GPM/TRMM data reading program guide (Python version)

2021/12/24

6th ed.

This document describes how to create a program (in Python) to read data from the Global Precipitation Measurement/Monitoring Mission (GPM/TRMM).
The sample programs described in this document have been tested with product version 07 for GPM/TRMM and with product version 5 for GSMaP.

Table of Contents

Introduction

This document explains how to read in GPM/TRMM data using Python.
The GPM and TRMM formats have been unified since version 06 products (equivalent to TRMM version 8), and the latest algorithm is version 07 (equivalent to TRMM version 9). The latest algorithm is version 07 (equivalent to TRMM version 9), which can be read in the same way in this sample program.
In addition to Python, there are other ways to read GPM data, as shown in Table 1.1. To determine which method to use, please refer to the "Loading Method Decision Flow" on the next page.
Table 1.2 lists the operating systems on which the sample programs used in this document were tested.

Table 1.1 Data loading methods

	Data loading method	Name of material	remarks
1	Using THOR	GPM/TRMM Data Loading Program Guide (THOR Edition)	
2	Use IDL	GPM/TRMM Data Loading Program Guide (IDL version)	
3	Use C	GPM/TRMM Data Loading Program Guide (C language version)	
4	Using FORTRAN	GPM/TRMM Data Loading Program Guide (FORTRAN Edition)	
5	Using Python	GPM/TRMM data reading program guide (Python version)	

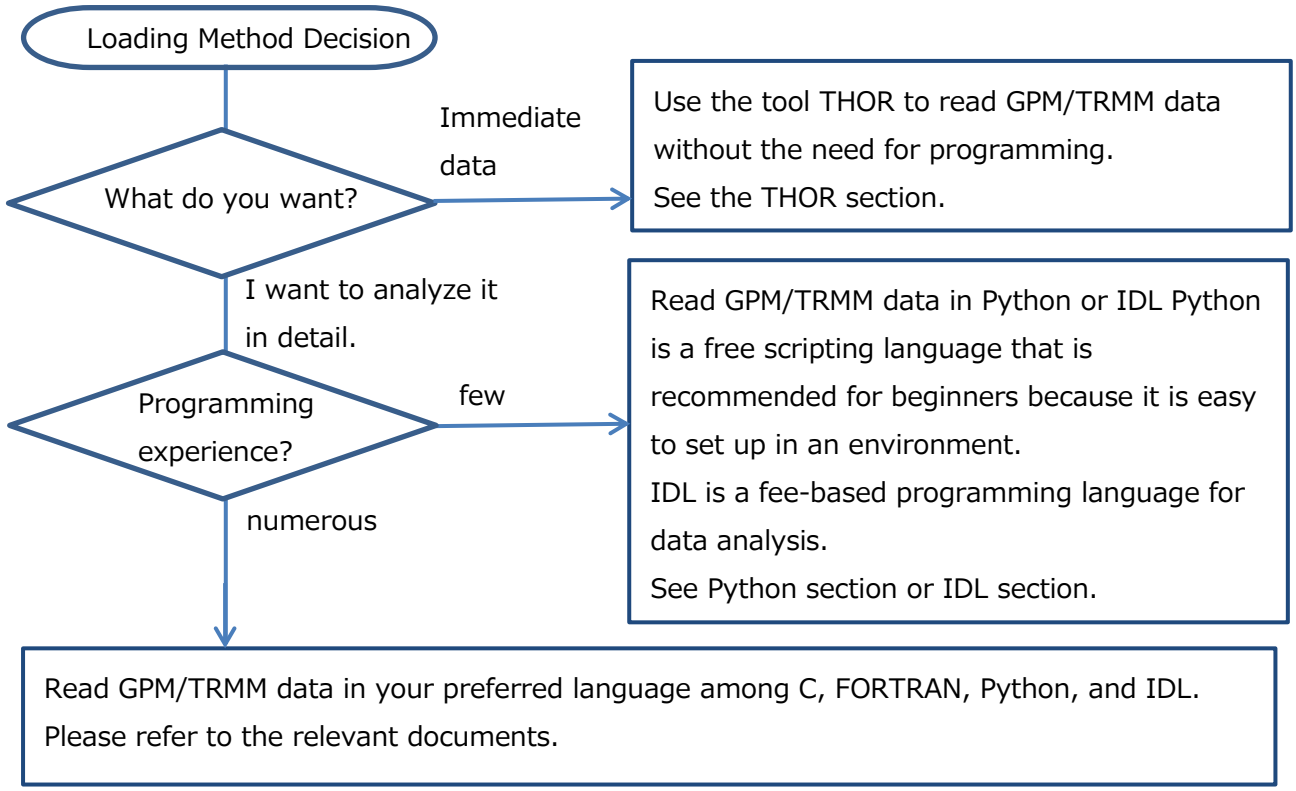


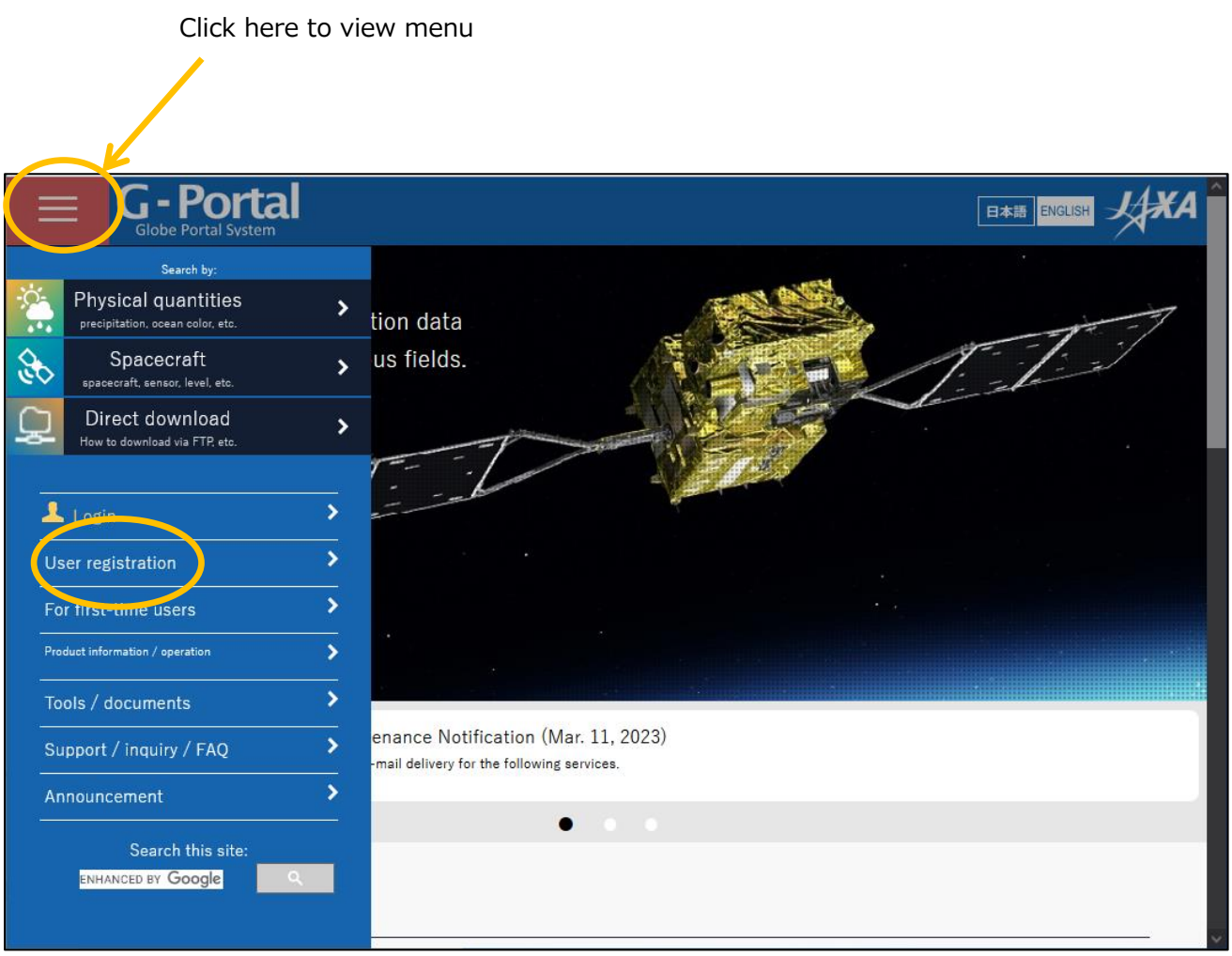
Table 1.2 Sample Program Operation Check Table

	sample program	Linux	Windows	remarks
1	c	○	-	
2	FORTRAN	○	-	
3	Python	○	○	
4	IDL	○	○	

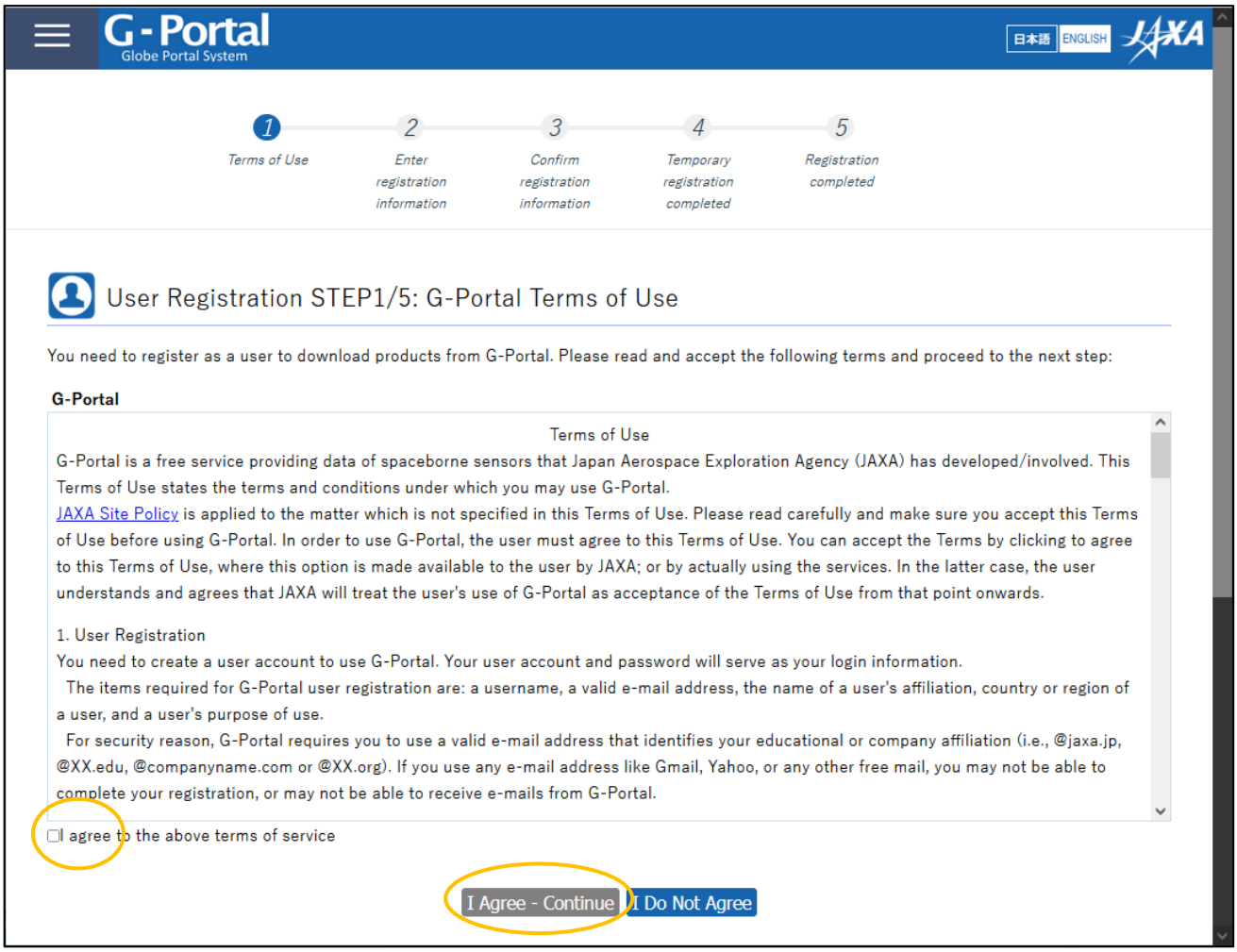
○ : Operation is confirmed. －: Operation is unconfirmed.

2. how to obtain GPM/TRMM data

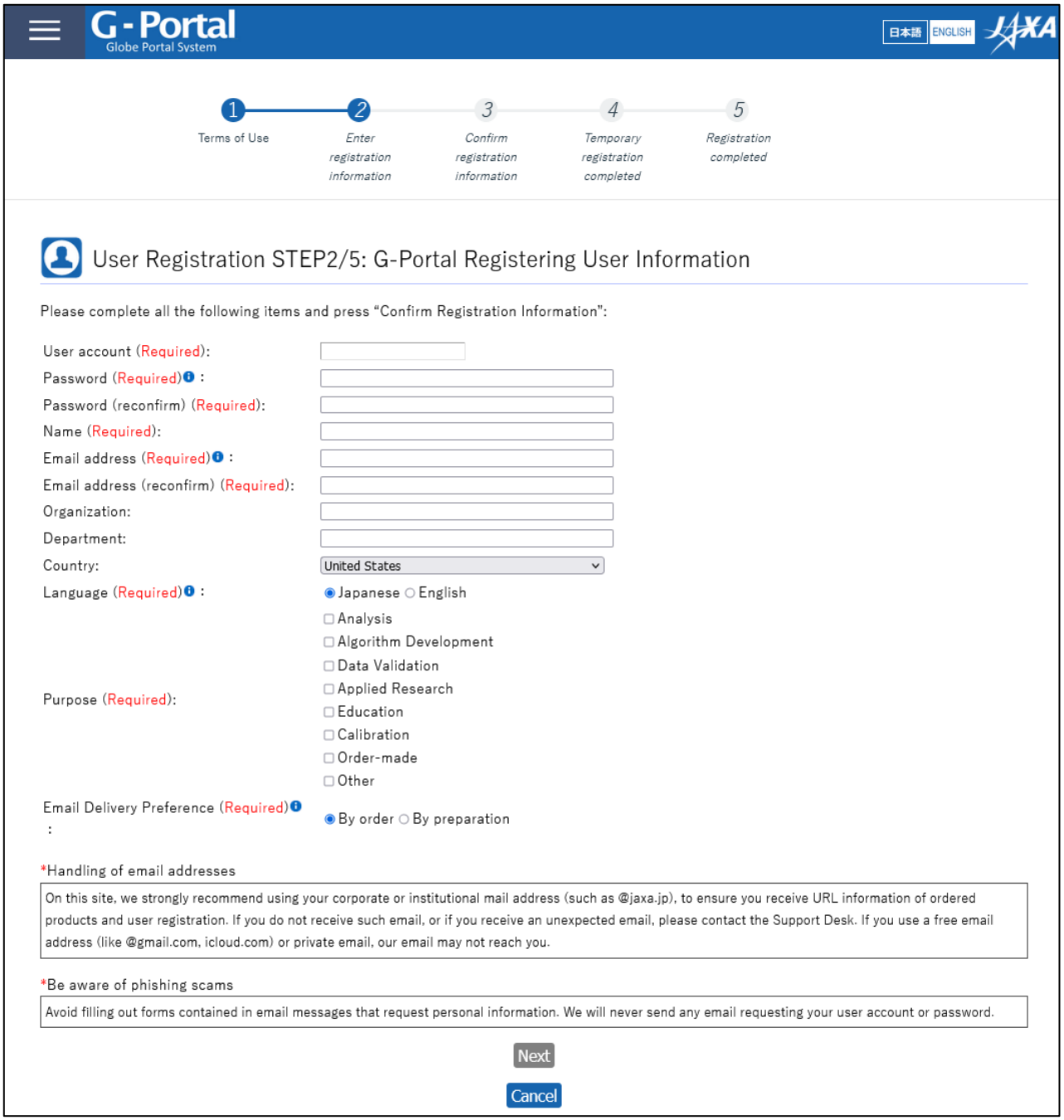
GPM/TRMM data can be obtained from the G-Portal site (https://www.gportal.jaxa.jp/gp/top.html). User registration is required to obtain the data. Please select "User Registration/Terms of Use" from the menu at the top of the G-Portal site to register as a user.



Read the terms and conditions and click "Agree and Next.



You will be taken to the user registration screen.



For the subsequent procedures and how to obtain data after user registration, please refer to "5.2 How to Use the Data Providing Service" in the "GPM Data Users Handbook". For information on how to obtain the "GPM Data Users Handbook," please refer to "3.

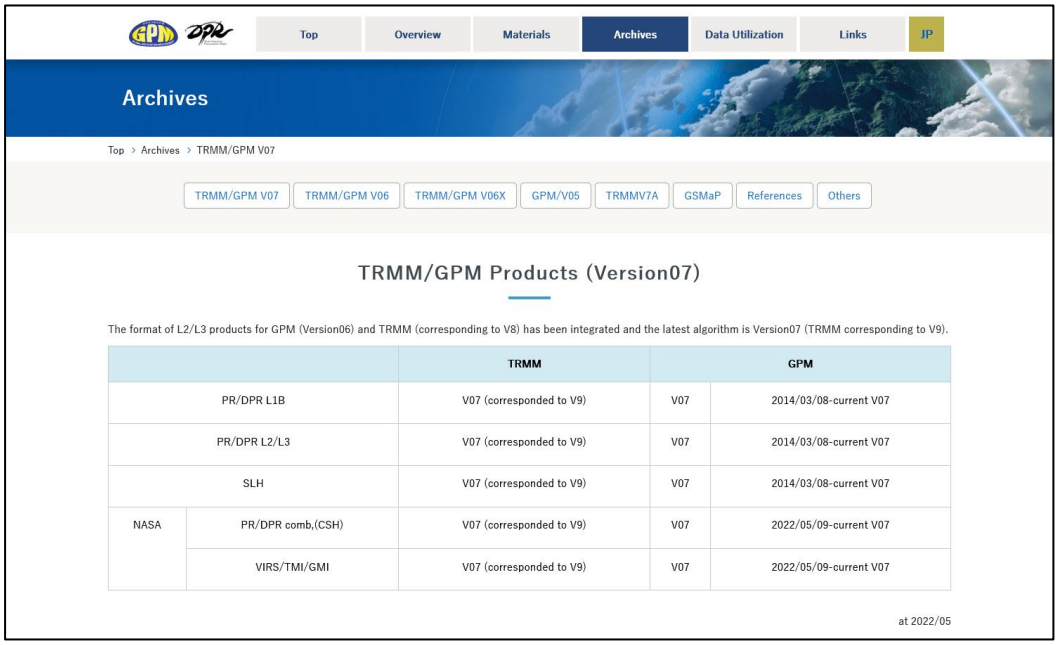
3. how to obtain related documents and sample programs

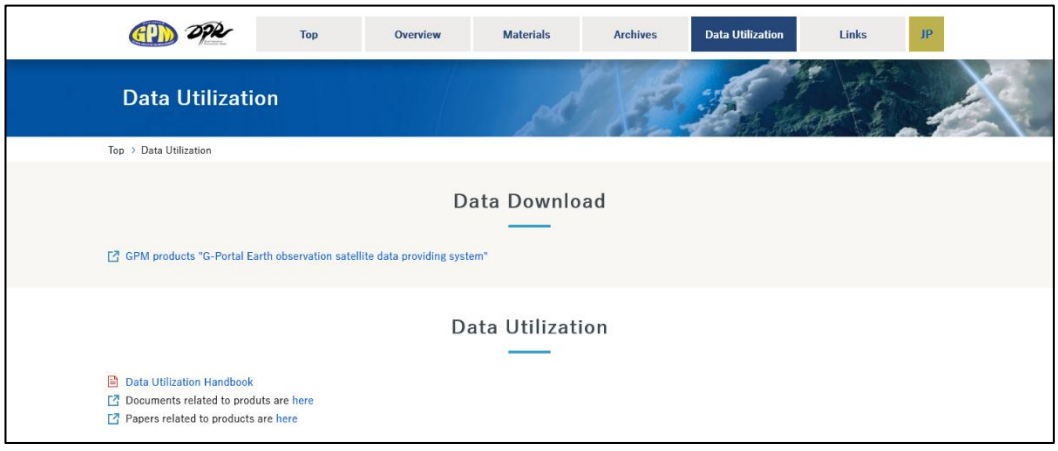
There are two types of documents related to GPM/TRMM data: documents related to data use and documents related to products. Both documents can be downloaded from the Global Precipitation Measurement Project (GPM) website (https://www.eorc.jaxa.jp/GPM/index.html). You can also download the sample codes described in this document from Top Page > Data Utilization

Documentation for GPM data use includes

GPM Data Application Handbook

file naming convention





Click "TRMM/GPM V07" to see the list of documents for product version 07.

The products, programs, and sample data described in this document are as follows. The product versions checked are version 5 for GSMaP and version 7 for the others.

Table 3.1 List of Sample Programs

product	sample program	sample data
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

About Python

Python is a free programming language that can be used on many operating systems, including Windows, Linux/Unix, and Mac. It is easy to read and write, has an extensive standard library, and at the same time is extensible to other programming languages and objects. In recent years, it has also been widely used in the fields of big data science and machine learning, and the number of users is rapidly increasing.

Python is an alternative to C and Fortran for handling satellite data without the need to install I/O libraries for satellite data and the dependency on Linux packages.

5.Python installation and configuration

The following is an example; we recommend that you refer to the appropriate Python environment setup, which is well explained in books and on the Internet. Note that there are two legacy versions of Python, 2.7 and the current 3, which are not compatible with each other. If you choose to use 3, please read the following as appropriate.

Rely on integrated development environment distribution

Put in Anaconda (-> https://www.continuum.io/downloads). It is recommended because you don't have to worry about dependencies. It also includes package management features.

Build from source (for advanced users)

After installing Python (https://www.python.jp/) itself, you will install PIP (or conda, easy_install), a package management system, and add modules using PIP, but be aware of

dependencies between modules and Be aware of the differences in the number of bits supported by the OS.

Once Python has been installed using either method (1) or (2), add the missing modules (i.e., libraries). The following is a list of those that we will use in this guide or that we think you should have at a minimum for satellite data analysis. The ones included in Anaconda by default are indicated by ☑.

- ☑ Numpy: Array arithmetic module
- ☑ Pandas: data analysis support module
- ☑ Matplotlib: drawing module
- ☐ Basemap: Map drawing support sub-module of Matplotlib
- ☑ h5py: I/O for HDF5 files (for GPM)
- ☐ pyHDF: I/O for HDF4 files (for TRMM)

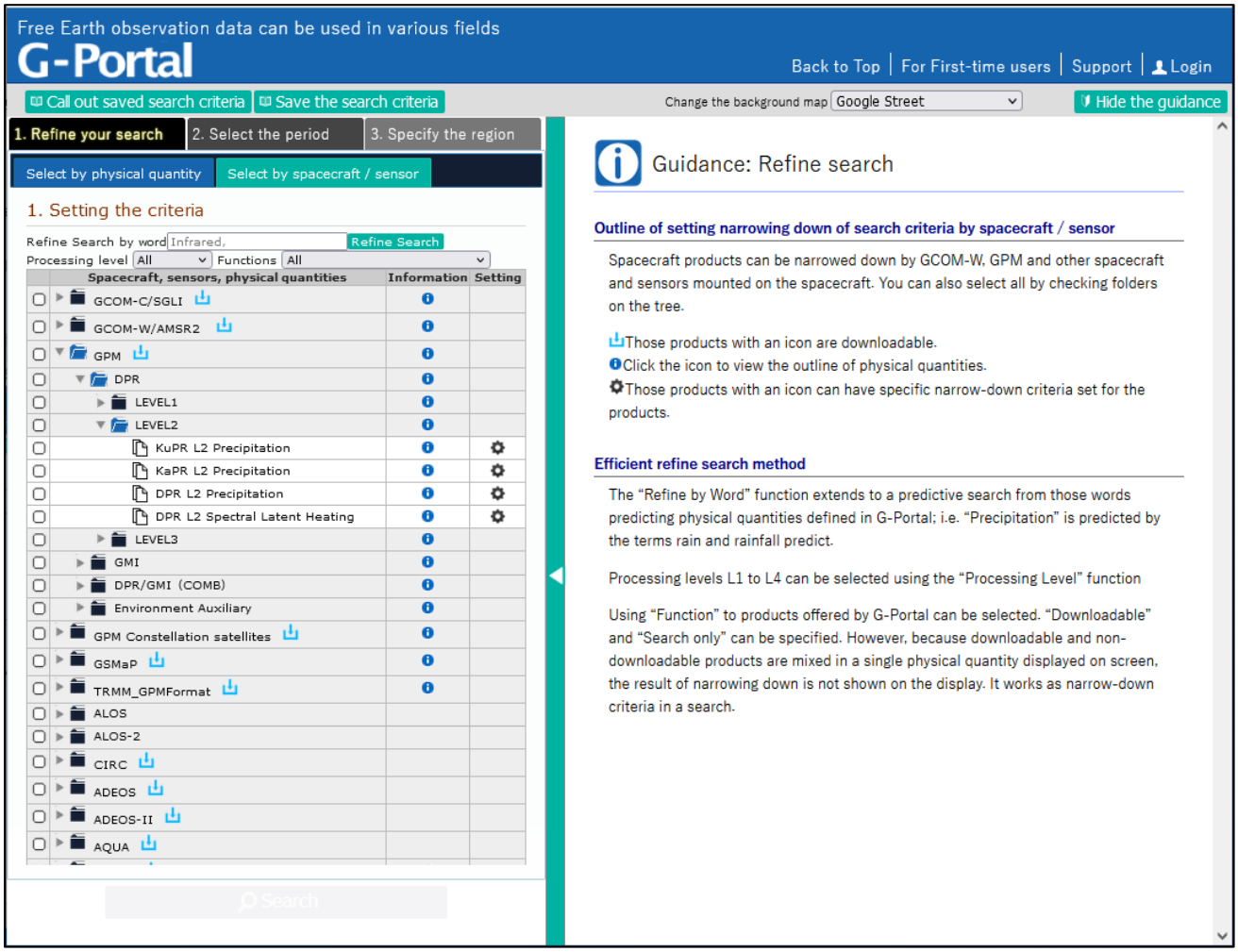
6. rudimentary practice

① DPR L2 data

Let's start by charting the DPR L2 data in HDF5 format.

As a preliminary step, save readDPRL2.py in any working folder. It is safe to use a working folder that does not contain spaces or double-byte characters.

Next, download the satellite data by registering as a user at JAXA's data distribution site G-Portal (https://www.gportal.jaxa.jp/gp/) (see "2. How to obtain GPM data"), and select "Search from satellites" > "GPM" > "DPR" > "LEVEL2" > "DPR L2 Precipitation" from the top page. DPR" > "LEVEL2" > "DPR L2 Precipitation"; 2. In the Period tab, select "6/21/2016 to 6/21/2016" for the observation date; 3. In the Range tab, select "Global"; and click "Search. Select the data whose observation start time is 04:50:18 from the list of search results.



Select "Next" to download the file. If the file is ZIP compressed, unzip it and place it in your working folder.

Just to be sure, check that the required modules have been installed. Open a terminal, terminal or command prompt and type

```
> python
```

(Python started up message)

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

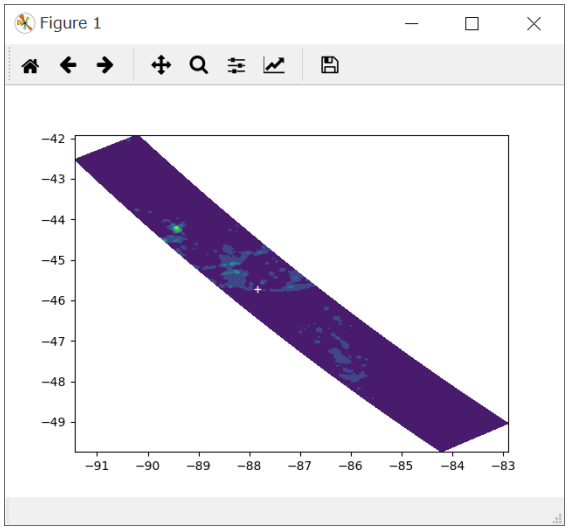
If an error occurs, it means that the software was not installed correctly and should be addressed. If nothing happens, it is OK. Close the terminal.

Now we are ready. Open a terminal, terminal, or command prompt in your working folder, and click on the

```
> python readDPRL2.py
```

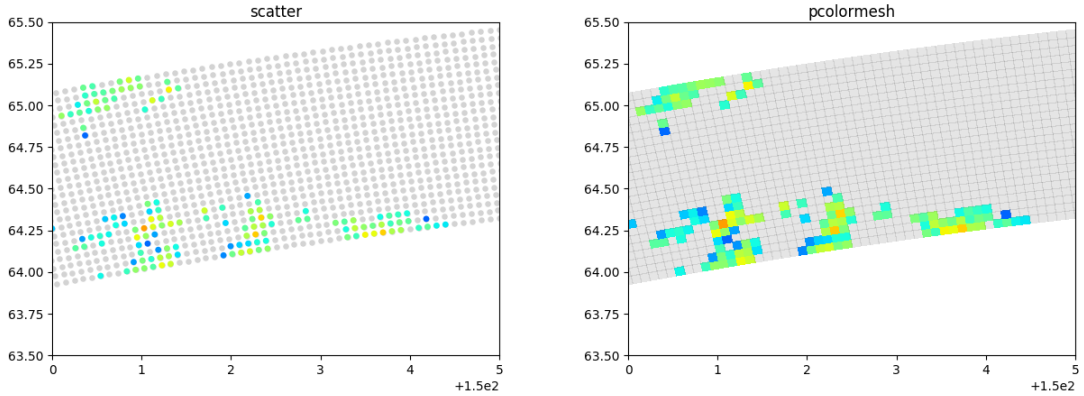
You should see the following standard output and figure window!

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



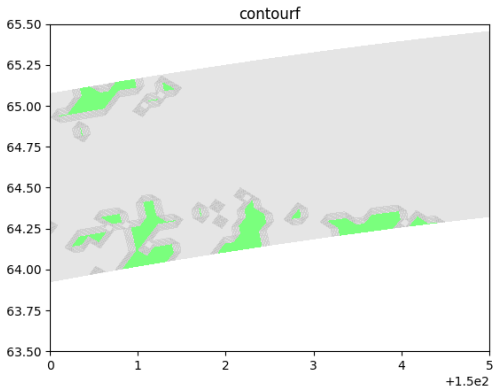
As you can see, the L2 data contains data along the path observed by the satellite. The sample program reads out all the data contained in the file, but if you know in advance where you need the data, you can speed up the process by specifying a cutout range at the time of reading out.

The Matplotlib used in the program is a basic and powerful drawing module that offers a variety of plotting methods. As a test, let's try drawing an expanded ground surface radar reflection factor (/SLV/zFactorCorrectedESurface) along the DPR L2 observation path in three different ways.



Pyplot.cater (left) and pyplot.pcolormesh (right)

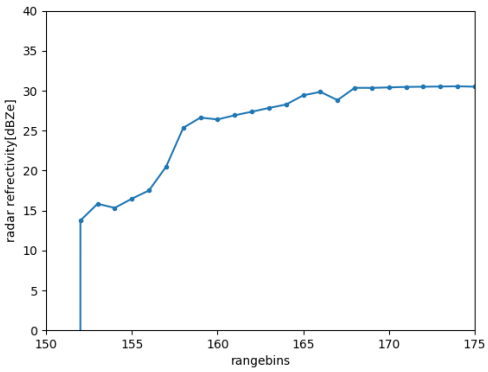
Each setting can be customized in detail. In this example, missing values are specified in gray.



Pyplot.contourf

Painted contour contourf is not suitable for magnifying large differences in values between adjacent pixels, such as radar reflection factors. For some larger scales and rainfall distributions such as GSMaP, it provides a reasonably smooth and pleasing image.

A unique feature of DPR is that it provides vertical distributions of physical quantities related to precipitation. The program readDPRL2_2.py reads the equivalent radar reflection factor SLV/zFactorCorrected of the three-dimensional physical quantity and displays a profile for a given footprint.



readDPRL2_2.py output example

The following is the source code for readDPRL2_1.py.

```
  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 scans
 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=rrr.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()
```

Declaration to use mumpy (array arithmetic module) under the

This is a declaration to use the pyplot function of matplotlib (drawing

HDF5 file name.

HDF5 files are read.

FS/Latitude, FS/Longitude are taken from the read data.

FS/SLV/precipRateESurface is extracted from the read data.

FS/SLV/zFactorCorrectedESurface is extracted from the read data.

A portion of the retrieved data is output to the screen

The retrieved precipRateESurface is plotted.

Drawing.

The following is the source code for readDPRL2_2.py.

```
 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 scans
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 file name.

HDF5 files are read.

FS/SLV/precipRateESurface is extracted from the read data.

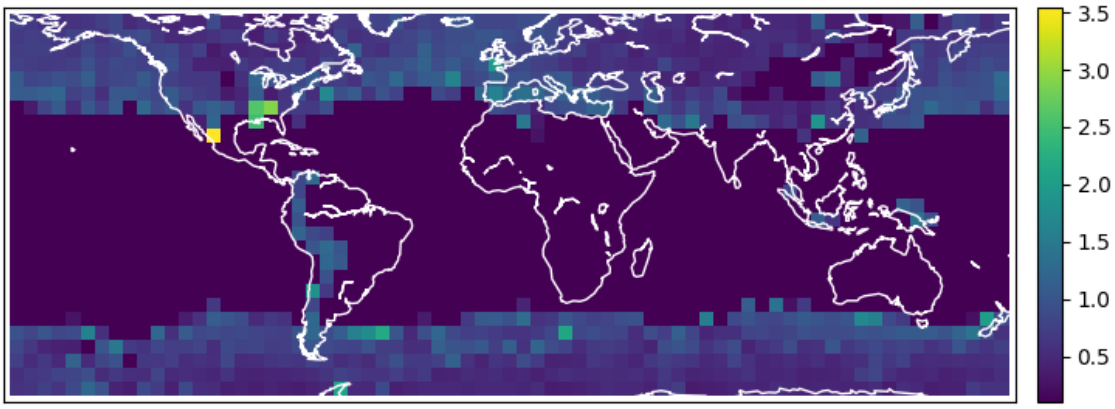
FS/SLV/precipRateESurface is extracted from the read data.

The retrieved precipRateESurface is plotted.

Drawing.

② DPR L3 data

The DPR Level 3 data will be the statistics of the physical quantities provided in L2; L3 stores data on 0.25-degree (G2) and 5-degree (G1) grids. Run readDPRL3.py in the same way as before in the folder where the data are stored. You will get the output as shown below.



readDPRL3.py output example

The following is the source code for readDPRL3.py.

```
 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()
```

Declares the use of Basemap to draw a map.

HDF5 file name.

HDF5 files are read.

Grids/G1/snowRateNearSurface/mean is taken from the read data.

The snowRateNSurf data is plotted in color on the map.

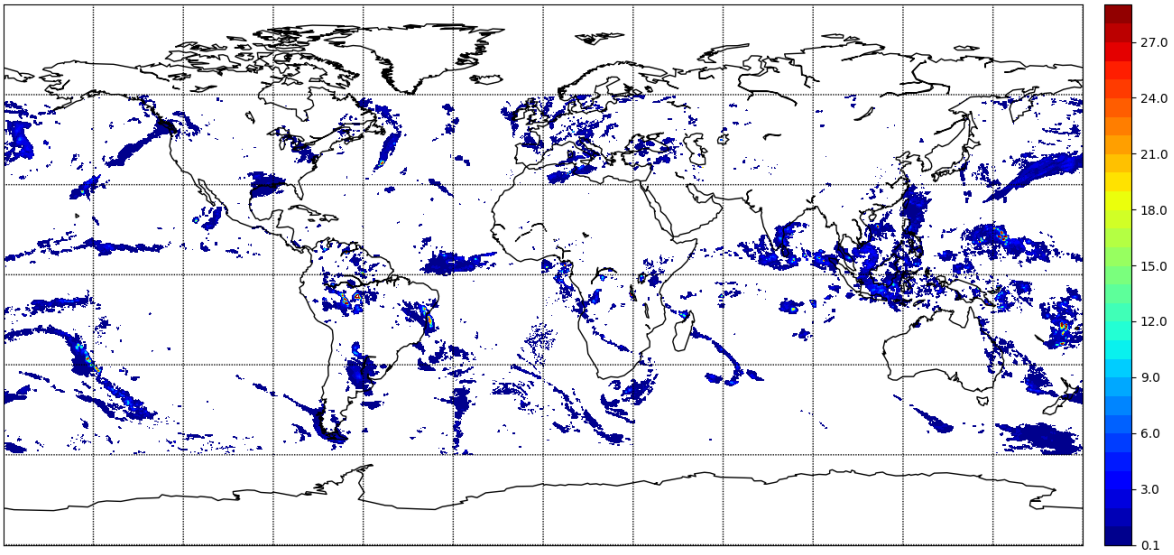
The projection method (cyl: equirectangular cylindrical method), resolution (c: abbreviation), and latitude and longitude of the edge are specified.

Lower left | Upper | Lower left | Longitude

Drawing.

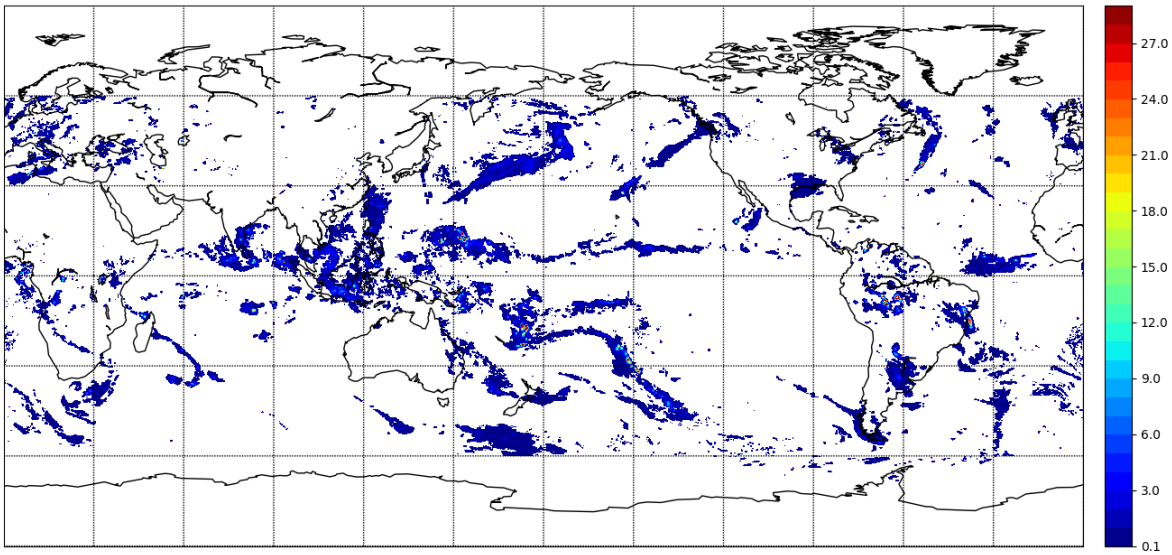
③ GSMaP data

GSMaP provides hourly global precipitation maps by combining data from multiple satellite-borne microwave radiometers and sounders, and is availablein a variety of formats (HDF5, Text, binary, netCDF, KML, geoTiff). Once the data file is opened, the same process can be applied to each format.



readGSMaP_netcdf.py Output example

* When using binary data, Basemap 1.0.5 or earlier has a horizontal line connecting the right edge and the left edge. This is because binary data contains data in 0° to -0°, while HDF and other data contain data in -180° to 180°. This can be solved by sorting the data according to the display range, or by using plot or pcolor instead of contourf (note the heavy processing).



readGSMaP_binary.py Output Example

The following is the source code for readGSMaP_netcdf.py.

```
 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.range(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.range(0,360,30))
43:m.drawparallels(np.range(-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 file name.

HDF5 files are read.

netCDF file is read.

The projection method (cyl: equirectangular cylindrical method), resolution (c: abbreviation), and latitude and longitude of the edge are specified.

Coastal, longitude, and latitude lines are drawn.

Drawing.

The following is the source code for readGSMaP_binary.py.

```
 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.range(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.range(0,360,30))
43:m.drawparallels(np.range(-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()
```

Rain gauge correction data is specified.

Rain gauge correction data is read in.

The read rain gauge correction data is Converted to 1800*3600 array.

The projection method (cyl: equirectangular cylindrical method), resolution (c: abbreviation), and latitude and longitude of the edge are specified.

Coastal, longitude, and latitude lines are drawn.

Drawing.

revision history

version number	Date	Revised contents	remarks
1	2018/1/2/24		
2	Mar 15, 2018	3. how to obtain related documents and sample programs: Table 3.1 sample program list added.	
3	2/7/2019		
4	Dec. 1, 2021	GSMaP product version 5 compatible. 6. elementary practice: source code modification readGSMaP_netcdf.py, readGSMaP_binary.py	
5	12/6/2021	1. modified to GPM/TRMM product version 7. 3. revised availability of related documentation and sample programs	
6	12/24/2021	Table 3.1 Sample data updated to V7 6 Corrected code description to match V7	