

Tools for PV Data Science

Applied math, statistics, and signal processing
for gaining insight from PV data

Bennet Meyers

Stanford University, Electrical Engineering Dept.
Grid Integration Systems and Mobility (GISMo) Group at SLAC

February 2019



Automating PV Data Analytics

Framing question for this talk: How might we **robustly automate** the process of **estimating the degradation** rate of an installed PV system when we have no model of the system nor correlated irradiance or meteorological data?

Automating PV Data Analytics

Framing question for this talk: How might we **robustly automate** the process of **estimating the degradation** rate of an installed PV system when we have no model of the system nor correlated irradiance or meteorological data?

- We'll explore four different methods that solve common **PV data science tasks**.
- These tasks will built on each other to achieve the goal above
- But the tasks are useful interesting in their own right as well!

Background: More Data, More Opportunities

- Increasing volume of PV system performance data creates opportunities for **monitoring system health** and **optimizing O&M activities**.
- 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, most PV system performance engineering work is focused on **utility scale power plants**...



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

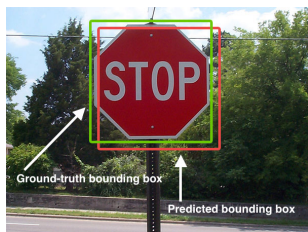


How can we deal with all this data?

Today is a heyday of data and applied mathematics...

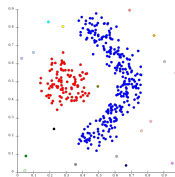
- *Probability theory*: network traffic management, genomics research, advertisement click-through optimization
- *Signal processing*: computer vision systems, wearable devices, GPS
- *Statistics*: image classification, clustering, voice recognition

And much more.



Object detection for a stop sign

by Adrian Rosebrock [CC BY-SA 4.0]



Clustering example

by Chire [CC BY-SA 3.0]

Introducing a new toolbox for PV data science

Today, we'll be looking at some concepts from applied math and exploring how we can use them to tackle some common tasks in a PV data analysis workflow.

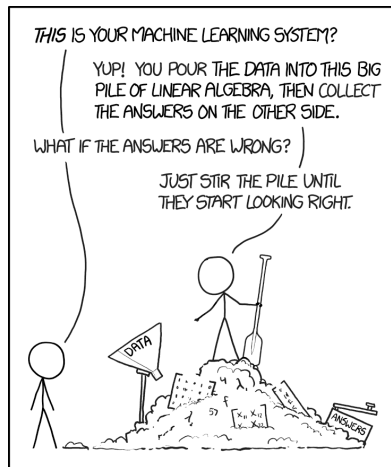
Tools in the toolbox

Matrix embeddings (*linear algebra*), Quantile regression (*statistics*), Smoothness metrics (*linear algebra*), Total variation filtering (*signal processing*), KDE clustering (*statistics*) Generalized low-rank modeling (*linear algebra*)

Everything presented here is implemented in BSD 2.0 open-source Python software:

- <https://github.com/bmeyers/solar-data-tools>
- <https://github.com/bmeyers/StatisticalClearSky>

Is this *Machine Learning*?



"Machine Learning"

by Randall Munrow [CC BY-NC 2.5]

Is this *Machine Learning*?

- Yes, generally this is all under the umbrella of **unsupervised machine learning**, the sense that we are learning from data that has not been labeled.
- Also very much in the category of **data science**.
- But this has a unique spin to it: We are generally going to be exploiting the *time structure* of the data when looking for patterns and clusters.
- **But**, we could just as easily classify the techniques according to more classical domains like statistics and signal processing.
- This is not *deep* ML. We are not employing neural networks.
- In fact, our mathematic workhorse will be **convex optimization** implemented in `cvxpy`¹ rather than **gradient descent**.


¹Python-embedded modeling language for convex optimization problems 

Table of Contents

- 1 The PV Power Matrix
- 2 Clear Day Detector
- 3 Time Shift Fixing
- 4 Clear Sky Signals and Degradation

Table of Contents

- 1 The PV Power Matrix
- 2 Clear Day Detector
- 3 Time Shift Fixing
- 4 Clear Sky Signals and Degradation

Data Preprocessing

- First, extract data from some source (database, csv files, etc.)
- Second, put data in **usable form** (extractions, parsing, joining, standardizing, augmenting, cleansing, consolidating and/or filtering)

Data Preprocessing

- First, extract data from some source (database, csv files, etc.)
- Second, put data in **usable form** (extractions, parsing, joining, standardizing, augmenting, cleansing, consolidating and/or filtering)

Matrix Representation

We're focusing on one particularly useful data transformation: putting a time-series power signal in a matrix.

Matrix Embedding

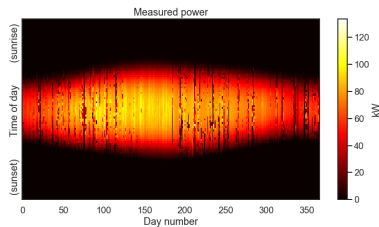
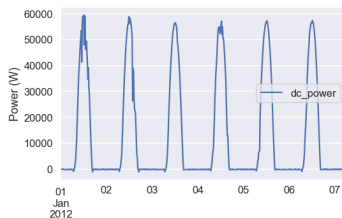
Embedding a PV power time series signal in a *matrix* is a convenient and powerful way to handle large amounts of data.

$$p \in \mathbf{R}^T \quad D = \begin{bmatrix} p_1 & p_{m+1} & \cdots & p_{(n-1) \cdot m + 1} \\ p_2 & p_{m+2} & \cdots & p_{(n-1) \cdot m + 2} \\ \vdots & \vdots & \ddots & \vdots \\ p_m & p_{2m} & \cdots & p_T \end{bmatrix} \in \mathbf{R}^{m \times n}$$

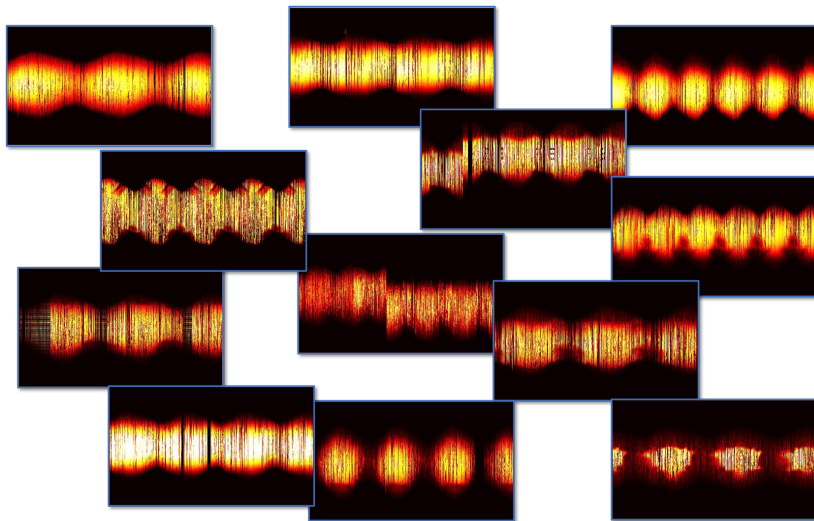
- m : number of measurements per day
- n : number of days in the data set
- $T = m \times n$: total number of measurements
- Called a **design matrix** in classic statistics

Matrix Embedding

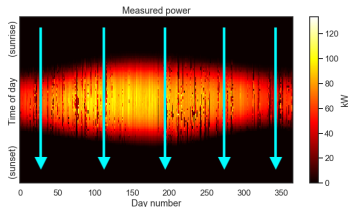
Embedding a PV power time series signal in a *matrix* is a convenient and powerful way to handle large amounts of data.



Powerful visualization tool

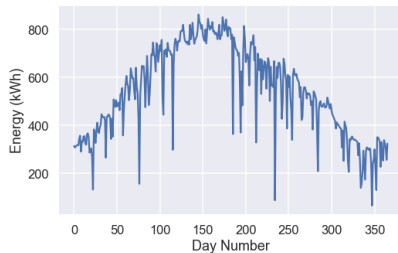


Matrix embedding makes basic operations very easy!

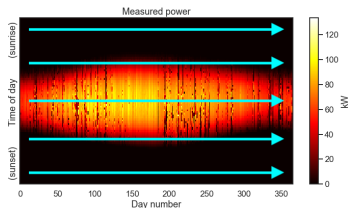


- Daily energy calculation is a column sum (with a scale factor)

```
daily_energy = c * np.sum(D, axis=0)
```

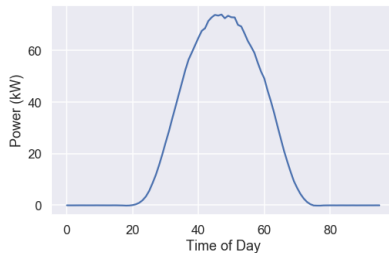


Matrix embedding makes basic operations very easy!

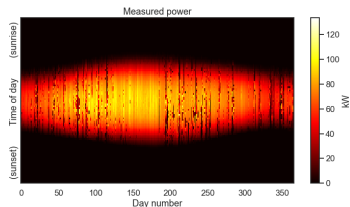


- Daily energy calculation is a column sum (with a scale factor)
- Expected power over a single day is a row average

```
exp_power = np.average(D, axis=1)
```

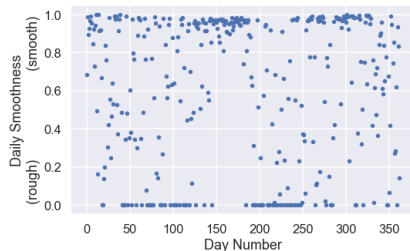


Matrix embedding makes basic operations very easy!



- Daily energy calculation is a column sum (with a scale factor)
- Expected power over a single day is a row average
- Other daily statistics are easily calculated, e.g. “smoothness,” a proxy for cloudiness

Daily Smoothness (more on this later)



Practical Considerations

```
In [2]: req_params = {
        'api_key': 'DEMO_KEY',
        "system_id": 1199,
        "year": "2016"
      }
base_url = 'https://developer.nrel.gov/api/pvdaq/v3/data_file?'
param_list = [str(item[0]) + '=' + str(item[1]) for item in req_params.items()]
req_url = base_url + '&'.join(param_list)
response = requests.get(req_url)
df = pd.read_csv(StringIO(response.text))
df.head()
```

Out[2]:

	SiteID	Date-Time	ac_power	dc_power	inv1_ac_power	inv1_dc_current	inv1_dc_power	inv1_dc_voltage	inv1_temp	inv2_ac_power	...	inv6_ac_power	inv
0	1199	2016-01-01 07:40:40	NaN	NaN	0.000000	0.000000	0.000000	274.000000	20.700000	0.000	...	0.000000	
1	1199	2016-01-01 07:45:41	0.000000	0.000000	0.000000	0.000000	0.000000	294.500000	20.700000	0.000	...	0.000000	
2	1199	2016-01-01 07:50:41	0.000000	0.000000	0.000000	0.000000	0.000000	288.250000	20.700000	0.000	...	0.000000	
3	1199	2016-01-01 07:55:41	18.982143	99.149174	3.285714	0.056714	17.459810	307.857143	20.785714	5.375	...	2.571429	
4	1199	2016-01-01 08:00:41	207.750000	519.652969	27.375000	0.266625	69.389156	260.250000	21.300000	32.500	...	27.625000	

5 rows x 39 columns

Practical Considerations

```
In [2]: req_params = {
        'api_key': 'DEMO_KEY',
        "system_id": 1199,
        "year": "2016"
      }
        base_url = 'https://developer.nrel.gov/api/pvdaq/v3/data_file?'
        param_list = [str(item[0]) + '=' + str(item[1]) for item in req_params.items()]
        req_url = base_url + '&'.join(param_list)
        response = requests.get(req_url)
        df = pd.read_csv(StringIO(response.text))
        df.head()
```

Out[2]:

	SiteID	Date-Time	ac_power	dc_power	inv1_ac_power	inv1_dc_current	inv1_dc_power	inv1_dc_voltage	inv1_temp	inv2_ac_power	...	inv6_ac_power	inv
0	1199	2016-01-01 07:40:40	NaN	NaN	0.000000	0.000000	0.000000	274.000000	20.700000	0.000	...	0.000000	
1	1199	2016-01-01 07:45:41	0.000000	0.000000	0.000000	0.000000	0.000000	294.500000	20.700000	0.000	...	0.000000	
2	1199	2016-01-01 07:50:41	0.000000	0.000000	0.000000	0.000000	0.000000	288.250000	20.700000	0.000	...	0.000000	
3	1199	2016-01-01 07:55:41	18.982143	99.149174	3.285714	0.056714	17.459810	307.857143	20.785714	5.375	...	2.571429	
4	1199	2016-01-01 08:00:41	207.750000	519.652969	27.375000	0.266625	69.389156	260.250000	21.300000	32.500	...	27.625000	

5 rows x 39 columns

Practical Considerations

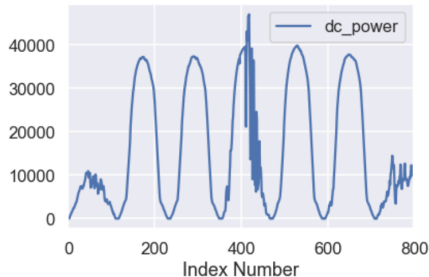
```
In [2]: req_params = {
        'api_key': 'DEMO_KEY',
        "system_id": 1199,
        "year": "2016"
      }
      base_url = 'https
      param_list = [str
      req_url = base_ur
      response = request
      df = pd.read_csv(
      df.head()
```

Out[2]:

	SiteID	Date-Time
0	1199	2016-01-01 07:40:40
1	1199	2016-01-01 07:45:41
2	1199	2016-01-01 07:50:41
3	1199	2016-01-01 07:55:41
4	1199	2016-01-01 08:00:41

5 rows x 39 columns

```
In [4]: df.iloc[:800].plot(y='dc_power')
      plt.xlabel('Index Number');
```



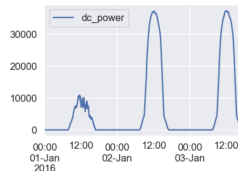
r6_ac_power	inv
0.000000	
0.000000	
0.000000	
2.571429	
27.625000	

Practical Considerations

- <https://github.com/bmeyers/solar-data-tools> has functions available to help with prepping data to make a PV power matrix.
- `standardize_time_axis` will attempt to “fix” any inconsistencies in the time stamps.
- A “standard” time axis is one where there is a constant time interval between consecutive entries—evenly spaced measurements.

```
In [5]: dfs = standardize_time_axis(df, datetimekey='Date-Time')
```

```
In [6]: dfs.iloc[:800].plot(y='dc_power');
```



```
In [7]: D = make_2d(dfs)
```

```
In [8]: plot_2d(D/1000);
```

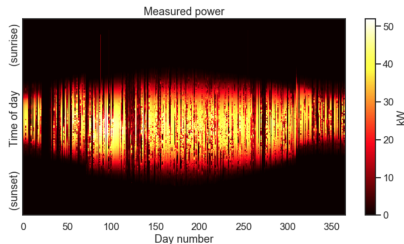
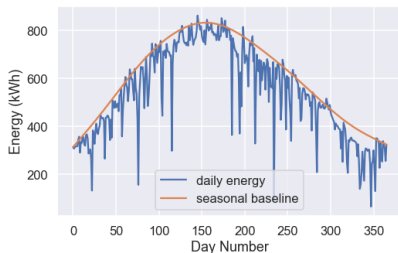


Table of Contents

- 1 The PV Power Matrix
- 2 Clear Day Detector
- 3 Time Shift Fixing
- 4 Clear Sky Signals and Degradation

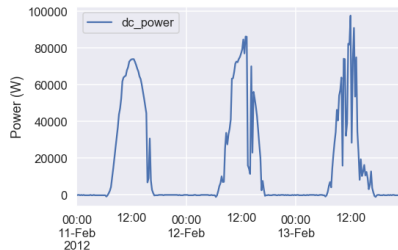
How to quickly find clear days in the data set?

Daily energy content



- Clear days have more energy relative to seasonal baseline
- Some high energy days can 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*

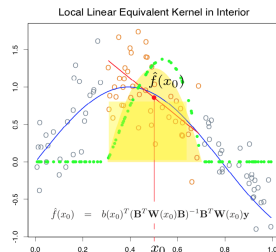
Local Quantile Regression

LQR is a combination of *local regression* and *quantile regression*.

Local Quantile Regression

LQR is a combination of *local regression* and *quantile regression*.

- *Local regression* fits a function to the data within a kernel or window



The Elements of Statistical Learning, p. 196,
by Hastie, Tibshirani, and Friedman

Local Quantile Regression

LQR is a combination of *local regression* and *quantile regression*.

- *Local regression* fits a function to the data within a kernel or window
- This can be recast as a general convex optimization problem

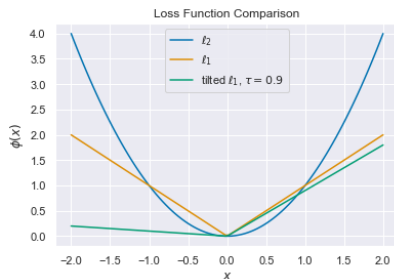
$$\|\hat{x} - x_{\text{cor}}\|_2^2 + \delta \|D\hat{x}\|_2^2$$

See §6.3.3 “Reconstruction, smoothing, and de-noising” in *Convex Optimization* by Boyd and Vandenberghe

Local Quantile Regression

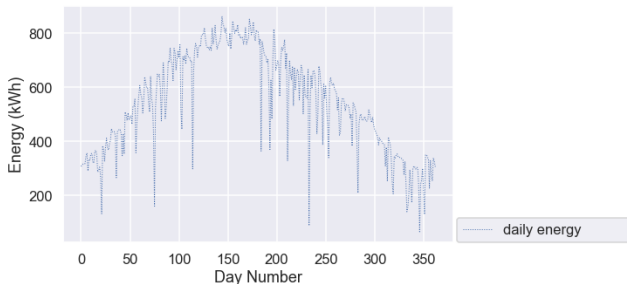
LQR is a combination of *local regression* and *quantile regression*.

- *Local regression* fits a function to the data within a kernel or window
- This can be recast as a general convex optimization problem
- And then replace the ℓ_2 cost function with the **tilted ℓ_1 penalty** to estimate the local quantile



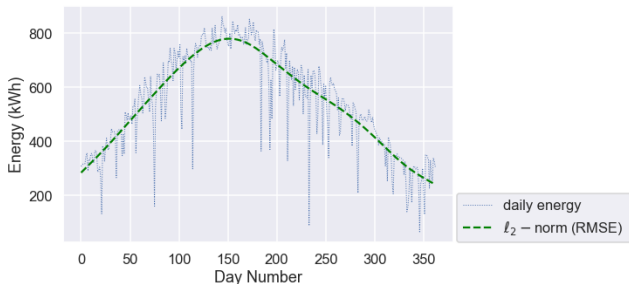
$$\phi_{\tau}(r) = \tau(x)_{+} + (1 - \tau)(x)_{-} = \frac{1}{2}|x| + \left(\tau - \frac{1}{2}\right)x$$

Local Quantile Regression



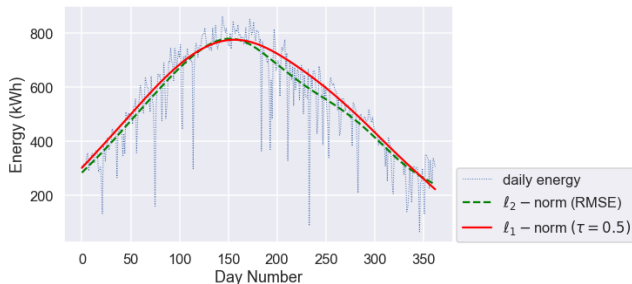
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



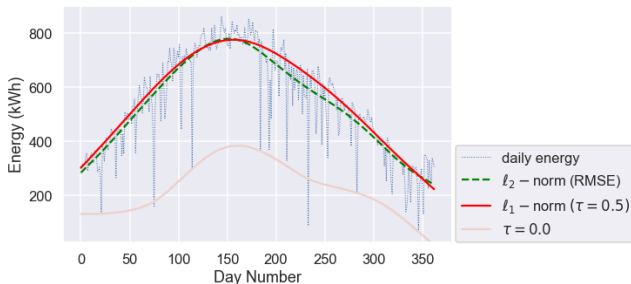
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



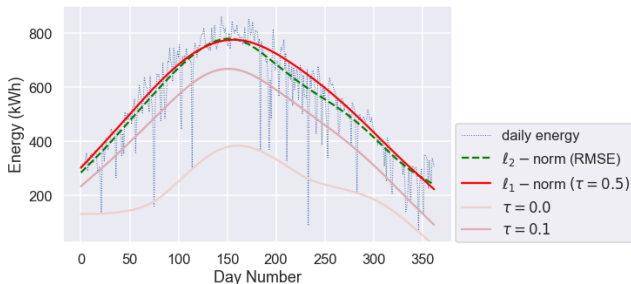
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



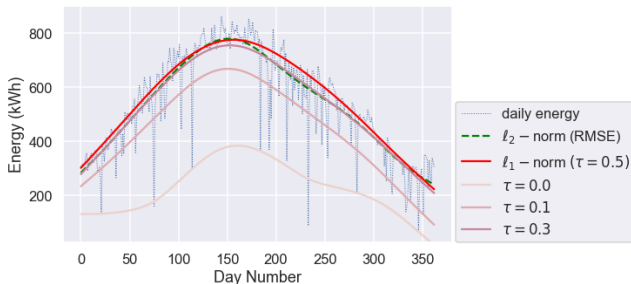
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



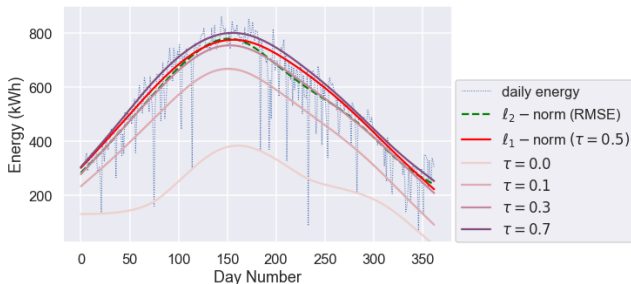
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



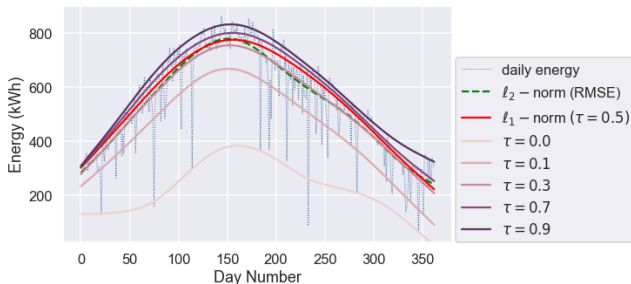
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

Local Quantile Regression



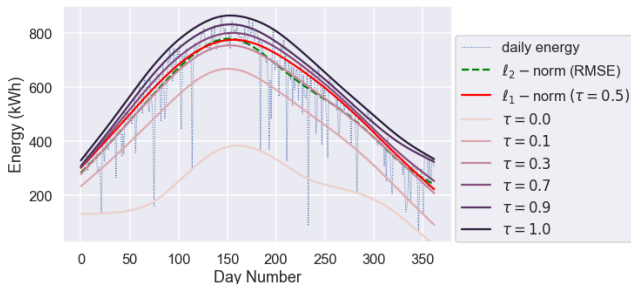
- ℓ_2 norm fits the local average and ℓ_1 norm fits the local (approximate) median
- τ sweeps through the local (approximate) percentiles of the data
- $\tau = 0.9$ works best for the clear day baseline: upper envelope fit with a little permeability

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

The *discrete difference* is the analogue of derivatives for discrete signals. It is a linear transformation (like derivatives), which means it is representable as a matrix operator. The first order difference is:

$$d^{(1)}[n] = x[n+1] - x[n] \implies d^{(1)} = \mathcal{D}x \quad (1)$$

where $\mathcal{D} \in \mathbf{R}^{(n-1)} \times n$ is the bidiagonal matrix

$$\mathcal{D} = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \quad (2)$$

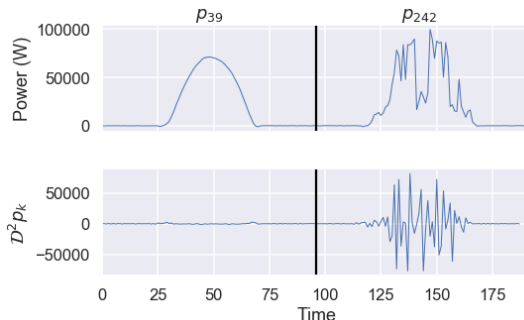
Higher order differences are constructed by repeated application of the first operator, exactly like derivatives: $d^{(2)} = \mathcal{D}^2x$.

Discrete Differences and Smoothness

- Our smoothness metric:

$$s_k = \|\mathcal{D}^2 p_k\|_2$$

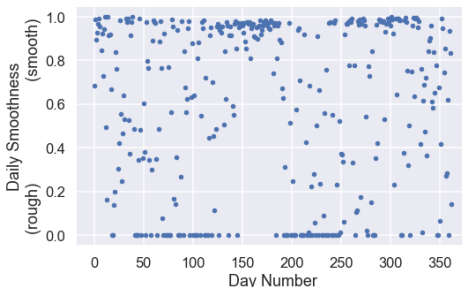
- $p_k \in \mathbf{R}^m$ is the k^{th} column of the PV power matrix
- $\mathcal{D}^2 p_k$ measures the local “curvature” of the signal



Discrete Differences and Smoothness

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

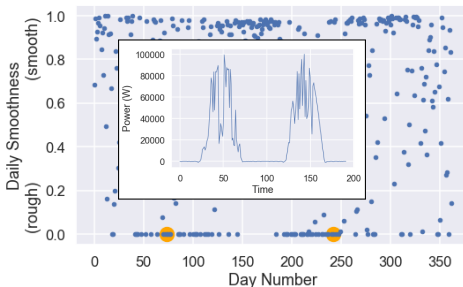
```
In [95]: 1 smoothness = np.linalg.norm(np.diff(D, n=2, axis=0), axis=0)
          2 smoothness -= np.quantile(smoothness, 0.8)
          3 smoothness /= np.min(smoothness)
          4 smoothness = np.clip(smoothness, 0, 1)
          5 plt.figure(figsize=(8, 5))
          6 plt.plot(smoothness, marker='.',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

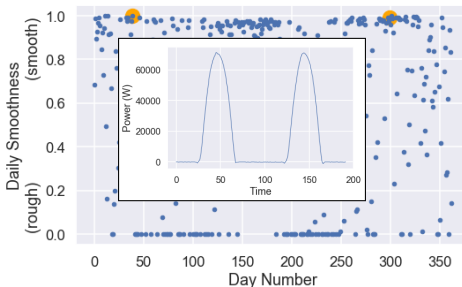
```
In [95]: 1 smoothness = np.linalg.norm(np.diff(D, n=2, axis=0), axis=0)
          2 smoothness -= np.quantile(smoothness, 0.8)
          3 smoothness /= np.min(smoothness)
          4 smoothness = np.clip(smoothness, 0, 1)
          5 plt.figure(figsize=(8, 5))
          6 plt.plot(smoothness, marker='.',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



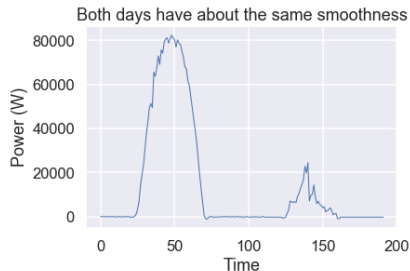
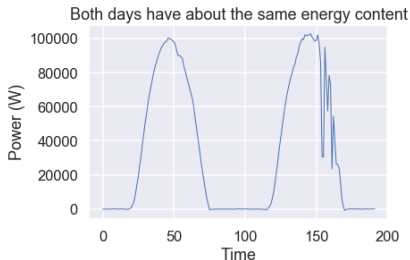
Discrete Differences and Smoothness

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

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

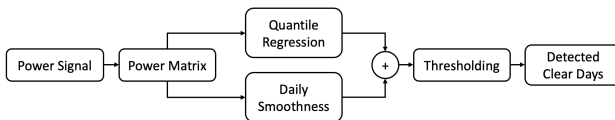


Both Methods Are Imperfect



- 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



- We can combine the two approaches to select for days that are both high-energy and smooth
- Check out `solardatatools.find_clear_days` for an implementation of this algorithm!

```
In [18]: dclear = find_clear_days(D)  
         plot_2d(D/1000, clear_days=dclear);
```

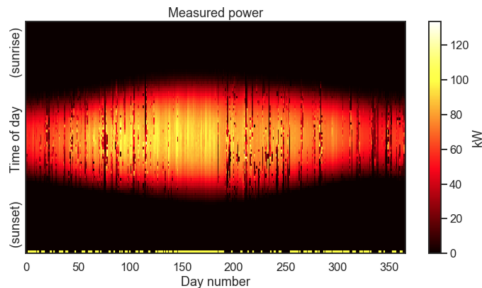
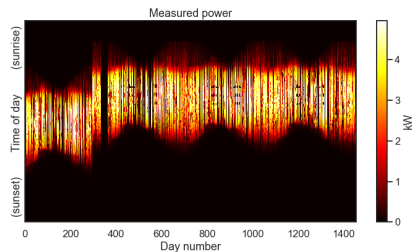
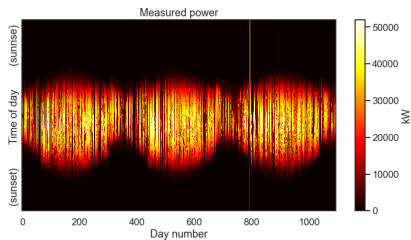


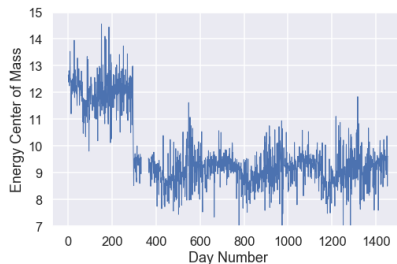
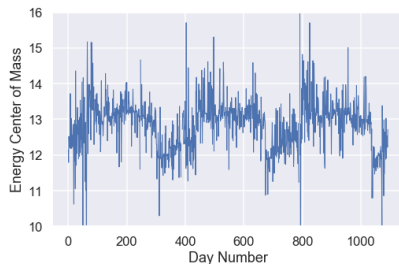
Table of Contents

- 1 The PV Power Matrix
- 2 Clear Day Detector
- 3 Time Shift Fixing**
- 4 Clear Sky Signals and Degradation

Time Shifts

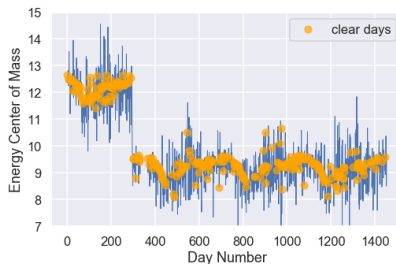
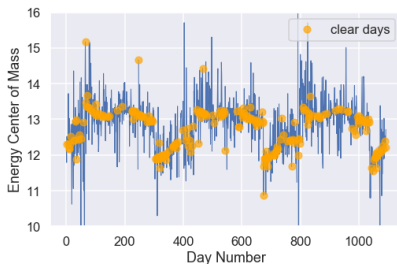


Time Shifts



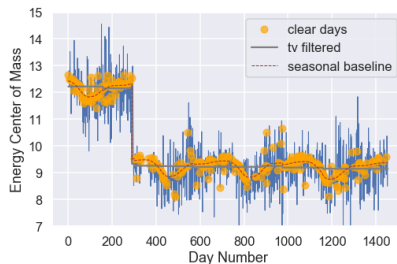
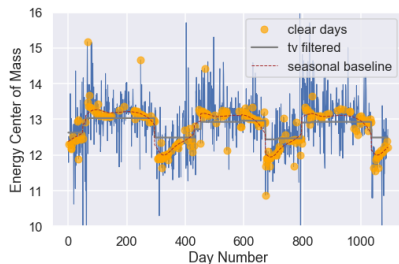
- Estimate solar noon on each day (energy center of mass)

Time Shifts



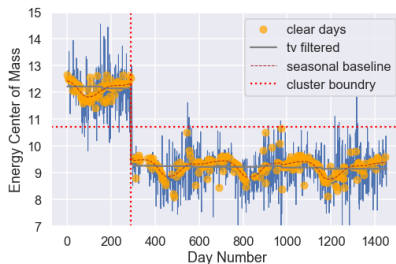
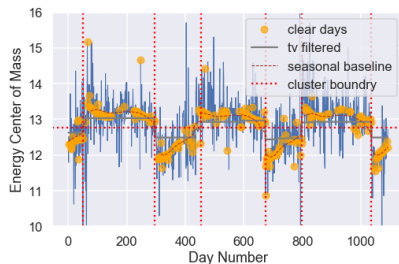
- Estimate solar noon on each day (energy center of mass)
- Filter for clear days using `solardatatools.find_clear_days`

Time Shifts



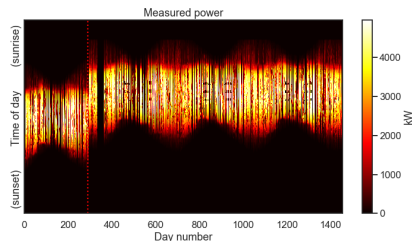
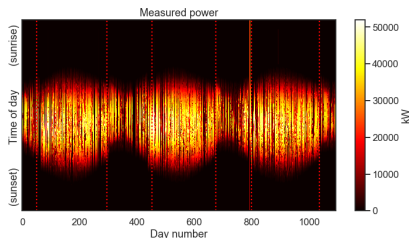
- Estimate solar noon on each day (energy center of mass)
- Filter for clear days using `solardatatools.find_clear_days`
- Total variation filter with seasonal baseline fit (signal separation)

Time Shifts



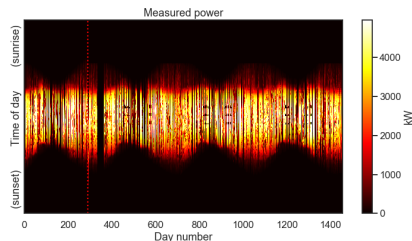
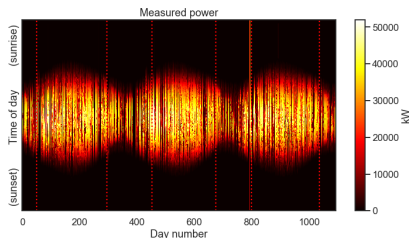
- Estimate solar noon on each day (energy center of mass)
- Filter for clear days using `solardatatools.find_clear_days`
- Total variation filter with seasonal baseline fit (signal separation)
- Kernel density estimation (KDE) clustering

Time Shifts



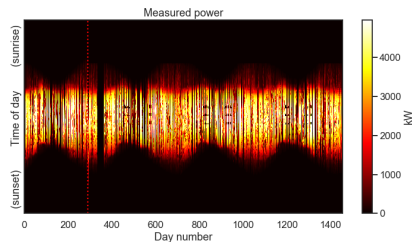
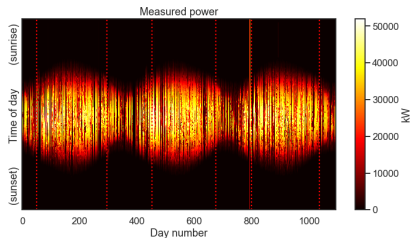
- Estimate solar noon on each day (energy center of mass)
- Filter for clear days using `solardatatools.find_clear_days`
- Total variation filter with seasonal baseline fit (signal separation)
- Kernel density estimation (KDE) clustering
- Fix the shifts!

Time Shifts



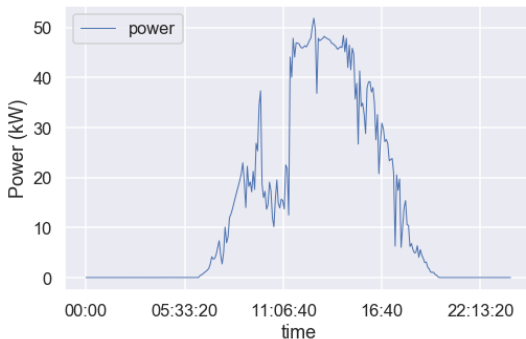
- Estimate solar noon on each day (energy center of mass)
- Filter for clear days using `solardatatools.find_clear_days`
- Total variation filter with seasonal baseline fit (signal separation)
- Kernel density estimation (KDE) clustering
- Fix the shifts!

Time Shifts

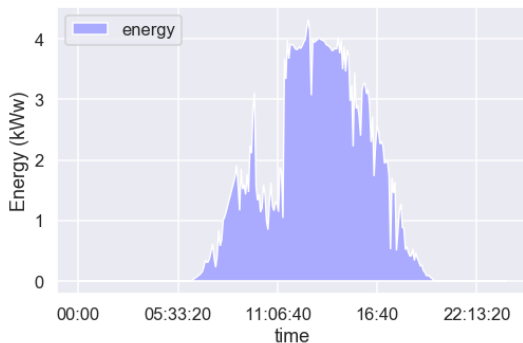


- Estimate solar noon on each day (energy center of mass)
- Filter for clear days using `solardatatools.find_clear_days`
- Total variation filter with seasonal baseline fit (signal separation)
- Kernel density estimation (KDE) clustering
- Fix the shifts!
- `solardatatools.fix_time_shifts`

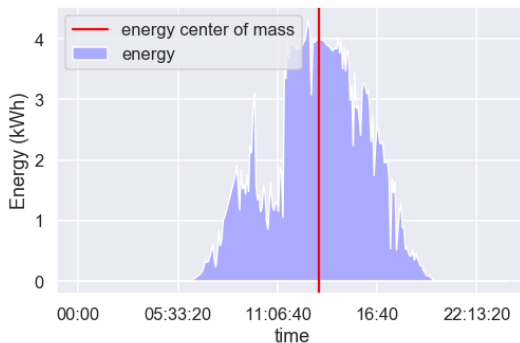
Solar Noon Estimate, a.k.a Energy Center of Mass



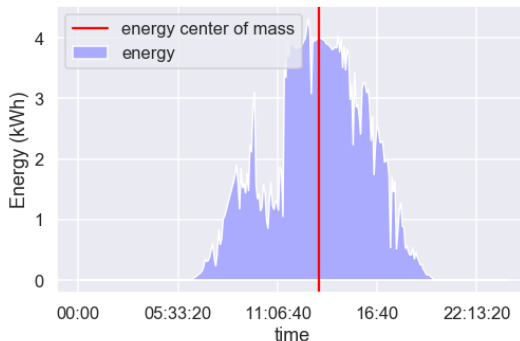
Solar Noon Estimate, a.k.a Energy Center of Mass



Solar Noon Estimate, a.k.a Energy Center of Mass



Solar Noon Estimate, a.k.a Energy Center of Mass



```
def energy_com(D):  
    div1 = np.dot(np.linspace(0, 24, D.shape[0]), D)  
    div2 = np.sum(D, axis=0)  
    s1 = np.empty_like(div1)  
    s1[:] = np.nan  
    msk = div2 != 0  
    s1[msk] = np.divide(div1[msk], div2[msk])  
    return s1
```

Total Variation Filtering with Seasonal Baseline Fitting

$$\underset{x,y,z}{\text{minimize}} \quad \phi_{\text{tv}}(x) + \phi_{\text{smooth}}(y) + \phi_{\text{huber}}(z)$$

$$\text{subject to} \quad s = x + y + z$$

$$y_i = y_{i+365}, \quad \sum_{i=1}^{365} y_i = 0$$

where s is the measured signal. We model this signal as the sum of three unseen signals: x a signal with low *total variation*, y a smooth signal that is 365-day periodic and sums to zero in a period, and z an error term.

$$\phi_{\text{tv}}(x) = \mu_1 \|\mathcal{D}x\|_1, \quad \phi_{\text{smooth}}(x) = \mu_2 \|\mathcal{D}^2x\|_2$$

$$\phi_{\text{huber}}(x) = \begin{cases} x^2 & \text{for } |x| \leq 1 \\ 2|x| - 1 & \text{otherwise} \end{cases}$$

Total Variation Filtering with Seasonal Baseline Fitting

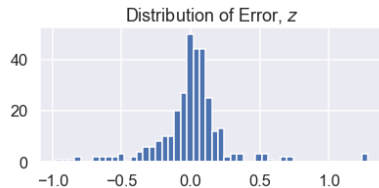
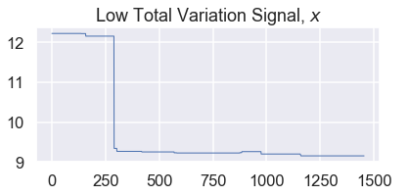
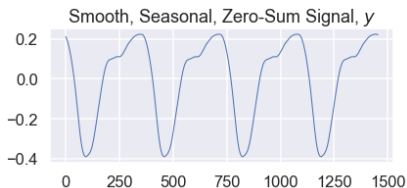
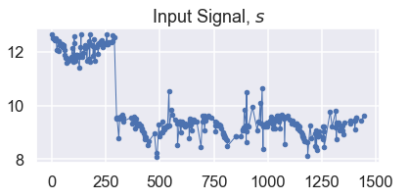
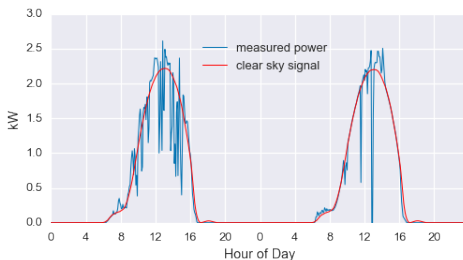


Table of Contents

- 1 The PV Power Matrix
- 2 Clear Day Detector
- 3 Time Shift Fixing
- 4 Clear Sky Signals and Degradation

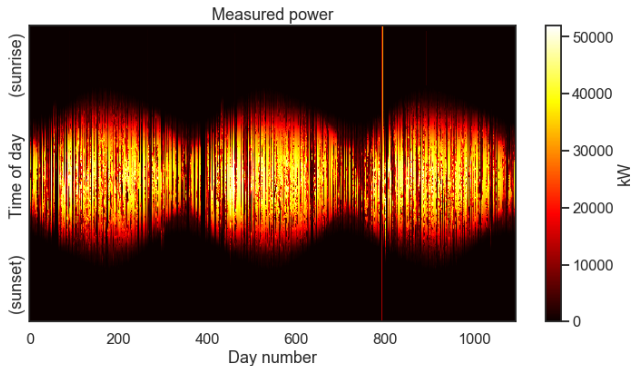
Statistical Clear Sky Fitting (SCSF)²

- Putting it all together: **robust, accurate, fully automatic PV system degradation estimation**
- SCSF fits a site-tuned, data-driven clear sky model without using classic, physical models
- No site information, no on-site data collection besides an inverter
- Makes use of everything so far plus *Generalized Low Rank Modeling*



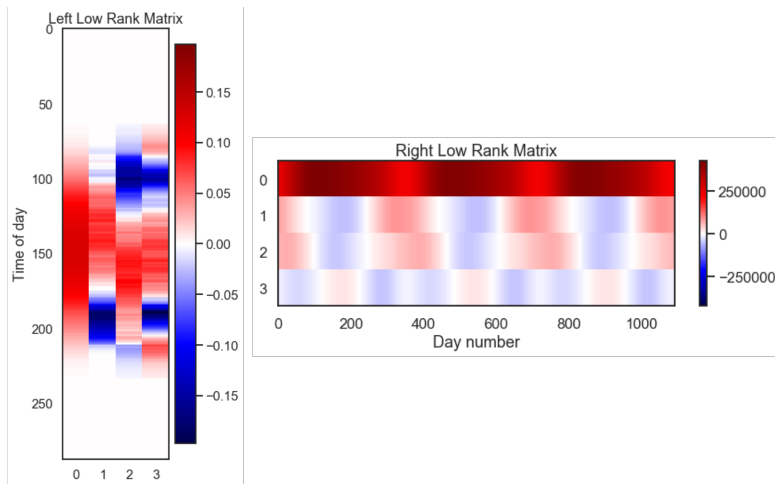
²"Statistical Clear Sky Fitting," Best Student Paper Area 8, PVSC45/WGPEC7

How Does SCSF Work?



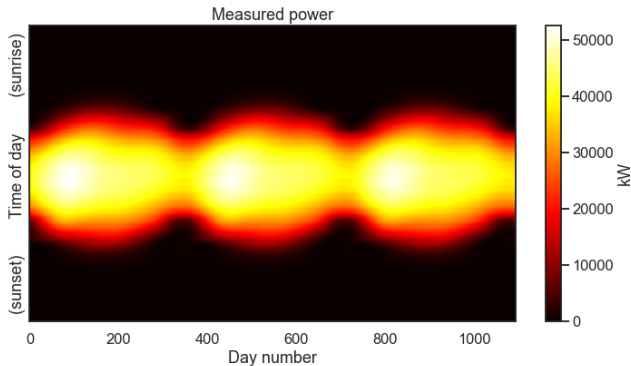
Starting with the PV power matrix...

How Does SCSF Work?



Find a low rank approximation...

How Does SCSF Work?



That estimates the clear sky signal when multiplied back together!

The Math of SCSF: Generalized Low Rank Modeling

Find a signal that

$$\begin{aligned}
 &\underset{L, R, \beta}{\text{minimize}} && \phi_{\tau}((D - LR) \mathbf{diag}(w)) + \mu_L \|\mathcal{D}^2 L\|_F \\
 &&& + \mu_R \|\mathcal{D}^2 R^T\|_F + \mu_R \|\mathcal{D}_{(1,365)} \tilde{R}^T\|_F \\
 &\text{subject to} && LR \geq 0 \\
 &&& \mathbf{1}^T \ell_j = 0, \quad j = 2, \dots, k \\
 &&& L_{i,j} = 0, \quad i \in \mathcal{Z}, j = 1, \dots, k \\
 &&& \beta = \frac{R_{1,j+365} - R_{1,j}}{R_{1,j}}, \quad i = 1, 2, \dots, n - 365
 \end{aligned}$$

The Math of SCSF: Generalized Low Rank Modeling

Find a signal that (1) is low-rank and close to the observed data,

$$\begin{aligned}
 &\underset{L, R, \beta}{\text{minimize}} \quad \phi_{\tau}((D - LR) \text{diag}(w)) + \mu_L \|\mathcal{D}^2 L\|_F \\
 &\quad + \mu_R \|\mathcal{D}^2 R^T\|_F + \mu_R \|\mathcal{D}_{(1,365)} \tilde{R}^T\|_F \\
 &\text{subject to} \quad LR \geq 0 \\
 &\quad \mathbf{1}^T \ell_j = 0, \quad j = 2, \dots, k \\
 &\quad L_{i,j} = 0, \quad i \in \mathcal{Z}, j = 1, \dots, k \\
 &\quad \beta = \frac{R_{1,j+365} - R_{1,j}}{R_{1,j}}, \quad i = 1, 2, \dots, n - 365
 \end{aligned}$$

The Math of SCSF: Generalized Low Rank Modeling

Find a signal that (1) is low-rank and close to the observed data, (2) is smooth each day,

$$\begin{aligned}
 &\underset{L, R, \beta}{\text{minimize}} && \phi_{\tau}((D - LR) \text{diag}(w)) + \mu_L \|\mathcal{D}^2 L\|_F \\
 &&& + \mu_R \|\mathcal{D}^2 R^T\|_F + \mu_R \|\mathcal{D}_{(1,365)} \tilde{R}^T\|_F \\
 &\text{subject to} && LR \geq 0 \\
 &&& \mathbf{1}^T \ell_j = 0, \quad j = 2, \dots, k \\
 &&& L_{i,j} = 0, \quad i \in \mathcal{Z}, j = 1, \dots, k \\
 &&& \beta = \frac{R_{1,j+365} - R_{1,j}}{R_{1,j}}, \quad i = 1, 2, \dots, n - 365
 \end{aligned}$$

The Math of SCSF: Generalized Low Rank Modeling

Find a signal that (1) is low-rank and close to the observed data, (2) is smooth each day, (3) changes slowly from day to day,

$$\begin{aligned}
 &\underset{L, R, \beta}{\text{minimize}} && \phi_{\tau}((D - LR) \text{diag}(w)) + \mu_L \|\mathcal{D}^2 L\|_F \\
 &&& + \mu_R \|\mathcal{D}^2 R^T\|_F + \mu_R \|\mathcal{D}_{(1,365)} \tilde{R}^T\|_F \\
 &\text{subject to} && LR \geq 0 \\
 &&& \mathbf{1}^T \ell_j = 0, \quad j = 2, \dots, k \\
 &&& L_{i,j} = 0, \quad i \in \mathcal{Z}, j = 1, \dots, k \\
 &&& \beta = \frac{R_{1,j+365} - R_{1,j}}{R_{1,j}}, \quad i = 1, 2, \dots, n - 365
 \end{aligned}$$

The Math of SCSF: Generalized Low Rank Modeling

Find a signal that (1) is low-rank and close to the observed data, (2) is smooth each day, (3) changes slowly from day to day, (4) has a *shape* that is 365-day periodic,

$$\begin{aligned}
 &\underset{L, R, \beta}{\text{minimize}} && \phi_{\tau}((D - LR) \text{diag}(w)) + \mu_L \|D^2 L\|_F \\
 &&& + \mu_R \|D^2 R^T\|_F + \mu_R \|D_{(1,365)} \tilde{R}^T\|_F \\
 &\text{subject to} && LR \geq 0 \\
 &&& \mathbf{1}^T \ell_j = 0, \quad j = 2, \dots, k \\
 &&& L_{i,j} = 0, \quad i \in \mathcal{Z}, j = 1, \dots, k \\
 &&& \beta = \frac{R_{1,j+365} - R_{1,j}}{R_{1,j}}, \quad i = 1, 2, \dots, n - 365
 \end{aligned}$$

The Math of SCSF: Generalized Low Rank Modeling

Find a signal that (1) is low-rank and close to the observed data, (2) is smooth each day, (3) changes slowly from day to day, (4) has a *shape* that is 365-day periodic, (5) is non-negative,

$$\begin{aligned}
 &\underset{L, R, \beta}{\text{minimize}} \quad \phi_{\tau}((D - LR) \mathbf{diag}(w)) + \mu_L \|D^2 L\|_F \\
 &\quad + \mu_R \|D^2 R^T\|_F + \mu_R \|D_{(1,365)} \tilde{R}^T\|_F \\
 &\text{subject to} \quad LR \geq 0 \\
 &\quad \mathbf{1}^T \ell_j = 0, \quad j = 2, \dots, k \\
 &\quad L_{i,j} = 0, \quad i \in \mathcal{Z}, j = 1, \dots, k \\
 &\quad \beta = \frac{R_{1,j+365} - R_{1,j}}{R_{1,j}}, \quad i = 1, 2, \dots, n - 365
 \end{aligned}$$

The Math of SCSF: Generalized Low Rank Modeling

Find a signal that (1) is low-rank and close to the observed data, (2) is smooth each day, (3) changes slowly from day to day, (4) has a *shape* that is 365-day periodic, (5) is non-negative, (6) contains all *energy* in the first column of L ,

$$\begin{aligned}
 &\underset{L, R, \beta}{\text{minimize}} && \phi_{\tau}((D - LR) \mathbf{diag}(w)) + \mu_L \|\mathcal{D}^2 L\|_F \\
 &&& + \mu_R \|\mathcal{D}^2 R^T\|_F + \mu_R \|\mathcal{D}_{(1,365)} \tilde{R}^T\|_F \\
 &\text{subject to} && LR \geq 0 \\
 &&& \mathbf{1}^T \ell_j = 0, \quad j = 2, \dots, k \\
 &&& L_{i,j} = 0, \quad i \in \mathcal{Z}, j = 1, \dots, k \\
 &&& \beta = \frac{R_{1,j+365} - R_{1,j}}{R_{1,j}}, \quad i = 1, 2, \dots, n - 365
 \end{aligned}$$

The Math of SCSF: Generalized Low Rank Modeling

Find a signal that (1) is low-rank and close to the observed data, (2) is smooth each day, (3) changes slowly from day to day, (4) has a *shape* that is 365-day periodic, (5) is non-negative, (6) contains all *energy* in the first column of L , (7) is zero at night,

$$\begin{aligned}
 &\underset{L, R, \beta}{\text{minimize}} && \phi_{\tau}((D - LR) \mathbf{diag}(w)) + \mu_L \|\mathcal{D}^2 L\|_F \\
 &&& + \mu_R \|\mathcal{D}^2 R^T\|_F + \mu_R \|\mathcal{D}_{(1,365)} \tilde{R}^T\|_F \\
 &\text{subject to} && LR \geq 0 \\
 &&& \mathbf{1}^T \ell_j = 0, \quad j = 2, \dots, k \\
 &&& L_{i,j} = 0, \quad i \in \mathcal{Z}, j = 1, \dots, k \\
 &&& \beta = \frac{R_{1,j+365} - R_{1,j}}{R_{1,j}}, \quad i = 1, 2, \dots, n - 365
 \end{aligned}$$

The Math of SCSF: Generalized Low Rank Modeling

Find a signal that (1) is low-rank and close to the observed data, (2) is smooth each day, (3) changes slowly from day to day, (4) has a *shape* that is 365-day periodic, (5) is non-negative, (6) contains all *energy* in the first column of L , (7) is zero at night, and (8) has a single YoY daily energy degradation rate.

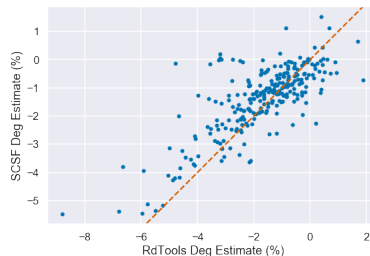
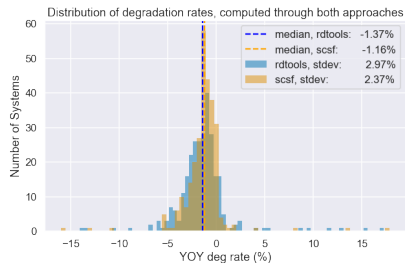
$$\begin{aligned}
 &\underset{L, R, \beta}{\text{minimize}} && \phi_{\tau}((D - LR) \mathbf{diag}(w)) + \mu_L \|\mathcal{D}^2 L\|_F \\
 &&& + \mu_R \|\mathcal{D}^2 R^T\|_F + \mu_R \|\mathcal{D}_{(1,365)} \tilde{R}^T\|_F \\
 &\text{subject to} && LR \geq 0 \\
 &&& \mathbf{1}^T \ell_j = 0, \quad j = 2, \dots, k \\
 &&& L_{i,j} = 0, \quad i \in \mathcal{Z}, j = 1, \dots, k \\
 &&& \beta = \frac{R_{1,j+365} - R_{1,j}}{R_{1,j}}, \quad i = 1, 2, \dots, n - 365
 \end{aligned}$$

Brief Notes on Implementation

- Problem is *non-convex* in two spots: the multiplication of L and R and the ratio definition of β .
- Solving this problem efficiently is the topic of the the *SCSF Algorithm* (see PVSC45 paper and new paper coming out this year)
- <https://github.com/bmeyers/StatisticalClearSky> implements the algorithm in a Python class structure
- Code has been deployed at scale—cluster of 21 Ubuntu machines analyzing >22GB of power data from hundreds of PV systems (completed in 1.5hrs)

NREL Collaboration: Validation of Degradation Estimate

- Working with NREL to compare SCSF to RdTools
- 278 PV systems from across the lower 48
- Median degradation rates agree to within 0.25%
- Standard deviation is 0.6% *smaller* for SCSF



In Conclusion

- Matrices are awesome
- Math is fun and useful
- But especially when combined with software!
 - <https://github.com/bmeyers/solar-data-tools>
 - <https://github.com/bmeyers/StatisticalClearSky>
- We can create robust, accurate, and automated data analysis pipelines
- Future work will explore basis representations for clustering and compression, estimating local system parameters, and automated loss factor analysis

Thank you!

Thank you to the GISMo team (past and present)
Laura Schelhas, Prof. Stephen Boyd



Chris Deline, Mike Deceglie, and Dirk Jordan

Questions?



This work is supported by DOE/SU Contract #3468, "PVInsight"