captest
Open Source Package for Reproducible Performance Testing

May 15, 2019 | Ben Taylor, Exyte Energy & Jessica Forbess, Sunshine Analytics
Agenda

- **Background – ASTM E2848**
- **Motivation**
  - Efficiency and Improved Workflow / Tools
  - Standardization of Calculations
  - Reproducible Test Calculations
- **Essential Functionality**
  - Package Structure
  - Load data
  - Identify Data to be used in regression
  - Visually review data
  - Aggregate Columns and/or Filter Columns
  - Filter rows
  - Perform regressions
  - Calculate reporting conditions
  - Results – summarize filtering steps
  - Results – compare regressions
- **Goals for Development**
  - Expand user base
  - Technical Review
  - OSS development and consensus building
  - New features, testing, bug squashing

[https://github.com/bt-/pvcaptest](https://github.com/bt-/pvcaptest)
Basis of ASTM E2848

Proving a constructed PV plant matches the model

- Multi-linear regression
  - AC Power ~ POA, ambient temperature, wind speed
  - Compare regressions of modeled and measured data to make sure they are within expected range
- Graph is rough visual, only shows POA correlation
- 11 page ASTM standard plus N (1-20) pages of test exhibit
- Complexity defining what data and test conditions are valid
Motivation

Efficiency and Improved Workflow
Motivation
Efficiency and Improved Workflow

Edit, Restart, & Run All

Using Bokeh and Holoviews Packages for Interactive Visualization
## Motivation

Standardization of Calculations

```python
In [16]: das.filter_irr(200, 2000)
```

```python
In [43]: pvc.get_summary(das, slm)
```

### Out[43]:

<table>
<thead>
<tr>
<th></th>
<th>filter_irr</th>
<th>filter_outliers</th>
<th>reg_cpt</th>
<th>rep_cond</th>
<th>filter_time</th>
<th>filter_pvsyst</th>
<th>filter Irr</th>
<th>reg_cpt</th>
</tr>
</thead>
<tbody>
<tr>
<td>das</td>
<td>1245</td>
<td>195</td>
<td>0</td>
<td>0</td>
<td>1441</td>
<td>397</td>
<td>284</td>
<td>264</td>
</tr>
<tr>
<td>custom_filter</td>
<td>1240</td>
<td>5</td>
<td>(function DataFrame.dropna at 0x0000000000000000)</td>
<td>0</td>
<td>0</td>
<td>1441</td>
<td>397</td>
<td>284</td>
</tr>
<tr>
<td>filter_irr</td>
<td>424</td>
<td>816</td>
<td>(200, 2000)</td>
<td>0</td>
<td>0</td>
<td>1441</td>
<td>397</td>
<td>284</td>
</tr>
<tr>
<td>filter_outliers</td>
<td>407</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>1441</td>
<td>397</td>
<td>284</td>
<td>264</td>
</tr>
<tr>
<td>reg_cpt</td>
<td>385</td>
<td>22</td>
<td>() filter: True summary: False</td>
<td>0</td>
<td>0</td>
<td>1441</td>
<td>397</td>
<td>284</td>
</tr>
<tr>
<td>rep_cond</td>
<td>385</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1441</td>
<td>397</td>
<td>284</td>
<td>264</td>
</tr>
<tr>
<td>filter_irr</td>
<td>294</td>
<td>91</td>
<td>(0.5, 1.5)</td>
<td>0</td>
<td>0</td>
<td>1441</td>
<td>397</td>
<td>284</td>
</tr>
<tr>
<td>reg_cpt</td>
<td>294</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1441</td>
<td>397</td>
<td>284</td>
<td>264</td>
</tr>
<tr>
<td>slm</td>
<td>1441</td>
<td>7319</td>
<td>0</td>
<td>0</td>
<td>1441</td>
<td>397</td>
<td>284</td>
<td>264</td>
</tr>
</tbody>
</table>

### Filter Implementation

```python
@updated_summary
def filter_irr(self, low, high, ref_val=None, col_name=None, inplace=True):
    """
    Filter on irradiance values.
    """
    if col_name is None:
        irr_col = self._get_poa_col()
    else:
        irr_col = col_name
    df_flt = flt Irr(self.df, irr_col, low, high, ref_val=ref_val)
    if inplace:
        self.df_flt = df_flt
    else:
        return df_flt
```

```python
def flt_irr(df, irr_col, low, high, ref_val=None):
    """
    Top level filter on irradiance values.
    """
    if ref_val is not None:
        low *= ref_val
        high *= ref_val
    df_renamed = df.rename(columns={irr_col: 'poa'})
    flt_str = '@low <= ' + 'poa' + ' <= @high'.
    index = df_renamed.query(flt_str).index
    return df.loc[index, :]
```
Standardization of Calculations

Complement ASTM E2848

- The ASTM standard defines a “Standard Test Method for Reporting Photovoltaic Non-Concentrator System Performance”
- One of the goals of the Captest package is to provide a standard set of functions/methods implementing the data manipulation, filtering, regressions, and visualization described in E2848
  - Pragmatically, will also need to allow for adjustments per exhibit language

Github as a platform for building consensus

- Capacity testing requires agreement between multiple parties – Developer / Owner / EPC / Financer
- If these parties agree on calculations within a standard open source library, the review of capacity test results should be much easier and more consistent project-to-project
Motivation
Reproducible Calculations

Capacity Test Reports in Jupyter Notebook

• Present data, visualizations, narrative text, images, and calculations together in a report that can be re-run.

Requirements to re-run notebook:

• Input data (.csv) measured & simulated
• Jupyter notebook (.ipynb) with analysis
• Environment with captest package and dependencies installed
Essential Functionality
Essential Functionality

1. Package structure
2. Load data – modeled or measured
   - Group data
3. Identify Data to be used in regression
4. Visually review data
5. Aggregate Columns and/or Filter Columns
6. Filter rows
7. Perform regressions
8. Calculate reporting conditions
9. Results – summarize filtering steps
10. Results – compare regressions

Link to live example.

https://github.com/bt-/pvcaptest
captest package

- captest module
  - CapData class
  - Capacity test functions

- future modules
  - NREL weather corrected PR test
Essential Functionality
Loading Data and Grouping Measurements

CapData.load_data()

- Two ways to load csv files:
  - Read a specific csv file – modeled data (tested on PVsyst) or measured data
  - Read all csv files in a directory – expects each file to have the same columns and appends rows

- Background tasks:
  - Group columns by measurement type and store in the ‘trans’ attribute as a dictionary
  - Copies data to df_flt attribute
  - If clear_sky is True, adds modeled clear sky GHI and POA columns to dataframe
Essential Functionality

Loading Data and Grouping Measurements

CapData.load_data()
- Calls CapData._CapData__set_trans
- irr : irradiance, irr, plane of array, poa, ghi, global, glob, w/m^2, w/m2, w/m, w/

<table>
<thead>
<tr>
<th>CapData.trans</th>
<th>irr-poa-</th>
<th>irr-ghi-</th>
<th>temp-amb-</th>
<th>temp-mod-</th>
<th>wind--</th>
<th>-mtr-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td>Weather Station 1 (Standard w/ POA GHI), Sun W/m^2</td>
<td>Weather Station 2 (Standard w/ POA GHI), Sun W/m^2</td>
<td>Weather Station 1 (Standard w/ POA GHI), TempF °F</td>
<td>Weather Station 2 (Standard w/ POA GHI), TempF °F</td>
<td>Weather Station 1 (Standard w/ POA GHI), WindSpe ed mph</td>
<td>Weather Station 2 (Standard w/ POA GHI), WindSpe ed mph</td>
</tr>
<tr>
<td>10/9/1990 0:00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.75067</td>
<td>15.64035</td>
<td>-0.00072</td>
</tr>
<tr>
<td>10/9/1990 0:05</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.73755</td>
<td>15.55192</td>
<td>-0.00888</td>
</tr>
<tr>
<td>10/9/1990 0:10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.64809</td>
<td>15.54152</td>
<td>0.000866</td>
</tr>
<tr>
<td>10/9/1990 0:15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.64177</td>
<td>15.31764</td>
<td>0.000598</td>
</tr>
<tr>
<td>10/9/1990 0:20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.61687</td>
<td>15.39555</td>
<td>-0.00909</td>
</tr>
<tr>
<td>10/9/1990 0:25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.57795</td>
<td>15.28356</td>
<td>-0.00074</td>
</tr>
<tr>
<td>10/9/1990 0:30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.45895</td>
<td>15.22055</td>
<td>0.004863</td>
</tr>
<tr>
<td>10/9/1990 0:35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.40791</td>
<td>15.2844</td>
<td>0.007353</td>
</tr>
</tbody>
</table>
**CapData. set_reg_trans**

- **power**='\text{-mtr-}',
- **poa**='\text{irr-poa-}',
- **t\_amb**='\text{temp-amb-}',
- **w\_vel**='\text{wind--}')

<table>
<thead>
<tr>
<th>CapData.reg_trans</th>
<th>poa</th>
<th>irr-poa-</th>
<th>t_amb</th>
<th>irr-ghi-</th>
<th>temp-amb-</th>
<th>temp-mod-</th>
<th>w_vel</th>
<th>wind--</th>
<th>power</th>
</tr>
</thead>
<tbody>
<tr>
<td>CapData.trans</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10/9/1990 0:00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.75067</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.77082</td>
</tr>
<tr>
<td>10/9/1990 0:05</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.73755</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.75303</td>
</tr>
<tr>
<td>10/9/1990 0:10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.64809</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.68944</td>
</tr>
<tr>
<td>10/9/1990 0:15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.64177</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.58212</td>
</tr>
<tr>
<td>10/9/1990 0:20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.61687</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.55575</td>
</tr>
<tr>
<td>10/9/1990 0:25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.57795</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.54595</td>
</tr>
<tr>
<td>10/9/1990 0:30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.45895</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.47273</td>
</tr>
<tr>
<td>10/9/1990 0:35</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.40791</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17.38453</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Weather Station 1 (Standard w/ POA GHI), Sun W/m²</th>
<th>Weather Station 2 (Standard w/ POA GHI), Sun W/m²</th>
<th>Weather Station 1 (Standard w/ POA GHI), Temp F °C</th>
<th>Weather Station 2 (Standard w/ POA GHI), Temp F °C</th>
<th>Weather Station 1 (Standard w/ POA GHI), Wind Speed mph</th>
<th>Weather Station 2 (Standard w/ POA GHI), Wind Speed mph</th>
<th>Elkor Production Meter, KW kW</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/9/1990 0:00</td>
<td>17.75067</td>
<td>17.77082</td>
<td>15.64035</td>
<td>15.52654</td>
<td>-0.00072</td>
<td>-0.00722</td>
<td>-8868</td>
</tr>
<tr>
<td></td>
<td>17.73755</td>
<td>17.75303</td>
<td>15.55192</td>
<td>15.70683</td>
<td>-0.00888</td>
<td>-0.00719</td>
<td>-8868</td>
</tr>
<tr>
<td>10/9/1990 0:10</td>
<td>17.64809</td>
<td>17.68944</td>
<td>15.54152</td>
<td>15.50686</td>
<td>0.008686</td>
<td>0.002557</td>
<td>-8868</td>
</tr>
<tr>
<td></td>
<td>17.64177</td>
<td>17.58212</td>
<td>15.31764</td>
<td>15.51052</td>
<td>0.000598</td>
<td>-0.01181</td>
<td>-8868</td>
</tr>
<tr>
<td>10/9/1990 0:20</td>
<td>17.61687</td>
<td>17.55575</td>
<td>15.39555</td>
<td>15.53016</td>
<td>-0.00909</td>
<td>0.005773</td>
<td>-8868</td>
</tr>
<tr>
<td></td>
<td>17.57795</td>
<td>17.54595</td>
<td>15.28356</td>
<td>15.28266</td>
<td>-0.00074</td>
<td>0.000666</td>
<td>-8868</td>
</tr>
<tr>
<td>10/9/1990 0:30</td>
<td>17.45895</td>
<td>17.47273</td>
<td>15.22055</td>
<td>15.36971</td>
<td>0.004863</td>
<td>0.000722</td>
<td>-8868</td>
</tr>
<tr>
<td></td>
<td>17.40791</td>
<td>17.38453</td>
<td>15.28244</td>
<td>15.25047</td>
<td>0.007353</td>
<td>0.005497</td>
<td>-8868</td>
</tr>
</tbody>
</table>

---

**Essential Functionality**

Identify data to be used in regression
Essential Functionality
Visually Review Data

CapData.plot()

- Creates Bokeh gridplot of data
- One plot per key in the translation (CapData.trans) dictionary attribute
- One line on each plot per column in the list of (CapData.trans) values
- Flexible – height, width, marker, columns of plots, plot subset of groups, change order of plots, include legends, merge plots (measured irradiance and modeled irradiance)
- Interactive- linked zooming and panning, hover tool
Essential Functionality
Visually Review Data

CapData.scatter
- Matplotlib poa vs. power scatter

CapData.scatter_hv
- Holoviews plot with or without linked time series
Essential Functionality
Visually Review Data

Holoviews plot with linked time series
Essential Functionality
Aggregate Columns of Data

CapData.agg_sensors()

- Aggregate measurements of the same variable from different sensors.

```
In [19]: das.agg_sensors(agg_map={'-inv-':'sum', 'irr-poa-':'mean', 'temp-amb-':'mean', 'wind--':'mean'})
Out[19]:
```

```
In [11]: das.df.head()
Out[11]:
```

```
<table>
<thead>
<tr>
<th>Inv. sum. agg</th>
<th>irr-poa mann. agg</th>
<th>temp. amb. mean. agg</th>
<th>wind. mean. agg</th>
<th>Weather Station 1 Sun, W/m^2</th>
<th>Weather Station 2 Sun, W/m^2</th>
<th>Weather Station 1 Sun2, W/m^2</th>
<th>Weather Station 2 Sun2, W/m^2</th>
<th>Weather Station 1 TempF</th>
<th>Weather Station 2 TempF</th>
<th>Elkon Product Meter KW, kW</th>
<th>Inverter 1 KW, kW</th>
<th>Inverter 2 KW, kW</th>
<th>Inverter 3 KW, kW</th>
<th>Inverter 4 KW, kW</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1200.0</td>
<td>0.0</td>
<td>17.760744</td>
<td>-0.003971</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>17.770821</td>
<td></td>
<td></td>
<td>-8665.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
</tr>
<tr>
<td>-1200.0</td>
<td>0.0</td>
<td>17.745260</td>
<td>-0.000036</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>17.737545</td>
<td></td>
<td></td>
<td>-8665.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
</tr>
<tr>
<td>-1200.0</td>
<td>0.0</td>
<td>17.668763</td>
<td>0.005622</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>17.648090</td>
<td></td>
<td></td>
<td>-8889.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
</tr>
<tr>
<td>-1200.0</td>
<td>0.0</td>
<td>17.611946</td>
<td>-0.006605</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>17.641772</td>
<td></td>
<td></td>
<td>-8889.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
</tr>
<tr>
<td>-1200.0</td>
<td>0.0</td>
<td>17.536308</td>
<td>-0.031650</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>17.616370</td>
<td></td>
<td></td>
<td>-8665.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
<td>-150.0</td>
</tr>
</tbody>
</table>
```

5 rows x 24 columns
Essential Functionality
Filter Methods – Remove rows of data

- Filters data stored in CapData.df_flt and saves result to CapData.df_flt
- Filter methods update summary data
- CapData.reset_flt will overwrite CapData.df_flt with DataFrame in CapData.df and reset summary data

```
In [25]: das.get_summary()
```

<table>
<thead>
<tr>
<th></th>
<th>pts_after_filter</th>
<th>pts_removed</th>
<th>filter_arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>das</td>
<td>1245</td>
<td>195</td>
<td>()</td>
</tr>
<tr>
<td>custom_filter</td>
<td>1240</td>
<td>5</td>
<td>(function DataFrame.dropna at 0x000000000008688...</td>
</tr>
<tr>
<td>filter_irr</td>
<td>424</td>
<td>816</td>
<td>(200, 2000),</td>
</tr>
<tr>
<td>filter_outliers</td>
<td>407</td>
<td>17</td>
<td>()</td>
</tr>
<tr>
<td>reg_cpt</td>
<td>385</td>
<td>22</td>
<td>() ('filter': True, 'summary': False)</td>
</tr>
</tbody>
</table>
Essential Functionality

Filter Methods

- filter_iri – low, high, percentage band around ref value
- filter_pvsyst – FShdBm, IL Pmin, IL Pmax, IL Vmin, IL Vmax
- filter_time – time between two dates or time period around a date
  - Future feature - Additional functionality and/or additional time filtering methods
- filter_outliers – applies elliptic envelope from scikit-learn
  - Future feature – lasso outliers on scatter plot to remove
- filter_pf – removes data where power factor is below given threshold
- filter_sensors – removes data where measurements of the same property differ by more than x%
  - Future feature – allow parameter to select hard value (for different exhibit requirements)
- filter_clearsky – keeps only clear data using the pvlib’s detect_clearsky
- custom_filter – updates summary information for pandas DataFrame methods or user custom functions
  - Can pass any function that takes a dataframe and returns the dataframe with rows removed
  - pd.DataFrame.dropna – remove rows with any or all missing data
  - pd.DataFrame.between_time – select data between times of day
Perform Regressions

- Captest uses statsmodels to run regressions, which in turn also uses patsy
- The regression equation defined in ASTM E2848 is defined by default in CapData.reg_fml

Regression results are saved to CapData.ols_model

```
In [51]: sim.reg_fml
Out[51]: 'power ~ poa + I(poa * poa) + I(poa * t_amb) + I(poa * w_vel) - 1'
```

```
In [52]: sim.ols_model.params
Out[52]:
    poa      I(poa * poa)      I(poa * t_amb)      I(poa * w_vel)
    7662.794466       -0.833516      -31.284538      -1.208963

In [53]: sim.ols_model.pvalues
Out[53]:
    poa      I(poa * poa)      I(poa * t_amb)      I(poa * w_vel)
    0.000000e+00       2.323363e-170      3.118074e-170      2.887841e-01
```
Essential Functionality

Calculate Reporting Conditions

• Captest.rep_cond used to calculate reporting conditions
• Separate statistical aggregation (mean, median, etc.) can be specified for each reporting conditions – POA, temperature, and wind speed
• ‘Irradiance Balance’ option - reporting irradiance determined by finding the irradiance that results in a balance of points within a +/- percent range of the reporting irradiance
• Always uses the filtered data in df_flt for calculations
• Can be used to calculate reporting conditions by month or at other frequencies
• Reporting conditions saved as a DataFrame to the CapData.rc attribute
  • Results function requires a CapData object with rc set and one with rc set to None

• See example notebook “Reporting Conditions Examples.ipynb” in github repository
• [https://github.com/bt-/pvcaptest/tree/master/examples](https://github.com/bt-/pvcaptest/tree/master/examples)
## Essential Functionality

### Summary of Filtering

In [46]: `pvc.get_summary(das, sim)`

Out[46]:

<table>
<thead>
<tr>
<th></th>
<th>pts_after_filter</th>
<th>pts_removed</th>
<th>filter_arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>das</td>
<td>filter_sensors</td>
<td>1245</td>
<td></td>
</tr>
<tr>
<td></td>
<td>custom_filter</td>
<td>1240</td>
<td>(&lt;function DataFrame.dropna at 0x0000000009638...</td>
</tr>
<tr>
<td></td>
<td>filter_irs</td>
<td>424</td>
<td>(200, 2000),</td>
</tr>
<tr>
<td></td>
<td>filter_outliers</td>
<td>407</td>
<td>0,</td>
</tr>
<tr>
<td></td>
<td>reg_cpt</td>
<td>385</td>
<td>(0), 'filter': True, 'summary': False</td>
</tr>
<tr>
<td></td>
<td>rep_cond</td>
<td>385</td>
<td>0,</td>
</tr>
<tr>
<td></td>
<td>filter_irs</td>
<td>294</td>
<td>(0.5, 1.5), 'ref_val': 762.7265615000001</td>
</tr>
<tr>
<td></td>
<td>reg_cpt</td>
<td>294</td>
<td>0,</td>
</tr>
<tr>
<td>sim</td>
<td>filter_time</td>
<td>1441</td>
<td>(0), 'test_date': '10/11/1990', 'days': 60</td>
</tr>
<tr>
<td></td>
<td>filter_irs</td>
<td>397</td>
<td>(200, 930),</td>
</tr>
<tr>
<td></td>
<td>filter_pvsyst</td>
<td>397</td>
<td>0,</td>
</tr>
<tr>
<td></td>
<td>filter_irs</td>
<td>284</td>
<td>(0.5, 1.5), 'ref_val': 762.7265615000001</td>
</tr>
<tr>
<td></td>
<td>reg_cpt</td>
<td>284</td>
<td>0,</td>
</tr>
</tbody>
</table>

das and sim are both CapData objects
Essential Functionality

Results

- CapData.res_summary
- sim and das are both CapData objects
- One but not both of the CapData objects must have reporting conditions set
- Both CapData objects must have regression results saved to CapData.ols_model
- Calculates and compares capacities using the regression parameter from each CapData object and the reporting conditions
- Expects the CapData object for simulated data first and measured data second.
Goals for Development
Expand user base

Add Users

• How many people here run or review capacity tests?
• Is this something you would consider using?
• What would stop you?
Goals for Development

Technical Review

Expert Signoff

- Independent Engineers who have more flexibility in reviewing test results could add this to the review process to compare to their current methods
- Researchers using ASTM for internal metrics can add this method
Goals for Development

Future work

Additional features / bug fixing

- More interactive plotting
- Better measurement grouping
- Improved documentation
- Other capacity test methods than ASTM e2848?
Links to Resources

- jupyter-notebook-beginner-guide
- ASTM E2848 - purchase required
- Pandas Documentation
- Holoviews Documentation
- pvlib Documentation
- Statsmodels Documentation