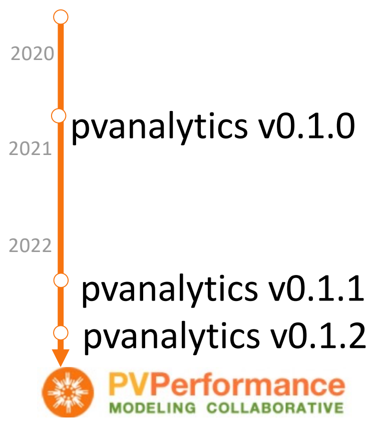
PVAnalytics: A Python Package for Automated Processing of Solar Time Series Data



Kirsten Perry (NREL), William Vining (Sandia), Kevin Anderson (NREL), Matthew Muller (NREL), Cliff Hansen (Sandia)

PV Performance Modeling and Monitoring Workshop
Salt Lake City, Aug 24, 2022

https://github.com/pvlib/pvanalytics

Contents

PVAnalytics Background

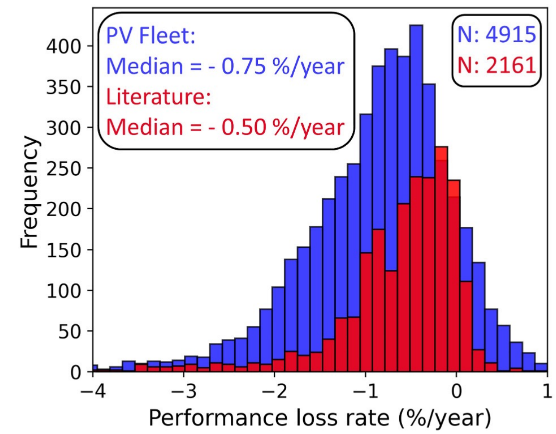
- Solar time series data can vary significantly in quality or lack critical metadata
- Several solar metrics dependent on data cleaning/filtering [1]
  - Performance loss rate (PLR)
  - Power production forecasting
  - Soiling loss
- **PVAnalytics Python library:** automated processing of solar time series data, including QA/QC
  - Data quality control and filtering
  - Identifying system characteristics, such as mounting configuration, tilt, and azimuth
  - Feature identification: clipping, day-night masking, clearsky detection
  - https://pvanalytics.readthedocs.io/en/stable/

[1] Lindig et. al. *International collaboration framework for the calculation of performance loss rates: Data quality, benchmarks, and trends (towards a uniform methodology).* Progress in Photovoltaics, 2021.

PVAnalytics Background (Continued)

- **Design Principles behind PVAnalytics:**
  - Open-source: tested, documented, and re-usable
  - Independent of analysis workflow
  - Collection point for code which implements published algorithms

- Collaboration between Sandia and NREL
  - Started as DuraMAT project: DOE-led consortium for PV module reliability and durability
  - Functions adapted from Solar Forecast Arbiter [1] and NREL PV Fleets Initiative [2]

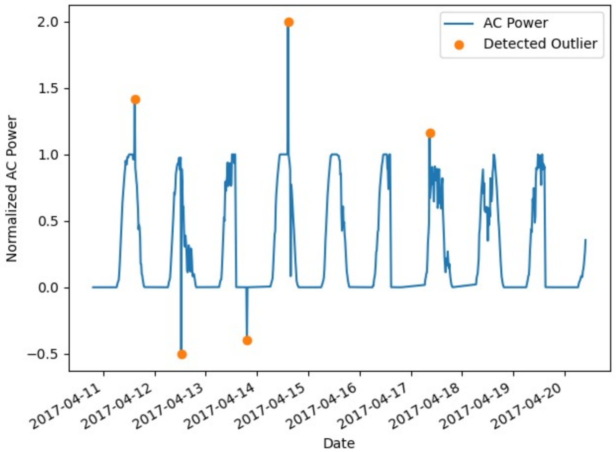


PLR distribution from the PV Fleets Initiative [2]

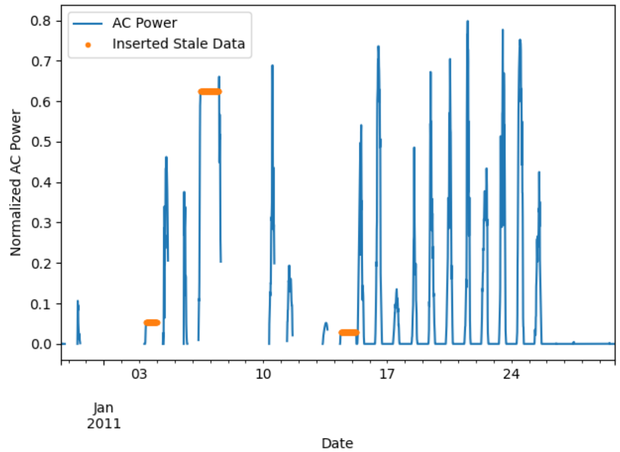
[1] https://solarforecastarbiter-core.readthedocs.io/en/latest/
[2] D. Jordan et. al. *Photovoltaic fleet degradation insights*. Progress in Photovoltaics, 2022.

Package Features: Basic Time Series Filtering

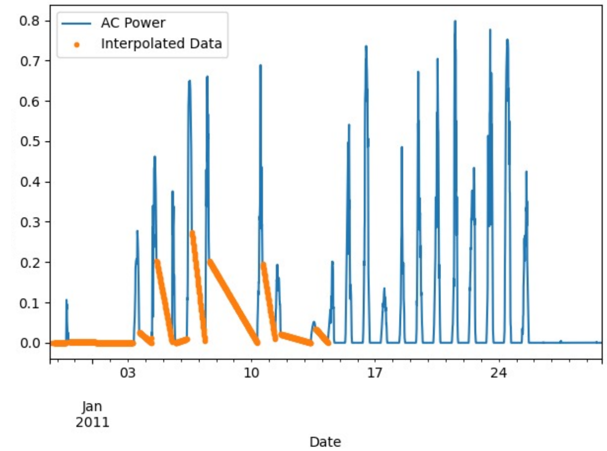
Outlier detection and filtering: Hampel, Z-score, and Tukey filters



Stale data detection and filtering: Looks for consecutive repeating data

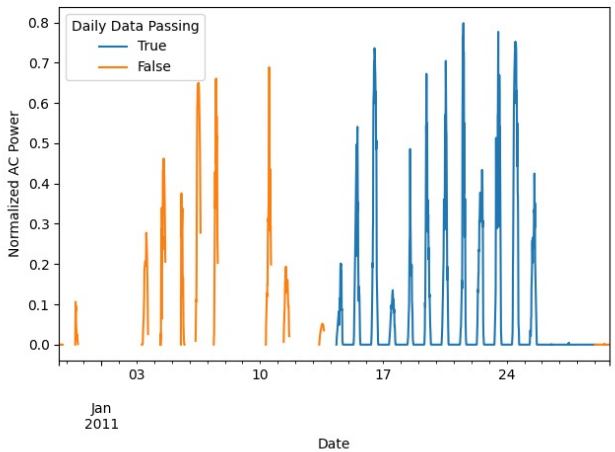


Interpolated data detection and filtering



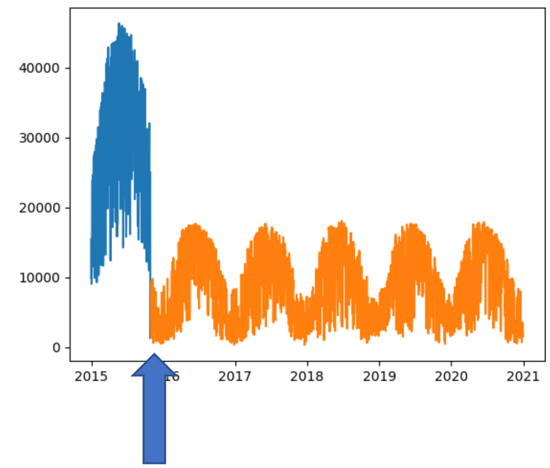
Package Features: Advanced Time Series Filtering

Detecting missing data periods: Assign daily data a "completeness" score



Filtering days based on daily "completeness" score

Data shift detection and filtering: Uses changepoint detection to find massive, abrupt capacity changes. Described further in [1]



Detected data shift here

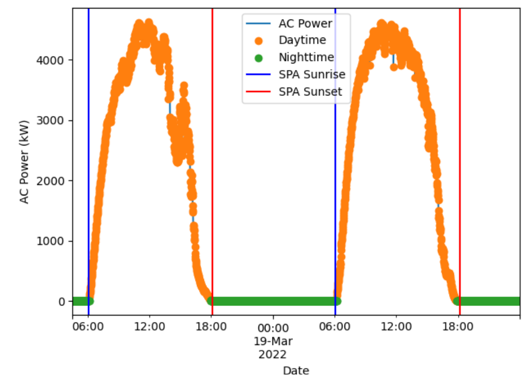
[1] K. Perry, M. Muller. *Automated Shift Detection in Sensor-Based PV Power and Irradiance Time Series*. 2022 PVSC.

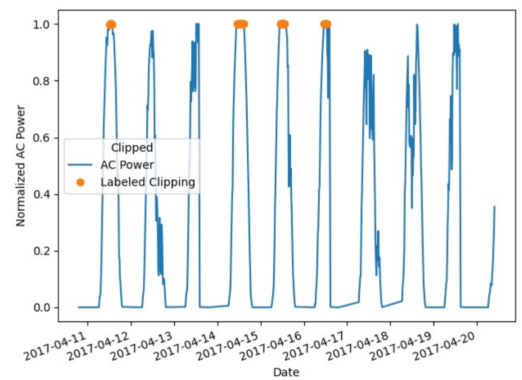
Package Features: Feature Detection

- **Day-night masking**
  - Logic-based routine for masking day periods from night periods
- **Clipping detection and filtering**
  - Adapted from logic-based filter described in [1]
- **Shading detection**
  - Uses morphological image processing methods to identify shadows in GHI data [2]

[1] K. Perry, et. al. *Performance comparison of clipping detection techniques in AC power time series*. 2021 PVSC.
[2] Martin, C. E., Hansen, C. W., An Image Processing Algorithm to Identify Near-Field Shading in Irradiance Measurements, preprint 2016

Day-night masking on an AC power time series

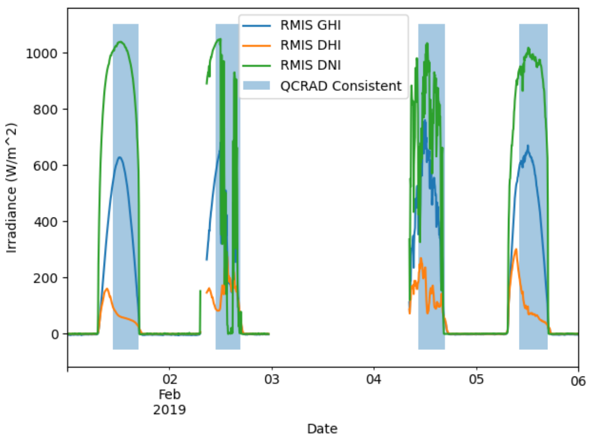




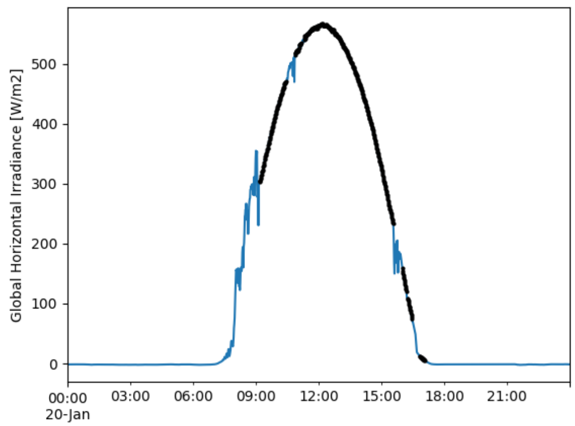
Masked clipping periods in time series data

Package Features: Irradiance Checks

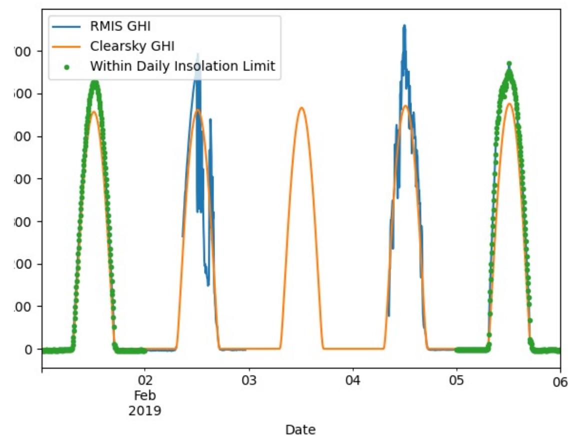
Irradiance quality checks: consistency and physical limits of GHI, DNI, and DHI using QCrad criteria



Clearsky period filtering: Reno clearsky method (1)



Clearsky day filtering: Compare GHI sensor-based data to clearsky data. Filter where GHI is within daily insolation limit

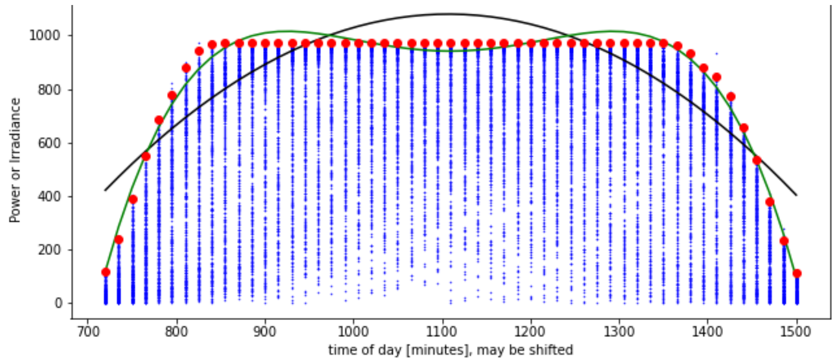


[1] Reno, M.J. and C.W. Hansen, "Identification of periods of clear sky irradiance in time series of GHI measurements" Renewable Energy, v90, p. 520-531, 2016.

Package Features: System Characteristics

- **Mounting configuration**
  - Fixed-tilt or single-axis tracking
  - Uses daily power profile to classify time series stream

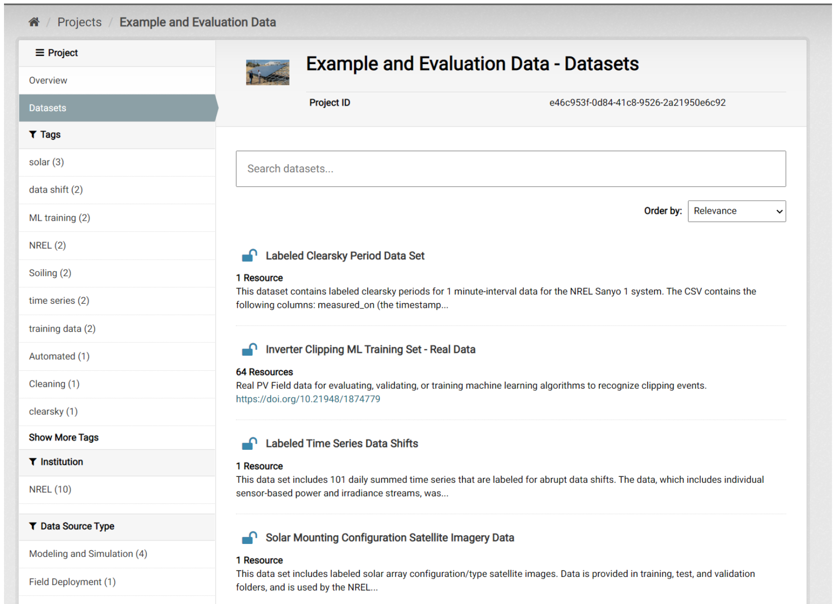
- **Azimuth and tilt**
  - Estimate using AC power time series
  - **Work in progress:** multiple methods in package are currently being validated



Daily power profile of a single-axis tracking system

Algorithm Validation

- Continued validation of each algorithm
  - **How well does each algorithm perform on labeled data sets?**
    - Quantifiable metrics: accuracy and F1-score
    - Labeled data sets to encourage further development
- Technical documentation/publications benchmarking each algorithm's performance

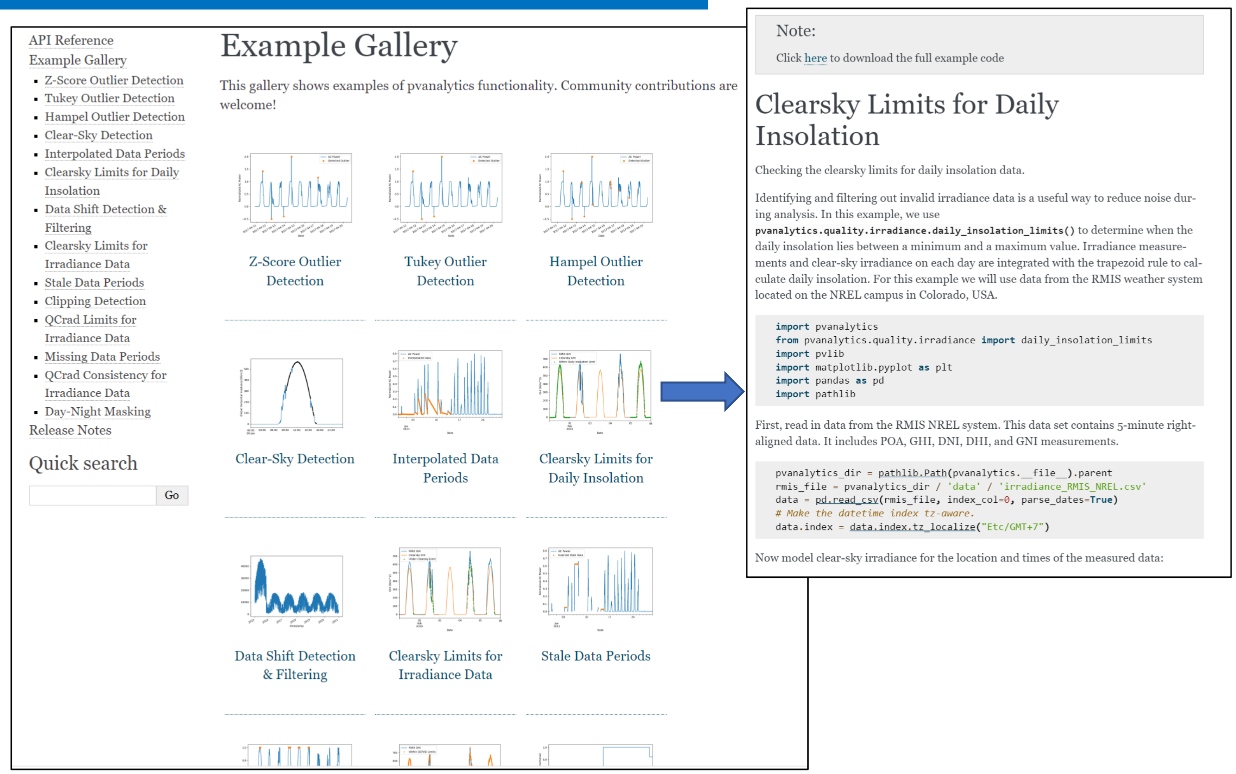


Publicly available, labeled data sets on the DuraMAT DataHub

https://datahub.duramat.org/project/example-data

Documentation: Example Gallery

- Example gallery for majority of the package functions (v0.1.2)
  - Example data for running each algorithm
  - Plots illustrating algorithm results



https://pvanalytics.readthedocs.io/en/stable/generated/gallery/index.html

Apply PVAnalytics to Your Own Data

How can you easily implement PVAnalytics functions to your own data?

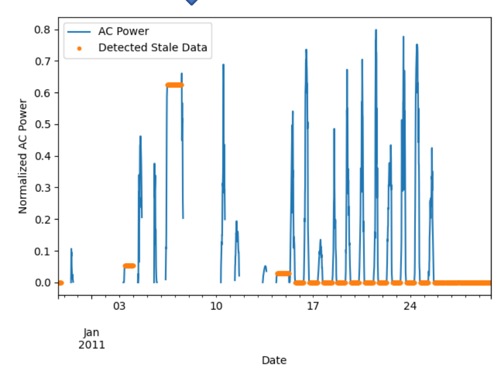
CSV containing data streams (power, irradiance, temperature)

is labeled as False. The data is sampled at 15-minute intervals.

```
pvanalytics_dir = pathlib.Path(pvanalytics.__file__).parent
file = pvanalytics_dir / 'data' / 'ac_power_inv_2173_stale_data.csv'
data = pd.read_csv(file, index_col=0, parse_dates=True)
data = data.asfreq("15T")
data['value_normalized'].plot()
data.loc[data["stale_data_mask"], "value_normalized"].plot(ls='', marker='.
plt.legend(labels=["AC Power", "Inserted Stale Data"])
plt.xlabel("Date")
plt.ylabel("Normalized [illegible]")
plt.tight_layout()
plt.show()
```

Import CSV into our example documentation, and change any metadata parameters (lat-long coordinates, data frequency, etc)

Run associated example!



Analyze outputs

https://pvanalytics.readthedocs.io/en/stable/generated/gallery/index.html

Documentation: Function Descriptions

- Page for each model function containing:
  - Brief description
  - Input parameters: data type, description
  - Outputs: data type, description
  - Published reference for the function, if applicable
  - Additional notes as needed
  - Examples in the gallery using the function

https://pvanalytics.readthedocs.io/en/stable/api.html

Function description for pvanalytics.quality.stale_values_diff

pvanalytics.quality.gaps.stale_values_diff

`pvanalytics.quality.gaps.stale_values_diff(x, window=6, rtol=1e-05, atol=1e-08, mark='tail')`

Identify stale values in the data.

For a window of length N, the last value (index N-1) is considered stale if all values in the window are close to the first value (index 0).

Parameters *rtol* and *atol* have the same meaning as in `numpy.allclose()`.

Parameters:
- **x** (*Series*) – data to be processed
- **window** (*int, default 6*) – number of consecutive values which, if unchanged, indicates stale data
- **rtol** (*float, default 1e-5*) – relative tolerance for detecting a change in data values
- **atol** (*float, default 1e-8*) – absolute tolerance for detecting a change in data values
- **mark** (*str, default 'tail'*) – How much of the window to mark `True` when a sequence of stale values is detected. Can one be of 'tail', 'end', or 'all'.
  - If 'tail' (the default) then every point in the window *except* the first point is marked `True`.
  - If 'end' then the first *window - 1* values in a stale sequence sequence are marked `False` and all subsequent values in the sequence are marked `True`.
  - If 'all' then every point in the window *including* the first point is marked `True`.

Returns: True for each value that is part of a stale sequence of data

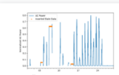
Return type: Series

Raises: **ValueError** – If *window* < *2* or *mark* is not one of 'tail', 'end', or 'all'.

Notes

Copyright (c) 2019 SolarArbiter. See the file LICENSES/SOLARFORECASTARBITER_LICENSE at the top level directory of this distribution and at https://github.com/pvlib/pvanalytics/blob/master/LICENSES/SOLARFORECASTARBITER_LICENSE for more information.

Examples using `pvanalytics.quality.gaps.stale_values_diff`

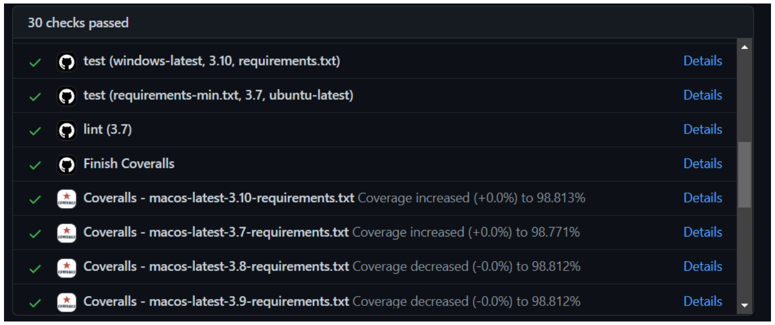


Stale Data Periods

Automated Testing

- Comprehensive unit-testing for all package functions
  - ~100% test coverage
  - Uses Pytest and Coveralls

- Since package is in its infancy, no speed benchmarks have been taken (yet!)



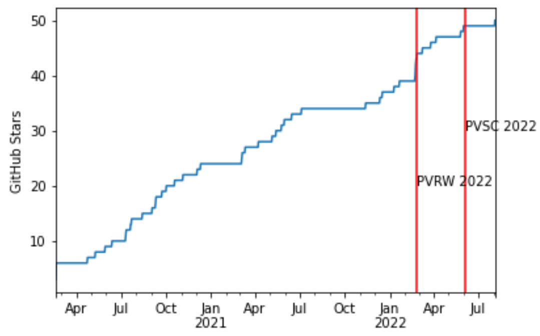
Package checks required to pass before merging PR

Current test coverage

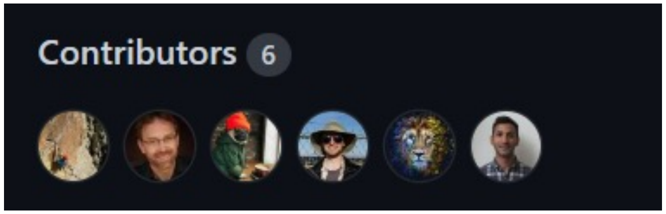
Community growth

- Github
  - 88 completed pull requests
  - Code contributions from 6 people (see lower right)
- Lots of opportunity to increase community growth as PVAnalytics is still in its infancy

- **You can contribute!**
  - Generate issues for features you'd like to see, add code via our PR process, etc.



Github stars over time



Special thanks to all our contributors!

PVAnalytics v0.1.3 and Beyond

- No expected ETA for next release but we're actively working on new functions/documentation
- Future version features:
  - Daylight savings time (DST) and time-drift detection algorithms for time series
  - Adding plotting module to easily validate time-series data visually

Thank you!

www.nrel.gov

kirsten.perry@nrel.gov

This work was authored in part by Alliance for Sustainable Energy, LLC, the manager and operator of the National Renewable Energy Laboratory for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under Solar Energy Technologies Office (SETO) Agreement Number 38258. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

NREL

Transforming ENERGY