1.
1.1　　　SST　　　　NOAA_NCDC_ERSST_v3b_SST.nc　　　　　　120~170　　　-5~5
　　　group_data - group_data.mean(dim='time')　　　　　　　　　　sst_dif

1.2
ds_anom_resample_m　　　　　　　　　　　　　　　　　　0
0　　　　　　　　　　　　　　　　　　　　　　　　SST　　　3
　　　　　　　　　　0.5　　　　　　　　El Nino　　　　　　-0.5
　La Nina　　　　　　　0

2.
2.1　　　　TOA　　　　　CERES_EBAF-TOA_200003-201701.nc　　　　toa_lw_all_mon　toa_sw_
all_mon　solar_mon
total_flux　　　　　　　　　　　　　　　　　　　net_flux　　　toa_net_all_mon
　　　　　　　　　　　　　　　　　　　　　　net_flux
2.2　　　incoming_solar　outgoing_longwave　outgoing_shortwave

2.3　　　TOA　　　　　　　　　　　　　　　　　　　　　　　1.0
　　　　　　　　　　-90　　90　　　　　　1

　　　　　　　　　　　　　　NumPy　sum
　　　Matplotlib　　　　　　　　　　　　　　　　　　　　1

2.4　　　　　　　　　　　　　　　　　25%　75%

2.5

3.
3.1　　GES DISC　　　　　　2023　9

3.2　　　5　　　　　　　　　　　　　DMS

In [1]:
```python
# 1. Niño 3.4 index
# Load modules
import numpy as np
import pandas as pd
import netCDF4
import xarray as xr
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec

# Show plots in the notebook
%matplotlib inline
```

In [2]:
```python
#1.1
SST = xr.open_dataset("NOAA_NCDC_ERSST_v3b_SST.nc", engine="netcdf4")
SST
```

Out[2]:

xarray.Dataset

▶ Dimensions: (**lat**: 89, **lon**: 180, **time**: 684)

▼ Coordinates:

lat	(lat)	float32	-88.0 -86.0 -84.0 ... 86.0 88.0	📄	🛢
lon	(lon)	float32	0.0 2.0 4.0 ... 354.0 356.0 …	📄	🛢
time	(time)	datetime64[ns]	1960-01-15 ... 2016-12-15	📄	🛢

▼ Data variables:

sst	(time, lat, lon)	float32	...	📄	🛢

▶ Indexes: (3)

▼ Attributes:

Conventions :	IRIDL
source :	https://iridl.ldeo.columbia.edu/SOURCES/.NOAA/.NCDC/.ERSST/.version3b/.sst/
history :	extracted and cleaned by Ryan Abernathey for Research Computing in Earth Science

In [3]: `SST['sst']`

Out[3]:

xarray.DataArray **'sst'** (**time**: 684, **lat**: 89, **lon**: 180)

[10957680 values with dtype=float32]

▼ Coordinates:

lat	(lat)	float32	-88.0 -86.0 -84.0 ... 86.0 88.0	📄 🗄
lon	(lon)	float32	0.0 2.0 4.0 ... 354.0 356.0 358.0	📄 🗄
time	(time)	datetime64[ns]	1960-01-15 ... 2016-12-15	📄 🗄

▶ Indexes: (3)

▼ Attributes:

pointwidth :	1.0
valid_min :	-3.0
valid_max :	45.0
units :	degree_Celsius
long_name :	Extended reconstructed sea surface temperature
standard_name :	sea_surface_temperature
iridl:hasSemanti…	iridl:SeaSurfaceTemperature

In [4]:
```python
# Compute monthly climatology for SST from Niño 3.4 region
group_data = SST.sst.sel(lon=slice(120,170),lat=slice(-5,5)).groupby('time.month')
sst_dif = group_data-group_data.mean(dim='time')

# subtract climatology from SST time series to obtain anomalies
resample_obj = sst_dif.resample(time="3M").mean(dim="time")
resample_obj
```

Out[4]:

xarray.DataArray   'sst'   (**time**: 229, **lat**: 5, **lon**: 26)

```
array([[[-0.4533596 , -0.43008804, -0.3652172 , ..., -0.5904255 ,
         -0.51613617, -0.5157356 ],
        [-0.14541245, -0.14106178, -0.20046997, ..., -0.60107803,
         -0.5806999 , -0.5200424 ],
        [ 0.03437614, -0.01860619, -0.1291542 , ..., -0.61279106,
         -0.5868416 , -0.55138206],
        [-0.03416824, -0.07881355, -0.139431  , ..., -0.5768242 ,
         -0.56368065, -0.5451031 ],
        [-0.11306   , -0.14630127, -0.18651962, ..., -0.47527504,
         -0.48386002, -0.49680328]],

       [[-0.29540953, -0.25229773, -0.21316402, ..., -0.6501789 ,
         -0.5796814 , -0.58689374],
        [-0.18128014, -0.12417793, -0.13654137, ..., -0.6904233 ,
         -0.68461037, -0.64244586],
        [-0.09715843, -0.08390108, -0.10546494, ..., -0.7069289 ,
         -0.6881733 , -0.6722056 ],
        [-0.18694179, -0.16128285, -0.128987  , ..., -0.64433545,
         -0.62889546, -0.6225446 ],
        [-0.27703476, -0.2525959 , -0.20511119, ..., -0.517519  ,
```

...

```
         0.51037025,  0.44631258],
        [ 0.31214967,  0.4855779 ,  0.7164224 , ...,  0.4436461 ,
          0.3200194 ,  0.2053426 ],
        [ 0.39565277,  0.5145791 ,  0.7320716 , ...,  0.39797845,
          0.23362541,  0.08429018],
        [ 0.44386673,  0.44989267,  0.5983505 , ...,  0.5368557 ,
          0.3789749 ,  0.21928024],
        [ 0.42669234,  0.40143776,  0.4725081 , ...,  0.714798  ,
          0.5879669 ,  0.46769652]],

       [[ 0.32543087,  0.3451271 ,  0.4029932 , ...,  0.51263714,
          0.4383192 ,  0.36778736],
        [ 0.42484474,  0.5078449 ,  0.57851505, ...,  0.34471035,
          0.22703075,  0.10994244],
        [ 0.5032301 ,  0.5828867 ,  0.66394806, ...,  0.27353382,
          0.13096333, -0.00620747],
        [ 0.46020794,  0.49208736,  0.58321095, ...,  0.37838078,
          0.25306892,  0.11438084],
        [ 0.3544016 ,  0.36249638,  0.44186687, ...,  0.5236778 ,
          0.4169016 ,  0.31012917]]], dtype=float32)
```

▼ Coordinates:

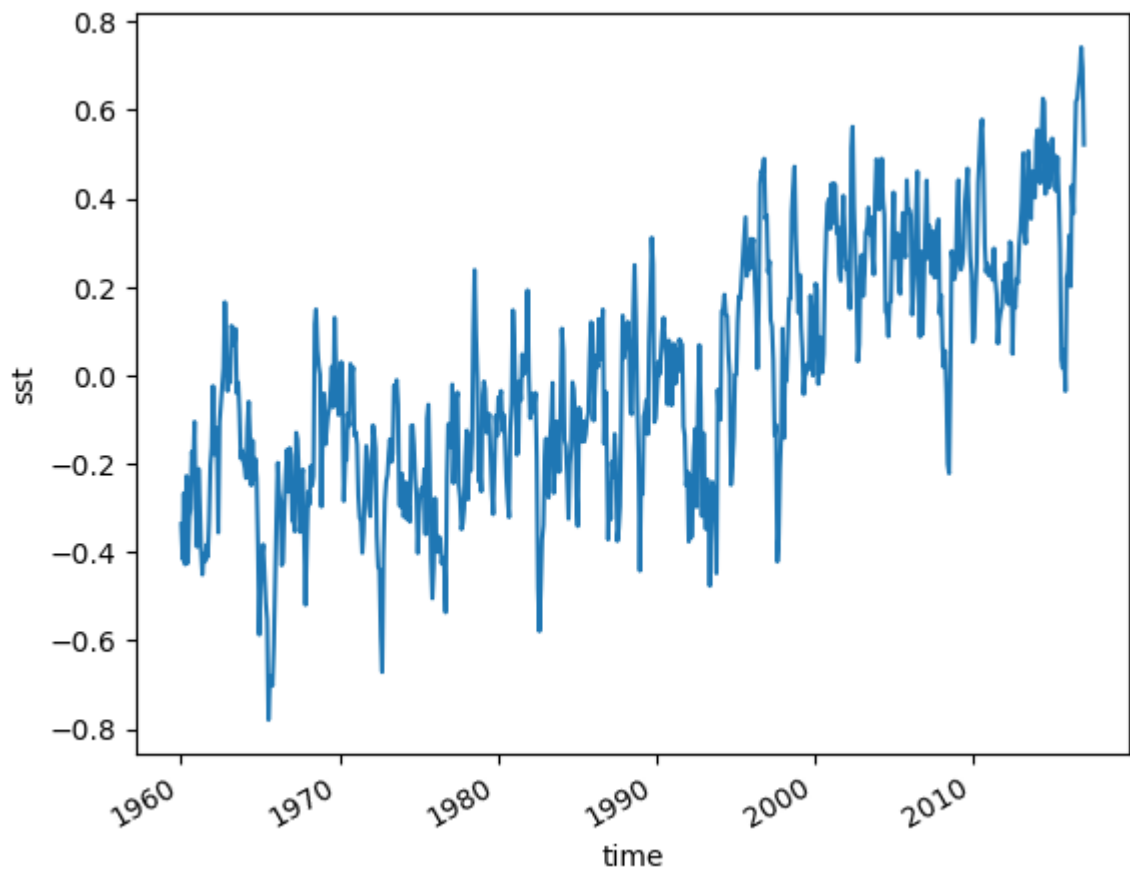|				
lat	(lat)	float32	-4.0 -2.0 0.0 2.0 4.0	📄 🗄
lon	(lon)	float32	120.0 122.0 124.0 ... 168.0 170.0	📄 🗄
time	(time)	datetime64[ns]	1960-01-31 ... 2017-01-31	📄 🗄

► Indexes: (3)

► Attributes: (0)

In [5]:
```
# 1.2
sst_dif.mean(dim=['lat','lon']).plot()
```

Out[5]: [<matplotlib.lines.Line2D at 0x238097dc190>]

In [6]: `resample_obj.mean(dim=['lat','lon']).plot()`

Out[6]: [<matplotlib.lines.Line2D at 0x23809833fd0>]

```
In [7]: ds_anom_resample_m=resample_obj.mean(dim=['lat','lon'])
        ds_anom_resample_m
```
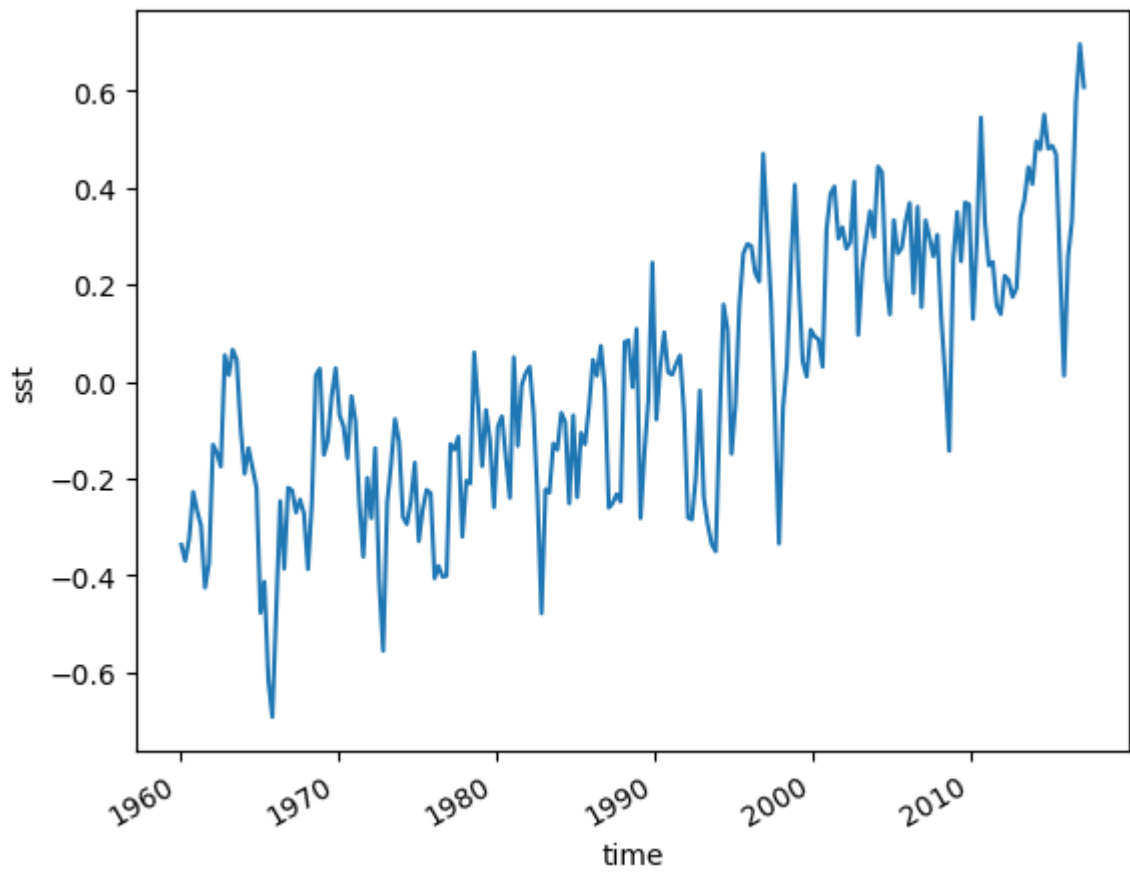
```
In [7]: ds_anom_resample_m=resample_obj.mean(dim=['lat','lon'])
        ds_anom_resample_m
```

Out[7]:

xarray.DataArray   'sst'   (**time**: 229)

array([-0.33638978, -0.37003502, -0.3239999 , -0.22765496, -0.26742154,
       -0.29706082, -0.42516705, -0.3743719 , -0.13013509, -0.14662217,
       -0.17526981, 0.05451763, 0.01313595, 0.06591826, 0.04469279,
       -0.09537051, -0.19031097, -0.13719894, -0.1779783 , -0.21954633,
       -0.47708625, -0.41306534, -0.61572886, -0.6919386 , -0.46526185,
       -0.24677637, -0.38609752, -0.21915027, -0.22480527, -0.2702479 ,
       -0.2433737 , -0.27216244, -0.38718328, -0.24891156, 0.01236948,
        0.02674307, -0.1512471 , -0.12314105, -0.02989539, 0.02746935,
       -0.06977199, -0.09244606, -0.15824564, -0.03030802, -0.08215355,
       -0.24998821, -0.36153477, -0.19859119, -0.2819085 , -0.13750277,
       -0.41236964, -0.5556232 , -0.2498695 , -0.1719188 , -0.07733187,
       -0.12359207, -0.27969757, -0.29421753, -0.2476332 , -0.16733189,
       -0.32939062, -0.26420408, -0.22348669, -0.23065028, -0.4060981 ,
       -0.38037926, -0.402714 , -0.4010484 , -0.12891535, -0.1408334 ,
       -0.11381914, -0.32007325, -0.2045178 , -0.21054327, 0.05988208,
       -0.05048161, -0.17434482, -0.05864822, -0.12214249, -0.25969198,
       -0.09298059, -0.07176815, -0.1562431 , -0.24005908, 0.05026476,
       -0.13279352, -0.00744956, 0.01719128, 0.03076849, -0.06558541,
       -0.2355391 , -0.47826445, -0.22260715, -0.22947134, -0.12784567,
       -0.14085631, -0.06479608, -0.08387943, -0.2515224 , -0.06982005,
       ...
       -0.19591552, -0.01864021, -0.23917682, -0.29464397, -0.33528358,
       -0.35045642, -0.07411855, 0.15897055, 0.10697694, -0.14921309,
       -0.04572915, 0.1568192 , 0.2641609 , 0.28375474, 0.2797301 ,
        0.22447583, 0.20625184, 0.4701414 , 0.31612307, 0.16429943,
       -0.07111615, -0.33452275, -0.05399809, 0.03474595, 0.23891602,
        0.40586895, 0.20059861, 0.04168706, 0.01015314, 0.10702178,
        0.09294897, 0.0868701 , 0.03048421, 0.31326863, 0.38727093,
        0.40259364, 0.29476655, 0.3183891 , 0.27422825, 0.28749415,
        0.4125064 , 0.09602283, 0.23268497, 0.294751 , 0.35175908,
        0.2979943 , 0.4437765 , 0.4319937 , 0.21467721, 0.13865069,
        0.33312672, 0.26419055, 0.2758488 , 0.33165234, 0.36844757,
        0.18227148, 0.3608378 , 0.15332824, 0.33264446, 0.2980027 ,
        0.2584041 , 0.30271897, 0.12783696, 0.01041856, -0.14296326,
        0.25190043, 0.3500043 , 0.2480731 , 0.36936525, 0.36543158,
        0.12898877, 0.29491633, 0.54474586, 0.3310301 , 0.23948544,
        0.24682468, 0.15716833, 0.13909237, 0.21837936, 0.209491 ,
        0.17454773, 0.19300571, 0.34000415, 0.3749026 , 0.442632 ,
        0.40747023, 0.4960373 , 0.4790274 , 0.5508579 , 0.48036876,
        0.48651356, 0.469067 , 0.24312 , 0.01210874, 0.25750467,
        0.33146283, 0.57795835, 0.6961747 , 0.6076585], dtype=float32)

▼ Coordinates:

| **time** | (time) | datetime64[ns] | 1960-01-31 ... 2017-01-31 |

► Indexes: (1)

► Attributes: (0)

In [8]:
```python
# draw the plot
df=pd.DataFrame(ds_anom_resample_m.where(ds_anom_resample_m>=0),columns=['anom>=0'])
df['anom<0']=pd.DataFrame(ds_anom_resample_m.where(ds_anom_resample_m<0))
df['date'] = pd.DataFrame(ds_anom_resample_m.time)
df.set_index('date',inplace=True)

plt.figure(figsize=(10,6),dpi=120)
plt.bar(np.arange(len(df['anom>=0'])),df['anom>=0'],color="red")
plt.bar(np.arange(len(df['anom<0'])),df['anom<0'],color="blue")
plt.plot(ds_anom_resample_m,'k-')

plt.axhline(y=0.5,color="red",linestyle='--',linewidth=0.5)
plt.axhline(y=-0.5,color="blue",linestyle='--',linewidth=0.5)
plt.axhline(y=0,color="black",linestyle='-',linewidth=1)
plt.legend(labels=['3mth mean','El Nino Threshold','La Nino Threshold'],loc=4)
plt.ylabel('anom mean',fontsize=14)
plt.xlabel('time',fontsize=14)
plt.title('SST Anomaly in Nino 3.4 Region',fontsize=16)
```
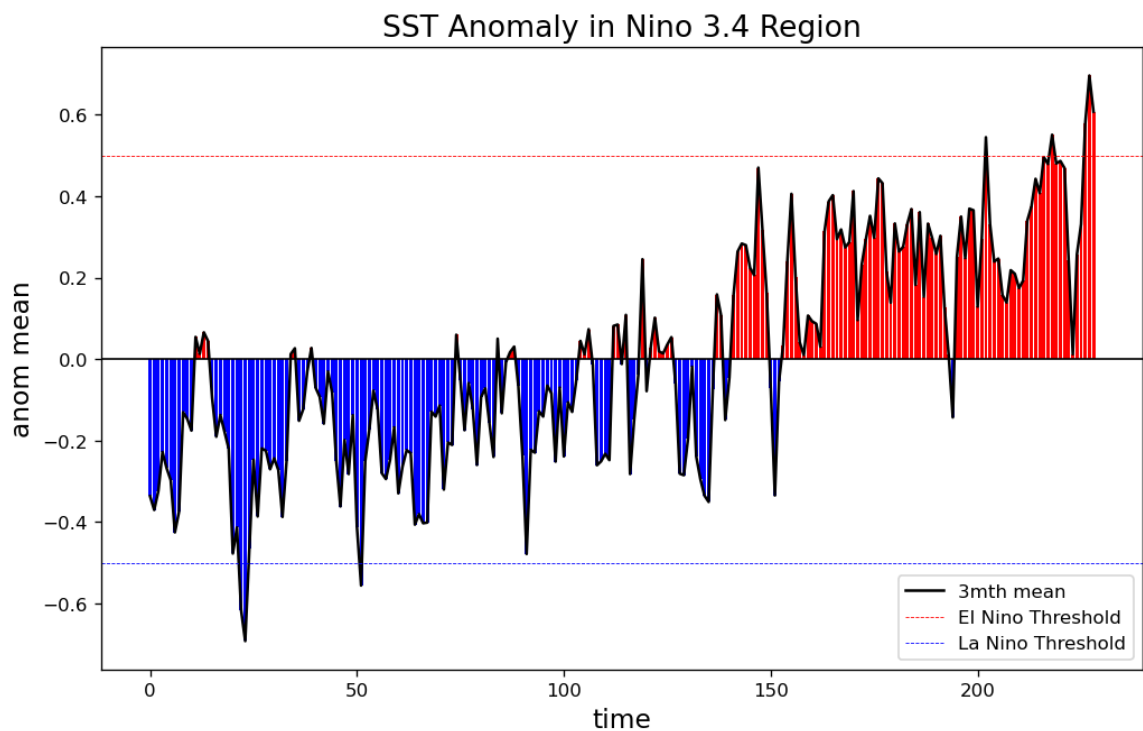
Out[8]: Text(0.5, 1.0, 'SST Anomaly in Nino 3.4 Region')

In [9]:
```python
#2 Earth's energy budget
#2.1
TOA = xr.open_dataset("CERES_EBAF-TOA_200003-201701.nc", engine="netcdf4")
TOA
```

Out[9]:

xarray.Dataset

▶ Dimensions: (**lon**: 360, **time**: 203, **lat**: 180)

▼ Coordinates:

lon	(lon)	float32	0.5 1.5 2.5 ... 357.5 358.5 …
time	(time)	datetime64[ns]	2000-03-15 ... 2017-01-15
lat	(lat)	float32	-89.5 -88.5 -87.5 ... 88.5 89.5

▼ Data variables:

toa_sw_all_mon	(time, lat, lon)	float32	...
toa_lw_all_mon	(time, lat, lon)	float32	...
toa_net_all_mon	(time, lat, lon)	float32	...
toa_sw_clr_mon	(time, lat, lon)	float32	...
toa_lw_clr_mon	(time, lat, lon)	float32	...
toa_net_clr_mon	(time, lat, lon)	float32	...
toa_cre_sw_mon	(time, lat, lon)	float32	...
toa_cre_lw_mon	(time, lat, lon)	float32	...
toa_cre_net_mon	(time, lat, lon)	float32	...
solar_mon	(time, lat, lon)	float32	...
cldarea_total_d…	(time, lat, lon)	float32	...
cldpress_total_…	(time, lat, lon)	float32	...
cldtemp_total_d…	(time, lat, lon)	float32	...
cldtau_total_da…	(time, lat, lon)	float32	...

▶ Indexes: (3)

▼ Attributes:

title :	CERES EBAF (Energy Balanced and Filled) TOA Fluxes. Monthly Averages and 07/2005 to 06/2015 Climatology.
institution :	NASA/LaRC (Langley Research Center) Hampton, Va
Conventions :	CF-1.4
comment :	Data is from East to West and South to North.
Version :	Edition 4.0; Release Date March 7, 2017
Fill_Value :	Fill Value is -999.0
DOI :	10.5067/TERRA+AQUA/CERES/EBAF-TOA_L3B.004.0
Production_Files :	List of files used in creating the present Master netCDF file: /homedir/nloeb/ebaf/monthly_means/adj_fluxes/deliverable/sw*.gz /homedir/nloeb/ebaf/monthly_means/adj_fluxes/deliverable/lw*.gz /homedir/nloeb/ebaf/monthly_means/adj_fluxes/deliverable/net*.gz /homedir/nloeb/ebaf/monthly_means/adj_fluxes/deliverable/solflx*.gz /homedir/nloeb/ebaf/monthly_means/out_glob.dat

In [10]:
```python
# Calculate the time-mean of the variables
mean_longwave = TOA['toa_lw_all_mon'].mean(dim='time')
mean_shortwave = TOA['toa_sw_all_mon'].mean(dim='time')
mean_solar = TOA['solar_mon'].mean(dim='time')

Create a 2D plot for the time-mean TOA longwave, shortwave, and solar radiation
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))
mean_longwave.plot(ax=ax1)
ax1.set_title('TOA Longwave Radiation')

mean_shortwave.plot(ax=ax2)
ax2.set_title('TOA Shortwave Radiation')

mean_solar.plot(ax=ax3)
ax3.set_title('Solar Radiation')

plt.tight_layout()
plt.show()
```
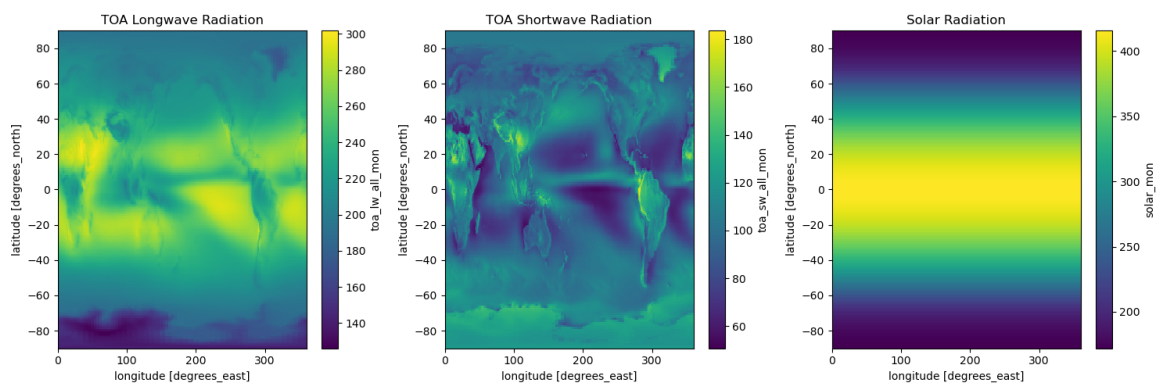
In [11]:
```python
# Add up the three variables and verify visually that they are equivalent to the TOA
total_flux = mean_solar - mean_longwave - mean_shortwave
net_flux = TOA['toa_net_all_mon'].mean(dim='time')

fig, (ax4, ax5) = plt.subplots(1, 2, figsize=(12, 5))
total_flux.plot(ax=ax4)
ax4.set_title('Total Flux')

net_flux.plot(ax=ax5)
ax5.set_title('Net Flux')

plt.tight_layout()
plt.show()

# We can see that these two axes have the totally same shape, which means that the th
```
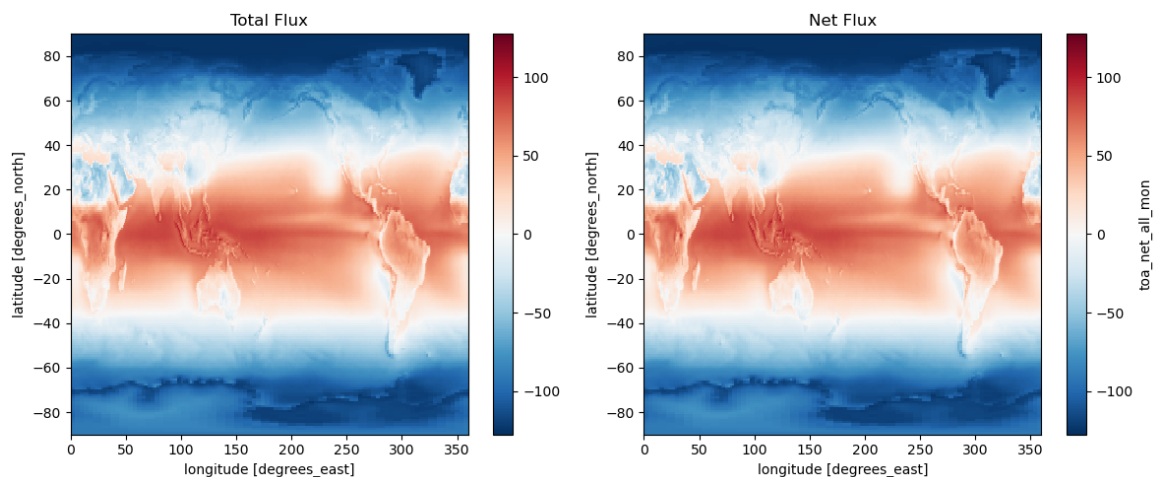
In [12]:
```python
# 2.2 Calculate and verify TOA incoming solar, outgoing longwave, and outgoing shortw
incoming_solar = TOA['solar_mon']+TOA['toa_net_all_mon']
outgoing_longwave = TOA['toa_lw_all_mon']+TOA['toa_cre_lw_mon']
outgoing_shortwave = TOA['toa_sw_all_mon']+TOA['toa_cre_sw_mon']

incoming_solar_mean = incoming_solar.mean(dim='time')
outgoing_longwave_mean = outgoing_longwave.mean(dim='time')
outgoing_shortwave_mean = outgoing_shortwave.mean(dim='time')

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))
incoming_solar_mean.plot(ax=ax1)
ax1.set_title('TOA incoming solar mean')

outgoing_longwave_mean.plot(ax=ax2)
ax2.set_title('TOA outgoing longwave mean')

outgoing_shortwave_mean.plot(ax=ax3)
ax3.set_title('TOA outgoing shortwave mean')

plt.tight_layout()
plt.show()
```
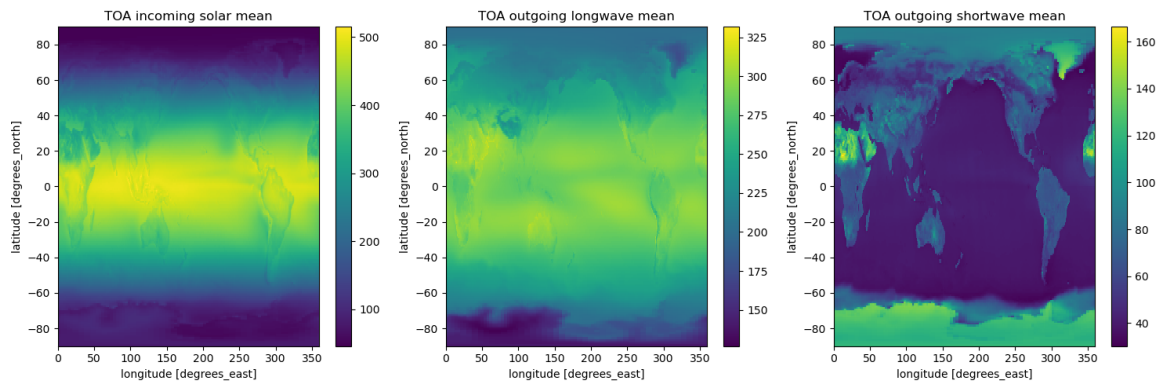
In [13]:
```python
# 2.3

latitude = TOA.variables['lat'][:]
net_radiation = TOA.variables['toa_net_all_mon'][:]

# Calculate total net radiation in each 1-degree latitude band
lat_resolution = 1.0
lat_bands = np.arange(-90, 91, lat_resolution)
net_radiation_total = np.zeros(len(lat_bands) - 1)

for i in range(len(lat_bands) - 1):
    lat_min, lat_max = lat_bands[i], lat_bands[i + 1]
    lat_mask = (latitude >= lat_min) & (latitude < lat_max)
    net_radiation_total[i] = np.sum(net_radiation[:, lat_mask])

# Plot the results
plt.figure(figsize=(10, 6), dpi=120)
plt.bar(lat_bands[:-1], net_radiation_total, width=lat_resolution, edgecolor='black')
plt.title('Total Net Radiation in 1-degree Latitude Bands')
plt.xlabel('Latitude (degrees)')
plt.ylabel('Total Net Radiation (W/m^2)')
plt.grid(True)
plt.show()
```
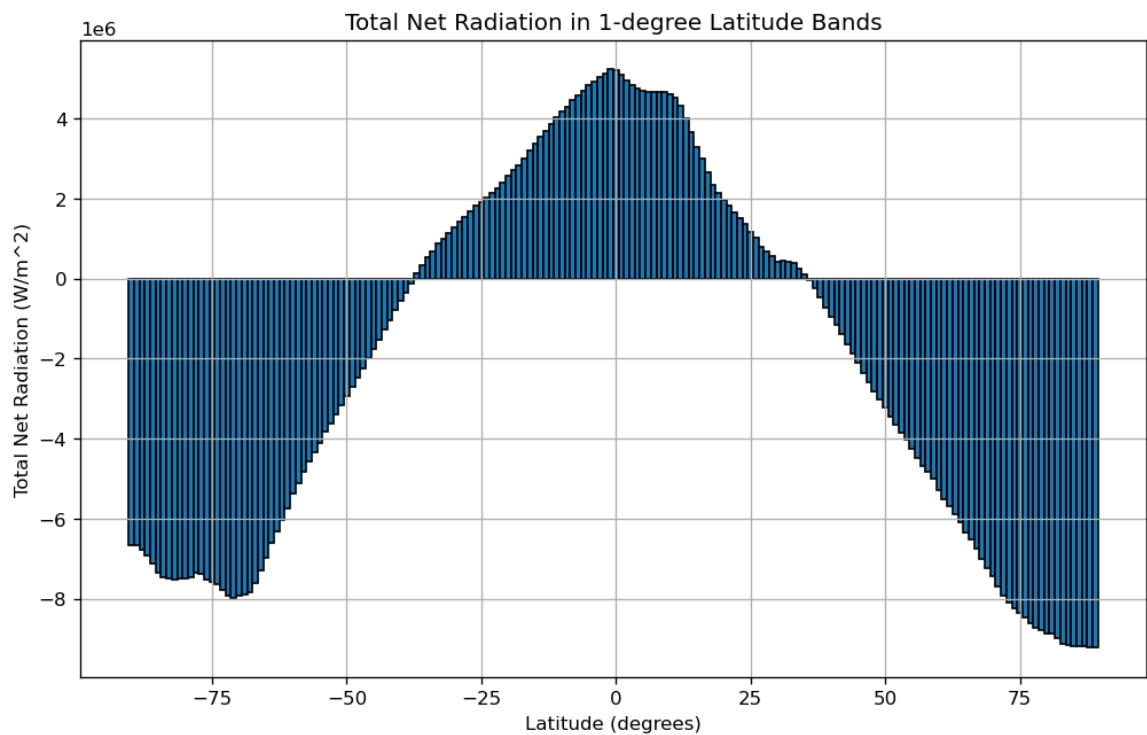
In [14]:
```python
# 2.4
swf = TOA['toa_cre_sw_mon']  # Shortwave radiation flux
lwf = TOA['toa_cre_lw_mon']  # Longwave radiation flux
cld = TOA['cldarea_total_daynight_mon']  # Total cloud area fraction

# Define low and high cloud thresholds
cf = TOA['cldarea_total_daynight_mon'] / 100

# Group the grid cells into low and high cloud area regions based on the cloud fracti
low_cloud_threshold = cf <= 0.25
high_cloud_threshold = cf >= 0.75

# Calculate time-mean for low and high cloud areas
swf_low_cloud_mean = swf.where(low_cloud_threshold).mean(dim='time')
swf_high_cloud_mean = swf.where(high_cloud_threshold).mean(dim='time')
lwf_low_cloud_mean = lwf.where(low_cloud_threshold).mean(dim='time')
lwf_high_cloud_mean = lwf.where(high_cloud_threshold).mean(dim='time')

# Plot
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(5, 16))
swf_low_cloud_mean.plot(ax=ax1)
ax1.set_title('swf low cloud mean')

swf_high_cloud_mean.plot(ax=ax2)
ax2.set_title('swf high cloud mean')

lwf_low_cloud_mean.plot(ax=ax3)
ax3.set_title('lwf low cloud mean')

lwf_high_cloud_mean.plot(ax=ax4)
ax4.set_title('lwf high cloud mean')

plt.tight_layout()
plt.show()
```

swf low cloud mean

swf high cloud mean

lwf low cloud mean

lwf high cloud mean
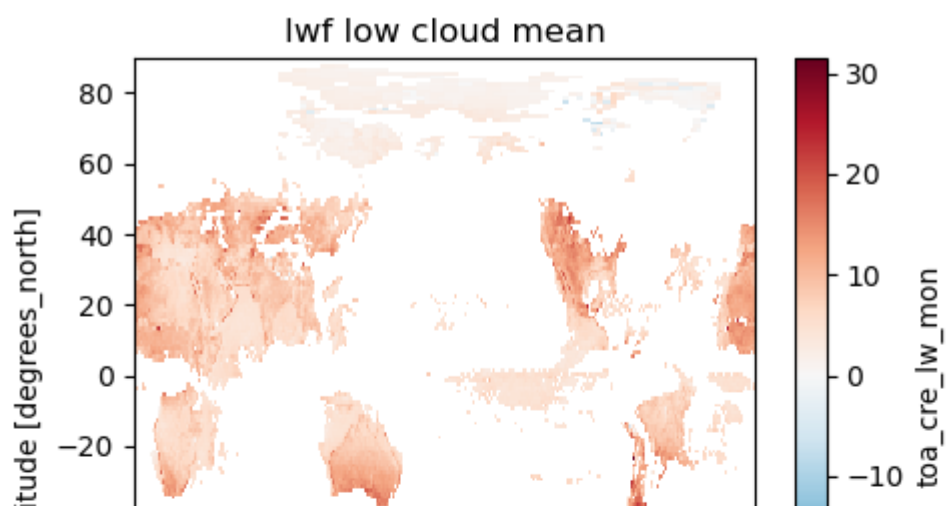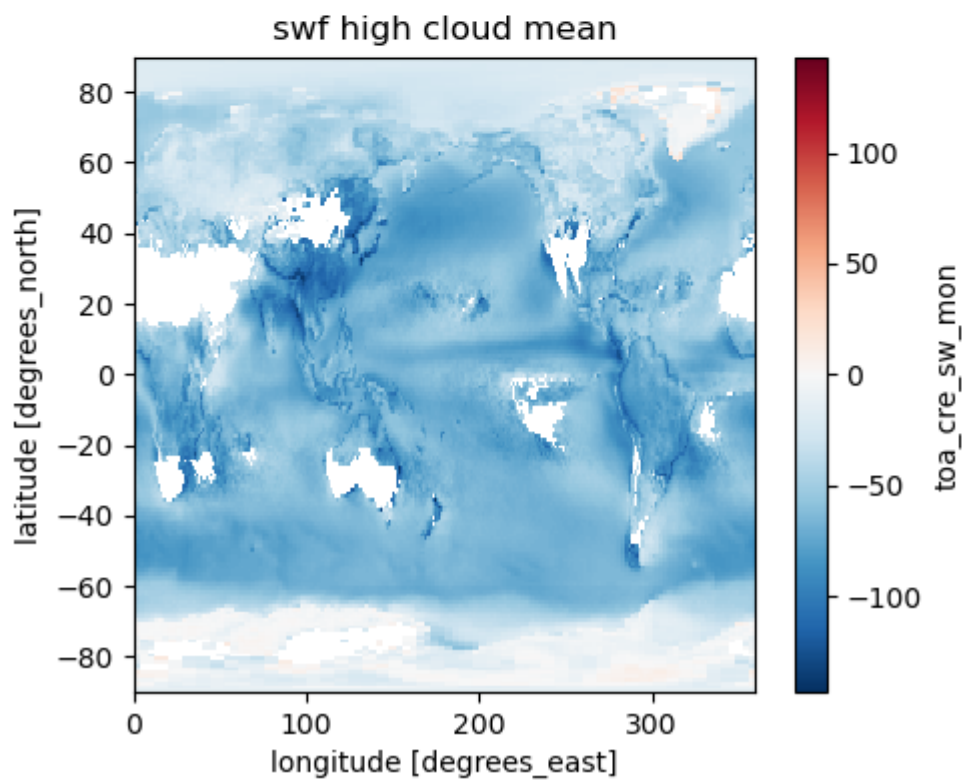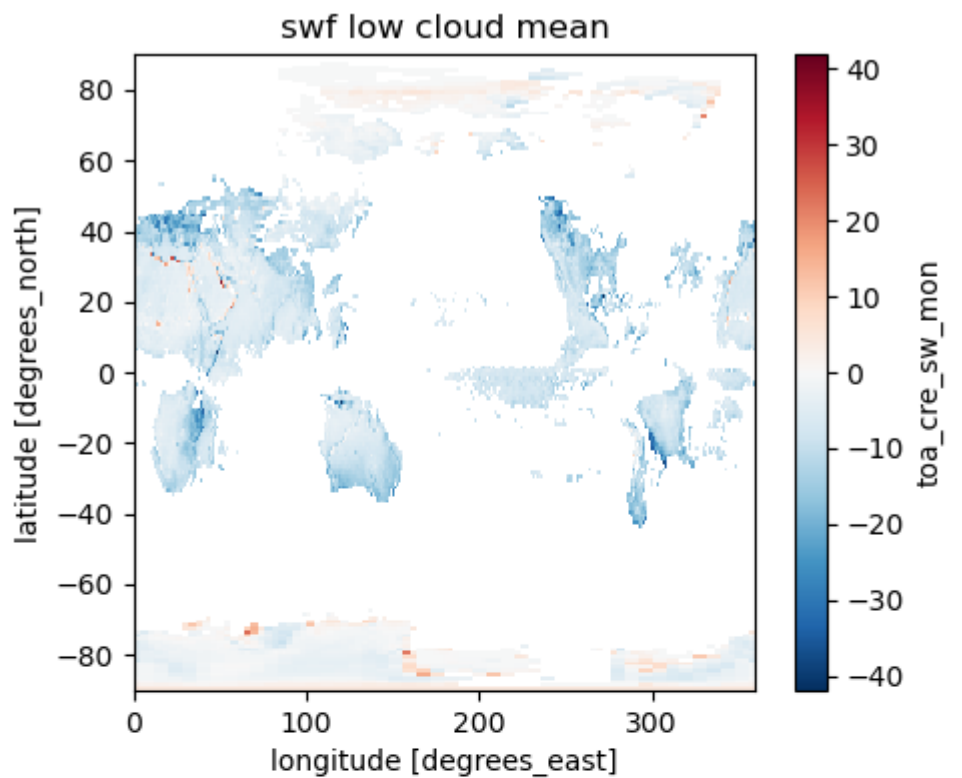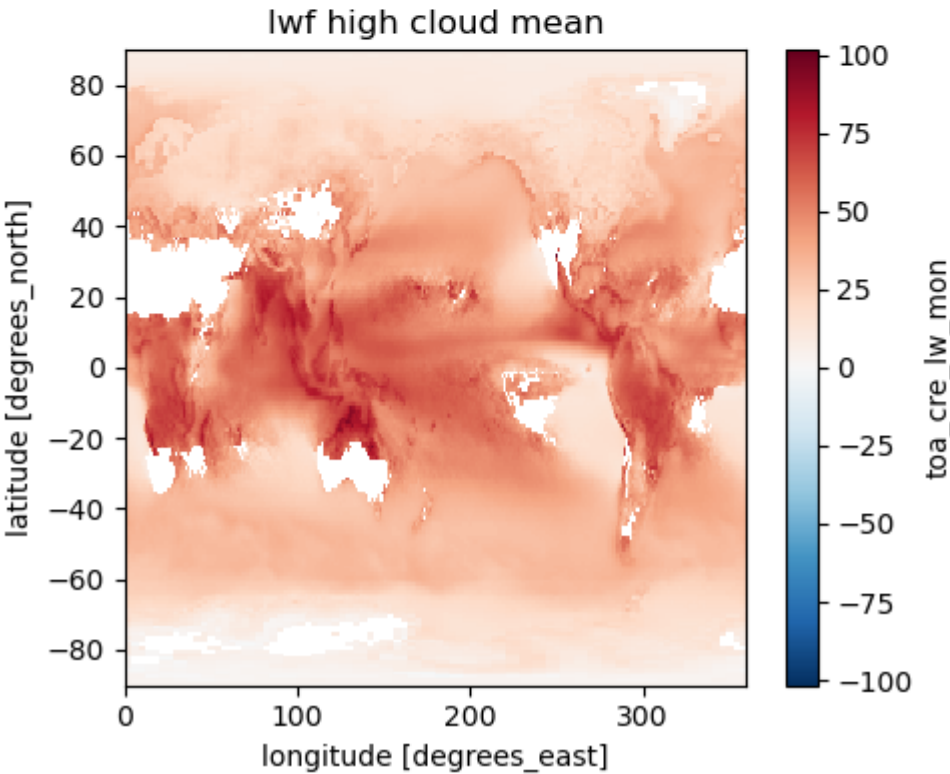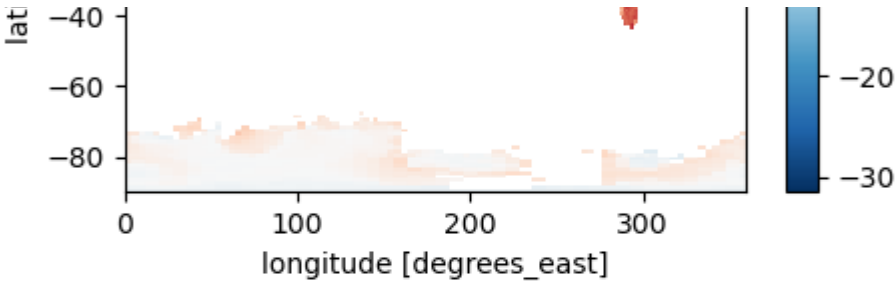
```python
In [15]: # 2.5
mean_longwave = TOA['toa_lw_all_mon'].mean(dim='time')
mean_shortwave = TOA['toa_sw_all_mon'].mean(dim='time')

lw_plus_low_cld = mean_longwave + lwf_low_cloud_mean
lw_plus_high_cld = mean_longwave + lwf_high_cloud_mean
sw_plus_low_cld = mean_longwave + swf_low_cloud_mean
sw_plus_high_cld = mean_longwave + swf_high_cloud_mean

fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(5, 16))
lw_plus_low_cld.plot(ax=ax1)
ax1.set_title('lw plus low cld')

lw_plus_high_cld.plot(ax=ax2)
ax2.set_title('lw plus high cld')

sw_plus_low_cld.plot(ax=ax3)
ax3.set_title('sw plus low cld')

sw_plus_high_cld.plot(ax=ax4)
ax4.set_title('sw plus high cld')

plt.tight_layout()
plt.show()
```

lw plus low cld

lw plus high cld

sw plus low cld

sw plus high cld
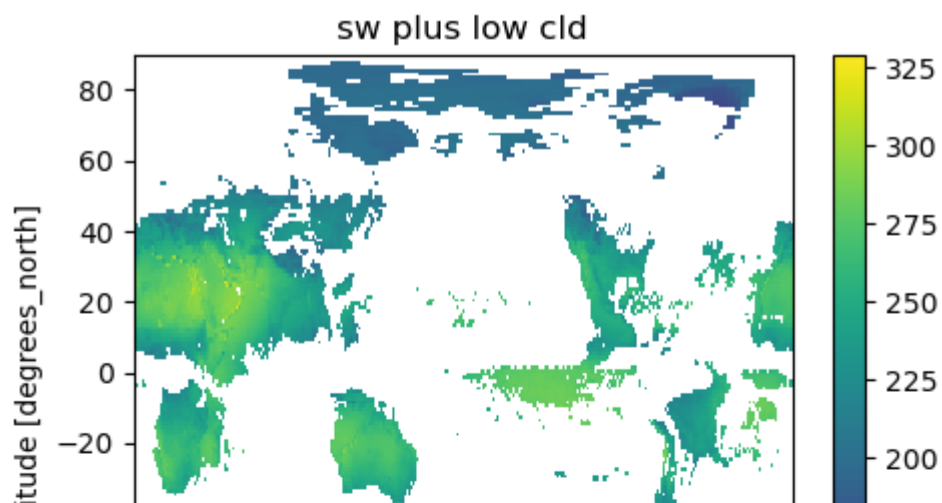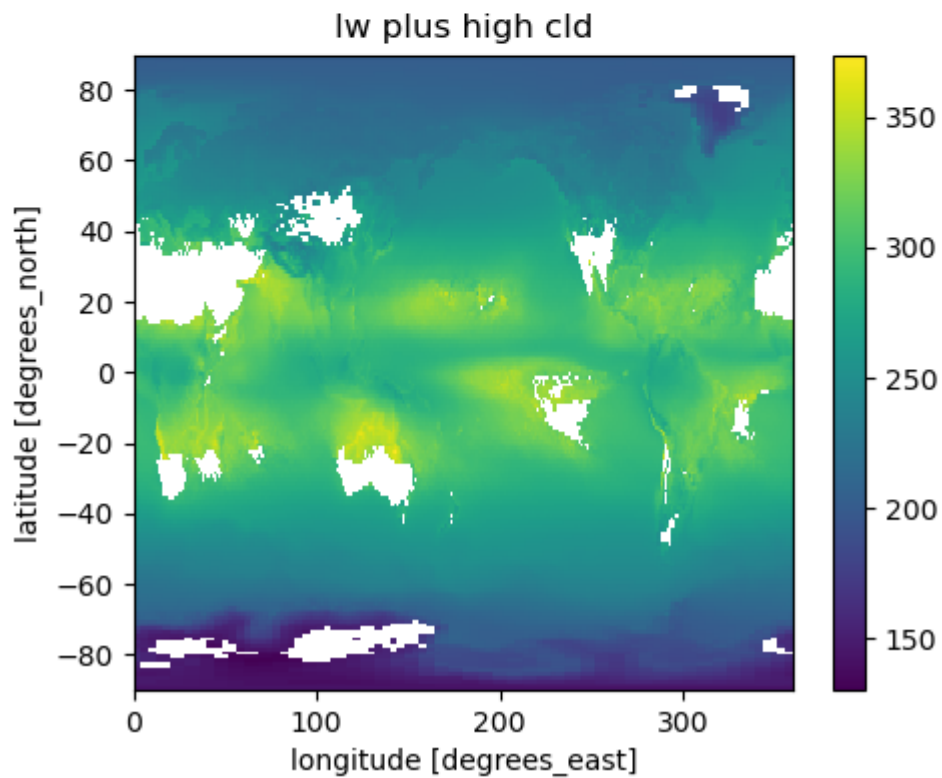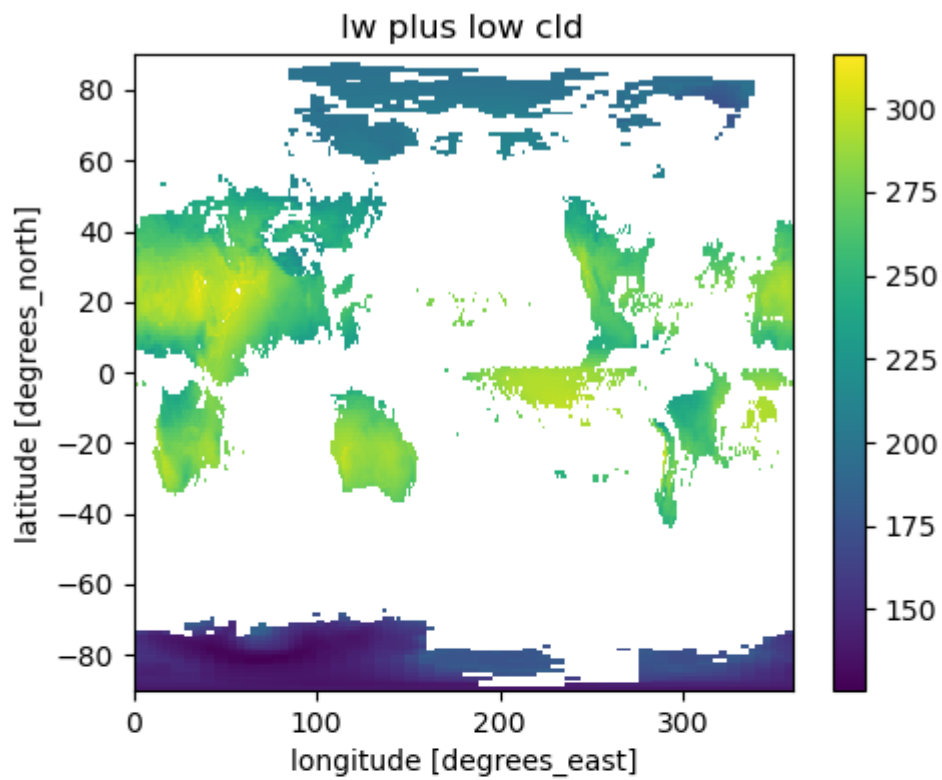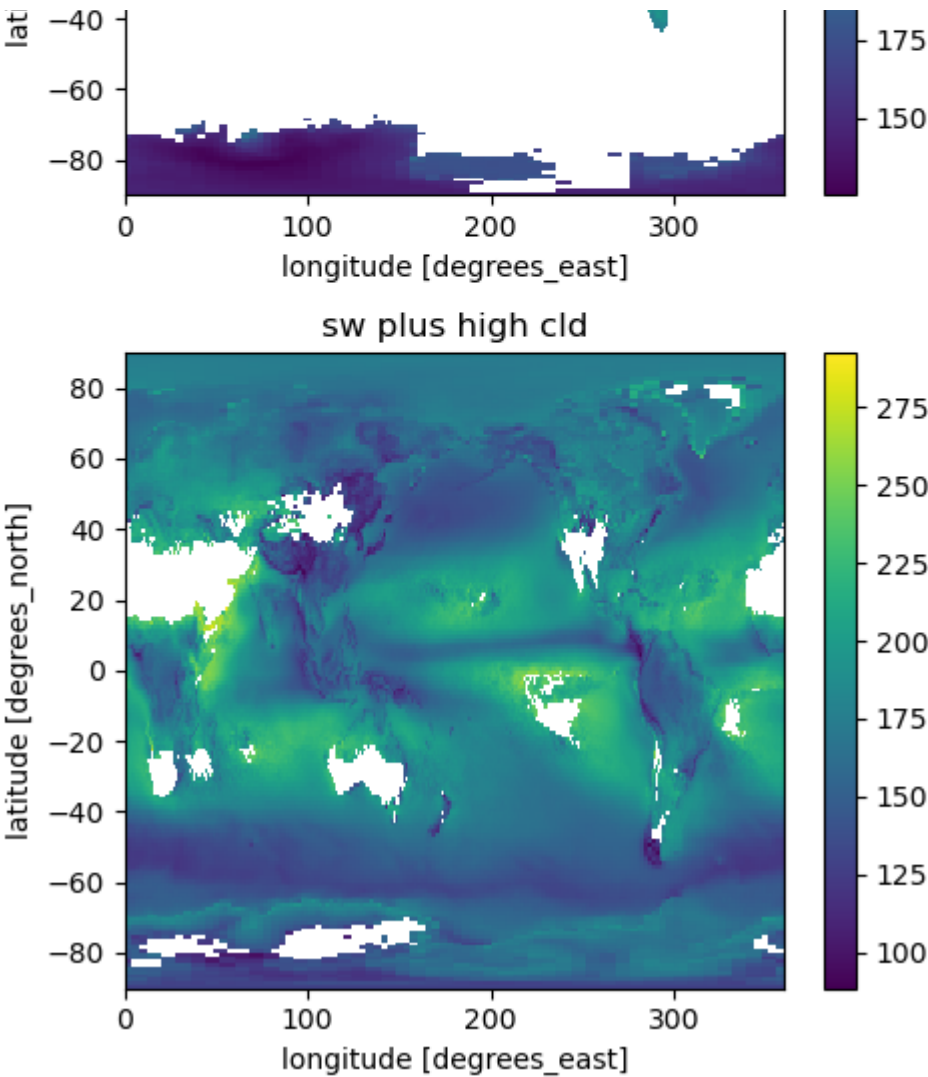
In [16]:
```python
# 3
data = xr.open_dataset("MERRA2_400.tavgU_2d_aer_Nx.202309.nc4", engine="netcdf4")
data
```

Out[16]:

xarray.Dataset

▶ Dimensions: (**lon**: 576, **lat**: 361, **time**: 24)

▼ Coordinates:

lon	(lon)	float64	-180.0 -179.4 ... 178.8 179.4	📄 🗄
lat	(lat)	float64	-90.0 -89.5 -89.0 ... 89.5 90.0	📄 🗄
time	(time)	datetime64[ns]	2023-09-01T00:30:00 ... 2023-09-...	📄 🗄

▶ Data variables: (50)

▶ Indexes: (3)

▶ Attributes: (30)

In [17]:
```python
# 3.1
data_clim=data.BCANGSTR.groupby("time.month")
data_clim.mean(dim=['lon','lat']).plot()
```

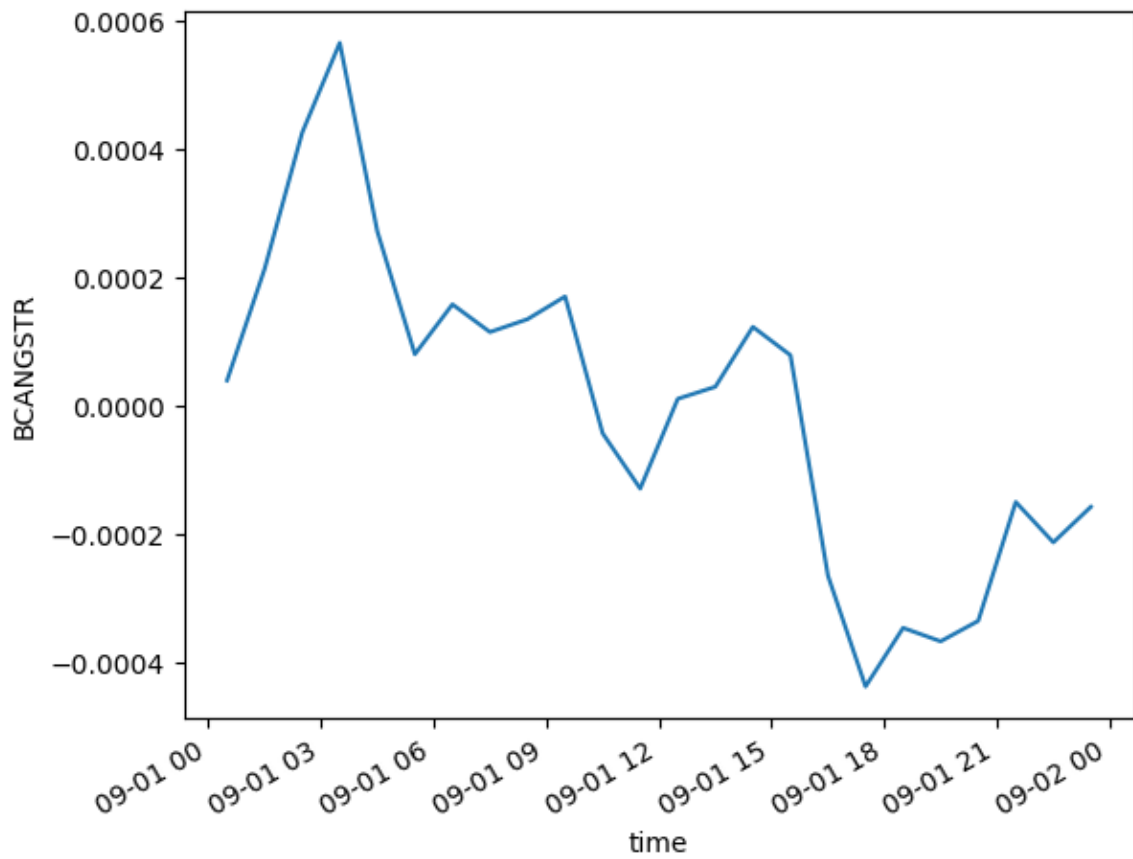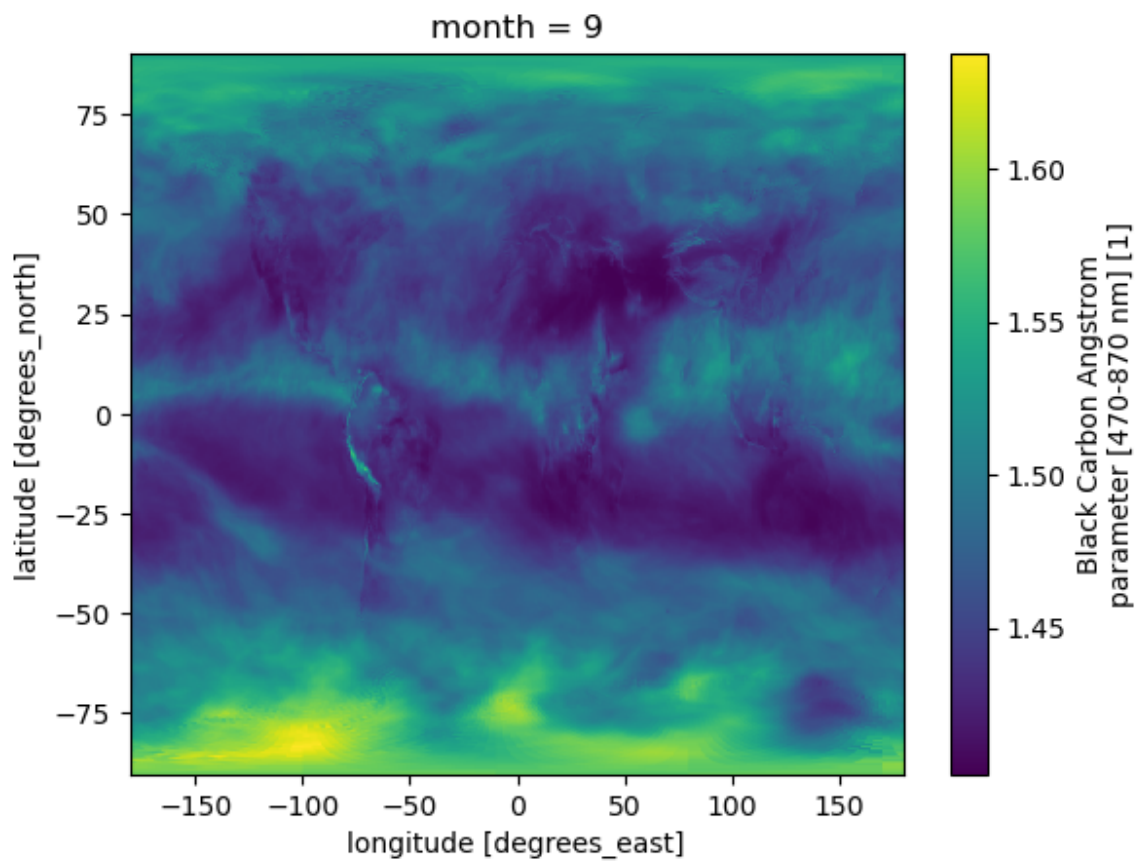Out[17]: [<matplotlib.lines.Line2D at 0x2380b088710>]
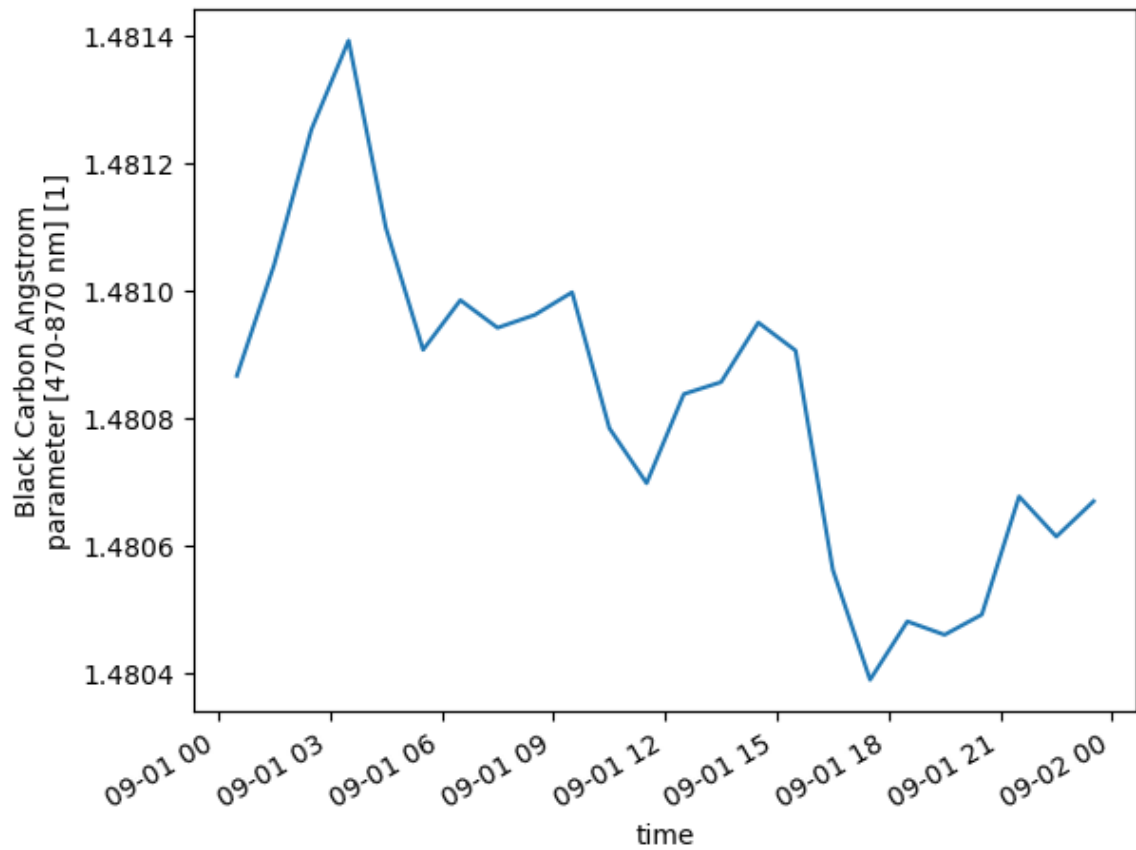
In [18]:
```
data_dif=data_clim-data_clim.mean(dim='time')
data_dif.mean(dim=['lon','lat']).plot()
```

Out[18]: [<matplotlib.lines.Line2D at 0x2381095bed0>]

In [19]:
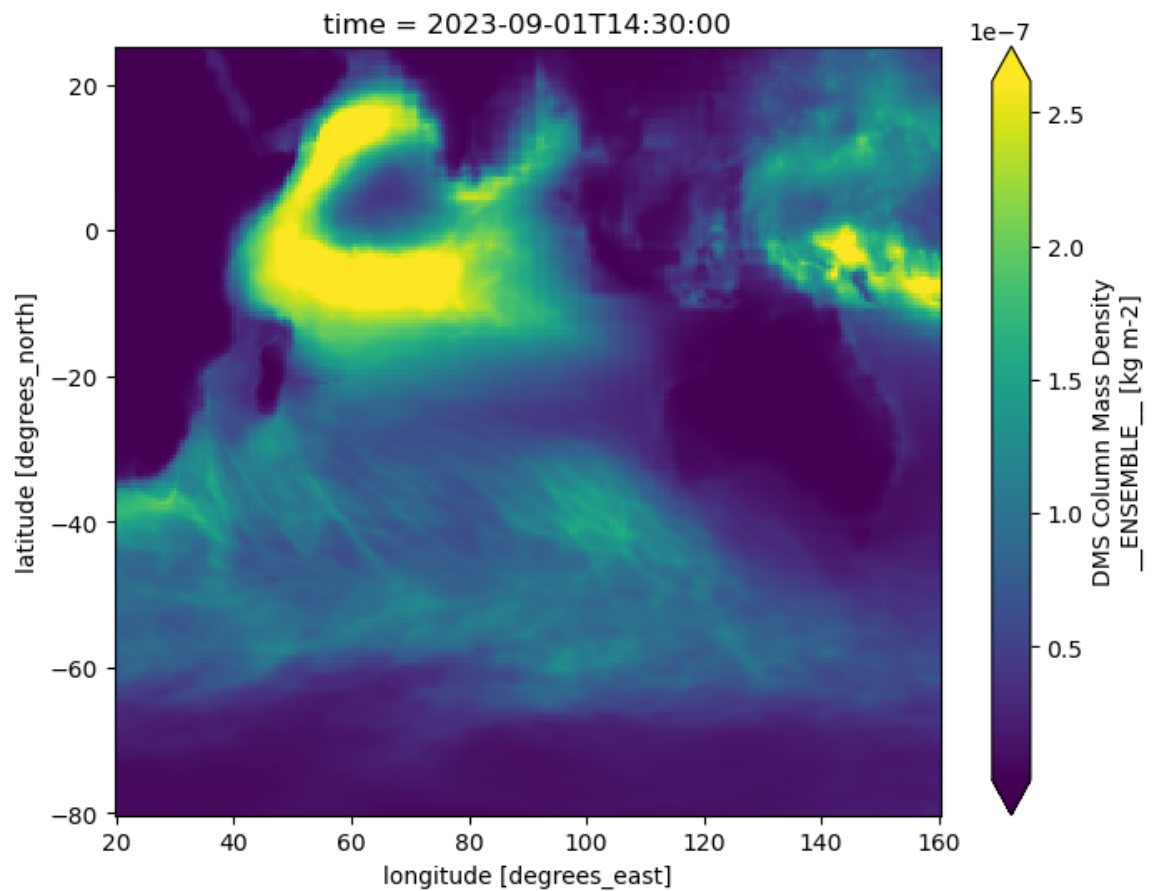```
# 3.2
# 3.2.1
data1=data.BCANGSTR.groupby("time.month").mean()
data1.plot()
```

Out[19]: <matplotlib.collections.QuadMesh at 0x238109c6650>

```
In [20]:  # 3.2.2
          data.DMSCMASS.isel(time=-10).sel(lon=slice(20, 160),
                                           lat=slice(-80, 25)).plot(robust=True, figsize=(8, 6))
```
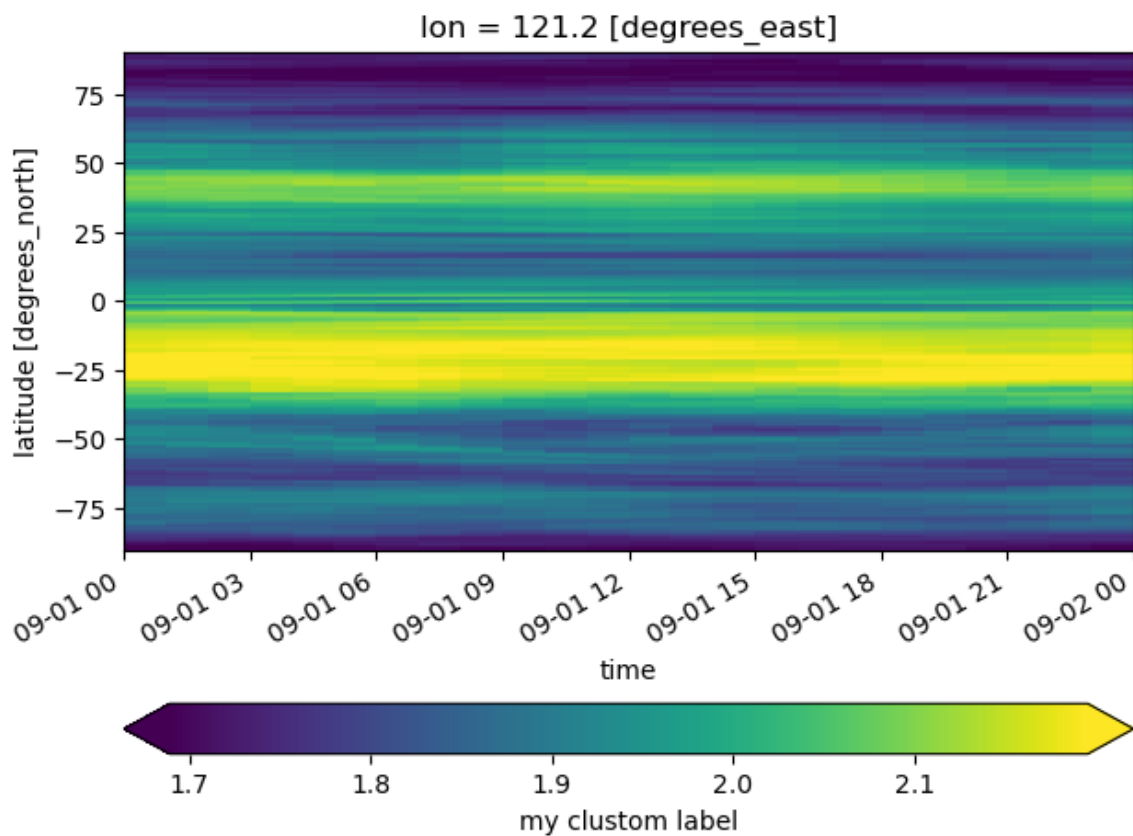
Out[20]: <matplotlib.collections.QuadMesh at 0x23810a0dc90>

In [21]:
```python
# 3.2.3
colorbar_kwargs = {
    "orientation": "horizontal",
    "label": "my clustom label",
    "pad": 0.2,
}

data.OCANGSTR.sel(lon=121.5, method='nearest').plot(
    x="time",
    robust=True,
    cbar_kwargs=colorbar_kwargs,
)

plt.tight_layout()
plt.show()
```
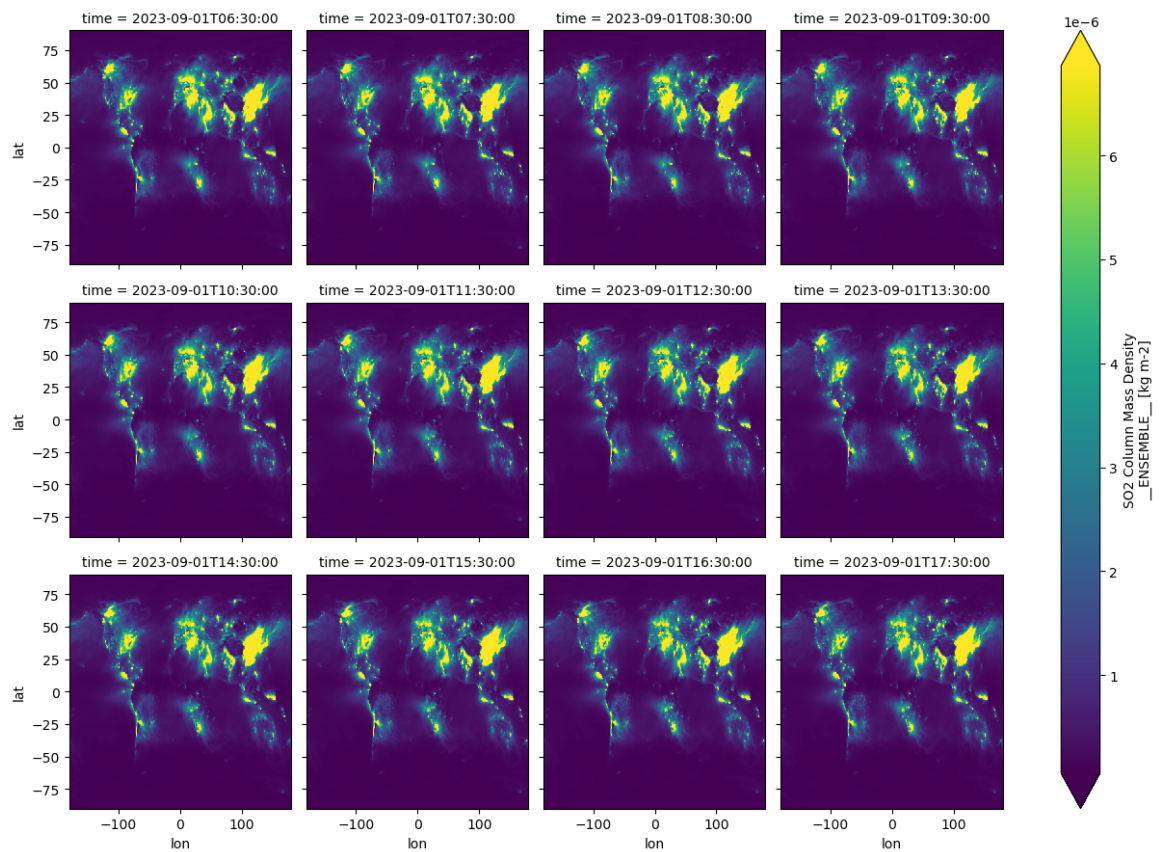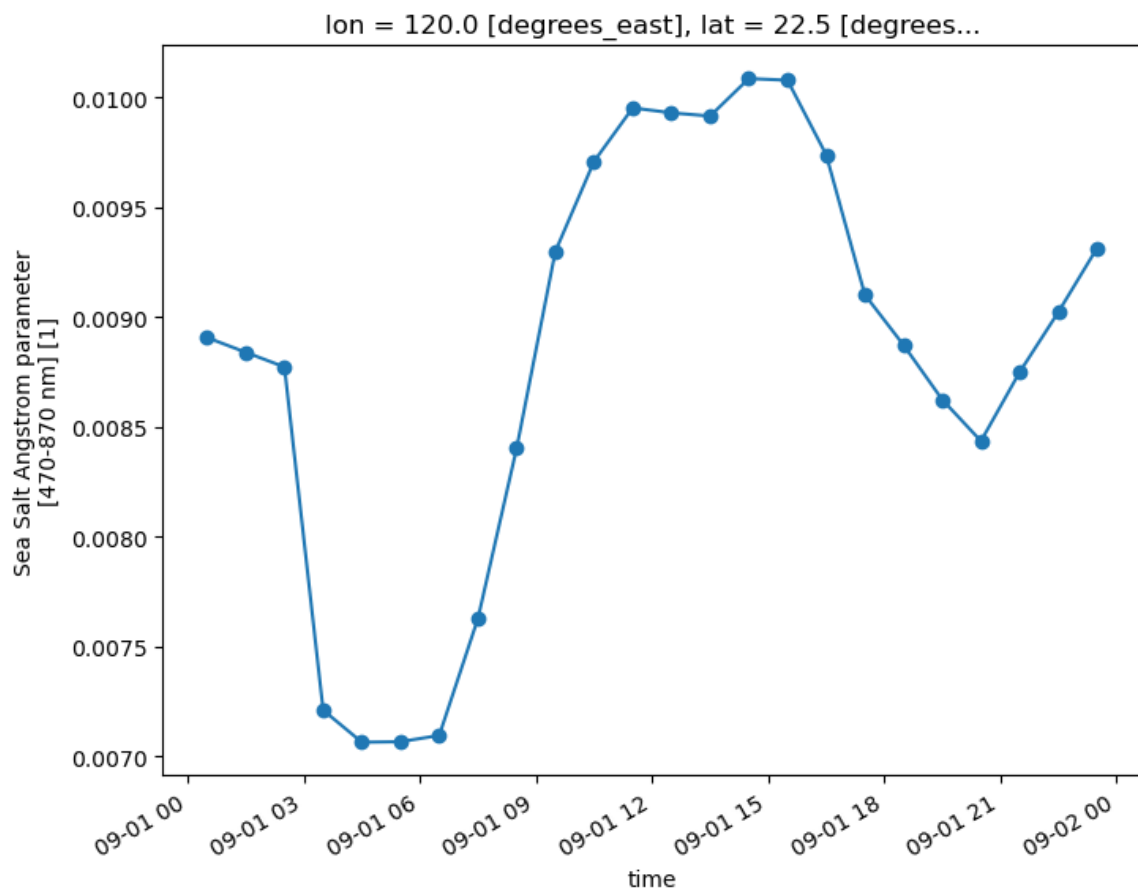
In [22]:
```
# 3.2.4
data.SO2CMASS.sel(time=slice("2023-09-01T06:30:00", "2023-09-01T17:30:00")).plot(col=
```

Out[22]: <xarray.plot.facetgrid.FacetGrid at 0x23810e38510>

In [23]:
```
# 3.2.5
data.SSANGSTR.sel(lon=120, lat=22.5, method='nearest').plot(marker="o", size=6)
```

Out[23]: [<matplotlib.lines.Line2D at 0x2380aba2c50>]

In [ ]: