Home (/) » Data Sets (/data-sets) » API Instructions

API Instructions

NSRDB and SAM Python APIs: Automated download of resource data and SAM simulation

- NSRDB Website: https://nsrdb.nrel.gov (https://nsrdb.nrel.gov)
- Get NSRDB API Key: https://developer.nrel.gov/signup/ (https://developer.nrel.gov/signup/)
- Register to use SAM: https://sam.nrel.gov (https://sam.nrel.gov)
- Download the SAM Software Development Kit (SDK): https://sam.nrel.gov/sdk (https://sam.nrel.gov/sdk)

This example shows how to use the NSRDB API for automated data download in Section 1 and then provides an example of using these data with the SAM Software Development Kit (SDK) in Section 2. A plotting example is offered in Section 3.

In [1]:

```
import pandas as pd
import numpy as np
import sys, os
from IPython.display import display
```

1. Request Data From NSRDB using Python API

The following section shows how to download NSRDB data for a specified year and location.

Declare input variables for api request:

In [4]:
```
# Declare all variables as strings. Spaces must be replaced
 with '+', i.e., change 'John Smith' to 'John+Smith'.
# Define the lat, long of the location and the year
lat, lon, year = 33.2164, -97.1292, 2010
# You must request an NSRDB api key from the link above
api_key = 'yourKey'
# Set the attributes to extract (e.g., dhi, ghi, etc.), sepa
rated by commas.
attributes = 'ghi,dhi,dni,wind_speed,air_temperature,solar_z
enith_angle'
# Choose year of data
year = '2010'
# Set leap year to true or false. True will return leap day
 data if present, false will not.
leap_year = 'false'
# Set time interval in minutes, i.e., '30' is half hour inte
rvals. Valid intervals are 30 & 60.
interval = '30'
# Specify Coordinated Universal Time (UTC), 'true' will use
 UTC, 'false' will use the local time zone of the data.
# NOTE: In order to use the NSRDB data in SAM, you must spec
ify UTC as 'false'. SAM requires the data to be in the
# local time zone.
utc = 'false'
# Your full name, use '+' instead of spaces.
your_name = 'John+Smith'
# Your reason for using the NSRDB.
reason_for_use = 'beta+testing'
# Your affiliation
your_affiliation = 'my+institution'
# Your email address
your_email = 'john.smith@server.com'
# Please join our mailing list so we can keep you up-to-date
on new developments.
mailing_list = 'true'

# Declare url string
url = 'http://developer.nrel.gov/api/solar/nsrdb_psm3_downlo
ad.csv?wkt=POINT({lon}%20{lat})&names={year}&leap_day={leap}
&interval={interval}&utc={utc}&full_name={name}&email={emai
l}&affiliation={affiliation}&mailing_list={mailing_list}&rea
son={reason}&api_key={api}&attributes={attr}'.format(year=ye
ar, lat=lat, lon=lon, leap=leap_year, interval=interval, utc
=utc, name=your_name, email=your_email, mailing_list=mailing
_list, affiliation=your_affiliation, reason=reason_for_use,
api=api_key, attr=attributes)
# Return just the first 2 lines to get metadata:
info = pd.read_csv(url, nrows=1)
# See metadata for specified properties, e.g., timezone and
 elevation
timezone, elevation = info['Local Time Zone'], info['Elevati
on']
```

In [5]:

```
# View metadata
info
```

Out[5]:

	Source	Location ID	City	State	Country	Latitude	Longitude
0	NSRDB	680780	-	-	-	33.21	-97.14

1 rows × 46 columns

In [6]:
```
# Return all but first 2 lines of csv to get data:
df = pd.read_csv('http://developer.nrel.gov/api/solar/nsrdb_psm3_download.csv?wkt=POINT({lon}%20{lat})&names={year}&leap_day={leap}&interval={interval}&utc={utc}&full_name={name}&email={email}&affiliation={affiliation}&mailing_list={mailing_list}&reason={reason}&api_key={api}&attributes={attr}'.format(year=year, lat=lat, lon=lon, leap=leap_year, interval=interval, utc=utc, name=your_name, email=your_email, mailing_list=mailing_list, affiliation=your_affiliation, reason=reason_for_use, api=api_key, attr=attributes), skiprows=2)

# Set the time index in the pandas dataframe:
df = df.set_index(pd.date_range('1/1/{yr}'.format(yr=year), freq=interval+'Min', periods=525600/int(interval)))

# take a look
print 'shape:',df.shape
df.head()
```

```
shape: (17520, 22)
```

Out[6]:

	Year	Month	Day	Hour	Minute	Clearsky DHI	Clearsky DNI
2010-01-01 00:00:00	2010	1	1	0	0	0	0
2010-01-01 00:30:00	2010	1	1	0	30	0	0
2010-01-01 01:00:00	2010	1	1	1	0	0	0
2010-01-01 01:30:00	2010	1	1	1	30	0	0
2010-01-01 02:00:00	2010	1	1	2	0	0	0

5 rows × 22 columns

In [7]:
```
# Print column names
print df.columns.values
```

```
['Year' 'Month' 'Day' 'Hour' 'Minute' 'Clearsky DHI' 'Clear
sky DNI'
 'Clearsky GHI' 'Cloud Type' 'Dew Point' 'DHI' 'DNI' 'Fill
Flag' 'GHI'
 'Snow Depth' 'Solar Zenith Angle' 'Temperature' 'Pressure'
 'Relative Humidity' 'Precipitable Water' 'Wind Direction'
'Wind Speed']
```

2. SAM Simulation

The following illustrates how to use the NSRDB data in a SAM simulation, using the Python Software Development Kit (SDK).

In [8]:
```
#import additional module for SAM simulation:
import site
# Use site.addsitedir() to set the path to the SAM SDK API.
 Set path to the python directory.
site.addsitedir('/Applications/sam-sdk-2015-6-30-r3/language
s/python/')
import sscapi
ssc = sscapi.PySSC()

# Resource inputs for SAM model:
wfd = ssc.data_create()
ssc.data_set_number(wfd, 'lat', lat)
ssc.data_set_number(wfd, 'lon', lon)
ssc.data_set_number(wfd, 'tz', timezone)
ssc.data_set_number(wfd, 'elev', elevation)
ssc.data_set_array(wfd, 'year', df.index.year)
ssc.data_set_array(wfd, 'month', df.index.month)
ssc.data_set_array(wfd, 'day', df.index.day)
ssc.data_set_array(wfd, 'hour', df.index.hour)
ssc.data_set_array(wfd, 'minute', df.index.minute)
ssc.data_set_array(wfd, 'dn', df['DNI'])
ssc.data_set_array(wfd, 'df', df['DHI'])
ssc.data_set_array(wfd, 'wspd', df['Wind Speed'])
ssc.data_set_array(wfd, 'tdry', df['Temperature'])

# Create SAM compliant object
dat = ssc.data_create()
ssc.data_set_table(dat, 'solar_resource_data', wfd)
ssc.data_free(wfd)

# Specify the system Configuration
# Set system capacity in MW
system_capacity = 4
ssc.data_set_number(dat, 'system_capacity', system_capacity)
# Set DC/AC ratio (or power ratio). See https://sam.nrel.go
v/sites/default/files/content/virtual_conf_july_2013/07-sam-
virtual-conference-2013-woodcock.pdf
ssc.data_set_number(dat, 'dc_ac_ratio', 1.1)
# Set tilt of system in degrees
```

```
ssc.data_set_number(dat, 'tilt', 25)
# Set azimuth angle (in degrees) from north (0 degrees)
ssc.data_set_number(dat, 'azimuth', 180)
# Set the inverter efficency
ssc.data_set_number(dat, 'inv_eff', 96)
# Set the system losses, in percent
ssc.data_set_number(dat, 'losses', 14.0757)
# Specify fixed tilt system (0=Fixed, 1=Fixed Roof, 2=1 Axis
Tracker, 3=Backtracted, 4=2 Axis Tracker)
ssc.data_set_number(dat, 'array_type', 0)
# Set ground coverage ratio
ssc.data_set_number(dat, 'gcr', 0.4)
# Set constant loss adjustment
ssc.data_set_number(dat, 'adjust:constant', 0)

# execute and put generation results back into dataframe
mod = ssc.module_create('pvwattsv5')
ssc.module_exec(mod, dat)
df['generation'] = np.array(ssc.data_get_array(dat, 'gen'))

# free the memory
ssc.data_free(dat)
ssc.module_free(mod)
```

Check the capacity Factor

In [9]:
```
# Divide sum of generation by the number of periods times th
e system size
df['generation'].sum() / (525600/int(interval) * system_capa
city)
```

Out[9]: 0.17350061605977407

In [10]:
```
# Total Energy:
df['generation'].sum()
```

Out[10]: 12158.923173468967

3. Plot the SAM simulation results

Define a plotting function using matplotlib:

In [11]:
```
%matplotlib inline
from matplotlib import pyplot as plt
plt.style.use('ggplot')

def nsrdb_plot(df, i):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax2 = ax.twinx()
    df['90 Degree Zenith'] = 90
    df[['GHI', 'DNI', 'DHI', 'Solar Zenith Angle', '90 Degree Zenith']][i:i+int(interval)].plot(ax=ax, figsize=(15,8), yticks=(np.arange(0,900,100)), style={'90 Degree Zenith': '--','Solar Zenith Angle': '-o', 'DNI': '-o', 'DHI': '-o', 'GHI': '-o'}, legend=False)
    df['generation'][i:i+30].plot(ax=ax2, yticks=(np.arange(0,4.5,0.5)), style={'generation': 'y-o'})
    ax.grid()
    ax.set_ylabel('W/m2')
    ax2.set_ylabel('kW')
    ax.legend(loc=2, ncol=5, frameon=False)
    ax2.legend(loc=1, frameon=False)
```

Take a Look at the Results

In [12]:
```
nsrdb_plot(df, 5050)
```