pvcaptest Evolves: Streamlined Bifacial Tests

Ben Taylor, Principal Consultant – Tailored Data Consulting

May 14th, 2026

What is pvcaptest?

Open-source Python package for running a capacity test following

ASTM E2848 – Standard Test Method for Reporting Photovoltaic Non-Concentrator System Performance

MIT Licensed

Introduced at PVPMC in 2019

Documented on Read the Docs

Install from PyPI

Or

with conda

Goal is to provide a full featured, flexible, open tool to allow all parties performing a test to replicate results with pvcaptest.

Background - What is an ASTM E2848 Capacity Test

1. Build and commission PV plant
2. Collect data through DAS / SCADA system
3. Quality check validate data

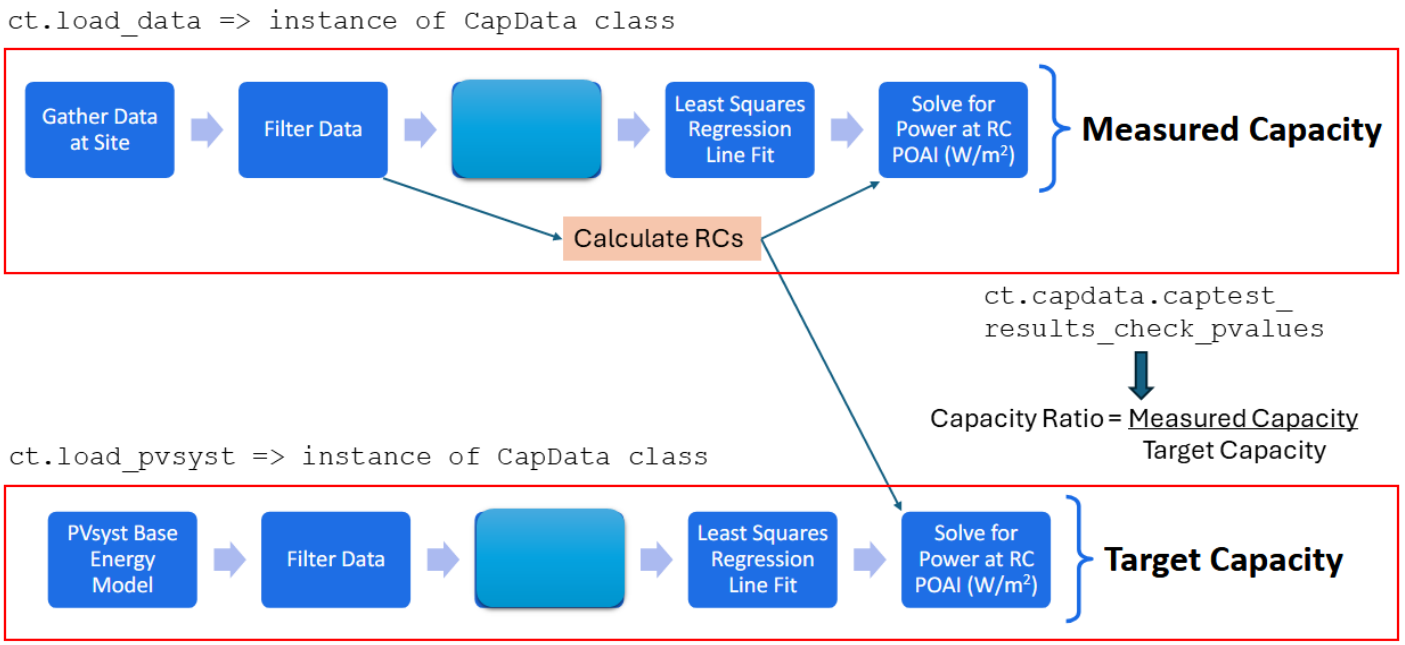
$$P = E_{POA}(a_1 + a_2 * E_{POA} + a_3 * T_a + a_4 * v)$$

4. Load data (csv, excel, parquet) into a data processing tool (Excel, R, Pandas, pvcaptest) — CapTest.from_params
5. Organize and transform data – column labels, units
6. Visualize data
7. Filter data - low irradiance, clear / cloudy periods, outliers, clipping, shading
8. Calculate reporting conditions
9. Fit regression: AC power ~ POA irradiance, ambient temperature, wind speed
10. Predict AC power with regression coefficients, regression equation, and reporting conditions

CapData methods
- Measured
- Modeled (PVsyst)

11. Compare predicted power from measured data vs predicted power from modeled data
12. Calculate uncertainty of result ?
13. Present / document results

Capacity Test with pvcaptest – Old Workflow



If it ain't broke, don't …

Issues with non Standard Regression Equations

pvcaptest was originally created to run tests for the regression equation defined in ASTM E2848

```
power ~ poa + I(poa * poa) + I(poa * t_amb) + I(poa * w_vel) - 1
```

There *was and is* flexibility provided by statsmodels + patsy, which allowed setting the regression equation to

```
power ~ poa + rpoa + 1
```

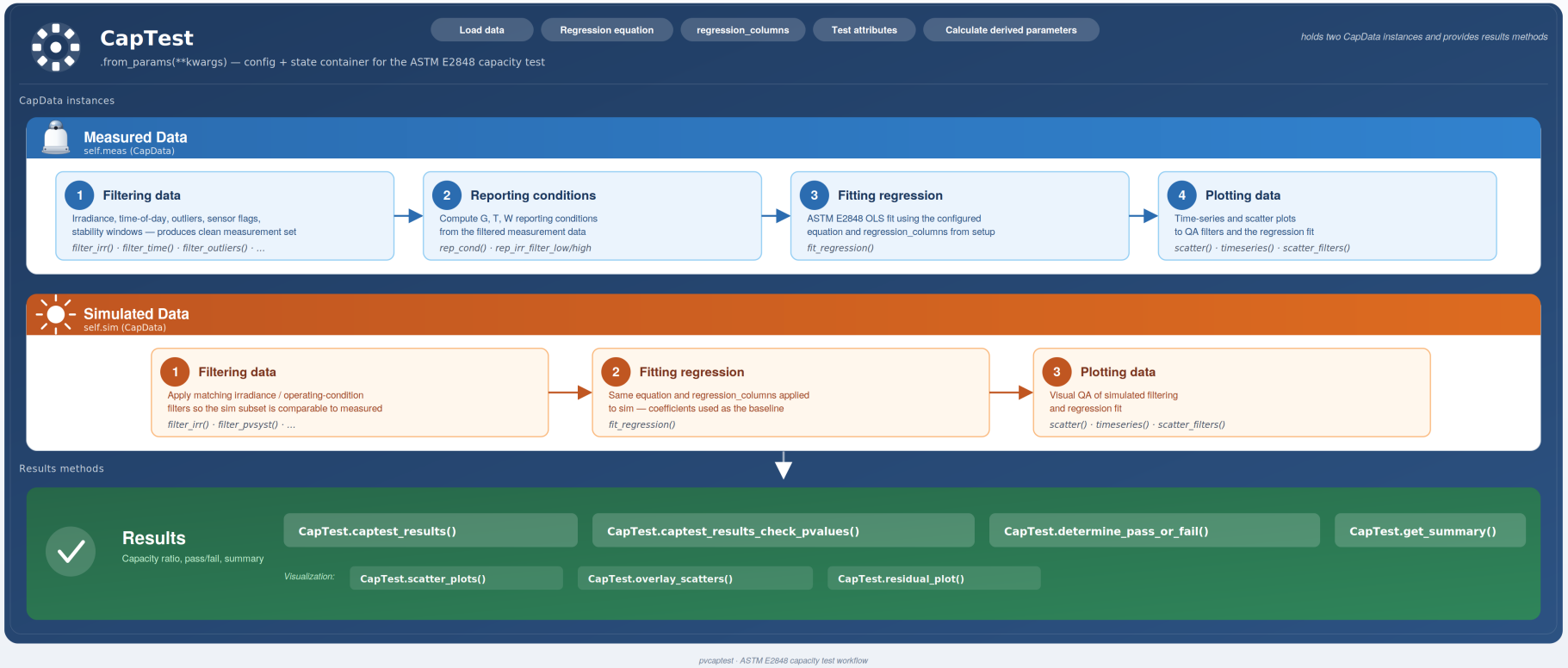
AND

```
meas.regression_columns = {
    'power': 'power_tc',
    'poa': 'poa_agg_mean',
    'rpoa': 'rpoa_agg_mean',
}
```

But, it was not convenient. pvcaptest was in the way, not making the testing easier

- manually pre-compute the power_tc column and manually add to `data` and `data_filtered`
- reporting conditions would not calculate a value for rear POA
- Scatter plot would not show temperature-corrected power vs rear POA

Capacity Test with pvcaptest – New Workflow



What's New – v0.13 to v0.15.1

	V0.13.0 – PVPMC 2024	V0.15.1 – Latest Release
Identifying Regression terms	• Mapping regression terms to a column name (met1_poa_wm2) or column group id ("irr_poa") • Support for updating mapping for aggregations of column groups ("irr_poa" --> "irr_poa_mean_agg")	• Predefined mappings of regression terms through functions to column names or aggregations of column groups • Recursive walking of mapping from bottom up to create calculated regression parameters
Reporting Conditions	• Reporting condition calculations were not flexible for non-standard regression equations	• Reporting condition calculation included in predefined tests • Reporting condition calc parses the right side of the regression equation
Scatter Plots	• Scatter plots of POA irradiance vs power – only POA vs power	• Scatter plot defined with test setup • Visually split plot by AM vs PM • Plot temperature-corrected power vs irradiance
Results	• Results through f(meas, sim)	• Results from CapTest methods

New Features Summary

Predefined Test Setups

captest.captest.TEST_SETUPS

Recursive processing of regression_cols dictionary

captest.CapData.process_regression_columns

Functions to calculate non-standard regressors

calcparams module

CapTest Class

Predefined Test Setups

- Five required keys per preset:
  - regression_formula
  - reg_cols_meas, reg_cols_sim
  - rep_conditions
  - scatter_plots
- Four presets:
  - e2848_default – standard ASTM E2848
  - **bifi_e2848_etotal – bifacial with e_total**
  - bifi_power_tc – temperature-corrected power
  - e2848_spec_corrected_poa – First Solar spectral correction
- Plus ‘custom’ for inline overrides

```
"bifi_power_tc": {
    "reg_cols_meas": {
        "power": (
            power_temp_correct,
            {
                "power": ("real_pwr_mtr", "sum"),
                "cell_temp": (
                    cell_temp,
                    {
                        "poa": ("irr_poa", "mean"),
                        "bom": (
                            bom_temp,
                            {
                                "poa": ("irr_poa", "mean"),
                                "temp_amb": ("temp_amb", "mean"),
                                "wind_speed": ("wind_speed", "mean"),
                            },
                        ),
                    },
                ),
            },
        ),
        "poa": ("irr_poa", "mean"),
        "rpoa": ("irr_rpoa", "mean"),
    },
    "reg_cols_sim": {
        "power": (
            power_temp_correct,
            {
                "power": "E_Grid",
                "cell_temp": "TArray",
            },
        ),
        "poa": "GlobInc",
        "rpoa": (rpoa_pvsyst, {"globbak": "GlobBak", "backshd": "BackShd"}),
    },
    "reg_fml": "power ~ poa + rpoa",
    "scatter_plots": scatter_bifi_power_tc,
    "rep_conditions": {
        "irr_bal": False,
        "percent_filter": 20,
        "front_poa": "poa",
        "func": {
            "poa": perc_wrap(60),
            "rpoa": "mean",
        },
    },
},
```

Predefined Test Setups

Harcoded Column Group IDs

- `irr_poa` — front-side plane-of-array irradiance (all setups)
- `irr_rpoa` — rear-side plane-of-array irradiance (`bifi_e2848_etotal`, `bifi_power_tc`)
- `real_pwr_mtr` — AC power meter (all setups)
- `temp_amb` — ambient temperature (all setups)
- `wind_speed` — wind speed (all setups)
- `humidity` — relative humidity (`e2848_spec_corrected_poa` only)
- `pressure` — station pressure (`e2848_spec_corrected_poa` only)

```
"bifi_power_tc": {
    "reg_cols_meas": {
        "power": (
            power_temp_correct,
            {
                "power": ("real_pwr_mtr", "sum"),
                "cell_temp": (
                    cell_temp,
                    {
                        "poa": ("irr_poa", "mean"),
                        "bom": (
                            bom_temp,
                            {
                                "poa": ("irr_poa", "mean"),
                                "temp_amb": ("temp_amb", "mean"),
                                "wind_speed": ("wind_speed", "mean"),
                            },
                        ),
                    },
                ),
            },
        ),
        "poa": ("irr_poa", "mean"),
        "rpoa": ("irr_rpoa", "mean"),
    },
    "reg_cols_sim": {
        "power": (
            power_temp_correct,
            {
                "power": "E_Grid",
                "cell_temp": "TArray",
            },
        ),
        "poa": "GlobInc",
        "rpoa": (rpoa_pvsyst, {"globbak": "GlobBak", "backshd": "BackShd"}),
    },
    "reg_fml": "power ~ poa + rpoa",
    "scatter_plots": scatter_bifi_power_tc,
    "rep_conditions": {
        "irr_bal": False,
        "percent_filter": 20,
        "front_poa": "poa",
        "func": {
            "poa": perc_wrap(60),
            "rpoa": "mean",
        },
    },
},
```

Recursive Processing Regression Columns

Engine behind calculated regressors

- regression_cols values can now be nested (callable, kwargs) tuples in addition to column-group strings
- Recursive walk of the dict materializes calculated columns into CapData.data, then rewrites regression_cols to point at them
- Supports two node types:
  - aggregation tuples – (column_group, agg_func)
  - calculation tuples – (callable, kwargs dict)
- Verbose output logs every aggregation and calculation step (see slide 12 for the bifi_e2848_etotal run)

```
my_meas_cols = {
    "power": (
        power_temp_correct,
        {
            "power": ("real_pwr_mtr", "sum"),
            "cell_temp": (
                cell_temp,
                {
                    "poa": ("irr_poa", "mean"),
                    "bom": (
                        bom_temp,
                        {
                            "poa": ("irr_poa", "mean"),
                            "temp_amb": ("temp_amb", "mean"),
                            "wind_speed": ("wind_speed", "mean"),
                        },
                    ),
                },
            ),
        },
    ),
    "poa": ("irr_poa", "mean"),
    "t_amb": ("temp_amb", "mean"),
    "w_vel": ("wind_speed", "mean"),
}
```

Recursive Processing - Example

Starting from bottom for the regression term "power"

1. **Bottom three use already existing aggregation of column groups. For example, ("irr_poa", "mean") => "irr_poa_mean_agg"**
2. **bom_temp – new function in module calcparams. Simple self-documenting calculations. Given values for kwargs "poa", "temp_amb", and "wind_speed", returns a Series called "bom_temp".**
3. **cell_temp – new function in module calcparams**
4. **power_temp_correct – new function in module calcparams**

```
my_meas_cols = {
    "power": (
        power_temp_correct,
        {
            "power": ("real_pwr_mtr", "sum"),
            "cell_temp": (
                cell_temp,
                {
                    "poa": ("irr_poa", "mean"),
                    "bom": (
                        bom_temp,
                        {
                            "poa": ("irr_poa", "mean"),
                            "temp_amb": ("temp_amb", "mean"),
                            "wind_speed": ("wind_speed", "mean"),
                        },
                    ),
                },
            ),
        },
    ),
```

```
CapData.regression_cols = {
    'power': 'power_temp_correct',
```

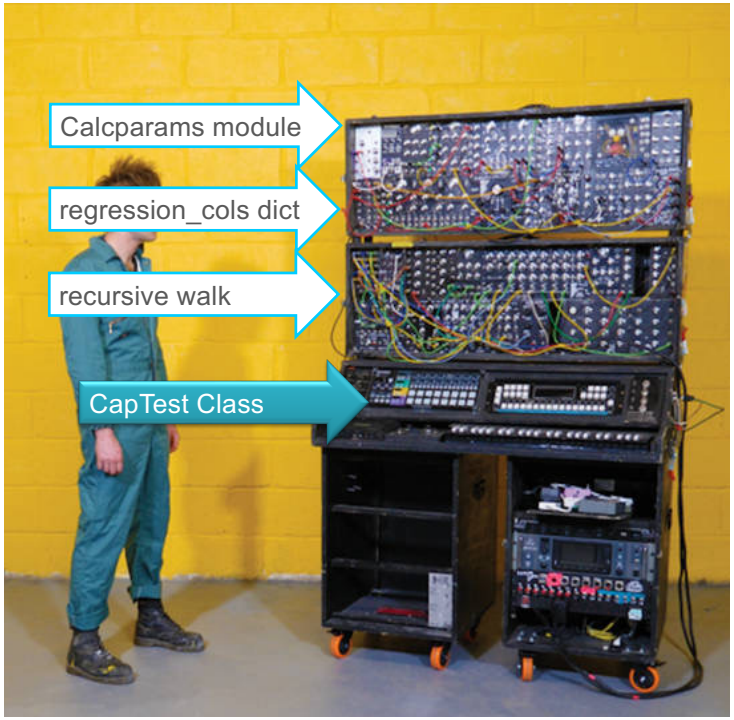
What's new – calcparams Module

Library of calculated regression parameters that ship with pvcaptest

- **Bifacial:**
  - `e_total` – poa + rpoa * bifaciality; options for bifacial fraction and rear shade loss
- **Temperature:**
  - `cell_temp`, `bom_temp`, `power_tc`
- **Spectral correction (First Solar):**
  - `apparent_zenith`, `absolute_airmass`, `precipitable_water_gueymard`, `spectral_factor_firstsolar`, `poa_spec_corrected`
- **Composable:**
  - Any function that takes a DataFrame and returns a Series
  - Drop in your own functions via the 'custom' test_setup

What's new – CapTest Class

- Holds measured + simulated CapData instances with shared config
- Three constructors:
  - CapTest(meas=, sim=, test_setup=…)
  - CapTest.from_params(…)
  - CapTest.from_yaml(path)
- setup() resolves preset and configures both CapData instances
- Results functions moved from:
  - captest_results, get_summary, residual_plot, overlay_scatters
- to_yaml round-trips full config

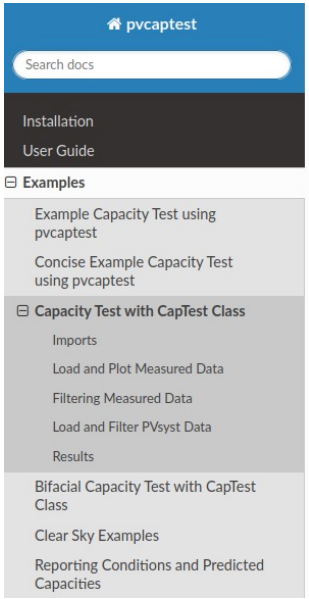


Example of CapTest.from_params

```
ts = CapTest.from_params(
    test_setup='bifi_e2848_etotal',
    meas_path='./data/example_meas_data_bifi.csv',
    sim_path='./data/pvsyst_example_HourlyRes_2_bifi.CSV',
    test_tolerance='+/- 3',
    ac_nameplate=6_000,
    bifaciality=0.7,
    meas_load_kwargs={
        'group_columns': './data/column_groups_bifi.xlsx',
    },
)
```

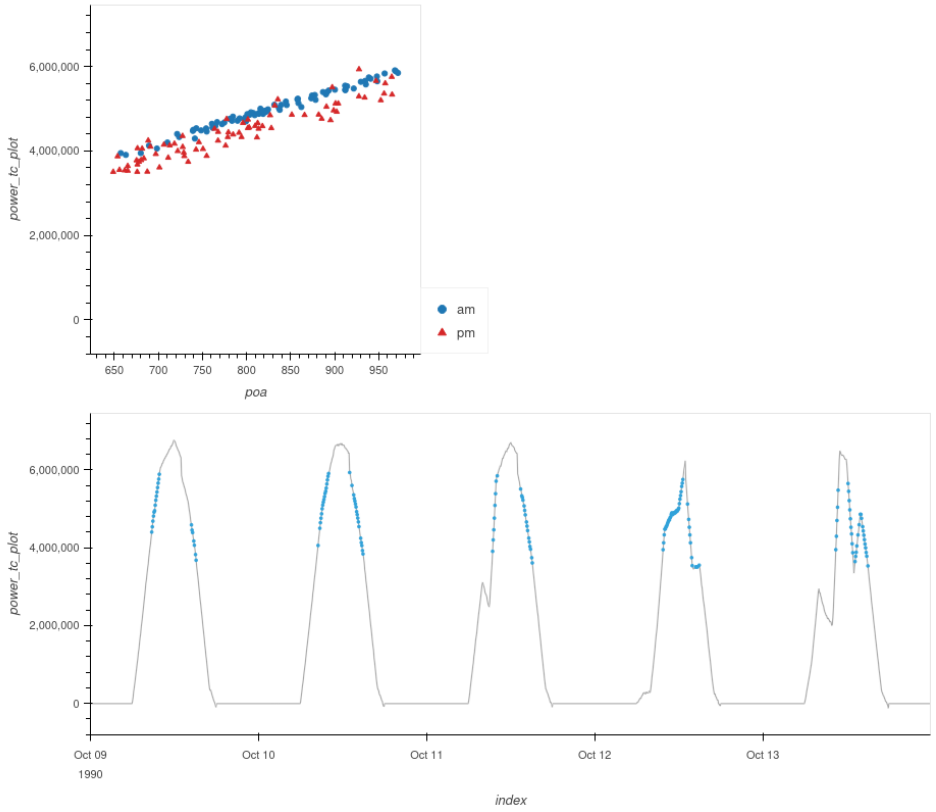
```
before calling get common timestep
5min
Aggregating the below 1 columns of the real_pwr_mtr group using the sum function. New column name: real_pwr_mtr_sum_agg:
    Example Project_Elkor Production Meter_Elkor Production Meter, KW


Aggregating the below 2 columns of the irr_poa group using the mean function. New column name: irr_poa_mean_agg:
    Example Project_Weather Station 1 (Standard w/ POA GHI)_Weather Station 1 (Standard w/ POA GHI), Sun
    Example Project_Weather Station 2 (Standard with POA GHI)_Weather Station 2 (Standard with POA GHI), Sun


Aggregating the below 2 columns of the irr_rpoa group using the mean function. New column name: irr_rpoa_mean_agg:
    Example Project_Weather Station 1_RPOA
    Example Project_Weather Station 2_RPOA


Calculating and adding "e_total" column as irr_poa_mean_agg + irr_rpoa_mean_agg * 0.7 * 1 * (1 - 0)
Aggregating the below 2 columns of the temp_amb group using the mean function. New column name: temp_amb_mean_agg:
    Example Project_Weather Station 1 (Standard w/ POA GHI)_Weather Station 1 (Standard w/ POA GHI), TempF
    Example Project_Weather Station 2 (Standard with POA GHI)_Weather Station 2 (Standard with POA GHI), TempF


Aggregating the below 2 columns of the wind_speed group using the mean function. New column name: wind_speed_mean_agg:
    Example Project_Weather Station 1 (Standard w/ POA GHI)_Weather Station 1 (Standard w/ POA GHI), WindSpeed
    Example Project_Weather Station 2 (Standard with POA GHI)_Weather Station 2 (Standard with POA GHI), WindSpeed


Calculating and adding "rpoa_pvsyst" column as GlobBak + BackShd.
Calculating and adding "e_total" column as GlobInc + rpoa_pvsyst * 0.7 * 1 * (1 - 0)
```

Link to Example

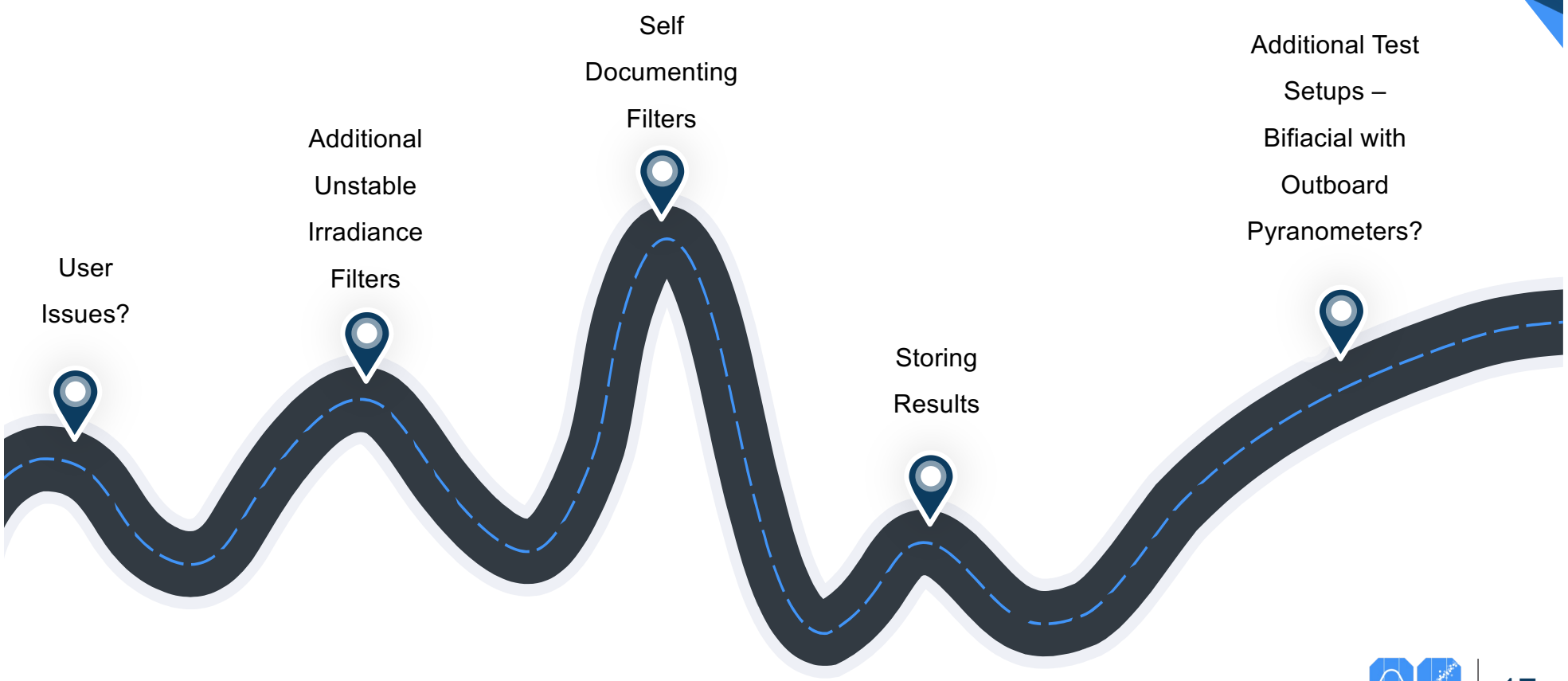


Improved Scatter Plots

```
ct.plotting.ScatterPlot(cd=ts.meas, timeseries=True, tc_power=True, tc_mode='replace', split_day=True).view()
```



Ongoing Work



Resources / Questions

Contact me – ben@td.consulting

www.td.consulting

Open an Issue on Github

Source Code on Github

Documentation & CapTest user guide on Read the Docs