GPM/TRMM
データ読み込みプログラムガイド
(Python 編)

2021/12/24

第六版

本書は全球降雨観測衛星(GPM/TRMM)のデータを読み込むプログラム（Python）の作成方法についてまとめたものです。
本書で解説するサンプルプログラムは GPM/TRMM はプロダクトバージョン０7、GSMaP はプロダクトバージョン５で動作を確認しています。

目次

1. はじめに

本書は GPM/TRMM データに対して Python を用いて読み込む方法について解説します。
GPM 及び TRMM はバージョン 06 プロダクト（TRMM バージョン 8 相当）からフォーマットを統一しており、最新のアルゴリズムはバージョン 07（TRMM バージョン 9 相当）となっています。本サンプルプログラムにて同様に読むことができます。
GPM データを読み込むには Python の他にも表 1.1 に示すような方法があります。どの方法で読み込むかについては、次頁の「読み込み方法判断フロー」を参考にして判断してください。
また、本資料で使用しているサンプルプログラムの動作を確認したＯＳの一覧を表 1.2 に示します。

表 1.1 データ読み込み方法

	データ読み込み方法	資料名	備考
1	THOR を使用する	GPM/TRMM データ読み込みプログラムガイド(THOR 編)	
2	IDL を使用する	GPM/TRMM データ読み込みプログラムガイド(IDL 編)	
3	C を使用する	GPM/TRMM データ読み込みプログラムガイド(C 言語編)	
4	FORTRAN を使用する	GPM/TRMM データ読み込みプログラムガイド(FORTRAN 編)	
5	Python を使用する	GPM/TRMM データ読み込みプログラムガイド(Python 編)	

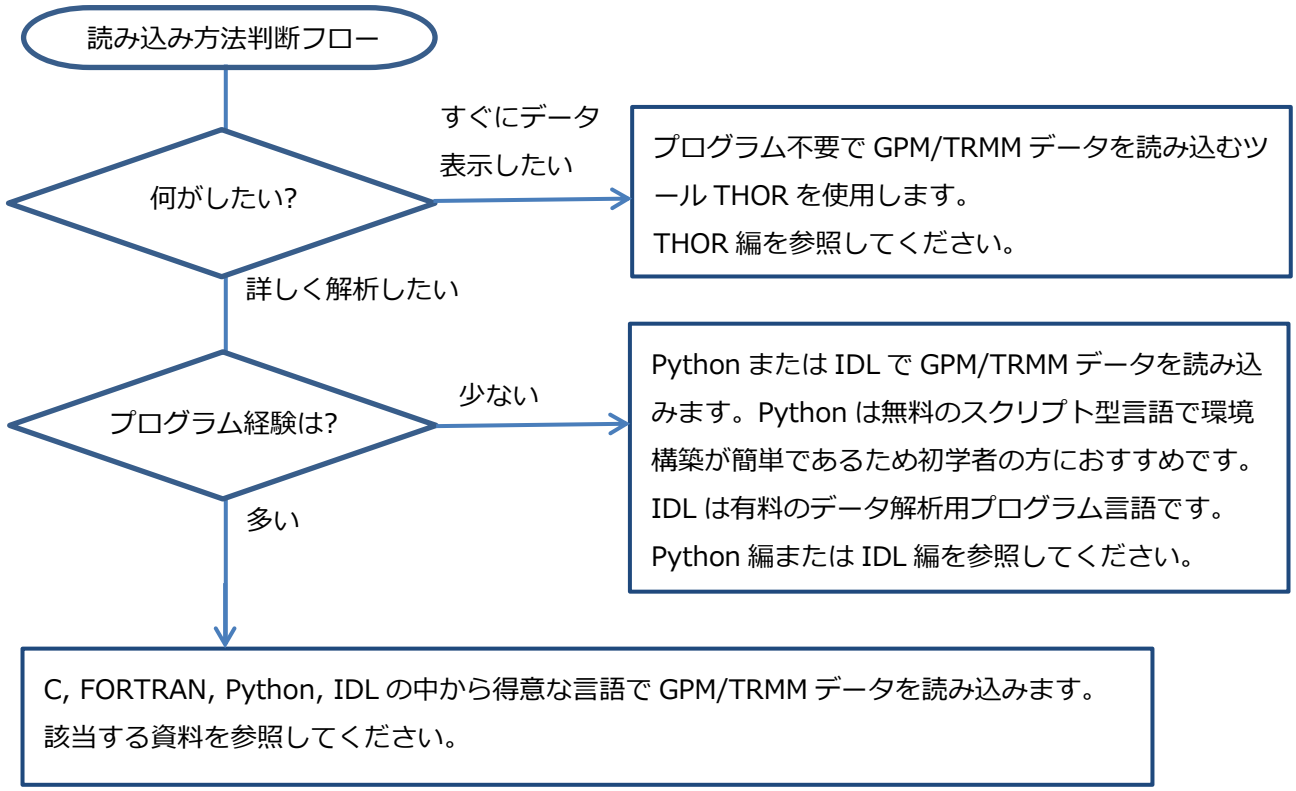


表 1.2 サンプルプログラム動作確認表

	サンプルプログラム	Linux	Windows	備考
1	C	○	—	
2	FORTRAN	○	—	
3	Python	○	○	
4	IDL	○	○	

2. GPM/TRMM データの入手方法

GPM/TRMM データは、G-Portal のサイト(https://www.gportal.jaxa.jp/gp/top.html)から取得することができます。　取得の際にはユーザ登録が必要になりますので、G-Portal のサイトの上部のメニューから「ユーザ登録／利用規約」を選択してユーザ登録を行ってください。

規約を読み「同意して次へ」をクリックします。

ユーザ登録画面になりますので、ユーザ登録を行います。

以降の手順や、ユーザ登録後のデータ取得方法については、「GPM データ利用ハンドブック」の「5.2 データ提供サービスの使い方」を参照してください。「GPM データ利用ハンドブック」の入手方法については「3. 関連文書、サンプルプログラムの入手方法」を参照してください。

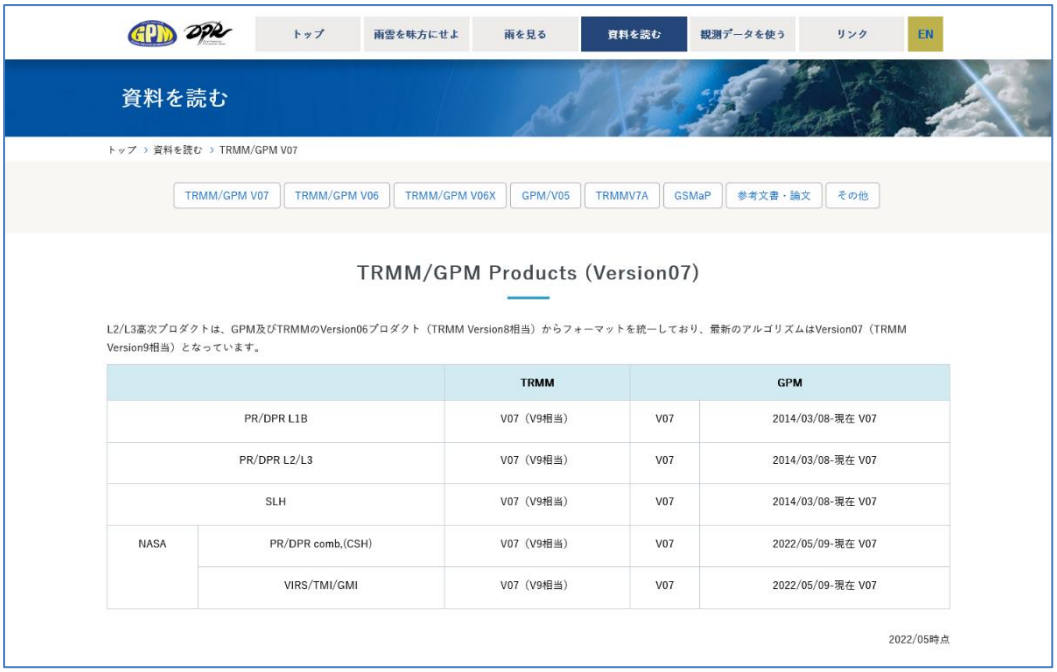
3. 関連文書、サンプルプログラムの入手方法

GPM/TRMM データの関連文書には、データ利用に関する文書と、プロダクトに関する文書があります。どちらも全球降水観測計画 GPM のサイト(https://www.eorc.jaxa.jp/GPM/index.html)のトップページ ＞ 資料を読む からダウンロードできます。また、本書で解説しているサンプルコードについてはトップページ ＞ 観測データを使う からダウンロードできます。

GPM データ利用に関する文書には以下のものがあります。

GPM データ利用ハンドブック

ファイル命名規約



「TRMM/GPM V07」をクリックするとプロダクトバージョン 07 の文書一覧が表示されます。Format Specification は各プロダクトのデータ仕様が記載されたドキュメントです。

本書で解説するプロダクトとプログラム、サンプルデータは以下の通りです。確認したプロダクトバージョンは GSMaP がバージョン 5 、その他はバージョン 7 です。

表 3.1 サンプルプログラム一覧

プロダクト	サンプルプログラム	サンプルデータ
L2DPR	readDPRL2_1.py	GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5
	readDPRL2_2.py	
L3DPR	readDPRL3.py	GPMCOR_DPR_1806_M_D3M_07X.EORC.h5
GSMaP	readGSMaP_netcdf.py	gsmap_now_rain.20211128.0500.nc
	readGSMaP_binary.py	gsmap_gauge_now.20211128.0500.dat.gz

4. Python について

Python は Windows、Linux/Unix、Mac など多くの OS で使うことのできるフリーのプログラミング言語です。読み書きしやすく、標準ライブラリが充実しており、同時に他のプログラミング言語やオブジェクトへの拡張性も備えています。また、近年ではビッグデータサイエンスや機械学習の分野で広く使われており、急激に使用者数を増やしています。

C や Fortran で衛星データを取り扱う際のハードルに衛星データ用 I/O ライブラリの導入と、Linux パッケージの依存関係がありました。Python はそれらの手間なしに衛星データをハンドリングできる選択肢の一つです。

5. Python のインストール、環境設定

以下は例です。Python の環境セットアップについては書籍やネット上の説明が充実しているので自分に合ったものを参照することをお勧めします。なお、Python にはレガシーバージョンの 2.7 系と現行の 3 系があり、互換性がありません。以降は 2.7 で説明します。3 を選択される場合は適宜読み替えてください。

① 統合開発環境ディストリビューションを頼る

Anaconda（→https://www.continuum.io/downloads）を入れましょう。依存関係を気にしなくていいのでおすすめです。パッケージ管理機能も含んでいます。

② ソースからビルドする（上級者向け）

Python（https://www.python.jp/ ）本体をインストールした後、パッケージ管理システムである PIP（あるいは conda, easy_install）を導入し、PIP を利用してモジュールを追加していくことになりますが、モジュール間の依存関係や OS の対応ビット数の違いに注意してください。

①か②の方法で Python がインストールできたら、足りないモジュール（＝ライブラリ）を追加します。このガイドで使ったり、衛星データ解析に最低限あるとよいと思われるものを挙げます。デフォルトで Anaconda に含まれているものを☑で示しています。

☑ Numpy ：配列演算モジュール
☑ Pandas：データ解析支援モジュール
☑ Matplotlib：描画モジュール
☐ Basemap：Matplotlib の地図描画支援サブモジュール
☑ h5py：HDF5 ファイルの I/O (GPM 用)
☐ pyHDF：HDF4 ファイルの I/O（TRMM 用）

6. 初歩の練習

① DPR L2 データ

まず HDF5 形式の DPR L2 のデータを図化してみましょう。

下準備として、readDPRL2.py を任意の作業フォルダに保存してください。作業フォルダは、半角スペースや全角文字を含まないディレクトリとするのが安全です。

次に衛星データを DL します。JAXA のデータ配布サイトの G-Portal（https://www.gportal.jaxa.jp/gp/）でユーザー登録（「２．GPM データの入手方法」を参照してください）の上、トップページの「衛星からの検索」＞「ＧＰＭ」＞「DPR」＞「ＬＥＶＥＬ２」＞「 DPR L2 降水量」を選択。２期間指定タブで、観測年月日に「2016/6/21～2016/6/21」を指定、３範囲指定タブで、「全球指定」を選択し、「検索する」をクリックします。検索結果一覧から観測開始時刻が 04:50:18 のデータを選択します。

そのまま「次へ」を選択し、ダウンロードしてください。約 256MB あります。ZIP 圧縮されていたら解凍し、作業フォルダに置きます。

念のため、必要なモジュールがインストールできていることを確認しましょう。端末・ターミナルまたはコマンドプロンプトを開き、以下のように入力します。

```
> python
```

(Python が立ち上がったメッセージ)

```
>>> import h5py
>>> import numpy
>>> import matplotlib
```

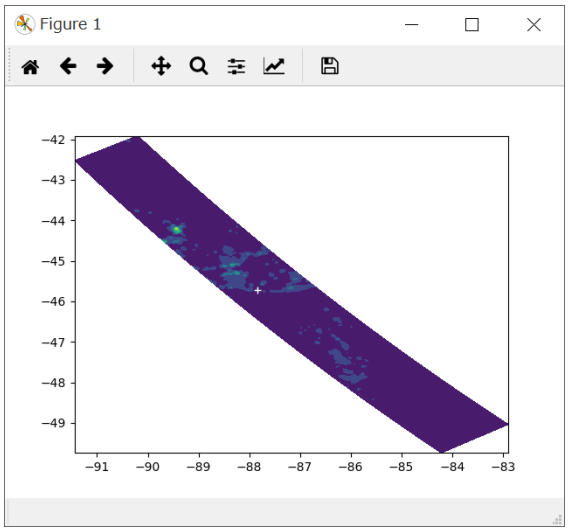
エラーが出たら、正しくインストールされていないことになりますので対処してください。何も起こらなければ OK です。ターミナルは閉じます。

さて準備が整いました。作業フォルダで端末・ターミナルまたはコマンドプロンプトを開き、

```
> python readDPRL2.py
```

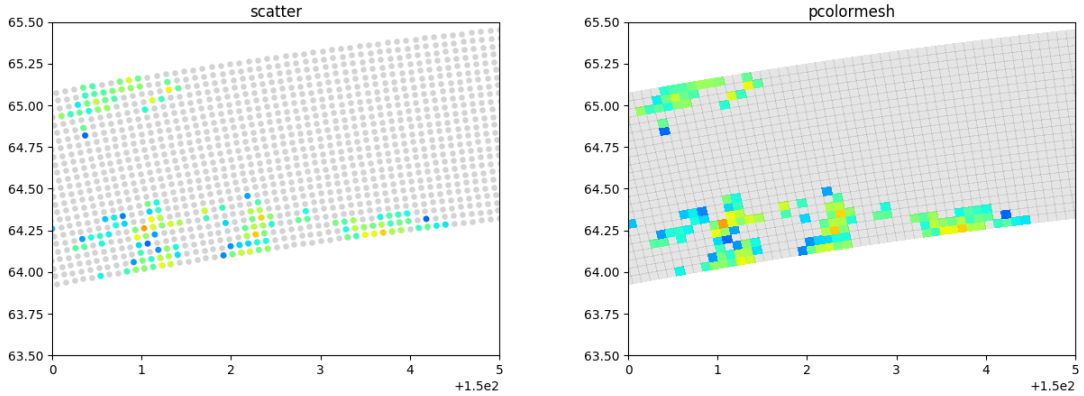
で以下の標準出力と図のウィンドウが表示されるはずです！

```
Estimated Surfase Rain at 270.310120,-45.718342:   3.067931
Number of Rainy cells/all: 15410/198375
```



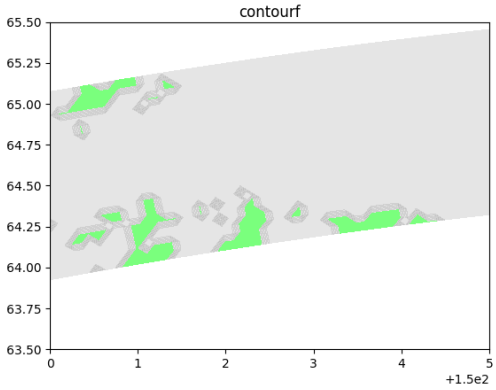
見てのとおり、L2 のデータは衛星が観測したパスに沿ったデータが入っています。サンプルプログラムではファイルに含まれる全データを読み出していますが、あらかじめ必要なデータの箇所が分かっていれば、読み出しの時点で切り出し範囲を指定することで処理を早くすることができます。

プログラムで使用している Matplotlib は基本的かつ強力な描画モジュールで、様々なプロット方法があります。試しに DPR L2 の観測パスに沿って地表面レーダ反射因子（/SLV/zFactorCorrectedESurface）を拡大したものを 3 つの方法で描画してみます。



Pyplot.catter(左)と pyplot.pcolormesh（右）

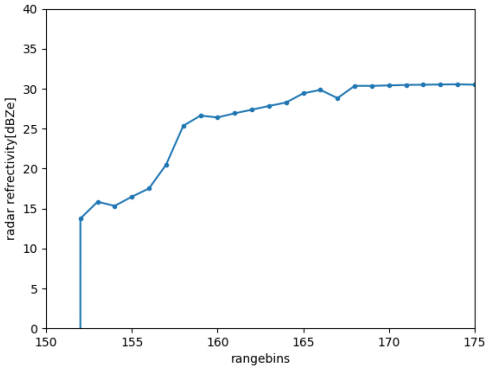
設定はそれぞれ細かくカスタマイズ可能です。この例では欠損値を灰色で指定しています。



Pyplot.contourf

塗り分け等高線の contourf はレーダ反射因子のように隣接するピクセル間の値の差が大きいものを拡大するには不向きです。ある程度大きなスケールや、GSMaP などの降雨分布などでは適度にスムージングされ、美しい画像が得られます。

DPR の特色は、降水に関する物理量の鉛直分布が得られることです。降水量、レーダ反射因子、粒子粒形分布などが三次元データとして提供されています。readDPRL2_2.py は三次元物理量の等価レーダ反射因子 SLV/zFactorCorrected を読み出し、あるフットプリントについてプロファイルを表示するプログラムです。



readDPRL2_2.py 出力例

以下は readDPRL2_1.py のソースコードです。

```
 1:import numpy as np
 2:import h5py
 3:import matplotlib.pyplot as plt
 4:import matplotlib.cm as cm
 5:
 6:# set name of file and observation mode
 7:filename='GPMCOR_DPR_2112070007_0140_044170_L2S_DD2_07A.h5'
 8:mode='FS'
 9:
10:with h5py.File(filename,"r") as infile:
11:    #convert data to numpy ndarray
12:    Lat=np.array(infile[mode+'/Latitude']).T
13:    Lon=np.array(infile[mode+'/Longitude']).T
14:    Lon=np.unwrap(Lon)
15:    precipRateESurf=np.array(infile[mode+'/SLV/precipRateESurface']).T
16:    ZeESurf=np.array(infile[mode+'/SLV/zFactorCorrectedESurface']).T
17:    # see file specification guide about the directories and parameters
18:    Nscan=Lon.shape[0]  # total number of scan
19:    Nray=Lon.shape[1]   # number of FOV(number of angle)
20:
21:
22:print 'Estimated Surfase Rain at %f,%f: %f'%
(Lon[9][7093],Lat[9][7093],precipRateESurf[9][7093]) # precepRateESurface at
Scan=7093,FOV=9
23:print 'Number of Rainy cells/all: %d/%d'% (len(np.where(precipRateESurf>0)[0]),
precipRateESurf.size)
24:
25:#slicing data. new array contains data of Scan 7000 to 7200
26:lon=Lon[:, 7000:7200]
27:lat=Lat[:, 7000:7200]
28:rr=precipRateESurf[:, 7000:7200]
29:
30:# calculate statistics of numpy ndarray (for example)
31:Mean=rr.mean()
32:STD=rr.std()
33:
34:#plot with Matplotlib
35:plt.plot(Lon[9][7093],Lat[9][7093],"+", c='white')
36:plt.pcolormesh(lon,lat,rr)
37:plt.show()
38:plt.close()
```

mumpy(配列演算モジュール)を np という名前で使用する宣言です。

matplotlib(描画モジュール)の pyplot 機能を plt という名前で使用する宣言です。

HDF5 ファイル名を指定しています。

HDF5 ファイルを読み込んでいます。

読み込んだデータから FS/Latitude, FS/Longitude を取り出しています。

読み込んだデータから FS/SLV/precipRateESurface を取り出しています。

読み込んだデータから FS/SLV/zFactorCorrectedESurface を取り出しています。

取り出したデータの一部を画面に出力しています

取り出した precipRateESurface をプロットしています。

描画しています。

以下は readDPRL2_2.py のソースコードです。

```
 1:import numpy as np
 2:import h5py
 3:import matplotlib.pyplot as plt
 4:
 5:#filename='GPMCOR_DPR_1606210450_0622_013140_L2S_DD2_05A.h5'
 6:mode='FS'
 7:
 8:
 9:with h5py.File(filename,"r") as infile:
10:    #convert data to numpy ndarray
11:    Lat=np.array(infile[mode+'/Latitude']).T
12:    Lon=np.array(infile[mode+'/Longitude']).T
13:    Lon=np.unwrap(Lon)
14:    precipRateESurf=np.array(infile[mode+'/SLV/precipRateESurface']).T
15:    Ze=np.array(infile[mode+'/SLV/zFactorCorrected']).T
16:    # see file specification guide about the directories and parameters
17:    Nscan=Lon.shape[0]  # total number of scan
18:    Nray=Lon.shape[1]   # number of FOV(number of angle)
19:
20:Ze.shape #nbin,nray,nscan
21:
22:plt.plot(Ze[:,9,7093],'.-')
23:plt.ylim(0,40)
24:plt.xlim(150,175)
25:plt.xlabel('rangebins')
26:plt.ylabel('radar refrectivity[dBZe]')
27:plt.show()
28:plt.close()
```

HDF5 ファイル名を指定しています。

HDF5 ファイルを読み込んでいます。

読み込んだデータから FS/SLV/precipRateESurface を取り出しています。

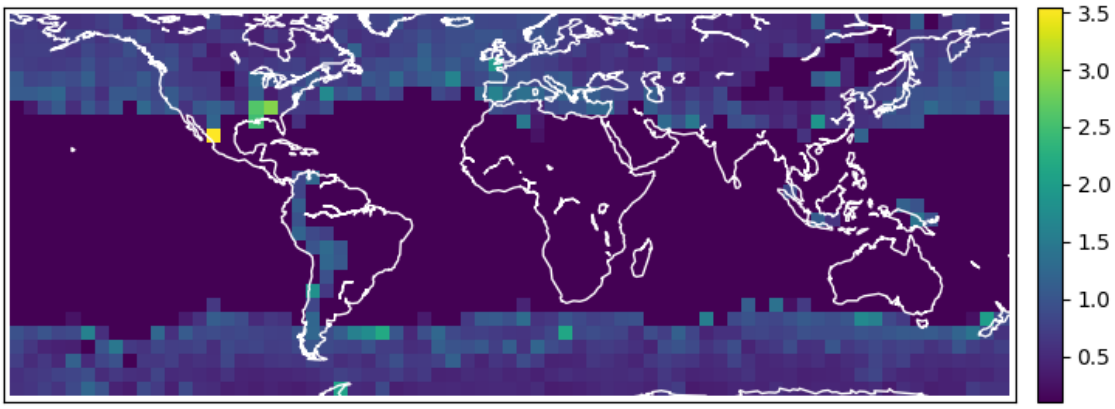
読み込んだデータから FS/SLV/precipRateESurface を取り出しています。

取り出した precipRateESurface をプロットしています。

描画しています。

② DPR L3 データ

DPR レベル 3 データは L2 で提供されている物理量の統計量となります。L3 では 0.25 度格子（G2）および 5 度格子（G1）でデータが格納されています。先ほどと同じ要領で readDPRL3.py をデータを置いたフォルダで実行してください。下図のような出力が得られます。



readDPRL3.py 出力例

以下は readDPRL3.py のソースコードです。

```
 1:import numpy as np
 2:import h5py
 3:import matplotlib.pyplot as plt
 4:from mpl_toolkits.basemap import Basemap
 5:
 6:# set name of file and grid
 7:filename='GPMCOR_DPR_1806_M_D3M_07X.EORC.h5'
 8:folder='/FS'
 9:grid='/G1'
10:
11:if grid=='/G1':
12:    spres=5.0
13:    xnum=72
14:    ynum=28
15:if grid=='/G2':
16:    spres=0.25
17:    xnum=1440
18:    ynum=536
19:
20:with h5py.File(filename,"r") as infile:
21:    snowRateNSurf=np.array(infile[folder+grid+'/snowRateNearSurface/mean']).T
22:    # see file specification guide about the directories and parameters
23:    # note the sort of array element is inverted to as in file specification
24:# in this program
25:    # that is the reason of transpose matrix method ".T"
26:
27:# Set Lat&Lon grid
28:Lo=np.array([-180.0+spres/2+i*spres for i in range(0,xnum)])
29:La=np.array([-70.0+spres/2+i*spres for i in range(0,ynum)])
30:Lon,Lat=np.meshgrid(Lo,La)
31:
32:#plot with Matplotlib
33:im=plt.pcolormesh(Lon,Lat,snowRateNSurf[:,:,4,2,2], vmin=0.1)
34: # [lon, lat, ch=4:DPRMS, surfacetype=2:all, raintype=2:all]
35:#set map
36:m=Basemap(projection='cyl',
37:        resolution='c',
38:        llcrnrlat=-70, urcrnrlat=70, llcrnrlon=-180, urcrnrlon=180)
39:m.drawcoastlines(color='white')
40:x,y=m(Lon, Lat) #compute map projection
41:# set colorbar
42:cb=m.colorbar(im, "right", size="2.5%")
43:
44:plt.show()
45:plt.close()
```

地図を描画する Basemap を使用する事を宣言しています。

HDF5 ファイル名を指定しています。

HDF5 ファイルを読み込んでいます。

読み込んだデータから Grids/G1/snowRateNearSurface/mean を取り出しています。

地図上に snowRateNSurf データをカラープロットしています。

投影法(cyl:正距円筒図法)、解像度(c:略)、端の緯度・経度を指定しています。

左下の緯度

右上の緯度

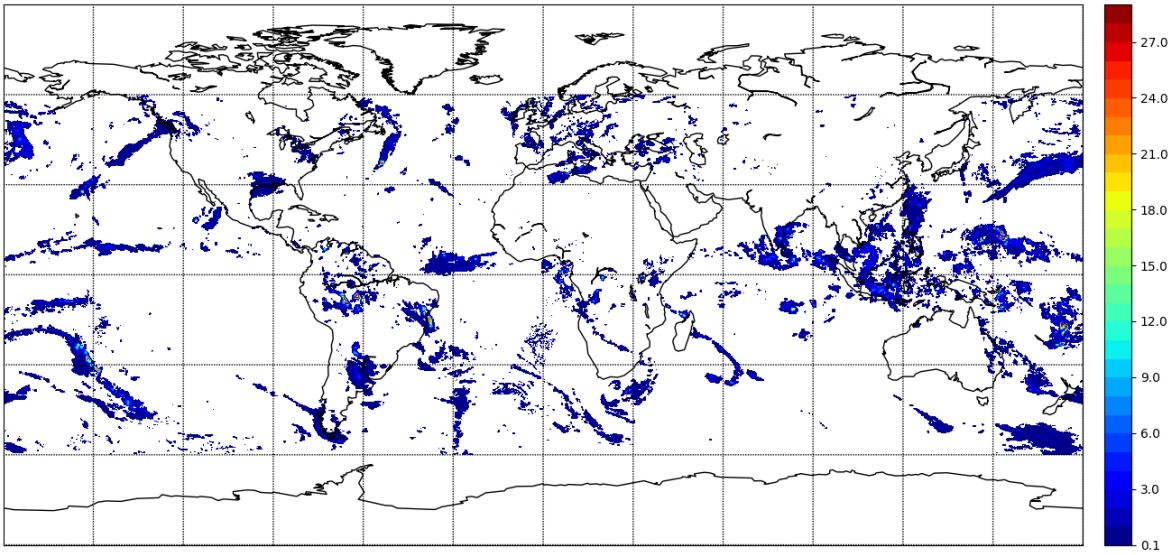
左下の経度

右上の経度

描画しています。

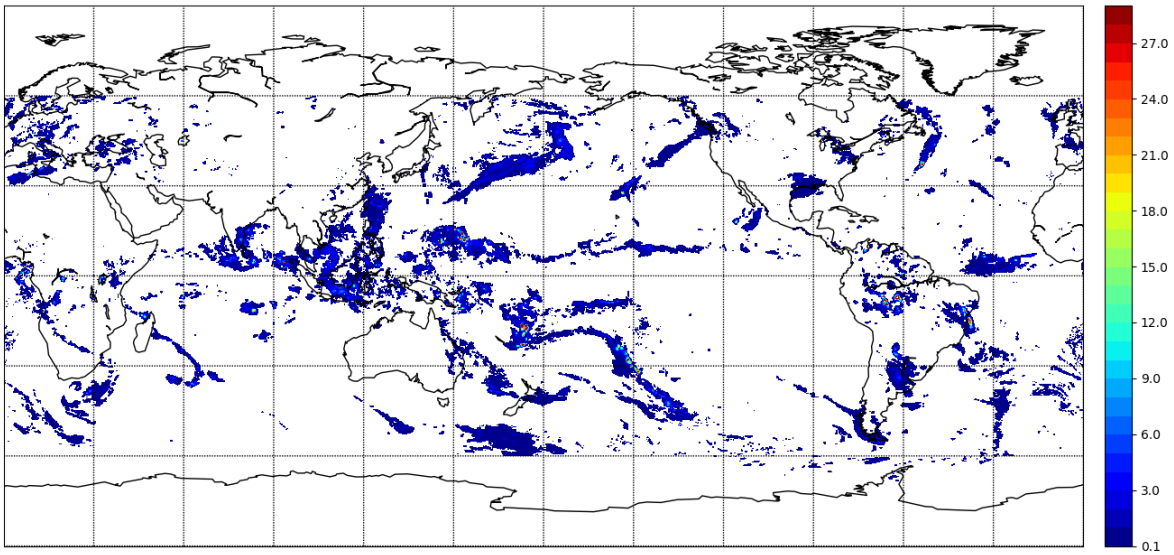
③　GSMaP データ

GSMaP は複数の衛星搭載マイクロ波放射計およびサウンダのデータを合成し一時間おきの世界の降水分布を提供しています。GSMaP は様々な形式（HDF5, Text, binary, netCDF, KML, geoTiff）で提供されています。データファイルを開く記述を各フォーマットに対応させれば後は同じ流れで処理することができます。



readGSMaP_netcdf.py 出力例

* binary データを使用した場合、Basemap1.0.5 以前では右端と左端がつながって横線が入ってしまいます。バイナリでは 0°～-0°でデータが入っているのに対して、HDF などは-180°～180°で入っているため。表示範囲に合わせてデータをソートする、プロットを contourf でなく plot や pcolor（処理が重いので注意）とすることで解決できます。Map の表示を 0°～360°にさせて対応することもできます。



readGSMaP_binary.py 出力例

以下は readGSMaP_netcdf.py のソースコードです。

```
 1:import numpy as np
 2:import matplotlib.pyplot as plt
 3:import matplotlib.cm as cm
 4:from mpl_toolkits.basemap import Basemap
 5:
 6:'''
 7:# in case of HDF
 8:import h5py
 9:h5='GPMMRG_MAP_1702010000_H_L3S_MCH_04B.h5'
10:
11:with h5py.File(h5,"r") as infile:
12:    Lat=np.array(infile['Grid'+'/Latitude'])
13:    Lon=np.array(infile['Grid'+'/Longitude'])
14:    hprecipRateGC=np.array(infile['Grid'+'/hourlyPrecipRateGC'])
15:# end of HDF case
16:'''
17:# in case of netCDF
18:import netCDF4
19:nc=netCDF4.Dataset('gsmap_now_rain.20211128.0500.nc','r')
20:#nc=netCDF4.Dataset('gsmap_now_flag.20211128.0500.nc','r')
21:#nc=netCDF4.Dataset('gsmap_nrt_rain.20211128.0100.nc','r')
22:Lon=nc.variables['Longitude'][:]
23:Lat=nc.variables['Latitude'][:]
24:hprecipRateGC=nc.variables['hourlyPrecipRateGC'][0,:,:]
25:nc.close()
26:# end of netCDF case
27:
28:#plot with Matplotlib
29:fig=plt.figure(figsize=(20,20))
30:# set the color interval
31:interval=list(np.arange(1,30,1))
32:interval.insert(0,0.1)
33:#set colormap
34:cmap=cm.jet
35:cmap.set_under('w', alpha=0)
36:
37:#set map
38:m=Basemap(projection='cyl',
39:        resolution='c',
40:        llcrnrlat=-90, urcrnrlat=90, llcrnrlon=-180, urcrnrlon=180)
41:m.drawcoastlines(color='black')
42:m.drawmeridians(np.arange(0,360,30))
43:m.drawparallels(np.arange(-90,90,30))
44:x,y=m(Lon, Lat) #compute map projection
45:im=plt.contourf(x,y,hprecipRateGC, interval, cmap=cmap, latlon=True)
46:# set colorbar
47:cb=m.colorbar(im, "right", size="2.5%")
48:
49:plt.show()
50:plt.close()
```

HDF5 ファイル名を指定しています。

HDF5 ファイルを読み込んでいます。

netCDF ファイルを読み込んでいます。

投影法(cyl:正距円筒図法)、解像度(c:略)、端の緯度・経度を指定しています。

海岸線、経度線、緯度線を引いています。

描画しています。

以下は readGSMaP_binary.py のソースコードです。

```
 1:import numpy as np
 2:import matplotlib.pyplot as plt
 3:import matplotlib.cm as cm
 4:from mpl_toolkits.basemap import Basemap
 5:
 6:# in case of binary
 7:import gzip
 8:import struct
 9:filename='gsmap_gauge_now.20211128.0500.dat.gz'
10:i=0
11:rain=[0]*3600*1800
12:with gzip.open(filename, "rb") as f:
13:    while True:
14:        data=f.read(4)
15:        if len(data) < 4:
16:            break
17:        rain[i]=struct.unpack('f',data)[0]
18:        i=i+1
19:hprecipRateGC=np.reshape(rain, (1800,3600))
20:# generate meshgrid because GSMaP binary does not contain locations
21:lo=[5+10*i for i in range(0,3600)]
22:Lo=0.01*np.array(lo)
23:la=[8995-10*i for i in range(0,1800)]
24:La= 0.01*np.array(la)
25:Lon,Lat=np.meshgrid(Lo,La)
26:# end of binary case
27:
28:#plot with Matplotlib
29:fig=plt.figure(figsize=(20,20))
30:# set the color interval
31:interval=list(np.arange(1,30,1))
32:interval.insert(0,0.1)
33:#set colormap
34:cmap=cm.jet
35:cmap.set_under('w', alpha=0)
36:
37:#set map
38:m=Basemap(projection='cyl',
39:        resolution='c',
40:        llcrnrlat=-90, urcrnrlat=90, llcrnrlon=0, urcrnrlon=360)
41:m.drawcoastlines(color='black')
42:m.drawmeridians(np.arange(0,360,30))
43:m.drawparallels(np.arange(-90,90,30))
44:x,y=m(Lon, Lat) #compute map projection
45:im=plt.contourf(x,y,hprecipRateGC, interval, cmap=cmap, latlon=True)
46:# set colorbar
47:cb=m.colorbar(im, "right", size="2.5%")
48:
49:plt.show()
50:plt.close()
```

雨量計補正データを指定しています。

雨量計補正データを読み込んでいます。

読み込んだ雨量計補正データを
1800*3600 の配列に変換しています。

投影法(cyl:正距円筒図法)、解像度(c:略)、端の緯度・経度を指定
しています。

海岸線、経度線、緯度線を引いています。

描画しています。

改版履歴

版数	日付	改版内容	備考
1	2018/1/24		
2	2018/3/15	3. 関連文書、サンプルプログラムの入手方法 : 表 3.1 サンプルプログラム一覧を追加	
3	2019/2/7		
4	2021/12/1	GSMaP プロダクトバージョン 5 対応。 6. 初歩の練習 : ソースコード修正 readGSMaP_netcdf.py、readGSMaP_binary.py	
5	2021/12/6	1. GPM/TRMM プロダクトバージョン 7 に修正。 3. 関連文書とサンプルプログラムの入手方法修正	
6	2021/12/24	表 3.1 サンプルデータを V7 に更新 6 コードの説明を V7 に合わせて修正	