PVInsight

Data Driven Approaches for Analyzing PV System Performance and Reliability

Bennet Meyers

Project Scientist, GISMo Team, SLAC National Accelerator Laboratory
PhD Candidate, Electrical Engineering, Stanford University

February 2020
DuraMAT Webinar Series

Table of Contents

Table of Contents

Background: More Data, More Opportunities

- Increasing volume of photovoltaic (PV) system performance data creates opportunities for monitoring system health and optimizing operations and maintenance (O&M) activities.
- Digital O&M $9b industry by 2024 ("The State of Digital O&M for the Solar Market", *Greentech Media*, 10/10/19)
- However, classic approaches—waterfall analysis, performance index analysis—require
  - A significant amount of engineering time
  - Knowledge of PV system modeling science and best practices
  - Accurate system configuration information
  - Access to reliable irradiance and meteorological data

New Approaches are Needed

For these reasons, existing PV system performance engineering methods are focused on utility scale power plants...

...rather than the rapidly increasing number of distributed rooftop systems.

Image credit: SunPower Corp.

Image credit: Google Earth

Utility vs. Distributed

	Utility	Distributed
Site model	✓	✗
Irradiance data	✓	✗
Meteorological data	✓	✗
People / PV system	> 1	$\ll 1$

- New approaches needed to analyze and managed distributed PV
- How to extract insight into system health from only a power signal?

The Goals of PVInsight

- Develop novel PV performance analysis techniques that are automatic and require only measured power
- Use cutting edge approaches to develop algorithms
  - Signal processing
  - Optimization
  - Unsupervised machine learning
- Publish tools as open-source software (GitHub, PyPI, Anaconda)
  - `solar-data-tools`: Data preprocessing, cleaning, and filtering
  - `statistical-clear-sky`: Clear sky modeling and degradation analysis
  - More packages coming!

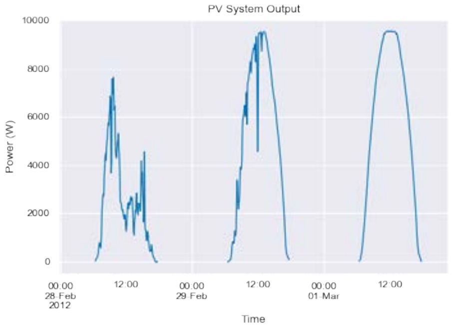
Table of Contents

Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection

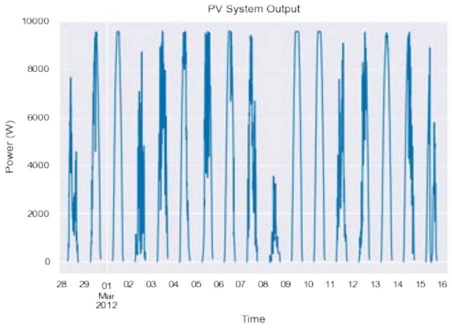
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



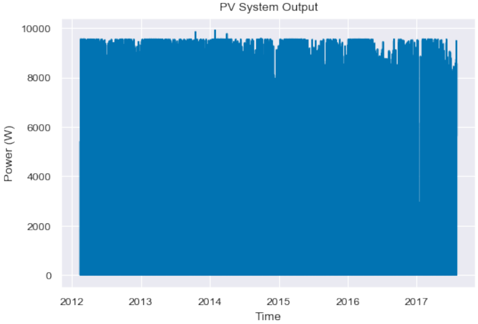
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



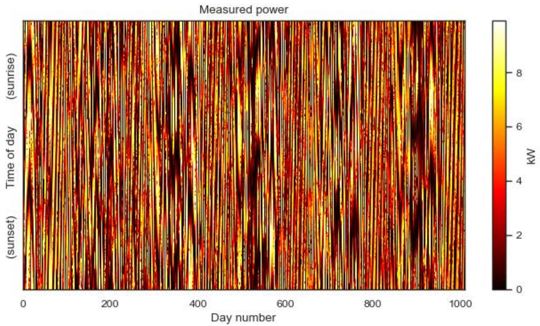
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



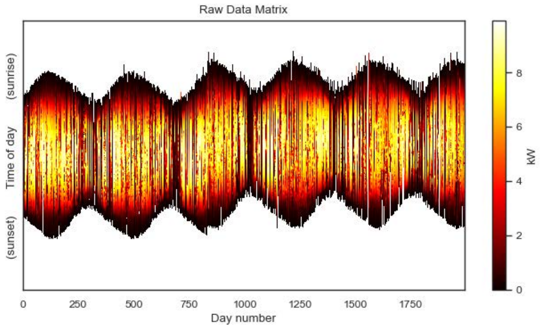
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



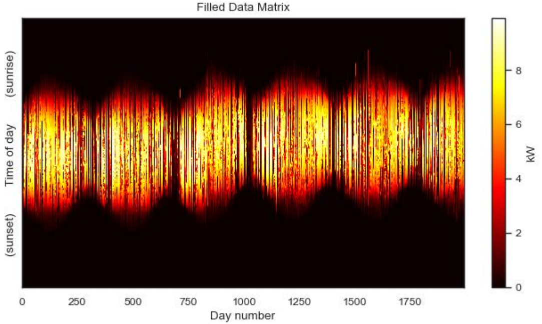
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



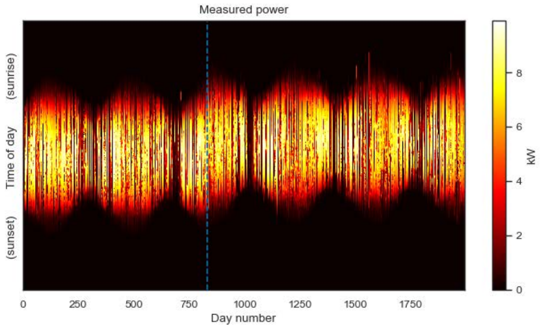
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



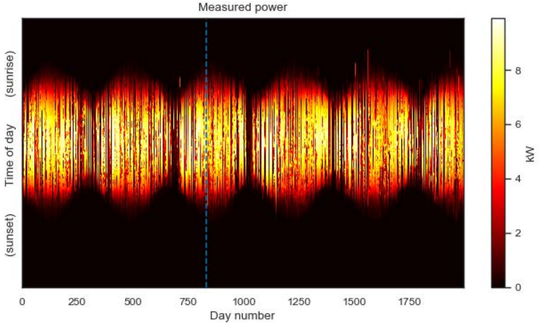
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



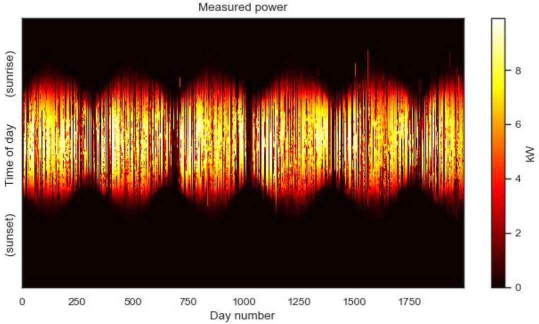
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



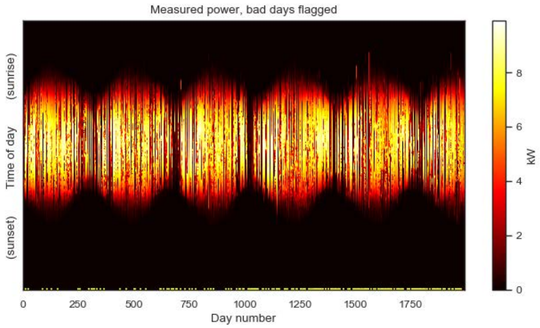
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



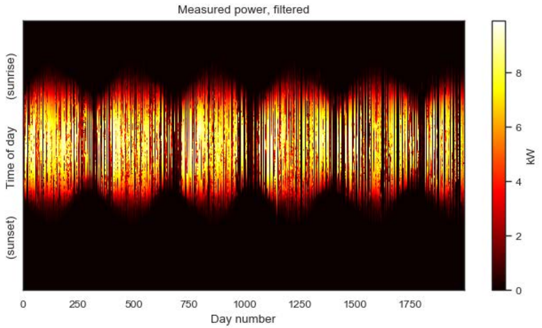
Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection



Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification
  - Inverter clipping detection
  - System capacity change detection

Running the `DataHandler` processing pipeline

```
In [33]: dh = DataHandler(data_frame)

In [34]: dh.run_pipeline(use_col='ac_power')

total time: 1.92 seconds
form matrix: 0.05, fill matrix: 0.63, fix time shifts: 0.43,
daily scores: 0.25, daily flags: 0.04, clear detect: 0.15,
clipping check: 0.19, data scoring: 0.17
```

Top-level reporting

First we view a quick summary of the data set.

```
In [35]: dh.report()

Length:                1459 days
Data sampling:         15 minute
Data quality score:    95.8%
Data clearness score:  53.3%
Inverter clipping:     False
Time shifts corrected: False
```

Python Software

`solar-data-tools` on GitHub, PyPI, and Anaconda.

Solar Data Tools

- Preprocessing
  - Time stamp cleaning
  - Matrix embedding
- Cleaning
  - Missing data filling
  - Time shift detection and correction ←
- Filtering
  - Data quality / corrupt data
  - Clear day / cloudy day identification ←
  - Inverter clipping detection
  - System capacity change detection

Running the `DataHandler` processing pipeline

```
In [33]: dh = DataHandler(data_frame)

In [34]: dh.run_pipeline(use_col='ac_power')

total time: 1.92 seconds
form matrix: 0.05, fill matrix: 0.63, fix time shifts: 0.43,
daily scores: 0.25, daily flags: 0.04, clear detect: 0.15,
clipping check: 0.19, data scoring: 0.17
```

Top-level reporting

First we view a quick summary of the data set.

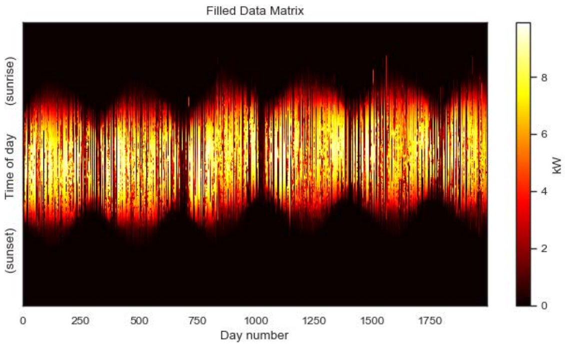
```
In [35]: dh.report()

Length:                 1459 days
Data sampling:          15 minute
Data quality score:     95.8%
Data clearness score:   53.3%
Inverter clipping:      False
Time shifts corrected:  False
```

Python Software

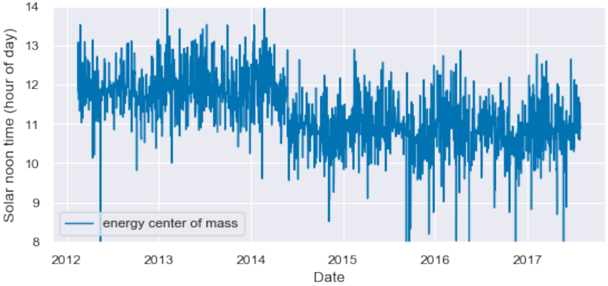
`solar-data-tools` on GitHub, PyPI, and Anaconda.

Time shift detection algorithm



Time shift detection algorithm

1. Estimate solar noon on each day from data

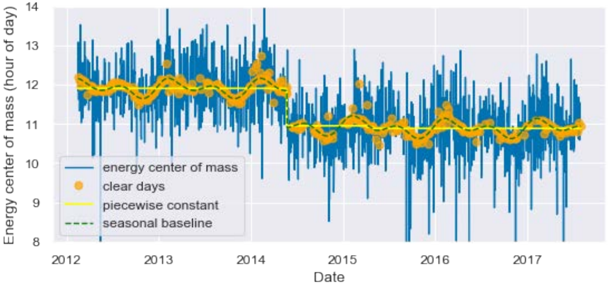


Two Options

- Calculate the *energy center of mass* on each day (default behavior)
- Find the sunrise and sunset times and take the average

Time shift detection algorithm

1. Estimate solar noon on each day from data
2. Fit signal separation model



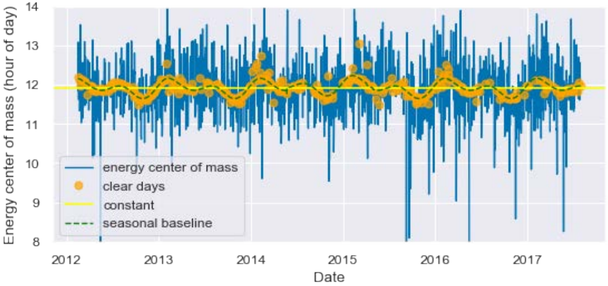
Optimal Signal Demixing (OSD)

A novel approach to blind signal separation, based on convex optimization[1]

[1] https://bmeyers.github.io/QualsSlides/

Time shift detection algorithm

1. Estimate solar noon on each day from data
2. Fit signal separation model
3. Identify shift points and estimate correction factors



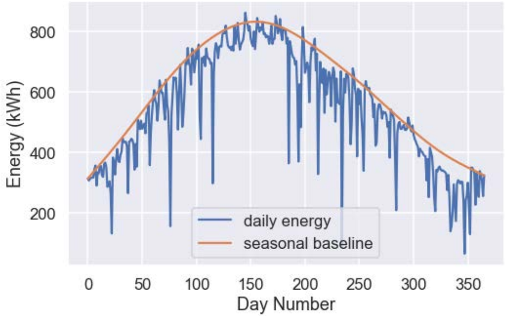
Piecewise constant → constant

The piecewise constant signal component automatically find the shift points and gives the estimate of the correction factor

[1] https://bmeyers.github.io/QualsSlides/

Clear day / cloudy day identification

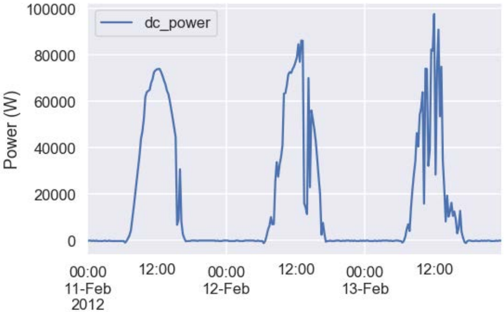
Daily energy content



- Clear days have more energy relative to seasonal baseline
- Some high energy days can see be partially cloudy

Tool: *local quantile regression*

Daily smoothness

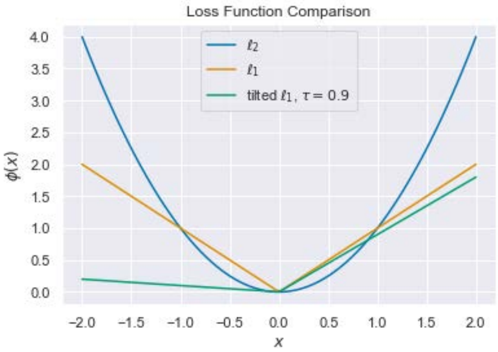


- Clear days are smoother in time than partially cloudy days
- Some very cloudy days can also exhibit smoothness

Tool: *discrete differences*

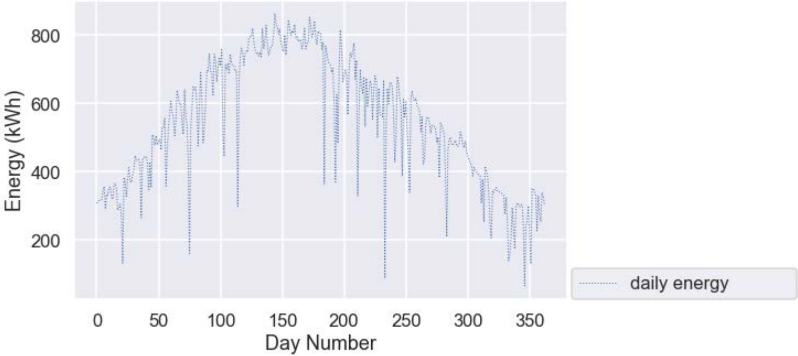
Estimating the seasonal baseline

- Again use OSD
- Separate measured daily energy signal into
  - A smooth, periodic signal
  - Non-Gaussian, skewed noise
- Use a *quantile regression* or *tilted* ℓ_1 cost function on residuals instead of typical ℓ_2 (sum-of-squares) loss



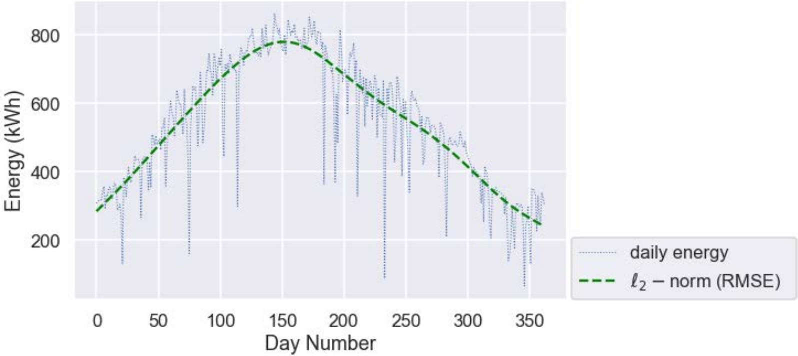
$$\phi_\tau(r) = \tau(x)_+ + (1-\tau)(x)_- = \frac{1}{2}|x| + \left(\tau - \frac{1}{2}\right)x$$

Local Quantile Regression



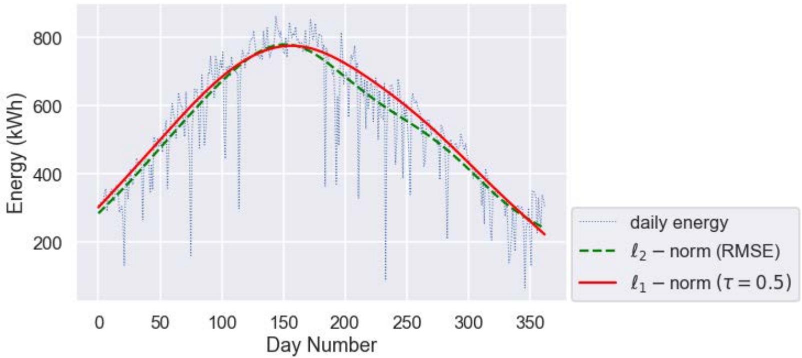
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



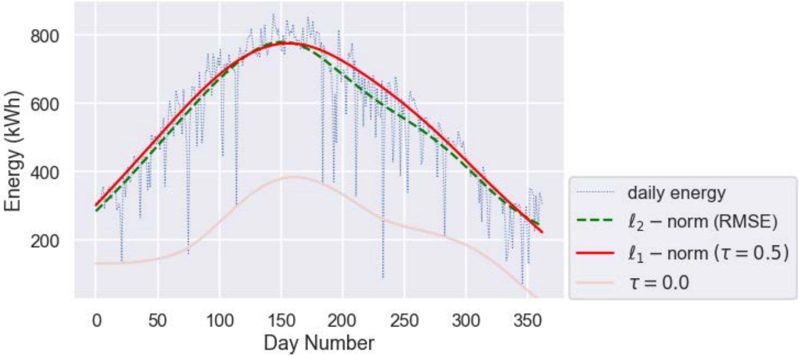
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



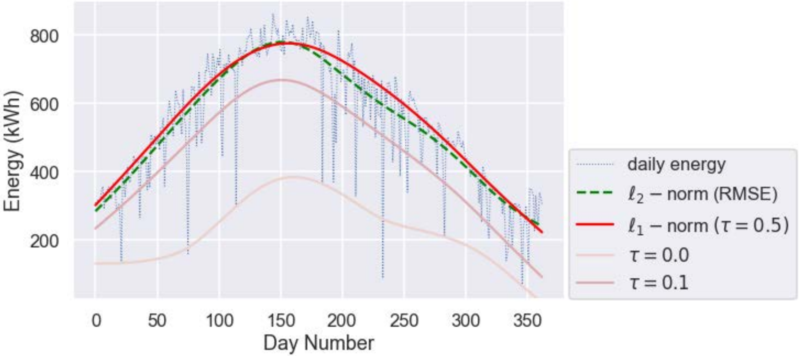
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



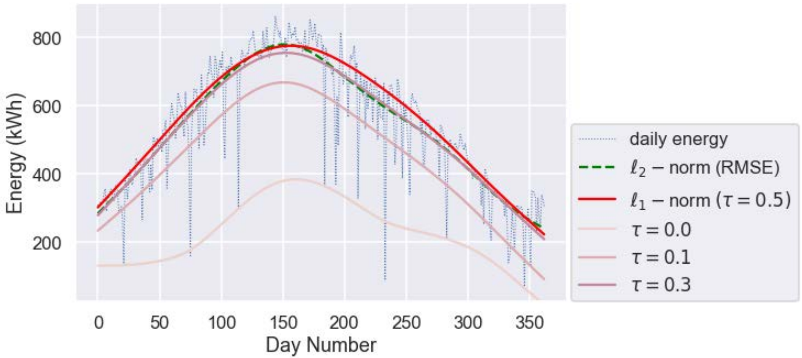
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



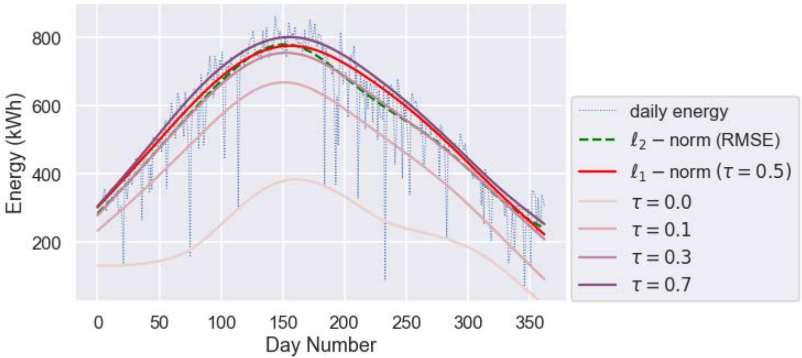
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



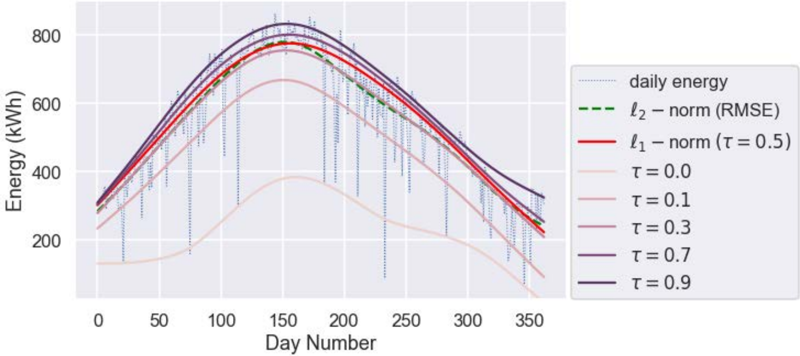
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



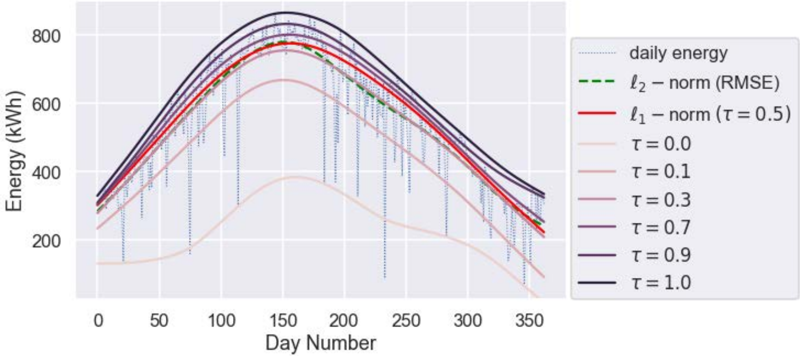
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



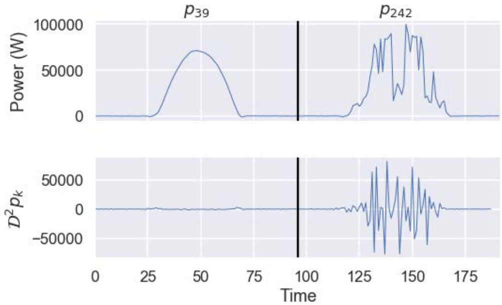
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Discrete Differences and Smoothness

- Our smoothness metric is the second-order discrete difference:

$$s_k = \|p_k[t-1] - 2p_k[t] + p_k[t+1]\|_2$$

- $p_k \in \mathbf{R}^m$ is the k^{th} column of the power matrix, the power signal on day k
- $p_k[t-1] - 2p_k[t] + p_k[t+1]$ measures the local "curvature" of the signal
- Taking the ℓ_2-norm of each daily segment measures the overall "roughness" of the day

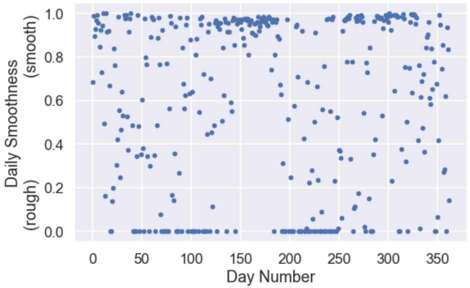


Discrete Differences and Smoothness

Finally, we do a little rescaling to turn s_k into a metric between 0 and 1.

In [95]:

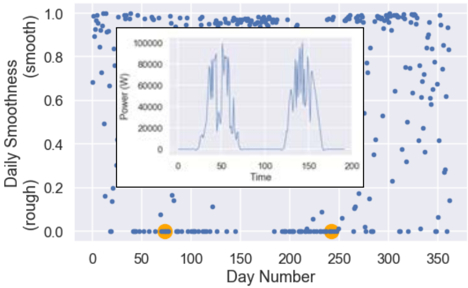
```
smoothness = np.linalg.norm(np.diff(D, n=2, axis=0), axis=0)
smoothness -= np.quantile(smoothness, 0.8)
smoothness /= np.min(smoothness)
smoothness = np.clip(smoothness, 0, 1)
plt.figure(figsize=(8, 5))
plt.plot(smoothness, marker='.', linestyle='None')
plt.ylabel('Daily Smoothness\n(rough)                    (smooth)')
plt.xlabel('Day Number');
```



Discrete Differences and Smoothness

Finally, we do a little rescaling to turn s_k into a metric between 0 and 1.

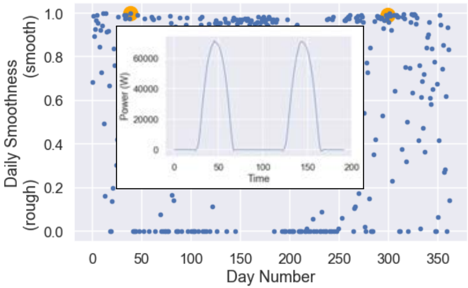
In [95]:
```
smoothness = np.linalg.norm(np.diff(D, n=2, axis=0), axis=0)
smoothness -= np.quantile(smoothness, 0.8)
smoothness /= np.min(smoothness)
smoothness = np.clip(smoothness, 0, 1)
plt.figure(figsize=(8, 5))
plt.plot(smoothness, marker='.', linestyle='None')
plt.ylabel('Daily Smoothness\n(rough)                          (smooth)')
plt.xlabel('Day Number');
```



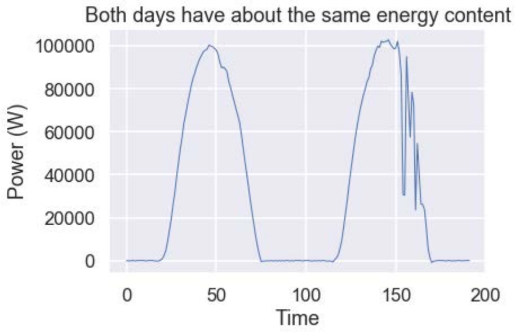
Discrete Differences and Smoothness

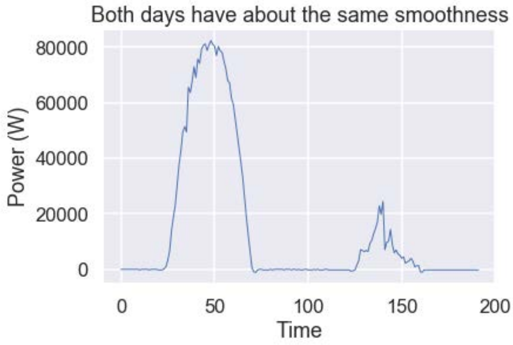
Finally, we do a little rescaling to turn s_k into a metric between 0 and 1.

In [95]:
```
smoothness = np.linalg.norm(np.diff(D, n=2, axis=0), axis=0)
smoothness -= np.quantile(smoothness, 0.8)
smoothness /= np.min(smoothness)
smoothness = np.clip(smoothness, 0, 1)
plt.figure(figsize=(8, 5))
plt.plot(smoothness, marker='.', linestyle='None')
plt.ylabel('Daily Smoothness\n(rough)                    (smooth)')
plt.xlabel('Day Number');
```



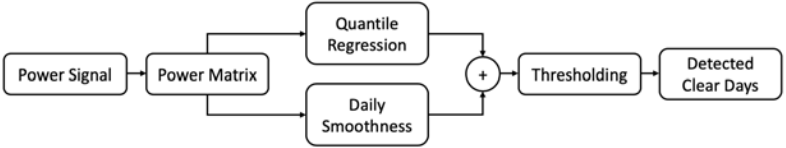
Both Methods Are Imperfect

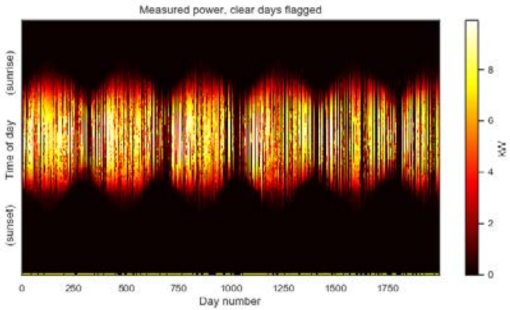




- Not all high energy days are clear, but *all clear days are high energy*.
- Not all smooth days are clear, but *all clear days are smooth*.

Putting it all together





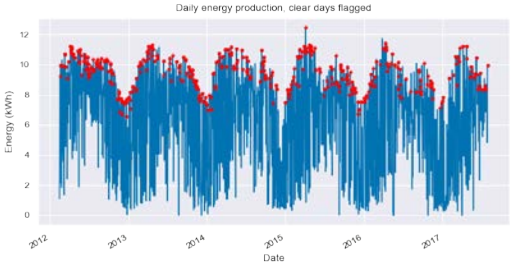
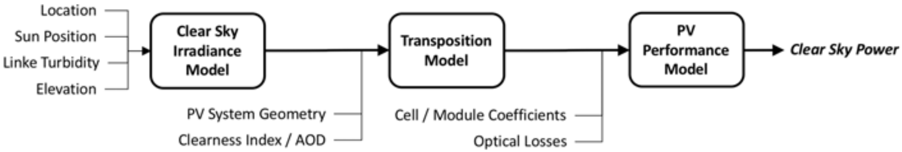


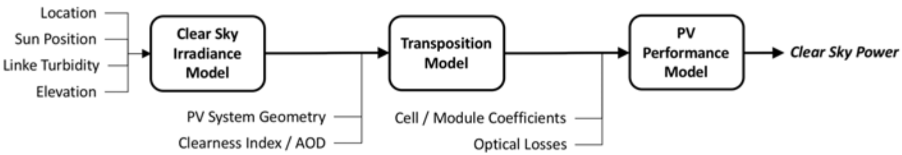
Table of Contents

Clear sky models



See `pvlib-python` (Holmgren, 2015) for implementation / open-source code

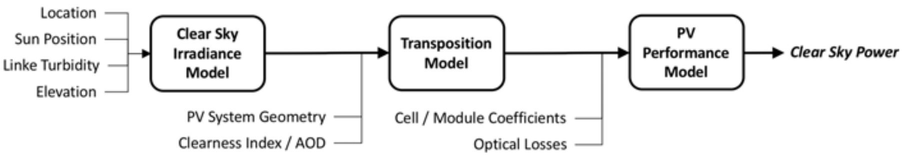
Clear sky models



- Requires lots of measured/estimated input data

See `pvlib-python` (Holmgren, 2015) for implementation / open-source code

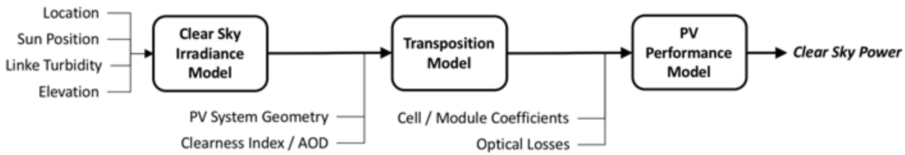
Clear sky models



- Requires lots of measured/estimated input data
- Difficult to tune model to match observed data

See `pvlib-python` (Holmgren, 2015) for implementation / open-source code

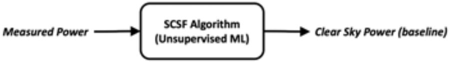
Clear sky models



- Requires lots of measured/estimated input data
- Difficult to tune model to match observed data
- Can't handle non-ideal behavior such as site shading

See `pvlib-python` (Holmgren, 2015) for implementation / open-source code

Unsupervised Machine Learning for Clear Sky Modeling: SCSF



- Statistical Clear Sky Fitting[2] (SCSF) estimates the clear sky power output of a system, given historical data.
- Starting with power data in matrix form, SCSF finds an approximate *factorization*

$$P \approx L \times R = P_{\text{clear sky}}$$

- This is known as *Generalized Low Rank Modeling*[3], related to PCA, SVD, etc.

Python Software

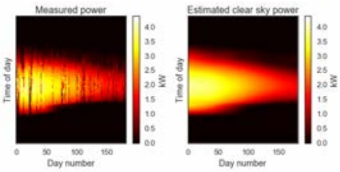
`statistical-clear-sky` on GitHub, PyPI, and Anaconda.

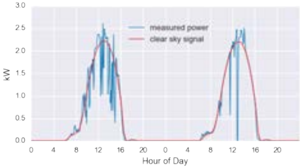
[2]B. Meyers, M. Tabone, and E. C. Kara, "Statistical Clear Sky Fitting Algorithm," 2018
[3]M. Udell, C. Horn, R. Zadeh, and S. Boyd, "Generalized Low Rank Models," 2016

SCSF Visualization

System 1

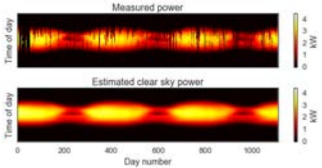
- 5-minute sampling
- 6 months of data

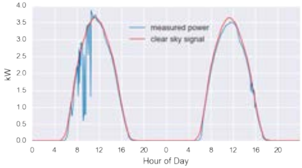




System 2

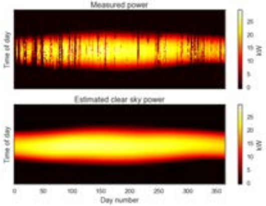
- 5-minute sampling
- 3 years of data

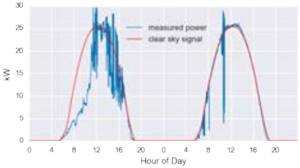




System 3

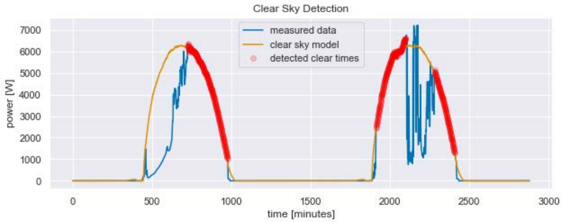
- 1-minute sampling
- 1 year of data





SCSF Applications

- Clearness index/clear data filtering
- Baseline estimation for statistical forecasting
- Shading and soiling analysis
- Degradation analysis



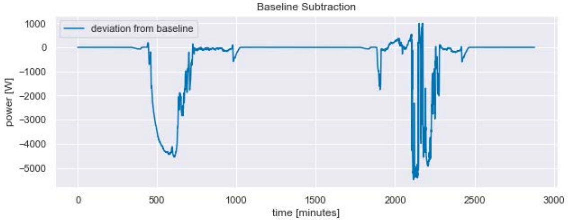
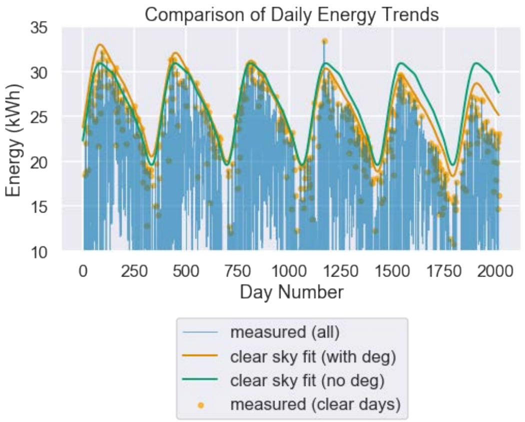


Table of Contents

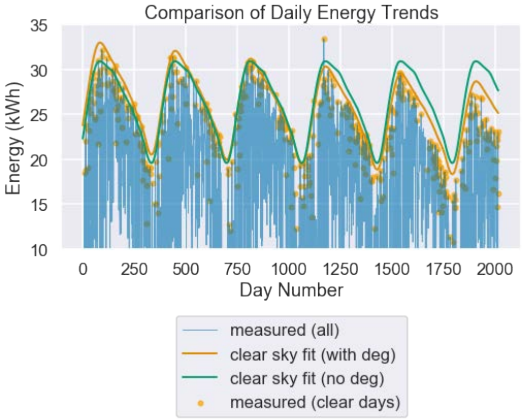
The SCSF Model Must Include a Degradation Term

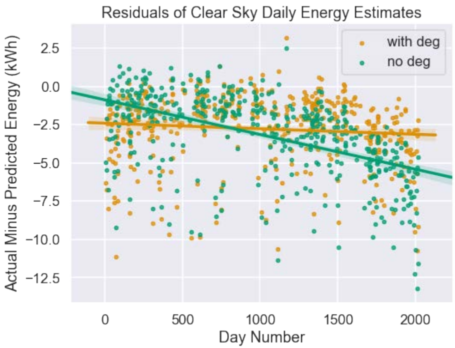
Strictly periodic models do a poor job
of fitting real-world, multi-year data sets.



The SCSF Model Must Include a Degradation Term

Strictly periodic models do a poor job
of fitting real-world, multi-year data sets.





Fitting the Degradation Term

- We model the degradation term as a year-over-year (YOY) percent change in energy output

$$\frac{d_{i+365} - d_i}{d_i} = \beta, \quad \text{for } i = 1 \ldots T - 365.$$

- We include a constraint on the SCSF model that all pairs of days must have the same YOY value
- This makes the math difficult $\rightarrow$ the paper[4] goes into the details of how this is handled

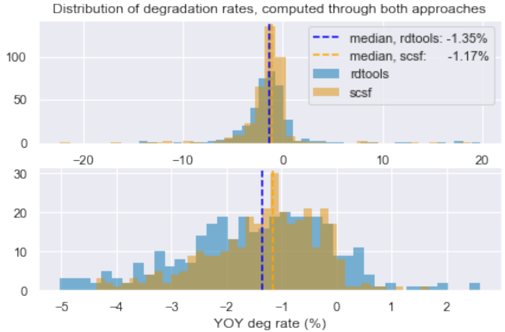
Python Software

Functionality is included in `statistical-clear-sky`.

[4] B. Meyers, M. Deceglie, C. Deline, and D. Jordan, "Signal Processing on PV Time-Series Data: Robust Degradation Analysis without Physical Models," *IEEE J-PV*, 2019.

Discussion

- See *J-PV* paper[5] for validation—worked with NREL on comparison to `RdTools`
- This unsupervised machine learning approach does not require models of the sites nor irradiance or meteorological data
- SCSF lends itself naturally to fleet-scale analysis of heterogeneous systems, where such supplementary data may be missing or incorrect
- Can additionally analyze irradiance signals to estimate sensor drift



[5] B. Meyers, M. Deceglie, C. Deline, and D. Jordan, "Signal Processing on PV Time-Series Data: Robust Degradation Analysis without Physical Models," *IEEE J-PV*, 2019.

Conclusion

- We're developing software tools to enable fleet-scale analysis of distributed rooftop solar PV systems
- We hope to show that power signals are very useful by themselves for digital O&M
- What I've shown here is an introduction to the problems we're solving
- Additional research includes:
  - Power signal clustering for shading analysis
  - Soiling estimation and additional system loss factors
  - System location and orientation estimation from power data
  - DuraMAT PV-Pro project (led by LBL)
- Find our code on GitHub, PyPI, and Anaconda
  - `solar-data-tools`
  - `statistical-clear-sky`
  - More to come!

Acknowledgments

Thank you so much to

- Our industry data sharing partners at SunPower, DNV GL, and Envision Digital
- NREL for sharing data and collaborating on the degradation research
  - Chris Deline, Mike Deceglie, Dirk Jordan
- The rest of our industry technical advisory committee
- Prof. Stephen Boyd at Stanford for his guidance on OSD and GLRM
- My colleagues in the Grid Integration Systems and Mobility (GISMo) team at SLAC
  - Laura Schelhas, Elpiniki Apostolaki-Iosifidou, Jonathan Goncalves, et al.

This material is based upon work supported by the U.S. Department of Energys Office of Energy Efficiency and Renewable Energy (EERE) under Solar Energy Technologies Office (SETO) Agreement Numbers 34911 ("PVInsight"), 30311, and 34348.

SOLAR ENERGY TECHNOLOGIES OFFICE U.S. Department Of Energy

bennetm@stanford.edu (email), @BennetMeyers (Twitter), bmeyers (GitHub)