MET_hobo Documentation

Release 0.2

Greg Cohn

Nov 30, 2021

Contents

Contents:

CHAPTER 1

Intro

1.1 MET_hobo

This module preforms basic QAQC on CSV files generated by Onset HOBO's creating a directory of .hobo and CSV files formatted for easy import into the GCE Matlab toolbox.

See module documentation for use.

1.1.1 Data QA Performed:

1. Columns that are empty or have erroneous text are removed,
2. timezone is checked and converted to a user defined value,
3. measurement units are checked and values converted to user defined units, and
4. timestep is synced to a standardized interval; Exp: tstep='5min', 11:05:35 becomes 11:05:00.

1.1.2 File Management Performed

All files are removed from the source directory and filed for storage. File movement is tracked in log files.

1.1.3 Release Notes

- **1.0- development finalized 11/22/21**
  - Released after beta testing in production QA workflow for bird datasets.
  - **What's new:**
    - Allow more flexible directory structures
    - Handle a wider range of input csv formats

        * **Improve error checks and clarify explanations when warnings occur**
            - extra checks for filepath errors
            - extra checks for header formatting
        * Allow *./config* location to be specified at the command line
        * Allow command line override of *./config* settings (time zone, units, etc.)
        * Adjust for package updates for final python 2.x states

- **0.2A- Released 4/16/18**
    - Beta version on development branch for integration in info management work flow
    - **What's new:**
        * Added file management for processing large, multi-project, server-based directories
        * Add processing for hobos without light intensity
        * Add processing for different time steps
        * Add Sphinx docs
        * Add setting control through *./config* file
- **BetaV0.2_hobo_only- Released 1/17/2018**
    - Fork from *verify* repository and remove all modules that do not deal with hobo files
- **BetaV0.1_module- Released 1/17/2018**
    - Last common commit with *verify* repository; verify module split into independent modules to load data from different sensors and preform sensor comparisons

1.2 Accepted File Format

There are many different export options for converting .hobo files to .csv in HOBOware, however not all formats can be processed by this module.

1.2.1 Format Requirements

- U.S. Date formats only: MDY or YMD
- U.S. Number formats only: -1,000.0 or -1000.0

1.2.2 Including GMT

A primary function of this module is correcting time zone. Because each sensor does not have it's own clock and simply counts time from the Launcher or PC clock, daylight savings is often included or excluded inconsistently throughout the life of a sensor. For this reason, time zone (GMT -#) must be included in the header so that it can be converted to the user's standard.

To export time zone from HOBOware:

1. Go to Preferences>>General>>Export Settings
2. *DE-SELECT* option, "No quotes or commas in headings, properties in parentheses"

1.3 Installation

`pip` and `conda install` are not yet supported.

1.3.1 Download the source code

You can directly download the repository by going to Bitbucket and selecting **Downloads** from the left-side menu.

Select a tab:

1. Branches - which will contain the latest changes to master, development, and feature branches
2. Tags - which will contain specific release versions. It is recommended that you use the latest release.

Download the .zip file for the version you want.

1.3.2 Clone the git repository

If you plan on using the program repeatedly, it's a good idea to clone this repository to your desktop using Git. This will allow you to get the most recent changes to the program. It will also allow you to share any changes or improvements that you make with the rest of the community.

There are great tutorials available, but in brief:

1. Install a current version of git or a GUI with git embedded[1]
2. Open a command shell and navigate to a parent directory where you want to store the module
3. Type: `git clone https://<username>@bitbucket.org/hjandrews/met_hobo.git` using your username without the angle brackets.

This creates a directory called `MET_hobo` that contains the module.

1.4 Run the Program

1.4.1 Edit file_path.config

You must define the three file directories (source, working, final) and define the timestep of the data being processed. Take care to use `\\` for Windows file paths or `/` which will work on any operating system. NEVER use `\` which is a special character; example:

```
In[36]: print('c:\new\forest\temp\nc:\\new\\forest\\temp')
Out[36]: c:
        ew orest    emp
        c:\new\forest\temp
```

By default, this file should be located in the top directory of the repository. However, the user can provide a path to this file as a keyword variable in *`file_manager.FileHandling.manage()`*.

[1] : SourceTree or GitHub Desktop

1.4.2 Accessing the Methods

From Python 2.x

Import the module and access the individual classes and methods directly. This allows parameters to be set directly.

file_manager.FileHandling.manage() will run the process on all files in a directory from start to finish.

From Terminal

The entire batch process can be initiated from a terminal if Python is in the system path. *file_manager Module* executes *file_manager.FileHandling.manage()* when executed and parameters are set by *config file*.

```
rem batch execute HOBO QAQC from DOS
python file_manager.py
```

To manually control time step, units, or time zone **from the terminal**:

```
rem edit QAQC settings of batch from DOS
.\>python -c "from file_manager import FileHandling; FileHandling().manage(time_step=
↪'20min')"
```

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 synchronized to whole values of this interval. Must be a Pandas timeseries string.

2.2.2 map_fname2dir

Optional parameter. This Python dictionary is used to map to multiple final storage directories. 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, projname right of "_">/<key, filename left of "_">/<filename>`

Example:

```
```
map_fname2dir = {'RS':'REFSTAND'}
file = 'RS12_20160901.csv'
# creates
<dir_final_storage>/REFSTAND/RS12/RS12_20160901.csv
```
```

CHAPTER 3

Python Module

3.1 file_manager Module

This module preforms QAQC methods in a batch. Methods were developed 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 a final directory which duplicates file names from the source directory.

class file_manager.**FileHandling**(*config='../file_path.config'*)

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 existing directory structure: ./<FileArchive>/<Project>/<Site>.

> **Warning:** Executes ./MET_hobo/file_path.config as Python file and saves variables to class object.

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

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

---

copy_processed_to_final_dir()

Copies processed (QC'd) files from local working directory to final directory using OS specifc DOS, bash, or shell command. Results are output log file.

Directory paths assigned to instance from file_path.config when instance is initialized.

cp <wdir/_processed> <dir_final_storage>

copy_selected_to_site_dir (*file_list*, *subdir*, *loc*)
Call OS specific system command to copy desired files from temporary working directory to final storage. Selects files by site using wildcard selection.

Example:

```
`cp <wdir/_processed/site*>  <dir_final_storage/proj_name/site_name/subdir>`

`cp //NewServer/hoboQA/_processed/RS12* //DataServer/REFSTANDS/RS12/_bulk_
↪export_clean`
```

Warning:

This method was modified per bitbucket issue issue #10 to create a simpler work flow
where file movement is more manually controlled. At 6ec103b it was superseded by *copy_processed_to_final_dir*, removing it from the workflow. It remains as a legacy method still in BETA testing.

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 copied to the final directory

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

Directory paths assigned to instance from file_path.config when instance is initialized. *cp <dir_source_files> <wdir/_data>*

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(*time_step=None*, *units='SI'*, *tz=-8*, *final_subdirs=False*)
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.

3 keyword variables are defined to allow the user to alter *format_QAQC_data()* settings. *units*, and *tz* (time zone) are set to default values, SI units and PST (GMT-8). To change these values, *manage()* must be called directly, through the terminal, or through Python. *time_step*, is defined in the *config file*. This argument only needs to be defined here if the user wants to override the config file at the command line.

qaqc_csv(*time_step=None*, *units='SI'*, *tz=-8*)
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*, *units*, *tz*)

Export the HOBO data to a GCE friendly csv file

Parameters

- **csvname** – str. Filepath to output csv file
- **units** – str. Units of output data. Example: 'SI'.
- **tz** – float. GMT time zone of output data series. Example: -8.

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*, *sep*, *lineno=-1*)
Extract column names from csv format.

From multiple header lines, this extracts a single line, and strips extra info, leaving only column names. File delimiter is used to split header into columns, and ',' is used to split info within a column.

Example:

```
Singles string header:
['"#","Date","Time, GMT-08:00","Temp, °C (LGR S/N: 920980, SEN S/N: 920980)",
↪"Intensity, Lux (LGR S/N:
920980, SEN S/N: 920980)"\n']

becomes a list of column strings:

['#', 'Date', 'Time', 'Temp', '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

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

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_delimiter(*header*, *lineno=-1*)
Find the delimiter used in the csv file.

AS of 3/9/21, the only possible delimiters when exporting from HOBOware are , ; and , . This method tests for which one is used, and returns the answer.

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 delimiter

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

Parameters **file_name** – str containing file path

Returns int that is index of last header line

> **Warning:** This is a simplistic filter that searches for the first row where there are < 8 letters. 8 letters allow for 12 hour time format (AM/PM) plus 'Logged', while separating number data from text headers
>
> Complex files with headers that are numerical and special character, or text data will break the method.

Example:

```
'Plot Title: RS12'
'#','Date Time, GMT-07:00','Temp, °C','Intensity, lum/ft²','Coupler Attached
↪','Stopped','End Of File'
1,11/17/2014 11:10:00 AM,3.472,16.0,Logged,,

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

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 **Example** and paste outside of block quotes. Variable names must match example EXACTLY.

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

Warning: Must use \\ or / in file paths.

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

time_step format

Example:

```
dir_source_files = "//server/bulk_export_CSV"

# Files are initially stored here before being moved to dir_final_storage
# This is used because dir_source_files and dir_final_storage may eventually be␣
↪remote locations where latency may be
# an issue.

dir_local_processing = "C:/HOBO_QA/"
```

(continues on next page)

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

(continued from previous page)

```
dir_final_storage = "//server/bulk_export_clean"

time_step = '5min'

# Optional additional argument maps file prefix to a project directory in dir_final_
↪storage
map_fname2dir = {"RS":"REFSTAND", "TS":"STREAMT"}
```

“””

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

h

Index