w4h

Release 0.0.22-dev

Author

Apr 30, 2024

CONTENTS:

CHAPTER

ONE

W4H PACKAGE

This is the wells4hydrogeology package.

It contains the functions needed to convert raw well descriptions into usable (hydro)geologic data.

w4h.**add_control_points**(*df_without_control*, *df_control=None*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *zcol='ELEV_FT'*, *controlpoints_crs='EPSG:4269'*, *output_crs='EPSG:5070'*, *description_col='FORMATION'*, *interp_col='INTERPRETATION'*, *target_col='TARGET'*, *verbose=False*, *log=False*, ***kwargs*)

Function to add control points, primarily to aid in interpolation. This may be useful when conditions are known but do not exist in input well database

Parameters

df_without_control
[pandas.DataFrame] Dataframe with current working data

df_control
[str, pathlib.Purepath, or pandas.DataFrame] Pandas dataframe with control points

well_key
[str, optional] The column containing the "key" (unique identifier) for each well, by default 'API_NUMBER'

xcol
[str, optional] The column in df_control containing the x coordinates for each control point, by default 'LONGITUDE'

ycol
[str, optional] The column in df_control containing the y coordinates for each control point, by default 'LATITUDE'

zcol
[str, optional] The column in df_control containing the z coordinates for each control point, by default 'ELEV_FT'

controlpoints_crs
[str, optional] The column in df_control containing the crs of points, by default 'EPSG:4269'

output_crs
[str, optional] The output coordinate system, by default 'EPSG:5070'

description_col
[str, optional] The column in df_control with the description (if this is used), by default 'FORMATION'

interp_col
[str, optional] The column in df_control with the interpretation (if this is used), by default 'INTERPRETATION'

target_col
[str, optional] The column in df_control with the target code (if this is used), by default 'TARGET'

verbose
[bool, optional] Whether to print information to terminal, by default False

log
[bool, optional] Whether to log information in log file, by default False

****kwargs**
Keyword arguments of pandas.concat() or pandas.read_csv that will be passed to that function, except for objs, which are df and df_control

Returns

pandas.DataFrame
Pandas DataFrame with original data and control points formatted the same way and concatenated together

w4h.**align_rasters**(*grids_unaligned=None*, *model_grid=None*, *no_data_val_grid=0*, *verbose=False*, *log=False*)

Reprojects two rasters and aligns their pixels

Parameters

grids_unaligned
[list or xarray.DataArray] Contains a list of grids or one unaligned grid

model_grid
[xarray.DataArray] Contains model grid

no_data_val_grid
[int, default=0] Sets value of no data pixels

log
[bool, default = False] Whether to log results to log file, by default False

Returns

alignedGrids
[list or xarray.DataArray] Contains aligned grids

w4h.**clip_gdf2study_area**(*study_area*, *gdf*, *log=False*, *verbose=False*)

Clips dataframe to only include things within study area.

Parameters

study_area
[geopandas.GeoDataFrame] Inputs study area polygon

gdf
[geopandas.GeoDataFrame] Inputs point data

log
[bool, default = False] Whether to log results to log file, by default False

Returns

gdfClip
[geopandas.GeoDataFrame] Contains only points within the study area

w4h.**combine_dataset**(*layer_dataset*, *surface_elev*, *bedrock_elev*, *layer_thick*, *log=False*)

Function to combine xarray datasets or datarrays into a single xr.Dataset. Useful to add surface, bedrock, layer thick, and layer datasets all into one variable, for pickling, for example.

Parameters

layer_dataset
[xr.DataArray] DataArray contining all the interpolated layer information.

surface_elev
[xr.DataArray] DataArray containing surface elevation data

bedrock_elev
[xr.DataArray] DataArray containing bedrock elevation data

layer_thick
[xr.DataArray] DataArray containing the layer thickness at each point in the model grid

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

xr.Dataset
Dataset with all input arrays set to different variables within the dataset.

w4h.**coords2geometry**(*df_no_geometry*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *zcol='ELEV_FT'*, *input_coords_crs='EPSG:4269'*, *output_crs='EPSG:5070'*, *use_z=False*, *wkt_col='WKT'*, *geometry_source='coords'*, *verbose=False*, *log=False*)

Adds geometry to points with xy coordinates in the specified coordinate reference system.

Parameters

df_no_geometry
[pandas.Dataframe] a Pandas dataframe containing points

xcol
[str, default='LONGITUDE'] Name of column holding x coordinate data in df_no_geometry

ycol
[str, default='LATITUDE'] Name of column holding y coordinate data in df_no_geometry

zcol
[str, default='ELEV_FT'] Name of column holding z coordinate data in df_no_geometry

input_coords_crs
[str, default='EPSG:4269'] Name of crs used for geometry

use_z
[bool, default=False] Whether to use z column in calculation

geometry_source
[str {'coords', 'wkt', 'geometry'}]

log
[bool, default = False] Whether to log results to log file, by default False

Returns

gdf
[geopandas.GeoDataFrame] Geopandas dataframe with points and their geometry values

w4h.**define_dtypes**(*undefined_df*, *datatypes=None*, *verbose=False*, *log=False*)

Function to define datatypes of a dataframe, especially with file-indicated dyptes

Parameters

undefined_df
[pd.DataFrame] Pandas dataframe with columns whose datatypes need to be (re)defined

datatypes
[dict, str, pathlib.PurePath() object, or None, default = None] Dictionary containing datatypes, to be used in pandas.DataFrame.astype() function. If None, will read from file indicated by dtype_file (which must be defined, along with dtype_dir), by default None

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

dfout
[pandas.DataFrame] Pandas dataframe containing redefined columns

w4h.**depth_define**(*df*, *top_col='TOP'*, *thresh=550.0*, *verbose=False*, *log=False*)

Function to define all intervals lower than thresh as bedrock

Parameters

df
[pandas.DataFrame] Dataframe to classify

top_col
[str, default = 'TOP'] Name of column that contains the depth information, likely of the top of the well interval, by default 'TOP'

thresh
[float, default = 550.0] Depth (in units used in df['top_col']) below which all intervals will be classified as bedrock, by default 550.0.

verbose
[bool, default = False] Whether to print results, by default False

log
[bool, default = True] Whether to log results to log file

Returns

df
[pandas.DataFrame] Dataframe containing intervals classified as bedrock due to depth

w4h.**export_dataframe**(*df*, *out_dir*, *filename*, *date_stamp=True*, *log=False*)

Function to export dataframes

Parameters

df
[pandas dataframe, or list of pandas dataframes] Data frame or list of dataframes to be exported

out_dir
[string or pathlib.Path object] Directory to which to export dataframe object(s) as .csv

filename
[str or list of strings] Filename(s) of output files

date_stamp
[bool, default=True] Whether to include a datestamp in the filename. If true, file ends with _yyyy-mm-dd.csv of current date, by default True.

log
[bool, default = True] Whether to log inputs and outputs to log file.

w4h.**export_grids**(*grid_data*, *out_path*, *file_id=''*, *filetype='tif'*, *variable_sep=True*, *date_stamp=True*, *verbose=False*, *log=False*)

Function to export grids to files.

Parameters

grid_data
[xarray DataArray or xarray Dataset] Dataset or dataarray to be exported

out_path
[str or pathlib.Path object] Output location for data export. If variable_sep=True, this should be a directory. Otherwise, this should also include the filename. The file extension should not be included here.

file_id
[str, optional] If specified, will add this after 'LayerXX' or 'AllLayers' in the filename, just before datestamp, if used. Example filename for file_id='Coarse': Layer1_Coarse_2023-04-18.tif.

filetype
[str, optional] Output filetype. Can either be pickle or any file extension supported by rioxarray.rio.to_raster(). Can either include period or not., by default 'tif'

variable_sep
[bool, optional] If grid_data is an xarray Dataset, this will export each variable in the dataset as a separate file, including the variable name in the filename, by default False

date_stamp
[bool, optional] Whether to include a date stamp in the file name., by default True

log
[bool, default = True] Whether to log inputs and outputs to log file.

w4h.**export_undefined**(*df*, *outdir*)

Function to export terms that still need to be defined.

Parameters

df
[pandas.DataFrame] Dataframe containing at least some unclassified data

outdir
[str or pathlib.Path] Directory to save file. Filename will be generated automatically based on today's date.

Returns

stillNeededDF
[pandas.DataFrame] Dataframe containing only unclassified terms, and the number of times they occur

w4h.**file_setup**(*well_data*, *metadata=None*, *data_filename='*ISGS_DOWNHOLE_DATA*.txt'*, *metadata_filename='*ISGS_HEADER*.txt'*, *log_dir=None*, *verbose=False*, *log=False*)

Function to setup files, assuming data, metadata, and elevation/location are in separate files (there should be one "key"/identifying column consistent across all files to join/merge them later)

This function may not be useful if files are organized differently than this structure. If that is the case, it is recommended to use the get_most_recent() function for each individual file if needed. It may also be of use to simply skip this function altogether and directly define each filepath in a manner that can be used by pandas.read_csv()

Parameters

well_data
[str or pathlib.Path object] Str or pathlib.Path to directory containing input files, by default str(repoDir)+'/resources'

metadata
[str or pathlib.Path object, optional] Str or pathlib.Path to directory containing input metadata files, by default str(repoDir)+'/resources'

data_filename
[str, optional] Pattern used by pathlib.glob() to get the most recent data file, by default '*ISGS_DOWNHOLE_DATA*.txt'

metadata_filename
[str, optional] Pattern used by pathlib.glob() to get the most recent metadata file, by default '*ISGS_HEADER*.txt'

log_dir
[str or pathlib.PurePath() or None, default=None] Directory to place log file in. This is not read directly, but is used indirectly by w4h.logger_function()

verbose
[bool, default = False] Whether to print name of files to terminal, by default True

log
[bool, default = True] Whether to log inputs and outputs to log file.

Returns

tuple
Tuple with paths to (well_data, metadata)

w4h.**fill_unclassified**(*df*, *classification_col='CLASS_FLAG'*)

Fills unclassified rows in 'CLASS_FLAG' column with np.nan

Parameters

df
[pandas.DataFrame] Dataframe on which to perform operation

Returns

df
[pandas.DataFrame] Dataframe on which operation has been performed

w4h.**get_current_date**()

1.1 Gets the current date to help with finding the most recent file

Parameters:
None

dateSuffix : str to use for naming output files

w4h.**get_drift_thick**(*surface_elev=None*, *bedrock_elev=None*, *layers=9*, *plot=False*, *verbose=False*, *log=False*)

Finds the distance from surface_elev to bedrock_elev and then divides by number of layers to get layer thickness.

Parameters

surface_elev
[rioxarray.DataArray] array holding surface elevation

bedrock_elev
[rioxarray.DataArray] array holding bedrock elevation

layers
[int, default=9] number of layers needed to calculate thickness for

plot
[bool, default=False] tells function to either plot the data or not

Returns

driftThick
[rioxarray.DataArray] Contains data array containing depth to bedrock at each point

layerThick
[rioxarray.DataArray] Contains data array with layer thickness at each point

w4h.**get_layer_depths**(*df_with_depths*, *surface_elev_col='SURFACE_ELEV'*, *layer_thick_col='LAYER_THICK'*, *layers=9*, *log=False*)

Function to calculate depths and elevations of each model layer at each well based on surface elevation, bedrock elevation, and number of layers/layer thickness

Parameters

df_with_depths
[pandas.DataFrame] Dataframe containing well metdata

layers
[int, default=9] Number of layers. This should correlate with get_drift_thick() input parameter, if drift thickness was calculated using that function, by default 9.

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

pandas.Dataframe
Dataframe containing new columns for depth to layers and elevation of layers.

w4h.**get_most_recent**(*dir=PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/wells4hydrogeology/envs/latest/lib/python packages/w4h/resources')*, *glob_pattern='*'*, *verbose=False*)

Function to find the most recent file with the indicated pattern, using pathlib.glob function.

Parameters

dir
[str or pathlib.Path object, optional] Directory in which to find the most recent file, by default str(repoDir)+'/resources'

glob_pattern
[str, optional] String used by the pathlib.glob() function/method for searching, by default '*'

Returns

pathlib.Path object
Pathlib Path object of the most recent file fitting the glob pattern indicated in the glob_pattern parameter.

w4h.**get_resources**(*resource_type='filepaths'*, *scope='local'*, *verbose=False*)

Function to get filepaths for resources included with package

Parameters

resource_type
[str, {'filepaths', 'data'}] If filepaths, will return dictionary with filepaths to sample data. If data, returns dictionary with data objects.

scope
[str, {'local', 'statewide'}] If 'local', will read in sample data for a local (around county sized) project. If 'state', will read in sample data for a statewide project (Illinois)

verbose
[bool, optional] Whether to print results to terminal, by default False

Returns

resources_dict
[dict] Dictionary containing key, value pairs with filepaths to resources that may be of interest.

w4h.**get_search_terms**(*spec_path='/home/docs/checkouts/readthedocs.org/user_builds/wells4hydrogeology/checkouts/latest/docs/re*
*spec_glob_pattern='*SearchTerms-Specific*'*, *start_path=None*,
*start_glob_pattern='*SearchTerms-Start*'*, *wildcard_path=None*,
*wildcard_glob_pattern='*SearchTerms-Wildcard'*, *verbose=False*, *log=False*)

Read in dictionary files for downhole data

Parameters

spec_path
[str or pathlib.Path, optional] Directory where the file containing the specific search terms is located, by default str(repoDir)+'/resources/'

spec_glob_pattern
[str, optional] Search string used by pathlib.glob() to find the most recent file of interest, uses get_most_recent() function, by default '*SearchTerms-Specific*'

start_path
[str or None, optional] Directory where the file containing the start search terms is located, by default None

start_glob_pattern
[str, optional] Search string used by pathlib.glob() to find the most recent file of interest, uses get_most_recent() function, by default '*SearchTerms-Start*'

wildcard_path
[str or pathlib.Path, default = None] Directory where the file containing the wildcard search terms is located, by default None

wildcard_glob_pattern
[str, default = '*SearchTerms-Wildcard'] Search string used by pathlib.glob() to find the most recent file of interest, uses get_most_recent() function, by default '*SearchTerms-Wildcard*'

log
[bool, default = True] Whether to log inputs and outputs to log file.

Returns

(specTermsPath, startTermsPath, wilcardTermsPath)
[tuple] Tuple containing the pandas dataframes with specific search terms, with start search terms, and with wildcard search terms

w4h.**get_unique_wells**(*df*, *wellid_col='API_NUMBER'*, *verbose=False*, *log=False*)

Gets unique wells as a dataframe based on a given column name.

Parameters

df
[pandas.DataFrame] Dataframe containing all wells and/or well intervals of interest

wellid_col
[str, default="API_NUMBER"] Name of column in df containing a unique identifier for each well, by default 'API_NUMBER'. .unique() will be run on this column to get the unique values.

log
[bool, default = False] Whether to log results to log file

Returns

wellsDF
DataFrame containing only the unique well IDs

w4h.**grid2study_area**(*study_area*, *grid*, *output_crs='EPSG:5070'*, *verbose=False*, *log=False*)

Clips grid to study area.

Parameters

study_area
[geopandas.GeoDataFrame] inputs study area polygon

grid
[xarray.DataArray] inputs grid array

output_crs
[str, default='EPSG:5070'] inputs the coordinate reference system for the study area

log
[bool, default = False] Whether to log results to log file, by default False

Returns

grid
[xarray.DataArray] returns xarray containing grid clipped only to area within study area

w4h.**layer_interp**(*points*, *grid*, *layers=None*, *interp_kind='nearest'*, *return_type='dataarray'*, *export_dir=None*, *target_col='TARG_THICK_PER'*, *layer_col='LAYER'*, *xcol=None*, *ycol=None*, *xcoord='x'*, *ycoord='y'*, *log=False*, *verbose=False*, ***kwargs*)

Function to interpolate results, going from points to grid data. Uses scipy.interpolate module.

Parameters

points
[list] List containing pandas dataframes or geopandas geoadataframes containing the point data. Should be resDF_list output from layer_target_thick().

grid
[xr.DataArray or xr.Dataset] Xarray DataArray or DataSet with the coordinates/spatial reference of the output grid to interpolate to

layers
[int, default=None] Number of layers for interpolation. If None, uses the length ofthe points list to determine number of layers. By default None.

interp_kind
[str, {'nearest', 'interp2d','linear', 'cloughtocher', 'radial basis function'}] Type of interpolation to use. See scipy.interpolate N-D scattered. Values can be any of the following (also shown in "kind" column of N-D scattered section of table here: https://docs.scipy.org/doc/scipy/tutorial/interpolate.html). By default 'nearest'

return_type
[str, {'dataarray', 'dataset'}] Type of xarray object to return, either xr.DataArray or xr.Dataset, by default 'dataarray.'

export_dir
[str or pathlib.Path, default=None] Export directory for interpolated grids, using w4h.export_grids(). If None, does not export, by default None.

target_col
[str, default = 'TARG_THICK_PER'] Name of column in points containing data to be interpolated, by default 'TARG_THICK_PER'.

layer_col
[str, default = 'Layer'] Name of column containing layer number. Not currently used, by default 'LAYER'

xcol
[str, default = 'None'] Name of column containing x coordinates. If None, will look for 'geometry' column, as in a geopandas.GeoDataframe. By default None

ycol
[str, default = 'None'] Name of column containing y coordinates. If None, will look for 'geometry' column, as in a geopandas.GeoDataframe. By default None

xcoord
[str, default='x'] Name of x coordinate in grid, used to extract x values of grid, by default 'x'

ycoord
[str, default='y'] Name of y coordinate in grid, used to extract x values of grid, by default 'y'

log
[bool, default = True] Whether to log inputs and outputs to log file.

****kwargs**
Keyword arguments to be read directly into whichever scipy.interpolate function is designated by the interp_kind parameter.

Returns

interp_data
[xr.DataArray or xr.Dataset, depending on return_type] By default, returns an xr.DataArray

object with the layers added as a new dimension called Layer. Can also specify return_type='dataset' to return an xr.Dataset with each layer as a separate variable.

w4h.**layer_target_thick**(*df*, *layers=9*, *return_all=False*, *export_dir=None*, *outfile_prefix=None*, *depth_top_col='TOP'*, *depth_bot_col='BOTTOM'*, *log=False*)

Function to calculate thickness of target material in each layer at each well point

Parameters

df
[geopandas.geodataframe] Geodataframe containing classified data, surface elevation, bedrock elevation, layer depths, geometry.

layers
[int, default=9] Number of layers in model, by default 9

return_all
[bool, default=False] If True, return list of original geodataframes with extra column added for target thick for each layer. If False, return list of geopandas.geodataframes with only essential information for each layer.

export_dir
[str or pathlib.Path, default=None] If str or pathlib.Path, should be directory to which to export dataframes built in function.

outfile_prefix
[str, default=None] Only used if export_dir is set. Will be used at the start of the exported filenames

depth_top_col
[str, default='TOP'] Name of column containing data for depth to top of described well intervals

depth_bot_col
[str, default='BOTTOM'] Name of column containing data for depth to bottom of described well intervals

log
[bool, default = True] Whether to log inputs and outputs to log file.

Returns

res_df or res
[geopandas.geodataframe] Geopandas geodataframe containing only important information needed for next stage of analysis.

w4h.**logger_function**(*logtocommence*, *parameters*, *func_name*)

Function to log other functions, to be called from within other functions

Parameters

logtocommence
[bool] Whether to perform logging steps

parameters
[dict] Dictionary containing parameters and their values, from function

func_name
[str] Name of function within which this is called

w4h.**merge_lithologies**(*well_data_df*, *targinterps_df*, *interp_col='INTERPRETATION'*, *target_col='TARGET'*, *target_class='bool'*)

Function to merge lithologies and target booleans based on classifications

Parameters

well_data_df
[pandas.DataFrame] Dataframe containing classified well data

targinterps_df
[pandas.DataFrame] Dataframe containing lithologies and their target interpretations, depending on what the target is for this analysis (often, coarse materials=1, fine=0)

target_col
[str, default = 'TARGET'] Name of column in targinterps_df containing the target interpretations

target_class, default = 'bool'
Whether the input column is using boolean values as its target indicator

Returns

df_targ
[pandas.DataFrame] Dataframe containing merged lithologies/targets

w4h.**merge_metadata**(*data_df*, *header_df*, *data_cols=None*, *header_cols=None*, *auto_pick_cols=False*, *drop_duplicate_cols=True*, *log=False*, *verbose=False*, ***kwargs*)

Function to merge tables, intended for merging metadata table with data table

Parameters

data_df
[pandas.DataFrame] "Left" dataframe, intended for this purpose to be dataframe with main data, but can be anything

header_df
[pandas.DataFrame] "Right" dataframe, intended for this purpose to be dataframe with metadata, but can be anything

data_cols
[list, optional] List of strings of column names, for columns to be included after join from "left" table (data table). If None, all columns are kept, by default None

header_cols
[list, optional] List of strings of columns names, for columns to be included in merged table after merge from "right" table (metadata). If None, all columns are kept, by default None

auto_pick_cols
[bool, default = False] Whether to autopick the columns from the metadata table. If True, the following column names are kept:['API_NUMBER', 'LATITUDE', 'LONGITUDE', 'BEDROCK_ELEV', 'SURFACE_ELEV', 'BEDROCK_DEPTH', 'LAYER_THICK'], by default False

drop_duplicate_cols
[bool, optional] If True, drops duplicate columns from the tables so that columns do not get renamed upon merge, by default True

log
[bool, default = False] Whether to log inputs and outputs to log file.

****kwargs**
kwargs that are passed directly to pd.merge(). By default, the 'on' and 'how' parameters are defined as on='API_NUMBER' and how='inner'

Returns

mergedTable
[pandas.DataFrame] Merged dataframe

w4h.**read_dict**(*file*, *keytype='np'*)

Function to read a text file with a dictionary in it into a python dictionary

Parameters

file
[str or pathlib.Path object] Filepath to the file of interest containing the dictionary text

keytype
[str, optional] String indicating the datatypes used in the text, currently only 'np' is implemented, by default 'np'

Returns

dict
Dictionary translated from text file.

w4h.**read_dictionary_terms**(*dict_file=None*, *id_col='ID'*, *search_col='DESCRIPTION'*, *definition_col='LITHOLOGY'*, *class_flag_col='CLASS_FLAG'*, *dictionary_type=None*, *class_flag=6*, *rem_extra_cols=True*, *verbose=False*, *log=False*)

Function to read dictionary terms from file into pandas dataframe

Parameters

dict_file
[str or pathlib.Path object, or list of these] File or list of files to be read

search_col
[str, default = 'DESCRIPTION'] Name of column containing search terms (geologic formations)

definition_col
[str, default = 'LITHOLOGY'] Name of column containing interpretations of search terms (lithologies)

dictionary_type
[str or None, {None, 'exact', 'start', 'wildcard',}]

Indicator of which kind of dictionary terms to be read in: None, 'exact', 'start', or 'wildcard' by default None.

- If None, uses name of file to try to determine. If it cannot, it will default to using the classification flag from class_flag
- If 'exact', will be used to search for exact matches to geologic descriptions
- If 'start', will be used as with the .startswith() string method to find inexact matches to geologic descriptions
- If 'wildcard', will be used to find any matching substring for inexact geologic matches

class_flag
[int, default = 1] Classification flag to be used if dictionary_type is None and cannot be otherwise determined, by default 1

rem_extra_cols
[bool, default = True] Whether to remove the extra columns from the input file after it is read in as a pandas dataframe, by default True

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

dict_terms
[pandas.DataFrame] Pandas dataframe with formatting ready to be used in the classification steps of this package

w4h.**read_grid**(*grid_path=None*, *grid_type='model'*, *no_data_val_grid=0*, *use_service=False*, *study_area=None*, *grid_crs=None*, *output_crs='EPSG:5070'*, *verbose=False*, *log=False*, ***kwargs*)

Reads in grid

Parameters

grid_path
[str or pathlib.Path, default=None] Path to a grid file

grid_type
[str, default='model'] Sets what type of grid to load in

no_data_val_grid
[int, default=0] Sets the no data value of the grid

use_service
[str, default=False] Sets which service the function uses

study_area
[geopandas.GeoDataFrame, default=None] Dataframe containing study area polygon

grid_crs
[str, default=None] Sets crs to use if clipping to study area

log
[bool, default = False] Whether to log results to log file, by default False

Returns

gridIN
[xarray.DataArray] Returns grid

w4h.**read_lithologies**(*lith_file=None*, *interp_col='LITHOLOGY'*, *target_col='CODE'*, *use_cols=None*, *verbose=False*, *log=False*)

Function to read lithology file into pandas dataframe

Parameters

lith_file
[str or pathlib.Path object, default = None] Filename of lithology file. If None, default is contained within repository, by default None

interp_col
[str, default = 'LITHOLOGY'] Column to used to match interpretations

target_col
[str, default = 'CODE'] Column to be used as target code

use_cols
[list, default = None] Which columns to use when reading in dataframe. If None, defaults to ['LITHOLOGY', 'CODE'].

log
[bool, default = True] Whether to log inputs and outputs to log file.

Returns

pandas.DataFrame
Pandas dataframe with lithology information

w4h.**read_model_grid**(*model_grid_path*, *study_area=None*, *no_data_val_grid=0*, *read_grid=True*, *node_byspace=True*, *grid_crs=None*, *output_crs='EPSG:5070'*, *verbose=False*, *log=False*)

Reads in model grid to xarray data array

Parameters

grid_path
[str] Path to model grid file

study_area
[geopandas.GeoDataFrame, default=None] Dataframe containing study area polygon

no_data_val_grid
[int, default=0] value assigned to areas with no data

readGrid
[bool, default=True] Whether function to either read grid or create grid

node_byspace
[bool, default=False] Denotes how to create grid

output_crs
[str, default='EPSG:5070'] Inputs study area crs

grid_crs
[str, default=None] Inputs grid crs

log
[bool, default = False] Whether to log results to log file, by default False

Returns

modelGrid
[xarray.DataArray] Data array containing model grid

w4h.**read_raw_csv**(*data_filepath*, *metadata_filepath*, *data_cols=None*, *metadata_cols=None*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *well_key='API_NUMBER'*, *encoding='latin-1'*, *verbose=False*, *log=False*, ***read_csv_kwargs*)

Easy function to read raw .txt files output from (for example), an Access database

Parameters

data_filepath
[str] Filename of the file containing data, including the extension.

metadata_filepath
[str] Filename of the file containing metadata, including the extension.

data_cols
[list, default = None] List with strings with names of columns from txt file to keep after reading. If None, ["API_NUMBER","TABLE_NAME","FORMATION","THICKNESS","TOP","BOTTOM"], by default None.

metadata_cols
[list, default = None] List with strings with names of columns from txt file to keep after reading. If None,

['API_NUMBER',"TOTAL_DEPTH","SECTION","TWP","TDIR","RNG","RDIR","MERIDIAN","QUARTERS
by default None

x_col
[str, default = 'LONGITUDE'] Name of column in metadata file indicating the x-location of the well, by default 'LONGITUDE'

ycol
[str, default = 'LATITUDE'] Name of the column in metadata file indicating the y-location of the well, by default 'LATITUDE'

well_key
[str, default = 'API_NUMBER'] Name of the column with the key/identifier that will be used to merge data later, by default 'API_NUMBER'

encoding
[str, default = 'latin-1'] Encoding of the data in the input files, by default 'latin-1'

verbose
[bool, default = False] Whether to print the number of rows in the input columns, by default False

log
[bool, default = False] Whether to log inputs and outputs to log file.

****read_csv_kwargs**
**kwargs that get passed to pd.read_csv()

Returns

(pandas.DataFrame, pandas.DataFrame/None)
Tuple/list with two pandas dataframes: (well_data, metadata) metadata is None if only well_data is used

w4h.**read_study_area**(*study_area=None*, *output_crs='EPSG:5070'*, *buffer=None*, *return_original=False*, *log=False*, *verbose=False*, ***read_file_kwargs*)

Read study area geospatial file into geopandas

Parameters

study_area
[str, pathlib.Path, geopandas.GeoDataFrame, or shapely.Geometry] Filepath to any geospatial file readable by geopandas. Polygon is best, but may work with other types if extent is correct.

study_area_crs
[str, tuple, dict, optional] CRS designation readable by geopandas/pyproj

buffer
[None or numeric, default=None] If None, no buffer created. If a numeric value is given (float or int, for example), a buffer will be created at that distance in the unit of the study_area_crs.

return_original
[bool, default=False] Whether to return the (reprojected) study area as well as the (reprojected) buffered study area. Study area is only used for clipping data, so usually return_original=False is sufficient.

log
[bool, default = False] Whether to log results to log file, by default False

verbose
[bool, default=False] Whether to print status and results to terminal

Returns

studyAreaIN
[geopandas dataframe] Geopandas dataframe with polygon geometry.

w4h.**read_wcs**(*study_area*,
wcs_url='https://data.isgs.illinois.edu/arcgis/services/Elevation/IL_Statewide_Lidar_DEM_WGS/ImageServer/WCSSe
res_x=30, *res_y=30*, *verbose=False*, *log=False*, ***kwargs*)

Reads a WebCoverageService from a url and returns a rioxarray dataset containing it.

Parameters

study_area
[geopandas.GeoDataFrame] Dataframe containing study area polygon

wcs_url
[str, default=lidarURL]

Represents the url for the WCS
res_x
[int, default=30] Sets resolution for x axis

res_y
[int, default=30] Sets resolution for y axis

log
[bool, default = False] Whether to log results to log file, by default False

****kwargs**

Returns

wcsData_rxr
[xarray.DataArray] A xarray dataarray holding the image from the WebCoverageService

w4h.**read_wms**(*study_area*, *layer_name='IL_Statewide_Lidar_DEM_WGS:None'*,
wms_url='https://data.isgs.illinois.edu/arcgis/services/Elevation/IL_Statewide_Lidar_DEM_WGS/ImageServer/WCSS
srs='EPSG:3857', *clip_to_studyarea=True*, *bbox=[-9889002.6155, 5134541.069716,*
-9737541.607038, 5239029.6274], *res_x=30*, *res_y=30*, *size_x=512*, *size_y=512*,
format='image/tiff', *verbose=False*, *log=False*, ***kwargs*)

Reads a WebMapService from a url and returns a rioxarray dataset containing it.

Parameters

study_area
[geopandas.GeoDataFrame] Dataframe containg study area polygon

layer_name
[str, default='IL_Statewide_Lidar_DEM_WGS:None'] Represents the layer name in the WMS

wms_url
[str, default=lidarURL] Represents the url for the WMS

srs
[str, default='EPSG:3857'] Sets the srs

clip_to_studyarea
[bool, default=True] Whether to clip to study area or not

res_x
[int, default=30] Sets resolution for x axis

res_y
[int, default=512] Sets resolution for y axis

size_x
[int, default=512] Sets width of result

size_y
[int, default=512] Sets height of result

log
[bool, default = False] Whether to log results to log file, by default False

Returns

wmsData_rxr
[xarray.DataArray] Holds the image from the WebMapService

w4h.**read_xyz**(*xyzpath*, *datatypes=None*, *verbose=False*, *log=False*)

Function to read file containing xyz data (elevation/location)

Parameters

xyzpath
[str or pathlib.Path] Filepath of the xyz file, including extension

datatypes
[dict, default = None] Dictionary containing the datatypes for the columns int he xyz file. If None, {'ID':np.uint32,'API_NUMBER':np.uint64,'LATITUDE':np.float64,'LONGITUDE':np.float64,'ELEV_FT' by default None

verbose
[bool, default = False] Whether to print the number of xyz records to the terminal, by default False

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

pandas.DataFrame
Pandas dataframe containing the elevation and location data

w4h.**remerge_data**(*classifieddf*, *searchdf*)

Function to merge newly-classified (or not) and previously classified data

Parameters

classifieddf
[pandas.DataFrame] Dataframe that had already been classified previously

searchdf
[pandas.DataFrame] Dataframe with new classifications

Returns

remergeDF
[pandas.DataFrame] Dataframe containing all the data, merged back together

w4h.**remove_bad_depth**(*df_with_depth*, *top_col='TOP'*, *bottom_col='BOTTOM'*, *depth_type='depth'*, *verbose=False*, *log=False*)

Function to remove all records in the dataframe with well interpretations where the depth information is bad (i.e., where the bottom of the record is neerer to the surface than the top)

Parameters

df_with_depth
[pandas.DataFrame] Pandas dataframe containing the well records and descriptions for each interval

top_col
[str, default='TOP'] The name of the column containing the depth or elevation for the top of the interval, by default 'TOP'

bottom_col
[str, default='BOTTOM'] The name of the column containing the depth or elevation for the bottom of each interval, by default 'BOTTOM'

depth_type
[str, {'depth', 'elevation'}] Whether the table is organized by depth or elevation. If depth, the top column will have smaller values than the bottom column. If elevation, the top column will have higher values than the bottom column, by default 'depth'

verbose
[bool, default = False] Whether to print results to the terminal, by default False

log
[bool, default = False] Whether to log results to log file, by default False

Returns

pandas.Dataframe
Pandas dataframe with the records remvoed where the top is indicatd to be below the bottom.

w4h.**remove_no_depth**(*df_with_depth*, *top_col='TOP'*, *bottom_col='BOTTOM'*, *no_data_val_table=''*, *verbose=False*, *log=False*)

Function to remove well intervals with no depth information

Parameters

df_with_depth
[pandas.DataFrame] Dataframe containing well descriptions

top_col
[str, optional] Name of column containing information on the top of the well intervals, by default 'TOP'

bottom_col
[str, optional] Name of column containing information on the bottom of the well intervals, by default 'BOTTOM'

no_data_val_table
[any, optional] No data value in the input data, used by this function to indicate that depth data is not there, to be replaced by np.nan, by default ''

verbose
[bool, optional] Whether to print results to console, by default False

log
[bool, default = False] Whether to log results to log file, by default False

Returns

df_with_depth
[pandas.DataFrame] Dataframe with depths dropped

w4h.**remove_no_description**(*df_with_descriptions*, *description_col='FORMATION'*, *no_data_val_table=''*, *verbose=False*, *log=False*)

Function that removes all records in the dataframe containing the well descriptions where no description is given.

Parameters

df_with_descriptions
[pandas.DataFrame] Pandas dataframe containing the well records with their individual descriptions

description_col
[str, optional] Name of the column containing the geologic description of each interval, by default 'FORMATION'

no_data_val_table
[str, optional] The value expected if the column is empty or there is no data. These will be replaced by np.nan before being removed, by default ''

verbose
[bool, optional] Whether to print the results of this step to the terminal, by default False

log
[bool, default = False] Whether to log results to log file, by default False

Returns

pandas.DataFrame
Pandas dataframe with records with no description removed.

w4h.**remove_no_topo**(*df_with_topo*, *zcol='ELEVATION'*, *no_data_val_table=''*, *verbose=False*, *log=False*)

Function to remove wells that do not have topography data (needed for layer selection later).

This function is intended to be run on the metadata table after elevations have attempted to been added.

Parameters

df_with_topo
[pandas.DataFrame] Pandas dataframe containing elevation information.

zcol
[str] Name of elevation column

no_data_val_table
[any] Value in dataset that indicates no data is present (replaced with np.nan)

verbose
[bool, optional] Whether to print outputs, by default True

log
[bool, default = False] Whether to log results to log file, by default False

Returns

pandas.DataFrame
Pandas dataframe with intervals with no topography removed.

w4h.**remove_nonlocated**(*df_with_locations*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *no_data_val_table=''*, *verbose=False*, *log=False*)

Function to remove wells and well intervals where there is no location information

Parameters

df_with_locations
[pandas.DataFrame] Pandas dataframe containing well descriptions

metadata_DF
[pandas.DataFrame] Pandas dataframe containing metadata, including well locations (e.g., Latitude/Longitude)

log
[bool, default = False] Whether to log results to log file, by default False

Returns

df_with_locations
[pandas.DataFrame] Pandas dataframe containing only data with location information

w4h.**run**(*well_data*, *surf_elev_grid*, *bedrock_elev_grid*, *model_grid=None*, *metadata=None*, *layers=9*, *well_data_cols=None*, *well_metadata_cols=None*, *description_col='FORMATION'*, *top_col='TOP'*, *bottom_col='BOTTOM'*, *depth_type='depth'*, *study_area=None*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *zcol='ELEVATION'*, *well_id_col='API_NUMBER'*, *lith_dict=None*, *lith_dict_start=None*, *lith_dict_wildcard=None*, *target_dict=None*, *target_name=''*, *export_dir=None*, *verbose=False*, *log=False*, ***kw_params*)

w4h.run() is a function that runs the intended workflow of the wells4hydrogeology (w4h) package. This means that it runs several constituent functions. The workflow that this follows is provided in the package wiki. It accepts the parameters of the constituent functions. To see a list of these functions and parameters, use *help(w4h.run)*.

The following functions used in w4h.run() are listed below, along with their parameters and default values for those parameters. See the documentation for the each of the individual functions for more information on a specific parameter:

file_setup

well_data | default = '<no default>'

metadata | default = None

data_filename | default = '*ISGS_DOWNHOLE_DATA*.txt'

metadata_filename | default = '*ISGS_HEADER*.txt'

log_dir | default = None

verbose | default = False

log | default = False

read_raw_csv

data_filepath | default = '<output of previous function>'

metadata_filepath | default = '<output of previous function>'

data_cols | default = None

metadata_cols | default = None

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

well_key | default = 'API_NUMBER'

encoding | default = 'latin-1'

verbose | default = False

log | default = False

read_csv_kwargs | default = {}

define_dtypes

undefined_df | default = '<output of previous function>'

datatypes | default = None

verbose | default = False

log | default = False

merge_metadata

data_df | default = '<output of previous function>'

header_df | default = '<output of previous function>'

data_cols | default = None

header_cols | default = None

auto_pick_cols | default = False

drop_duplicate_cols | default = True

log | default = False

verbose | default = False

kwargs | default = {}

coords2geometry

df_no_geometry | default = '<output of previous function>'

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

zcol | default = 'ELEV_FT'

input_coords_crs | default = 'EPSG:4269'

output_crs | default = 'EPSG:5070'

use_z | default = False

wkt_col | default = 'WKT'

geometry_source | default = 'coords'

verbose | default = False

log | default = False

read_study_area

study_area | default = None

output_crs | default = 'EPSG:5070'

buffer | default = None

return_original | default = False

log | default = False

verbose | default = False

read_file_kwargs | default = {}

clip_gdf2study_area

study_area | default = '<output of previous function>'

gdf | default = '<output of previous function>'

log | default = False

verbose | default = False

read_grid

grid_path | default = None

grid_type | default = 'model'

no_data_val_grid | default = 0

use_service | default = False

study_area | default = None

grid_crs | default = None

output_crs | default = 'EPSG:5070'

verbose | default = False

log | default = False

kwargs | default = {}

add_control_points

df_without_control | default = '<output of previous function>'

df_control | default = None

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

zcol | default = 'ELEV_FT'

controlpoints_crs | default = 'EPSG:4269'

output_crs | default = 'EPSG:5070'

description_col | default = 'FORMATION'

interp_col | default = 'INTERPRETATION'

target_col | default = 'TARGET'

verbose | default = False

log | default = False

kwargs | default = {}

remove_nonlocated

df_with_locations | default = '<output of previous function>'

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

no_data_val_table | default = ''

verbose | default = False

log | default = False

remove_no_topo

df_with_topo | default = '<output of previous function>'

zcol | default = 'ELEVATION'

no_data_val_table | default = ''

verbose | default = False

log | default = False

remove_no_depth

df_with_depth | default = '<output of previous function>'

top_col | default = 'TOP'

bottom_col | default = 'BOTTOM'

no_data_val_table | default = ''

verbose | default = False

log | default = False

remove_bad_depth

df_with_depth | default = '<output of previous function>'

top_col | default = 'TOP'

bottom_col | default = 'BOTTOM'

depth_type | default = 'depth'

verbose | default = False

log | default = False

remove_no_description

df_with_descriptions | default = '<output of previous function>'

description_col | default = 'FORMATION'

no_data_val_table | default = ''

verbose | default = False

log | default = False

get_search_terms

spec_path | default = '/home/docs/checkouts/readthedocs.org/user_builds/wells4hydrogeology/checkouts/latest/docs/resource

spec_glob_pattern | default = '*SearchTerms-Specific*'

start_path | default = None

start_glob_pattern | default = '*SearchTerms-Start*'

wildcard_path | default = None

wildcard_glob_pattern | default = '*SearchTerms-Wildcard'

verbose | default = False

log | default = False

read_dictionary_terms

dict_file | default = None

id_col | default = 'ID'

search_col | default = 'DESCRIPTION'

definition_col | default = 'LITHOLOGY'

class_flag_col | default = 'CLASS_FLAG'

dictionary_type | default = None

class_flag | default = 6

rem_extra_cols | default = True

verbose | default = False

log | default = False

specific_define

df | default = '<output of previous function>'

terms_df | default = '<output of previous function>'

description_col | default = 'FORMATION'

terms_col | default = 'DESCRIPTION'

verbose | default = False

log | default = False

split_defined

df | default = '<output of previous function>'

classification_col | default = 'CLASS_FLAG'

verbose | default = False

log | default = False

start_define

df | default = '<output of previous function>'

terms_df | default = '<output of previous function>'

description_col | default = 'FORMATION'

terms_col | default = 'DESCRIPTION'

verbose | default = False

log | default = False

wildcard_define

df | default = '<output of previous function>'

terms_df | default = '<output of previous function>'

description_col | default = 'FORMATION'

terms_col | default = 'DESCRIPTION'

verbose | default = False

log | default = False

remerge_data

classifieddf | default = '<output of previous function>'

searchdf | default = '<output of previous function>'

fill_unclassified

df | default = '<output of previous function>'

classification_col | default = 'CLASS_FLAG'

read_lithologies

lith_file | default = None

interp_col | default = 'LITHOLOGY'

target_col | default = 'CODE'

use_cols | default = None

verbose | default = False

log | default = False

merge_lithologies

well_data_df | default = '<output of previous function>'

targinterps_df | default = '<output of previous function>'

interp_col | default = 'INTERPRETATION'

target_col | default = 'TARGET'

target_class | default = 'bool'

align_rasters

grids_unaligned | default = None

model_grid | default = None

no_data_val_grid | default = 0

verbose | default = False

log | default = False

get_drift_thick

surface_elev | default = None

bedrock_elev | default = None

layers | default = 9

plot | default = False

verbose | default = False

log | default = False

sample_raster_points

raster | default = None

points_df | default = None

well_id_col | default = 'API_NUMBER'

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

new_col | default = 'SAMPLED'

verbose | default = False

log | default = False

get_layer_depths

df_with_depths | default = '<output of previous function>'

surface_elev_col | default = 'SURFACE_ELEV'

layer_thick_col | default = 'LAYER_THICK'

layers | default = 9

log | default = False

layer_target_thick

df | default = '<output of previous function>'

layers | default = 9

return_all | default = False

export_dir | default = None

outfile_prefix | default = None

depth_top_col | default = 'TOP'

depth_bot_col | default = 'BOTTOM'

log | default = False

layer_interp

points | default = '<no default>'

grid | default = '<no default>'

layers | default = None

interp_kind | default = 'nearest'

return_type | default = 'dataarray'

export_dir | default = None

target_col | default = 'TARG_THICK_PER'

layer_col | default = 'LAYER'

xcol | default = None

ycol | default = None

xcoord | default = 'x'

ycoord | default = 'y'

log | default = False

verbose | default = False

kwargs | default = {}

export_grids

grid_data | default = '<no default>'

out_path | default = '<no default>'

file_id | default = ''

filetype | default = 'tif'

variable_sep | default = True

date_stamp | default = True

verbose | default = False

log | default = False"

w4h.**sample_raster_points**(*raster=None*, *points_df=None*, *well_id_col='API_NUMBER'*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *new_col='SAMPLED'*, *verbose=False*, *log=False*)

Sample raster values to points from geopandas geodataframe.

Parameters

raster
[rioxarray data array] Raster containing values to be sampled.

points_df
[geopandas.geodataframe] Geopandas dataframe with geometry column containing point values to sample.

well_id_col
[str, default="API_NUMBER"] Column that uniquely identifies each well so multiple sampling points are not taken per well

xcol
[str, default='LONGITUDE'] Column containing name for x-column, by default 'LONGITUDE.' This is used to output (potentially) reprojected point coordinates so as not to overwrite the original.

ycol
[str, default='LATITUDE'] Column containing name for y-column, by default 'LATITUDE.' This is used to output (potentially) reprojected point coordinates so as not to overwrite the original. new_col : str, optional

new_col
[str, default='SAMPLED'] Name for name of new column containing points sampled from the raster, by default 'SAMPLED'.

verbose
[bool, default=True] Whether to send to print() information about progress of function, by default True.

log
[bool, default = False] Whether to log results to log file, by default False

Returns

points_df
[geopandas.geodataframe] Same as points_df, but with sampled values and potentially with reprojected coordinates.

w4h.**sort_dataframe**(*df*, *sort_cols=['API_NUMBER', 'TOP']*, *remove_nans=True*)

Function to sort dataframe by one or more columns.

Parameters

df
[pandas.DataFrame] Dataframe to be sorted

sort_cols
[str or list of str, default = ['API_NUMBER','TOP']] Name(s) of columns by which to sort dataframe, by default ['API_NUMBER','TOP']

remove_nans
[bool, default = True] Whether or not to remove nans in the process, by default True

Returns

df_sorted
[pandas.DataFrame] Sorted dataframe

w4h.**specific_define**(*df*, *terms_df*, *description_col='FORMATION'*, *terms_col='DESCRIPTION'*, *verbose=False*, *log=False*)

Function to classify terms that have been specifically defined in the terms_df.

Parameters

df
[pandas.DataFrame] Input dataframe with unclassified well descriptions.

terms_df
[pandas.DataFrame] Dataframe containing the classifications

description_col
[str, default='FORMATION'] Column name in df containing the well descriptions, by default 'FORMATION'.

terms_col
[str, default='DESCRIPTION'] Column name in terms_df containing the classified descriptions, by default 'DESCRIPTION'.

verbose
[bool, default=False] Whether to print up results, by default False.

Returns

df_Interps
[pandas.DataFrame] Dataframe containing the well descriptions and their matched classifications.

w4h.**split_defined**(*df*, *classification_col='CLASS_FLAG'*, *verbose=False*, *log=False*)

Function to split dataframe with well descriptions into two dataframes based on whether a row has been classified.

Parameters

df
[pandas.DataFrame] Dataframe containing all the well descriptions

classification_col
[str, default = 'CLASS_FLAG'] Name of column containing the classification flag, by default 'CLASS_FLAG'

verbose
[bool, default = False] Whether to print results, by default False

log
[bool, default = False] Whether to log results to log file

Returns

Two-item tuple of pandas.Dataframe
tuple[0] is dataframe containing classified data, tuple[1] is dataframe containing unclassified data.

w4h.**start_define**(*df*, *terms_df*, *description_col='FORMATION'*, *terms_col='DESCRIPTION'*, *verbose=False*, *log=False*)

Function to classify descriptions according to starting substring.

Parameters

df
[pandas.DataFrame] Dataframe containing all the well descriptions

terms_df
[pandas.DataFrame] Dataframe containing all the startswith substrings to use for searching

description_col
[str, default = 'FORMATION'] Name of column in df containing descriptions, by default 'FORMATION'

terms_col
[str, default = 'FORMATION'] Name of column in terms_df containing startswith substring to match with description_col, by default 'FORMATION'

verbose
[bool, default = False] Whether to print out results, by default False

log
[bool, default = True] Whether to log results to log file

Returns

df
[pandas.DataFrame] Dataframe containing the original data and new classifications

w4h.**verbose_print**(*func*, *local_variables*, *exclude_params=[]*)

w4h.**wildcard_define**(*df*, *terms_df*, *description_col='FORMATION'*, *terms_col='DESCRIPTION'*, *verbose=False*, *log=False*)

Function to classify descriptions according to any substring.

Parameters

df
[pandas.DataFrame] Dataframe containing all the well descriptions

terms_df
[pandas.DataFrame] Dataframe containing all the startswith substrings to use for searching

description_col
[str, default = 'FORMATION'] Name of column in df containing descriptions, by default 'FORMATION'

terms_col
[str, default = 'FORMATION'] Name of column in terms_df containing startswith substring to match with description_col, by default 'FORMATION'

verbose
[bool, default = False] Whether to print out results, by default False

log
[bool, default = True] Whether to log results to log file

Returns

df
[pandas.DataFrame] Dataframe containing the original data and new classifications

w4h.**xyz_metadata_merge**(*xyz*, *metadata*, *verbose=False*, *log=False*)

Add elevation to header data file.

Parameters

xyz
[pandas.Dataframe] Contains elevation for the points

metadata
[pandas dataframe] Header data file

log
[bool, default = False] Whether to log results to log file, by default False

Returns

headerXYZData
[pandas.Dataframe] Header dataset merged to get elevation values

1.2 Submodules

1.2.1 w4h.classify module

The Classify module contains functions for defining geological intervals into a preset subset of geologic interpretations.

w4h.classify.**depth_define**(*df*, *top_col='TOP'*, *thresh=550.0*, *verbose=False*, *log=False*)

Function to define all intervals lower than thresh as bedrock

Parameters

df
[pandas.DataFrame] Dataframe to classify

top_col
[str, default = 'TOP'] Name of column that contains the depth information, likely of the top of the well interval, by default 'TOP'

thresh
[float, default = 550.0] Depth (in units used in df['top_col']) below which all intervals will be classified as bedrock, by default 550.0.

verbose
[bool, default = False] Whether to print results, by default False

log
[bool, default = True] Whether to log results to log file

Returns

df
[pandas.DataFrame] Dataframe containing intervals classified as bedrock due to depth

w4h.classify.**export_undefined**(*df*, *outdir*)

Function to export terms that still need to be defined.

Parameters

df
[pandas.DataFrame] Dataframe containing at least some unclassified data

outdir
[str or pathlib.Path] Directory to save file. Filename will be generated automatically based on today's date.

Returns

stillNeededDF
[pandas.DataFrame] Dataframe containing only unclassified terms, and the number of times they occur

w4h.classify.**fill_unclassified**(*df*, *classification_col='CLASS_FLAG'*)

Fills unclassified rows in 'CLASS_FLAG' column with np.nan

Parameters

df
[pandas.DataFrame] Dataframe on which to perform operation

Returns

df
[pandas.DataFrame] Dataframe on which operation has been performed

w4h.classify.**get_unique_wells**(*df*, *wellid_col='API_NUMBER'*, *verbose=False*, *log=False*)

Gets unique wells as a dataframe based on a given column name.

Parameters

df
[pandas.DataFrame] Dataframe containing all wells and/or well intervals of interest

wellid_col
[str, default="API_NUMBER"] Name of column in df containing a unique identifier for each well, by default 'API_NUMBER'. .unique() will be run on this column to get the unique values.

log
[bool, default = False] Whether to log results to log file

Returns

wellsDF
DataFrame containing only the unique well IDs

w4h.classify.**merge_lithologies**(*well_data_df*, *targinterps_df*, *interp_col='INTERPRETATION'*, *target_col='TARGET'*, *target_class='bool'*)

Function to merge lithologies and target booleans based on classifications

Parameters

well_data_df
[pandas.DataFrame] Dataframe containing classified well data

targinterps_df
[pandas.DataFrame] Dataframe containing lithologies and their target interpretations, depending on what the target is for this analysis (often, coarse materials=1, fine=0)

target_col
[str, default = 'TARGET'] Name of column in targinterps_df containing the target interpretations

target_class, default = 'bool'
Whether the input column is using boolean values as its target indicator

Returns

df_targ
[pandas.DataFrame] Dataframe containing merged lithologies/targets

w4h.classify.**remerge_data**(*classifieddf*, *searchdf*)

Function to merge newly-classified (or not) and previously classified data

Parameters

classifieddf
[pandas.DataFrame] Dataframe that had already been classified previously

searchdf
[pandas.DataFrame] Dataframe with new classifications

Returns

remergeDF
[pandas.DataFrame] Dataframe containing all the data, merged back together

w4h.classify.**sort_dataframe**(*df*, *sort_cols=['API_NUMBER', 'TOP']*, *remove_nans=True*)

Function to sort dataframe by one or more columns.

Parameters

df
[pandas.DataFrame] Dataframe to be sorted

sort_cols
[str or list of str, default = ['API_NUMBER','TOP']] Name(s) of columns by which to sort dataframe, by default ['API_NUMBER','TOP']

remove_nans
[bool, default = True] Whether or not to remove nans in the process, by default True

Returns

df_sorted
[pandas.DataFrame] Sorted dataframe

w4h.classify.**specific_define**(*df*, *terms_df*, *description_col='FORMATION'*, *terms_col='DESCRIPTION'*, *verbose=False*, *log=False*)

Function to classify terms that have been specifically defined in the terms_df.

Parameters

df
[pandas.DataFrame] Input dataframe with unclassified well descriptions.

terms_df
[pandas.DataFrame] Dataframe containing the classifications

description_col
[str, default='FORMATION'] Column name in df containing the well descriptions, by default 'FORMATION'.

terms_col
[str, default='DESCRIPTION'] Column name in terms_df containing the classified descriptions, by default 'DESCRIPTION'.

verbose
[bool, default=False] Whether to print up results, by default False.

Returns

df_Interps
[pandas.DataFrame] Dataframe containing the well descriptions and their matched classifications.

w4h.classify.**split_defined**(*df*, *classification_col='CLASS_FLAG'*, *verbose=False*, *log=False*)

Function to split dataframe with well descriptions into two dataframes based on whether a row has been classified.

Parameters

df
[pandas.DataFrame] Dataframe containing all the well descriptions

classification_col
[str, default = 'CLASS_FLAG'] Name of column containing the classification flag, by default 'CLASS_FLAG'

verbose
[bool, default = False] Whether to print results, by default False

log
[bool, default = False] Whether to log results to log file

Returns

Two-item tuple of pandas.Dataframe
tuple[0] is dataframe containing classified data, tuple[1] is dataframe containing unclassified data.

w4h.classify.**start_define**(*df*, *terms_df*, *description_col='FORMATION'*, *terms_col='DESCRIPTION'*, *verbose=False*, *log=False*)

Function to classify descriptions according to starting substring.

Parameters

df
[pandas.DataFrame] Dataframe containing all the well descriptions

terms_df
[pandas.DataFrame] Dataframe containing all the startswith substrings to use for searching

description_col
[str, default = 'FORMATION'] Name of column in df containing descriptions, by default 'FORMATION'

terms_col
[str, default = 'FORMATION'] Name of column in terms_df containing startswith substring to match with description_col, by default 'FORMATION'

verbose
[bool, default = False] Whether to print out results, by default False

log
[bool, default = True] Whether to log results to log file

Returns

df
[pandas.DataFrame] Dataframe containing the original data and new classifications

w4h.classify.**wildcard_define**(*df*, *terms_df*, *description_col='FORMATION'*, *terms_col='DESCRIPTION'*, *verbose=False*, *log=False*)

Function to classify descriptions according to any substring.

Parameters

df
[pandas.DataFrame] Dataframe containing all the well descriptions

terms_df
[pandas.DataFrame] Dataframe containing all the startswith substrings to use for searching

description_col
[str, default = 'FORMATION'] Name of column in df containing descriptions, by default 'FORMATION'

terms_col
[str, default = 'FORMATION'] Name of column in terms_df containing startswith substring to match with description_col, by default 'FORMATION'

verbose
[bool, default = False] Whether to print out results, by default False

log
[bool, default = True] Whether to log results to log file

Returns

df
[pandas.DataFrame] Dataframe containing the original data and new classifications

1.2.2 w4h.clean module

The Clean module contains functions for cleaning the data (i.e., removing data not to be used in further analysis)

w4h.clean.**remove_bad_depth**(*df_with_depth*, *top_col='TOP'*, *bottom_col='BOTTOM'*, *depth_type='depth'*, *verbose=False*, *log=False*)

Function to remove all records in the dataframe with well interpretations where the depth information is bad (i.e., where the bottom of the record is neerer to the surface than the top)

Parameters

df_with_depth
[pandas.DataFrame] Pandas dataframe containing the well records and descriptions for each interval

top_col
[str, default='TOP'] The name of the column containing the depth or elevation for the top of the interval, by default 'TOP'

bottom_col
[str, default='BOTTOM'] The name of the column containing the depth or elevation for the bottom of each interval, by default 'BOTTOM'

depth_type
[str, {'depth', 'elevation'}] Whether the table is organized by depth or elevation. If depth, the top column will have smaller values than the bottom column. If elevation, the top column will have higher values than the bottom column, by default 'depth'

verbose
[bool, default = False] Whether to print results to the terminal, by default False

log
[bool, default = False] Whether to log results to log file, by default False

Returns

pandas.Dataframe
Pandas dataframe with the records remvoed where the top is indicatd to be below the bottom.

w4h.clean.**remove_no_depth**(*df_with_depth*, *top_col='TOP'*, *bottom_col='BOTTOM'*, *no_data_val_table=''*, *verbose=False*, *log=False*)

Function to remove well intervals with no depth information

Parameters

df_with_depth
[pandas.DataFrame] Dataframe containing well descriptions

top_col
[str, optional] Name of column containing information on the top of the well intervals, by default 'TOP'

bottom_col
[str, optional] Name of column containing information on the bottom of the well intervals, by default 'BOTTOM'

no_data_val_table
[any, optional] No data value in the input data, used by this function to indicate that depth data is not there, to be replaced by np.nan, by default ''

verbose
[bool, optional] Whether to print results to console, by default False

log
[bool, default = False] Whether to log results to log file, by default False

Returns

df_with_depth
[pandas.DataFrame] Dataframe with depths dropped

w4h.clean.**remove_no_description**(*df_with_descriptions*, *description_col='FORMATION'*, *no_data_val_table=''*, *verbose=False*, *log=False*)

Function that removes all records in the dataframe containing the well descriptions where no description is given.

Parameters

df_with_descriptions
[pandas.DataFrame] Pandas dataframe containing the well records with their individual descriptions

description_col
[str, optional] Name of the column containing the geologic description of each interval, by default 'FORMATION'

no_data_val_table
[str, optional] The value expected if the column is empty or there is no data. These will be replaced by np.nan before being removed, by default ‘’

verbose
[bool, optional] Whether to print the results of this step to the terminal, by default False

log
[bool, default = False] Whether to log results to log file, by default False

Returns

pandas.DataFrame
Pandas dataframe with records with no description removed.

w4h.clean.**remove_no_topo**(*df_with_topo*, *zcol='ELEVATION'*, *no_data_val_table=''*, *verbose=False*, *log=False*)

Function to remove wells that do not have topography data (needed for layer selection later).

This function is intended to be run on the metadata table after elevations have attempted to been added.

Parameters

df_with_topo
[pandas.DataFrame] Pandas dataframe containing elevation information.

zcol
[str] Name of elevation column

no_data_val_table
[any] Value in dataset that indicates no data is present (replaced with np.nan)

verbose
[bool, optional] Whether to print outputs, by default True

log
[bool, default = False] Whether to log results to log file, by default False

Returns

pandas.DataFrame
Pandas dataframe with intervals with no topography removed.

w4h.clean.**remove_nonlocated**(*df_with_locations*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *no_data_val_table=''*, *verbose=False*, *log=False*)

Function to remove wells and well intervals where there is no location information

Parameters

df_with_locations
[pandas.DataFrame] Pandas dataframe containing well descriptions

metadata_DF
[pandas.DataFrame] Pandas dataframe containing metadata, including well locations (e.g., Latitude/Longitude)

log
[bool, default = False] Whether to log results to log file, by default False

Returns

df_with_locations
[pandas.DataFrame] Pandas dataframe containing only data with location information

1.2.3 w4h.core module

The Core module contains core functions of the package used in other modules or as primary functions in the package. This includes the main run() function that allows rapid data analysis, a function to retrieve sample data, and functions that are used throughout the package for logging and printing verbose outputs.

w4h.core.**get_resources**(*resource_type='filepaths'*, *scope='local'*, *verbose=False*)

Function to get filepaths for resources included with package

Parameters

resource_type
[str, {'filepaths', 'data'}] If filepaths, will return dictionary with filepaths to sample data. If data, returns dictionary with data objects.

scope
[str, {'local', 'statewide'}] If 'local', will read in sample data for a local (around county sized) project. If 'state', will read in sample data for a statewide project (Illinois)

verbose
[bool, optional] Whether to print results to terminal, by default False

Returns

resources_dict
[dict] Dictionary containing key, value pairs with filepaths to resources that may be of interest.

w4h.core.**logger_function**(*logtocommence*, *parameters*, *func_name*)

Function to log other functions, to be called from within other functions

Parameters

logtocommence
[bool] Whether to perform logging steps

parameters
[dict] Dictionary containing parameters and their values, from function

func_name
[str] Name of function within which this is called

w4h.core.**run**(*well_data*, *surf_elev_grid*, *bedrock_elev_grid*, *model_grid=None*, *metadata=None*, *layers=9*, *well_data_cols=None*, *well_metadata_cols=None*, *description_col='FORMATION'*, *top_col='TOP'*, *bottom_col='BOTTOM'*, *depth_type='depth'*, *study_area=None*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *zcol='ELEVATION'*, *well_id_col='API_NUMBER'*, *lith_dict=None*, *lith_dict_start=None*, *lith_dict_wildcard=None*, *target_dict=None*, *target_name=''*, *export_dir=None*, *verbose=False*, *log=False*, ***kw_params*)

w4h.run() is a function that runs the intended workflow of the wells4hydrogeology (w4h) package. This means that it runs several constituent functions. The workflow that this follows is provided in the package wiki. It accepts the parameters of the constituent functions. To see a list of these functions and parameters, use *help(w4h.run)*.

The following functions used in w4h.run() are listed below, along with their parameters and default values for those parameters. See the documentation for the each of the individual functions for more information on a specific parameter:

file_setup

well_data | default = '<no default>'

metadata | default = None

data_filename | default = '*ISGS_DOWNHOLE_DATA*.txt'

metadata_filename | default = '*ISGS_HEADER*.txt'

log_dir | default = None

verbose | default = False

log | default = False

read_raw_csv

data_filepath | default = '<output of previous function>'

metadata_filepath | default = '<output of previous function>'

data_cols | default = None

metadata_cols | default = None

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

well_key | default = 'API_NUMBER'

encoding | default = 'latin-1'

verbose | default = False

log | default = False

read_csv_kwargs | default = {}

define_dtypes

undefined_df | default = '<output of previous function>'

datatypes | default = None

verbose | default = False

log | default = False

merge_metadata

data_df | default = '<output of previous function>'

header_df | default = '<output of previous function>'

data_cols | default = None

header_cols | default = None

auto_pick_cols | default = False

drop_duplicate_cols | default = True

log | default = False

verbose | default = False

kwargs | default = {}

coords2geometry

df_no_geometry | default = '<output of previous function>'

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

zcol | default = 'ELEV_FT'

input_coords_crs | default = 'EPSG:4269'

output_crs | default = 'EPSG:5070'

use_z | default = False

wkt_col | default = 'WKT'

geometry_source | default = 'coords'

verbose | default = False

log | default = False

read_study_area

study_area | default = None

output_crs | default = 'EPSG:5070'

buffer | default = None

return_original | default = False

log | default = False

verbose | default = False

read_file_kwargs | default = {}

clip_gdf2study_area

study_area | default = '<output of previous function>'

gdf | default = '<output of previous function>'

log | default = False

verbose | default = False

read_grid

grid_path | default = None

grid_type | default = 'model'

no_data_val_grid | default = 0

use_service | default = False

study_area | default = None

grid_crs | default = None

output_crs | default = 'EPSG:5070'

verbose | default = False

log | default = False

kwargs | default = {}

add_control_points

df_without_control | default = '<output of previous function>'

df_control | default = None

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

zcol | default = 'ELEV_FT'

controlpoints_crs | default = 'EPSG:4269'

output_crs | default = 'EPSG:5070'

description_col | default = 'FORMATION'

interp_col | default = 'INTERPRETATION'

target_col | default = 'TARGET'

verbose | default = False

log | default = False

kwargs | default = {}

remove_nonlocated

df_with_locations | default = '<output of previous function>'

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

no_data_val_table | default = ''

verbose | default = False

log | default = False

remove_no_topo

df_with_topo | default = '<output of previous function>'

zcol | default = 'ELEVATION'

no_data_val_table | default = ''

verbose | default = False

log | default = False

remove_no_depth

df_with_depth | default = '<output of previous function>'

top_col | default = 'TOP'

bottom_col | default = 'BOTTOM'

no_data_val_table | default = ''

verbose | default = False

log | default = False

remove_bad_depth

df_with_depth | default = '<output of previous function>'

top_col | default = 'TOP'

bottom_col | default = 'BOTTOM'

depth_type | default = 'depth'

verbose | default = False

log | default = False

remove_no_description

df_with_descriptions | default = '<output of previous function>'

description_col | default = 'FORMATION'

no_data_val_table | default = ''

verbose | default = False

log | default = False

get_search_terms

spec_path | default = '/home/docs/checkouts/readthedocs.org/user_builds/wells4hydrogeology/checkouts/latest/docs/resource

spec_glob_pattern | default = '*SearchTerms-Specific*'

start_path | default = None

start_glob_pattern | default = '*SearchTerms-Start*'

wildcard_path | default = None

wildcard_glob_pattern | default = '*SearchTerms-Wildcard'

verbose | default = False

log | default = False

read_dictionary_terms

dict_file | default = None

id_col | default = 'ID'

search_col | default = 'DESCRIPTION'

definition_col | default = 'LITHOLOGY'

class_flag_col | default = 'CLASS_FLAG'

dictionary_type | default = None

class_flag | default = 6

rem_extra_cols | default = True

verbose | default = False

log | default = False

specific_define

df | default = '<output of previous function>'

terms_df | default = '<output of previous function>'

description_col | default = 'FORMATION'

terms_col | default = 'DESCRIPTION'

verbose | default = False

log | default = False

split_defined

df | default = '<output of previous function>'

classification_col | default = 'CLASS_FLAG'

verbose | default = False

log | default = False

start_define

df | default = '<output of previous function>'

terms_df | default = '<output of previous function>'

description_col | default = 'FORMATION'

terms_col | default = 'DESCRIPTION'

verbose | default = False

log | default = False

wildcard_define

df | default = '<output of previous function>'

terms_df | default = '<output of previous function>'

description_col | default = 'FORMATION'

terms_col | default = 'DESCRIPTION'

verbose | default = False

log | default = False

remerge_data

classifieddf | default = '<output of previous function>'

searchdf | default = '<output of previous function>'

fill_unclassified

df | default = '<output of previous function>'

classification_col | default = 'CLASS_FLAG'

read_lithologies

lith_file | default = None

interp_col | default = 'LITHOLOGY'

target_col | default = 'CODE'

use_cols | default = None

verbose | default = False

log | default = False

merge_lithologies

well_data_df | default = '<output of previous function>'

targinterps_df | default = '<output of previous function>'

interp_col | default = 'INTERPRETATION'

target_col | default = 'TARGET'

target_class | default = 'bool'

align_rasters

grids_unaligned | default = None

model_grid | default = None

no_data_val_grid | default = 0

verbose | default = False

log | default = False

get_drift_thick

surface_elev | default = None

bedrock_elev | default = None

layers | default = 9

plot | default = False

verbose | default = False

log | default = False

sample_raster_points

raster | default = None

points_df | default = None

well_id_col | default = 'API_NUMBER'

xcol | default = 'LONGITUDE'

ycol | default = 'LATITUDE'

new_col | default = 'SAMPLED'

verbose | default = False

log | default = False

get_layer_depths

df_with_depths | default = '<output of previous function>'

surface_elev_col | default = 'SURFACE_ELEV'

layer_thick_col | default = 'LAYER_THICK'

layers | default = 9

log | default = False

layer_target_thick

df | default = '<output of previous function>'

layers | default = 9

return_all | default = False

export_dir | default = None

outfile_prefix | default = None

depth_top_col | default = 'TOP'

depth_bot_col | default = 'BOTTOM'

log | default = False

layer_interp

points | default = '<no default>'

grid | default = '<no default>'

layers | default = None

interp_kind | default = 'nearest'

return_type | default = 'dataarray'

export_dir | default = None

target_col | default = 'TARG_THICK_PER'

layer_col | default = 'LAYER'

xcol | default = None

ycol | default = None

xcoord | default = 'x'

ycoord | default = 'y'

log | default = False

verbose | default = False

kwargs | default = {}

export_grids

grid_data | default = '<no default>'

out_path | default = '<no default>'

file_id | default = ''

filetype | default = 'tif'

variable_sep | default = True

date_stamp | default = True

verbose | default = False

log | default = False"

w4h.core.**verbose_print**(*func*, *local_variables*, *exclude_params=[]*)

1.2.4 w4h.export module

The Export module contains functions for exporting processed data.

w4h.export.**export_dataframe**(*df*, *out_dir*, *filename*, *date_stamp=True*, *log=False*)

Function to export dataframes

Parameters

df
[pandas dataframe, or list of pandas dataframes] Data frame or list of dataframes to be exported

out_dir
[string or pathlib.Path object] Directory to which to export dataframe object(s) as .csv

filename
[str or list of strings] Filename(s) of output files

date_stamp
[bool, default=True] Whether to include a datestamp in the filename. If true, file ends with _yyyy-mm-dd.csv of current date, by default True.

log
[bool, default = True] Whether to log inputs and outputs to log file.

w4h.export.**export_grids**(*grid_data*, *out_path*, *file_id=''*, *filetype='tif'*, *variable_sep=True*, *date_stamp=True*, *verbose=False*, *log=False*)

Function to export grids to files.

Parameters

grid_data
[xarray DataArray or xarray Dataset] Dataset or dataarray to be exported

out_path
[str or pathlib.Path object] Output location for data export. If variable_sep=True, this should be a directory. Otherwise, this should also include the filename. The file extension should not be included here.

file_id
[str, optional] If specified, will add this after 'LayerXX' or 'AllLayers' in the filename, just before datestamp, if used. Example filename for file_id='Coarse': Layer1_Coarse_2023-04-18.tif.

filetype
[str, optional] Output filetype. Can either be pickle or any file extension supported by rioxarray.rio.to_raster(). Can either include period or not., by default 'tif'

variable_sep
[bool, optional] If grid_data is an xarray Dataset, this will export each variable in the dataset as a separate file, including the variable name in the filename, by default False

date_stamp
[bool, optional] Whether to include a date stamp in the file name., by default True

log
[bool, default = True] Whether to log inputs and outputs to log file.

1.2.5 w4h.layers module

The Layers module contains functions for splitting data into a layered model and for interpolating data within the layers

w4h.layers.**combine_dataset**(*layer_dataset*, *surface_elev*, *bedrock_elev*, *layer_thick*, *log=False*)

Function to combine xarray datasets or datarrays into a single xr.Dataset. Useful to add surface, bedrock, layer thick, and layer datasets all into one variable, for pickling, for example.

Parameters

layer_dataset
[xr.DataArray] DataArray contining all the interpolated layer information.

surface_elev
[xr.DataArray] DataArray containing surface elevation data

bedrock_elev
[xr.DataArray] DataArray containing bedrock elevation data

layer_thick
[xr.DataArray] DataArray containing the layer thickness at each point in the model grid

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

xr.Dataset
Dataset with all input arrays set to different variables within the dataset.

w4h.layers.**get_layer_depths**(*df_with_depths*, *surface_elev_col='SURFACE_ELEV'*, *layer_thick_col='LAYER_THICK'*, *layers=9*, *log=False*)

Function to calculate depths and elevations of each model layer at each well based on surface elevation, bedrock elevation, and number of layers/layer thickness

Parameters

df_with_depths
[pandas.DataFrame] Dataframe containing well metdata

layers
[int, default=9] Number of layers. This should correlate with get_drift_thick() input parameter, if drift thickness was calculated using that function, by default 9.

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

pandas.Dataframe
Dataframe containing new columns for depth to layers and elevation of layers.

w4h.layers.**layer_interp**(*points*, *grid*, *layers=None*, *interp_kind='nearest'*, *return_type='dataarray'*, *export_dir=None*, *target_col='TARG_THICK_PER'*, *layer_col='LAYER'*, *xcol=None*, *ycol=None*, *xcoord='x'*, *ycoord='y'*, *log=False*, *verbose=False*, ***kwargs*)

Function to interpolate results, going from points to grid data. Uses scipy.interpolate module.

Parameters

points
[list] List containing pandas dataframes or geopandas geoadataframes containing the point data. Should be resDF_list output from layer_target_thick().

grid
[xr.DataArray or xr.Dataset] Xarray DataArray or DataSet with the coordinates/spatial reference of the output grid to interpolate to

layers
[int, default=None] Number of layers for interpolation. If None, uses the length ofthe points list to determine number of layers. By default None.

interp_kind
[str, {'nearest', 'interp2d','linear', 'cloughtocher', 'radial basis function'}] Type of interpolation to use. See scipy.interpolate N-D scattered. Values can be any of the following (also shown in "kind" column of N-D scattered section of table here: https://docs.scipy.org/doc/scipy/tutorial/interpolate.html). By default 'nearest'

return_type
[str, {'dataarray', 'dataset'}] Type of xarray object to return, either xr.DataArray or xr.Dataset, by default 'dataarray.'

export_dir
[str or pathlib.Path, default=None] Export directory for interpolated grids, using w4h.export_grids(). If None, does not export, by default None.

target_col
[str, default = 'TARG_THICK_PER'] Name of column in points containing data to be interpolated, by default 'TARG_THICK_PER'.

layer_col
[str, default = 'Layer'] Name of column containing layer number. Not currently used, by default 'LAYER'

xcol
[str, default = 'None'] Name of column containing x coordinates. If None, will look for 'geometry' column, as in a geopandas.GeoDataframe. By default None

ycol
[str, default = 'None'] Name of column containing y coordinates. If None, will look for 'geometry' column, as in a geopandas.GeoDataframe. By default None

xcoord
[str, default='x'] Name of x coordinate in grid, used to extract x values of grid, by default 'x'

ycoord
[str, default='y'] Name of y coordinate in grid, used to extract x values of grid, by default 'y'

log
[bool, default = True] Whether to log inputs and outputs to log file.

****kwargs**
Keyword arguments to be read directly into whichever scipy.interpolate function is designated by the interp_kind parameter.

Returns

interp_data
[xr.DataArray or xr.Dataset, depending on return_type] By default, returns an xr.DataArray object with the layers added as a new dimension called Layer. Can also specify return_type='dataset' to return an xr.Dataset with each layer as a separate variable.

w4h.layers.**layer_target_thick**(*df*, *layers=9*, *return_all=False*, *export_dir=None*, *outfile_prefix=None*, *depth_top_col='TOP'*, *depth_bot_col='BOTTOM'*, *log=False*)

Function to calculate thickness of target material in each layer at each well point

Parameters

df
[geopandas.geodataframe] Geodataframe containing classified data, surface elevation, bedrock elevation, layer depths, geometry.

layers
[int, default=9] Number of layers in model, by default 9

return_all
[bool, default=False] If True, return list of original geodataframes with extra column added for target thick for each layer. If False, return list of geopandas.geodataframes with only essential information for each layer.

export_dir
[str or pathlib.Path, default=None] If str or pathlib.Path, should be directory to which to export dataframes built in function.

outfile_prefix
[str, default=None] Only used if export_dir is set. Will be used at the start of the exported filenames

depth_top_col
[str, default='TOP'] Name of column containing data for depth to top of described well intervals

depth_bot_col
[str, default='BOTTOM'] Name of column containing data for depth to bottom of described well intervals

log
[bool, default = True] Whether to log inputs and outputs to log file.

Returns

res_df or res
[geopandas.geodataframe] Geopandas geodataframe containing only important information needed for next stage of analysis.

w4h.layers.**merge_metadata**(*data_df*, *header_df*, *data_cols=None*, *header_cols=None*, *auto_pick_cols=False*, *drop_duplicate_cols=True*, *log=False*, *verbose=False*, ***kwargs*)

Function to merge tables, intended for merging metadata table with data table

Parameters

data_df
[pandas.DataFrame] "Left" dataframe, intended for this purpose to be dataframe with main data, but can be anything

header_df
[pandas.DataFrame] "Right" dataframe, intended for this purpose to be dataframe with meta-data, but can be anything

data_cols
[list, optional] List of strings of column names, for columns to be included after join from "left" table (data table). If None, all columns are kept, by default None

header_cols
[list, optional] List of strings of columns names, for columns to be included in merged table after merge from "right" table (metadata). If None, all columns are kept, by default None

auto_pick_cols
[bool, default = False] Whether to autopick the columns from the metadata table. If True, the following column names are kept:['API_NUMBER', 'LATITUDE', 'LONGITUDE', 'BEDROCK_ELEV', 'SURFACE_ELEV', 'BEDROCK_DEPTH', 'LAYER_THICK'], by default False

drop_duplicate_cols
[bool, optional] If True, drops duplicate columns from the tables so that columns do not get renamed upon merge, by default True

log
[bool, default = False] Whether to log inputs and outputs to log file.

****kwargs**
kwargs that are passed directly to pd.merge(). By default, the 'on' and 'how' parameters are defined as on='API_NUMBER' and how='inner'

Returns

mergedTable
[pandas.DataFrame] Merged dataframe

1.2.6 w4h.mapping module

The Mapping module contains the functions used for geospatial analysis throughout the package. This includes some input/output as well as functions to make manipulatin of geospatial data more simple

w4h.mapping.**align_rasters**(*grids_unaligned=None*, *model_grid=None*, *no_data_val_grid=0*, *verbose=False*, *log=False*)

Reprojects two rasters and aligns their pixels

Parameters

grids_unaligned
[list or xarray.DataArray] Contains a list of grids or one unaligned grid

model_grid
[xarray.DataArray] Contains model grid

no_data_val_grid
[int, default=0] Sets value of no data pixels

log
[bool, default = False] Whether to log results to log file, by default False

Returns

alignedGrids
[list or xarray.DataArray] Contains aligned grids

w4h.mapping.**clip_gdf2study_area**(*study_area*, *gdf*, *log=False*, *verbose=False*)

Clips dataframe to only include things within study area.

Parameters

study_area
[geopandas.GeoDataFrame] Inputs study area polygon

gdf
[geopandas.GeoDataFrame] Inputs point data

log
[bool, default = False] Whether to log results to log file, by default False

Returns

gdfClip
[geopandas.GeoDataFrame] Contains only points within the study area

w4h.mapping.**coords2geometry**(*df_no_geometry*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *zcol='ELEV_FT'*, *input_coords_crs='EPSG:4269'*, *output_crs='EPSG:5070'*, *use_z=False*, *wkt_col='WKT'*, *geometry_source='coords'*, *verbose=False*, *log=False*)

Adds geometry to points with xy coordinates in the specified coordinate reference system.

Parameters

df_no_geometry
[pandas.Dataframe] a Pandas dataframe containing points

xcol
[str, default='LONGITUDE'] Name of column holding x coordinate data in df_no_geometry

ycol
[str, default='LATITUDE'] Name of column holding y coordinate data in df_no_geometry

zcol
[str, default='ELEV_FT'] Name of column holding z coordinate data in df_no_geometry

input_coords_crs
[str, default='EPSG:4269'] Name of crs used for geometry

use_z
[bool, default=False] Whether to use z column in calculation

geometry_source
[str {'coords', 'wkt', 'geometry'}]

log
[bool, default = False] Whether to log results to log file, by default False

Returns

gdf
[geopandas.GeoDataFrame] Geopandas dataframe with points and their geometry values

w4h.mapping.**get_drift_thick**(*surface_elev=None*, *bedrock_elev=None*, *layers=9*, *plot=False*, *verbose=False*, *log=False*)

Finds the distance from surface_elev to bedrock_elev and then divides by number of layers to get layer thickness.

Parameters

surface_elev
[rioxarray.DataArray] array holding surface elevation

bedrock_elev
[rioxarray.DataArray] array holding bedrock elevation

layers
[int, default=9] number of layers needed to calculate thickness for

plot
[bool, default=False] tells function to either plot the data or not

Returns

driftThick
[rioxarray.DataArray] Contains data array containing depth to bedrock at each point

layerThick
[rioxarray.DataArray] Contains data array with layer thickness at each point

w4h.mapping.**grid2study_area**(*study_area*, *grid*, *output_crs='EPSG:5070'*, *verbose=False*, *log=False*)

Clips grid to study area.

Parameters

study_area
[geopandas.GeoDataFrame] inputs study area polygon

grid
[xarray.DataArray] inputs grid array

output_crs
[str, default='EPSG:5070'] inputs the coordinate reference system for the study area

log
[bool, default = False] Whether to log results to log file, by default False

Returns

grid
[xarray.DataArray] returns xarray containing grid clipped only to area within study area

w4h.mapping.**read_grid**(*grid_path=None*, *grid_type='model'*, *no_data_val_grid=0*, *use_service=False*, *study_area=None*, *grid_crs=None*, *output_crs='EPSG:5070'*, *verbose=False*, *log=False*, ***kwargs*)

Reads in grid

Parameters

grid_path
[str or pathlib.Path, default=None] Path to a grid file

grid_type
[str, default='model'] Sets what type of grid to load in

no_data_val_grid
[int, default=0] Sets the no data value of the grid

use_service
[str, default=False] Sets which service the function uses

study_area
[geopandas.GeoDataFrame, default=None] Dataframe containing study area polygon

grid_crs
[str, default=None] Sets crs to use if clipping to study area

log
[bool, default = False] Whether to log results to log file, by default False

Returns

gridIN
[xarray.DataArray] Returns grid

w4h.mapping.**read_model_grid**(*model_grid_path*, *study_area=None*, *no_data_val_grid=0*, *read_grid=True*, *node_byspace=True*, *grid_crs=None*, *output_crs='EPSG:5070'*, *verbose=False*, *log=False*)

Reads in model grid to xarray data array

Parameters

grid_path
[str] Path to model grid file

study_area
[geopandas.GeoDataFrame, default=None] Dataframe containing study area polygon

no_data_val_grid
[int, default=0] value assigned to areas with no data

readGrid
[bool, default=True] Whether function to either read grid or create grid

node_byspace
[bool, default=False] Denotes how to create grid

output_crs
[str, default='EPSG:5070'] Inputs study area crs

grid_crs
[str, default=None] Inputs grid crs

log
[bool, default = False] Whether to log results to log file, by default False

Returns

modelGrid
[xarray.DataArray] Data array containing model grid

w4h.mapping.**read_study_area**(*study_area=None*, *output_crs='EPSG:5070'*, *buffer=None*, *return_original=False*, *log=False*, *verbose=False*, ***read_file_kwargs*)

Read study area geospatial file into geopandas

Parameters

study_area
[str, pathlib.Path, geopandas.GeoDataFrame, or shapely.Geometry] Filepath to any geospatial file readable by geopandas. Polygon is best, but may work with other types if extent is correct.

study_area_crs
[str, tuple, dict, optional] CRS designation readable by geopandas/pyproj

buffer
[None or numeric, default=None] If None, no buffer created. If a numeric value is given (float or int, for example), a buffer will be created at that distance in the unit of the study_area_crs.

return_original
[bool, default=False] Whether to return the (reprojected) study area as well as the (reprojected) buffered study area. Study area is only used for clipping data, so usually return_original=False is sufficient.

log
[bool, default = False] Whether to log results to log file, by default False

verbose
[bool, default=False] Whether to print status and results to terminal

Returns

studyAreaIN
[geopandas dataframe] Geopandas dataframe with polygon geometry.

w4h.mapping.**read_wcs**(*study_area*, *wcs_url='https://data.isgs.illinois.edu/arcgis/services/Elevation/IL_Statewide_Lidar_DEM_WGS/ImageSer*
res_x=30, *res_y=30*, *verbose=False*, *log=False*, ***kwargs*)

Reads a WebCoverageService from a url and returns a rioxarray dataset containing it.

Parameters

study_area
[geopandas.GeoDataFrame] Dataframe containing study area polygon

wcs_url
[str, default=lidarURL]

Represents the url for the WCS
res_x
[int, default=30] Sets resolution for x axis

res_y
[int, default=30] Sets resolution for y axis

log
[bool, default = False] Whether to log results to log file, by default False

****kwargs**

Returns

wcsData_rxr
[xarray.DataArray] A xarray dataarray holding the image from the WebCoverageService

w4h.mapping.**read_wms**(*study_area*, *layer_name='IL_Statewide_Lidar_DEM_WGS:None'*,
wms_url='https://data.isgs.illinois.edu/arcgis/services/Elevation/IL_Statewide_Lidar_DEM_WGS/ImageSe
srs='EPSG:3857', *clip_to_studyarea=True*, *bbox=[-9889002.6155, 5134541.069716,*
-9737541.607038, 5239029.6274], *res_x=30*, *res_y=30*, *size_x=512*, *size_y=512*,
format='image/tiff', *verbose=False*, *log=False*, ***kwargs*)

Reads a WebMapService from a url and returns a rioxarray dataset containing it.

Parameters

study_area
[geopandas.GeoDataFrame] Dataframe containg study area polygon

layer_name
[str, default='IL_Statewide_Lidar_DEM_WGS:None'] Represents the layer name in the WMS

wms_url
[str, default=lidarURL] Represents the url for the WMS

srs
[str, default='EPSG:3857'] Sets the srs

clip_to_studyarea
[bool, default=True] Whether to clip to study area or not

res_x
[int, default=30] Sets resolution for x axis

res_y
[int, default=512] Sets resolution for y axis

size_x
[int, default=512] Sets width of result

size_y
[int, default=512] Sets height of result

log
[bool, default = False] Whether to log results to log file, by default False

Returns

wmsData_rxr
[xarray.DataArray] Holds the image from the WebMapService

w4h.mapping.**sample_raster_points**(*raster=None*, *points_df=None*, *well_id_col='API_NUMBER'*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *new_col='SAMPLED'*, *verbose=False*, *log=False*)

Sample raster values to points from geopandas geodataframe.

Parameters

raster
[rioxarray data array] Raster containing values to be sampled.

points_df
[geopandas.geodataframe] Geopandas dataframe with geometry column containing point values to sample.

well_id_col
[str, default="API_NUMBER"] Column that uniquely identifies each well so multiple sampling points are not taken per well

xcol
[str, default='LONGITUDE'] Column containing name for x-column, by default 'LONGITUDE.' This is used to output (potentially) reprojected point coordinates so as not to overwrite the original.

ycol
[str, default='LATITUDE'] Column containing name for y-column, by default 'LATITUDE.' This is used to output (potentially) reprojected point coordinates so as not to overwrite the original. new_col : str, optional

new_col
[str, default='SAMPLED'] Name for name of new column containing points sampled from the raster, by default 'SAMPLED'.

verbose
[bool, default=True] Whether to send to print() information about progress of function, by default True.

log
[bool, default = False] Whether to log results to log file, by default False

Returns

points_df
[geopandas.geodataframe] Same as points_df, but with sampled values and potentially with reprojected coordinates.

w4h.mapping.**xyz_metadata_merge**(*xyz*, *metadata*, *verbose=False*, *log=False*)

Add elevation to header data file.

Parameters

xyz
[pandas.Dataframe] Contains elevation for the points

metadata
[pandas dataframe] Header data file

log
[bool, default = False] Whether to log results to log file, by default False

Returns

headerXYZData
[pandas.Dataframe] Header dataset merged to get elevation values

1.2.7 w4h.read module

The Read module contains funtions primarily for the input of data through the reading of data files, as well as support functions to carry out this task

w4h.read.**add_control_points**(*df_without_control*, *df_control=None*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *zcol='ELEV_FT'*, *controlpoints_crs='EPSG:4269'*, *output_crs='EPSG:5070'*, *description_col='FORMATION'*, *interp_col='INTERPRETATION'*, *target_col='TARGET'*, *verbose=False*, *log=False*, ***kwargs*)

Function to add control points, primarily to aid in interpolation. This may be useful when conditions are known but do not exist in input well database

Parameters

df_without_control
[pandas.DataFrame] Dataframe with current working data

df_control
[str, pathlib.Purepath, or pandas.DataFrame] Pandas dataframe with control points

well_key
[str, optional] The column containing the "key" (unique identifier) for each well, by default 'API_NUMBER'

xcol
[str, optional] The column in df_control containing the x coordinates for each control point, by default 'LONGITUDE'

ycol
[str, optional] The column in df_control containing the y coordinates for each control point, by default 'LATITUDE'

zcol
[str, optional] The column in df_control containing the z coordinates for each control point, by default 'ELEV_FT'

controlpoints_crs
[str, optional] The column in df_control containing the crs of points, by default 'EPSG:4269'

output_crs
[str, optional] The output coordinate system, by default 'EPSG:5070'

description_col
[str, optional] The column in df_control with the description (if this is used), by default 'FORMATION'

interp_col
[str, optional] The column in df_control with the interpretation (if this is used), by default 'INTERPRETATION'

target_col
[str, optional] The column in df_control with the target code (if this is used), by default 'TARGET'

verbose
[bool, optional] Whether to print information to terminal, by default False

log
[bool, optional] Whether to log information in log file, by default False

****kwargs**
Keyword arguments of pandas.concat() or pandas.read_csv that will be passed to that function, except for objs, which are df and df_control

Returns

pandas.DataFrame
Pandas DataFrame with original data and control points formatted the same way and concatenated together

w4h.read.**define_dtypes**(*undefined_df*, *datatypes=None*, *verbose=False*, *log=False*)

Function to define datatypes of a dataframe, especially with file-indicated dyptes

Parameters

undefined_df
[pd.DataFrame] Pandas dataframe with columns whose datatypes need to be (re)defined

datatypes
[dict, str, pathlib.PurePath() object, or None, default = None] Dictionary containing datatypes, to be used in pandas.DataFrame.astype() function. If None, will read from file indicated by dtype_file (which must be defined, along with dtype_dir), by default None

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

dfout
[pandas.DataFrame] Pandas dataframe containing redefined columns

w4h.read.**file_setup**(*well_data*, *metadata=None*, *data_filename='*ISGS_DOWNHOLE_DATA*.txt'*, *metadata_filename='*ISGS_HEADER*.txt'*, *log_dir=None*, *verbose=False*, *log=False*)

Function to setup files, assuming data, metadata, and elevation/location are in separate files (there should be one "key"/identifying column consistent across all files to join/merge them later)

This function may not be useful if files are organized differently than this structure. If that is the case, it is recommended to use the get_most_recent() function for each individual file if needed. It may also be of use to simply skip this function altogether and directly define each filepath in a manner that can be used by pandas.read_csv()

Parameters

well_data
[str or pathlib.Path object] Str or pathlib.Path to directory containing input files, by default str(repoDir)+'/resources'

metadata
[str or pathlib.Path object, optional] Str or pathlib.Path to directory containing input metadata files, by default str(repoDir)+'/resources'

data_filename
[str, optional] Pattern used by pathlib.glob() to get the most recent data file, by default '*ISGS_DOWNHOLE_DATA*.txt'

metadata_filename
[str, optional] Pattern used by pathlib.glob() to get the most recent metadata file, by default '*ISGS_HEADER*.txt'

log_dir
[str or pathlib.PurePath() or None, default=None] Directory to place log file in. This is not read directly, but is used indirectly by w4h.logger_function()

verbose
[bool, default = False] Whether to print name of files to terminal, by default True

log
[bool, default = True] Whether to log inputs and outputs to log file.

Returns

tuple
Tuple with paths to (well_data, metadata)

w4h.read.**get_current_date**()

1.2.7.1 Gets the current date to help with finding the most recent file

Parameters:
None

dateSuffix : str to use for naming output files

w4h.read.**get_most_recent**(*dir=PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/wells4hydrogeology/envs/latest/lib/packages/w4h/resources')*, *glob_pattern='*'*, *verbose=False*)

Function to find the most recent file with the indicated pattern, using pathlib.glob function.

Parameters

dir
[str or pathlib.Path object, optional] Directory in which to find the most recent file, by default str(repoDir)+'/resources'

glob_pattern
[str, optional] String used by the pathlib.glob() function/method for searching, by default '*'

Returns

pathlib.Path object
Pathlib Path object of the most recent file fitting the glob pattern indicated in the glob_pattern parameter.

w4h.read.**get_search_terms**(*spec_path='/home/docs/checkouts/readthedocs.org/user_builds/wells4hydrogeology/checkouts/latest/*, *spec_glob_pattern='*SearchTerms-Specific*'*, *start_path=None*, *start_glob_pattern='*SearchTerms-Start*'*, *wildcard_path=None*, *wildcard_glob_pattern='*SearchTerms-Wildcard'*, *verbose=False*, *log=False*)

Read in dictionary files for downhole data

Parameters

spec_path
[str or pathlib.Path, optional] Directory where the file containing the specific search terms is located, by default str(repoDir)+'/resources/'

spec_glob_pattern
[str, optional] Search string used by pathlib.glob() to find the most recent file of interest, uses get_most_recent() function, by default '*SearchTerms-Specific*'

start_path
[str or None, optional] Directory where the file containing the start search terms is located, by default None

start_glob_pattern
[str, optional] Search string used by pathlib.glob() to find the most recent file of interest, uses get_most_recent() function, by default '*SearchTerms-Start*'

wildcard_path
[str or pathlib.Path, default = None] Directory where the file containing the wildcard search terms is located, by default None

wildcard_glob_pattern
[str, default = '*SearchTerms-Wildcard'] Search string used by pathlib.glob() to find the most recent file of interest, uses get_most_recent() function, by default '*SearchTerms-Wildcard*'

log
[bool, default = True] Whether to log inputs and outputs to log file.

Returns

(specTermsPath, startTermsPath, wilcardTermsPath)
[tuple] Tuple containing the pandas dataframes with specific search terms, with start search terms, and with wildcard search terms

w4h.read.**read_dict**(*file*, *keytype='np'*)

Function to read a text file with a dictionary in it into a python dictionary

Parameters

file
[str or pathlib.Path object] Filepath to the file of interest containing the dictionary text

keytype
[str, optional] String indicating the datatypes used in the text, currently only 'np' is implemented, by default 'np'

Returns

dict
Dictionary translated from text file.

w4h.read.**read_dictionary_terms**(*dict_file=None*, *id_col='ID'*, *search_col='DESCRIPTION'*, *definition_col='LITHOLOGY'*, *class_flag_col='CLASS_FLAG'*, *dictionary_type=None*, *class_flag=6*, *rem_extra_cols=True*, *verbose=False*, *log=False*)

Function to read dictionary terms from file into pandas dataframe

Parameters

dict_file
[str or pathlib.Path object, or list of these] File or list of files to be read

search_col
[str, default = 'DESCRIPTION'] Name of column containing search terms (geologic formations)

definition_col
[str, default = 'LITHOLOGY'] Name of column containing interpretations of search terms (lithologies)

dictionary_type
[str or None, {None, 'exact', 'start', 'wildcard',}]

Indicator of which kind of dictionary terms to be read in: None, 'exact', 'start', or 'wildcard' by default None.

- If None, uses name of file to try to determine. If it cannot, it will default to using the classification flag from class_flag
- If 'exact', will be used to search for exact matches to geologic descriptions
- If 'start', will be used as with the .startswith() string method to find inexact matches to geologic descriptions
- If 'wildcard', will be used to find any matching substring for inexact geologic matches

class_flag
[int, default = 1] Classification flag to be used if dictionary_type is None and cannot be otherwise determined, by default 1

rem_extra_cols
[bool, default = True] Whether to remove the extra columns from the input file after it is read in as a pandas dataframe, by default True

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

dict_terms
[pandas.DataFrame] Pandas dataframe with formatting ready to be used in the classification steps of this package

w4h.read.**read_lithologies**(*lith_file=None*, *interp_col='LITHOLOGY'*, *target_col='CODE'*, *use_cols=None*, *verbose=False*, *log=False*)

Function to read lithology file into pandas dataframe

Parameters

lith_file
[str or pathlib.Path object, default = None] Filename of lithology file. If None, default is contained within repository, by default None

interp_col
[str, default = 'LITHOLOGY'] Column to used to match interpretations

target_col
[str, default = 'CODE'] Column to be used as target code

use_cols
[list, default = None] Which columns to use when reading in dataframe. If None, defaults to ['LITHOLOGY', 'CODE'].

log
[bool, default = True] Whether to log inputs and outputs to log file.

Returns

pandas.DataFrame
Pandas dataframe with lithology information

w4h.read.**read_raw_csv**(*data_filepath*, *metadata_filepath*, *data_cols=None*, *metadata_cols=None*, *xcol='LONGITUDE'*, *ycol='LATITUDE'*, *well_key='API_NUMBER'*, *encoding='latin-1'*, *verbose=False*, *log=False*, ***read_csv_kwargs*)

Easy function to read raw .txt files output from (for example), an Access database

Parameters

data_filepath
[str] Filename of the file containing data, including the extension.

metadata_filepath
[str] Filename of the file containing metadata, including the extension.

data_cols
[list, default = None] List with strings with names of columns from txt file to keep after reading. If None, ["API_NUMBER","TABLE_NAME","FORMATION","THICKNESS","TOP","BOTTOM"], by default None.

metadata_cols
[list, default = None] List with strings with names of columns from txt file to keep after reading. If None, ['API_NUMBER","TOTAL_DEPTH","SECTION","TWP","TDIR","RNG","RDIR","MERIDIAN","QUARTERS
by default None

x_col
[str, default = 'LONGITUDE'] Name of column in metadata file indicating the x-location of the well, by default 'LONGITUDE'

ycol
[str, default = 'LATITUDE'] Name of the column in metadata file indicating the y-location of the well, by default 'LATITUDE'

well_key
[str, default = 'API_NUMBER'] Name of the column with the key/identifier that will be used to merge data later, by default 'API_NUMBER'

encoding
[str, default = 'latin-1'] Encoding of the data in the input files, by default 'latin-1'

verbose
[bool, default = False] Whether to print the number of rows in the input columns, by default False

log
[bool, default = False] Whether to log inputs and outputs to log file.

****read_csv_kwargs**
**kwargs that get passed to pd.read_csv()

Returns

(pandas.DataFrame, pandas.DataFrame/None)
Tuple/list with two pandas dataframes: (well_data, metadata) metadata is None if only well_data is used

w4h.read.**read_xyz**(*xyzpath*, *datatypes=None*, *verbose=False*, *log=False*)

Function to read file containing xyz data (elevation/location)

Parameters

xyzpath
[str or pathlib.Path] Filepath of the xyz file, including extension

datatypes
[dict, default = None] Dictionary containing the datatypes for the columns int he xyz file. If None, {'ID':np.uint32,'API_NUMBER':np.uint64,'LATITUDE':np.float64,'LONGITUDE':np.float64,'ELEV_FT'
by default None

verbose
[bool, default = False] Whether to print the number of xyz records to the terminal, by default False

log
[bool, default = False] Whether to log inputs and outputs to log file.

Returns

pandas.DataFrame
Pandas dataframe containing the elevation and location data

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX

S

V

W

X