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

1. 新規プロジェクトの立ち上げと準備

Jupyter Notebook を立ち上げる。

「Hello World!」という文字が表示されるプログラムを実行する。

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

[Cell] => [Run Cells]から、セル毎にプログラムを実行することができる（[CTL]+[Tab]+[Enter]でも同様に、セル毎にプログラムを実行することができる）。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)
```

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。

```
>>#####################################      #以降に書かれている文字は、
>>##  作業ディレクトリの変更                  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
>>import os                                  os ライブラリをインポートする
>>os.chdir('C:¥¥ SatelliteData')             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。「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。

プロジェクトの保存は、左上のディスクマークのアイコンでできる。また、保存するプロジェクト名の変更は、[File] => [rename]からできる。「2 回生.生物学実.Python 演習_氏名」など適切は名前に変更すること。

2. 衛星データの可視化

衛星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。

```
>>####################################
>># 課題 1. 土地被覆分類の作図
>>####################################
>>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                     数値計算に必要なライブラリ「matplotlib.cm」をインポートする
>>
>># size of data
>>sample = 675                           衛星データの横（経度方向）方向のピクセル数
>>line   = 600                           衛星データの縦（経度方向）方向のピクセル数
>>
>>infile = 'MOD12¥¥MOD12.LandCover.byt'  MOD12 ディレクトリにある、ファイルを指定
>>
>># Read Two-dimensional Binary File
>>fin = open(infile, '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.uint8, 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。
                  土地被覆分類データは、8 ビットの符号無し整数なので、uint8 を指定する。
>>fin.close()      開いた fin ファイルを閉じる
>>LAND = dbuf.reshape(sample, line, order='F')
                  開いたバイナリファイルを 675x 600 の二次元データにわりあて、LAND 変数に代入する。
                  LAND は土地を表す略語として定義した（分かりやすい変数名をつけること）
>>                可読性のため、sample などの定数を直接、じか書きしない（書いても動作するが）
>>
>>fig = plt.figure(figsize=(5, 5), dpi=100)
                  作図する図のサイズを定義する。縦 500 pixel×横 5 pixel の図を 100 dpi で作図。
>>plt.imshow(np.transpose(LAND), clim=(0, 16), cmap=cm.tab20)
                  imshow を使って画像を表示する。（matplotlib.pyplot ライブラリの imshow 関数）
                  clim(0,16)で、0～16 のレンジ(土地被覆は 16 種類)で画像を表示することを明示する。
                  cmap でカラーテーブルを指定する。今回は、tab20 カラーテーブルを使用する。
                  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',fraction=0.04)    カラーバーを表示する　（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。
```

上記のプログラムを実行すると図 1 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。

次に、NDVI データを可視化する。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 1 からの変更箇所は黄色にマークするので、意味を理解した上で、改変すること。

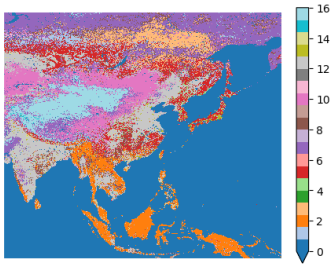

図 1. 表示された土地被覆分類

Example 2

```
>>###################################
>># 課題 2. NDVI の作図
>>###################################
>>infile = 'NDVI¥¥MOD13.NDVI.200401.flt'
>>
>># Read Two-dimensional Binary File
>>fin = open(infile, 'rb')
>>dbuf = np.fromfile(fin, dtype=np.float32, count=-1)
        NDVI データは、32 ビットの浮動小数点なので、float32 を指定する。
>>fin.close()
>>NDVI = dbuf.reshape(sample, line, order='F')
>>
>>fig = plt.figure(figsize=(5, 5), dpi=100)
>>plt.imshow(np.transpose(NDVI), clim=(0, 1), cmap=cm.jet)
        clim(0,1)で、0.0～1.0 のレンジで NDVI を表示することを明示する。
        cmap でカラーテーブルを指定する。今回は、jet カラーテーブルを使用する。
>>clb = plt.colorbar(extend='min',fraction=0.04)
>>plt.axis('off')
>>plt.show()
```

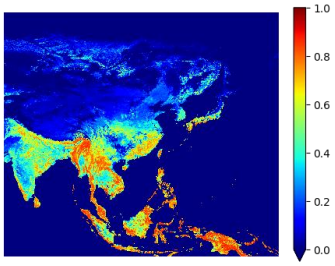

図 2. 表示された NDVI

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、同様に気温と降水量の分布の作図をしてみましょう。

Example 3

```
>>###################################
>># 課題 3. 気温の作図
>>###################################
>>infile = 'NCEP¥¥NCEP.TEMP.200401.flt'
>>
>># Read Two-dimensional Binary File
>>fin = open(infile, 'rb')
>>dbuf = np.fromfile(fin, dtype=np.float32, count=-1)
>>fin.close()
>>TAVE = dbuf.reshape(sample, line, order='F')
>>
>>fig = plt.figure(figsize=(5, 5), dpi=100)
>>plt.imshow(np.transpose(TAVE), clim=(-10, 30), cmap=cm.jet)
        clim(-10,30)で、-10℃～30℃のレンジで月平均気温を表示する
>>clb = plt.colorbar(extend='min',fraction=0.04)
>>plt.axis('off')
>>plt.show()
```

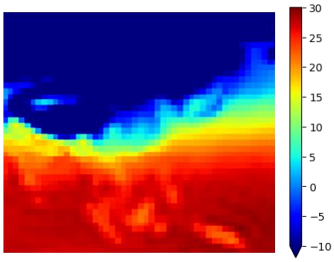

図 3. 表示された NDVI

図 3 のように作図された気温は、大陸と海洋の区別が付きにくいので、海洋のデータを白く塗りつぶして大陸だけの気温の分布を書いてみます。

Example 4

```
>>###################################
>># 課題 4. 気温の作図　(海を白にする)
>>###################################
>>infile = 'NCEP¥¥NCEP.TEMP.200401.flt'
>>
>># Read Two-dimensional Binary File
>>fin = open(infile, 'rb')
>>dbuf = np.fromfile(fin, dtype=np.float32, count=-1)
>>fin.close()
>>TAVE = dbuf.reshape(sample, line, order='F')
>>
>>idx = np.where(LAND==0) 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 LAND データが 0 の場合（海洋である場合）のピクセルの
            インデックスの一覧を、idx に格納する。
>>TAVE[idx] = np.nan
        where 関数で得られたピクセルに対して、異常値（np.nan）を代入する。
>>
>>fig = plt.figure(figsize=(5, 5), dpi=100)
>>ax = plt.imshow(np.transpose(TAVE), clim=(-20, 30), cmap=cm.jet)
```

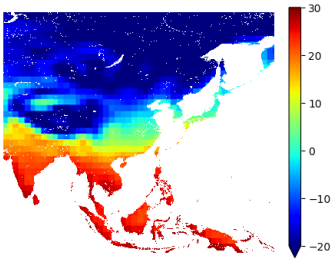

図 4. 表示された気温

```
>>clb = plt.colorbar(extend='min',fraction=0.04)
>>plt.axis('off')
>>
>>ax.cmap.set_bad('w')
>>plt.show()
```

異常値を白（'w'）に塗りつぶす。

同様に、降水量データを作図する。Example 4 を少し改変するだけで、作図のためのプログラムできる。

Example 5

```
>>####################################
>># 課題 5. 降水量の作図　(海を白にする)
>>####################################
>>infile = 'GPCP¥¥GPCP.PREC.200401.flt'
>>
>># Read Two-dimensional Binary File
>>fin = open(infile, 'rb')
>>dbuf = np.fromfile(fin, dtype=np.float32, count=-1)
>>fin.close()
>>PREC = dbuf.reshape(sample, line, order='F')
>>
>>idx = np.where(LAND==0)
>>PREC[idx] = np.nan
>>
>>fig = plt.figure(figsize=(5, 5), dpi=100)
>>ax = plt.imshow(np.transpose(PREC), clim=(0, 200), cmap=cm.jet)
>>clb = plt.colorbar(extend='min',fraction=0.04)
>>plt.axis('off')
>>
>>ax.cmap.set_bad('w')
>>plt.show()
```

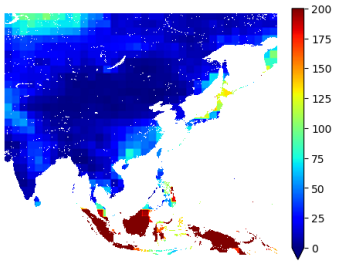

図 5. 表示された降水量

次に、領域内の同じ土地被覆分類の地点の NDVI、気温、降水量の平均値を計算する。次の例は、土地被覆が 5 番の地点、つまり混合林が代表している地点の平均値を算出するプログラムだ。

Example 6

```
>>###################################
>># 課題 6. 領域平均値の計算
>>###################################
>>idx_ndvi = np.where((LAND == 5) & (NDVI > 0))
        LAND が 5 番（混合林）で NDVI が正の場合のインデックスを idx_ndvi に抽出する。
>>mean_NDVI = np.mean(NDVI[idx_ndvi])
        抽出した地点の NDVI を平均して、結果を meam_NDVI 変数に代入する。
        np.mean は、平均値を計算する関数
>>
>>idx = np.where(LAND == 5)          LAND が 5 番（混合林）のインデックスを idx に抽出する。
>>mean_T = np.mean(TAVE[idx])       idx で抽出された気温の平均値を mean_T に代入する。
>>mean_P = np.mean(PREC[idx])       idx で抽出された降水量の平均値を mean_P に代入する。
>>
>>print(mean_NDVI, mean_T, mean_P)          print は結果を表示する関数
```

プログラムを実行すると、下記のような結果が出力される。

```
0.262519 -11.3298 26.4044
```

Example 2～6 までで、混合林（LAND==5）が代表する地点の 1 月の NDVI、気温、降水量の領域平均値が計算された。課題では、いくつかの土地被覆に対して 1～12 月の結果を出力して結果を考察する。Example 2～6 を各月、各土地被覆に実行することも可能だが、下の Example 7 を改変すれば、一度に 1～12 月までの結果を出力することができる。Example 7 は NDVI と気温を出力するプログラムであるが、これに降水量を出力できるように改変して課題に必要なデータを計算せよ。

Example 7

```
>>###################################
>># 課題 7. 一括実行
>>###################################
>>iTarget_Land = 5          対象とする土地被覆を定義（5 番：混合林の場合）
>>
>>fig = plt.figure(figsize=(15, 5))
>>for iMonth in range(1, 13):      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
                                   iMonth = 1, 2, 3, …, 11, 12  まで繰り返す。
>>      infile = "NDVI¥¥MOD13.NDVI.2004{0:02d}".format(iMonth) + ".flt"
                ファイル名を作って、infile に代入している。
                "{0:02d}".format(iMonth)は、月を整数から文字列に変換し 2 桁で表示させている。
>>      fin = open(infile, 'rb')
>>      dbuf = np.fromfile(fin, dtype=np.float32, count=-1)
>>      fin.close()
```

```
>>      NDVI = dbuf.reshape(sample, line, order='F')
>>
>>      plt.subplot(2, 6, iMonth)          縦 2，横 6 個からなるサブのグラフ中の iMonth 番目に結果を表示させる

>>      im = plt.imshow(np.transpose(NDVI), clim=(0.0, 1.0), cmap=cm.jet)
>>      im.cmap.set_bad('w')
>>      stitle="{0:02d}".format(iMonth)          図のサブタイトル（stitle）に月を書く
>>      plt.title(stitle)
>>      plt.axis('off')
>>
>>      idx_ndvi = np.where((LAND == iTarget_Land) & (NDVI > 0))
>>      mean_NDVI = np.mean(NDVI[idx_ndvi])
>>
>>      infile = "NCEP¥¥NCEP.TEMP.2004{0:02d}".format(iMonth) + ".flt"
                    ファイル名を作って、infile に代入している。
>>      fin = open(infile, 'rb')
>>      dbuf = np.fromfile(fin, dtype=np.float32, count=-1)
>>      fin.close()
>>      TAVE = dbuf.reshape(sample, line, order='F')
>>
>>      この部分に降水量に関するファイル読み込みと平均値の計算結果を変数(mean_PREC)に代入するコードを記述する。
>>
>>      idx = np.where(LAND == iTarget_Land)
>>      mean_TAVE = np.mean(TAVE[idx])
>>      mean_PREC = np.mean(PREC[idx])
>>      print(mean_NDVI, mean_TAVE, mean_PREC)
>>plt.show()
```

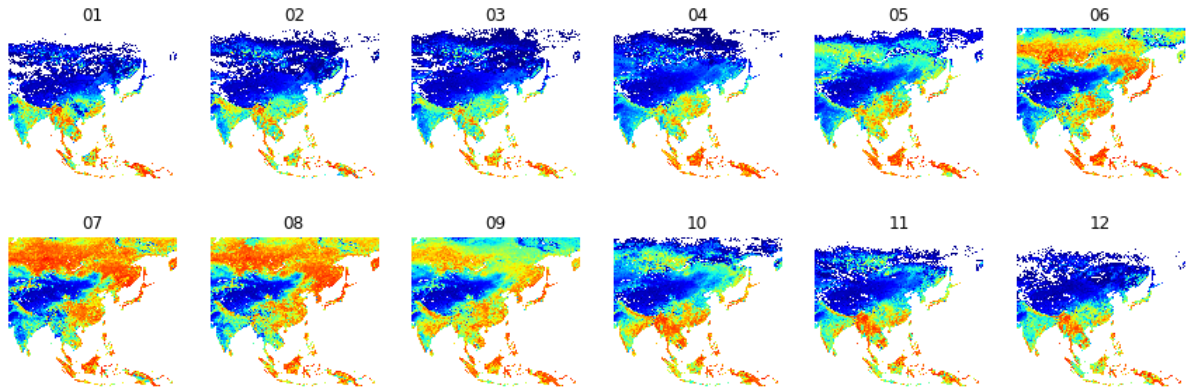

図 6. 2004 年 1～12 月の NDVI の季節変化

これまでの結果を Excel に入力して図を描くこともできるが、Python を用いて下記のようなプログラムを使って図示することもできる。Python での図示に挑戦したい学生は、以下を参考にして Python を使って綺麗な図を描画してみよう。

Example 8

```
>>###################################
>># 月別値の配列
>>###################################
>>arryMonth = [1,2,3,4,5,6,7,8,9,10,11,12]
>>arryNDVI = [0.262519, 0.283223, 0.272145, 0.315765, 0.534816, 0.735341, 0.775702, 0.770384, 0.696187, 0.543937, 0.413245, 0.27273]
>>arryTAVE = [-11.3298, -7.51536, -4.4829, 2.52374, 10.3688, 17.2388, 18.9142, 17.558, 12.9035, 5.27765, -2.24817, -11.7525]
>>arryPREC = [26.4044, 37.1635, 44.5022, 54.4573, 95.6562, 97.1807, 123.91, 114.237, 95.7438, 58.2198, 47.0205, 40.143]
>>
>>plt.figure(figsize=(4, 5), dpi=100)
>>###################################
>># 上段　NDVI　線グラフ
>>###################################
>>plt.subplot(2, 1, 1)                          上段にグラフを定義
>>plt.plot(arryMonth, arryNDVI, "o-", c='black')   x 軸に月、y 軸に NDVI の散布図を plot 関数で描く
>>
>>plt.ylim(0, 1)                                y 軸の範囲を 0～1 に指定
>>plt.xticks( [] )                              x 軸の軸ラベルは何も描画しない
>>plt.yticks( [0.0, 0.2, 0.4, 0.6, 0.8, 1.0] )  y 軸は 0.2 ごとの刻みに描画する
>>
>>plt.ylabel('NDVI')                            y 軸のラベルに「NDVI」と書く
>>plt.grid(b=True, which='major', color="gray", linestyle = '--', zorder=0)      背景にグリッドを書く
>>
>>###################################
>># 下段　第 1 軸 気温　線グラフ
>>###################################
>>ax2 = plt.subplot(2, 1, 2)                    下段にグラフを定義
>>lns1, = plt.plot(arryMonth, arryTAVE, "o-", c='black', zorder=2, label='Temperature')
                                  x 軸に月、y 軸に気温の散布図を plot 関数で描く
>>plt.ylim(-20, 30)               y 軸の範囲を-20～30℃に指定
>>plt.yticks( [-20, -10, 0, 10, 20, 30] )  y 軸は 10℃ごとの刻みに描画する
>>plt.ylabel('Temperature (C)')          y 軸のラベルに「Temperature (C)」と書く
>>
>>###################################
>># 下段　第 2 軸 降水量棒グラフ
>>###################################
>>plt2 =plt.twinx()                 第二軸を定義
>>lns2 = plt2.bar(arryMonth, arryPREC, color='gray', alpha=0.8, zorder=1, label='Precipitation')
                                  x 軸に月、y 軸に降水量の棒グラフを bar 関数で描く
>>plt.ylim(0, 250)                 y 軸の範囲を 0～250 mm に指定
>>
>>plt.yticks( [0, 50, 100, 150, 200, 250] )      y 軸は 50 mm ごとの刻みに描画する
```

```
>>plt.ylabel('Precipitation (mm)')
>>
>>plt.xticks( [1,2,3,4,5,6,7,8,9,10,11,12] )
>>plt.xlabel('month')
>>plt.grid(b=True, which='major', color="gray", linestyle = '--', zorder=0)
```

y 軸のラベルに「Precipitation (mm)」と書く

x 軸に月を書く

x 軸にラベル「month」を書く

背景にグリッドを書く

```
>>###################################
>># 図の注釈と下段図の統計値の記述
>>###################################
>>axlegend = plt.legend([lns1, lns2], ["Temperature", "Precipitation"], loc=0, prop={'size': 7})
>>axlegend.get_frame().set_edgecolor('#000000')
```

legend は図に注釈を記述する関数

loc は注釈を記述する場所を指定する（0 は右上）

```
>>
>>font_size = 6
```

図下段に記述する統計値のフォントサイズを指定（ここでは 6 ポイントを指定）

```
>>strText = "Mean temp. : {:.1f} C".format(np.mean(arryTAVE))
```

np.mean は平均を計算

「{:.1f}」で小数点以下一桁の実数表示させている。

```
>>plt.text(0.3, 230, strText, size=font_size)
```

plt.text は任意の文字列を描画する関数

横 0.3，縦 230 の位置に描画している。

```
>>strText = "Maximum temp. : {:.1f} C".format(np.max(arryTAVE))
```

np.max は最高値を計算

```
>>plt.text(0.3, 215, strText, size=font_size)
>>strText = "Minimum temp. : {:.1f} C".format(np.min(arryTAVE))
```

np.min は最低値を計算

```
>>plt.text(0.3, 200, strText, size=font_size)
>>strText = "Annual prec. : {:.1f} mm".format(np.sum(arryPREC))
```

np.sum は合計を計算

```
>>plt.text(0.3, 185, strText, size=font_size)

>>plt.show()
```

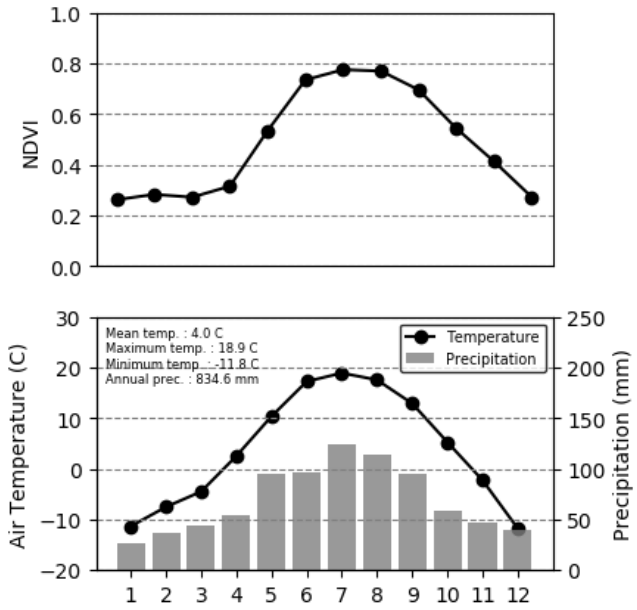

図 7. Python でプロットした NDVI、気温、降水量の季節変化
（2004 年の混合林の領域平均値）

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