Solcast / **howto-pandas**

How-To Solcast - Python, Pandas, MatPlotLib, and PvLib

#pandas #radiation #data-frame #howto #matplotlib #forecast #python #pvlib #solar-energy

29 commits | 1 branch | 0 releases | 1 contributor | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Siliconrob Update README.md — Latest commit d8d59c8 on Oct 25, 2017

imgs	Add sample code fragments	a year ago
solcast_frames	Add sample code fragments	a year ago
tests	Removed Anchorage as timezone point	a year ago
.gitignore	Add in python code	a year ago
.travis.yml	Build configs	a year ago
LICENSE	Add in python code	a year ago
README.md	Update README.md	a year ago
appveyor.yml	Force install pandas module	a year ago
main.ipynb	added notebook	a year ago
main.py	Typos and wording changes	a year ago
pytest.ini	Add start of pytests	a year ago
requirements.linux.txt	Add travis.yml without numpy+mkl windows binary wheel file	a year ago
requirements.txt	Consolidate solcast and pvlib code to main.py	a year ago

README.md

HowTo Solcast - Python, Pandas, MatPlotLib, and PvLib

NOTE: You will need to register your user account to obtain an API key https://solcast.com.au/api/register. Without an API key you will not be able to successfully obtain valid API results.

Windows setup Solcast API Key

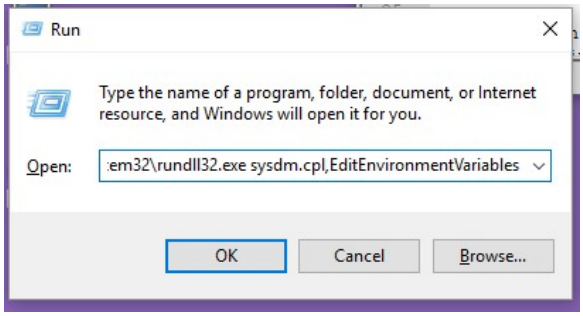
Setup System/User `environment variable` . Details on advanced editing StackOverflow superuser walkthrough

```
WinKey + R
```

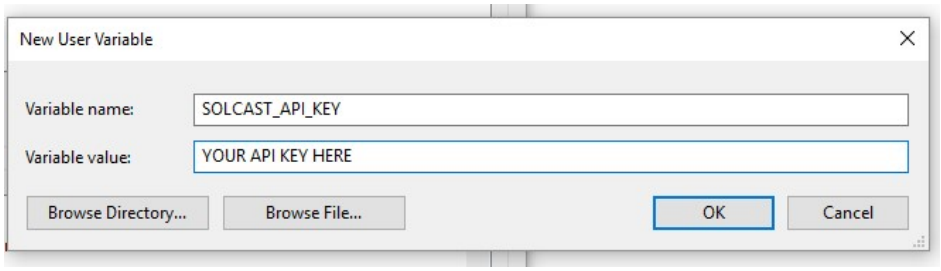
Copy and Paste the following text to the **Open:** input text box for the Run Dialog

```
%windir%\System32\rundll32.exe sysdm.cpl,EditEnvironmentVariables
```

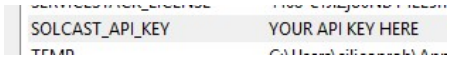
That will present this screen



Add a user or system `environment variable` to hold the Solcast API key. User environment variables will only be available to your particular user, system environment variables are shared for all users on the system



After you have added the `environment variable` you will see the key listed in the current variables



NOTE: To reference this key you will need to reopen your shell prompt to read these variables again from the system (cmd, command.com, powershell, etc)

Linux / mac OS

Open a terminal prompt

- mac OS: Spotlight search for `terminal`
- Linux: Open `bash`

```
nano .bash_profile
```

If you do not have nano it is simpler text editor than `vi` . Use your package manager to download and install or use `vi` . The preferred package manager for mac OS is Homebrew and once installed on your system you can issue similar commands to Linux `apt-get` and `yum` with the `brew` package manager.

Add the Solcast API Key to your user profile variables.

```
 GNU nano 2.0.6                    File: .bash_profile

alias ll='ls -lG'
alias updatedb="sudo /usr/libexec/locate.updatedb"

export PATH="/usr/local/bin:$PATH"
export GOPATH=$HOME/golang
export GOROOT=/usr/local/go
export PATH=$PATH:$GOPATH/bin
export LDFLAGS=-L/usr/local/opt/openssl/lib
export CPPFLAGS=-I/usr/local/opt/openssl/include
export PKG_CONFIG_PATH=/usr/local/opt/openssl/lib/pkgconfig
export SOLCAST_API_KEY=YOUR API KEY HERE
```

This how to demonstrates working with using the Solcast API to load pandas data frames and then plot different chart data. Tested and built with Python 3.6.2

This is a demonstration project that shows an example of how to use the solcast-py library to load a Pandas data frame and then display a single field graphically using MatPlotLib. A familiarity with Python is not required.

Clone or download this project

Install all the require dependencies

If you are not familiar with the python module pip, please refer to this PIP for Python

```
pip install --upgrade -r requirements.txt
```

Alternatively you may need to use sudo the `-H` instructs `sudo` to install dependencies to local users home directory instead of system

```
sudo -H pip install --upgrade -r requirements.txt
```

If you are receiving an error when installing the netcdf4 package on Mac such as:

```
ValueError: did not find netCDF version 4 headers
```

or

```
ValueError: did not find hdf5 version 4 headers
```

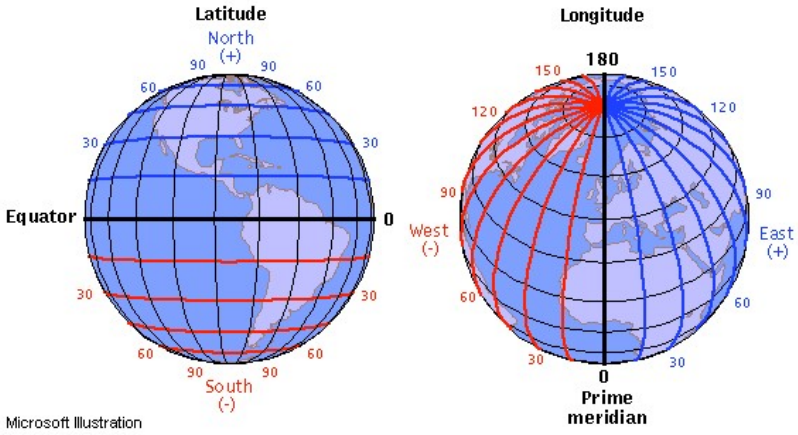
These two packages need to be installed onto your Mac with homebrew. Click this link to install homebrew here. These packages are located in the homebrew/science section. You will need to "tap" this section to access it. More information is found here.

NOTE: Installing hdf5 on mac will take **a long time**. Give it time.

Latitude and Longitude

- First as stated above you will need an API key to make valid API requests to the Solcast system.
- Second for all current library calls you will need a valid Lat/Lng coordinate in the EPSG:4326 format. If you are familiar with modern web maps you most likely have used the expected format or a decimal point that expresses a position on the Earth.

Clarification as I often forget the coordinate planes of Latitude and Longitude along with bounds.



Credits - Learner.org

The Solcast API expects **West** for Longitude and **South** for Latitude to be expressed as a negative numbers

Example Locations on the Globe

Name	Latitude	Longitude
Sydney, Australia	-33.865143	151.209900
Mumbai, India	19.228825	72.854118
Tokyo, Japan	35.6895	139.69171
Paris, France	48.864716	2.349014
Los Angeles, USA	34.052235	-118.243683

Power/Radiation library method interaction

In the file **main.py** is a minimal set of commands to obtain the following charts. The break in the continuity between the **Estimated Actuals** and the **Forecast** predicted values on the time scale is due to that the time period falls between an in place Estimated Actual result being recorded and a next period available Forecast. Currently all period results are broken into **30** minute timestamp periods. This howto uses the `period_end` fields as the indexed field for the DataFrames (x - axis on the following charts)

Within this project workspace is folder `solcast_frames` that provides some `solcast-py` library helper translators to `pandas` DataFrames. In the `main.py` file these helper classes are referenced below.

```
# This line is only for using the matplotlib
import matplotlib.pyplot as plt
from solcast_frames.latlng import LatLng
from solcast_frames.radiationframehandler import RadiationFrameHandler
from solcast_frames.powerframehandler import PowerFrameHandler

plt.interactive(False) # Turn this off to create plots
```

These following python classes are now available to use `LatLng` , `RadiationFrameHandler` , `PowerFrameHandler`

First you should create a location to obtain results for solcast forecasts. Example below.

```
location = LatLng(lat=-33.86785, lng=151.215256, name="Sydney", tag="No wild Koalas", timezone="Australia/Sydney")
```

The `LatLng` helper class if no keyword arguments are supplied will set your location to (0,0) on the globe

The `LatLng` class contains a `desc()` method to give you a string output of its current details if needed i.e.
`print(location.desc())`

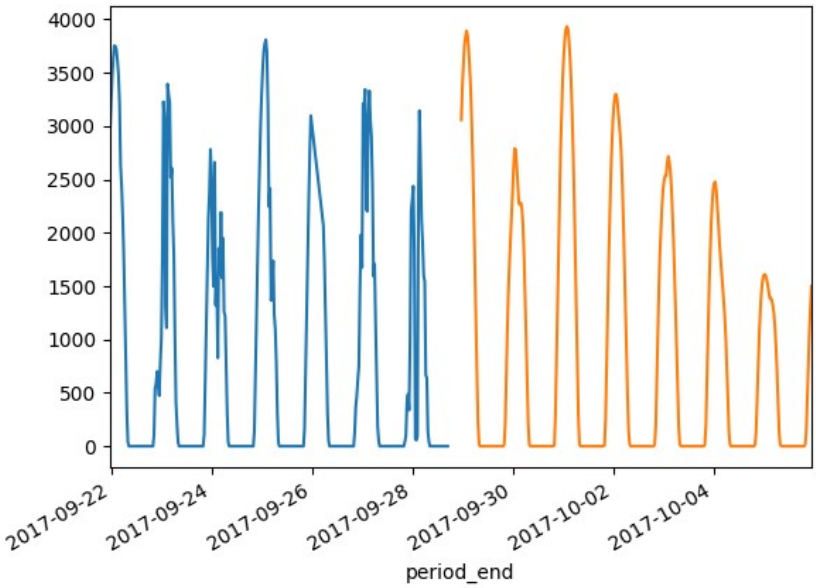
To obtain and plot the current estimates and forecast from the Solcast API for **PvPower** use the following code

```
# Plot the power `pv_estimate`
# Required fields latlng valid position, capacity as integer > 0
# Adding in an optional keyword argument of an azimuth of 0
# The following optional keyword arguments are recognized by the solcast-py library
# azimuth | range: [-180 to 180] default: 0 in Southern Hemisphere, 180 in Northern Hemisphere
# tilt | range: [0 to 90] | default: 23
# install_date | format: yyyyMMdd Will be ignored if a loss_factor is supplied
# latest | [True, False] | default False
power_estimated_actuals = PowerFrameHandler.estimated_actuals(location, 5000, azimuth=0)
power_estimated_actuals.pv_estimate.plot()

# Plot the power `pv_estimate`
# Required fields latlng valid position, capacity as integer > 0
# The following optional keyword arguments are recognized by the solcast-py library
# azimuth | range: [-180 to 180] default: 0 in Southern Hemisphere, 180 in Northern Hemisphere
# tilt | range: [0 to 90] | default: 23
```

```
# install_date | format: yyyyMMdd Will be ignored if a loss_factor is supplied
# loss_factor | [0 to 1] | default 0.9
fx_solcast_power = PowerFrameHandler.forecast(location, 5000)
fx_solcast_power.pv_estimate.plot()
plt.show()
```

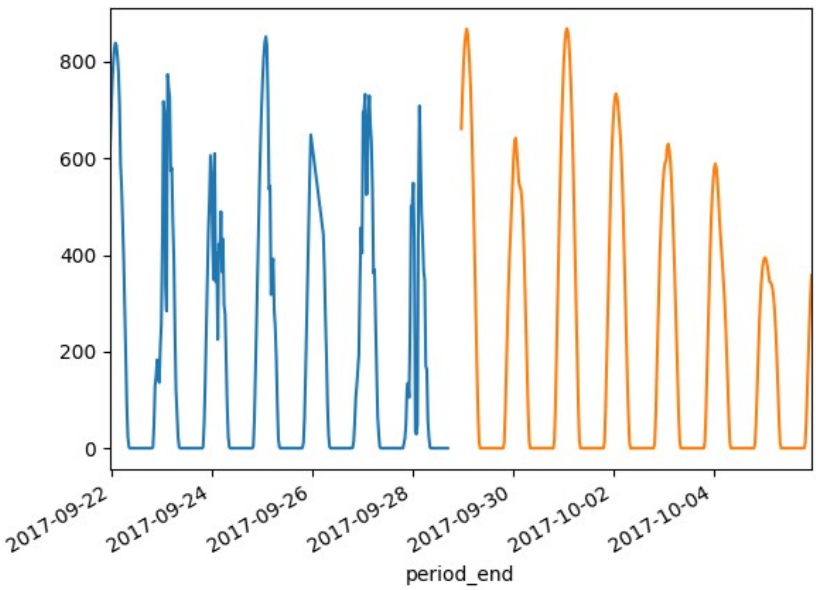
Power `Estimated Actuals` with `Forecast ( pv_estimate` over `period_end )`



To obtain and plot the current estimates and forecast from the Solcast API for **Radiation** use the following code

```
# Plot the radiation `ghi` field
# Required fields latlng valid position
# The following optional keyword arguments are recognized by the solcast-py library
# latest | [True, False] | default False
radiation_estimated_actuals = RadiationFrameHandler.estimated_actuals(location)
radiation_estimated_actuals.ghi.plot()
# Plot the radiation `ghi` field
# Required fields latlng valid position
fx_solcast_radiation = RadiationFrameHandler.forecast(location)
fx_solcast_radiation.ghi.plot()
plt.show()
```

Radiation `Estimated Actuals` with `Forecast` (`ghi` over `period_end`)



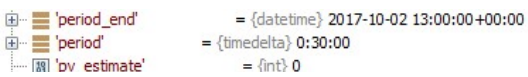
Details of Solcast Python Client Library Solcast Python API client library

The Solcast Python library is designed as a synchronous web service and does not include the graphing or advanced data options of Pandas/MatPlotLib.

The `PowerFrameHandler.py` and `RadiationFrameHandler.py` classes in this howto under `solcast_frames` example convert the list value resultsets into `pandas` dataframes aligned and indexed by the `period_end` DateTime field (all datetimes are expressed in UTC timezone).

Example data results for Power DataFrames with columns available:

Power DataType

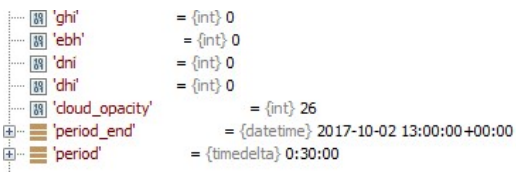


Power DataFrame contents

```
                           period  pv_estimate
period_end
2017-10-02 19:30:00+00:00 00:30:00     0.000000
2017-10-02 20:00:00+00:00 00:30:00    27.628064
2017-10-02 20:30:00+00:00 00:30:00   243.625359
```

Example data results for Radiation DataFrames with columns available:

Radiation Estimated Actuals DataType



Estimated Actuals Radiation DataFrame contents

```
                           cloud_opacity  dhi  dni  ebh  ghi    period
period_end
2017-10-02 13:00:00+00:00             26    0    0    0    0 00:30:00
2017-10-02 12:30:00+00:00             31    0    0    0    0 00:30:00
2017-10-02 12:00:00+00:00              0    0    0    0    0 00:30:00
```

Radiation Forecasts DataType

Forecast Radiation DataFrame contents

```
                           air_temp  azimuth  cloud_opacity  dhi  dni  dni10  dni90  ebh  ghi  ghi10  ghi90
period  zenith
period_end
2017-10-02 19:30:00+00:00        17      -97             17    1    0      0      0    0    1      1      1
00:30:00      94
2017-10-02 20:00:00+00:00        17      -93             33   13   10      4     27    2   15     12     19
00:30:00      88
2017-10-02 20:30:00+00:00        17      -88              7   45  196     43    275   33   79     57     86
00:30:00      82
```

Integrating with pvlib

In the file **main.py** is a minimal set of commands to obtain the following charts. The pvlib library is a powerful open source tool that is used in computing and forecasting photovoltaic solar cell arrays. There are numerous options and details that can be configured, the following example compares pvlib forecasts with Solcast forecasts and how to obtain datasets that can be compared and graphed.

```
# This line is only for using the matplotlib
import matplotlib.pyplot as plt
from solcast_frames.latlng import LatLng
from solcast_frames.radiationframehandler import RadiationFrameHandler
import timeit
import pandas as pd
import datetime
# import pvlib forecast models
from pvlib.forecast import GFS, NAM, NDFD, HRRR, RAP

plt.interactive(False) # Turn this off to create plots

#
# Following code is from http://pvlib-python.readthedocs.io/en/latest/forecasts.html
#

# Changing the location pvlib default model uses Tucson for examples
location = LatLng(lat=32.2, lng=-110.9, name="Tucson", tag="Cactus Land", timezone="US/Arizona")
print(location.desc())

start_time = timeit.default_timer()
radiationForecast = RadiationFrameHandler.forecast(location)
```

```
elapsed = timeit.default_timer() - start_time
print("Solcast Radiation Forecast Location: %s Time: %s (seconds)" % (location.name, '%.6f' % elapsed))

# specify time range with timezone
start = pd.Timestamp(datetime.date.today(), tz=location.timezone)
end = start + pd.Timedelta(days=7)

start_time = timeit.default_timer()
# fx is a common abbreviation for forecast
fx_model = GFS() # From Forecast models http://pvlib-python.readthedocs.io/en/latest/api.html#forecast-models
fx_data = fx_model.get_processed_data(location.lat, location.lng, start, end)
elapsed = timeit.default_timer() - start_time
print("pvlib (GFS) Radiation Forecast Location: %s Time: %s (seconds)" % (location.name, '%.6f' % elapsed))


plt.plot(fx_data.ghi, label="ghi - PVLIB (GFS)")
plt.plot(radiationForecast.ghi, label="ghi - Solcast", linestyle='dashdot')

plt.plot(fx_data.dhi, label="dhi - PVLIB (GFS)")
plt.plot(radiationForecast.dhi, label="dhi - Solcast", linestyle='dashdot')

plt.plot(fx_data.dni, label="dni - PVLIB (GFS)")
plt.plot(radiationForecast.dni, label="dni - Solcast", linestyle='dashdot')

plt.legend()
plt.show()
```

Radiation Forecast **pvlib** vs **Solcast** `ghi` `dhi` , `dni`