Use ORNL DAAC TDS Data Resources in Python

1. Python Modules used in this Tutorial

IPython [*http://ipython.org*]
IPython provides a rich architecture for interactive computing with Python.

Pydap [*http://www.pydap.org*]
Pydap is a pure Python library implementing the Data Access Protocol, also known as DODS or OPeNDAP. You can use Pydap as a client to access hundreds of scientific datasets in a transparent and efficient way through the Internet; or as a server to easily distribute your data from a variety of formats.

matplotlib [*http://matplotlib.org*]
matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Basemap [*http://matplotlib.org/basemap*]
The matplotlib basemap toolkit is a library for plotting 2D data on maps in Python. It is similar in functionality to the matlab mapping toolbox, the IDL mapping facilities, GrADS, or the Generic Mapping Tools.

NumPy [*http://www.numpy.org*]
NumPy is the fundamental package for scientific computing with Python.

2. Usage Examples

Find data of your interest in ORNL DAAC THREDDS data server
Go to http://thredds.daac.ornl.gov/thredds/catalogs/ornldaac/ornldaac.html to browse the ORNL DAAC THREDDS data server and find data of your interest. In the following examples, it's assumed the data of interest is *climate6190.nc* in the "GLOBAL 30-YEAR MEAN MONTHLY CLIMATOLOGY, 1961-1990 (NEW ET AL.)" data set. Get its OPeNDAP data access URL (http://thredds.daac.ornl.gov/thredds/dodsC/ornldaac/542/climate6190.nc4) from the OPeNDAP Dataset Access Form.

Retrieve, analyze, and plot Data from ORNL DAAC THREDDS data server
[*Script: http://thredds.daac.ornl.gov/tutorials/tds/thredds_tutorial_python_pydap.py*]
Note: See inline comments for detailed information

```
# Import modules
In[1]: from pydap.client import open_url
In[2]: from mpl_toolkits.basemap import Basemap
In[3]: import numpy as np
In[4]: import matplotlib.pyplot as plt
```

```
# Connect to select data source in ORNL DAAC TDS
In[5]: dataset =
open_url("http://thredds.daac.ornl.gov/thredds/dodsC/ornldaac/542/clima
te6190.nc4")

# Display data source metadata
# List all variables
In[6]: dataset.keys()
Out[6]:
['month',
 'climatology_bounds',
 'latitude',
 'longitude',
 'CLD',
 'DTR',
 'FRS',
 'PRE',
 'RAD',
 'WET',
 'TMP',
 'TMX',
 'TMN',
 'VAP',
 'WND']

# Select variable RAD and check its metadata
# Check all attributes of variable RAD
In[7]: dataset.RAD.attributes
Out[7]:
{'_ChunkSize': [6, 180, 360],
 '_FillValue': -9999,
 'cell_methods': 'time: mean within months time: mean over years',
 'long_name': 'Radiation',
 'scale_factor': 0.1,
 'units': 'w/m2',
 'valid_range': [0, 1000]}
# Check dimensions and size of variable RAD
In[8]: dataset.RAD.array.dimensions
Out[8]: ('month', 'latitude', 'longitude')
In[9]: dataset.RAD.array.shape
Out[9]: (12, 360, 720)

# Subset variable RAD
# Get January Radiation in North America (West: -170, South: 10, East:
-50, North: 84)
In[10]: rad_na_jan = dataset.RAD[0,200:348,380:620]
# Get coordinates of the subset for use in plotting
In[11]: lat = rad_na_jan.latitude[:]
In[12]: lon = rad_na_jan.longitude[:]
# Mask out missing pixels
In[13]: rad_na_jan =
np.ma.masked_where(rad_na_jan.RAD==dataset.RAD._FillValue,
rad_na_jan.RAD)
In[14]: rad_na_jan =rad_na_jan*dataset.RAD.scale_factor
# Get July Radiation in North America (West: -170, South: 10, East: -
50, North: 84)
In[15]: rad_na_jul = dataset.RAD[6,200:348,380:620]
```

```
# Mask out missing pixels

In[16]: rad_na_jul =
np.ma.masked_where(rad_na_jul.RAD==dataset.RAD._FillValue,
rad_na_jul.RAD)
In[17]: rad_na_jul =rad_na_jul*dataset.RAD.scale_factor

# Calculate radiation difference between July and January
In[18]: rad_na_diff=rad_na_jul-rad_na_jan

# Create plot
In[19]: fig = plt.figure(figsize=(12,8))
In[20]: ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

# Setting up the map
# We define a map in Lambert Conformal Conic (LCC) Projection
# Note: map projection doesn't have to be same as data projection
In[21]: m = Basemap(llcrnrlon=-145.5,\
                    llcrnrlat=1.,\
                    urcrnrlon=-2.566,\
                    urcrnrlat=46.352,\
                    rsphere=(6378137.00,6356752.3142),\
                    resolution='l',\
                    area_thresh=1000.,\
                    projection='lcc',\
                    lat_1=50.,
                    lon_0=-100.,
                    ax=ax)
# Reproject data coordinates to map projection (LCC)
In[22]: X, Y = m(*np.meshgrid(lon, lat))

# Eliminate time dimension
In[23]: Z = np.mean(rad_na_diff, axis=0)

# Map data values to colors
In[24]: colors = m.contourf(X, Y, Z)
In[25]: cb = plt.colorbar(colors)
In[26]: cb.set_label(dataset.RAD.units)

# Draw a high-resolution land-sea mask as an image, with land and ocean
colors specified. The land-sea mask is derived from the GSHHS coastline
data, and there are several coastline options and pixel sizes to choose
from.
In[27]: m.drawlsmask()

# Draw other auxiliary features
In[28]: m.drawcoastlines(linewidth=0.5)
In[29]: m.drawcountries()
In[30]: m.drawstates()

# Draw map grids
In[31]: parallels = np.arange(0.,80,20.)
In[32]: m.drawparallels(parallels,labels=[1,0,0,1])
In[33]: meridians = np.arange(10.,360.,30.)
In[34]: m.drawmeridians(meridians,labels=[1,0,0,1])

# Set a title for the plot
In[35]: ax.set_title(dataset.RAD.long_name)
```

```
# Save plot into a PNG image
In[36]: fig.savefig('RAD.png')
```

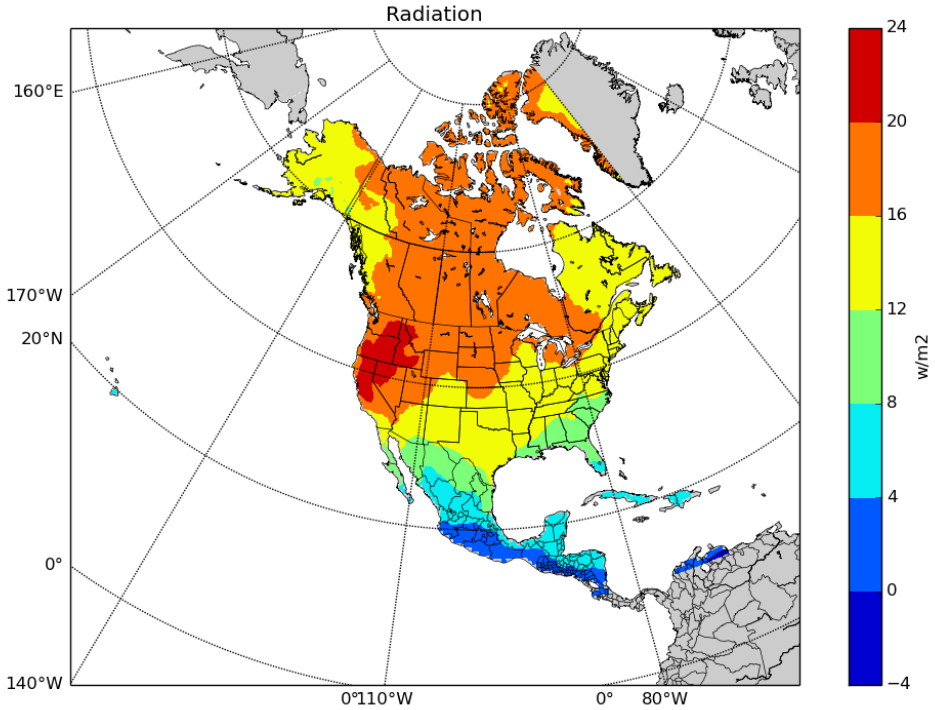

Figure 1. Difference between 30-yr mean monthly climatology in July and January (July – January)