Python を用いた人工衛星（MODIS）データの解析　手引書

担当：植山雅仁
実習場所：B11 棟・238 号室

1. 新規プロジェクトの立ち上げと準備

Jupyter Notebook を立ち上げ、[New]ボタンから Python [default]をクリックして、新規プロジェクトを立ち上げる。新規プロジェクトが立ち上がると、File タブから rename をクリックし、任意のプロジェクト名（例えば、MODIS.Python 実習）を入力する。

「Hello World!」という文字が表示されるプログラムを実行する。以下を入力し、[Shift]＋[Tab]＋[Enter]を同時に押すと、下に結果が表示される（Cell タブから、[Run Cells]をクリックしても同様にプログラムを実行できる）。

```
>> print('Hello World!')
```

このように、Python では入力した単位ごとに、対話的に結果が得ることができる。以下のように四則演算も可能である。

```
>>a = 1 + 1
>>print(a)
>>b = 10 * 10 / 2
>>print(b)
```

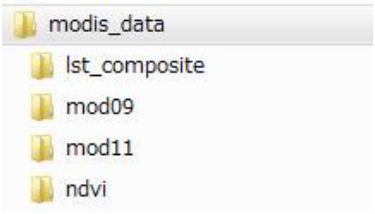


図 1. 衛星データが保存されるディレクトリ構造

作業ディレクトリの確認は以下のコマンドでできる。

```
>> pwd
```

はじめに、作業ディレクトリを衛星データが保存されている変更（change directory）する。ディレクトリの変更は、以下のコマンドでできる。

```
>>import os                          os ライブラリをインポートする
>> os.chdir( 'c:¥¥modis_data')       os.chdir は、指定した先に作業ディレクトリを移動する関数
```

Windows ではディレクトリの階層に「¥」の表記使われるが、Python では「¥¥」と表記することに注意する。本実習では、図 1 にならったディレクトリ構造でデータが保存されているものとする。ここで、mod09 ディレクトリには反射率のバイナリデータが、mod11 には地表面温度のバイナリデータが保存されており、ndvi は計算される NDVI データを保存するためのディレクトリ、lst_composite は地表面温度のコンポジットを保存するためのディレクトリである。「cd」でディレクトリの移動が終わったら、「pwd」で作業ディレクトリが移動できているかを確認する。

2. 衛星データの可視化

地表面温度のデータの可視化のために以下のプログラムを入力してみよう。よく理解してプログラム課題を進めること。本テキストでは、プログラム中に赤字でプログラムの意味を注釈する。赤字で記載されている箇所については、入力する必要はない。解説はプログラム初心者が理解できる用語を心がけているため、正しい情報処理用語になっていないところもある点に留意してほしい。

Example 1

Python では、必要とするライブラリを必要なときにインポートする。さまざまなモジュールを組み合わせて利用することで、高度な処理をするプログラムを迅速に構築することができる。

```
>>import matplotlib.pyplot as plt        作図に必要なライブラリ「matplotlib.pyplot」をインポートする
>>import matplotlib.cm as cm             作図に必要なライブラリ「matplotlib.cm」をインポートする
>>import numpy as np                     数値計算に利用するライブラリ「numpy」をインポートする
>>
>># show LST data                        #以降に書かれている文字は、コメントとして実行時に読み飛ばされる
                                         後からの可読性向上のため、積極的にコメントを書くこと
>>sample_mod11 = 99                      衛星データの横（経度方向）方向のピクセル数
>>line_mod11 = 99                        衛星データの縦（緯度方向）方向のピクセル数
>>
>>scale = 0.02                           デジタル値を物理値に変換するための係数
>>offset = -273.15 # K                   K を℃に変換するために 273.15℃差し引く
>>
>>strLST_file = 'mod11¥¥MOD11A2.A2008145.OSAKA.LST_Day_1km.dat'
                                         mod11 ディレクトリにある、ファイルを指定
>>
>># Read Two-dimensional Binary File
>>fin = open(strLST_file, 'rb')          指定したファイルを open コマンドで開く
                    rb オプションは、read binary の略でバイナリファイルを開くことを明示している。
                    開いたファイルは fin 変数の中に格納される。
>>dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
                    np.fromfile コマンドで、fin の中のデータを dbuf 変数に取り出す。
                    地表面温度は、16 ビットの符号無し整数なので、uint16 を指定する。
>>fin.close()       開いた fin ファイルを閉じる

>>dLST = dbuf.reshape(sample_mod11, line_mod11, order='F') * scale + offset
                    開いたバイナリファイルを 99 x 99 の二次元データにわりあて、dLST 変数に代入する。
                    scale, offset を使って、バイナリ値から℃に単位換算する。
                    LST は Land Surface Temperature の略（分かりやすい変数名をつけること）
>>                  可読性のため、sample_mod11 などの定数を直接、じか書きしない（書いても動作するが）
>>plt.imshow(np.transpose(dLST), clim=(0, 40), cmap=cm.jet)
                    imshow を使って画像を表示する。（matplotlib.pyplot ライブラリの imshow 関数）
                    clim(0,40)で、0～40℃のレンジで画像を表示することを明示する。
```

cm.jet でカラーテーブルを指定する。以下から、いろいろなカラーテーブルを試してみよう。

https://matplotlib.org/examples/color/colormaps_reference.html

np.transpose は転置行列を計算する関数。これを省略すると 90 度回転した画像が表示される。

```
>>clb = plt.colorbar(extend='min')      カラーバーを表示する　（コメントアウトするとどうなる？）
>>clb.ax.set_title('daytime LST')       カラーバーのタイトルを表示する（コメントアウトするとどうなる？）
>>plt.title('MODIS LST')                図のタイトルを表示する（コメントアウトするとどうなる？）
>>plt.axis('off')                       図の軸を消す（コメントアウトするとどうなる？）
>>plt.show()                            図を表示する。
```

上の行で、「%matplotlib inline」を記入しておけば、plt.show()は省略できる。

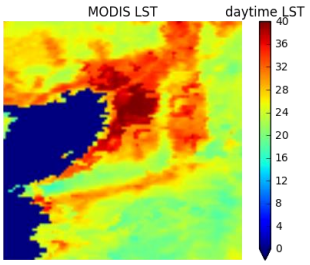


図 2. 表示された地表面温度 (2008 年 145 日)

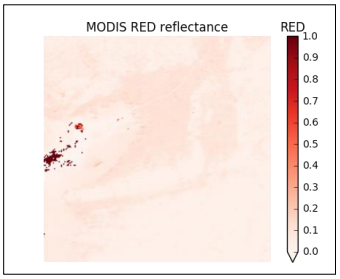
上記のプログラムを実行すると図 2 のような画像が表示される。これがバイナリファイルの読み込みと、可視化の一連の流れとなる。以降のプログラムでは、同じ文法については赤字で注釈しないので、このプログラムの意味をよく理解したうえで、次に進むこと。

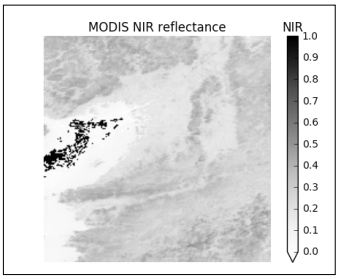
MOD09 の反射率のデータから、植生指数を計算して可視化する。実習資料の 3 節をあわせて読むこと。データの読み込みや可視化については Example 1 のプログラムと似ているため、プログラムソースをコピーして、適宜、変更するとよい。

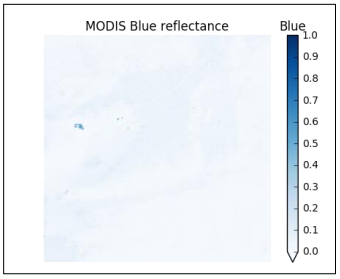
Example 2

```
>>%matplotlib inline
>>import matplotlib.pyplot as plt
>>import matplotlib.cm as cm
>>import numpy as np
>>
>># show MOD09 data
>>sample_mod09 = 198                    mod09 データは、198 x 198 のデータ
>>line_mod09 = 198                      mod09 データは、198 x 198 のデータ
>>
>>scale_mod09 = 0.000100                バイナリ値から反射率への変換係数
>>offset_mod09 = 0.0
>>
>>strRED_file = 'mod09¥¥MOD09A1.A2008145.OSAKA.sur_refl_b01.dat'      赤色バンドのファイル名
>>strNIR_file = 'mod09¥¥MOD09A1.A2008145.OSAKA.sur_refl_b02.dat'      近赤外バンドのファイル名
>>strBLE_file = 'mod09¥¥MOD09A1.A2008145.OSAKA.sur_refl_b03.dat'      青色バンドのファイル名
>>strGRN_file = 'mod09¥¥MOD09A1.A2008145.OSAKA.sur_refl_b04.dat'      緑色バンドのファイル名
>>
>># Read Two-dimensional Binary File          データの読み込みは example 1　とほぼ同じ
>>fin = open(strRED_file, 'rb')
>>dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>fin.close()
```

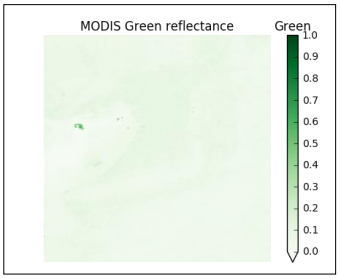
```
>>dRED = dbuf.reshape(sample_mod09, line_mod09, order='F') * scale_mod09 + offset_mod09
>>
>>fin = open(strNIR_file, 'rb')
>>dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>fin.close()
>>dNIR = dbuf.reshape(sample_mod09, line_mod09, order='F') * scale_mod09 + offset_mod09
>>
>>fin = open(strBLE_file, 'rb')
>>dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>fin.close()
>>dBLUE = dbuf.reshape(sample_mod09, line_mod09, order='F') * scale_mod09 + offset_mod09
>>
>>fin = open(strGRN_file, 'rb')
>>dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>fin.close()
>>dGREEN = dbuf.reshape(sample_mod09, line_mod09, order='F') * scale_mod09 + offset_mod09
>>
>># Show data
>>plt.imshow(np.transpose(dRED), clim=(0, 1), cmap=cm.Reds)      Reds のカラーテーブルを利用して赤色反射率の表示
>>clb = plt.colorbar(extend='min')
>>clb.ax.set_title('RED')
>>plt.title('MODIS RED reflectance')
>>plt.axis('off')
>>plt.show()
>>
>>plt.imshow(np.transpose(dNIR), clim=(0, 1), cmap=cm.Greys)      Greys のカラーテーブルを利用して近赤外反射率の表示
>>clb = plt.colorbar(extend='min')
>>clb.ax.set_title('NIR')
>>plt.title('MODIS NIR reflectance')
>>plt.axis('off')
>>plt.show()
>>
>>plt.imshow(np.transpose(dBLUE), clim=(0, 1), cmap=cm.Blues)    Blues のカラーテーブルを利用して青色反射率の表示
>>clb = plt.colorbar(extend='min')
>>clb.ax.set_title('Blue')
>>plt.title('MODIS Blue reflectance')
>>plt.axis('off')
>>plt.show()
>>
>>plt.imshow(np.transpose(dGREEN), clim=(0, 1), cmap=cm.Greens) Greens のカラーテーブルを利用して緑色反射率の表示
>>clb = plt.colorbar(extend='min')
```







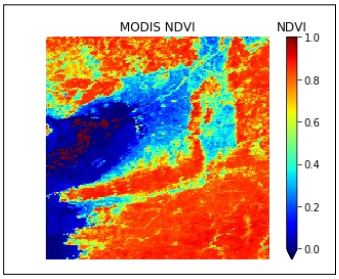
```
>>clb.ax.set_title('Green')
>>plt.title('MODIS Green reflectance')
>>plt.axis('off')
>>plt.show()
>>
```




```
>># Calculating NDVI
>>dNDVI = (dNIR - dRED) / (dNIR + dRED)
>>
>>plt.imshow(np.transpose(dNDVI), clim=(0, 1), cmap=cm.jet)
>>clb = plt.colorbar(extend='min')
>>clb.ax.set_title('NDVI')
>>plt.title('MODIS NDVI')
>>plt.axis('off')
>>plt.show()
>>
```

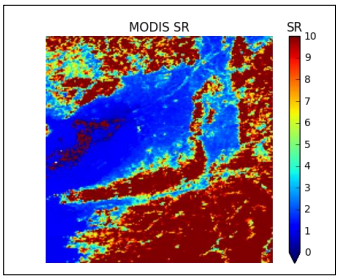
NDVI の計算

0～1 のレンジで、jet のカラーテーブルで NDVI を表示




```
>># Calculating SR
>>dSR = dNIR / dRED
>>
>>plt.imshow(np.transpose(dSR), clim=(0, 10), cmap=cm.jet)
>>clb = plt.colorbar(extend='min')
>>clb.ax.set_title('SR')
>>plt.title('MODIS SR')
>>plt.axis('off')
>>plt.show()
>>
```

Simple Ratio の計算と表示




```
>># Calculating EVI
>>C1 = 6.0
>>C2 = 7.5
>>L = 1.0
>>dEVI = (dNIR - dRED) / (dNIR + C1 * dRED - C2 * dBLUE + L)
>>
>>plt.imshow(np.transpose(dEVI), clim=(0, 0.3), cmap=cm.jet)
>>clb = plt.colorbar(extend='min')
>>clb.ax.set_title('EVI')
>>plt.title('MODIS EVI')
>>plt.axis('off')
>>plt.show()
>>
```

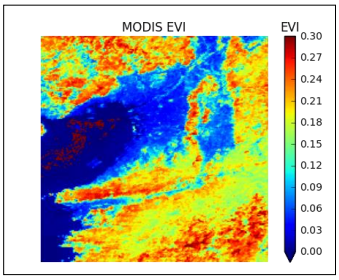
EVI の計算と表示

EVI 計算のための定数

EVI 計算のための定数

EVI 計算のための定数

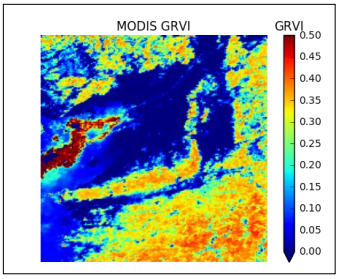
可読性のためプログラムでは C1 などの定数を直接式の中に書かない




```
>># Calculating GRVI
```

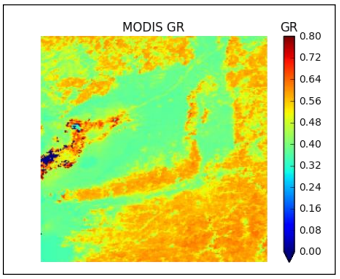
GRVI の計算と表示

```
>>dGRVI = (dGREEN - dRED) / (dGREEN + dRED)
>>
>>plt.imshow(np.transpose(dGRVI), clim=(0, 0.5), cmap=cm.jet)
>>clb = plt.colorbar(extend='min')
>>clb.ax.set_title('GRVI')
>>plt.title('MODIS GRVI')
>>plt.axis('off')
>>plt.show()
```




```
>>
>># Calculating GR
>>dGR = dGREEN / (dRED + dGREEN + dBLUE)
>>
>>plt.imshow(np.transpose(dGR), clim=(0, 0.8), cmap=cm.jet)
>>clb = plt.colorbar(extend='min')
>>clb.ax.set_title('GR')
>>plt.title('MODIS GR')
>>plt.axis('off')
>>plt.show()
```

Green Ratio の計算と表示



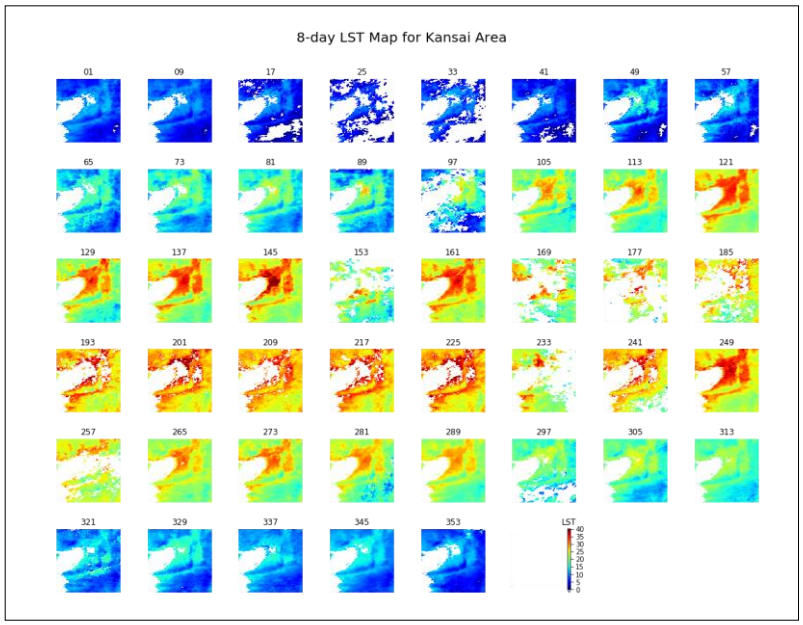
Example 2 は、Example 1 のコピー&ペーストでほぼ完成できることが理解できたと思います。このように、一度、プログラムを作ってしまうと、以降、流用できることがプログラムを書くことの利点の一つです。二つの例を通して 1 シーンについての衛星画像の可視化を習得できたと思います。次は、フォルダ内にある 1 年分の衛星画像を一度に可視化するプログラムを書いてみよう。Example 3 は、2008 年 1 年分の日中の地表面温度を可視化するプログラムです。Example 3 を改変して、夜間の表面温度についても同様に可視化してみよう。

Example 3

```
>>%matplotlib inline
>>import matplotlib.pyplot as plt
>>import matplotlib.cm as cm
>>import numpy as np
>>
>># show LST data
>>sample_mod11 = 99
>>line_mod11 = 99
>>
>>scale = 0.02
>>offset = -273.15 # K
>>
>>iYear = 2008
>>
>>fig = plt.figure(figsize=(15, 12))
```

解析年を表す変数（6 行下でファイル名を作る際に利用する）

表示する図のサイズを指定する（横 1500 pixel, 縦 1200 pixel ）



```
>>for i in range(0, 45):
>>      iday = i * 8 + 1
>>
>>      strInFile="mod11¥¥MOD11A2.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.LST_Day_1km.dat"
>>      fin = open(strInFile, 'rb')
>>      dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>      fin.close()
>>      dLST = dbuf.reshape(sample_mod11, line_mod11, order='F') * scale + offset
>>
>>      plt.subplot(6, 8, i+1)
>>      im = plt.imshow(np.transpose(dLST), clim=(0.0, 40.0), cmap=cm.jet)
>>      im.cmap.set_under('w')
>>      stitle="{0:02d}".format(iday)
>>      plt.title(stitle)
>>      plt.axis('off')
>>
>># for colorbar
>plt.subplot(6, 8, i+2)
>im = plt.imshow(np.transpose(dLST * 0.0 -999), clim=(0.0, 40.0), cmap=cm.jet)
>plt.axis('off')
>>
>clb = plt.colorbar(extend='min')
>clb.ax.set_title('LST')
>>
>fig.suptitle("8-day LST Map for Kansai Area", fontsize=20)
>plt.tight_layout()
>plt.subplots_adjust(top=0.9)
```

for 文は、指定した回数、スコープ内をループして繰り返し計算する
Python では、インデントされている領域を一つのスコープとみなす
繰り返しのたびに i が 0～44 の値をとり、計 45 回ループする。
（8-day なので、1 年で 45 枚の衛星画像がある）
for 構文の後ろに、「:」が必要なことに注意する。

解析対象の日を計算する（1, 9, … 353 と 8 日ずつ繰り上がる）

ファイル名を作って、strInFile に代入している。
Python では、文字列を「+」で連結することができる。
"{0:04d}".format(iYear)は、年を整数から文字列に変換し 4 桁で表示させている。
"{0:03d}".format(iday)は、日を整数から文字列に変換し 3 桁で表示させている。
03d とすることで、1 日目を「001」と 0 を付与して表示させている。
ここで「d」は変換対象が整数であることを明示している。

縦 6, 横 8 個からなるサブのグラフ中の i+1 番目に結果を表示させる

ここまでがインデントされた for 文のスコープになる。ここまでを 45 回繰り返す

カラーバーを表示させるために真っ白に表示されるダミーを書く

カラーバーを表示させるために真っ白に表示されるダミーを書く

カラーバーを表示させるために真っ白に表示されるダミーを書く

課題：Example 3 を参考に、8-day の NDVI を 1 年分計算して一度に可視化するプログラム（Example 4）を完成させよ（図 3 のような出力が出るプログラムを作る）。

Example 4

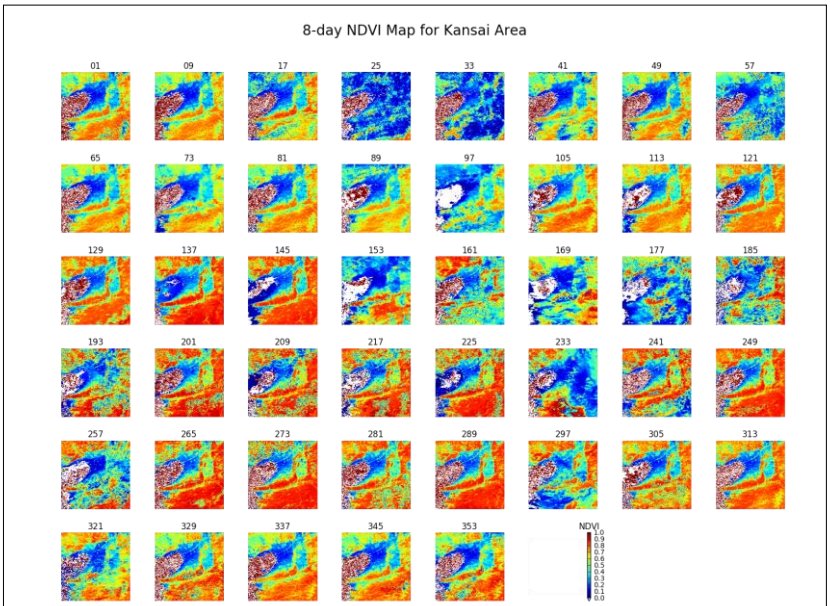


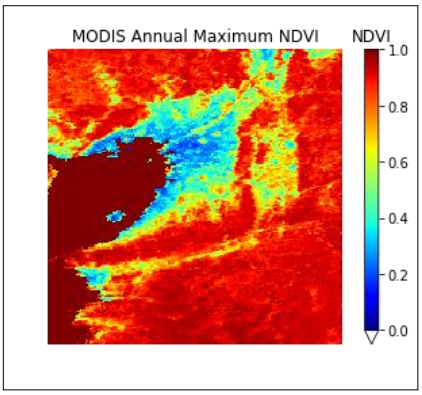
図 3. 2008 年 1 年間の 8 日コンポジットの日中の地表面温度の分布

```
>>%matplotlib inline
>>import matplotlib.pyplot as plt
>>import matplotlib.cm as cm
>>import numpy as np
>>
>>sample_mod09 =
>>line_mod09 =
>>
>>scale_mod09 =
>>offset_mod09 =
>>
>>iYear = 2008
>>
>>fig = plt.figure(figsize=(15, 12))
>>for i in range(0, 45):
>>      iday = i * 8 + 1
>>
>>      # 反射率のデータを読み込むコードを以下に書く
>>
>>      dNDVI = (dNIR - dRED) / (dNIR + dRED)          NDVI を計算している
>>
>>      # NDVI のデータを可視化するコードを以下に書く
>>
>>      # write LST data
>>      strOut_NDVI_File="ndvi¥¥MOD09A1.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.ndvi.dat"
                  ndvi ディレクトリに結果を保存するためのファイル名を作っている。
>>      fout = open(strOut_NDVI_File, 'wb')      'wb'  write binary  バイナリ書き込みモードでファイルを開く
>>      fout.write(np.float32(np.transpose(dNDVI)))          NDVI ファイルを書き出している
                  32 ビットの浮動小数点(float32)で書き出すことを明示している。
                  （この指定がなければ、64 ビットのデータとして保存される。）
>>      fout.close() 書き出しのために開いたファイルを閉じる
>>
>># カラーバーを表示するコードを以下に書く
>>
>># 図のタイトルを表示するコードを以下に書く
```

3. コンポジット画像の作成

NDVI を 1 年分、最大値合成法(Maximum Value Composite; MVC)でコンポジットし、領域の最大植物活性量を可視化する。アルゴリズムは、ファイルを 1 日目から 353 日目まで順に読み込み、最大値を計算するプログラムになる。実習資料の 4 節をあわせて読むこと。

Example 5




```
>># calculate annual composite NDVI
>>%matplotlib inline
>>import matplotlib.pyplot as plt
>>import matplotlib.cm as cm
>>import numpy as np
>>
>># show LST data
>>sample_mod09 = 198
>>line_mod09 = 198
>>
>>scale_ndvi = 1.0
>>offset_ndvi = 0.0
>>
>>iYear = 2008
>>
>>dNDVI_max = np.zeros([sample_mod09, line_mod09])  最大値を格納するための空のデータを準備する
        np.zeros は、numpy ライブラリーの zeros 関数で、配列要素が 0 で初期化されたデータを作る。
        198 x 198 の 0 で初期化されたデータが、dNDVI_max として生成された。
>>for i in range(0, 45):
>>      iday = i * 8 + 1
>>
>>      strIn_NDVI_File="ndvi¥¥MOD09A1.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.ndvi.dat"
>>        Example 4 で保存した、8 日毎の NDVI データを読み込むため、ファイル名を生成してる。
>>      fin = open(strIn_NDVI_File, 'rb')
>>      dbuf = np.fromfile(fin, dtype=np.float32, count=-1)
>>      fin.close()
>>      dNDVI = dbuf.reshape(sample_mod09, line_mod09, order='F') * scale_ndvi + offset_ndvi
>>
>>      idx = np.where(dNDVI_max < dNDVI)   where 関数は、指定された条件が当てはまるインデックスを返す
                この場合、ここまでの NDVI の最大値よりも読み込んだ NDVI のほうが高いピクセルの
                インデックスの一覧を、idx に格納している。
>>      dNDVI_max[idx] = dNDVI[idx]          この行までが for のスコープ
                where 関数で得られたピクセルに対して、最大 NDVI を読み込んだ NDVI で置き換える
>>
>>plt.imshow(np.transpose(dNDVI_max), clim=(0, 1), cmap=cm.jet)
>>clb = plt.colorbar(extend='min')
>>clb.ax.set_title('NDVI')
>>plt.title('MODIS Annual Maximum NDVI')
>>plt.axis('off')
>>plt.show()
```

4. 地点データの抽出

衛星画像の中から、指定した地点のデータを抽出するプログラムを記述する。下記の例は、大阪府立大学を対象に NDVI を抽出するプログラムであるが、一部を改変して日中・夜間の地表面温度や他の地点のデータについても抜き出せるようにプログラムを改変せよ。実習資料の 5 節をあわせて読むこと。

Example 6

```
>>import math as math                                    数学計算のライブラリの math をインポート
>>          numpy など他のライブラリは、これまでのプログラムでインポート済みなので省略する。
>>Lat_upper = 35.0                                      衛星画像の領域の緯度の北端
>>Lat_lower = 34.0                                      衛星画像の領域の緯度の南端
>>Lon_left    = 135.0                                   衛星画像の領域の緯度の西端
>>Lon_right = 136.0                                     衛星画像の領域の緯度の東端
>>
>>sample = 198
>>line = 198
>>
>># point geolocation
>>Lat_point = 34.0 + 32.0 / 60 + 48.23 / 60/ 60          抜き出す地点の緯度（大阪府立大学の地点）
>>Lon_point = 135.0 + 30.0 / 60 + 8.45 / 60/ 60          抜き出す地点の経度（大阪府立大学の地点）
>>
>>Lat = math.ceil( - (Lat_point - Lat_upper) / (Lat_upper - Lat_lower) * line)   衛星画像中のピクセルの計算（式 6）
>>Lon = math.ceil( (Lon_point - Lon_left) / (Lon_right - Lon_left) * sample)    衛星画像中のピクセルの計算（式 6）
          math.ceil は、切り上げた整数を返す関数
>>
>>fout_csv = open('OPU_NDVI.csv', 'w')          csv ファイルに書き出しのため、任意のファイル名を指定する。
                                    csv (comma separated value)    :    コンマで区切られたファイル
                                    Binary ファイルではないので、wb でなく w を指定する。
                                    ディレクトリを指定していないので、ファイルは作業ディレクトリにできる。
>> fout_csv.write('day,NDVI¥n')                    ファイルの 1 行目にヘッダ（day, NDVI）を書く
>>          write 関数は書き出しのための関数、「¥n」は行の最後に改行を加えるためのコード
>>iYear = 2008
>>for i in range(0, 45):
>>      iday = i * 8 + 1
>>
>>      strIn_NDVI_File="ndvi¥¥MOD09A1.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.ndvi.dat"
>>
>>      fin = open(strIn_NDVI_File, 'rb')
>>      dbuf = np.fromfile(fin, dtype=np.float32, count=-1)
>>      fin.close()
>>      dNDVI = dbuf.reshape(sample, line, order='F')
>>
```

```
>>      strLine = "{0:d}".format(iday) + ",{0:f}".format(dNDVI[Lat, Lon]) +"¥n"
```
書き出しのための行を文字列として整形している。
“日付, NDVI”の形式で書かれる。

```
>>      fout_csv.write(strLine)
```
整形した文字列を write 関数を使ってファイルに書き出す。

```
>>      print(iday, "¥t:", dNDVI[Lat, Lon])
```
ねんのため、画面にも print 関数で結果を表示する。

```
>>
```
「¥t」はタブ区切りで出力することを明示している。

```
>>fout_csv.close()
```

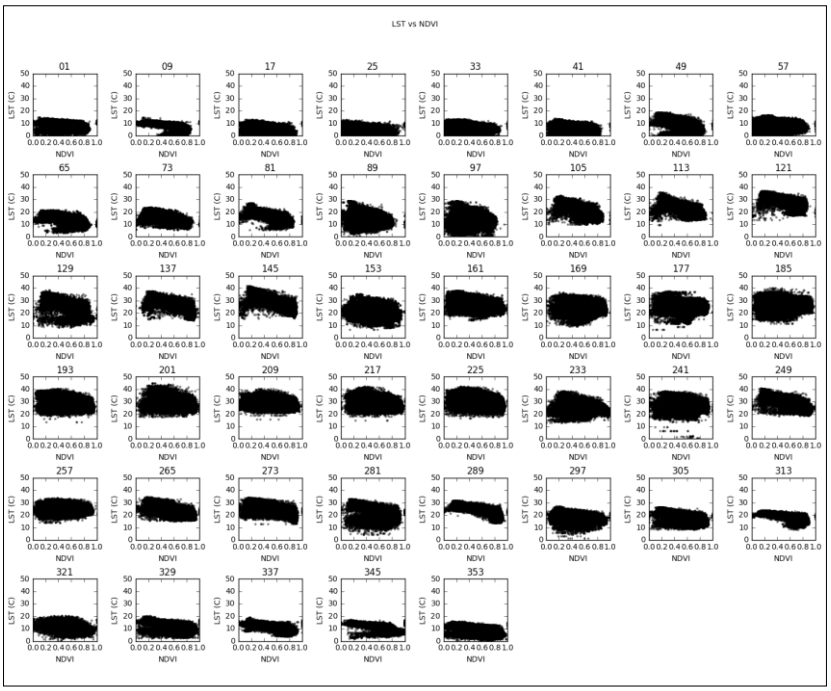
5. 地表面温度と植生指数の関係の可視化

衛星画像のシーンごとの地表面温度と植生指数の関係を散布図で可視化する。雲のない日を選んで、両者の関係を見て、NDVI と地表面温度の関係がどういった要因によって起こっているかを考察せよ。冬季と夏季では傾向が異なるか、日中と夜間では傾向が異なるか、異なる植生指数では関係に違いが出るかを評価し、その理由を考察せよ。下記は日中の地表面温度と NDVI を比較する例である。

Exmaple 7

```
>>%matplotlib inline
>>import matplotlib.pyplot as plt
>>import matplotlib.cm as cm
>>import numpy as np
>>import scipy.ndimage
```
画像調整のため scipy. ndimage ライブラリをインポート

```
>>
>>sample_mod09 = 198
>>line_mod09 = 198
>>
>>sample_mod11 = 99
>>line_mod11 = 99
>>
>>scale_mod11 = 0.02
>>offset_mod11 = -273.15 # K
>>
>>iYear = 2008
>>
>>fig = plt.figure(figsize=(15, 12))
>>for i in range(0, 45):
>>      iday = i * 8 + 1
>>
>>      strInFile="mod11¥¥MOD11A2.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.LST_Day_1km.dat"
>>      fin = open(strInFile, 'rb')
>>      dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>      fin.close()
>>      dLST = dbuf.reshape(sample_mod11, line_mod11, order='F') * scale_mod11 + offset_mod11
```



```
>>      dLST = scipy.ndimage.zoom(dLST, sample_mod09 / sample_mod11, order=0)
```
LST の空間解像度は 1 km、NDVI は 500 m であるため、同じ領域であっても画像サイズが異なる。そこで、zoom 関数を使って LST を 99x99=>198x198 へリサンプリングする。
倍率は「sample_mod09 / sample_mod11」、つまり 2 倍である。

```
>>
>>      strIn_NDVI_File="ndvi¥¥MOD09A1.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.ndvi.dat"
>>      fin = open(strIn_NDVI_File, 'rb')
>>      dbuf = np.fromfile(fin, dtype=np.float32, count=-1)
>>      fin.close()
>>      dNDVI = dbuf.reshape(sample_mod09, line_mod09, order='F')
>>
>>      plt.subplot(6, 8, i+1)
>>      plt.plot(dNDVI, dLST, 'o', c="black", markeredgewidth=0.0, alpha = 0.5, ms = 3)
```
plot 関数は散布図を描く関数（x 軸：dNDVI、y 軸：dLST）
詳細：https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html
'o':データを点で表示
c="black"　：黒色でプロットする
alpha = 0.5　：点の半透明度
ms = 23　：点の大きさ

```
>>      plt.xlim(0.0, 1.0)
```
x 軸のレンジを指定（NDVI のレンジは 0.0～1.0）

```
>>      plt.ylim(0, 50)
```
y 軸のレンジを指定（LST のレンジは 0.0～50.0℃）

```
>>      stitle="{0:02d}".format(iday)
>>      plt.title(stitle)
>>      plt.xlabel("NDVI")
>>      plt.ylabel("LST (C)")
>>
>>fig.suptitle("LST vs NDVI")
>>plt.tight_layout()
>>plt.subplots_adjust(top=0.9)
```

ここまでが、実験実習としての課題であるが、リモートセンシングを卒業研究のテーマにする学生や、さらなる学習に挑戦したい学生は、以下のサンプルプログラムを動かし、理解すると役立つと思う。

6. RGB 反射率データから True Color の可視化

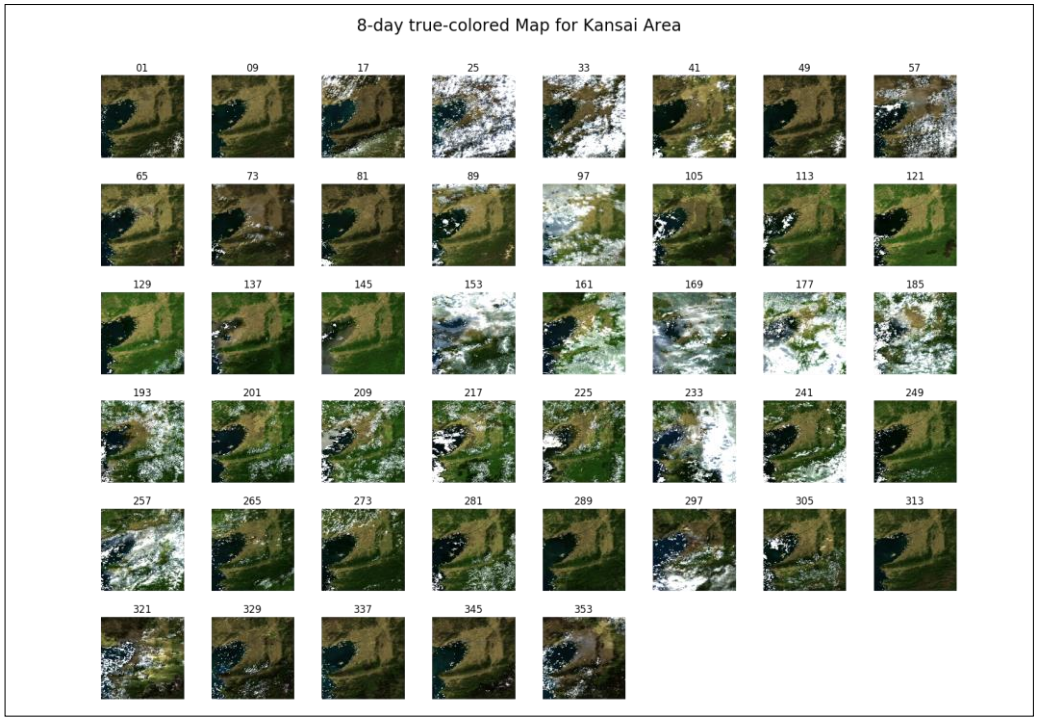
以下は、RGB（赤色・緑色・青色）波長の反射率から、写真のような可視画像を表示するプログラムです。

Example 8

```
>>%matplotlib inline
>>import matplotlib.pyplot as plt
>>import matplotlib.cm as cm
>>import numpy as np
```

```
>>
>>sample_mod09 = 198
>>line_mod09 = 198
>>
>>scale_mod09 = 0.000100
>>offset_mod09 = 0.0
>>
>>contrast_R = 5                    輝度をあげて描画するための補正係数
>>contrast_G = 5                    輝度をあげて描画するための補正係数
>>contrast_B = 5                    輝度をあげて描画するための補正係数
>>
>>iYear = 2008
>>
>>fig = plt.figure(figsize=(15, 12))
>>#figs, axes = plt.subplots(nrows=6, ncols=8, figsize=(15, 12))
>>for i in range(0, 45):
>>    iday = i * 8 + 1
>>
>>    strIn_RED_File="mod09¥¥MOD09A1.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.sur_refl_b01.dat"
>>    strIn_BLUE_File="mod09¥¥MOD09A1.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.sur_refl_b03.dat"
>>    strIn_GREEN_File="mod09¥¥MOD09A1.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.sur_refl_b04.dat"
>>
>>    fin = open(strIn_RED_File, 'rb')
>>    dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>    fin.close()
>>    dRED = dbuf.reshape(sample_mod09, line_mod09, order='F') * scale_mod09 * contrast_R + offset_mod09
>>    dRED = np.transpose(dRED)
>>
>>    fin = open(strIn_BLUE_File, 'rb')
>>    dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>    fin.close()
>>    dBLUE = dbuf.reshape(sample_mod09, line_mod09, order='F') * scale_mod09 * contrast_B + offset_mod09
>>    dBLUE = np.transpose(dBLUE)
>>
>>    fin = open(strIn_GREEN_File, 'rb')
>>    dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>    fin.close()
>>    dGREEN = dbuf.reshape(sample_mod09, line_mod09, order='F') * scale_mod09 * contrast_G + offset_mod09
>>    dGREEN = np.transpose(dGREEN)
>>
>>    idx = np.where(dRED > 0.9)          反射率が 0.9 を越えるピクセルは雲として強調表示する。
```

```
>>      dRED[idx] = 1.0                          反射率を強制的に 1.0 にして強調表示する。
>>      dGREEN[idx] = 1.0
>>      dBLUE[idx] = 1.0
>>
>>      idx = np.where(dGREEN > 0.9)             反射率が 0.9 を越えるピクセルは雲として強調表示する。
>>      dRED[idx] = 1.0
>>      dGREEN[idx] = 1.0
>>      dBLUE[idx] = 1.0
>>
>>      idx = np.where(dBLUE > 0.9)              反射率が 0.9 を越えるピクセルは雲として強調表示する。
>>      dRED[idx] = 1.0
>>      dGREEN[idx] = 1.0
>>      dBLUE[idx] = 1.0
>>
>>      truecolor = np.stack([dRED, dGREEN, dBLUE], axis=2)      True Color の合成
>>
>>      plt.subplot(6, 8, i+1)
>>      im = plt.imshow(truecolor)
>>      im.cmap.set_under('w')
>>      stitle="{0:02d}".format(iday)
>>      plt.title(stitle)
>>      plt.axis('off')
>>
>>fig.suptitle("8-day true-colored Map for Kansai Area", fontsize=20)
>>plt.tight_layout()
>>plt.subplots_adjust(top=0.9)
```



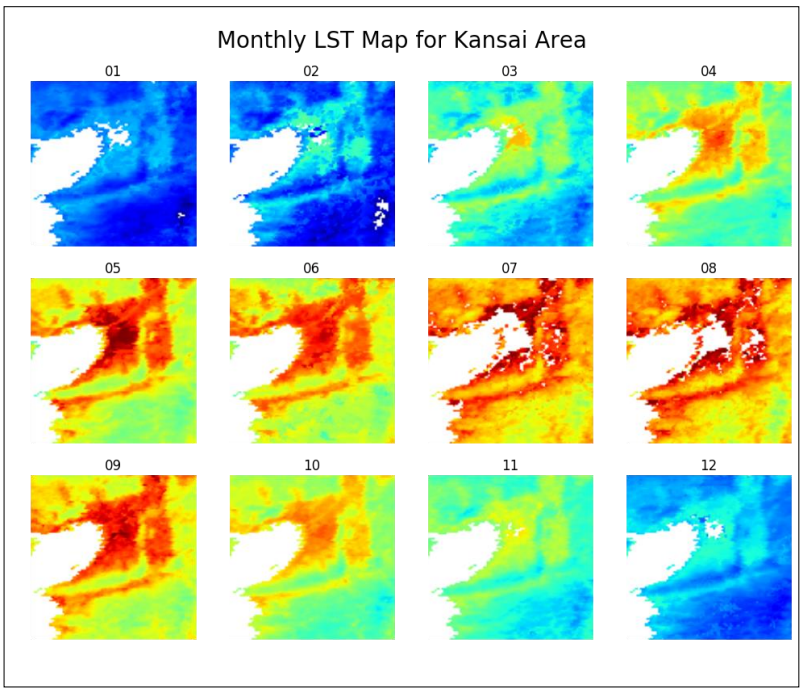
7. 月別コンポジット

以下は、月別コンポジットデータを作成し、保存、可視化するプログラムです。

Example 9

```
>>%matplotlib inline
>>import matplotlib.pyplot as plt
>>import matplotlib.cm as cm
>>import numpy as np
>>
>>def get_month(iDOY):
>>    iMonth = -1
>>    if iDOY >= 1 and iDOY <= 31:
>>        iMonth = 1
>>    if iDOY >= 32 and iDOY <= 59:
>>        iMonth = 2
>>    if iDOY >= 60 and iDOY <= 90:
>>        iMonth = 3
>>    if iDOY >= 91 and iDOY <= 120:
>>        iMonth = 4
>>    if iDOY >= 121 and iDOY <= 151:
>>        iMonth = 5
>>    if iDOY >= 152 and iDOY <= 181:
>>        iMonth = 6
>>    if iDOY >= 182 and iDOY <= 212:
>>        iMonth = 7
>>    if iDOY >= 213 and iDOY <= 243:
>>        iMonth = 8
>>    if iDOY >= 244 and iDOY <= 273:
>>        iMonth = 9
>>    if iDOY >= 274 and iDOY <= 304:
>>        iMonth = 10
>>    if iDOY >= 305 and iDOY <= 334:
>>        iMonth = 11
>>    if iDOY >= 335 and iDOY <= 366:
>>        iMonth = 12
>>    return iMonth
>>
>>sample_mod11 = 99
>>line_mod11 = 99
>>
```

Python では、「def」で関数（サブルーチン）を定義することができる。
サブルーチン get_month は通日(iDOY)を入力すると iMonth が返される
この関数は、通日を入力として、その日が該当する月を返す関数である。
if 文を使っての条件分岐
この場合は、iDOY が 1 以上、31 以下の場合は 1 月としている
return で結果（iMonth）をメインプログラムに返す。



```
>>scale = 0.02
>>offset = -273.15 # K
>>
>>iYear = 2008
>>
>>dLST_max = np.zeros([sample_mod11, line_mod11]) -9999
>>
>>fig = plt.figure(figsize=(10, 8))
>>
>>for i in range(0, 45):
>>    iday = i * 8 + 1
>>    iMonth = get_month(iday)
>>
>>    if i == 0:
>>        iMonthOld = iMonth
>>
>>    if iMonthOld != iMonth or i == 44:        一つ前のループ時の月と異なれば書き出しと可視化をする
                                                 ループの最後（i が 44 のとき）も書き出しと可視化をする
>>        plt.subplot(3, 4, iMonthOld)
>>        im = plt.imshow(np.transpose(dLST_max), clim=(0.0, 40.0), cmap=cm.jet)
>>        im.cmap.set_under('w')
>>        stitle="{0:02d}".format(iMonthOld)
>>        plt.title(stitle)
>>        plt.axis('off')
>>
>>        # write Composited data
>>        strOut_LST_File="lst_composite¥¥MOD11A2.A" + "{0:04d}".format(iYear) + ".{0:02d}".format(iMonthOld) + ".OSAKA.LST_Day_1km.dat"
>>        fout = open(strOut_LST_File, 'wb')
>>        dLST_max = np.transpose(dLST_max).copy()
>>        fout.write(np.float32(dLST_max))
>>        fout.close()
>>
>>        dLST_max = np.zeros([sample_mod11, line_mod11]) -9999
>>
>>    strInFile="mod11¥¥MOD11A2.A" + "{0:04d}".format(iYear) + "{0:03d}".format(iday) + ".OSAKA.LST_Day_1km.dat"
>>    fin = open(strInFile, 'rb')
>>    dbuf = np.fromfile(fin, dtype=np.uint16, count=-1)
>>    fin.close()
>>    dLST = dbuf.reshape(sample_mod11, line_mod11, order='F') * scale + offset
>>
>>    idx = np.where(dLST_max < dLST)
```

```
>>      dLST_max[idx] = dLST[idx]
>>
>>      iMonthOld = iMonth
>>
>>plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0, hspace=0)
>>
>>fig.suptitle("Monthly LST Map for Kansai Area", fontsize=20)
>>plt.tight_layout()
>>plt.subplots_adjust(top=0.9)
```

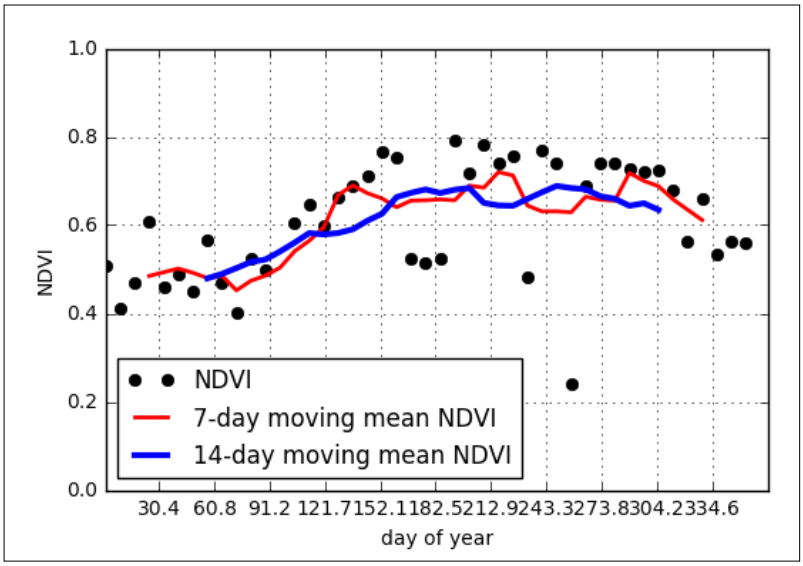
8. CSV ファイルの読み込みと地点データの可視化

Example 6 で CSV ファイルとして出力した地点データを Python で読み込み可視化する。CSV ファイルの読み方はいくつかあるが、Pandas ライブラリを使用するのが簡単であるため、以下にその例を記載する。

Example 10

```
>>%matplotlib inline
>>import matplotlib.pyplot as plt
>>import pandas as pd                          Pandas ライブラリの読み込み
>>
>>strFile = 'OPU_NDVI.csv'
>>
>># read file
>>df = pd.read_csv(strFile, skiprows=0)        read_csv 関数を使ってテキストデータの読み込み
>>                                            読み込んだ結果は、df 変数に格納
>>plt.plot(df.day, df.NDVI,    'o', c="black", label="NDVI")
        X 軸に df 内の day 変数、Y 軸に df 内の NDVI 変数とする散布図を描く
>>plt.ylabel('NDVI')
>>plt.xlabel('day of year')
>>
>>ndvi_7 = pd.rolling_mean(df.NDVI, center = True, window =7)          7 日移動平均を計算
>>ndvi_14 = pd.rolling_mean(df.NDVI, center = True, window =14)        14 日移動平均を計算
>>
>>plt.plot(df.day, ndvi_7, c="red", linewidth = 2, label="7-day moving mean NDVI")    7 日移動平均を描く
>>plt.plot(df.day, ndvi_14, c="blue", linewidth = 3, label="14-day moving mean NDVI") 14 日移動平均を描く
>>              linewidth は線の太さ、label は注釈に使用するデータの名前を明示している
>>plt.grid(b=True, which='major', color="black")       図に破線グリッドを加える
>>
>>ax = plt.gca()
>>ax.set_xticks(np.arange(0, 365, 365/12))             365 日を 12 等分、つまり月ごとにグリッドを引く
>>
>>plt.legend(loc=0)                                    注釈を加える
```

```
>>plt.xlim(1, 365)
>>plt.ylim(0.0, 1.0)
```



上記のプログラムで地点データをプロットすると、異常値と思われる NDVI の低下が見られます。簡易的に、NDVI が大きく落ち込んだデータを異常値とみなして、異常値を除去した後、内挿補完するためのプログラムを以下に記載します。このプログラムは、下記のホームページを参考に作りました。

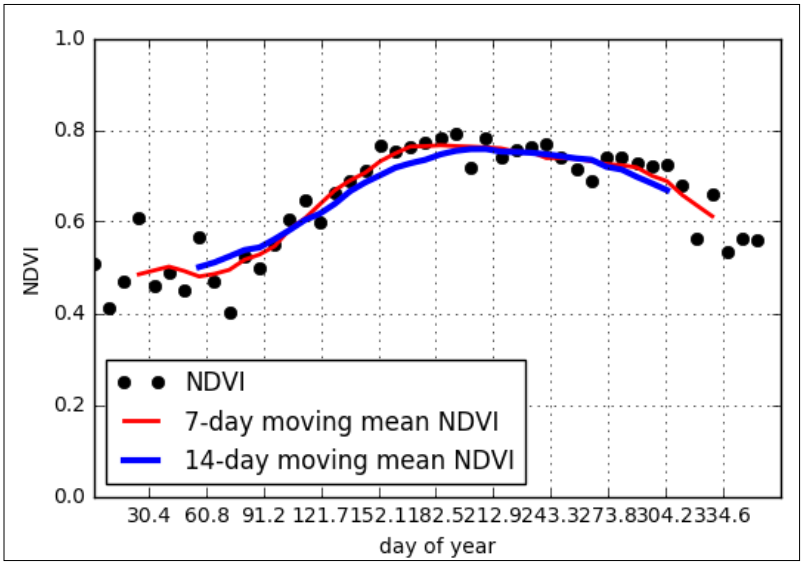
https://stackoverflow.com/questions/6518811/interpolate-nan-values-in-a-numpy-array

Example 11

```
>>from scipy import interpolate
>>
>>def nan_helper(y):                                    異常値を識別するためのサブルーチン
>>      """Helper to handle indices and logical indices of NaNs.
>>
>>      Input:
>>            - y, 1d numpy array with possible NaNs
>>      Output:
>>            - nans, logical indices of NaNs
>>            - index, a function, with signature indices= index(logical_indices),
>>               to convert logical indices of NaNs to 'equivalent' indices
>>      Example:
>>            >>> # linear interpolation of NaNs
>>            >>> nans, x= nan_helper(y)
>>            >>> y[nans]= np.interp(x(nans), x(~nans), y[~nans])
>>      """
>>
>>      return np.isnan(y), lambda z: z.nonzero()[0]
>>
>>dErrorThreshold = 0.1                                 異常値を識別すための任意の閾値
>>
```

```
>>ndvi = np.array(df.NDVI)
>>diff = ndvi_7 – ndvi                          7 日移動平均 NDVI から当該日の NDVI の差を計算
>>
>>idx = np.where(diff > dErrorThreshold)        差が閾値以上だと、異常値だとして
>>ndvi[idx] = None                              値を None に変更する
>>
>>nans, x= nan_helper(ndvi)                     関数 non_helper を実行し、エラー値を検出する
>>ndvi[nans]= np.interp(x(nans), x(~nans), ndvi[~nans])   エラー値を内挿補完する
```

異常値が除去されたデータが ndvi 変数の中に格納されます。これを再度、Example 10 を改変して可視化すると以下のような結果が得られます。



参考文献

市井和仁, 2017. Python を利用したデータ解析入門. 千葉大学環境リモートセンシング研究センター. 講義資料