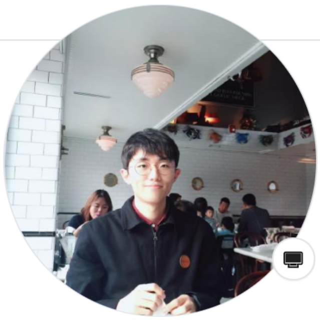
강사 소개



Kwan-Gu, Baek
Kwan-Gu

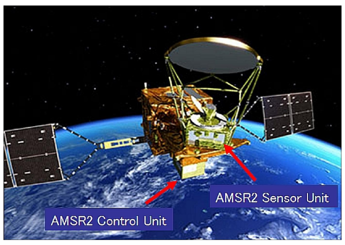
@python @Sql @SpatialInformation
@AtmosphericScience @datascience
@MachineLearning
@SeoulNationalUniversity

- 이름(메일): **백관구(bouguereau1130@gmail.com)**
- 소속: 서울대 지구환경과학부 대기과학전공 석박통합과정 4 년
- 분야: 머신러닝(딥러닝) | 공간정보학(고해상도 자료 생산 알고리즘)
- 언어: **파이썬** (5 년차)

위성관측: 해빙 (활용편 46쪽부터) - 시호연 저자님

1. GCOM-W 위성의 AMSR-2 센서 관측 자료 읽기 & 표출하기

자료 제공 기관	• JAXA (일본 우주항공연구개발기구)
위성	• GCOM-W
센서	• AMSR-2
자료	• Level 3 daily gridded 편광밝기온도, 해빙점유율
격자 형식	• Polar stereographic grid, 25 km 해상도



1) 25 km polar stereographic grid 위경도 binary 파일 읽기

2) HDF5 형식의 6.9 GHz 편광밝기온도, 해빙점유율 파일 읽기

3) 지도 위에 plot 하는 함수 작성과 자료 표출

1) 25 km polar stereographic grid 위경도 binary 파일 읽기

	AMSR-E/AMSR2 Unified L3 Daily 25 km Brightness Temperatures & Sea Ice Concentration Polar Grids
Description	Grids that determine the latitude of a given pixel for the 25 km grids for either hemisphere (psn for the Northern Hemisphere and pss for the Southern Hemisphere). These latitude grids are in binary format and are stored as 4-byte integers (little endian) scaled by 100,000 (divide the stored value by 100,000 to get decimal degrees). Each array location (i, j) contains the latitude value at the center of the corresponding data grid cells. psn25lats_v3.dat: 304 columns x 448 rows, range = [31.102, 89.836]

격자의 위/경도 정보를 읽어오는 함수 정의

In [1]:

```python
import numpy as np

print(np.__version__)
```

```
1.19.1
```

In [2]:

```python
def ReadCoordinate(filename):
    value = np.fromfile(filename, dtype = np.int32) # 자료 읽어 오기
    value = value.reshape(448, 304) # 배열 형태 변경
    value = value * 1e-5 # 스케일링 복구

    return value
```

In [3]:

```python
fileLon = "./data/psn25lons_v3.dat"
fileLat = "./data/psn25lats_v3.dat"

lons = ReadCoordinate(fileLon)
lats = ReadCoordinate(fileLat)

print(lons.shape, np.min(lons), np.max(lons))
print(lats.shape, np.min(lats), np.max(lats))
```

```
(448, 304) -180.00000000000003 179.81398000000002
(448, 304) 31.102670000000003 89.83682
```

2-1) HDF5 형식의 6.9 GHz 편광밝기온도 자료 읽기

밝기온도	• 수직편광 밝기온도(TBV) • 수평편광 밝기온도(TBH)

HDF5 파일에 포함된 모든 키 확인

In [4]:

```
import h5py

print(h5py.__version__)
```

```
2.10.0
```

In [5]:

```
hdf = h5py.File("./data/AMSR-2/2019/01/06_25km/GW1AM2_20190101_01D_PNMA_L3SGT06LA2220220.h5",
'r')

keys = []
hdf.visit(keys.append)

print(keys)
```

```
['Brightness Temperature (H)', 'Brightness Temperature (V)', 'Time Information']
```

키의 속성 확인

In [6]:

```
for attr, val in hdf["Brightness Temperature (H)"].attrs.items():
    print(attr, val)
```

```
SCALE FACTOR [0.01]
UNIT [b'K']
```

편광밝기온도를 읽어오는 함수 정의

In [7]:

```
def readTB(filename):
    fTB = h5py.File(filename, 'r')

    TBH = np.array(fTB["Brightness Temperature (H)"]) * 0.01 # 수평편광 밝기온도를 불러오고 스케
일링
    TBV = np.array(fTB["Brightness Temperature (V)"]) * 0.01 # 수직편광 밝기온도를 불러오고 스케
일링

    TBH[TBH == 655.34] = np.nan # 결측치 처리
    TBV[np.where(TBV == 655.34)] = np.nan

    return(TBH, TBV)
```

In [37]:

```
a = [[2, 3, 4], # [0, 2], [1, 1]
    [3, 4, 5]]
a = np.array(a)

print(np.shape(a))
```

```
(2, 3)
```

In [42]:

```
b = np.where(a == 4)

b[0] # 첫 번째 인덱스 (행)

b[1]
```

Out[42]:

```
array([2, 1], dtype=int64)
```

In [43]:

```
a = np.array([1, 2, 3])

np.where(a == 3)
```

Out[43]:

```
(array([2], dtype=int64),)
```

In [8]:

```
fileTB = "./data/AMSR-2/2019/01/06_25km/GW1AM2_20190101_01D_PNMA_L3SGT06LA2220220.h5"

TBH, TBV = readTB(fileTB)

print(TBH.shape, np.nanmin(TBH), np.nanmax(TBH))
print(TBV.shape, np.nanmin(TBV), np.nanmax(TBV))
```

```
(448, 304) 75.59 325.37
(448, 304) 148.25 337.55
```

2-2) HDF5 형식의 해빙점유율 자료 읽기

해빙점유율	• 하나의 격자에서 얼마만큼의 면적이 해빙으로 덮여 있는지 (단위: %)

해빙점유율을 읽어오는 함수 정의

In [9]:

```python
def readSIC(filename):
    fSIC = h5py.File(filename, "r")
    SIC = np.array(fSIC["Geophysical Data"]) * 0.1 # shape: (448, 304, 1)

    SIC = SIC[:, :, 0]  # 시간축 제거 (y, x, t) -> (y, x)
    SIC[np.where(SIC < 0)] = np.nan

    return(SIC)
```

In [10]:

```python
fileSIC = "./data/AMSR-2/2019/01/SIC/GW1AM2_20190101_01D_PNMA_L3SGSICLC3300300.h5"
SIC = readSIC(fileSIC)

print(SIC.shape, np.nanmin(SIC), np.nanmax(SIC))
```

```
(448, 304) 0.0 100.0
```

3-1) Basemap을 활용한 자료 표출

지도 그리는 함수 정의

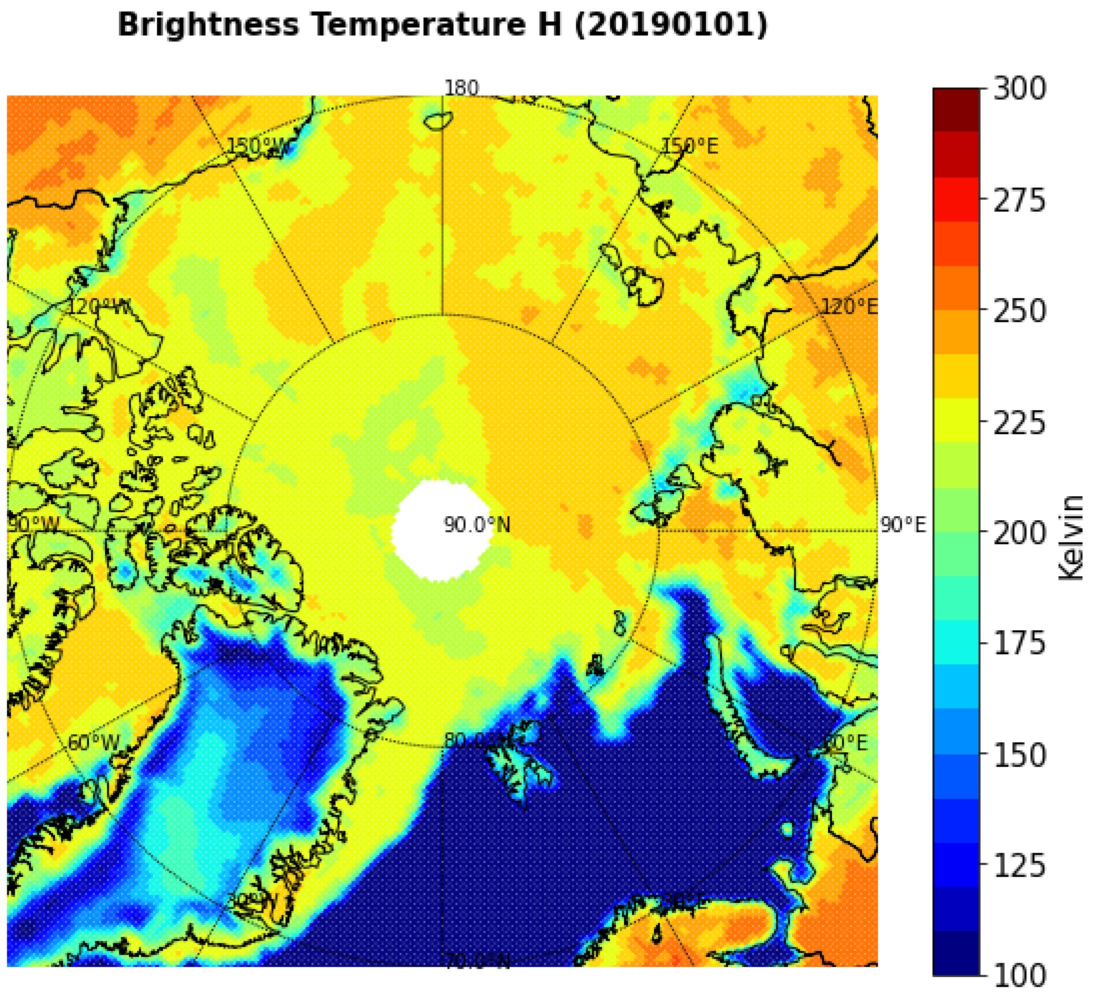
In [11]:

```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap


def PolarStereoMap(data0, lons, lats, min, max, levels, title, cbartitle):
    data = data0.copy()

    data[np.where(data > max)] = max
    data[np.where(data < min)] = min

    boundinglat = 70
    lon_0 = 0

    m = Basemap(projection = "npstere", resolution = "l", boundinglat = boundinglat, lon_0 = lon
_0)
    x, y = m(lons, lats) # lon, lat을 map 상의 x, y 좌표로 변환

    fig = plt.figure(figsize = (8, 7))
    m.drawcoastlines() # 해안선 그리기
    m.drawmeridians(np.arange(0, 360, 30), latmax = 80) # 경도선 그리기
    m.drawparallels([60, 70, 80], thick = 100) # 위도선 그리기
    m.drawmapboundary(color = "None")

    cs = m.scatter(x, y, c = data, s = 4, cmap = plt.cm.get_cmap("jet", levels), vmin = min, vma
x = max)

    # 지도에 위도 라벨 표시
    latlabels = []
    lons_latlabel = [0] * 3 # [0, 0, 0]
    lats_latlabel = np.arange(3) * 10. + boundinglat

    for templat in lats_latlabel:
        latlabels.append(str(templat) + "° N")
    xx, yy, = m(lons_latlabel, lats_latlabel)
    for label, xpt, ypt in zip(latlabels, xx, yy):
        plt.text(xpt, ypt, label)

    # 지도에 경도 라벨 표시
    lonlabels = []
    lons_lonlabel = np.array([30, 60, 90, 120, 150, 180, -150, -120, -90, -60, -30])
    lats_lonlabel = [boundinglat] * len(lons_lonlabel) # [boundinglat, boundinglat, boundinglat,
...]

    for templon in lons_lonlabel:
        if templon > 0 and templon < 180:
            lonlabels.append(str(templon) + "° E")
        if templon == 180:
            lonlabels.append(str(templon))
        if templon < 0:
            lonlabels.append(str(- templon) + "° W")

    xx, yy, = m(lons_lonlabel, lats_lonlabel)

    for label, xpt, ypt in zip(lonlabels, xx, yy):
        plt.text(xpt, ypt, label)

    plt.title(title, pad = 30, fontsize = 15, weight = "bold") # 그림 제목

    cbar = plt.colorbar(cs)
    cbar.ax.set_ylabel(cbartitle, fontsize = 15) # 컬러바 y 라벨
    cbar.ax.tick_params(labelsize = 15)
    cbar.set_clim(min, max) # 컬러바의 최대, 최소

    plt.tight_layout()

    plt.show()
```

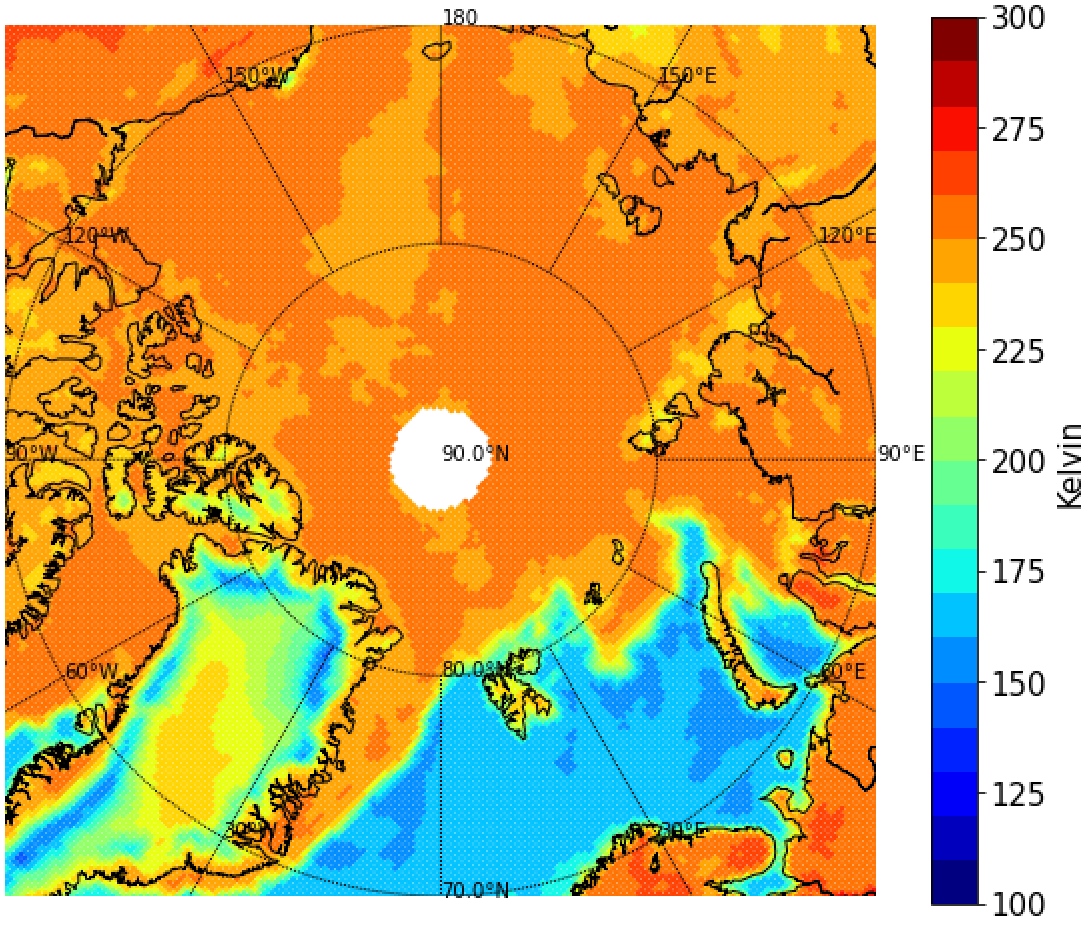
In [12]:

```
PolarStereoMap(TBH, lons, lats, 100, 300, 20, "Brightness Temperature H (20190101)", "Kelvin")
PolarStereoMap(TBV, lons, lats, 100, 300, 20, "Brightness Temperature V (20190101)", "Kelvin")
PolarStereoMap(SIC, lons, lats, 0, 100, 20, "Sea Ice Concentration (20190101)", "%")
```

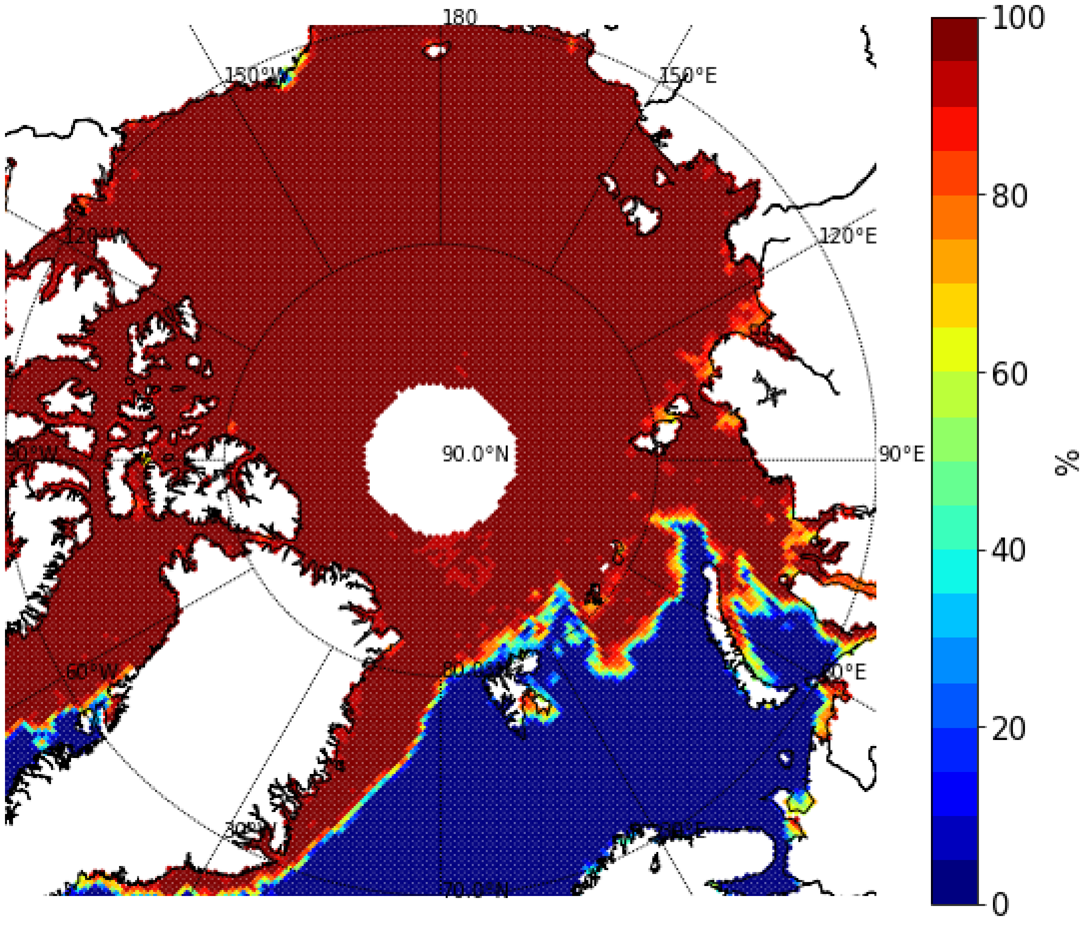
```
C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:14:
MatplotlibDeprecationWarning:
The dedent function was deprecated in Matplotlib 3.1 and will be removed in 3.3. U
se inspect.cleandoc instead.

C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:59:
MatplotlibDeprecationWarning:
The set_clim function was deprecated in Matplotlib 3.1 and will be removed in 3.3.
Use ScalarMappable.set_clim instead.
```



Brightness Temperature V (20190101)



Sea Ice Concentration (20190101)



3-2) 해빙점유율 자료를 사용하여 SIE 시계열 그리기

SIE | • Sea Ice Extent: 해빙점유율이 15% 이상인 격자들의 면적

파일을 하나씩 읽은 뒤 SIC>15% 이상인 격자 수를 세어 저장

In [13]:

```
import glob


file_list_SIC = glob.glob("./data/SIC/*")

print(file_list_SIC)
```

```
['./data/SIC₩₩GW1AM2_20130101_01D_PNMA_L3SGSICLC3300300.h5', './data/SIC₩₩GW1AM2_2
0140101_01D_PNMA_L3SGSICLC3300300.h5', './data/SIC₩₩GW1AM2_20150101_01D_PNMA_L3SGS
ICLC3300300.h5', './data/SIC₩₩GW1AM2_20160101_01D_PNMA_L3SGSICLC3300300.h5', './da
ta/SIC₩₩GW1AM2_20170101_01D_PNMA_L3SGSICLC3300300.h5', './data/SIC₩₩GW1AM2_2018010
1_01D_PNMA_L3SGSICLC3300300.h5', './data/SIC₩₩GW1AM2_20190101_01D_PNMA_L3SGSICLC33
00300.h5']
```

In [14]:

```
SIE_series = []

for i in range(len(file_list_SIC)):
    SIC = readSIC(file_list_SIC[i])
    spotSICgt15 = np.where(SIC >= 15)[0]
    SIE = len(spotSICgt15)
    SIE_series.append(SIE)
```

In [29]:

```
SIE_series
```

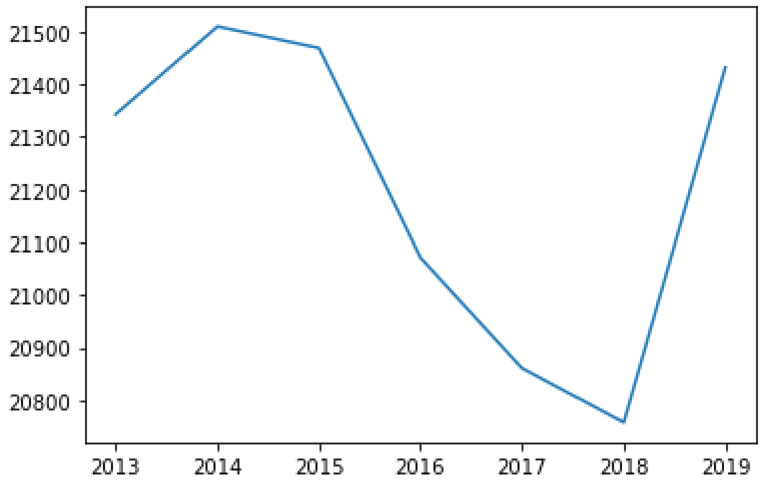
Out[29]:

```
[21343, 21510, 21469, 21071, 20861, 20758, 21432]
```

시계열 그리기

In [15]:

```
years = np.arange(2013, 2019 + 1)

plt.plot(years, SIE_series)

plt.show()
```



2. 위성관측 밝기온도로부터 해빙방출률 산출하기

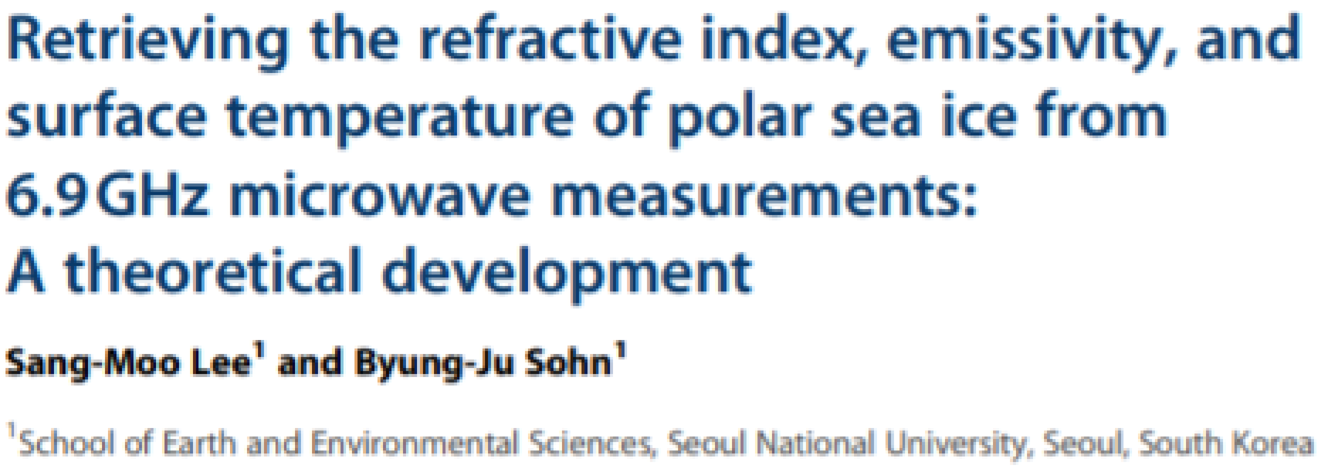
1) 이론 설명

2) 수평/수직 편광밝기온도를 사용한 해빙방출률 계산 함수 작성

3) 작성한 함수를 위성관측자료에 적용

4) 결과 표출

1) 이론 설명



Retrieving the refractive index, emissivity, and surface temperature of polar sea ice from 6.9 GHz microwave measurements: A theoretical development

Sang-Moo Lee[1] and Byung-Ju Sohn[1]

[1]School of Earth and Environmental Sciences, Seoul National University, Seoul, South Korea

요약!!!

$$\frac{\mathrm{EH}}{\mathrm{EV}} = \frac{1-\mathrm{RH}}{1-\mathrm{RV}} \qquad \text{\# 키르히호프 법칙}$$

$$= \frac{1-\mathrm{RH}}{1-\mathrm{Fresnel(RH)}} \qquad \text{\# RV=Fresnel(RH)}$$

$$= \underset{\text{관측되는 값}}{\boxed{\frac{\mathrm{TBH}}{\mathrm{TBV}}}} \qquad \text{\#EV와 EH의 비가 TBV와 TBH의 비와 같다}$$

2) 수평/수직 편광밝기온도를 사용한 해빙방출률 계산 함수 작성

프레넬 방정식 함수 정의

In [16]:

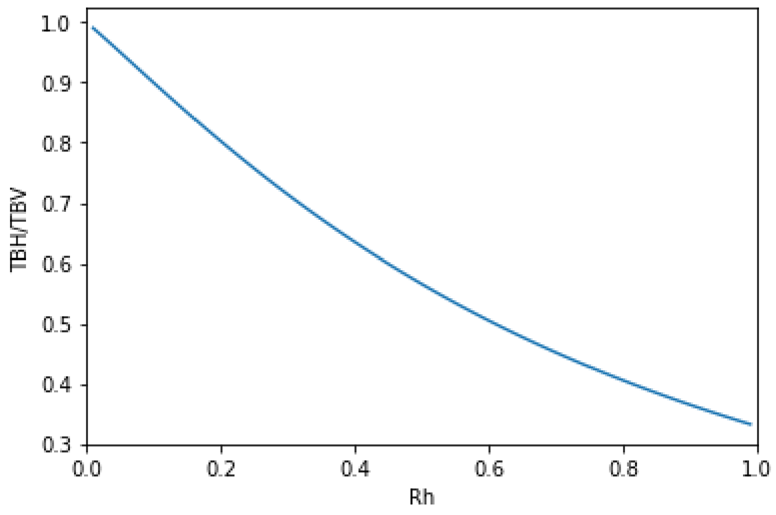
```
def CombFresEq(Rh, theta): # Combined Fresnel equation(수평편광반사도, 위성관측 각도)
    cos = np.cos(2. * np.deg2rad(theta))
    Rv = (Rh ** 2.) * ( (1. + cos / np.sqrt(Rh)) / (1. + cos * np.sqrt(Rh))) ** 2.

    return(Rv)
```

수평편광반사도(RH)에 따른 이론적 수평-수직밝기온도(TBH/TBV)

In [17]:

```
Rh = np.arange(0., 1. + 0.01, 0.01)
ratio = (1. - Rh) / (1. - CombFresEq(Rh, 55.)) # 이론적인 TBH/TBV 비율

plt.plot(Rh, ratio)

plt.xlim(0, 1)
plt.xlabel("Rh")
plt.ylabel("TBH/TBV")

plt.show()
```

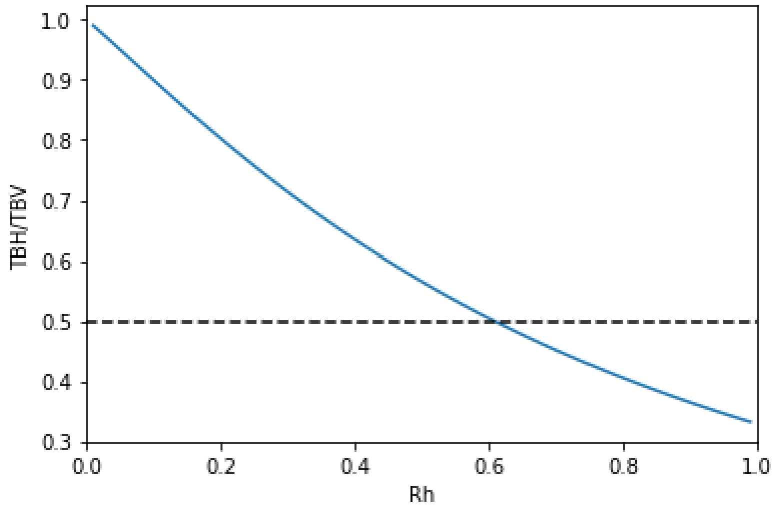
```
C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:3: R
untimeWarning: divide by zero encountered in true_divide
  This is separate from the ipykernel package so we can avoid doing imports until
C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:3: R
untimeWarning: invalid value encountered in multiply
  This is separate from the ipykernel package so we can avoid doing imports until
C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:2: R
untimeWarning: invalid value encountered in true_divide
```



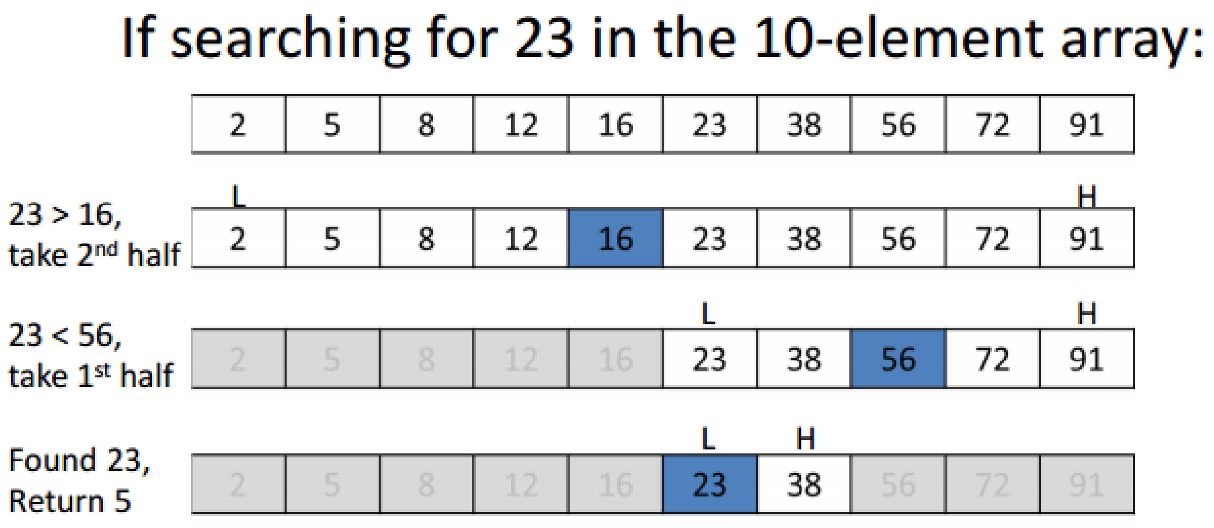
가상의 관측된 밝기온도 비(0.5) 표시

In [18]:

```
plt.plot(Rh, ratio)

ratio_sample = 0.5
plt.plot([0,1], [ratio_sample] * 2, "k--")

plt.xlim(0, 1)
plt.xlabel("Rh")
plt.ylabel("TBH/TBV")

plt.show()
```



단조 감소 함수이므로 binary search 알고리즘을 활용해 Rh 값을 찾을 수 있다!



Binary search 알고리즘 함수 정의

In [19]:

```
def solver(ratio, theta):
    Rh_now = 0.5
    Rh_0 = 0.0001
    Rh_1 = 0.9999

    while True:
        ratio_now = (1. - Rh_now) / (1. - CombFresEq(Rh_now, theta))

        if abs(ratio_now - ratio) < 0.0001:
                return(Rh_now)
                break
        else:
            if ratio_now - ratio > 0.:
                Rh_0 = Rh_now
                Rh_1 = Rh_1
                Rh_now = (Rh_now + Rh_1) / 2.
            else:
                Rh_0 = Rh_0
                Rh_1 = Rh_now
                Rh_now = (Rh_now + Rh_0) / 2.
```

In [30]:

```
ratio_sample
```

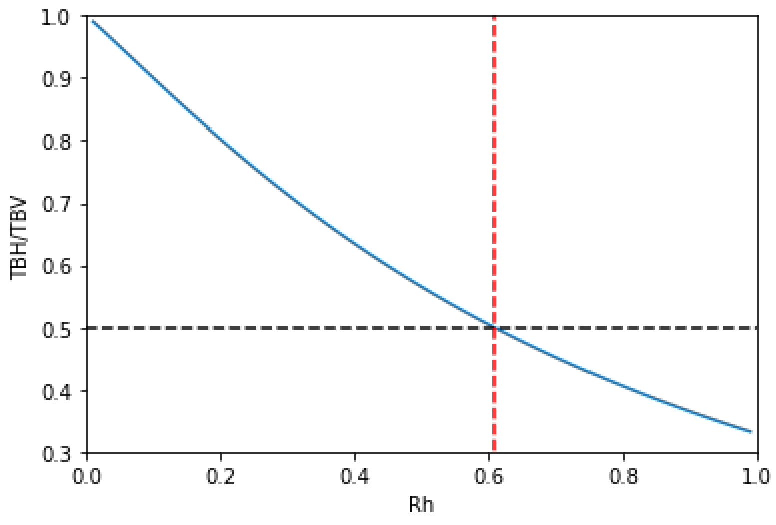
Out[30]:

```
0.5
```

In [20]:

```
Rh_solution = solver(ratio_sample, 55)
print(Rh_solution)

plt.plot(Rh, ratio)
plt.plot([0,1], [ratio_sample]*2, 'k--')
plt.plot([Rh_solution]*2, [0,1], 'r--')

plt.xlim(0, 1)
plt.ylim(0.3, 1)
plt.xlabel('Rh')
plt.ylabel('TBH/TBV')

plt.show()
```

```
0.6088649414062501
```

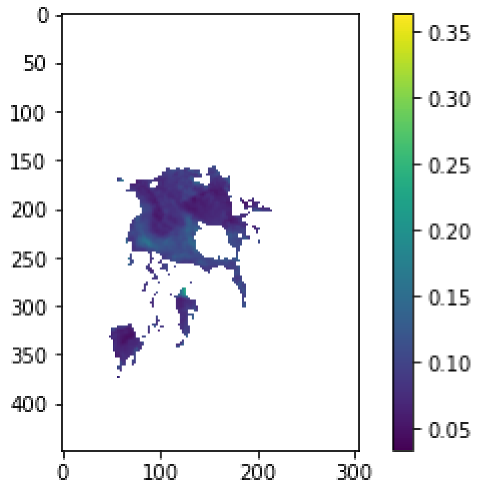


3) 작성한 함수를 위성관측자료에 적용

수평-수직 밝기온도 비가 유한하고, 해빙점유율이 95% 이상인 격자에 대해서만 계산을 수행

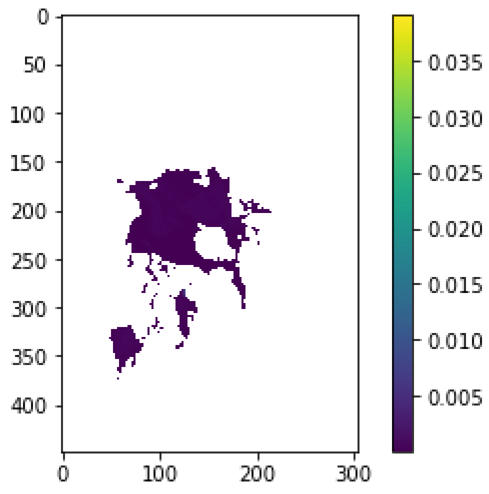
In [21]:

```
ratios = TBH / TBV
RH = np.full((448,304), np.nan)

for i in range(448):
    for j in range(304):
        ratio = ratios[i,j]

        if (np.isfinite(ratio)) & (SIC[i,j] >= 95):
            RH[i,j] = solver(ratio, 55)

plt.imshow(RH)
plt.colorbar()

plt.show()
```



In [22]:

```
RV = CombFresEq(RH, 55.)

plt.imshow(RV)
plt.colorbar()

plt.show()
```

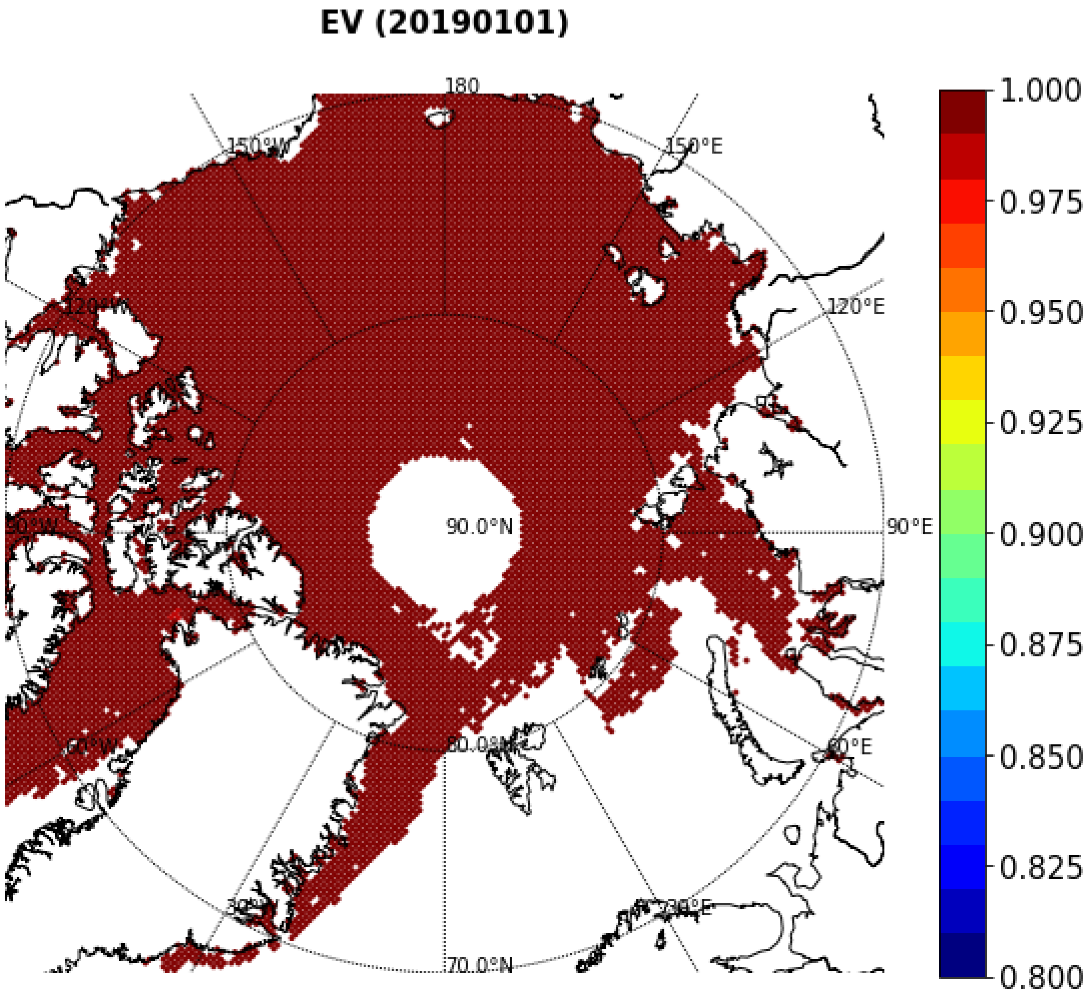


4) 결과 표출

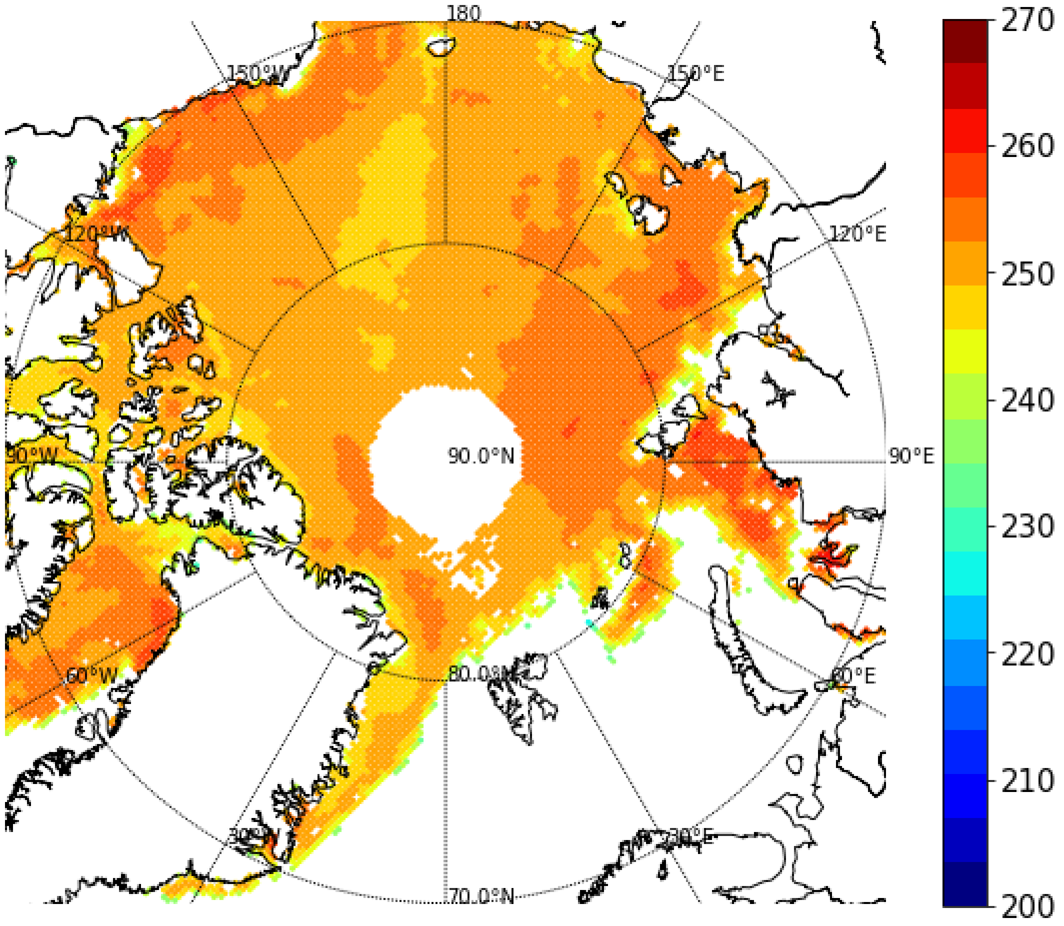
In [23]:

```
EV = 1. - RV
EH = 1. - RH

Ts_H = TBH / EH  #해빙 표면온도
Ts_V = TBV / EV

# def PolarStereoMap(data0, lons, lats, min, max, levels, title, cbartitle)

PolarStereoMap(EV, lons, lats, 0.8, 1, 20, 'EV (20190101)', '')
PolarStereoMap(Ts_V, lons, lats, 200, 270, 20, 'TsV (20190101)', '')
PolarStereoMap(EH, lons, lats, 0.8, 1, 20, 'H-pol emissivity (20190101)', '')
PolarStereoMap(Ts_H, lons, lats, 200, 270, 20, 'TsH (20190101)', '')
```

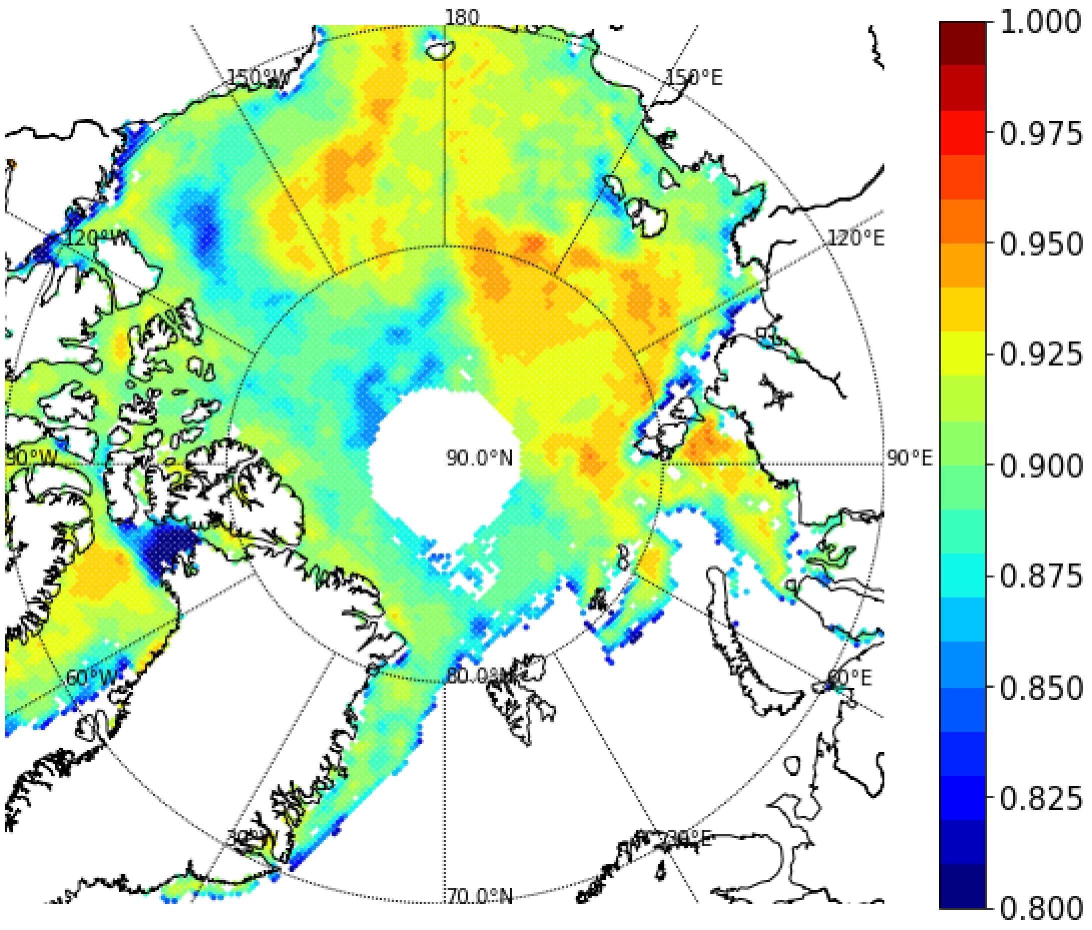
```
C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:14:
MatplotlibDeprecationWarning:
The dedent function was deprecated in Matplotlib 3.1 and will be removed in 3.3. U
se inspect.cleandoc instead.

C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:59:
MatplotlibDeprecationWarning:
The set_clim function was deprecated in Matplotlib 3.1 and will be removed in 3.3.
Use ScalarMappable.set_clim instead.
```



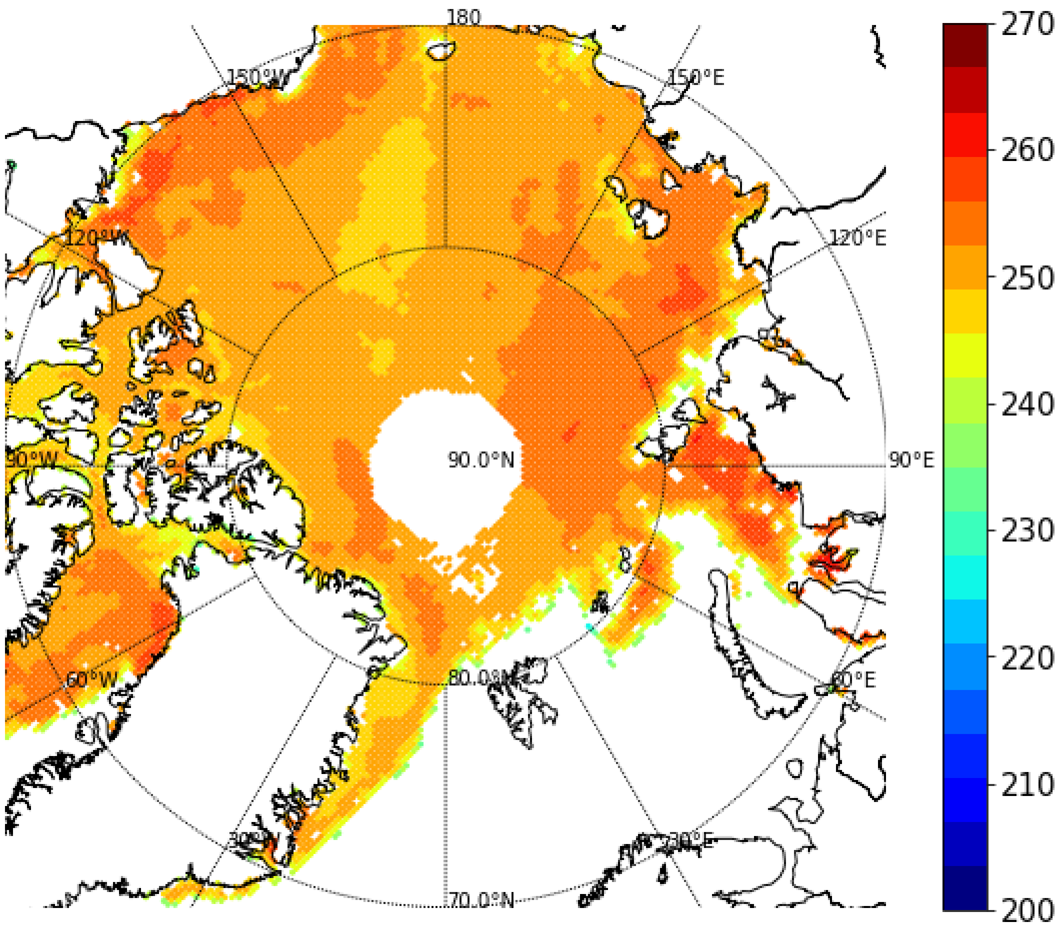
TsV (20190101)



H-pol emissivity (20190101)



TsH (20190101)



3. ICESat-2 위성 트랙자료를 격자자료로 변환하기

자료 제공 기관	• NSIDC
위성	• ICESat-2
센서	• lidar
자료	• Total freeboard: 지표에서 반사된 레이저 신호가 돌아오는 시간
격자 형식	• 트랙 자료

1) ICESat-2 freeboard 자료 읽고 표출하기

2) 25 km polar stereographic grid 에서 track 자료의 위경도와 가장 가까운 위치 찾기

3) track 자료를 grid 자료로 변환

4) 결과 표출

1) ICESat-2 freeboard 자료 읽고 표출하기

Freeboard 자료 읽는 함수 정의

In [24]:

```python
def readIS2(fileS2):
    f = h5py.File(fileS2, 'r')

    gt = f['gt1l/freeboard_beam_segment']

    free = np.array(gt['beam_fb_height'])
    free[np.where(free > 100)] = np.nan

    lon = np.array(gt['longitude'])
    lat = np.array(gt['latitude'])

    return(lon, lat, free)
```

In [25]:

```
filename_IS2 ='./data/ICESat-2/ATL10-01_20190101005132_00550201_002_01.h5'
lons_sat, lats_sat, free_sat = readIS2(filename_IS2)

print(lons_sat.shape, lats_sat.shape, free_sat.shape)
```

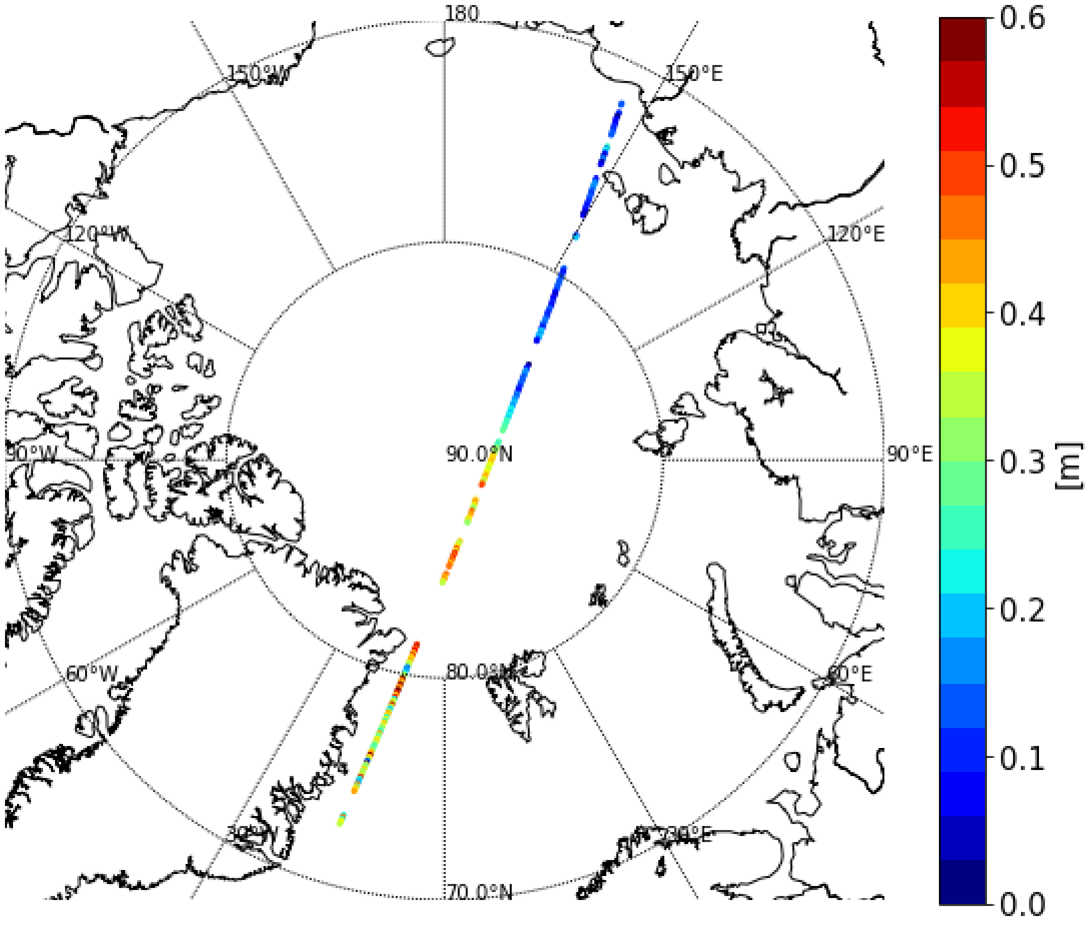
```
(390,) (390,) (390,)
```

In [26]:

```
fileLon = './data/psn25lons_v3.dat'
fileLat = './data/psn25lats_v3.dat'

lons_grid = ReadCoordinate(fileLon)
lats_grid = ReadCoordinate(fileLat) # shape: (448, 304)

# def PolarStereoMap(data0, lons, lats, min, max, levels, title, cbartitle)

PolarStereoMap(free_sat, lons_sat, lats_sat, 0, 0.6, 20, 'Total Freeboard Thickness (Track)',
'[m]')
```

```
C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:14:
MatplotlibDeprecationWarning:
The dedent function was deprecated in Matplotlib 3.1 and will be removed in 3.3. U
se inspect.cleandoc instead.

C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:59:
MatplotlibDeprecationWarning:
The set_clim function was deprecated in Matplotlib 3.1 and will be removed in 3.3.
Use ScalarMappable.set_clim instead.
```

Total Freeboard Thickness (Track)



2) 25 km polar stereographic grid 에서 track 자료의 위경도와 가장 가까운 위치 찾기

Haversine 공식: 북극 지역에서는 구면 상 두 점의 거리를 계산하기 위해 구면 삼각법으로 유도! (https://kayuse88.github.io/haversine/ (https://kayuse88.github.io/haversine/))

In [31]:

```
np.sum(np.isnan(free_sat))
```

Out[31]:

```
117
```

In [32]:

```
np.isnan(free_sat)
```

Out[32]:

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False, False,  True,  True,  True, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False,  True,  True,  True,  True,  True, False,
       False, False, False, False, False, False, False, False, False,
       False,  True,  True,  True,  True,  True,  True, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True,  True,  True,  True,  True,
        True, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False,  True,
        True,  True,  True,  True, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True,  True,  True,  True,  True,  True,  True,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True,  True,  True, False, False, False, False,
       False, False,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False, False, False, False, False, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True])
```

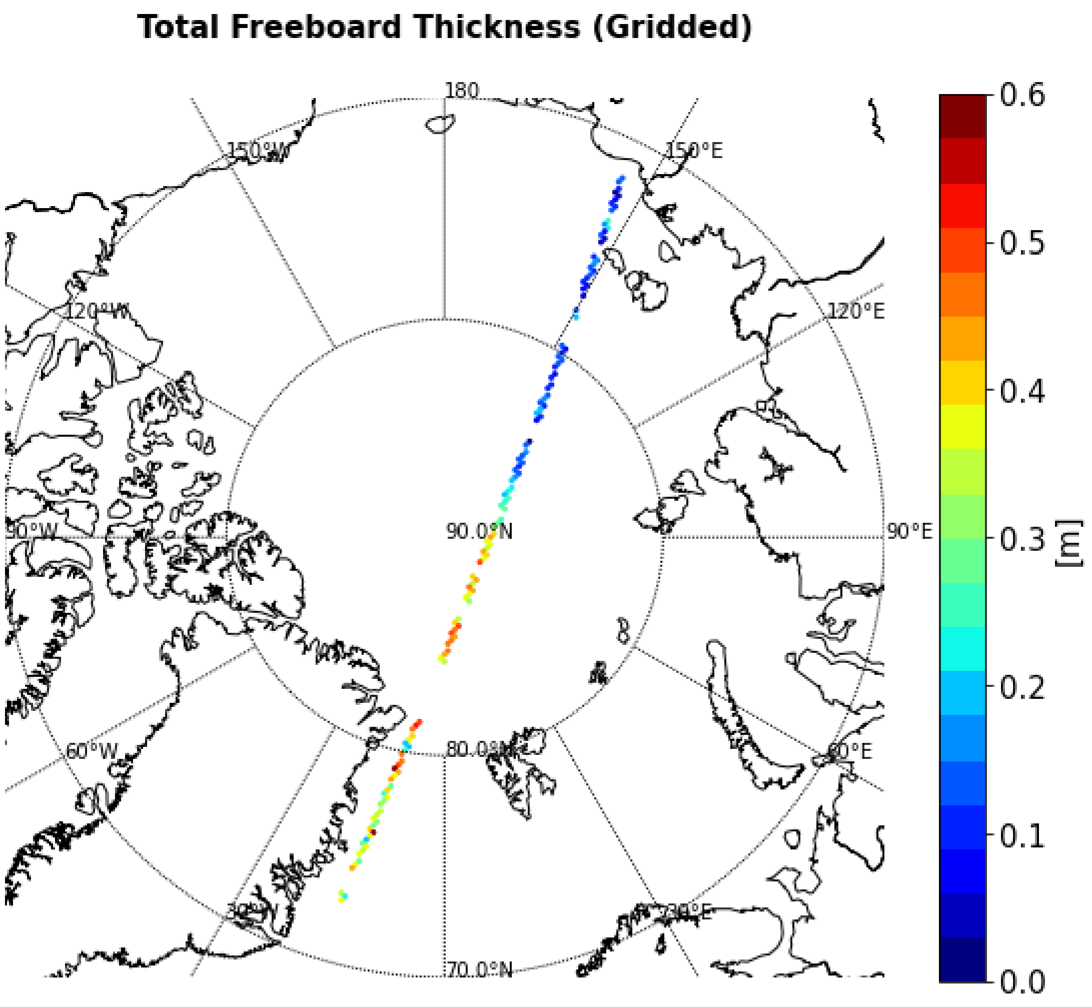
위성관측 위치에 가장 가까운 격자를 찾아 그 격자에 관측치를 할당한 후,

각 격자에 할당된 관측치를 평균

In [28]:

```
# 모든 각도의 단위를 "도"에서 "radian"으로 변환
lons_sat_rad = np.radians(lons_sat)
lats_sat_rad = np.radians(lats_sat)

lons_grid_rad = np.radians(lons_grid)
lats_grid_rad = np.radians(lats_grid)

Re = 6371.228 #km

total = np.zeros((448,304)) # 자료 할당을 위한 grid
count = np.zeros((448,304)) # 할당된 자료의 수 count

for i in range(len(free_sat)):
    free0 = free_sat[i]

    if (np.isinf(free0)) | (np.isnan(free0)):
        continue # 다음 loop 실행

    lon0_rad = lons_sat_rad[i]
    lat0_rad = lats_sat_rad[i]

    havs_lat = np.sin((lats_grid_rad - lat0_rad) / 2.) ** 2.
    havs_lon = np.sin((lons_grid_rad - lon0_rad) / 2.) ** 2.
    dist_array = 2. * Re * np.arcsin(np.sqrt(havs_lat + np.cos(lats_grid_rad) * np.cos(lat0_rad)
* havs_lon))

    spot = np.where(dist_array == np.nanmin(dist_array))
    dist0 = dist_array[spot]

    if dist0 > np.sqrt(12.5 ** 2 + 12.5 ** 2):
        continue

    total[spot] = total[spot] + free0
    count[spot] = count[spot] + 1.


free_grid = total / count

# def PolarStereoMap(data0, lons, lats, min, max, levels, title, cbartitle)

PolarStereoMap(free_grid, lons_grid, lats_grid, 0, 0.6, 20, 'Total Freeboard Thickness (Gridde
d)', '[m]')
```

```
C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:36:
RuntimeWarning: invalid value encountered in true_divide
C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:14:
MatplotlibDeprecationWarning:
The dedent function was deprecated in Matplotlib 3.1 and will be removed in 3.3. U
se inspect.cleandoc instead.

C:₩Users₩bkk11₩Anaconda3₩envs₩lecture₩lib₩site-packages₩ipykernel_launcher.py:59:
MatplotlibDeprecationWarning:
The set_clim function was deprecated in Matplotlib 3.1 and will be removed in 3.3.
Use ScalarMappable.set_clim instead.
```


Total Freeboard Thickness (Gridded)
180
150°W
150°E
120°W
120°E
90°W
90.0°N
90°E
60°W
80.0°N
60°E
30°W
30°E
70.0°N
0.6
0.5
0.4
0.3
0.2
0.1
0.0
[m]