(df_buf, stpcond['normalization'],
stpcond['temp_norm'],stpcond['pres_norm'])


 #---
    #calc absorption
    #---
    mac = ae33_extract_mac(nas)

    cfactor_buf = nas.metadata['multi_scattering_corr_fact']
    if (cfactor_buf == None):
        if (default_tape=="Magee AE33-FT"):
            cfactor_buf = 1.57
        elif (default_tape=="Magee M8050"):
            cfactor_buf = 1.57
        elif (default_tape== "Magee M8060"):
            cfactor_buf = 1.39


    filtertype= nas.metadata['filter_type']
    if (filtertype=="Magee AE33-FT"):
        cfactor = 1.57
    elif (filtertype=="Magee M8050"):
        cfactor = 1.57
    elif (filtertype== "Magee M8060"):
        cfactor = 1.39
    else:
        if (default_tape=="Magee AE33-FT"):
            cfactor = 1.57
        elif (default_tape=="Magee M8050"):
            cfactor = 1.57
        elif (default_tape== "Magee M8060"):
            cfactor = 1.39
#        sys.exit("wrong data format: no valid entry found for 'Filter type")


    if (cfactor != cfactor_buf):
        print ("Wrong data format: 'Multi-scattering correction factor' and 'Filter
type' are inconsistent")
        print ("NO DATA LOADED")
        sys.exit("Wrong data format: 'Multi-scattering correction factor' and 'Filter
type' are inconsistent")
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
    df_buf['abs370']=df_buf['ebc370']* mac[0]/(cfactor*hfactor)
    df_buf['abs470']=df_buf['ebc470']* mac[1]/(cfactor*hfactor)
    df_buf['abs525']=df_buf['ebc525']* mac[2]/(cfactor*hfactor)
    df_buf['abs590']=df_buf['ebc590']* mac[3]/(cfactor*hfactor)
    df_buf['abs660']=df_buf['ebc660']* mac[4]/(cfactor*hfactor)
    df_buf['abs880']=df_buf['ebc880']* mac[5]/(cfactor*hfactor)
    df_buf['abs950']=df_buf['ebc950']* mac[6]/(cfactor*hfactor)



    #---
    #flagging
    #---

    for i in range (len(df_buf)):
        if (pd.isna(df_buf['abs370'][i])
            and pd.isna(df_buf['abs470'][i])
            and pd.isna(df_buf['abs525'][i])
            and pd.isna(df_buf['abs590'][i])
            and pd.isna(df_buf['abs660'][i])
            and pd.isna(df_buf['abs880'][i])
            and pd.isna(df_buf['abs950'][i])):
            df_buf['numf'][i]=addtoflag(df_buf['numf'][i], "999")
        # operation
        if (pd.isna(df_buf['status'][i])):

            if (np.bitwise_and(df_buf['status'][i],3) != 0):
                df_buf['numf'][i] = addtoflag(df_buf['numf'][i],"456")
            #flow

            if (np.bitwise_and(df_buf['status'][i] >> 2,3) != 0 ):
                df_buf['numf'][i] = addtoflag(df_buf['numf'][i],"456")
            #led
            if (np.bitwise_and(df_buf['status'][i] >> 4,3) != 0 ):
                df_buf['numf'][i] = addtoflag(df_buf['numf'][i],"456")
            #chamber
            if (np.bitwise_and(df_buf['status'][i] >> 6,1) != 0 ):
                df_buf['numf'][i] = addtoflag(df_buf['numf'][i],"456")
            #filter
            if (np.bitwise_and(df_buf['status'][i] >> 7,3) == 3 ):#0=ok, 1=30 spots
                                                                    #left, 2=5 spots
left, 3=error/end
                df_buf['numf'][i] = addtoflag(df_buf['numf'][i],"456")
            #procedures
            if (np.bitwise_and(df_buf['status'][i] >> 10,6) != 0 ):
                df_buf['numf'][i] = addtoflag(df_buf['numf'][i],"456")
            #connection
            if (np.bitwise_and(df_buf['status'][i] >> 13,1) != 0 ):
                df_buf['numf'][i] = addtoflag(df_buf['numf'][i],"456")
            #cleanair
            if (np.bitwise_and(df_buf['status'][i] >> 14,1) != 0 ):
                df_buf['numf'][i] = addtoflag(df_buf['numf'][i],"456")
            #cdcard
            if (np.bitwise_and(df_buf['status'][i] >> 15,1) != 0 ):
                df_buf['numf'][i] = addtoflag(df_buf['numf'][i],"456")
        #---
        #gen nasa ames obj
        #---
    df_buf1=pd.DataFrame({"start_time" : df_buf['start_time'],
                          "end_time": df_buf['end_time'],
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
                                 "temp": df_buf['temp'],
                                 "pres": df_buf['pres'],
                                 "abs370": df_buf['abs370'],
                                 "abs470": df_buf['abs470'],
                                 "abs525": df_buf['abs525'],
                                 "abs590": df_buf['abs590'],
                                 "abs660": df_buf['abs660'],
                                 "abs880": df_buf['abs880'],
                                 "abs950": df_buf['abs950'],
                                 "numf": df_buf['numf'] })

    Ae33_nas=EbasNasaAmes()

    Ae33_nas.metadata=nas.metadata.copy()

    meta=Ae33_nas.metadata
    time=DatetimeInterval(meta.reference_date,meta.creation_time)
    time_index=nas.find_sample_time_index(time)
    Ae33_nas.sample_times=[]
    for tim in range(time_index[0], time_index[1]+1):

        Ae33_nas.sample_times.append(DatetimeInterval(Time(nas)['startime'][tim],
Time(nas)['endtime'][tim]))


    #setting up the variables


    setup_variables(Ae33_nas,df_buf1['temp'] ,"temperature, K, Location=instrument
internal, Matrix=instrument" ,
                    "temperature","K","T_int", nas.variables[0].flags)

    setup_variables(Ae33_nas,df_buf1['pres'],"pressure, hPa, Location=instrument
internal, Matrix=instrument",
                    "pressure","hPa","p_int",nas.variables[2].flags)
    setup_variables(Ae33_nas,df_buf1['abs370'],"aerosol_absorption_coefficient, 1/Mm,
Wavelength=370.0 nm",

"aerosol_absorption_coefficient","1/Mm","abs370",nas.variables[20].flags)
    setup_variables(Ae33_nas,df_buf1['abs470'],"aerosol_absorption_coefficient, 1/Mm,
Wavelength=470.0 nm",

"aerosol_absorption_coefficient","1/Mm","abs470",nas.variables[29].flags)
    setup_variables(Ae33_nas,df_buf1['abs525'],"aerosol_absorption_coefficient, 1/Mm,
Wavelength=525.0 nm",

"aerosol_absorption_coefficient","1/Mm","abs525",nas.variables[38].flags)
    setup_variables(Ae33_nas,df_buf1['abs590'],"aerosol_absorption_coefficient, 1/Mm,
Wavelength=590.0 nm",

"aerosol_absorption_coefficient","1/Mm","abs590",nas.variables[47].flags)
    setup_variables(Ae33_nas,df_buf1['abs660'],"aerosol_absorption_coefficient, 1/Mm,
Wavelength=660.0 nm",

"aerosol_absorption_coefficient","1/Mm","abs660",nas.variables[56].flags)
    setup_variables(Ae33_nas,df_buf1['abs880'],"aerosol_absorption_coefficient, 1/Mm,
Wavelength=880.0 nm",

"aerosol_absorption_coefficient","1/Mm","abs880",nas.variables[65].flags)
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
    setup_variables(Ae33_nas,df_buf1['abs950'],"aerosol_absorption_coefficient, 1/Mm, 
Wavelength=950.0 nm",

"aerosol_absorption_coefficient","1/Mm","abs950",nas.variables[74].flags)
    #writting the metadata

    Ae33_nas.metadata.comp_name= "aerosol_absorption_coefficient"
    Ae33_nas.metadata.unit="1/Mm"
    Ae33_nas.metadata.vol_std_temp="273.15 K"
    Ae33_nas.metadata.vol_std_pressure="1013.25 hPa"
    Ae33_nas.metadata.type="TI"
    Ae33_nas.metadata.datalevel="1"

    #Cleaning the metadata

    # del Ae33_nas.metadata["filter_type"]
    # del Ae33_nas.metadata["multi_scattering_corr_fact"]
    # del Ae33_nas.metadata["max_attenuation"]
    # del Ae33_nas.metadata["leakage_factor_zeta"]
    # del Ae33_nas.metadata["comp_param_kmax"]
    # del Ae33_nas.metadata["comp_param_kmin"]
    # del Ae33_nas.metadata["comp_thresh_atten1"]
    # del Ae33_nas.metadata["comp_thresh_atten2"]


    return (Ae33_nas, df_buf)


def Time (nas):
    meta=nas.metadata
    time=DatetimeInterval(meta.reference_date,meta.creation_time)
    time_index=nas.find_sample_time_index(time)
    endtime=[]
    startime=[]
    if time_index:
        for tim in range(time_index[0], time_index[1]+1):
            startime.append(nas.sample_times[tim][0])
            endtime.append(nas.sample_times[tim][1])
    start_end_time={'startime':startime
                   ,'endtime':endtime}
    #print(type(start_end_time["startime"][0]))
    return (start_end_time)




def extrstpcond (ebasobj_ae33_lv0, nas):
    # stp correction if needed

    T_buf= 273.15
    p_buf= 1013.25
    T_norm= True
    p_norm = True

    norm = True

    md_temp = nas.metadata['vol_std_temp']
    md_pres= nas.metadata['vol_std_pressure']

    if (pd.isna(md_temp)):
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
        sys.exit("wrong data format: no accaptable entry for 'Volume1 std. 
temperature' in metadata")
    elif (type(md_temp)==str):
        if (md_temp.find(" ")>0):
            T_buf,unit=md_temp.split(' ')
            T_buf = float (T_buf)
        else :
            T_buf,unit = md_temp.split('K')
    elif (md_temp == "instrument internal"):
        T_norm= False
    elif (md_temp==273.15):
        T_norm=True
    else:
        sys.exit("wrong data format: no accaptable entry for 'Volume2 std. 
temperature' in metadata")
    if (pd.isna(md_pres)):
        sys.exit("wrong data format: no accaptable entry for 'Volume3 std. pressure' 
in metadata")
    elif (type(md_pres)==str):
        if (md_pres.find(" ")>0):
            p_buf,unit=md_pres.split(' ')
            p_buf = float (p_buf)
        else :
            p_buf,unit = md_pres.split('hPa')
    elif (md_pres == "instrument internal"):
        p_norm= False
    elif (md_pres==1013.25):
        p_norm=True
    else :
        sys.exit("wrong data format: no accaptable entry for 'Volume4 std. pressure' 
in metadata")
    if (T_norm==False and p_norm == False):
        norm = False
    elif (T_norm != p_norm) :
        sys.exit("wrong data format: mixed normalization condition are not allowed")
    l = {'normalization': norm, 'temp_norm': T_buf,'pres_norm':p_buf}

    return (l)


def ae33_stpcprr (ae33, normalized= True, temp = 273.15, pres=1013.25):
    ae33_new=ae33

    if (normalized and temp == 273.15 and pres == 1013.15):
        return (ae33_new)
    elif (normalized):
        factor = temp/273.15 * 1013.25/pres
    else :
        factor = ae33["temp"]/273.15 *1013.25/ae33["pres"]
    ae33_new['ebc370']= ae33_new['ebc370']*factor
    ae33_new['ebc470']= ae33_new['ebc470']*factor
    ae33_new['ebc525']= ae33_new['ebc525']*factor
    ae33_new['ebc590']= ae33_new['ebc590']*factor
    ae33_new['ebc660']= ae33_new['ebc660']*factor
    ae33_new['ebc880']= ae33_new['ebc880']*factor
    ae33_new['ebc950']= ae33_new['ebc950']*factor
    return (ae33_new)
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
def ae33_extract_mac (nas):
    Mac=[]
    for i in range (77):
        if ("Measurement uncertainty=20.0 %" in nas.vname(i)[0]):
            rest,mac= nas.vname(i)[0].split ("Mass absorption cross section=")
            mac=mac.split(' ')
            Mac.append(float(mac[0]))
    return (Mac)


def addtoflag (flag,new):
    flag_new= "9.999999999999999999999999999999999999999999999"
    pattern = re.compile(r'^[0-9]{1}[.][0-9]{45}$')
    if not(re.search(pattern,flag)):
        warnings.warn("error data format: wrong flag format")
        return (flag_new)
    pattern = re.compile(r'^[0-9]{3}$')
    if not (re.search(pattern,flag)):
        warnings.warn("error data format: wrong flag format")
        return (flag)
    flag_=flag_new[2:47]
    flag_1=[x for x in range (0,45,3)]
    flag_2=[x for x in range (2,45,3)]
    v=[]
    for i,j in zip(flag_1,flag_2):
        v.append(flag_[i:j+1])
    for flags in v :
        if(flags==new):
            warnings.WarningMessage("flag already exists")
            return(flag)
            break
    for i in enumerate(v):
        if(v[i]== '000'):
            v[i]= new
            break
    flag = "0." +( v[1] + v[2] + v[3] +v[4] + v[5] + v[6] + v[7] +
                  v[8] +v[9] + v[10] + v[11] + v[12] + v[13] + v[14] + v[0])

    return (flag)



def flags (datavector,digits=2):

    if (all(datavector==float("NaN"))):
        mv=10
    else :
        mv= datavector.max()
        if(mv==float("NaN") or mv<10):
            mv=10
        elif not(mv % 10):
            mv=10*mv
    rmv= 10**ceil(np.log10(mv))-10**(-digits)
    rmv= round(rmv,digits)
    out = "{:,.2f}".format(rmv).strip()
    # hier noch eine fehlerabfrage bzgl format rein, es kÃ¶nnen nicht beli mantissen
berÃ¼cksichtigt werden
    return(out)
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
def setup_variables(outfile,values,name,extname,extunit,header,flag):
    values = values.tolist() # values for the three samples; missing value is None!
    flags = flag
    metadata = DataObject()
    metadata.comp_name=name
    metadata.extname=extname
    metadata.header=header
    metadata.extunit=extunit
    outfile.variables.append(DataObject(values_=values, flags=flags, flagcol=True,
                                        metadata=metadata))
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

7.5. Appendix E: BC RT source apportionment calculation codes

```
# -*- coding: utf-8 -*-
"""
Automated AE33 file import from selected folder. NRT-SA
"""

import pandas as pd
import numpy as np

import os
import sys

import time
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler

import Import_Lvl0
import Lvl0_to_Lvl1

from scipy import stats
import datetime

################################# DEFINITIONS #################################

#-------------------- USER-DEFINED --------------------
class quality_control:
    # values for defining outliers and qualitiy control
    DL = 0                          #detection limit
    upper_AAE = 3                   #upper limit for AAE
    lower_AAE = 0.7                 #lower limit for AAE
    r2 = 0.9                        #correlation coefficient for curve fit
    upper_ratio = 1.5               #upper ratio before/after fiter change
    lower_ratio = 0.5               #lower ratio before/after fiter change

    hfactor = 2                     #ACTRIS harmonization coefficient

    default_tape = "Magee AE33-FT"  #default filter tape (in case tape type is not
mentioned in level 0 data file)

    good_flags = 0                  #flags which are accepted as good data


class SA_values:
    # alpha values used for eBC SA (always 470/950 nm)
    alpha_lf = 0.9
    alpha_sf = 1.68


# which averaging algorithm to use
avg_type = 0        # 0: AE33 internal calculation, 1: general averaging

# where the AE33 data is stored
folder_path = 'C:/Users/tobler_a/Documents/Datalystica/RI Urbans/EBAS files'

#where the txt file should be created
output_path = 'C:/Users/tobler_a/Documents/Datalystica/RI Urbans/AE33 Output'
file_name = 'Test'
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
#-------------------------------------------------------


#----------------------- GENERAL -----------------------
class instrument_settings:
    MAC = [18.47, 14.54, 13.14, 11.58, 10.35, 7.77, 7.19]
    wavelengths = [370, 470, 520, 590, 660, 880, 950]
    S = 0.785e-4        #filter surface area (m^2)
    sigma = 7.77        #sigma_air
#-------------------------------------------------------


###################### FUNCTIONS ######################

class MyHandler(FileSystemEventHandler):
    def wait_till_file_is_created(self, source_path):
        historicalSize = -1
        while (historicalSize != os.path.getsize(source_path)):
            historicalSize = os.path.getsize(source_path)
            time.sleep(1) # Wait

    def on_created(self, event): # when file is created

        #-----------
        #data import
        #-----------
        self.wait_till_file_is_created(event.src_path)      #wait for new file to be 
fully created

        #check file extension, don't import last_data.txt files only .nas
        file_type = os.path.splitext(event.src_path)[-1].lower()
        if file_type != ".nas":
            return 0

        nas = Import_Lvl0.nasa_ames_read(event.src_path)
        print("New file was added - % s." % event.src_path)



        #-----------
        #data preparation/some calculations
        #-----------
        ebasobj_ae33_lvl0 = Import_Lvl0.ebasobj_ae33_lv0(nas)

        ebasobj_ae33_lvl1 = Lvl0_to_Lvl1.ae33_proclvl1(ebasobj_ae33_lvl0, nas, 
quality_control.hfactor, quality_control.default_tape)
        (ebasobj_ae33_lvl1_nasa, ebasobj_ae33_lvl1_df) = ebasobj_ae33_lvl1

        #list of dataframes from imported files
        appended_data_lvl0.append(ebasobj_ae33_lvl0)
        appended_data_lvl1.append(ebasobj_ae33_lvl1_df)

        appended_nasa.append(ebasobj_ae33_lvl1_nasa)

        #append to dataframe which contains only the necessary data (eBC, abs, ATN, 
flags)
        curr_lvl0 = appended_data_lvl0[-1]
        curr_lvl1 = appended_data_lvl1[-1]

        GetCurrData(curr_lvl0, curr_lvl1)
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
        #create one big dataframe with data (for new BC calculation needed)
        all_data = pd.concat([appended_latest_data[i] for i in
range(len(appended_latest_data))], axis=0, ignore_index=True)
        # all_data.apply(pd.to_numeric, errors='ignore').info()

        #average to 15 mins
        avg_data_raw = average_extr(all_data)

        #recalculation of abs
        avg_data = babs_calc(avg_data_raw, nas)

        #AAE calculations
        avg_data.loc[:,'AAE(370/950)'] =
np.log(avg_data['babs1']/avg_data['babs7'])/np.log(950/370)
        avg_data.loc[:,'AAE(470/950)'] =
np.log(avg_data['babs2']/avg_data['babs7'])/np.log(950/470)





        #-----------
        #filter data
        #-----------
        # filter data points below detection limit
        avg_data['flags'].values[(avg_data['avg_eBC6'] < quality_control.DL) &
(avg_data['flags'] == 000)] = 147
        avg_data['flags'].values[avg_data['avg_eBC6'] < 0] = 999      # negative
value due to tape advance within the 15 min avg.


        # filter data points with Angström exponents outside the defined limit
        avg_data['flags'].values[(avg_data['AAE(470/950)'] <
quality_control.lower_AAE) | (avg_data['AAE(470/950)'] > quality_control.upper_AAE)]
= 147


        # filter data points with correlations below defined limit
        babs_forFit = avg_data.loc[:, 'babs1':'babs7'].T
        babs_forFit.index = instrument_settings.wavelengths

        temp_x = np.log(babs_forFit.index/babs_forFit.index[6])

        iter_babs_forFit = iter(babs_forFit)
        next(iter_babs_forFit)       #skip first element, is always NaN (eBC
recalculation)
        for column in iter_babs_forFit:
            temp_y = np.log(babs_forFit[column]/babs_forFit.iloc[6][column])

            slope, intercept, r_value, p_value, std_err = stats.linregress(temp_x,
temp_y)
            avg_data.at[column,'slope'] = slope
            avg_data.at[column,'R2'] = r_value*r_value
        avg_data['flags'].values[(avg_data['R2'] < quality_control.r2) &
(avg_data['flags'] == 000)] = 147



        #-----------
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
        # SA calculations (Sandradewi et al.) using the above defined alpha values
        #-----------
        SA = pd.DataFrame(data=None)
        SA['950_sf'] = (avg_data['babs2'] - ((950/470)**SA_values.alpha_lf *
avg_data['babs7'])) / ((950/470) ** SA_values.alpha_sf - (950/470) **
SA_values.alpha_lf)
        SA['950_lf'] =  avg_data['babs7']  - SA['950_sf']
        SA['470_sf'] = (950/470) ** SA_values.alpha_sf * SA['950_sf']
        SA['470_lf'] =  avg_data['babs2']  - SA['470_sf']

        SA['fraction_lf'] = SA['950_lf'] / avg_data['babs7']
        #set so that fraction is between 0 and 1 (no negative values allowed when
positive eBC6 values)
        SA[SA['fraction_lf'] > 1] = 1
        SA[SA['fraction_lf'] < 0] = 0

        avg_data.loc[:,'BC_lf'] = avg_data.loc[:,'avg_eBC6'] *
SA.loc[:,'fraction_lf']
        avg_data.loc[:,'BC_sf'] = avg_data.loc[:,'avg_eBC6'] -
avg_data.loc[:,'BC_lf']



        #-----------
        #write files
        #-----------
        #write current dataframe in .txt so that it can be imported when restarted
        df_file_name = folder_path + '/' + file_name + '_lastData.txt'
        all_data.to_csv(df_file_name, index=False)

        #write NRT-eBC result file
        if not os.path.exists(output_path):
            os.makedirs(output_path)

        time_now_str = datetime.datetime.now().strftime('%Y_%m_%d_%H_%M_%S')
        new_file_name = output_path + '/' + file_name + '_' + time_now_str + '.txt'

        avg_short = avg_data[['start_time', 'avg_eBC6', 'BC_lf', 'BC_sf', 'flags']]

        with open(new_file_name, "w") as new_file:
            avg_short.to_string(new_file, index=False)



 # ------------------------------------------

def GetParameters():

    curr_import = appended_nasa[-1]

    zfactor = curr_import.metadata['leakage_factor_zeta']

    filtertype= curr_import.metadata['filter_type']
    if (filtertype=="Magee AE33-FT"):
        cfactor = 1.57
    elif (filtertype=="Magee M8050"):
        cfactor = 1.57
    elif (filtertype== "Magee M8060"):
        cfactor = 1.39
    else:
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
        sys.exit("wrong data format: no valid entry found for 'Filter type")

    return(zfactor, cfactor)


 # -------------------------------------------

def GetCurrData(curr_lvl0, curr_lvl1):

    #time
    new_data = pd.DataFrame(curr_lvl1['start_time'])

    #flags
    new_data['flags'] = curr_lvl1['numf']

    #flow, C, Z
    #get missing parameters from metadata
    (Z, C) = GetParameters()
    zfactor.append(Z)
    cfactor.append(C)
    new_data['flow1'] = curr_lvl0.iloc[:,3]
    new_data['cfactor'] = cfactor[-1]
    new_data['zfactor']=  zfactor[-1]
    new_data['hfactor']=  quality_control.hfactor

    #k parameter
    new_data['k1'] = curr_lvl0.iloc[:,22]
    new_data['k2'] = curr_lvl0.iloc[:,31]
    new_data['k3'] = curr_lvl0.iloc[:,40]
    new_data['k4'] = curr_lvl0.iloc[:,49]
    new_data['k5'] = curr_lvl0.iloc[:,58]
    new_data['k6'] = curr_lvl0.iloc[:,67]
    new_data['k7'] = curr_lvl0.iloc[:,76]

    #ATN
    new_data['att1_1'] = curr_lvl0.iloc[:,23].astype(float)
    new_data['att2_1'] = curr_lvl0.iloc[:,32].astype(float)
    new_data['att3_1'] = curr_lvl0.iloc[:,41].astype(float)
    new_data['att4_1'] = curr_lvl0.iloc[:,50].astype(float)
    new_data['att5_1'] = curr_lvl0.iloc[:,59].astype(float)
    new_data['att6_1'] = curr_lvl0.iloc[:,68].astype(float)
    new_data['att7_1'] = curr_lvl0.iloc[:,77].astype(float)

    # new_data.apply(pd.to_numeric, errors='ignore').info()

    appended_latest_data.append(new_data)



# -------------------------------------------

def average_extr(all_data):
    avg_data = pd.DataFrame(all_data[(all_data['start_time'].dt.minute == 0) |
(all_data['start_time'].dt.minute == 15) | (all_data['start_time'].dt.minute == 30) |
(all_data['start_time'].dt.minute == 45)])

    for i in range(len(instrument_settings.wavelengths)):
        newBC_str = 'avg_eBC'+str(i+1)
        ATN_str = 'att'+str(i+1)+'_1'
        K_str = 'k'+str(i+1)
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
        avg_data.loc[:,newBC_str] =
(instrument_settings.S*((avg_data[ATN_str].diff(1))/100))/(avg_data['flow1']*10**-
6*(1-avg_data['zfactor'])*instrument_settings.sigma*avg_data['cfactor']*(1-
avg_data[K_str]*avg_data[ATN_str])*15)*10**3

    # appended_avg_data.append(avg_data_extr)

    return(avg_data)

# ------------------------------------------

def babs_calc(avg_data, nas):
    mac = Lvl0_to_Lvl1.ae33_extract_mac(nas)

    for i in range(len(instrument_settings.wavelengths)):
        BC_str = 'avg_eBC'+str(i+1)
        babs_str = 'babs'+str(i+1)
        avg_data.loc[:, babs_str] =
(avg_data[BC_str]*mac[i]/(avg_data['cfactor']*avg_data['hfactor']))


    appended_avg_data.append(avg_data)

    return(avg_data)


################################# EXECUTION #################################

appended_data_lvl0 = []
appended_data_lvl1 = []
appended_nasa = []

zfactor = []
cfactor = []

appended_latest_data = []
appended_avg_data = []

#check if data from previous run is available
previous_data_str = folder_path + '/' + file_name + '_lastData.txt'
if os.path.isfile(previous_data_str) :
    old_data = pd.read_csv(previous_data_str,parse_dates=[0])
    appended_latest_data.append(old_data)


# check permanently for new files in defined input folder
observer = Observer()
event_handler = MyHandler() # create event handler
# set observer to use created handler in directory
observer.schedule(event_handler, path = folder_path, recursive=False)

observer.start()
# sleep until keyboard interrupt, then stop + rejoin the observer
try:
    while True:
        time.sleep(10)
except KeyboardInterrupt:
    observer.stop()
    print("Observer Stopped")
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
observer.join()
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

7.6. Appendix F: Determination of site-specific α-values

The "aethalometer model" by Sandradewi et al. (2008) separates liquid and solid fuel combustion based on the fact that these two combustion sources exhibit a different light absorption wavelength dependence. The absorption exponent (α-value) for traffic (liquid fuel) is typically characterized by values between 0.9 and 1.1, whereas solid fuel is less well-characterized because it depends not only on the combustion material but also on the combustion efficiency and the burning conditions, e.g. values up to 2.2 (Sandradewi et al., 2008) can be found in literature So for this reason, using side-specific α-value is crucial. Since for most sites, ^{14}C measurement for the determination of these values are not available, the Angström absorption exponent (AAE) distribution could be used to determine site-specific α-values by applying stringent filter of r^2>0.99 for the fit of ln(babs) vs the ln(wavelength) and use the tails for this filtered data for the site-specific α-values.

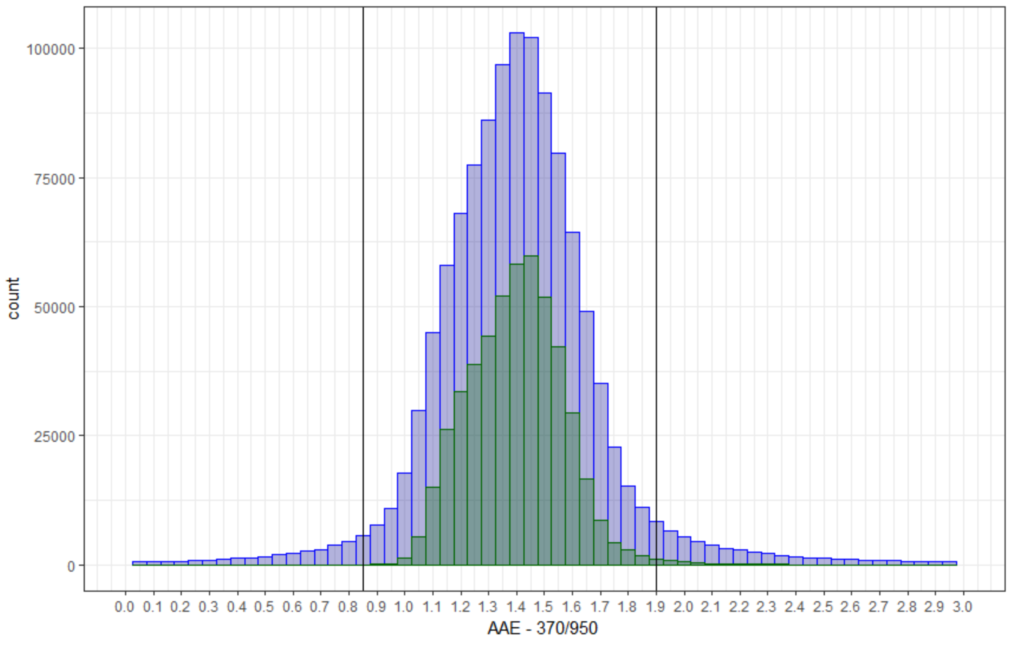


Figure C.1. AAE calculated from the ratio of the 370 nm and 950 nm channels (blue) and from the fit where the values were filtered for r^2>0.99.

The python script **AAE_dist_past.py** should be used offline to determine the site-specific α-values. With this code, past data can be loaded to python and the histogram can be plotted. Since the code is also based on NASA Ames files (.nas), it requires the **Import_Lvl0.py** and **Lvl0_to_Lvl1.py** python script to run properly. The assumption of determining the α-values like that is that there is almost pure liquid and solid combustion at both ends of the distribution, respectively. Therefore, probably months of data should be loaded, especially during summer months, where the absence of wood combustion could become a problem otherwise.

```
# -*- coding: utf-8 -*-
"""
Import all AE33 file from folder and calculate AAE and plot the histogram to get the
station-specific alpha values.
"""

import glob
import pandas as pd
import numpy as np
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
from scipy import stats
import matplotlib.pyplot as plt

import Import_Lvl0
import Lvl0_to_Lvl1

################################################################################
# some definitions
MAC = [18.47, 14.54, 13.14, 11.58, 10.35, 7.77, 7.19]
wavelengths = [370, 470, 520, 590, 660, 880, 950]


# values for defining outliers and qualitiy control
DL = 0                  #detection limit
upper_AAE = 3           #upper limit for AAE
lower_AAE = 0.7         #lower limit for AAE
r2 = 0.9                #correlation coefficient for curve fit
upper_ratio = 1.5       #upper ratio before/after fiter change
lower_ratio = 0.5       #lower ratio before/after fiter change


# where the AE33 data is stored
folder_path = 'C:/Users/tobler_a/Documents/Datalystica/RI Urbans/EBAS Files'

################################################################################

# some functions
def getAllData(all_files):
    appended_data_lvl0 = []
    appended_data_lvl1 = []

    for file in all_files:
        #use functions from Mohamed to get level 1 data

        nas = Import_Lvl0.nasa_ames_read(file)
        ebasobj_ae33_lvl0 = Import_Lvl0.ebasobj_ae33_lv0(nas)

        ebasobj_ae33_lvl1 = Lvl0_to_Lvl1.ae33_proclvl1(ebasobj_ae33_lvl0, nas)

        (ebasobj_ae33_lvl1_nasa, ebasobj_ae33_lvl1_df) = ebasobj_ae33_lvl1

        appended_data_lvl0.append(ebasobj_ae33_lvl0)
        appended_data_lvl1.append(ebasobj_ae33_lvl1_df)

    total_lvl0  = pd.concat(appended_data_lvl0, ignore_index=True)
    total_lvl1  = pd.concat(appended_data_lvl1, ignore_index=True)

    return(total_lvl0, total_lvl1)

 ################################




########################  EXECUTION  ########################

# load data from all .nas files in indicated folder
all_files = glob.glob(folder_path + "/*.nas")
data = getAllData(all_files)
(Lvl0, Lvl1) = data
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245

```
# AAE calculations
Lvl1.loc[:,'AAE(370/950)'] = np.log(Lvl1['abs370']/Lvl1['abs950'])/np.log(950/370)
Lvl1.loc[:,'AAE(470/950)'] = np.log(Lvl1['abs470']/Lvl1['abs950'])/np.log(950/470)


#AAE histogram
plt.subplot(1, 2, 1)
plt.hist(Lvl1['AAE(370/950)'], bins=np.arange(min(Lvl1['AAE(370/950)']),
max(Lvl1['AAE(370/950)']) + 0.05, 0.05), color='silver')
plt.xlabel('AAE(370/950)')
plt.ylabel('count')

plt.subplot(1, 2, 2)
plt.hist(Lvl1['AAE(470/950)'], bins=np.arange(min(Lvl1['AAE(470/950)']),
max(Lvl1['AAE(470/950)']) + 0.05, 0.05), color='silver')
plt.xlabel('AAE(470/950)')
plt.ylabel('count')

plt.tight_layout()

#AAE histogram only filtered data (R^2 > 0.99 over the 7 babs)
babs_forFit = Lvl1.loc[:, 'abs370':'abs950'].T
babs_forFit.index = wavelengths

for column in babs_forFit:
   temp_x = np.log(babs_forFit.index/babs_forFit.index[6])
   temp_y = np.log(babs_forFit[column]/babs_forFit.iloc[6][column])

   slope, intercept, r_value, p_value, std_err = stats.linregress(temp_x, temp_y)
   Lvl1.at[column,'slope'] = slope
   Lvl1.at[column,'R2'] = r_value*r_value
AAE_filtered = Lvl1.drop(Lvl1[Lvl1.R2 < 0.99 ].index)

plt.subplot(1, 2, 1)
plt.hist(AAE_filtered['AAE(370/950)'],
bins=np.arange(min(AAE_filtered['AAE(370/950)']), max(AAE_filtered['AAE(370/950)']) +
0.05, 0.05), color='teal')
plt.subplot(1, 2, 2)
plt.hist(AAE_filtered['AAE(470/950)'],
bins=np.arange(min(AAE_filtered['AAE(470/950)']), max(AAE_filtered['AAE(470/950)']) +
0.05, 0.05), color='teal')
….
```

RI-URBANS (www.RIURBANS.eu) is supported by the European Commission under the Horizon 2020 – Research and Innovation Framework Programme, H2020-GD-2020, Grant Agreement number: 101036245