aghplctools

Release 4.7.1

Oct 08, 2021

Contents:

The *aghplctools* package is created by the Hein Group at the University of British Columbia (UBC) which provides data types and tools for interacting with Agilent ChemStation.

CHAPTER 1

aghplctools package

1.1 Subpackages

1.1.1 aghplctools.data package

aghplctools.data.batch module

Batch data processing tools

aghplctools.data.batch.**batch_convert_signals_to_csv**(*folder_path: str*, **additional_signals*, *verbose: Union[bool, int] = True*)

Iterates through all .D files in the target directory and writes the signals of those files to csv. Additional signals may be specified to "reprocess" the data.

Parameters

- **folder_path** – folder path to iterate through
- **additional_signals** – additional signals to process. Supported inputs are agilent specification strings (e.g. 'DAD1 A, Sig=210,4 Ref=360,100'), DADSignalInfo objects, or dictionaries of keyword arguments for instantiation of DADSignalInfo objects
- **verbose** – logging flag or level for function (prints progress info to console)

aghplctools.data.batch.**batch_report_text_to_xlsx**(*folder: str*, *watchfor: str = 'Report.TXT'*)

Batch converts all report text files in a directory to xlsx

Parameters

- **folder** – directory to search
- **watchfor** – file name to watch for

aghplctools.data.batch.**pull_hplc_data_from_folder**(*folder*, *targets*, *wiggle=0.01*, *watchfor='Report.TXT'*)

Pulls the HPLC integrations for all report files within the specified directory. This function was designed to pull all data from a given day. This method only pulls data which exists in the reports, which can result in asymmetric data for timepoint analysis (i.e. it assumes that subsequent runs are unrelated to others). If the data in a folder are for time-course analysis, use `pull_hplc_data_from_folder_timepoint()`.

Parameters

- **folder** – The folder to search for report files
- **targets** – target dictionary of the form {'name': [wavelength, retention time], ...}
- **wiggle** – the wiggle time around retention times
- **watchfor** – the name of the report file to watch

Returns dictionary of HPLCTarget instances in the format {'name': HPLCTarget, ...}

Return type dict

aghplctools.data.batch.**pull_hplc_data_from_folder_timepoint**(*folder*, *wiggle=0.02*, *watchfor='Report.TXT'*)

Pulls all HPLC data from a folder assuming that the contents of a folder are from an ordered, time-course run (i.e. the contents of one report are related to the others in the folder). The method will automatically watch for new retention times and will prepopulate appearing values with zeros. The resulting targets will have a consistent number of values across the folder.

Parameters

- **folder** – The folder to search for report files
- **wiggle** – the wiggle time around retention times
- **watchfor** – the name of the report file to watch

Returns dictionary of HPLCTarget instances in the format {wavelength: {retention_time: HPLCTarget, ...}, ...}

Return type dict

aghplctools.data.sample module

Data types for interacting directly with .D samples (e.g. reprocessing, loading signals directly)

class aghplctools.data.sample.**DADSignal**(*wavelength: Union[float, unithandler.base.UnitFloat]*, *bandwidth: Union[float, unithandler.base.UnitFloat] = 1.0*, *reference: Union[DADSignal, aghplctools.data.sample.DADSignalInfo, str] = None*, *name: str = None*, *spectrum: aghplctools.data.sample.DADSpectrum = None*)

Bases: *aghplctools.data.sample.DADSignalInfo*

Class describing a DAD signal and its data.

Parameters

- **wavelength** – wavelength for the signal
- **bandwidth** – band width for the wavelength (signal is centered on the wavelength with this width)

- **reference** – reference information for the signal
- **name** – convenience name for the signal
- **spectrum** – a DADSpectrum object which will be referenced for retrieving data.

as_data_table() → list
Returns the signal as a list-style data table with appropriate headers and data

Returns data table

as_iterable_data_table()
Returns an iterable which yields a data table with appropriate headers and data

Returns data table as iterable

band_string
A string representation of the band specified (e.g. "210 (4) nm")

bandwidth
band width for the signal band

classmethod create_from_DADSignalInfo(*obj: aghplctools.data.sample.DADSignalInfo*, *spectrum: aghplctools.data.sample.DADSpectrum*) → aghplctools.data.sample.DADSignal
generates a DADSignal object from a DADSignalInfo object and a spectrum

mean_referenced_intensities
mean referenced band (mean unreferenced intensities minus the mean intensities of the reference)

mean_unreferenced_intensities
mean unreferenced intensities for the band

reference
reference band for the signal band

retention_times
retention times associated with the intensity array

unreferenced_intensities
unreferenced intensities for the band

wavelength
wavelength for the signal

write_signal_to_csv(*filename: str*, *overwrite: bool = False*) → str
Writes the signal intensities to the specified csv file.

Parameters

- **filename** – file name to write to
- **overwrite** – whether to overwrite the file if it already exists

Returns file path that was written

class aghplctools.data.sample.**DADSignalInfo**(*wavelength: Union[float, unithandler.base.UnitFloat]*, *bandwidth: Union[float, unithandler.base.UnitFloat] = 1.0*, *reference: Union[DADSignalInfo, str] = None*, *name: str = None*)

Bases: object

Class describing a DAD signal and its parameters

Parameters

- **wavelength** – wavelength for the signal
- **bandwidth** – band width for the wavelength (signal is centered on the wavelength with this width)
- **reference** – reference information for the signal
- **name** – convenience name for the signal

DEFAULT_TIME_UNIT = 'min'

DEFAULT_WAVELENGTH_UNIT = 'nm'

agilent_specification_string
the specification string describing this instance (can be passed to create_from_string to reinstantiate)

bandwidth
bandwidth for the signal band

classmethod create_from_CH_file(*file_path: Union[str, pathlib.Path]*) → aghplctools.data.sample.DADSignalInfo
Creates a DADSignal info instance from a channel file.

Parameters **file_path** – target file path

classmethod create_from_agilent_string(*string: str*, *name_override: str = None*) → aghplctools.data.sample.DADSignalInfo
Creates a class instance from a standard Agilent signal description string (e.g. 'DAD1 A, Sig=210,4 Ref=360,100')

Parameters

- **string** – signal description string
- **name_override** – override for name specification

Returns DADSignal object

classmethod get_signals_in_directory(*file_path: Union[str, pathlib.Path]*) → List[aghplctools.data.sample.DADSignalInfo]
Creates a list of signals based on the .CH files in a directory.

Parameters **file_path** – path to target directory

Returns list of signal info objects

classmethod get_values_from_agilent_string(*string: str*) → dict
Parses a standard Agilent signal description string (e.g. 'DAD1 A, Sig=210,4 Ref=360,100') and returns a dictionary of parsed values (can be used to instantiate a DADSignalInfo instance).

Parameters **string** – signal description string

Returns dictionary of parameters

reference
Reference band for the signal band

wavelength
Wavelength for the signal

class aghplctools.data.sample.**DADSpectrum**(*filename=None*, *ftype=None*, *data=None*)
Bases: aston.tracefile.agilent_uv.AgilentCSDAD2

An object describing an Agilent DAD spectrum for a sample. Inherits Aston AgilentCSDAD2 and has additional methods for retrieving band information.

Parameters

- **filename** – target filetype
- **ftype** –
- **data** –

classmethod create_from_D_file(*file_path: Union[pathlib.Path, str]*) → aghplctools.data.sample.DADSpectrum

Creates a DADSpectrum instance from an Agilent .D file

Parameters **file_path** – path to .D sample file

Returns interpreted .D file with metadata and loaded UV data

get_band_intensities(*wavelength: float*, *bandwidth: float = 1.0*) → numpy.ndarray

Retrieve array of values described by the wavelength and band width described. The returned array will have shape [wavelength, retention time]. The corresponding wavelengths are given by DADSpectrum.get_band_wavelengths and the retention times by DADSpectrum.retention_times.

Parameters

- **wavelength** – wavelength
- **bandwidth** – band width

Returns array of band intensities

get_band_mean_intensity(*wavelength: float*, *bandwidth: float = 1.0*) → numpy.ndarray

Retrieve the intensity array described by the wavelength and bandwidth described. The returned array will be the mean of the intensities in the band (wavelength - bandwidth / 2, wavelength + bandwidth / 2).

Parameters

- **wavelength** – wavelength
- **bandwidth** – band width

Returns array of mean intensities

get_band_wavelengths(*wavelength: float*, *bandwidth: float = 1.0*) → list

Returns a list of wavelengths corresponding to the band specified.

Parameters

- **wavelength** – wavelength
- **bandwidth** – band width

Returns

get_component_spectrum(*retention_start: float*, *retention_end: float*) → numpy.ndarray

Retrieves the component spectrum for the provided retention time slice.

Parameters

- **retention_start** – retention time start
- **retention_end** – retention time end

Returns

get_intensities_from_signal(*signal: aghplctools.data.sample.DADSignalInfo*) → numpy.ndarray

Retrieve the intensity array described by the DADSignalInfo object.

Parameters **signal** – signal descriptor

Returns array of mean intensities

maximum_wavelength_array
Array of the wavelengths for the maximum intensity at each retention time

retention_times
retention times corresponding to the data array (min)

total_absorbance_chromatogram
The total absorbance chromatogram for the spectrum (sum of all intensities for each retention time)

wavelengths
list of wavelengths for the DAD

write_to_allotrope(*filename: str*)

class aghplctools.data.sample.**HPLCSample**(*sample_file_name: str, method_name: str, signals: Union[List[aghplctools.data.sample.DADSignalInfo], List[aghplctools.data.sample.DADSignal], List[str]], datetimestamp: Union[str, datetime.datetime] = None, dad_spectrum: aghplctools.data.sample.DADSpectrum = None, ms_spectra: List[aghplctools.data.sample.MSSpectrum] = None, directory: str = None*)

Bases: *aghplctools.data.sample.HPLCSampleInfo*

Data class for describing an HPLC sample containing metadata and spectral data.

Parameters

- **sample_file_name** – name for sample
- **datetimestamp** – date and time stamp for when the sample was run
- **method_name** – name of method used to run the sample
- **signals** – list of signals associated with the run
- **dad_spectrum** – DADSpectrum object with loaded data
- **ms_spectra** – list of mass spectra
- **directory** – directory path where the sample may be found

add_signal(*new_signal: Union[aghplctools.data.sample.DADSignalInfo, dict, str]*) → aghplctools.data.sample.DADSignal
Adds a new signal to the HPLCSample instance.

Parameters **new_signal** – new signal to add. Supported inputs are Agilent specification strings (e.g. 'DAD1 A, Sig=210,4 Ref=360,100') DADSignalInfo objects or a dictionary of keyword arguments for instantiating the same.

Returns the created signal

classmethod create_from_D_file(*file_path: Union[pathlib.Path, str]*) → aghplctools.data.sample.HPLCSample
Creates an HPLCSample instance from a .D file.

Parameters **file_path** – file path to Agilent .D folder

Returns instantiated HPLCSample with loaded data

classmethod create_from_acaml(*acaml: Union[str, xml.etree.ElementTree.ElementTree]*) → aghplctools.data.sample.HPLCSampleInfo

not supported for HPLCSample class

classmethod create_from_xml(*xml_path: Union[str, xml.etree.ElementTree.ElementTree]*) → aghplctools.data.sample.HPLCSampleInfo

Creates sample structure from a Sample.xml file (old style metadata) in the desired .D folder)

Parameters **xml_path** – path to xml file or parsed element tree root

Returns parsed Sample instance

write_signals_to_csv(*directory: Union[str, pathlib.Path] = None*, *overwrite: bool = False*) → List[str]

Writes the signals to csv in the directory specified. If no directory is specified, the csv files will be written to the directory path specified in the directory attribute of the instance.

Parameters

- **directory** – directory path
- **overwrite** – whether to overwrite files if they already exist

Returns file paths written

write_signals_to_xlsx(*output_file: Union[str, pathlib.Path] = None*) → str

Writes the signals to a single excel file.

Parameters **output_file** – target file path. If this is not specified

Returns path to the written file

class aghplctools.data.sample.**HPLCSampleInfo**(*sample_file_name: str, method_name: str, signals: Union[List[aghplctools.data.sample.DADSignalInfo], List[str]], datetimestamp: Union[str, datetime.datetime] = None*)

Bases: object

Data class for describing an HPLC sample.

Parameters

- **sample_file_name** – name for sample
- **datetimestamp** – date and time stamp for when the sample was run
- **method_name** – name of method used to run the sample
- **signals** – list of signals associated with the run

as_dict() → dict

Returns the sample data as a dictionary

classmethod auto_create(*target_path: Union[str, pathlib.Path]*) → aghplctools.data.sample.HPLCSampleInfo

Attempts to automatically create an instance from metadata in the target folder

Parameters **target_path** – path to metadata file or folder containing metadata files

Returns HPLCSampleInfo instance

classmethod create_from_acaml(*acaml: Union[str, xml.etree.ElementTree.ElementTree]*) → aghplctools.data.sample.HPLCSampleInfo

Creates sample structure from an acaml file. (use **sequence.acam_** in the desired .D folder)

Parameters **acaml** – path to acaml file or parsed element tree root

Returns parsed Sample instance

classmethod create_from_xml(*xml_path: Union[str, xml.etree.ElementTree.ElementTree]*) → aghplctools.data.sample.HPLCSampleInfo
Creates sample structure from a Sample.xml file (old style metadata) in the desired .D folder)

Parameters **xml_path** – path to xml file or parsed element tree root

Returns parsed Sample instance

date
date which the sample was run on

static find_acaml(*acaml_path: Union[str, pathlib.Path]*) → xml.etree.ElementTree.ElementTree
Finds an acaml file and loads the element tree

Parameters **acaml_path** – path to acaml file or directory containing acaml file

classmethod find_and_get_metadata(*target_path: Union[str, pathlib.Path]*) → dict
Attempts to locate and parse metadata files in both old (Result.xml) and new (ACAML) formats. If neither file type can be found, an error will be raised.

Parameters **target_path** – target path to search

Returns parsed dictionary for creating HPLCSampleInfo instance

classmethod get_values_from_acaml(*acaml: Union[str, pathlib.Path, xml.etree.ElementTree.ElementTree]*) → dict
Gets relevant values from an acaml file. (use **sequence.acam_** in the desired .D folder)

Parameters **acaml** – path to acaml file or parsed element tree root

Returns dictionary of values of interest

classmethod get_values_from_result_xml(*xml_path: Union[str, pathlib.Path]*) → dict
Retrieves values from a Result.xml file. This is an old-style ChemStation metadata file (~B.04.03 era).

Parameters **xml_path** – path to xml or directory containing xml file

classmethod get_values_from_sample_xml(*xml_path: Union[str, pathlib.Path]*) → dict
Retrieves values from a Sample.xml file. From ChemStation C.01.07

Parameters **xml_path** – path to xml or directory containing xml file

classmethod get_values_from_xml(*xml_path: Union[str, pathlib.Path]*) → dict
Attempts to find a Result.xml file and parse sample information from that.

Parameters **xml_path** – path to xml or directory containing xml file

timestamp
Time of the day when the sample was run

class aghplctools.data.sample.**MSSpectrum**(*filename=None*, *ftype=None*, *data=None*)
Bases: aston.tracefile.agilent_ms.AgilentMS

An object describing an Agilent DAD spectrum for a sample. Inherits Aston AgilentCSDAD2 and has additional methods for retrieving band information.

Parameters

- **filename** – target filetype
- **ftype** –
- **data** –

auto_resolution (*npeaks: int = 4*) → float
Attempts to automatically determine the resolution of the spectrum.

Parameters **npeaks** – number of peakds to try to find

Returns estimated resolution

classmethod create_from_D_file (*file_path: Union[pathlib.Path, str]*) → List[aghplctools.data.sample.MSSpectrum]
Creates a MSSpectrum instance from an Agilent .D file

Parameters **file_path** – path to .D file

Returns instance

extract_function_time_tic ()
duck-type method for PythoMS

functions
duck type function information (expected in PythoMS)

get_ion_intensities (*start_mz: float*, *end_mz: float = None*) → numpy.ndarray
Returns the intensity integral array (reconstructed single ion monitoring) for the provided ion m/z window.

Parameters

- **start_mz** – start m/z ratio for the region
- **end_mz** – end m/z ratio for the region.

get_spectrum_of_retention_period (*start_time: float*, *end_time: float*) → numpy.ndarray
Returns the intensity array for the mass spectrum in the retention time region provided.

Parameters

- **start_time** – start retention time (min)
- **end_time** – end retention time (min)

get_tic_of_function (*function: int*) → numpy.ndarray
duck-type method for retrieving the TIC (expected in PythoMS)

get_timepoints_of_function (*function: int*) → numpy.ndarray
duck-type method for retrieving the timepoints (expected in PythoMS)

masses
array of wavelengths for the DAD

retention_times
retention times corresponding to the data array (min)

summed_intensity_array
returns the summed intensity array of the spectrum

summed_spectrum
returns the mz and summed intensity array for the entire run

aghplctools.data.sample.**bisect_slice** (*array*, *minimum_value: float*, *maximum_value: float*) → Tuple[int, int]
Finds the slice indicies for a minimum and maximum value in an array.

Parameters

- **array** – array like bisectable (assumes sorted)
- **minimum_value** – minimum value

- **maximum_value** – maximum value

Returns slice indicies

aghplctools.data.sample.**check_or_locate_file**(*path: Union[str, pathlib.Path], file_name: str*) → pathlib.Path

Checks whether the provided path points to the provided file name. If not, checks whether the path is a directory and searches for the file in the directory. If there are multiple occurences of the provided file name in a directory, the first is returned.

Parameters

- **path** – path to search
- **file_name** – target file name

Returns path to desired file

aghplctools.data.sample.**retrieve_metadata_from_channel**(*path: Union[str, pathlib.Path]*) → dict

Retrieves metadata from a .CH file

Parameters **path** – path to read

Returns returns a dictionary containing metadata from the channel

aghplctools.data.sample.**strptime_agilent_dt**(*dt_string: str*) → datetime.datetime

Performs strptime on Agilent datetime string

Parameters **dt_string** – agilent datetime strings

Returns parsed datetime object

aghplctools.data.time_course module

Tools for monitoring time-course data (tracking signals over time)

class aghplctools.data.time_course.**HPLCTarget**(*wavelength: float, retention_time: float, name: str = None, wiggle: float = 0.2, zero_pad: int = 0*)

Bases: `object`

A data storage class for tracking the retention time, area, width, and height of a target HPLC retention target over multiple sample acquisitions.

Parameters

- **wavelength** (*float*) – wavelength to track the target on
- **retention_time** (*float*) – retention time to look for the target
- **name** (*str*) – convenience name
- **wiggle** (*float*) – wiggle value in minutes for finding the target around the retention_time (the window will be [retention_time-wiggle, retention_time+wiggle])
- **zero_pad** – adds n zeros to the front of the value lists

add_from_pulled(*signals*, *timepoint=None*)

Retrieves values from the output of the pull_hplc_area function and stores them in the instance.

Parameters

- **signals** (*dict*) – output dictionary from pull_hplc_area
- **timepoint** (*float*) – timepoint to save (if None, the current time will be retrieved)

Returns area, height, width, timepoint

Return type tuple

add_value(*area*, *width=0.0*, *height=0.0*, *timepoint=None*)
Adds a value to the tracker lists.

Parameters

- **area** (*float*) – area to add (required)
- **width** (*float*) – width to add (optional)
- **height** (*float*) – height to add (optional)
- **timepoint** (*float*) – timepoint to use (if None, the current time will be called)

retrieve_index(*index*)
Retrieves the values of the provided index.

Parameters **index** – pythonic list index

Returns {area, width, height, timepoint}

Return type dict

retrieve_timepoint(*timepoint*)
Retrieves the values of the provided timepoint.

Parameters **timepoint** (*float*) – time point to retrieve

Returns {area, width, height, timepoint}

Return type dict

aghplctools.data.time_course.**find_max_area**(*signals*)
Returns the wavelength and retention time corresponding to the maximum area in a set of HPLC peak data.

Parameters **signals** (*dict*) – dict[wavelength][retention time (float)][width/area/height]

Returns

aghplctools.data.time_course.**plot**(*yvalues*, *xvalues=None*, *xlabel='injection #'*, *ylabel=None*, *hline=None*)
plots one set of values :param yvalues: list of y values :param xvalues: list of x values (optional) :param xlabel: label for x :param ylabel: label for y :param hline: plot a horizontal line at this value if specified :return:

aghplctools.data.time_course.**stackedplot**(*rets*, *xlabel='injection #'*)
Creates a stacked plot for the dictionary generated by pull_hplc_data_from_folder :param rets: dictionary of retetion times :param xlabel: optional changing of x label

1.1.2 aghplctools.ingestion package

The `ingestion` module contains modules and methods for parsing and ingesting report files produced by Agilent ChemStation. The exact structure of the exported files may change between installations of ChemStation, but hopefully one of these tools will work for your purposes.

While the report txt and csv files contain metadata information, a more complete set of metadata may be found in the `sequence.acam_` file which may be found in the `*.D` directory. We have written ingesters for this metadata and created data classes around that metadata. You can find these tools in `data.sample`.

aghplctools.ingestion.csv module

The `ingestion.csv` module contains methods for retrieving peak tables (`pull_hplc_area_from_csv`) and metadata (`pull_metadata_from_csv`) from csv report files. The structure of the report csv files (numerous files are generated) appear to follow some arcane method that is difficult to extract context from. These extraction tools have only worked for us on runs which are not externally referenced. Your mileage may vary.

`aghplctools.ingestion.csv.`**`pull_hplc_area_from_csv`**(*folder*, *report_name='Report'*)

Pulls HPLC area data from the specified Agilent HPLC CSV report files. Returns the data tables for each wavelength in dictionary format. Each wavelength table is a dictionary with retention time: peak area format.

Due to the unconventional way Agilent structures its CSV files pulling the data is a bit awkward. In essence, the report consists of one CSV files containing all the metadata, and further CSV files (one per detector signal) containing the data, but without column headers or other metadata. Thus, this function extracts bot data and metadata and stores them in the same format as the text based data parsing.

Parameters

- **`folder`** – The folder to search for report files
- **`report_name`** – File name (without number or extension) of the report file

Returns dictionary *dict[wavelength][retention time (float)][width/area/height]*

`aghplctools.ingestion.csv.`**`pull_metadata_from_csv`**(*folder*, *report_name='Report'*)

Pulls run metadata from the specified Agilent HPLC CSV report files. Returns the metadata describing the sample in dictionary format.

Parameters

- **`folder`** – The folder to search for report files
- **`report_name`** – File name (without number or extension) of the report file

Returns dictionary containing the metadata

aghplctools.ingestion.text module

The *ingestion.text* module contains methods for parsing report text files (named by default `'Report.TXT'`). The method you are most likely to use in this module is `pull_hplc_area_from_txt`. This method pulls an HPLC area table from a text file when provided with a valid file path. This method will return a dictionary with the following structure:

Example Output::

{

wavelength: {

retention_time: { 'Width': float, 'Area': float, 'Height': float, 'Peak': int, 'Type': str, 'RetTime': float,

}

`wavelength` and `retention_time` will be floats, and the type of each value is noted in the above dictionary. Regex matches have been included for each column type we have encountered. If you encounter an error, please create an issue and provide an example file so that we might expand our matching capabilities.

For convenience, a `report_text_to_xlsx` method is included which parses a peak table from a `Report.TXT` file and saves it to an Excel `xlsx` file.

`aghplctools.ingestion.text.`**`build_peak_regex`**(*signal_table: str*)

Builds a peak regex from a signal table

Parameters **signal_table** – block of lines associated with an area table

Returns peak line regex object (<=3.6 _sre.SRE_PATTERN, >=3.7 re.Pattern)

aghplctools.ingestion.text.**chunk_string**(*string*, *n_chars_list*)
Chunks a string by n_characters, returning the characters and the remaining string

Parameters

- **string** (*str*) – string to chunk
- **n_chars_list** (*lst*) – list of number of characters to return

Returns chunk, remaining string

aghplctools.ingestion.text.**parse_area_report**(*report_text: str*) → dict
Interprets report text and parses the area report section, converting it to dictionary.

Parameters **report_text** – plain text version of the report.

Raises **ValueError** – if there are no peaks defined in the report text file

Returns dictionary of signals in the form dict[wavelength][retention time (float)][Width/Area/Height/etc.]

aghplctools.ingestion.text.**pull_hplc_area**(*filename*)
Legacy name for pull_hplc_area_from_txt

Returns dictionary dict[wavelength][retention time (float)][width/area/height]

aghplctools.ingestion.text.**pull_hplc_area_from_txt**(*filename*)
Pulls HPLC area data from the specified Agilent HPLC output file Returns the data tables for each wavelength in dictionary format. Each wavelength table is a dictionary with retention time: peak area format.

Parameters **filename** (*str*) – path to file

Returns dictionary dict[wavelength][retention time (float)][Width/Area/Height/etc.]

aghplctools.ingestion.text.**report_text_to_xlsx**(*target_file: Union[str, pathlib.Path], output_file: Union[str, pathlib.Path] = None*) → str
Ingests the specified report text and outputs it to an excel file.

Parameters

- **target_file** – path to target report text file
- **output_file** – path to output to (if not provided, it will be saved to "Report.xlsx" in the same directory as the report text file

Returns path to the XLSX file that was written

1.2 aghplctools.local_paths module

Automatic determination of local ChemStation paths on this system

class aghplctools.local_paths.**AcquisitionSearch**(*data_path: Union[str, pathlib.Path], cycle_time: float = 1.0, always_search: bool = False, autostart: bool = True*)
Bases: object

An Agilent ChemStation data path monitoring class. This class will monitor for sequence flags and will live-update acquiring status, the current data file, and the current sample number.

Parameters

- **data_path** – file path to the data directory
- **cycle_time** – cycle time to check for updates to the file
- **always_search** – flag to enable continuous searching, even when a file has been located in another instance
- **autostart** – flag to control autostart (if the all started flag is set, setting this to True will prevent the monitor thread from starting and require the user to start the thread manually)

acquiring
whether acquiring is indicated in the target directory

classmethod acquiring_instance() → aghplctools.local_paths.AcquisitionSearch
retrieve the currently acquiring instance

classmethod acquiring_instance_num_and_file() → Tuple[int, pathlib.Path]
Retrieves the current number and file name of the currently acquiring path

Returns sample number, sample path

acquiring_path
path to the acquiring file

current_file
currently acquiring file indicated in acquiring file

static current_num_and_file(*path: Union[str, pathlib.Path]*) → Tuple[int, str]
Returns the current number in the sequence and the name of the data file being acquired.

Parameters **path** – path to parse

Returns current file number, current file name

current_number
current acquiring number indicated in acquiring file

data_paths = []

find_acquiring() → Optional[pathlib.Path]
Locates ACQUIRING.TXT files in the directory. This file appears when ChemStation is acquiring a sequence. The search prioritizes newer subdirectories.

Returns path to acquiring.txt (if found)

classmethod get_by_path(*path: Union[str, pathlib.Path], always_search: bool = False, autostart: bool = True*) → aghplctools.local_paths.AcquisitionSearch
Retrieves an instance by path. If the path is already being monitored, the existing instance is returned. Otherwise creates a new instance.

Parameters

- **path** – pathlike
- **always_search** – flag to enable continuous searching, even when a file has been located in another instance
- **autostart** – flag to control autostart (if the all started flag is set, setting this to True will prevent the monitor thread from starting and require the user to start the thread manually)

instances = []

classmethod kill_all_monitors()
terminates all monitor threads

kill_monitor()
cleanly terminates the monitor thread

newsorted_subdirectories
subdirectories of the root path sorted by date modified in newest to oldest order

classmethod parent_of_any_path(*path: Union[str, pathlib.Path]*) → bool
Checks whether the provided path is a parent of any path instance.

Parameters **path** – path to check

Returns parent of any path

parent_of_path(*path: Union[str, pathlib.Path]*) → bool
Checks whether the provided path is a parent of the instance's path. (The instance's path is a subfolder of the provided path.

Parameters **path** – pathlike

Returns provided path is parent

classmethod sequence_is_running() → bool
True if any acquisition search instance is aware of an acquiring flag file

start_monitor()
starts the acquiring monitor thread

classmethod start_monitoring_all_paths()
starts the monitor thread on all data paths

subdirectories
subdirectories of the root folder

classmethod wait_for_acquiring(*timeout: float = None*, *cycle_time: float = 0.1*) → Optional[aghplctools.local_paths.AcquisitionSearch]
Waits for the acquiring flag file to appear in registered instances. Once found, the acquiring instance is returned.

Parameters

- **timeout** – Optional timeout to prevent eternal waits
- **cycle_time** – cycle time for checks

Returns acquiring instance once located

class aghplctools.local_paths.**ChemStationConfig**(*data_path: Union[str, pathlib.Path] = None*, *core_path: Union[str, pathlib.Path] = None*, *method_path: Union[str, pathlib.Path] = None*, *sequence_path: Union[str, pathlib.Path] = None*, *version: str = None*, *number: int = None*)

Bases: object

A class for managing pathing attributes for ChemStation instances installed on the current system.

Parameters

- **data_path** – default data path for the installation
- **core_path** – core installation path (location of the "CORE" directory)
- **method_path** – path to methods
- **sequence_path** – path to sequences

- **version** – ChemStation version
- **number** – published ChemStation number

DEFAULT_INI_LOCATION = 'C:\\ProgramData\\Agilent Technologies\\ChemStation\\ChemStatio

classmethod construct_from_env(*env_name: str = 'hplcfolder'*)
Constructs an instance from an environment variable name. If the environment variable is not set, no action is taken.

Parameters **env_name** – environment variable name

classmethod construct_from_ini(*ini_path: Union[str, pathlib.Path] = None*) → List[aghplctools.local_paths.ChemStationConfig]
Constructs ChemStation config instances as defined in the provided INI file. If no INI is provided, the default ChemStation INI location will be used.

Parameters **ini_path** – path to INI file location. The provided INI file is expected to have the structure used by ChemStation ini files.

Returns instances created via the ini

core_path
path to the CORE installation folder

data_path
path to the default data directory

classmethod get_by_data_path(*path: Union[str, pathlib.Path]*) → aghplctools.local_paths.ChemStationConfig
Retrieves an instance by its data path. If the data path is not associated with an instance, an error is raised.

Parameters **path** – path to check for

Returns chemstation config instance

method_path
path to the method save directory

registered_chemstations = []

sequence_path
path to the sequence save directory

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

D

E

F

G

H

I

K

M

N

P

R

S

T

U

W