EcanAlloUsageTools Documentation

Release 1.0.0

Mike Kittridge

Jun 29, 2019

SECTIONS

This package contains a core class (AlloUsage) that provides a variety of methods to extract and combine allocation and usage data. It is primarily designed to return the allocation and usage data as a time series over a period of time in the past.

At the moment, these tools are only usable from within the ECan network. A future installment will optionally utilize external facing web service calls once they have been established.

The GitHub repository is found here. Feedback and contributions are welcome.

CHAPTER

ONE

INSTALLATION

Install via pip:

```
pip install EcanAlloUsageTools
```

Or conda:

```
conda install -c mullenkamp EcanAlloUsageTools
```

1.1 Requirements

The main dependencies are Pandas, pdsql, and seaborn.

CHAPTER

TWO

HOW TO USE ECANALLOUSAGETOOLS

This section will describe how to use the EcanAlloUsageTools package. Nearly all result outputs are Pandas DataFrames.

2.1 Get time series data

The most common use case is to extract a variety of time series data in the form of allocation, metered allocation, lowflow restricted allocation, lowflow restricted metered allocation, and usage datasets. All numeric results returned have the units of m^3.

First, you will need to know which of the above datasets you want. The associated dataset codes are the following: allocation = allo metered allocation = metered_allo lowflow restricted allocation = restr_allo lowflow restricted metered allocation = metered_restr_allo usage = usage

Please see *Package References* for all possible input parameters and filters.

Example:

```
import pandas as pd
from allotools import AlloUsage

pd.options.display.max_columns = 10

# Parameters
from_date = '2015-07-01'
to_date = '2018-06-30'

datasets = ['allo', 'restr_allo', 'metered_allo', 'metered_restr_allo', 'usage']
freq = 'A-JUN'
groupby = ['crc', 'wap', 'date']
site_filter = {'CatchmentGroupName': ['Ashburton River']}

export_path = r'E:\allousagetest'

# Time series extraction
a1 = AlloUsage(from_date, to_date, site_filter=site_filter)

ts1 = a1.get_ts(datasets, freq, groupby, usage_allo_ratio=10).round()

# Plotting
a1.plot_group('A-JUN', val='total', group='crc', with_restr=True, export_path=export_
↪path)

a1.plot_stacked('A-JUN', val='total', export_path=export_path)
```

CHAPTER

THREE

PACKAGE REFERENCES

3.1 Base class

class allotools.**AlloUsage**(*from_date='1900-07-01'*, *to_date='2020-06-30'*, *site_filter=None*, *crc_filter=None*, *include_hydroelectric=False*)

Class to to process the allocation and usage data at ECan.

Parameters

- **from_date** (*str or None*) – The start date of the consent and the final time series. In the form of '2000-01-01'. None will return all consents and subsequently all dates.
- **to_date** (*str or None*) – The end date of the consent and the final time series. In the form of '2000-01-01'. None will return all consents and subsequently all dates.
- **site_filter** (*dict*) – A dict in the form of {str: [values]} to select specific values from a specific column in the ExternalSite table.
- **crc_filter** (*dict*) – A dict in the form of {str: [values]} to select specific values from a specific column in the CrcAllo table.
- **crc_wap_filter** (*dict*) – A dict in the form of {str: [values]} to select specific values from a specific column in the CrcWapAllo table.
- **in_allo** (*bool*) – Should only the consumptive takes be included?
- **include_hydroelectric** (*bool*) – Should hydroelectric takes be included?

Returns with all of the base sites, allo, and allo_wap DataFrames

Return type AlloUsage object

3.2 Get the time series data

AlloUsage.**get_ts**(*self*, *datasets*, *freq*, *groupby*, *irr_season=False*, *usage_allo_ratio=2*, *combine_meters=False*)

Function to create a time series of allocation and usage.

Parameters

- **datasets** (*list of str*) – The dataset types to be returned. Must be one or more of {ds}.
- **freq** (*str*) – Pandas time frequency code for the time interval. Must be one of 'D', 'W', 'M', 'A', or 'A-JUN'.

- **groupby** (*list of str*) – The fields that should grouped by when returned. Can be any variety of fields including crc, take_type, allo_block, 'Wap', CatchmentGroupName, etc. Date will always be included as part of the output group, so it doesn't need to be specified in the groupby.
- **irr_season** (*bool*) – Should the calculations and the resulting time series be only over the irrigation season? The irrigation season is from October through to the end of April.
- **usage_allo_ratio** (*int or float*) – The cut off ratio of usage/allocation. Any usage above this ratio will be removed from the results (subsequently reducing the metered allocation).
- **combine_meters** (*bool*) – When estimating the metered allocation, if one meter on a consent has usage data should all meters on the consent be considered metered? True, will be generous, False will not.
- **Results** –
- **-------** –
- **DataFrame** – Indexed by the groupby (and date)

3.3 plotting methods

AlloUsage.**plot_group**(*self*, *freq*, *val='Total'*, *group='SwazName'*, *with_restr=True*, *yaxis_mag=1000000*, *yaxis_lab='Million'*, *col_pal='pastel'*, *export_path=''*, ***kwargs*)

Function to plot the allocation, metered allocation, and usage as a time series barchart with three adjacent bars per time period. Optionally with restriction volumes.

Parameters

- **freq** (*str*) – The Pandas time series freq.
- **val** (*str*) – The volume value columns. Must be one of 'total', 'gw', or 'sw'.
- **group** (*str*) – The grouping of the plot sets. Where each plot will be broken into the group values.
- **with_restr** (*bool*) – Should the restriction volumes be included in the plots?
- **yaxis_mag** (*int*) – The magnitude that the volumes should be divided by and plotted with on the Y axis.
- **yaxis_lab** (*str*) – The label of the Y axis.
- **col_pal** (*str*) – The seaborn color palette to use.
- **export_path** (*str*) – The path where all the plots will be saved.
- ****kwargs** – Any kwargs to be passed to get_ts.

Returns But outputs many png files to the export_path.

Return type None

AlloUsage.**plot_stacked**(*self*, *freq*, *val='Total'*, *stack='WaterUse'*, *group='SwazName'*, *yaxis_mag=1000000*, *yaxis_lab='Million'*, *col_pal='pastel'*, *export_path=''*, ***kwargs*)

Function to plot the allocation stacked by a specific 'stack' group as a time series barchart.

Parameters

- **freq** (*str*) – The Pandas time series freq.
- **val** (*str*) – The allocation volume column. Must be one of 'Total', 'Gw', or 'Sw'.
- **stack** (*str*) – The field of categories used for the volume stacking.
- **group** (*str*) – The grouping of the plot sets. Where each plot will be broken into the group values.
- **with_restr** (*bool*) – Should the restriction volumes be included in the plots?
- **yaxis_mag** (*int*) – The magnitude that the volumes should be divided by and plotted with on the Y axis.
- **yaxis_lab** (*str*) – The label of the Y axis.
- **col_pal** (*str*) – The seaborn color palette to use.
- **export_path** (*str*) – The path where all the plots will be saved.
- ****kwargs** – Any kwargs to be passed to get_ts.

Returns But outputs many png files to the export_path.

Return type None

3.4 API Pages

—

CHAPTER

FOUR

LICENSE AND TERMS OF USAGE

This package is licensed under the terms of the Apache License Version 2.0 and can be found on the GitHub project page.

INDEX