MET_hobo Documentation

Release 0.2

Greg Cohn

Nov 25, 2021

Contents

Contents:

CHAPTER 1

Intro

1.1 MET_hobo

This module processes data from hobo data loggers preforming basic QAQC and creating a directory of .hobo and CSV files formatted for easy import into GCE. Erroneous text and empty columns are removed, timezone is check and converted to user defined value, measurement units are checked and values converted to user defined units, and timestep is synced to a standardized interval. All files are removed from the source directory and filed for storage. All file movement is tracked in log files.

CHAPTER 2

Config File

2.1 Assigning directories

`MET_hobo/file_path.config` must be **edited by each user.** It defines 3 directories:

- `dir_source_files` - QAQC will be attempted on every csv file in this directory. The module contains options for wiping original files from this directory. This can be a server path.
- `dir_local_processing` - A temporary working directory. This directory is populated during processing, and wipes all temporary files and folders when processing is complete. It is recommended that this directory be local to the machine running the module.
- `dir_final_storage` - A directory where processed files, and any non-csv files, will be ultimately saved

> **Warning:** The file is directly executed by Python, and must follow standard Python syntax, or it will generate an error.

2.2 Other Config Parameters

2.2.1 time_step

Each file will be synchronize to whole values of this interval. Must be a Pandas timeseries string.

2.2.2 map_fname2dir

Module contains options to use this parameter. This Python dictionary is used to map to final storage directory. The key (left side of :) is an identifying string of characters that will be in every file name. The values (right side of :) are a project name associated with those files. This will then place all identified files into <dir_final_storage>//<value>//<filename left of "_">//<filename>

CHAPTER 3

Python Module

3.1 file_manager Module

This module makes a batch call to QAQC methods devolped to process csv files created by HOBO sensors at meteorological sites on the HJ Andrews experimental forest. It also preforms other file storage and management functions. For a specified directory, it processes all files and creates a directory of new, processed csv files.

QAQC methods are imported from `hobo_qaqc.HOBOdata.reformat_HOBO_csv()`.

When module is called `FileHandling.manage()` is executed.

This module is designed to minimize any read/write times by copying all files locally, preforming all processes, and then transferring files to final directories. This is ideal with external or network drives, but if all directories are local, it will create source and final directories with duplicate file names.

class file_manager.**FileHandling**

Processes all files in assigned directory for timezone, units, and timestep sync, and converts values where necessary. Contains methods for archiving using .zip, wiping directories after processing, and adding to .//metdat directory structure.

Todo: possible change from sys.platform to os.name to decrease package dependencies

possible change from shutil.rmtrees to os.remove os.rmdir

---

copy_to_final_dir (*file_list*, *subdir*, *loc*)

Call OS specific system command to copy from temporary working directory to final storage. Selects files by site using wildcard selection.

Example: RS12*

Parameters

- **file_list** – List of str to select files from. *Example: ['RS12','RS04'] copies files 'RS12*' and 'RS04*'*

- **subdir** – str. Destination subdirectory within final storage directory. Files are moved to here.
- **loc** – str. Directory where files are currently located.

Returns List of strings of each filename copped to the final directory

copy_to_wdir()

Copies source files to local working directory using OS specifc DOS, bash, or shell command. Results are output to log file.

del_files_frm_srcdir()

Wipe all files from the src_dir, defined in file_path.config as dir_source_files. All files and sub- folders in this directory will be wiped.

If source directory and final directory are the same, this process will abort.

> **Warning:** This uses destructive methods which will erase any and all contents of the target directory and any sub- directories within.
>
> shutil.rmtree()

Returns List of strings of each filename wiped from the source directory

del_temp_folders()

This is to wipe temporary processing folders in the working directory. The convention maintained by this module is that all temp folders have the "_" prefix

If any files are still in _processed, and have not been copied to a final storage directory, deletion of this directory will be aborted.

> **Warning:** This uses destructive methods which will erase any and all contents of the target directory and any sub- directories within.
>
> shutil.rmtree()

index_files()

Identify files in source directory. Create list of .hobo, .csv, .log files, and any other file type encountered.

Identify site as any prefix to the left of "_" in filename and generate a list of unique sites.

manage()

Execute file managment.

1. Copy files to working directory (_data).
2. Create list of .csv, .hobo, and .logs files in working directory.
3. Attempt to preform QAQC on all .csv files and transfer to _processed.
4. Create a .zip file for all .hobo files from each site. Disabled per bitbucket issue #10 .
5. Copy all files with .csv, .log, and unknown extension to final storage.
6. Delete temporary folders in working directory.
7. Wipe original source directory. This directory contains files where QAQC was not preformed. Disabled per bitbucket issue #10 .
8. Write log file.

qaqc_csv()

Attempt to QAQC all csv files for timezone, timestep sync, and units.

For list of .csv files generated by *index_files()*, call *hobo_qaqc.HOBOdata.reformat_HOBO_csv()*.

Returns list. strings of filenames processed with \n at end.

Returns int. number of csv files

Returns int. number of files processed

set_log_header()

Create header for log file. Assigns first items to list self.logs.

write_log()

Write log to file. <final storage directory>//logs//hobo_qaqc_<date>.log.

Log is a list of strings until this function is called.

zip_hobo_files()

Collect all files with .hobo extension and write to a zip file in the temp directory _processed.

Naming convetion is <site>_<today's date>.zip, where site is any filename prefix to the left of "_".

For list of .hobo files generated by *index_files()*

Returns List of strings of each filename and it's zipped filename with a \n at the end

Returns int. Count of hobo files

Returns int. Count of zipped files

3.2 hobo_qaqc Module

class hobo_qaqc.**HOBOdata**

Load and process data from HOBO loggers produced by the ONSET company.

Handles csv files exported from the HoboWare program. The native format for HOBO loggers is a .hobo file. This proprietary binary file is not handled here and must be converted to a csv.

This class syncs timesteps, checks time zones, and units, and converts where needed.

export_to_GCE_csv(*csvname*)

Export the HOBO data to a GCE friendly csv file

Parameters **csvname** – str. Filepath to output csv file

format_QAQC_data(*units='SI', tz=-8, tstep='5min'*)

Reformat the data using basic QAQC for SI or US units and time zone consistency regardless of daylight savings.

Parameters

- **units** – str. keyword argument. The desired system of units. Default is 'SI'.
- **tz** – flt. keyword argument. The desired time zone as an offset from Greenwich Mean Time. Default is -8 (PST)
- **tstep** – keyword argument. Interval to round time stamps to. Default '5min'.

Note: tstep is input to the function `HOBOdata.format_sync_timestep()`. Valid types are listed there.

format_intensity(*col='Intensity'*, *unit='Lux'*)
Format light intensity records in desired units

Parameters

- **col** – keyword argument. str. Name of column containing light intensity data. Defaults to 'Intensity'.
- **unit** – keyword argument. str defining desired units. Default is 'Lux' (SI)

format_sync_timestep(*n_min='5min'*)
Sync timestamps to a defined measurement interval. Timestamps are increased to the next defined interval.

Parameters **n_min** – str. keyword argument. Interval to round time stamps to. Default '5min'.

Note: This uses the function ceil to round up to the next interval. The interval provided must match a known type and contain both a number and a letter such as '1D' to round up to the next whole day.

See documentation for valid types[1]

Warning: This will change the index and timestamp of every record.

format_temp(*col='Temp'*, *unit='C'*)
Format temperature records to desired units

Parameters

- **col** – keyword argurment. str. Name of column containing temperature data. Defaults to 'Temp'
- **unit** – keyword argument. str defining desired unit. Default is 'C'

format_timezone(*tz=-8*)
Check that timezone is correct, and if not, adjust the time zone.

Parameters **tz** – a timezone as number of hours offset from Greenwhich Mean Time

get_csv_GMT_offset(*header*, *lineno=-1*)
Get timezone as an offset from Greenwhich Mean Time from the header file

Parameters

- **lineno** – keyword argument. index of header array. Function operates on specified index. Default -1
- **header** – array of header lines where each line is a single string.

Returns string of timezone offset from GMT

Example:

String for PST '-08:00'

get_csv_col(*header*, *lineno=-1*)
Extract column names from csv format

[1] : https://pandas.pydata.org/pandas-docs/stable/timeseries.html#offset-aliases

Parameters

- **header** – array of header lines where each line is a single string.
- **lineno** – keyword argument. index of header array. Function operates on specified index. Default -1

Returns array of column names.

get_csv_intensity_unit (*header*, *lineno=-1*)
Get unit for sunlight intensity

Parameters

- **header** – array of header lines where each line is a single string
- **lineno** – keyword argument. index of header array. Function operates on specified index. Default -1

Returns str defining units for sunlight intensity

get_csv_sn (*header*, *lineno=-1*)

Parameters

- **header** – array of header lines where each line is a single string.
- **lineno** – keyword argument. index of header array. Function operates on specified index. Default -1

Returns str containing serial number

get_csv_temp_unit (*header*, *lineno=-1*)
Get unit for temperature records

Parameters

- **header** – array of header lines where each line is a single string.
- **lineno** – keyword argument. index of header array. Function operates on specified index. Default -1

Returns str with single letter defining units for temperature.

get_header_nlines (*file_name*)
Estimate how many header lines exist in a file

Parameters **file_name** –

Returns int that is index of last header line

> **Warning:** This is a simplistic filter that searches for the first row where there are no quotes and returns line_num - 1 on a 1 based index.
>
> Complex files with quotes around data fields, or no quotes in header lines will not be caught.

Example:

'Plot Title: RS12' '#','Date Time, GMT-07:00','Temp, °C','Intensity, lum/ft^2','Coupler Attached','Stopped','End Of File' 1,11/17/2014 11:10:00 AM,3.472,16.0,,,

returns 2

get_timestamp_col (*col*)
Time stamps can be exported by HOBO into either 1 or 2 columns

Parameters **col** – an array of column names

Returns list of index locations

Returns list of column name(s) that make the timestamp

intensity_lumft2_to_lux(*intensity*)

Convert light intensity records from lumen ft-2 into Lux

Parameters **intensity** – an intensity value or list of intensity values in lumen ft-2

Returns an intensity or list of intensity values in Lux

is_intensity_lux()

Read units definition from header and return True if units are Lux

Returns Boolean. True if light intensity is recorded in Lux

is_temp_celsius()

Read units definition from header and return true if units are celsius

Returns Boolean. True if temperature is recorded in celsius.

is_timezone_correct(*tz*)

Check the timezone in which data was recorded against the expected timezone

Parameters **tz** – a timezone as number of hours offset from Greenwhich Mean Time

Returns Boolean

load_csv_data(*fname*)

Load csv file output by HOBO pendants into a Pandas DataFrame.

Parameters **fname** – str. Filepath of csv data file

read_csv_header(*file_name*)

Read the header lines from the beginning of a file. Reads n_lines, and stores them as headers object.

Parameters **file_name** – str. File path of file to be read.

reformat_HOBO_csv(*infname*, *outfname=None*, *units='SI'*, *tz=-8*, *tstep='5min'*)

Imports a csv file output by HoboWare software and checks for:

- units
- timezone
- time sync (09:07 vs 09:05)

File is converted to specified settings and exported to a GCE friendly format.

Parameters

- **infname** – str. Filename to read
- **outfname** – str. Filename to ouput. Defaults to same as infname
- **units** – str. System of units desired. Defaults to SI
- **tz** – int or flt. Timezone as offset from GMT
- **tstep** – str. Time interval to sync to. Default is '5min'. See `HOBOdata.format_sync_timestep()` or[2] for valid formats.

set_data_GMT_offset(*hr_offset*)

Define time zone of DataFrame timestamps in offset from UTC/GMT

[2] : https://pandas.pydata.org/pandas-docs/stable/timeseries.html#offset-aliases

Parameters **hr_offset** – floating point of time zone in hours difference from Greenwhich Mean Time

temp_F_to_C(*temp*)

Convert temperature records from Fahrenheit

Parameters **temp** – a temperature value or list of temperature values in degrees fahrenheit.

Returns a temperature value or list of temperature values in degrees celsius

3.3 Config

“”“

Warning: Change file paths before use! Copy variable names EXACTLY from example.

Files can be located on a remote server, or locally, however a directory needs to be defined for:

- source files - QAQC will be attempted on every csv file in this directory.
- local processing (should be local to executing console)
- storage of processed data

Within the final storage directory, subdirectories will be created to sort files by project, then site.

Example dir_source_files = “\server/HOBO_DROP/”

dir_local_processing = “C:/HOBO_DROP/”

dir_final_storage = “\server/”

Note: This file is directly executed by Python. Python syntax must be enforced.

time_step format

“”“

map_fname2dir = {“RS”:”REFSTAND”, “TS”:”STREAMT”}

time_step = ‘15min’ # other options for timestep # ‘15min’ ‘10min’ ‘30min’ ‘1H’ ‘1d’ ‘1W’ ‘1m’ # @amkennedy will fill in timestep doc

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

h

Index

T

W

Z