acequia

Release 0.01

Thomas de Meij

Feb 25, 2021

INTRODUCTION

Acequia is a python package for wrangling groundwater head data.

CHAPTER

ONE

GETTING STARTED

Instructins for installing Acequia

CHAPTER

TWO

API DOCS - CLASSES FOR HOLDING DATA

The documentation of Acequia's API is generated automatically from the documentation in de python code. The following classes form the main building blocks for working with Acequia:

2.1 module gwseries

This module contains the base object GwSeries for maintaining a groundwater series

Examples

gw = GwSeries.from_dinogws(<filepath to dinocsv file>) gw = GwSeries.from_json(<filepath to acequia json file>)

class acequia.gwseries.**GwSeries**(*heads=None*, *locprops=None*, *tubeprops=None*)

Groundwater heads time series management

from_dinogws(*filepath*)

read heads series from dinoloket csv file

from_json(*filepath*)

read heads series from json file

to_json(*filepath*)

read heads series from json file

to_csv(*filepath*)

read heads series from json file

heads(ref,freq) return timeseries with measured heads

name() return heads series name

locprops(minimal) return location properties, optional minimal=True

tubeprops(last) return tube properties, optinal only last row (last=True)

stats(ref) return descriptice statistics

describe() return selection of properties and descriptive statistics

gxg() return tabel with gxg (desciptive statistics for groundwater series used in the Netherlands)

Examples

To create a GwSeries object from file: >>>gw = GwSeries.from_dinogws(<filepath to dinocsv file>) >>>gw = GwSeries.from_json(<filepath to acequia json file>)

To get GwSeries properties: >>>GwSeries.heads() >>>GwSeries.locprops() >>>GwSeries.name() >>>GwSeries.heads1428()

To export GwSeries data: >>>GwSeries.to_csv(<filename>) >>>GwSeries.To_json(<filename>)

Notes

Head measurements are stored in meters relatieve to welltopStores and served in several units: mwelltop,mref,msurfacelevel.

Valid row names for locprops and column names for tubeprops are stored in class variables locprops_names and tubeprops_names: >>> print(acequia.GwSeries.locprops_names) >>> print(acequia.GwSeries.tubeprops_names)

classmethod from_dinogws (*filepath*)
Read tno dinoloket csvfile with groundwater measurements and return data as gwseries object

Parameters **filepath** (*str*) – path to dinocsv file with measured groundwater heads

Returns **result**

Return type GwSeries object

Example

gw = GwSeries.from_dinogws(<filepath>) jsondict = gw.to_json(<filepath>) gw.from_json(<filepath>)

classmethod from_json (*filepath=None*)
Read gwseries object from json file

to_json (*dirpath=None*)

Create json string from GwWeries object and optionally write to file

Parameters **dirpath** (*str*) – directory json file will be written to (if dirpath is not given no textfile will be written and only OrderedDict with valid JSON wil be retruned)

Returns

Return type OrderedDict with valid json

Note: If no value for dirpath is given, a valid json string is returned. If a value for dirpath is given, nothing is returned and a json file will be written to a file with the series name in dirpath.

---

to_csv (*path=None*, *ref=None*)
Export groundwater heads series to simple csv file

Parameters **path** (*str*) – csv file wil be exported to path, if path is a directory, series will be saved as <path><name>.csv. if path is not given, file is saved in present directory.

Examples

Save heads to simple csv: >>>aq.GwSeries.to_csv(<dirpath>) Read back with standard Pandas: >>>pd.read_csv(<filepath>, parse_dates=['date'],

index_col='date', squeeze=True)

name()
Return groundwater series name

locname()
Return series location name

locprops(*minimal=False*)
return location properties as pd.DataFrame

Parameters **minimal** (*bool, default=False*) – return only minimal selection of columns

Returns

Return type pd.DataFrame

tubeprops(*last=False*, *minimal=False*)
Return tube properties

Parameters

- **last** (*booean, default False*) – retun only the last row of tube properties without date
- **minimal** (*bool, default False*) – return only minimal selection of columns

Returns

Return type pd.DataFrame

surface()
Return last known surface level

heads(*ref='datum'*, *freq=None*)
Return groundwater head measurements

Parameters

- **ref** (*{'mp','datum','surface'}, default 'datum'*) – choosen reference for groundwater heads
- **freq** (*None or any valid Pandas Offset Alias*) – determine frequency of time series

Returns **result**

Return type pandas time Series

Notes

Parameter 'ref' determines the reference level for the heads: 'mp' : elative to well top ('measurement point') 'datum': relative to chosen level (would be meter +NAP for the

Netherlands, or TAW for Belgium)

'surface' : relative to surface level (meter min maaiveld)

Parameter 'freq' determines the time series frequency by setting the Pandas Offset Alias. When 'freq' is None, no resampling is applied. Logical values for 'freq' would be: 'H' : hourly frequency 'D' : calender day frequency 'W' : weekly frequency 'M' : month end frequency 'MS': month start freuency 'Q' : quarter end frequency 'QS': quarter start frequency 'A' : year end frequency 'AS': year start frequency

timestats(*ref=None*)

Return descriptice statistics

Parameters **ref** (*{'mp','datum','surface'}, default 'datum'*) – choosen reference level for groundwater heads

Returns

Return type pd.DataFrame

describe(*ref=None*, *gxg=False*)

Return selection of properties and descriptive statistics

Parameters

- **ref** (*{'mp','datum','surface'}, default 'datum'*) – choosen reference level for groundwater heads
- **gxg** (*bool, default False*) – add GxG descriptive statistics

Returns

Return type pd.DataFrame

tubeprops_changes(*proptype='mplevel'*)

Return timeseries with tubeprops changes

Parameters **proptype** (*['mplevel','surfacelevel','filtop','filbot']*) – tubeproperty that is shown in reference cange graph

Returns

Return type pd.Series

plotheads(*proptype=None*, *filename=None*)

Plot groundwater heads time series

Parameters **proptype** (*['mplevel','surfacelevel','filtop','filbot']*) – tubeproperty that is shown in reference cange graph if not given, no reference plot will be shown

gxg(*ref=None*)

Return table with Gxg desciptive statistics

2.2 module gwlist

2.3 module gwlocs

This module contains the object GwLocs

class acequia.gwlocs.**GwLocs**(*filedir=None*, *pathlist=None*, *filetype=None*, *groups=None*)
Manage multiple groundwater heads series from one well location

Parameters

- **filedir** (*str*) – directory with source files
- **pathlist** (*list, optional*) – list of sourcefile names (without path)
- **filetype** (*['.csv','.json'], optional*) – source file type
- **groups** (*list, optional*) – list of location names, sublists are allowed (see examples)

sourcetable()
return list of location and heads series names

gwseries(*self*, *loc=None*)
return list of GwSeries objects

Examples

Create GwLocs object:

>>>locs = GwLocs(filedir=<jsondir>)

Return table of locations and series:

>>>tbl = locs.loctable()

Return al series for location B16D0037:

>>>gws = GwLocs.gwseries(loc='B16D0037')

Return list of GwSeries objects for locations in list:

>>>names = ['B16D0037','B27G0237',['B28D1635','B28B1388'],'B28B1389']

>>>for loc in GwLocs(filedir=<jsondir>,groups=names):

print(f'{names[i]} group size is {len(gws)}')

Explicitly iterate over locations:

>>>locs = GwLocs(filedir=<jsondir>,groups=names)

>>>for i in range(len(locs)):

gws = next(locs)

print(f'{names[i]} group size is {len(gws)}')

filetable()
Return table of source files for all series

gwseries(*loc=None*)
Return list of GwSeries objects with location name <loc>

Parameters **loc** (*str*) – name of location

Returns

Return type list of GwSeries objects

Notes

When parameter loc is a list of locations names, GwSeries objects for all these locations will be returned.

2.4 module headsdif

Class Headsdif has methods for calculating and plotting heads differences between multiple groundwater series

acequia.headsdif.**headsdif_from_gwseries**(*heads=None*, *locname=None*, *refcol=None*)
Return HeadsDif object with data or None for invalid data

Parameters

- **heads** (*list of GwSeries objects*) – measured heads of multiple groundwater series
- **locname** (*str*) – location name for annotating graphs
- **refcol** (*str, optional*) – series name to use as reference

Returns

Return type HeadsDif object or None

Examples

>>>hd = headsdif_from_gwseries(heads=<heads>,locname=<locname>, refcol=<refcol>)

class acequia.headsdif.**HeadsDif**(*heads=None*, *locname=None*, *refcol=None*)
Calculates head differences and descriptive statistics for multiple groundwater head series

Parameters

- **heads** (*list,pd.DataFrame*) – measured heads of multiple groundwater series
- **locname** (*str*) – location name for annotating graphs
- **refcol** (*str*) – series name to use as reference

Notes

Class HeadsDif requires input of a list of multiple GwSeries objects or pd.Series objects with valid head data. No checking is performed.

To check input data before creating a HeadsDif object, us the custom function headsdif_from_gwseries(). This function returns None when no valid heads could be calculated.

Examples

Create headdiff object without checking input: >>>hd = acequia.headsdif(heads=<list>,locname=<str>, refcol=<str>)

Call function headsdif_from_gwseires() and clean up data before creating a headsdiff object: >>>hd = headsdif_from_gwseries(heads=<heads>,locname=<locname>,

refcol=<refcol>)

>>>if hd is None: >>> continue

table_headsdif(*ref=None*)
Return table with head differences relative to series ref. If ref is not set, first series is taken as reference.

Returns

Return type pd.DataFrame

table_headsref()
Return table with heads relative to mean of entire series

difsums(*period='quarter'*)
Return table with head differences by season

Parameters **seasons** (*{'quarter','half-year'}*) – aggregation level

Returns

Return type pd.DataFrame

date_seasons(*dtindex*, *period='quarter'*)
Return list with seasons for each datetime in index

Parameters

- **dtindex** (*pd.Datarame,pd.DateTimeIndex*) – index with dates
- **seasons** (*{'quarter','half-year'}*) – aggregation level

Returns

Return type list with season for each datetime

plot_time(*figpath=None*, *figsize=None*, *colors=None*)
Plot al heads in one graph and all head differences below that in one figure

plot_head(*figpath=None*, *color='season'*)
Plot head difference by reference head value

plot_freq()
Plot head differences as grid of frequency plots

Returns

Return type fig,ax

CHAPTER

THREE

API DOCS - PLOT CLASSES

The documentation of Acequia's API is generated automatically from the documentation in de python code. The following classes provide functionality for plotting:

3.1 module plotheads

class acequia.plots.plotheads.**PlotHeads**(*ts=[]*, *ref='datum'*, *lbs=None*, *mps=None*, *title=None*, *xlabel=None*, *ylabel=None*, *xlim=None*, *ylim=None*, *colors=None*, *plotargs=None*, *plot=True*)

Class for plotting customized graph of groundwater head series

fig()
Return figure

mindate()
Return very first date in list of heads series

maxdate()
Return very last date in list of heads series

nyears()
Return maximum number of years with measurements

plotheads(*title=None*, *xlabel=None*, *ylabel=None*, *xlim=None*, *ylim=None*, *colors=None*)
Plot groundwater heads and reference graph

save(*filename*, *dpi=None*)
Save figure to file

Parameters

- **filename** (*str*) – filename for saving figure (including extension)
- **dpi** (*number, default 200.0*) – dpi for output figure

3.2 module tsmodelstatsplot

CHAPTER

FOUR

API DOCS - STATISTICS CLASSES

The documentation of Acequia's API is generated automatically from the documentation in de python code. The following classes provide functionality for calculating descriptive statistics from time series:

4.1 module timestats

acequia.stats.timestats.**timestats**(*ts*, *ref=None*, *name=None*)

Return table of groundwater head time series statistics

Parameters

- **ts** (*pd.Series, aq.GwSeries*) – Groundwater head time series
- **ref** (*['datum','surface','mp'], optional, default 'datum'*) – Reference level vfor groundwater heads
- **name** (*str, optional*) – Groundwater heads series name

Returns

Return type pd.DataFrame

class acequia.stats.timestats.**TimeStats**(*ts*, *ref=None*, *name=None*)

Return descriptive statistics of time series

Parameters

- **ts** (*pd.Series, aq.GwSeries*) – timeseries with groundwater head measurments
- **ref** (*str, ['datum','surface'], optinal*) – reference level for measurements
- **name** (*str, optional*) – ground water heads series name

Examples

ts = <valid groundwater heads series of GwSeries object> tsr = aq.TimeStats(ts) tsr.stats()

Notes

Custom function aq.timestats(ts) returns TimeStats.stats() directly.

stats()
Return time series desciptive statistics

Returns

Return type pd.Dataframe

4.2 module quantiles

4.3 module Gxg

This module contains a class GwGxg that calculates some descriptive statistics from a series of groundwater head measurements used by groundwater practitioners in the Netherlands

The structure and many of the methods in this module are adopted from the Pastas module dutch.py)

acequia.stats.gxg.**stats_gxg**(*ts*, *ref='datum'*)
Return table with GxG statistics

Parameters

- **ts** (*aq.GwSeries, pd.Series*) – Groundwater head time series
- **ref** (*{'datum','surface'}, optional*) – Reference level for groundwater heads

Returns

Return type pd.DataFrame

class acequia.stats.gxg.**Gxg**(*gw*, *srname=None*, *ref=None*)
Calculate descriptive statistics for series of measured heads

Parameters

- **gw** (*aq.GwSeries, pd.Series*) – timeseries with groundwater head measurements
- **ref** (*['datum','surface'], default 'surface'*) – reference level for measurements

Notes

Traditionally in the Netherlands groundwater head series are decribed using decriptive statistics that characterise the mean highest level (GHG), the mean lowest level (GLG) and the mean spring level (GVG). These statistics are defined on head series with measurements on the 14th and 28th of each month. Therefore, heads series are internally resampled before calculating statistics.

For further reference: P. van der SLuijs and J.J. de Gruijter (1985). 'Water table classes: a method to decribe seasonal fluctuation and duration of water table classes on Dutch soil maps.' Agricultural Water Management 10 (1985) 109 - 125. Elsevier Science Publishers, Amsterdam.

vg3()
Return VG3 (Spring Level) for each year

VG3 is calculated as the mean of groundwater head levels on 14 march, 28 march and 14 april

Returns

Return type pd.Series

vg1(*maxlag=7*)
Return VG (Spring Level) for each year

VG1 is calculated as measurement nearest to 1 april

Returns

Return type pd.Series

xg()
Return table of GxG groundwater statistics for each hydrological year

Returns

Return type pd.DataFrame

gxg()
Return table with GxG for one head series

Returns

Return type pd.DataFrame

gt()
Return groundwater class table

Returns

Return type str

PYTHON MODULE INDEX

a

INDEX

T

V

X