# Package ‘pv.ts.sandbox’

November 16, 2023

**Title** Photovoltaic Time Series Sandbox

**Version** 0.0.0.9000

**Description** Data visualization with shiny, leaflet, and shinyDashboard. Use the Map tab in the dashboard for a quick glance at time series data. For a more in-depth analysis, use the analyze tab. Functions in this package can be used alone, but were primarily designed to work with the dashboard. License MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Suggests** devtools,
knitr,
rmarkdown,
roxygen2,
testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Depends** R (>= 3.0.2),
shiny (>= 1.7.5)

**LazyData** true

**Imports** data.table,
dplyr,
ggplot2,
ggridges,
viridis,
hrbrthemes,
ggExtra,
rlang,
lubridate,
naniar,
shinydashboard,
leaflet,
DT,
htmlwidgets,
visdat,
scattermore,
magrittr,
hms,
shinycssloaders

**VignetteBuilder** knitr

---

`calculate_monthly_energy`

## *Calculating Monthly Energy of PV System*

---

### Description

Use when data frame is preprocessed after load_datasets() function. Calculates how much energy is yielded from given PV system for all 12 months

### Usage

```
calculate_monthly_energy(
  data,
  timestamp.var,
  current.var,
  voltage.var,
  power.var = NULL
)
```

### Arguments

| | |
|---|---|
| `data` | A data.table of time series data with columns containing (current output and voltage output data) or (power output data). |
| `timestamp.var` | A character. The name of the column in `data` that represents time. |
| `current.var` | A character. The name of the column in `data` that represents output current, "A-dc". |
| `voltage.var` | A character. The name of the column in `data` that represents output voltage, "V-dc". |
| `power.var` | A character. Default NULL. The name of the column that represents the power output values in `data`. See details. |

### Value

A data.table with energy yielded in month intervals

### Examples

```
energy_monthly <- calculate_monthly_energy(data = dt, timestamp.var = "TIMESTAMP",
                                        current.var = "PV1AMP", voltage.var = "PV2VLT")
energy_monthly <- calculate_monthly_energy(data = dt, timestamp.var = "TIMESTAMP",
                              power.var = "POWOUT1")
```

## count_missing *Count Number of Missing Values*

### Description

Use when data frame is cleaned after using dataset_cleaning() function. Provides a numeric value of data cells with NULL or empty values

### Usage

```
count_missing(data)
```

### Arguments

`data` A dataframe

### Value

The total number of NA's in `data`.

### Examples

```
num_missing <- count_missing(data = dt)
```

## `count_missing_col` *Count Number of Missing Values for Each Column*

### Description

Use when data frame is cleaned after using dataset_cleaning() function. Provides a data.table of numeric values for each column containing NULL or empty values

### Usage

```
count_missing_col(data)
```

### Arguments

`data` A dataframe

### Value

A numeric list containing count of NA's in each column.

### Examples

```
missing_col_count <- count_missing_col(dt)
```

## `dashboard` *Photovoltaic System Dashboard*

### Description

Photovoltaic System Dashboard

### Usage

```
dashboard(
    dt_list,
    metadata,
    timestamp_var = "TIMESTAMP",
    timestamp_format = "%b %d %Y-%H:%M:%S"
  )
```

### Arguments

`dt_list` List of data.tables. Obtained from `load_datasets()`

`metadata` The metadata for dt_list. See `load_metadata()` and `save_metadata_template()`

`timestamp_var` A character. The name of the column in the datasets that represents time. Typically "timestamp" or "TIMESTAMP".

`timestamp_format` The format of the column specified by `timestamp_var`. Ensure all datasets share identical formats.

## `data_summary` *Generating a Summary*

### Description

Create a general summary of a dataset. Statistics include mean, standard deviation, minimum, maximum, median, and mode.

### Usage

```
data_summary(data, digits = 3)
```

### Arguments

`data` A data.table of PV data

`digits` A numeric value. The cutoff for decimal values.

### Value

A summary as a data.table

### Examples

```
df_summ <- data_summary(data = schoolA_sample, digits = 2)
```

## `date_range` *Date Range*

### Description

`date_range()` returns a vector containing the minimum and maximum timestamps. Uses the base function `range()`.

- The `timestamp_var` column must be of class POSIXt.

### Usage

```
date_range(data, timestamp_var)
```

### Arguments

`data` A dataframe of time series data.

`timestamp_var` A character. The name of the column in `data` that represents time.

### Value

A vector containing the start date and end date

### Examples

```
range <- date_range(data = schoolA_sample, timestamp_var = "TIMESTAMP")
```

## `get_mode` *Get statistical Mode of List*

### Description

Helper function to find element(s) that appear the most in a list A string will have the mode of the given list.

### Usage

```
get_mode(lst, na.rm = FALSE, digits = 4)
```

### Arguments

`lst` A vector or list type

`na.rm` Whether to remove NA's prior to calculation

`digits` Number of decimal places to round at

### Value

A string containing the mode(s)

## `load_datasets` *Load Datasets*

### Description

Using the metadata file, load all files listed in metadata$Name_of_File as a list of data.tables. Con vert the timestamp column to class POSIXt. Append YEAR, MONTH, MONTHYEAR, MDAY, WEEK, and HOUR columns for use in other package functions.

### Usage

```
load_datasets(
  data_dir = getwd(),
  metadata = "",
  timestamp_var = "TIMESTAMP",
  timestamp_format = "%b %d %Y-%H:%M:%S"
)
```

### Arguments

`data_dir` Preferred: current directory. The full path to the *directory* containing PV dataset

`metadata` The metadata loaded as a dataframe. See `load_metadata()`

`timestamp_var` A character. The name of the column in the datasets that represents time.

`timestamp_format` The format of the timestamp column. Ensure all datasets share identical formats

### Details

Ensure that your data meets the criteria & requirements outlined in `save_metadata_template()` The run time of this function is dependent on the size of the provided datasets.

### Value

A list of data.tables indexed according to how they are indexed in the provided metadata

### Examples

```
# loading metadata
md_path <- list.files(system.file('extdata', package = 'pv.ts.sandbox'),
                         full.names = TRUE)[1]

md <- load_metadata(path = md_path)

# directory path
data_directory <- system.file('extdata', package = 'pv.ts.sandbox')

data_list <- load_datasets(data_directory, md)
```

## `load_metadata` *Load Metadata*

### Description

Use after filling out the metadata (see save_metadata_template) to load the metadata into the environment as a data.table.

### Usage

```
load_metadata(path)
```

### Arguments

`path` The full path to the *file* containing relevant metadata.

### Value

A data.table

### Examples

```
md_path <- list.files(system.file('extdata', package = 'pv.ts.sandbox'), full.names =
TRUE) md_path <- md_path[1]

md <- load_metadata(path = md_path)
```

## `plot_heatmap` *Plot Heatmap*

### Description

Visualize a time series heatmap for the entire dataset categorized by years, months, and hours.

### Usage

```
plot_heatmap(data, plot_var, plot_title = NULL)
```

### Arguments

`data` A data.table of timeseries data

`plot_var` The desired variable to plot

`plot_title` Optional plot title

### Value

A gg

### Examples

```
md_path <- list.files(system.file('extdata', package = 'pv.ts.sandbox'),
                          full.names = TRUE)[1]
```

```
md <- load_metadata(path = md_path)

# directory path
data_directory <- system.file('extdata', package = 'pv.ts.sandbox')

# use load_datasets to add necessary columns
data_list <- load_datasets(data_directory, md)

plot_heatmap(data = data_list[[1]], plot_var = "PVTMP1",
             plot_title = "Hourly Photovoltaic Temperatures - SchoolA" )
```

## `plot_missing` *Visualize Missing Values*

### Description

Provides a quick over-view of the missingness inside a dataframe using functions from visdat and naniar.

### Usage

```
plot_missing(
  data,
 type_option = c("gg_miss_var", "vis_miss", "vis_dat", "gg_miss_case",
  "miss_point"), as_percent = FALSE,
  plot_title = NULL,
  miss_point_plotx = NULL,
  miss_point_ploty = NULL
)
```

### Arguments

`data` A data.table

`type_option` The type of plot to return. Must be one of c("gg_miss_var", "vis_miss", "vis_dat", "gg_miss_case", "miss_point")

`as_percent` Some plot types have the option to view a percentage instead of a total

number `plot_title` Plot label. Default is none.

`miss_point_plotx`
For type_option = "miss_point" only. The x-axis variable to plot

`miss_point_ploty`
For type_option = "miss_point" only. The y-axis variable to plot

### Details

- `gg_miss_var`: Displays a line chart representing missingness of all columns in data. See naniar::gg_miss_var
- `vis_miss`: Displays a graphic of missingness for all columns in data. Includes specific

per centages. See visdat::vis_miss

- vis_dat: Displays a graphic of all columns in data color-coded by data type. See vis dat::vis_dat
- gg_miss_case: Displays a graphic that represents missingness at each row in the data. See naniar::gg_miss_case
- miss_point: Requires miss_point_plotx & miss_point_ploty parameters. For specific variable to variable comparison. See naniar::geom_miss_point

### Value

A gg

### Examples

```
plot_missing(data = schoolB_sample, type_option = "gg_miss_var", as_percent = TRUE)
plot_missing(data = schoolE_sample, type_option = "miss_point",
             miss_point_plotx = "PV1VLT", miss_point_ploty = "PV1AMP")
```

## plot_raw *Scatter Plot w/ Linear Trend*

### Description

Creates a basic scatter plot of the raw data using ggplot2 with scattermore::geom_scattermore.

### Usage

```
plot_raw(data, plot_x = "TIMESTAMP", plot_y, pointsize = 0, plot_title = NULL)
```

### Arguments

data a data.table

plot_x the column name in data containing x-values

plot_y the column name in data containing y-values

pointsize Numeric. Represents radius of point. See scattermore::geom_scattermore or Details.

plot_title Plot label. Default is none.

### Details

From scattermore::geom_scattermore pointsize: "Radius of rasterized point. Use 0 for single pixels (fastest)."

### Value

a gg

### Examples

```
p <- plot_raw(data = schoolC_sample, plot_x = "TIMESTAMP", plot_y = "WNDDIR",
                  plot_title = "School C Sample Data")
```

## `plot_ridgeline` *Generates a Ridgeline Plot using ggrides*

### Description

Ridgeline plots (also known as Joyplots) help visualize the distribution of a numeric variable for several groups. Temperature is the most common variable for Ridgeline plots in the photovoltaic community.

### Usage

```
plot_ridgeline(
  data,
  plot_var,
  time_period = c("year", "monthyear", "month", "week", "timestamp"),
  color_theme = c("A", "B", "C", "D", "E", "F", "G", "H"),
  ridge_scale = 3,
  x_title = NULL,
  y_title = NULL,
  plot_title = NULL,
  shape_option = "density_ridges",
  legend_name = NULL,
  custom_theme = NULL
)
```

### Arguments

`data` A data.table of time series data.

`plot_var` A character or numeric index of the column in `data` to plot.

`time_period` Either "year", "month", "week", or "timestamp". The columns for these can be generated from `load_datasets()`.

`color_theme` The color theme of the plot. See viridis::scale_fill_viridis for details on

themes. `ridge_scale` Increase or decrease ridge size.

`x_title` x-axis label. Default is the name of `plot_var`.

`y_title` y-axis label. Default is `time_period`.

`plot_title` Plot label. Default is none.

`shape_option` Adjust the base shape of the ridges. See ggridges::geom_ridgeline_gradient for list of shapes.

`legend_name` Legend title.

`custom_theme` See `theme()` for a full customization list. See the details section below for the default theme.

### Details

If the default custom.theme = NULL remains the default theme is: `theme(legendposition = 'right', panel.spacing = unit(0.1, "lines"), axis.text.x = element_text(size = 15), axis.text.y = element_text(size = 15), plot.background = element_rect(fill = 'white', color = 'white'))`

### Value

A Ridgeline plot of class "ggplot"

### Examples

```
p <- plot_ridgeline(data = schoolA_sample, plot_var = 5, time_period = 'month', color_theme = 'C')

p <- plot_ridgeline(data = schoolD_sample, plot_var = "AMBTMP", time_period = 'MONTH',
            color_theme = 'D', x_title = 'Ambient Temperature', legend_name = 'Temp.
            [°C]')
```

---

## `sample_metadata` *Sample Metadata*

---

### Description

Relevant metadata for schools A, B, C, D, and E. See `sample_data`.

### Usage

```
sample_metadata
```

### Format

An object of class data.table (inherits from data.frame) with 5 rows and 5 columns.

Name_of_File The name of the file containing timeseries data. Include the file extension. School The name of the school or organization where the photovoltaic system is located Latitude, Longitude The geographical coordinates of the photovoltaic system Climate_Zone The climate zone/ area of the location. c("1", "2", "3", "4", "5E", "5W", "7")

---

## `sample_schools` *Sample Time Series Data*

---

## Description

A 24 hour sample of photovoltaic data over 15 minute intervals. Values in sample_metadata correspond.

## Usage

```
schoolA_sample

schoolB_sample

schoolC_sample

schoolD_sample

schoolE_sample
```

## Format

An object of class `data.table` (inherits from `data.frame`) with 96 rows and 45 columns.

## Details

TIMESTAMP is of class character. Remaining classes are numeric.

TIMESTAMP Date & Time data of format '%b %d %Y-%H:%M:%S'

BAT2AM Current out of battery bank [A-dc]

BAT1AM Current into battery bank [A-dc]

BATVLT Battery bank voltage [V-dc]

PV1AMP, PV2AMP, PV3AMP PV array output current [A-dc]

PV1VLT, PV2VLT, PV3VLT PV array output voltage [V-dc]

FMUGNT Net cumulative energy from grid [MWh-ac]

FMUGPW Net average power from grid [kW-ac]

FMUGPF Power factor at grid [decimal]

IMPTOT Imported cumulative energy from grid [MWh-ac]

EXPTOT Exported cumulative energy to grid [MWh-ac]

FMUG1V, FMUG2V, FMUG3V Inverter voltage from utility grid [V-ac]

FMUG1A, FMUG2A, FMUG3A Inverter current from utility grid [A-ac]

TOCLNT Net energy to critical load [MWh-ac]

TOCLPW Net power to critical load [kW-ac]

TOCLPF Power factor at critical load [decimal]

TOCTOT Cumulative energy to critical load [MWh-ac]

FMCTOT Cumulative energy from critical load [MWh-ac]

TOCL1V, TOCL2V, TOCL3V Inverter voltage to critical load [V-ac]

TOCL1A, TOCL2A, TOCL3A Inverter current to critical load [A-ac]

WINDDIR Wind direction [0-255]

POAIRR Plane of array irradiance [W/mB2]

AMBTMP Ambient temperature [B0F]

PVTMP1 Module temperature [B0F]

WNDSPD Wind speed [m/s]

BATTMP Battery bank temperature [B0F]

CC1AMP, CC2AMP, CC3AMP Charge controller output current [A-dc]

CC1VLT, CC2VLT, CC3VLT Charge controller 1 output voltage [V-dc]

PVPRED Predicted PV power [kW-dc]

`save_metadata_template`

## *Save Metadata Template*

### Description

Saves a copy of the metadata template. *This template is required for dashboard* File save name: "metadata_template.csv"

### Usage

```
save_metadata_template(path = NULL)
```

### Arguments

`path` A directory path to save the .csv template. Default is the current working direc tory.

### Details

utils::write.csv is used to save the template. See metadata_template and sample_metadata.

### Examples

```
save_metadata_template()
```

## `template` *Metadata Template*

### Description

Download with save_metadata_template. Please carefully follow the instructions when filling out the template. Load using load_metadata to ensure it has been properly filled out. *Metadata required for use of dashboard*.

### Usage

```
template
```

### Format

An object of class `data.table` (inherits from `data.frame`) with 12 rows and 51 columns.

### Details

See `metadata_sample` for an example.

## `time_frequency` *Time Frequency*

### Description

Finds median and mean timestamp in given photovoltaic data

### Usage

```
time_frequency(data, timestamp.var)
```

### Arguments

`data` A dataframe containing timeseries data

`timestamp.var` A character. The name of the column in `data` that represents time.

### Value

A data.table with MEAN, MEDIAN, and TIME_INTERVAL(s) columns representing respective statistic values

## `yoy_degradation` *Year-Over-Year Degradation*

### Description

Calculate degradation annually using the year-over-year method.

- Uses the formula: 100 * ((totalPower_month2 - totalPower_month1) / totalPower_month1) to calculate average monthly change.
- Then groups by year to calculate the annual average.

### Usage

```
yoy_degradation(
  data,
  timestamp.var,
  current.var,
  voltage.var,
  power.var = NULL
)
```

### Arguments

`data` A dataframe of time series data with columns containing (current output and voltage output data) or (power output data).

`timestamp.var` A character. The name of the column in `data` that represents time.

`current.var` A character. The name of the column in `data` that represents output current, "A-dc".

`voltage.var` A character. The name of the column in `data` that represents output voltage, "V-dc".

`power.var` A character. Default NULL. The name of the column that represents the power output values in `data`. See details.

### Details

Use `power.var` when power output is already calculated and is present in the data. Otherwise, use `current.var` in combination with `voltage.var`.

### Value

A data.table containing the mean power output and % change in power output by month

### Examples

```
pv_deg <- yoy_degradation(data = schoolB_sample, timestamp.var = "TIMESTAMP",
                          current.var = "PV1AMP", voltage.var = "PV2VLT")
```

## Dashboard Tutorial (with metadata example)

This section provides a visual example of the metadata template. Other aids can be found in the package itself with:

```
      ?sample_metadata
```

| Name_of_File | School | Latitude | Longitude | Climate.zone |
|---|---|---|---|---|
| schoolA_sample.rda | schoolA | 30.21181 | -85.64371 | 1 |
| schoolB_sample.rda | schoolB | 29.47800 | -80.61210 | 5 |

To edit the template for your own data, the template can be saved with:

```
      > save_metadata_template()
```

The file “metadata_template.csv” can now be found in the current working directory.

# Walkthrough Vignette [R]

This section covers the “Getting Started & Dashboard Walkthrough” vignette for the R package, pv.ts.sandbox.

# Getting Started & Dashboard Walkthrough

This package provides all the necessary functions for populating the pv.ts.sandbox dashboard.

# Getting Started

## Metadata

The `dashboard()` function is dependent on a metadata template, this is what populates the map markers. Firstly, download the template, it will be saved to your working directory with the file name ‘metadata_template.csv’.

```
library(pv.ts.sandbox)
#> Loading required package: shiny
save_metadata_template()
```

**To fill out the template:**

- Name_of_File: This column should contain the file names with extension of the data to use with the dashboard. If the data is found in the file ‘my_data.csv’, then ‘my_data.csv’ should be written here.
- School: This column should contain the name of the geographical location where that file’s data was collected. If the data is from the University of Central Florida,then ‘University of Central Florida’ would be written here.
- Latitude: This column should contain the latitude coordinate corresponding to the data. For the University of Central Florida, ‘28.602520092994908’ would be entered. There is not a significant digit requirement, feel free to be as precise or imprecise as necessary.
- Longitude: For This column should contain the latitude coordinate corresponding to the data. For the University of Central Florida, ‘-81.2000838882772’ would be entered. There is not a significant digit requirement, feel free to be as precise or imprecise as necessary.
- Climate zone: This column should contain the climate zone the data exists in. *If this information is unknown it can be left blank.* For the University of Central Florida, ‘1’ would be entered.
- Before loading the template, DELETE the first column, second row, and third row.

Detailed instructions are also provided in the template itself, After completing this, the file should be loaded into your global environment.

```
# load the completed metadata into the global environment
md <- load_metadata(path = md_path)
```

## Loading Data Sets

Loading the data is done using the function `load_datasets()`. This function takes 4 parameters:

- data_dir: This is the full path to the folder containing the files specified in the metadata. If the data is stored in a folder on the desktop titled ‘data’, then the path would look something like “C:/Users/name/Desktop/data”. Therefore, data_dir = “C:/Users/name/Desktop/data”.

- metadata: This is the completed metadata template that should now be loaded into the global environment from the function load_metadata(). To continue with the example from above, metadata = md.
- timestamp_var: This is the column name containing date and time. In the sample data provided with this package this column is named ‘TIMESTAMP’. Therefore, timestamp_var = “TIMESTAMP”. **All data sets must have a column with this name.**
- timestamp_format: This is the format of your date-time data. *If `load_datasets()` is used, keep the default for this parameter.* In the sample data the format is ‘Jan 01 2000-12:00:00’. Therefore, using this format, timestamp_format = “%b %d %Y-%H:%M:%S”. **All data sets must have identical timestamp formats.**

```
# build list of data sets based on provided directory and completed metadata
data_list <- load_datasets(data_directory, md)
```

## Launch the Dashboard

Now that both the metadata and the list of data sets has been loaded into the environment, the dashboard can be launched.

Typical User:

```
md <- load_metadata(path)
data_list <- load_datasets(data_directory, md)
dashboard(dt_list = data_list, metadata = md,
          timestamp_var = "TIMESTAMP",
          timestamp_format = "%b %d %Y-%H:%M:%S")
```

Developer:

****ensure `pv.ts.sandbox.Rproj` is the current, active project in RStudio****

```
library(devtools)
load_all()
md <- load_metadata(path)
data_list <- load_datasets(data_directory, md)
dashboard(dt_list = data_list, metadata = md,
          timestamp_var = "TIMESTAMP",
          timestamp_format = "%b %d %Y-%H:%M:%S")
```

# Dashboard Walkthrough

If using RStudio, the Shiny dashboard will load in a new native window. Otherwise, the dashboard will load in your default browser.

## Map Tab

This is the landing page of the dashboard.

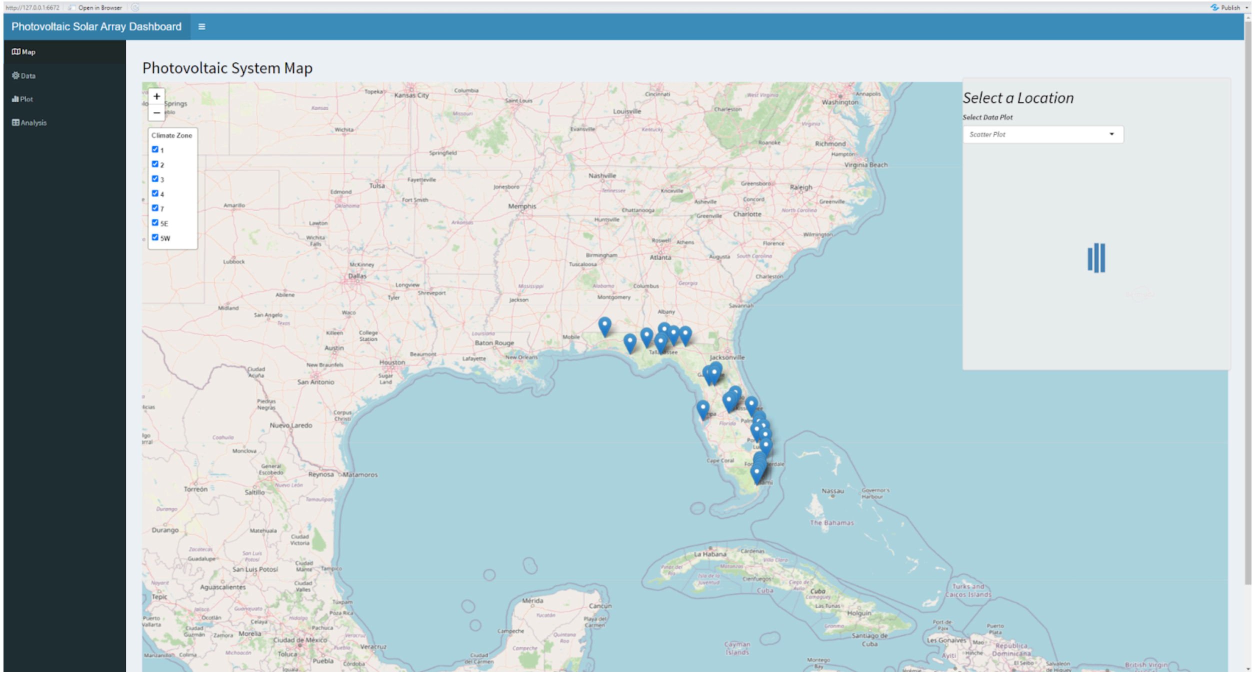


***Figure [#]**: R Dashboard Geospatial Visualization - Leaflet*

Each of the markers represents one of the data sets that were specified in the metadata. To view the name of the data set, hover over the marker.

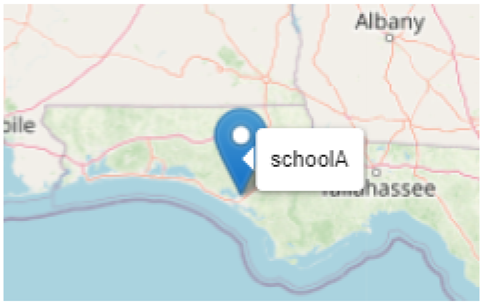
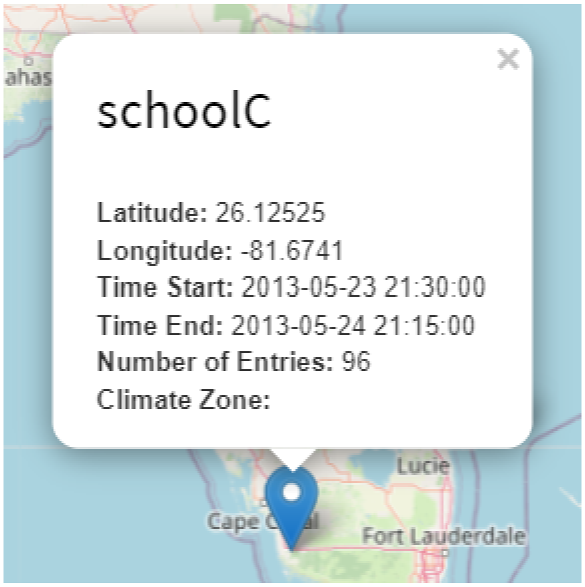


***Figure [#]**: R Dashboard Shiny Marker Hover Functionality*

Clicking on a marker reveals additional information about that data such as time range and number of rows.



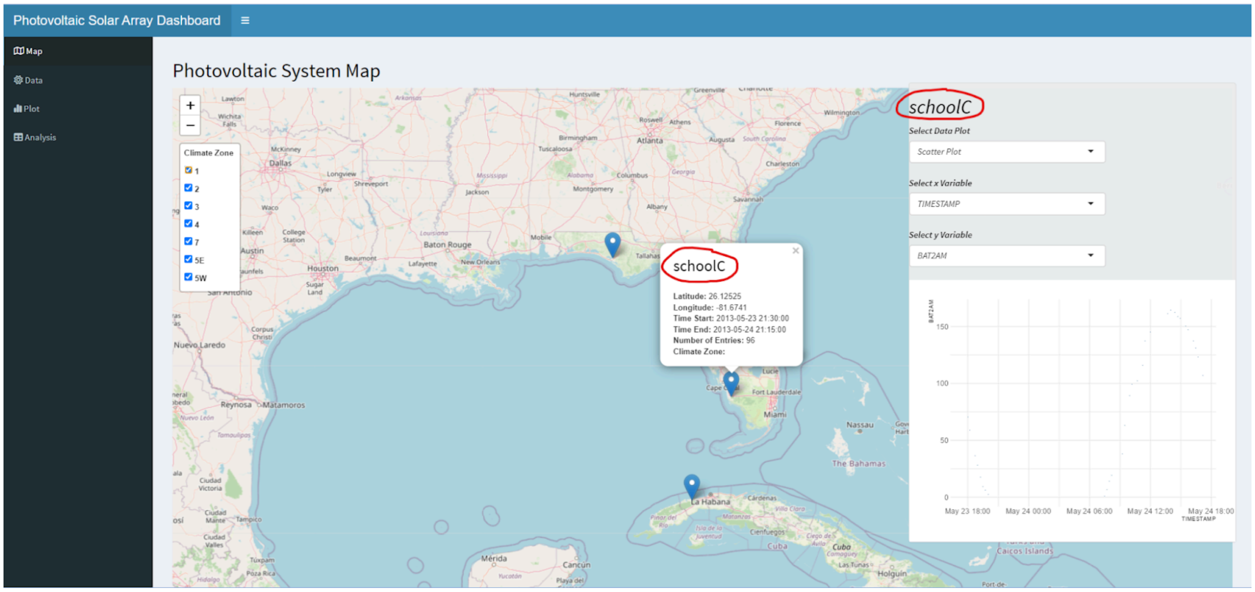
schoolC

**Latitude:** 26.12525
**Longitude:** -81.6741
**Time Start:** 2013-05-23 21:30:00
**Time End:** 2013-05-24 21:15:00
**Number of Entries:** 96
**Climate Zone:**

***Figure [#]***: *R Dashboard Shiny Marker Click Functionality*

## Quick Plots

Clicking on a marker also selects it for the plotting. Notice how now that a marker has been clicked, the name is also present in the plot window. To select a different data set, simply select a new marker.



Photovoltaic Solar Array Dashboard

Photovoltaic System Map

schoolC

Latitude: 26.12525
Longitude: -81.6741
Time Start: 2013-05-23 21:30:00
Time End: 2013-05-24 21:15:00
Number of Entries: 96
Climate Zone:

Select Data Plot: Scatter Plot

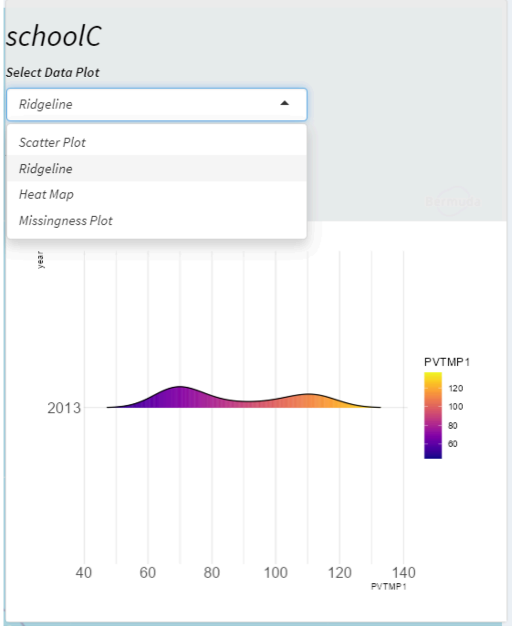
Select x Variable: TIMESTAMP

Select y Variable: BAT2AM

***Figure [#]***: *R Dashboard: Selecting a Location*

For scatter plots, each value is represented by one pixel as shown above. Both the x and y axis of the scatter plot can be changed with their respective drop down menus. The available selections will be the column names of the selected data.

The plot type can be changed with the 'Select Data Plot' drop down.



schoolC

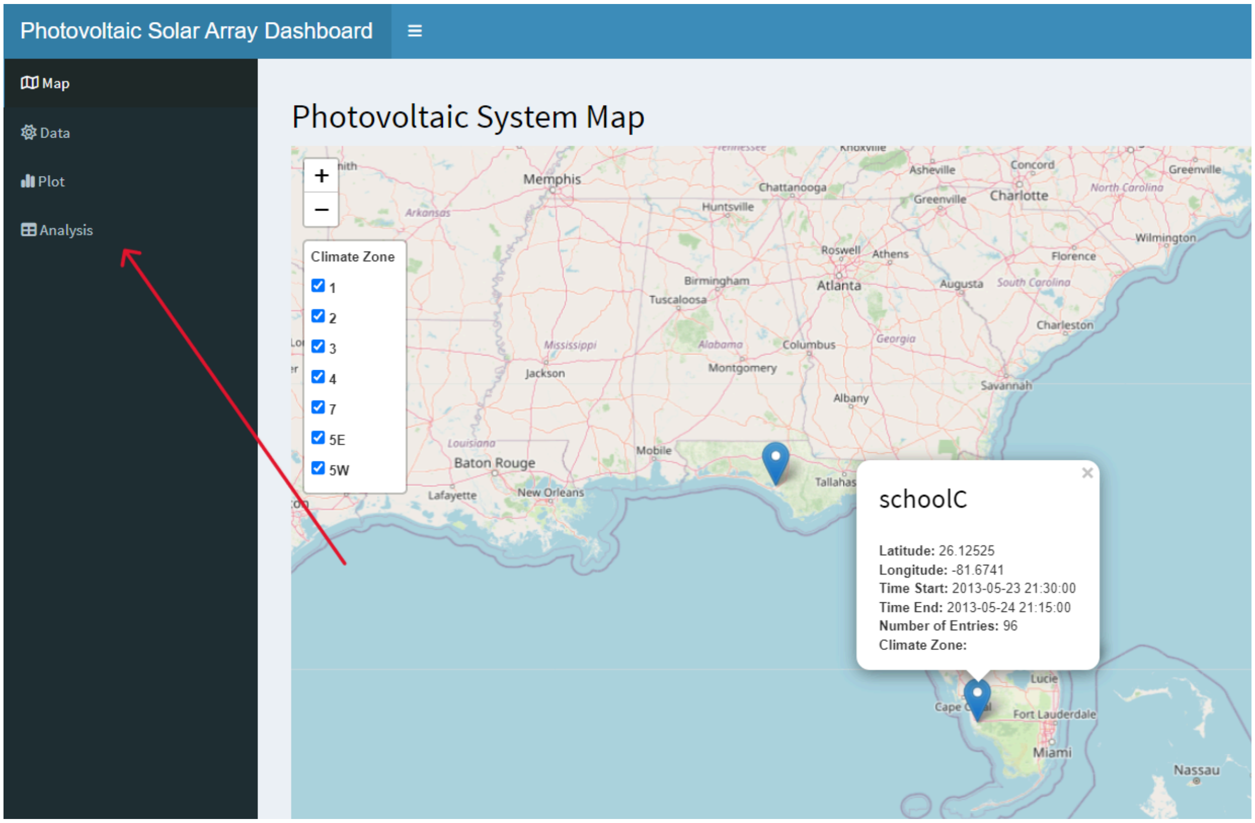
Select Data Plot: Ridgeline

***Figure [#]****: R Dashboard: Map Page, select plot type*

After selecting the ‘Ridgeline’ option, the scatter plot is replaced with a ridgeline plot. The x variable can be changed by selecting a different variable from the drop down menu. The y-axis for ridgeline plots is year (other options available in the ‘Plot’ tab).

## Data Tab

To navigate to a different page simply select it from the panel on the left.

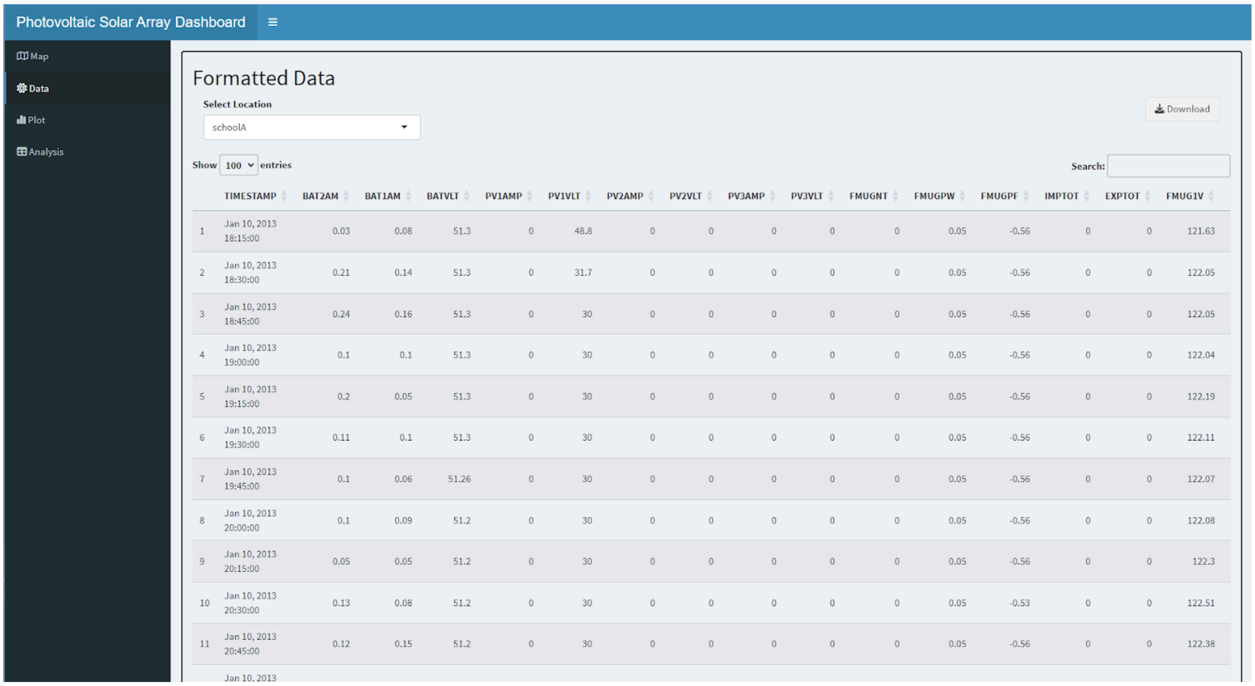


schoolC

**Latitude:** 26.12525
**Longitude:** -81.6741
**Time Start:** 2013-05-23 21:30:00
**Time End:** 2013-05-24 21:15:00
**Number of Entries:** 96
**Climate Zone:**

***Figure [#]****: R Dashboard: ShinyDashboard: Switch tabs*

Selecting ‘Data’ displays the data. At this point the type/class changes have been made to the data, all values are untouched.



Formatted Data

Select Location: schoolA

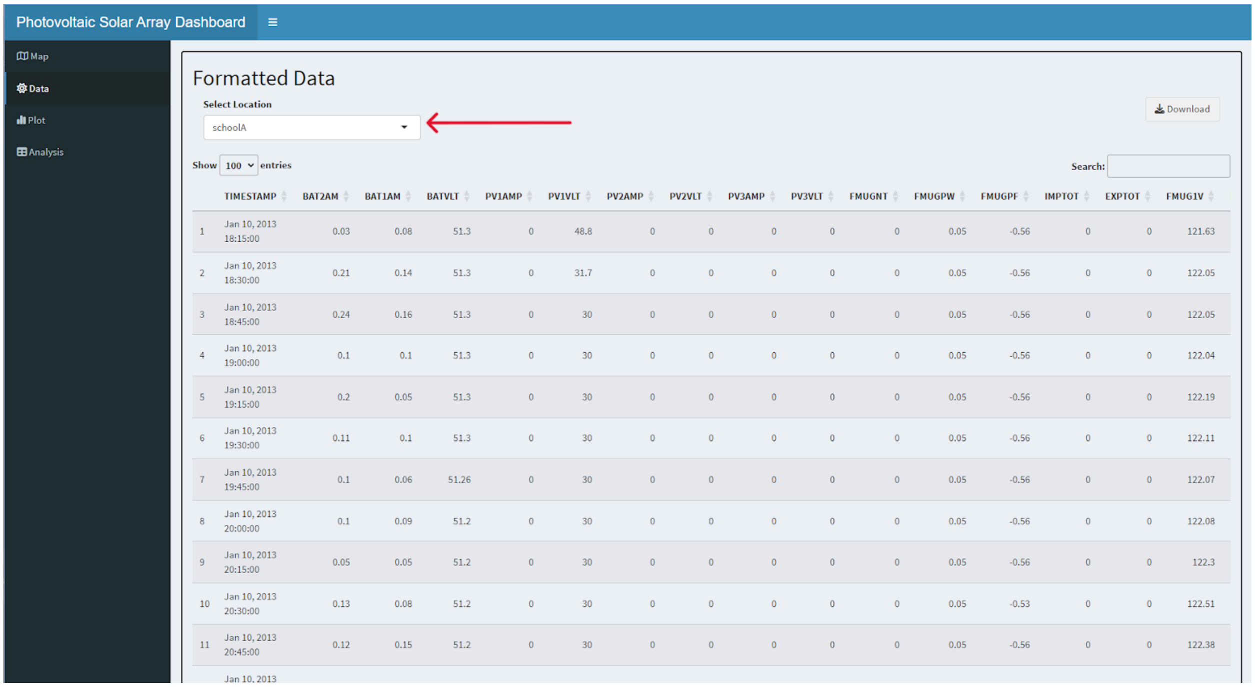
Show 100 entries

Search:

| | TIMESTAMP | BAT2AM | BAT1AM | BATVLT | PV1AMP | PV1VLT | PV2AMP | PV2VLT | PV3AMP | PV3VLT | FMUGNT | FMUGPW | FMUGPF | IMPTOT | EXPTOT | FMUG1V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Jan 10, 2013 18:15:00 | 0.03 | 0.08 | 51.3 | 0 | 48.8 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 121.63 |
| 2 | Jan 10, 2013 18:30:00 | 0.21 | 0.14 | 51.3 | 0 | 31.7 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.05 |
| 3 | Jan 10, 2013 18:45:00 | 0.24 | 0.16 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.05 |
| 4 | Jan 10, 2013 19:00:00 | 0.1 | 0.1 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.04 |
| 5 | Jan 10, 2013 19:15:00 | 0.2 | 0.05 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.19 |
| 6 | Jan 10, 2013 19:30:00 | 0.11 | 0.1 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.11 |
| 7 | Jan 10, 2013 19:45:00 | 0.1 | 0.06 | 51.26 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.07 |
| 8 | Jan 10, 2013 20:00:00 | 0.1 | 0.09 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.08 |
| 9 | Jan 10, 2013 20:15:00 | 0.05 | 0.05 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.3 |
| 10 | Jan 10, 2013 20:30:00 | 0.13 | 0.08 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.53 | 0 | 0 | 122.51 |
| 11 | Jan 10, 2013 20:45:00 | 0.12 | 0.15 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.38 |
| | Jan 10, 2013 | | | | | | | | | | | | | | | |

***Figure [#]**: R Dashboard: View data*

A different data set can be selected from the drop down menu and the data can be navigated using the 'Show # entries' menu just below, or via the search bar to the right.



Formatted Data

Select Location: schoolA

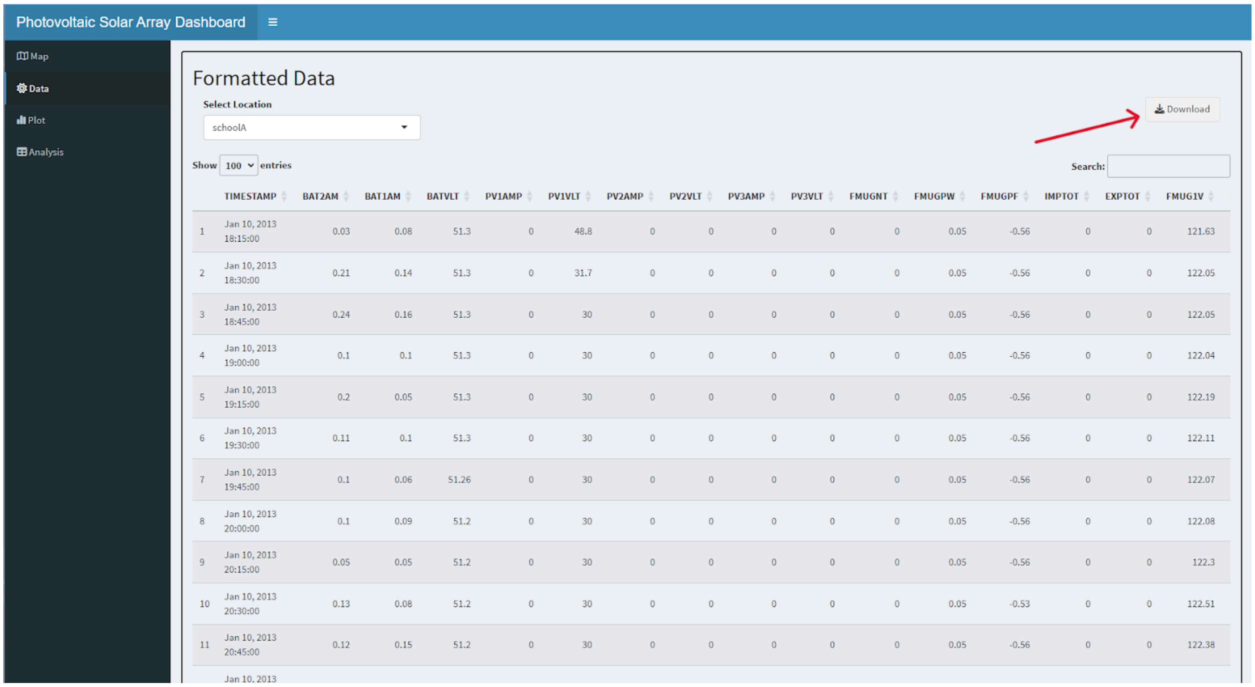
Show 100 entries

Search:

| | TIMESTAMP | BAT2AM | BAT1AM | BATVLT | PV1AMP | PV1VLT | PV2AMP | PV2VLT | PV3AMP | PV3VLT | FMUGNT | FMUGPW | FMUGPF | IMPTOT | EXPTOT | FMUG1V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Jan 10, 2013 18:15:00 | 0.03 | 0.08 | 51.3 | 0 | 48.8 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 121.63 |
| 2 | Jan 10, 2013 18:30:00 | 0.21 | 0.14 | 51.3 | 0 | 31.7 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.05 |
| 3 | Jan 10, 2013 18:45:00 | 0.24 | 0.16 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.05 |
| 4 | Jan 10, 2013 19:00:00 | 0.1 | 0.1 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.04 |
| 5 | Jan 10, 2013 19:15:00 | 0.2 | 0.05 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.19 |
| 6 | Jan 10, 2013 19:30:00 | 0.11 | 0.1 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.11 |
| 7 | Jan 10, 2013 19:45:00 | 0.1 | 0.06 | 51.26 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.07 |
| 8 | Jan 10, 2013 20:00:00 | 0.1 | 0.09 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.08 |
| 9 | Jan 10, 2013 20:15:00 | 0.05 | 0.05 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.3 |
| 10 | Jan 10, 2013 20:30:00 | 0.13 | 0.08 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.53 | 0 | 0 | 122.51 |
| 11 | Jan 10, 2013 20:45:00 | 0.12 | 0.15 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.38 |
| | Jan 10, 2013 | | | | | | | | | | | | | | | |

*__Figure [#]__: R Dashboard: Change viewed dataset*

The download button allows you to save the current data. The download button will open the download manager for the system. From there, the file name and directory can be edited as usual.



Formatted Data

Select Location: schoolA

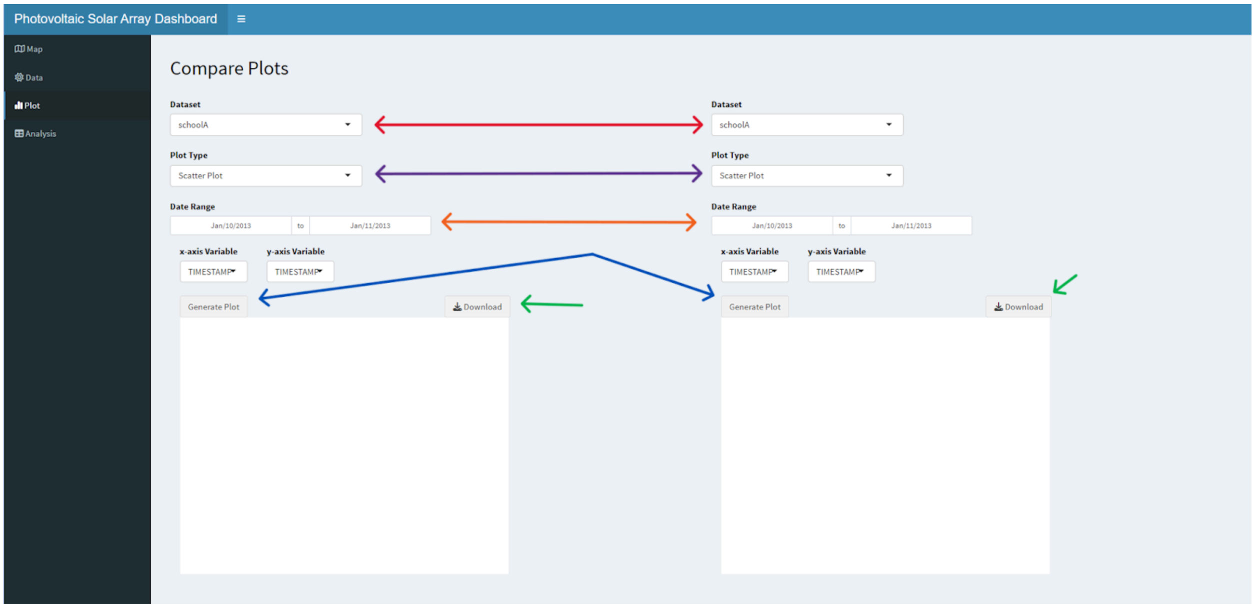
Show 100 entries

| | TIMESTAMP | BAT2AM | BAT1AM | BATVLT | PV1AMP | PV1VLT | PV2AMP | PV2VLT | PV3AMP | PV3VLT | FMUGNT | FMUGPW | FMUGPF | IMPTOT | EXPTOT | FMUG1V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Jan 10, 2013 18:15:00 | 0.03 | 0.08 | 51.3 | 0 | 48.8 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 121.63 |
| 2 | Jan 10, 2013 18:30:00 | 0.21 | 0.14 | 51.3 | 0 | 31.7 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.05 |
| 3 | Jan 10, 2013 18:45:00 | 0.24 | 0.16 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.05 |
| 4 | Jan 10, 2013 19:00:00 | 0.1 | 0.1 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.04 |
| 5 | Jan 10, 2013 19:15:00 | 0.2 | 0.05 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.19 |
| 6 | Jan 10, 2013 19:30:00 | 0.11 | 0.1 | 51.3 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.11 |
| 7 | Jan 10, 2013 19:45:00 | 0.1 | 0.06 | 51.26 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.07 |
| 8 | Jan 10, 2013 20:00:00 | 0.1 | 0.09 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.08 |
| 9 | Jan 10, 2013 20:15:00 | 0.05 | 0.05 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.3 |
| 10 | Jan 10, 2013 20:30:00 | 0.13 | 0.08 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.53 | 0 | 0 | 122.51 |
| 11 | Jan 10, 2013 20:45:00 | 0.12 | 0.15 | 51.2 | 0 | 30 | 0 | 0 | 0 | 0 | 0 | 0.05 | -0.56 | 0 | 0 | 122.38 |
| | Jan 10, 2013 | | | | | | | | | | | | | | | |

*__Figure [#]__: R Dashboard: Download dataset*

# Plot Tab

Selecting ’Plot” from the sidebar will allow the comparison of data sets side-by-side.



Compare Plots

Dataset: schoolA | Dataset: schoolA

Plot Type: Scatter Plot | Plot Type: Scatter Plot

Date Range: Jan/10/2013 to Jan/11/2013 | Date Range: Jan/10/2013 to Jan/11/2013

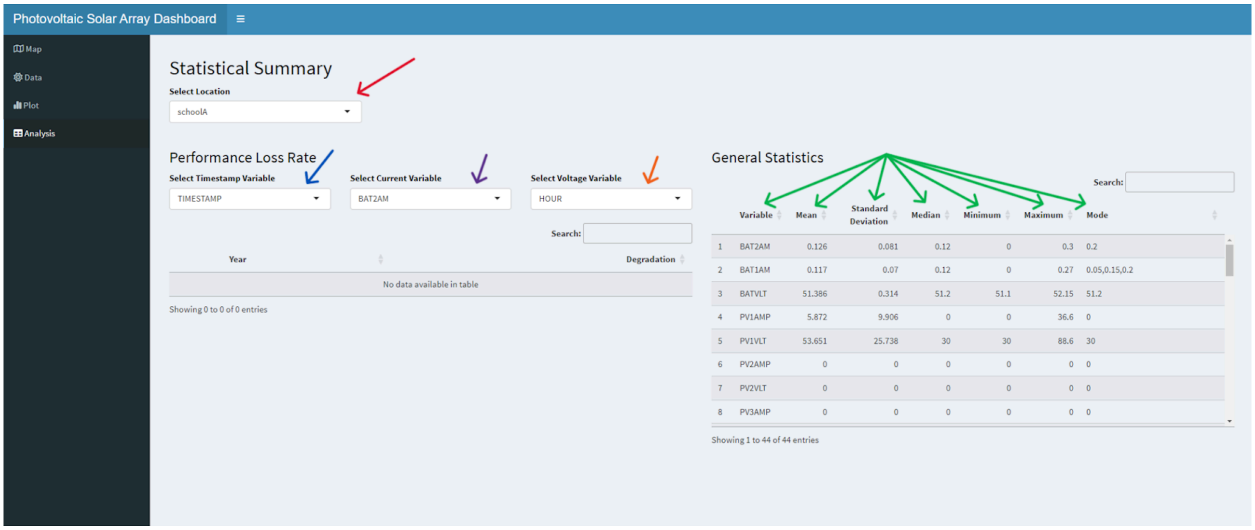
x-axis Variable: TIMESTAMP | y-axis Variable: TIMESTAMP | x-axis Variable: TIMESTAMP | y-axis Variable: TIMESTAMP

***Figure [#]**: R Dashboard: Plots Page, comparison of 2 datasets*

Select the data sets to compare with the 'Dataset' drop down menu (red). This allows the comparison across data sets, or within data sets. Select the plot type (purple). These options are identical to those from the 'Map' tab. This tab allows for manipulation of the time frame of the data with the 'Date Range' option (orange). Currently selections are only possible in full day increments. Once all desired selections have been made, render the plot with the 'Generate Plot' buttons (blue). These plots can be downloaded for external use via the 'Download' buttons (green). This will open the download manager used by the system. From there, the file name and directory can be edited as usual.

# Analysis Tab

This tab displays the relevant statistics of the selected data.



Statistical Summary

Select Location: schoolA

Performance Loss Rate

Select Timestamp Variable: TIMESTAMP

Select Current Variable: BAT2AM

Select Voltage Variable: HOUR

Search:

| Year | Degradation |
|---|---|
| No data available in table | |

Showing 0 to 0 of 0 entries

General Statistics

Search:

| | Variable | Mean | Standard Deviation | Median | Minimum | Maximum | Mode |
|---|---|---|---|---|---|---|---|
| 1 | BAT2AM | 0.126 | 0.081 | 0.12 | 0 | 0.3 | 0.2 |
| 2 | BAT1AM | 0.117 | 0.07 | 0.12 | 0 | 0.27 | 0.05,0.15,0.2 |
| 3 | BATVLT | 51.386 | 0.314 | 51.2 | 51.1 | 52.15 | 51.2 |
| 4 | PV1AMP | 5.872 | 9.906 | 0 | 0 | 36.6 | 0 |
| 5 | PV1VLT | 53.651 | 25.738 | 30 | 30 | 88.6 | 30 |
| 6 | PV2AMP | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | PV2VLT | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | PV3AMP | 0 | 0 | 0 | 0 | 0 | 0 |

Showing 1 to 44 of 44 entries

***Figure [#]****: R Dashboard: Analysis Page. Statistically Summary & Performance Loss Rate Analysis*

The data to summarize can be changed with the 'Select Location' menu (red). Performance Loss Rate (PLR) analysis is calculated based on the selected parameter (blue, purple, orange). 'Timestamp' (blue) is initially filled with the first POSIXt column. 'Current' (purple) is initially filled with the first numeric column. 'Voltage' is initially filled with the last numeric column and the drop-down options are reversed. Once selections have been made, the table will populate. PLR calculations are performed using a twist on the year-over-year degradation method. Traditionally:

```
# Traditional degradation rate
degradation <- 100 * ((totalPower_year2 - totalPower_year1) / totalPower_year1)
```

In this package degradation is first calculated on a monthly scale, and then condensed by taking the mean of month grouped by year.

The 'General Statistics' section (green) displays the common summary statistics, mean, median, mode, standard deviation, minimum, and maximum, for all numeric columns in the data.

## Performance Loss Rate (Year-Over-Year Degradation)

Degradation is defined as the loss of power produced related to the rated power. In order to find the annual degradation rate, we must first find the annual power output of the photovoltaic system. In these packages this was done by simply multiplying current and voltage

$$P = IV$$

As a starting point for this calculation, we assumed all other variables are held constant. Specifically, this means there was no degradation in any other components of the system and that weather and irradiance were constant for the entire year. This allowed us to performance a simple percent difference calculation, following the formula:

$$DegradationYear_n = (100 * (TotalPowerOutYear_{n-1} - TotalPowerOutYear_n))/TotalPowerOutYear_{n-1}$$

Realistically, other components of the photovoltaic system will wear down over time, and the weather and irradiance will not remain constant for a full year. However, this formula is a quick and efficient calculation that determines if a photovoltaic system is performing within the manufacturer's guaranteed range, or if further data collection is required to perform a more in depth analysis.