Python Processing library update

Alberto Berjón, Alberto Redondas, Bentorey Hernández, Virgilio Carreño, M. Rodríguez Valido, Javier López-Solano, Daniel Santana & Sergio F. León-Luis.

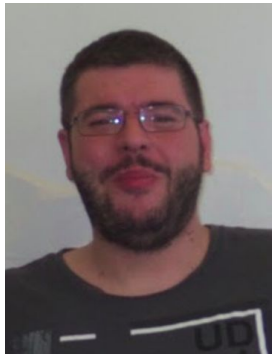
- This project started as a collaboration between the RBCC-E and Kipp & Zonen.
- The main objective was to develop a library for Brewer data processing which can be reused in different projects.
- Now It is integrated in the EUBREWNET DB (http://rbcce.aemet.es/eubrewnet/)
- This library is freely distributed to the Brewer community. Therefore it has paid special attention to the library documentation and readability of the code.
- The standard algorithm for Brewer data processing is based on the version 4.1 of the brewer software, written in GWBASIC, and the Brewer MkIII - Operator's Manual (revE), available through the Kipp & Zonen web page (http://www.kippzonen.com).

GOBIERNO DE ESPAÑA
MINISTERIO DE AGRICULTURA, ALIMENTACIÓN Y MEDIO AMBIENTE
AEMet Agencia Estatal de Meteorología

Development Team

Alberto Redondas
aredondasm@aemet.es

Alberto Berjón
aberjona@ull.es



Bentorey Hernández
bhernandez@fg.ull.es

The documentation of the Python library for the Brewer data processing is published on the EUBREWNET documentation page:

http://rbcce.aemet.es/dokuwiki/doku.php?id=devel:brewerpythonmodule

The last version of the brewer python is available from:

http://rbcce.aemet.es/svn/python/libbrewer/

The Brewer module provide functions to analyse Brewer data, but it doesn't provide any function to retrieve the data. For this purpose, we have developed another python module:

http://rbcce.aemet.es/svn/python/libbrewerjson/

This module download data from EUBRENET DB.

Importing python module:

1. Putting brewer.py module in the same directory
2. Adding the path to brewer.py module

```
import sys
sys.path.append("/home/to/module/")
```

3. Putting brewer.py module in the standard location for third-party Python modules

Platform	Standard installation location
Windows	C:\PythonXY\Lib\site-packages
Unix	/usr/local/lib/pythonX.Y/site-packages

The module is designed as a set of functions used to obtain derived products from Brewer data. These functions use structured data in dictionaries (individual measures) or lists of dictionaries.

A dictionary in Python is a collection of unordered values accessed by key rather than by index:

{u'raw_counts_w4': 1075885, u'raw_counts_w5': 1190052, u'raw_counts_w0': 628562, u'raw_counts_w1': 699934, u'raw_counts_w2': 804122, u'raw_counts_w3': 980275, u's_id': 36962, u'airmass': 2.54, u'nd_filter_position': 0, u'mmmm_gmt': 369.87, u'dark_count': 204, u'gmt': datetime.datetime(2015, 1, 1, 6, 9, 52), u'double_ratio1': 434, u'double_ratio2': 308, u'upper_slit': 6, 'n_summary': 36962, u'mmmm': 300.115459096786, u'date': datetime.date(2015, 1, 1), u'single_ratio4': 463, u'single_ratio1': 1911, u'single_ratio3': 419, u'single_ratio2': 1299, u'lower_slit': 0, u'temp': 17.0, u'zenith_angle': 112.721, u'filter': u'a', u'cycles': 20}

- JSON
  - raw_counts_w4 : 1075885
  - raw_counts_w5 : 1190052
  - raw_counts_w0 : 628562
  - raw_counts_w1 : 699934
  - raw_counts_w2 : 804122
  - raw_counts_w3 : 980275
  - s_id : 36962
  - airmass : 2.54
  - nd_filter_position : 0
  - mmmm_gmt : 369.87
  - dark_count : 204
  - gmt : "2015/01/01 06:09:52"
  - double_ratio1 : 434
  - double_ratio2 : 308
  - upper_slit : 6
  - n_summary : 36962
  - mmmm : 300.115459096786
  - date : "2015/01/01"
  - single_ratio4 : 463
  - single_ratio1 : 1911
  - single_ratio3 : 419
  - single_ratio2 : 1299
  - lower_slit : 0
  - temp : 17
  - zenith_angle : 112.721
  - filter : "a"
  - cycles : 20

Path: JSON

Key: JSON : copy


```
{
   "raw_counts_w4": 1075885,
   "raw_counts_w5": 1190052,
   "raw_counts_w0": 628562,
   "raw_counts_w1": 699934,
   "raw_counts_w2": 804122,
   "raw_counts_w3": 980275,
   "s_id": 36962,
   "airmass": 2.54,
   "nd_filter_position": 0,
   "mmmm_gmt": 369.87,
   "dark_count": 204,
   "gmt": "2015/01/01 06:09:52",
   "double_ratio1": 434,
   "double_ratio2": 308,
   "upper_slit": 6,
   "n_summary": 36962,
   "mmmm": 300.115459096786,
   "date": "2015/01/01",
   "single_ratio4": 463,
   "single_ratio1": 1911,
   "single_ratio3": 419,
   "single_ratio2": 1299
```


modify

goto node | +expand all | -collapse all | expand node

☑ show value ☑ show img ☐ show ico

Brewerjson.py module: Auxiliary Functions

Function	Short Description
getDataList()	Retrieve brewer data from EUBREWNET DB using Basic Authentication.
writejson()	Write list of dictionaries (measures, config,..) to file.
readjson()	Read list of dictionaries (measures, config,..) from file.

Functions from brewer.py module use four different data structures for both input and output. These structures are **python dictionaries**:

Ozone data structures
measure (DS, SL)
config
hg
bdaily

UV data structures
uv
uvr

MEASURE (imput)

Field	Description
'gmt'	datetime object
'nd_filter_position'	ND filter position of #2 Filterwheel (in steps)
'cycles'	# of cycles
'dark_count'	dark count
'raw_counts_w0'	raw counts wavelength #0
'raw_counts_w1'	raw counts wavelength #1
'raw_counts_w2'	raw counts wavelength #2
'raw_counts_w3'	raw counts wavelength #3
'raw_counts_w4'	raw counts wavelength #4
'raw_counts_w5'	raw counts wavelength #5
'temp'	Brewer temperature
'n_summary'	Summary identifier to which the measure belongs

MEASURE (output)

Field	Description
'single_ratio1'	single ratio #1 MS(4)
'single_ratio2'	single ratio #2 MS(5)
'single_ratio3'	single ratio #3 MS(6)
'single_ratio4'	single ratio #4 MS(7)
'double_ratio1'	double ratio #1 MS(8)
'double_ratio2'	double ratio #2 MS(9)
'measure_w0'	c/s for wl#0 corrected by instrument effects
'measure_w1'	c/s for wl#1 corrected by instrument effects
'measure_w2'	c/s for wl#2 corrected by instrument effects
'measure_w3'	c/s for wl#3 corrected by instrument effects
'measure_w4'	c/s for wl#4 corrected by instrument effects
'measure_w5'	c/s for wl#5 corrected by instrument effects

Field	Description
'zenith_angle'	solar zenith angle
'apparent_zenith_angle'	solar zenith angle corrected by atmospheric refraction
'azimuth_angle'	solar azimuth angle
'airmass'	air mass for ozone layer
'airmass_rayleigh'	air mass for molecular atmosphere
'o3'	ozone
'so2'	sulphur dioxide
'o3_sl'	ozone corrected by SL
'so2_sl'	sulphur dioxide corrected by SL
'o3_slcor'	SL correction for ozone
'so2_slcor'	SL correction for sulphur dioxide
'hg_flag'	Flag from HG measurement

CONFIG

Field	Description
'latitude'	Site latitude in decimal degrees (positive latitudes are north of the equator)
'longitude'	Site longitude in decimal degrees (positive longitudes are west of the Prime Meridian)
'press'	Pressure
'o3tempcoef'	Ozone temperature coefficient for slit 0
'oslit1'	Ozone temperature coefficient for slit 1
'oslit2'	Ozone temperature coefficient for slit 2
'oslit3'	Ozone temperature coefficient for slit 3
'oslit4'	Ozone temperature coefficient for slit 4
'oslit5'	Ozone temperature coefficient for slit 5
'o3o3rate'	Ozone on ozone ratio
'so2so2rate'	SO 2 on SO 2 ratio

Field	Description
'o3so2rate'	Ozone on SO 2 ratio
'etco3rate'	ETC on ozone ratio
'etcso2rate'	ETC on SO 2 ratio
'deadtime'	Dead time (seconds)
'nfilter0'	Neutral density of filter 0
'nfilter1'	Neutral density of filter 1
'nfilter2'	Neutral density of filter 2
'nfilter3'	Neutral density of filter 3
'nfilter4'	Neutral density of filter 4
'nfilter5'	Neutral density of filter 5
'it'	slit sampling time

CONFIG

Field	Description
'be2'	Rayleigh optical depth (multiplied by 10000*log10(e)) for wavelength #1
'be3'	Rayleigh optical depth (multiplied by 10000*log10(e)) for wavelength #2
'be4'	Rayleigh optical depth (multiplied by 10000*log10(e)) for wavelength #3
'be5'	Rayleigh optical depth (multiplied by 10000*log10(e)) for wavelength #4
'be6'	Rayleigh optical depth (multiplied by 10000*log10(e)) for wavelength #5
'w_so2_0'	weight for double ratio #1 MS(8) calculation, for wavelength #0
'w_so2_2'	weight for double ratio #1 MS(8) calculation, for wavelength #1
'w_so2_3'	weight for double ratio #1 MS(8) calculation, for wavelength #2
'w_so2_4'	weight for double ratio #1 MS(8) calculation, for wavelength #3
'w_so2_5'	weight for double ratio #1 MS(8) calculation, for wavelength #4

Field	Description
'w_so2_6'	weight for double ratio #1 MS(8) calculation, for wavelength #5
'w_o3_0'	weight for double ratio #2 MS(9) calculation, for wavelength #0
'w_o3_2'	weight for double ratio #2 MS(9) calculation, for wavelength #1
'w_o3_3'	weight for double ratio #2 MS(9) calculation, for wavelength #2
'w_o3_4'	weight for double ratio #2 MS(9) calculation, for wavelength #3
'w_o3_5'	weight for double ratio #2 MS(9) calculation, for wavelength #4
'w_o3_6'	weight for double ratio #2 MS(9) calculation, for wavelength #5
'sl_r5'	Calibration reference for double ratio #1 of standard lamp
'sl_r6'	Calibration reference for double ratio #2 of standard lamp
'wstepn'	Micrometer set on installation to this step #

- Functions
  - Products
  - Control
  - Astronomical
  - Airmass
  - Data Reduction
  - Auxiliary Functions

Brewer.py module: Products

Function	Short Description
airmass_calc()	Calculates sun position and air mass (rayleigh and ozone) for brewer individual measurements.
airmass_calcl()	Calculates sun position and air mass (rayleigh and ozone) for a list of brewer measurements.
instrumental_cor()	Applies instrumental corrections (darkcount, deadtime, temperature, filter attenuation) to brewer individual measurements.
instrumental_corl()	Applies instrumental corrections (darkcount, deadtime, temperature, filter attenuation) to a list of brewer measurements.
rayleigh_cor()	Applies Rayleigh corrections to brewer instrumentally corrected individual measurements.
rayleigh_corl()	Applies Rayleigh corrections to a list of brewer instrumentally corrected measurements.
o3_so2()	Calculates O_3 and SO_2 from the brewer corrected individual measurements.
o3_so2_l()	Calculates O_3 and SO_2 from a list of brewer corrected measurements.
o3_so2_sl()	Calculates Standard Lamp correction for O_3 and SO_2.
uv_cal()	Calculates UV Irradiance and DUV.

Brewer.py module: Control

Function	Short Description
check_hg()	Returns a list with the HG flags.
sl_dailymedian()	Returns the daily median double ratio of SL.
sl_tma()	Returns the triangular moving average double ratio of SL.
temp_filter()	Returns a filtered list of measurements based on the temperature.
gmt_filter()	Returns a filtered list of measurements based on the time (GMT).
outliers_filter()	Returns a list of measurements removing outliers.
measure2summary()	Returns the summary everaging in a list of measurements.
tempfit()	Returns the linear regression between instrumentally corrected measurements and temperature.
tempanalysis()	Compares results for old, new and zero temperature coefficientes.

Brewer.py module: Astronomical

Function	Short Description
solar_ha_dec_et()	Returns the solar hour angle, declination and equation of time.
solar_zenith_angle()	Returns the solar zenith angle.
solar_azimuth_angle()	Returns the solar azimuth angle.
moon_ha_dec()	Returns the moon hour angle and declination.
moon_zenith_angle()	Returns the moon zenith angle.
moon_azimuth_angle()	Returns the moon azimuth angle.
min_sza()	Returns the minimum solar zenith angle (maximum elevation).
sza_to_time()	Returns the times at wich solar zenith angle is achieved for a specific day.
apparent_zenith_angle()	Returns zenith angle of the sun/moon corrected by refraction, also known as apparent zenith angle.

Brewer.py module: Airmass

Function	Short Description
brewer_airmass()	Returns the air mass calculated for a delta layer at a fixed altitude.

Brewer.py module: Data Reduction

Function	Short Description
darkcount_cor()	Returns a list with the dark count corrected measurements.
deadtime_cor()	Returns a list with the dead time corrected measurements.
temp_cor()	Returns a list with the temperature corrected measurements.
attenuation_cor()	Returns a list with the filter attenuation corrected measurements.

Brewer.py module: Auxiliary Functions

Function	Short Description
wsum()	Returns the weighted sum from all measurements on a list.
weeknum()	Returns the weighted sum from all measurements on a list.
weektodate()	Returns the weighted sum from all measurements on a list.

Example of use:

```
import brewer
import brewerjson
import datetime

brewerid=185
user='azores'
password='azowork'
date1=datetime.date(2016,4,15)
date2=datetime.date(2016,4,20)

ds=brewerjson.getDataList(brewerid,user,password,"DS",date1,date2)
config=brewerjson.getDataList(brewerid,user,password,"ConfigbyDate",date1,date2)

ds=brewer.airmass_calcl(ds,config)
ds=brewer.instrumental_corl(ds,config)
ds=brewer.rayleigh_corl(ds,config)
ds=brewer.o3_so2_l(ds,config)
```

Thank you for your attention