```
import os
import glob

import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
from pyresample.geometry import GridDefinition, SwathDefinition
from pyresample.kd_tree import resample_nearest
import numpy as np

FILE_NAME = 'MOD06_L2.A2023159.0705.061.2023163162431.hdf'
DATAFIELD_NAME = 'cloud_top_height_1km'

from pyhdf.SD import SD, SDC

i = 0
for file in list(glob.glob('MOD03.A2023156*.hdf')):
    reader = open(file)
    hdf = SD(file, SDC.READ)
    # Read geolocation dataset.
    lat = hdf.select('Latitude')
    latitude = lat[:,:]
    lon = hdf.select('Longitude')
    longitude = lon[:,:]
    if i == 0 :
        latitude_m = latitude
        longitude_m = longitude
    else:
        latitude_m = np.vstack([latitude_m, latitude])
        longitude_m = np.vstack([longitude_m, longitude])
    i = i + 1

i = 0
for file in list(glob.glob('MOD06_L2.A2023156*.hdf')):
    reader = open(file)
    hdf = SD(file, SDC.READ)
    # Read dataset.
    data2D = hdf.select(DATAFIELD_NAME)
    data = data2D[:,:].astype(np.double)
    # Retrieve attributes.
    attrs = data2D.attributes(full=1)
    aoa=attrs["add_offset"]
    add_offset = aoa[0]
    fva=attrs["_FillValue"]
    _FillValue = fva[0]
    sfa=attrs["scale_factor"]
    scale_factor = sfa[0]
```

```
    ua=attrs["units"]
    units = ua[0]
    data[data == _FillValue] = np.nan
    data = (data - add_offset) * scale_factor
    datam = np.ma.masked_array(data, np.isnan(data))
    if i == 0 :
        data_m = datam
    else:
        data_m = np.vstack([data_m, datam])
    i = i + 1
#
**********************************Regridding**************************
***************
# Define SwathDefinition.
    swathDef = SwathDefinition(lons=longitude_m, lats=latitude_m)

# Define GridDefinition.
# 0.1 degree is about 10.11km, which is close enough to native
resolution.
    cellSize = 0.1
    min_lon = -180#np.min(longitude_m)
    max_lon = 180#np.max(longitude_m)
    min_lat = -90#np.min(latitude_m)
    max_lat = 90#np.max(latitude_m)
    x0, xinc, y0, yinc = (min_lon, cellSize, max_lat, -cellSize)
    nx = int(np.floor((max_lon - min_lon) / cellSize))
    ny = int(np.floor((max_lat - min_lat) / cellSize))
    x = np.linspace(x0, x0 + xinc*nx, nx)
    y = np.linspace(y0, y0 + yinc*ny, ny)
    lon_g, lat_g = np.meshgrid(x, y)
    grid_def = GridDefinition(lons=lon_g, lats=lat_g)
    # Set radius_of_influence in meters.
    ri = 10000
    result = resample_nearest(swathDef, data_m, grid_def,
                              radius_of_influence=ri, epsilon=0.5,
                              fill_value=np.nan)
    [cols, rows] = result.shape



---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[5], line 77
     75 # Set radius_of_influence in meters.
     76 ri = 10000
---> 77 result = resample_nearest(swathDef, data_m, grid_def,
     78                           radius_of_influence=ri, epsilon=0.5,
     79                           fill_value=np.nan)
```

```
     80 [cols, rows] = result.shape

File ~\anaconda3\envs\Geopandas\Lib\site-packages\pyresample\
kd_tree.py:105, in resample_nearest(source_geo_def, data,
target_geo_def, radius_of_influence, epsilon, fill_value, reduce_data,
nprocs, segments)
     62 def resample_nearest(source_geo_def,
     63                       data,
     64                       target_geo_def,
   (...)
     69                       nprocs=1,
     70                       segments=None):
     71     """Resamples data using kd-tree nearest neighbour
approach.
     72
     73     Parameters
   (...)
    103         Source data resampled to target geometry
    104     """
--> 105     return _resample(source_geo_def, data, target_geo_def,
'nn',
    106                      radius_of_influence, neighbours=1,
    107                      epsilon=epsilon, fill_value=fill_value,
    108                      reduce_data=reduce_data, nprocs=nprocs,
segments=segments)

File ~\anaconda3\envs\Geopandas\Lib\site-packages\pyresample\
kd_tree.py:268, in _resample(source_geo_def, data, target_geo_def,
resample_type, radius_of_influence, neighbours, epsilon, weight_funcs,
fill_value, reduce_data, nprocs, segments, with_uncert)
    257 """Resamples swath using kd-tree approach."""
    258 valid_input_index, valid_output_index, index_array,
distance_array = \
    259     get_neighbour_info(source_geo_def,
    260                        target_geo_def,
   (...)
    265                        nprocs=nprocs,
    266                        segments=segments)
--> 268 return get_sample_from_neighbour_info(resample_type,
    269                                        target_geo_def.shape,
    270                                        data, valid_input_index,
    271                                        valid_output_index,
    272                                        index_array,
    273
distance_array=distance_array,
    274
weight_funcs=weight_funcs,
    275                                        fill_value=fill_value,
    276                                        with_uncert=with_uncert)
```

```
File ~\anaconda3\envs\Geopandas\Lib\site-packages\pyresample\
kd_tree.py:610, in get_sample_from_neighbour_info(resample_type,
output_shape, data, valid_input_index, valid_output_index,
index_array, distance_array, weight_funcs, fill_value, with_uncert)
    607     data = data.ravel()
    609 if valid_input_index.size != data.shape[0]:
--> 610     raise ValueError('Mismatch between geometry and dataset')
    612 is_multi_channel = (data.ndim > 1)
    613 valid_input_size = valid_input_index.sum()

ValueError: Mismatch between geometry and dataset
```

```
print(data_m.data.shape)
print(latitude_m.shape)
longitude_m.shape
```

```
(26420, 1354)
(26420, 1354)
```

```
(26420, 1354)
```