Assignment 03

LinQiangMa

November 2023

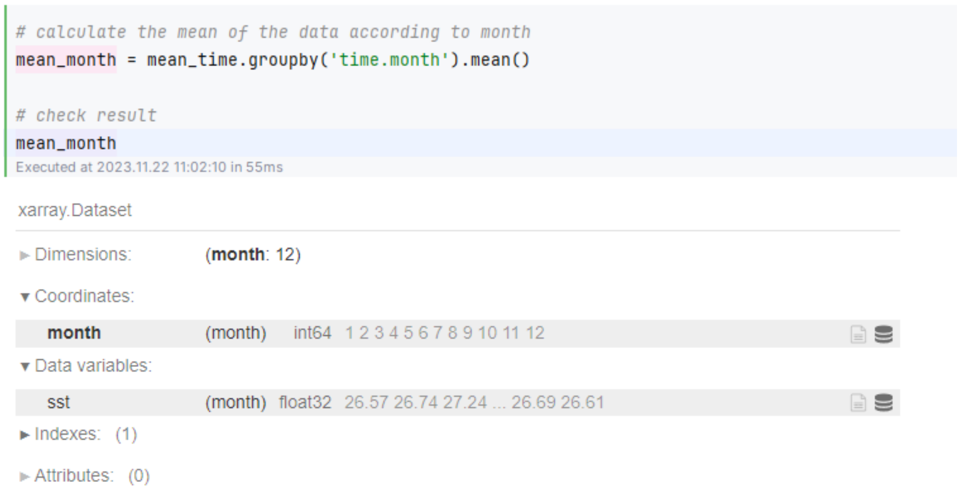
Problem1

1.1

```
# import libraries
import numpy as np
import pandas as pd
import xarray as xr
import matplotlib as mpl
import matplotlib.pyplot as plt
from datetime import datetime
from matplotlib.ticker import FuncFormatter
import matplotlib.gridspec as gridspec
import statsmodels.api as sm
import pandas as pd
import hvplot.xarray
%matplotlib inline
#%%
# import data
data = xr.open_dataset("NOAA_NCDC_ERSST_v3b_SST.nc", engine="netcdf4")

# check data
data
#%%
# 1.1
# slice the area needed (5N-5S, 170W-120W)
data = data.sel(lat=slice(-5,5), lon=slice(190, 240))

# check data
data
#%%
# check null values
data.isnull().sum()
#%%
# calculate the mean of the data according to time
mean_time = data.mean(dim=['lat', 'lon'])

#check mean_time
mean_time
#%%
# calculate the mean of the data according to month
mean_month = mean_time.groupby('time.month').mean()

# check result
mean_month
#%%
# find the anomalies
anomalies =mean_time.groupby('time.month') - mean_month

# check result
```

```
anomalies
#%%
```

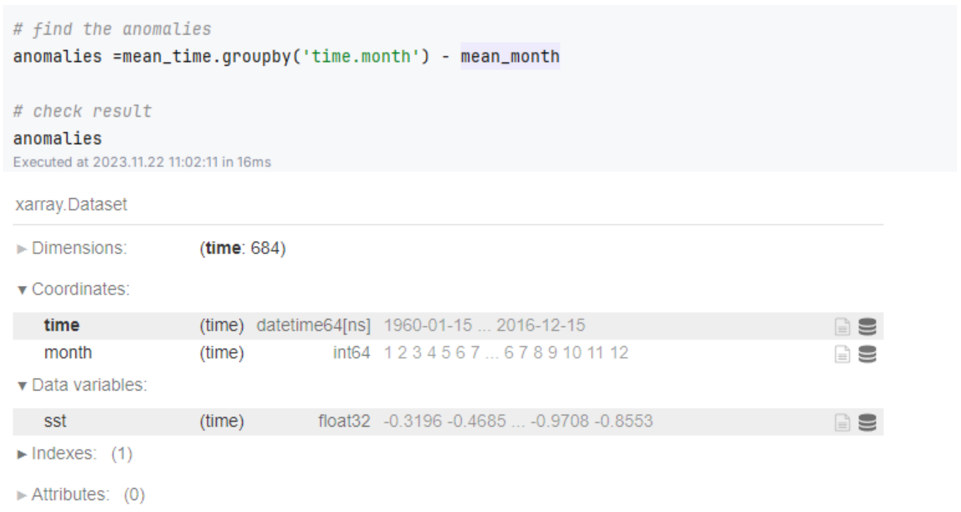
output:



```
# calculate the mean of the data according to month
mean_month = mean_time.groupby('time.month').mean()

# check result
mean_month
Executed at 2023.11.22 11:02:10 in 55ms
```

```
xarray.Dataset
▶ Dimensions:       (month: 12)
▼ Coordinates:
    month           (month)   int64    1 2 3 4 5 6 7 8 9 10 11 12
▼ Data variables:
    sst             (month)   float32  26.57 26.74 27.24 ... 26.69 26.61
▶ Indexes: (1)
▶ Attributes: (0)
```

Figure 1: monthly climatology for SST



```
# find the anomalies
anomalies =mean_time.groupby('time.month') - mean_month

# check result
anomalies
Executed at 2023.11.22 11:02:11 in 16ms
```

```
xarray.Dataset
▶ Dimensions:       (time: 684)
▼ Coordinates:
    time            (time)   datetime64[ns]  1960-01-15 ... 2016-12-15
    month           (time)   int64           1 2 3 4 5 6 7 ... 6 7 8 9 10 11 12
▼ Data variables:
    sst             (time)   float32         -0.3196 -0.4685 ... -0.9708 -0.8553
▶ Indexes: (1)
▶ Attributes: (0)
```

Figure 2: anomalies

1.2

```
# 1.2
# calculate the rolling mean
mean_rolling = anomalies.rolling(time=3, center=False, min_periods=1).mean()

# check result
mean_rolling
#%%
# transform to dataframe to plot, I am inspired by my roommate Zhouzhou
tf = mean_rolling.to_dataframe().reset_index()
anomalies_df = anomalies.to_dataframe().reset_index()

# check  result
tf
#%%
# add rolling mean column and check result
anomalies_df['mean_rolloing'] = tf['sst']
anomalies_df
#%%
plt.figure(figsize=(18,6))

# adjust the color
colour = np.where(anomalies_df['sst']>0, 'red', 'blue')

# plot bar chart
#Zhouzhou inspired me to adjust the width
plt.bar(anomalies_df['time'], anomalies_df['sst'], width=100, color=colour)

#plot rolling_mean
plt.plot(anomalies_df['time'],anomalies_df['mean_rolloing'], color='black',label='3thm running mean')

#plot the zero and threshold line
plt.plot(anomalies_df['time'],np.zeros(684),color='black')
plt.plot(anomalies_df['time'],np.zeros(684)+0.5,color='red', linestyle='--',label='El Nino Threshold')
plt.plot(anomalies_df['time'],np.zeros(684)-0.5,color='blue', linestyle='--',label='LA Nina Threshold'

# adjust the label and ticks
# set label
plt.xlabel('year')
plt.ylabel('Anomaly in Degrees ℃')

#set ticks
y = [-3.0,-2.0,-1.0,0.0,1.0,2.0,3.0]
plt.yticks(y,labels=y)
plt.xlim(datetime(1960,1,1),datetime(2016,12,31))
#plot title
plt.title('SST Anomaly in Nino 3.4 Region(5N-5S,120-170W)')

#plot grid
plt.grid(linestyle='--',axis='x')
plt.xlim()

#tick parameter
plt.tick_params(axis='y',left=True,direction='in',which='both')
plt.tick_params(axis='x',top=False,bottom=True,direction='out',which='both')
plt.legend()
```

```
#%%
```

output:

```
<matplotlib.legend.Legend at 0x1733bf02b10>
```

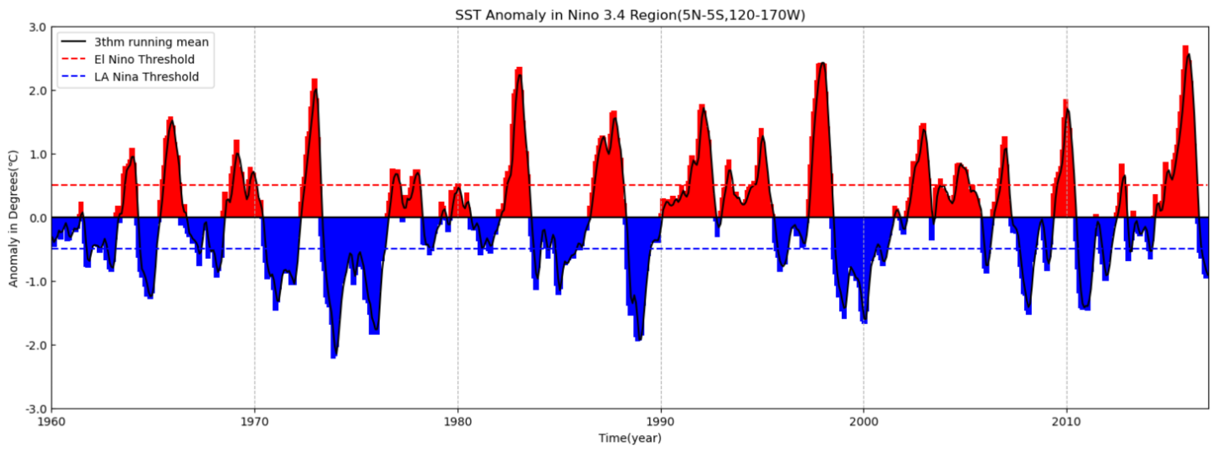


Figure 3: anomalies

Problem2

2.1

```
# import data
data = xr.open_dataset('CERES_EBAF-TOA_200003-201701.nc',engine='netcdf4')

# check data
data.isnull().sum()
#%%
# check data
data
#%% md

#%%
# calculate the mean according to time
mean_time = data.mean(dim='time')

# calculate and check the result
mean_time["result1"]=mean_time["solar_mon"]+mean_time['toa_sw_all_mon']+mean_time['toa_lw_all_mon']
mean_time["result2"]=mean_time["solar_mon"]-mean_time['toa_sw_all_mon']-mean_time['toa_lw_all_mon']
mean_time
#%%
# 2.1
plt.figure(figsize=(12,10),dpi=120)

# plot the short wave
plt.subplot(2,3,1)
plt.contourf(mean_time['toa_sw_all_mon'],cmap='Reds')
plt.colorbar()
plt.clim(-300, 300)
plt.title('short wave')

# plot the long wave
plt.subplot(2,3,2)
plt.contourf(mean_time['toa_lw_all_mon'],cmap='Reds')
plt.colorbar()
plt.clim(-300, 300)
plt.title('long wave')

# plot the sun solar radiation
plt.subplot(2,3,3)
plt.contourf(mean_time["solar_mon"],cmap='Reds')
plt.colorbar()
plt.clim(-150, 450)
plt.title('solar radiation')

# plot the net flux
plt.subplot(2,3,4)
plt.contourf(mean_time["toa_net_all_mon"],cmap='Reds')
plt.colorbar()
plt.clim(-300, 300)
plt.title('the net flux')


# plot the calculate result "solar_mon"+'toa_sw_all_mon'+'toa_lw_all_mon'
plt.subplot(2,3,5)
```

```python
plt.contourf(mean_time["result1"],cmap='Reds')
plt.colorbar()
plt.clim(-300, 1000)
plt.title("solar+sw+ls")

# plot the calculate result "solar_mon"-('toa_sw_all_mon'+'toa_lw_all_mon')
plt.subplot(2,3,6)
plt.contourf(mean_time["result2"],cmap='Reds')
plt.colorbar()
plt.clim(-300, 300)
plt.title("solar-sw-ls")
#%%
```

From the graph, we can observe that net flux =solar radiation - longwave - shortwave

output:

```
Text(0.5, 1.0, 'solar-sw-ls')
```

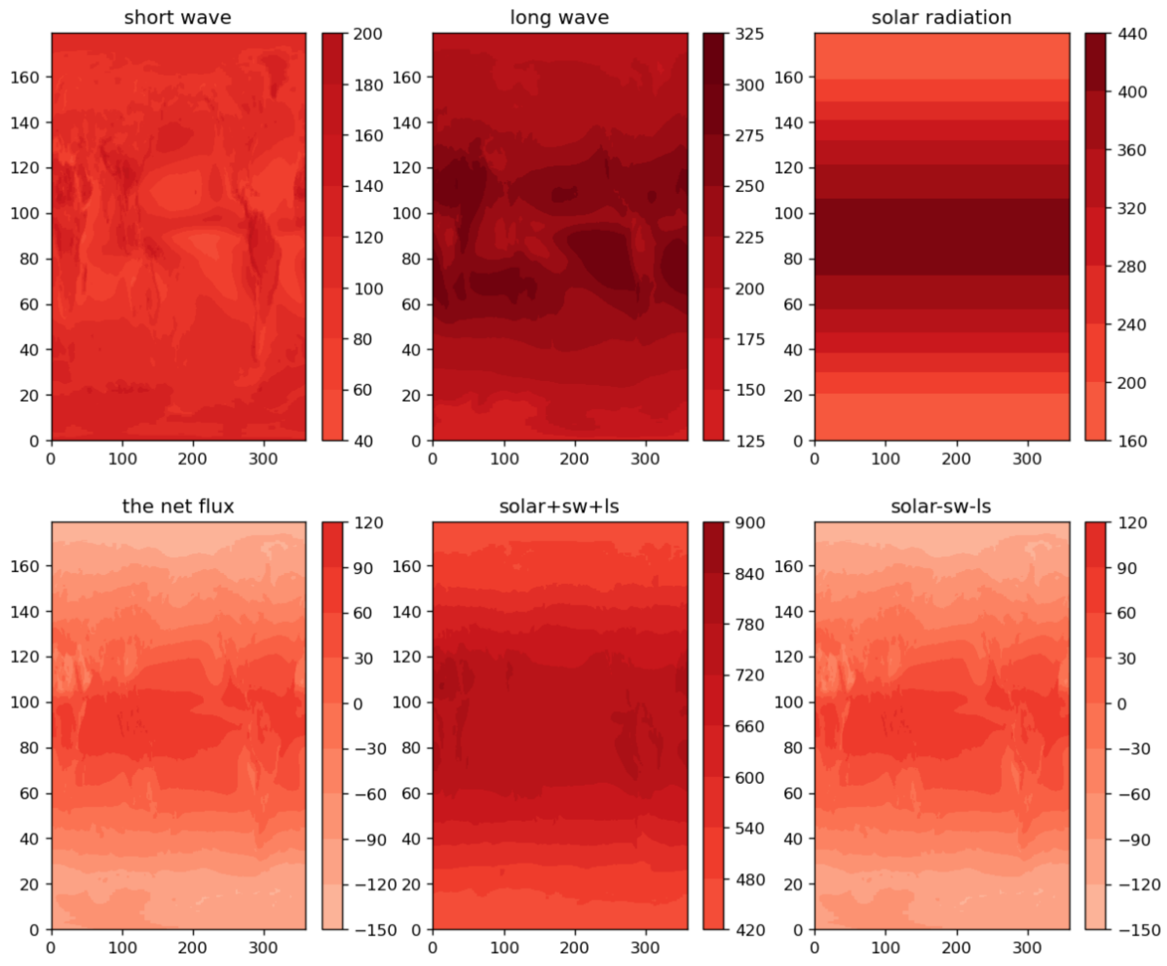
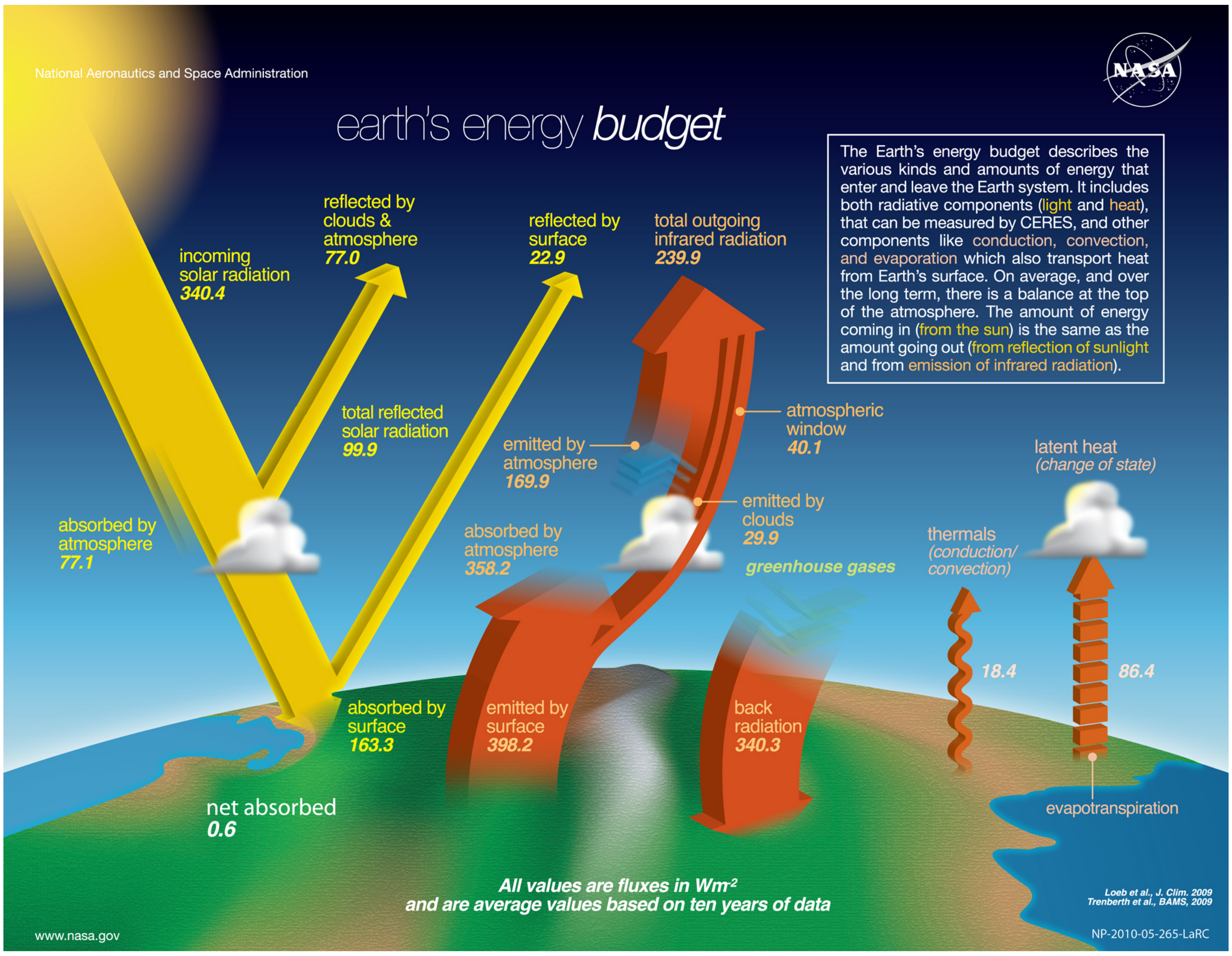


Figure 4: Enter Caption

2.2

```
# 2.2
# Calculate the TOA incoming solar
mean_time.solar_mon.sum()/360/180
#%%
# Calculate the TOA outgoing longwave
mean_time.toa_lw_all_mon.sum()/360/180
#%%
# Calculate the TOA outgoing shortwave
mean_time.toa_sw_all_mon.sum()/360/180
#%%
```

Finding:
The mean temperature1 in 2003 = -1.27 °C
The median temperature1 in 2003 = -0.5345 °C
The max temperature1 in 2003 = 21.04 °C
The min temperature1 in 2003 = -28.66 °C
The std of temperature1 in 2003 = 9.69 °C

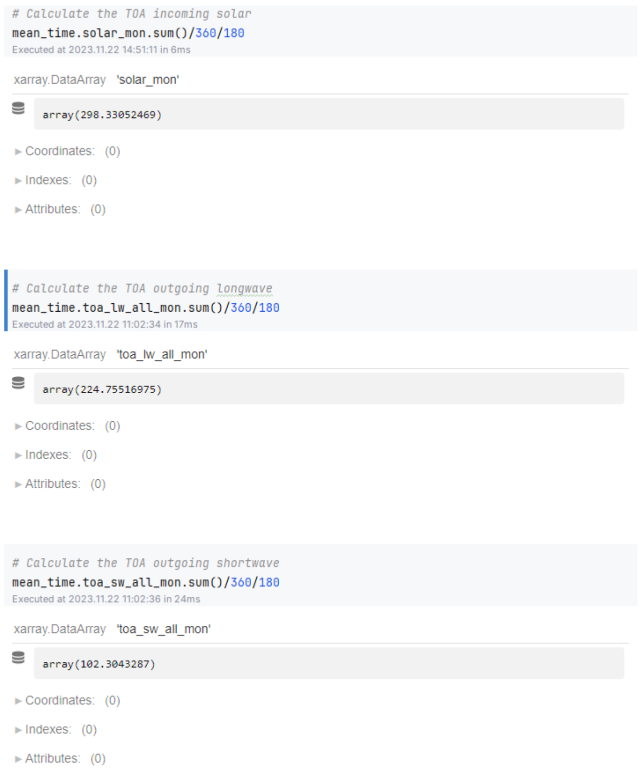

National Aeronautics and Space Administration
earth's energy budget
incoming solar radiation 340.4
reflected by clouds & atmosphere 77.0
reflected by surface 22.9
total outgoing infrared radiation 239.9
total reflected solar radiation 99.9
emitted by atmosphere 169.9
atmospheric window 40.1
latent heat (change of state)
absorbed by atmosphere 77.1
emitted by clouds 29.9
absorbed by atmosphere 358.2
greenhouse gases
thermals (conduction/ convection)
18.4
86.4
absorbed by surface 163.3
emitted by surface 398.2
back radiation 340.3
evapotranspiration
net absorbed 0.6
All values are fluxes in Wm-2 and are average values based on ten years of data
Loeb et al., J. Clim. 2009
Trenberth et al., BAMS, 2009
www.nasa.gov
NP-2010-05-265-LaRC

The Earth's energy budget describes the various kinds and amounts of energy that enter and leave the Earth system. It includes both radiative components (light and heat), that can be measured by CERES, and other components like conduction, convection, and evaporation which also transport heat from Earth's surface. On average, and over the long term, there is a balance at the top of the atmosphere. The amount of energy coming in (from the sun) is the same as the amount going out (from reflection of sunlight and from emission of infrared radiation).

Figure 5: energy budget

From the graph, we can observe that
the TOA incoming solar = 298, which is 340 in figure 5
outgoing longwave = 225 which is 240 in figure 5
outgoing shortwave = 102, which is 100 in figure 5
the max error is about 10%, I think they approximately match up with the cartoon in figure 5

output:



```
# Calculate the TOA incoming solar
mean_time.solar_mon.sum()/360/180
Executed at 2023.11.22 14:51:11 in 6ms

xarray.DataArray 'solar_mon'
array(298.33052469)
► Coordinates: (0)
► Indexes: (0)
► Attributes: (0)

# Calculate the TOA outgoing longwave
mean_time.toa_lw_all_mon.sum()/360/180
Executed at 2023.11.22 11:02:34 in 17ms

xarray.DataArray 'toa_lw_all_mon'
array(224.75516975)
► Coordinates: (0)
► Indexes: (0)
► Attributes: (0)

# Calculate the TOA outgoing shortwave
mean_time.toa_sw_all_mon.sum()/360/180
Executed at 2023.11.22 11:02:36 in 24ms

xarray.DataArray 'toa_sw_all_mon'
array(102.3043287)
► Coordinates: (0)
► Indexes: (0)
► Attributes: (0)
```

Figure 6: result

2.3

```
# 2.3
# Calculate and plot the  net radiation in each 1-degree latitude band.
mean_lat=mean_time.sum(dim='lon')/180

#check the result
mean_lat
#%%

# plot the bar
plt.figure(figsize=(10,8),dpi=120)

# set color
color = np.where(mean_lat['toa_net_all_mon']>0, 'red','blue')

# plot
plt.bar(mean_lat['lat'],mean_lat['toa_net_all_mon'],color=color,width=1)

# set grid
plt.grid(alpha=0.3,axis='x')

# set label
plt.xlabel('Altitude(°)')
plt.ylabel('net flux(W/m^2)')
plt.tick_params(axis='y',direction='in')
# set title
plt.title('Net total radiation in each 1-degree latitude')
```

output:

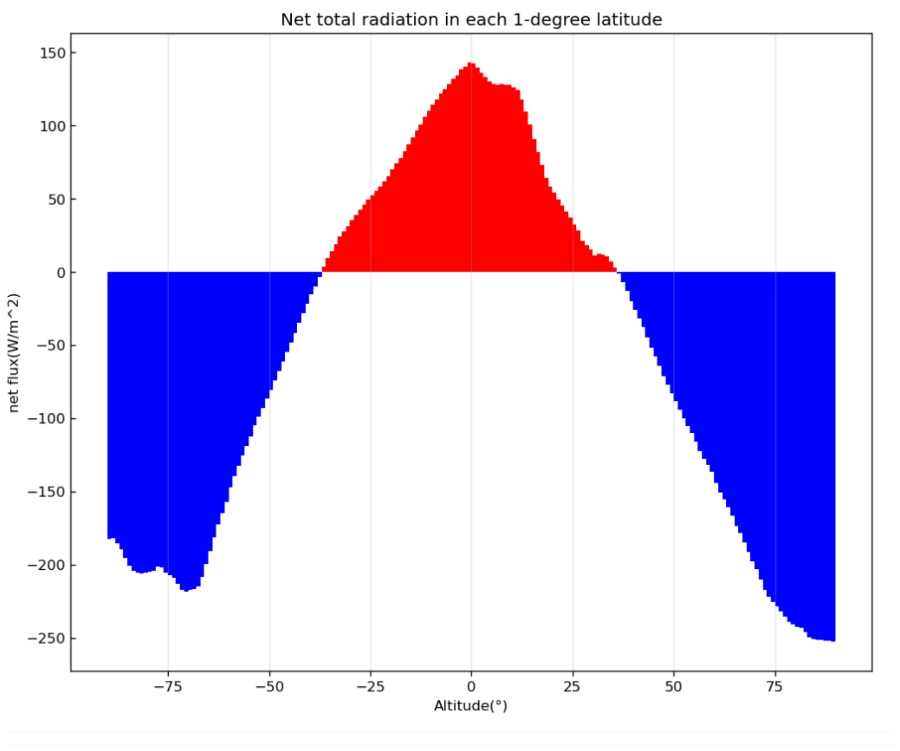


Figure 7: Net total radiation in each 1-degree latitude

2.4

```
# 2.4
# select cloud area  I was inspired by QiMingLiu
lowcloud_area = data.where((data['cldarea_total_daynight_mon']<25)|(data['cldarea_total_daynight_mon']=
lowcloud_area = lowcloud_area.mean(dim='time')
highcloud_area = data.where((data['cldarea_total_daynight_mon']>75)|(data['cldarea_total_daynight_mon']
highcloud_area = highcloud_area.mean(dim='time')

# check the result
lowcloud_area
#%%
# check the result
highcloud_area
#%%
# plot
plt.figure(figsize=(12,8),dpi=120)
grid = plt.GridSpec(2, 2)

# plot the long wave of low cloud areas and high cloud areas
# low cloud areas
plt.subplot(grid[0:1, 0:1])
plt.contourf(lowcloud_area['toa_lw_all_mon'],cmap='Reds')
plt.colorbar()
plt.clim(0,400)
plt.title('Longwave of lowcloud area')

#high cloud areas
plt.subplot(grid[0:1, 1:2])
plt.contourf(highcloud_area['toa_lw_all_mon'],cmap='Reds')
plt.colorbar()
plt.clim(0,400)
plt.title('Longwave of lowcloud area')

# plot the long wave of low cloud areas and high cloud areas
# low cloud areas
plt.subplot(grid[1:2, 0:1])
plt.contourf(lowcloud_area['toa_sw_all_mon'],cmap='Reds')
plt.colorbar()
plt.clim(0,400)
plt.title('Shortwave of lowcloud area')

#high cloud areas
plt.subplot(grid[1:2, 1:2])
plt.contourf(highcloud_area['toa_sw_all_mon'],cmap='Reds')
plt.colorbar()
plt.clim(0,400)
plt.title('Shortwave of highcloud area')
```

output:

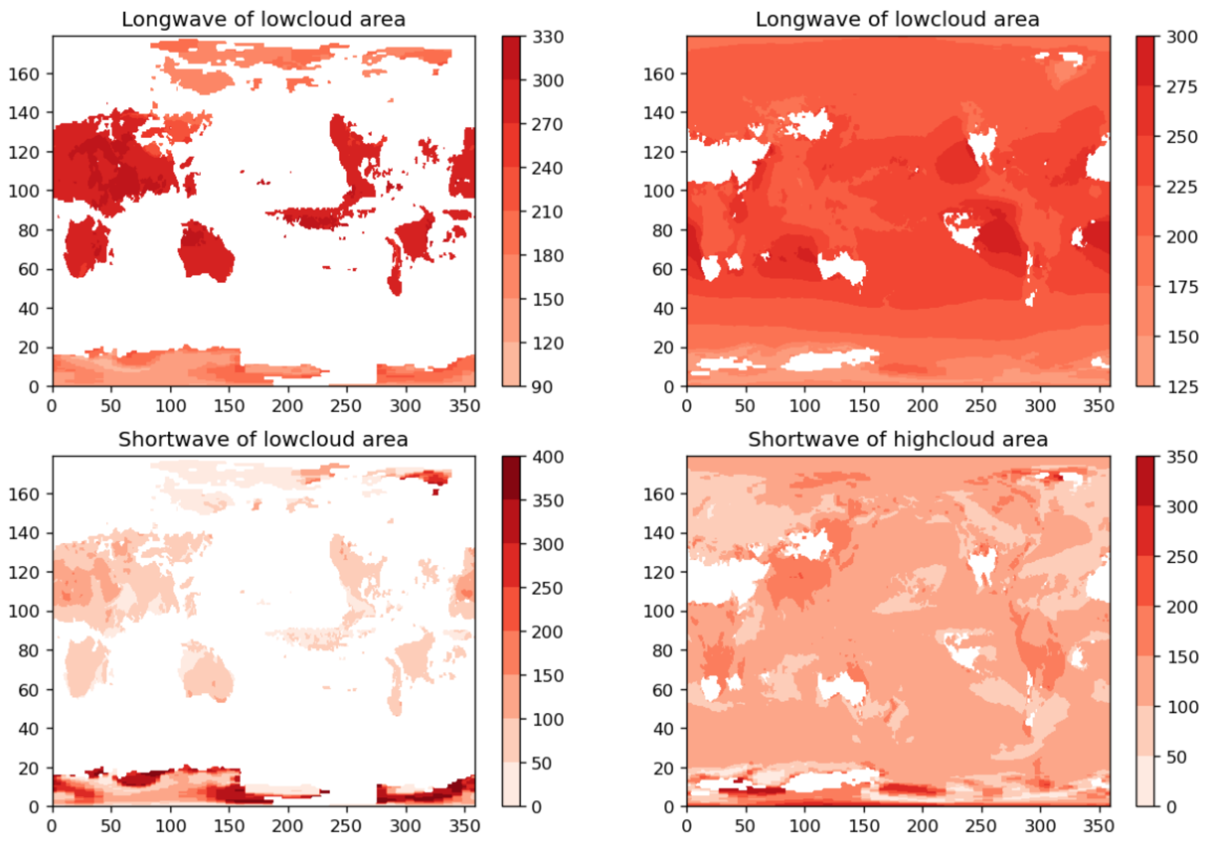


Figure 8: cloud area

2.5

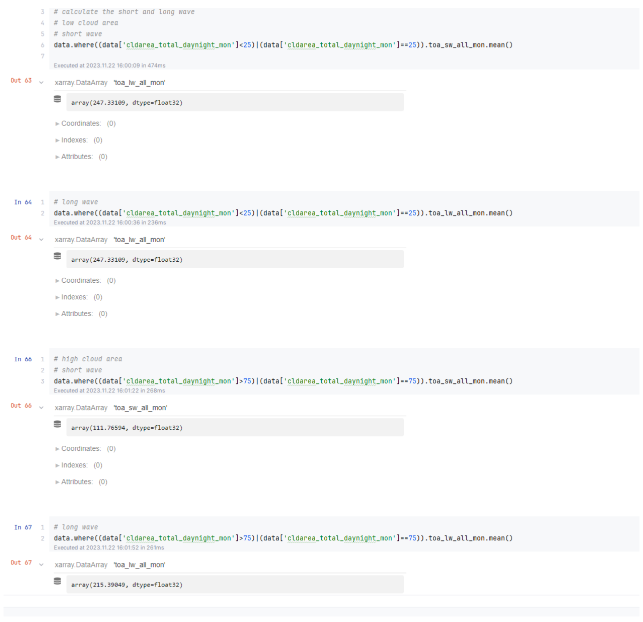
```
# 2.5

# calculate the short and long wave
# low cloud area
# short wave
data.where((data['cldarea_total_daynight_mon']<25)|
(data['cldarea_total_daynight_mon']==25)).toa_sw_all_mon.mean()

#%%
# long wave
data.where((data['cldarea_total_daynight_mon']<25)|
(data['cldarea_total_daynight_mon']==25)).toa_lw_all_mon.mean()

#%%
# high cloud area
# short wave
data.where((data['cldarea_total_daynight_mon']>75)|
(data['cldarea_total_daynight_mon']==75)).toa_sw_all_mon.mean()

#%%
# long wave
data.where((data['cldarea_total_daynight_mon']>75)|
(data['cldarea_total_daynight_mon']==75)).toa_lw_all_mon.mean()
```

The result is:
for low cloud area:
shortwave = 97
longwave = 247

for high cloud area
shortwave = 111
longwave = 215

From the result, I think the clouds can absorb the longwave radiation and reflect the short wave radiation

output:



```
# calculate the short and long wave
# low cloud area
# short wave
data.where((data['cldarea_total_daynight_mon']<25)|(data['cldarea_total_daynight_mon']==25)).toa_sw_all_mon.mean()

Executed at 2023.11.22 16:00:09 in 474ms
Out 63  xarray.DataArray 'toa_lw_all_mon'
    array(247.33109, dtype=float32)
  ▸ Coordinates: (0)
  ▸ Indexes: (0)
  ▸ Attributes: (0)

In 64
# long wave
data.where((data['cldarea_total_daynight_mon']<25)|(data['cldarea_total_daynight_mon']==25)).toa_lw_all_mon.mean()
Executed at 2023.11.22 16:00:36 in 236ms
Out 64  xarray.DataArray 'toa_lw_all_mon'
    array(247.33109, dtype=float32)
  ▸ Coordinates: (0)
  ▸ Indexes: (0)
  ▸ Attributes: (0)

In 66
# high cloud area
# short wave
data.where((data['cldarea_total_daynight_mon']>75)|(data['cldarea_total_daynight_mon']==75)).toa_sw_all_mon.mean()
Executed at 2023.11.22 16:01:22 in 268ms
Out 66  xarray.DataArray 'toa_sw_all_mon'
    array(111.76594, dtype=float32)
  ▸ Coordinates: (0)
  ▸ Indexes: (0)
  ▸ Attributes: (0)

In 67
# long wave
data.where((data['cldarea_total_daynight_mon']>75)|(data['cldarea_total_daynight_mon']==75)).toa_lw_all_mon.mean()
Executed at 2023.11.22 16:01:52 in 261ms
Out 67  xarray.DataArray 'toa_lw_all_mon'
    array(215.39049, dtype=float32)
```

Figure 9: result

Problem3

3.1

```
# problem3
#%%
# 3.1
# import data
data = xr.open_dataset('sst.mnmean.v4.nc',engine='netcdf4')

#check data
data
#%%
# calculate the mean temperature
mean_time = data.sst.mean(dim=['lat', 'lon'])
# check the result
mean_time
#%%
# transform to dataframe
meandf = mean_time.to_dataframe().reset_index()
# check result
meandf
#%%
# apply hp filter to remove the seasonal cycle, for monthly data ,lamb = 14400,trend is the longterm-t
# to remove the seasonal cycle  , inspired by my fellow WeifengSu
cycle, trend = sm.tsa.filters.hpfilter(meandf['sst'],lamb = 14400)

# polt
plt.figure(figsize=(10,5),dpi=120)
plt.plot(meandf['time'],trend)
plt.title('Change trend of Sea surface')
plt.tick_params(axis='y',left=True,direction='in',which='both')
plt.xlabel('Time(year)')
plt.ylabel('Temperature(℃)')
plt.grid()
```

output:

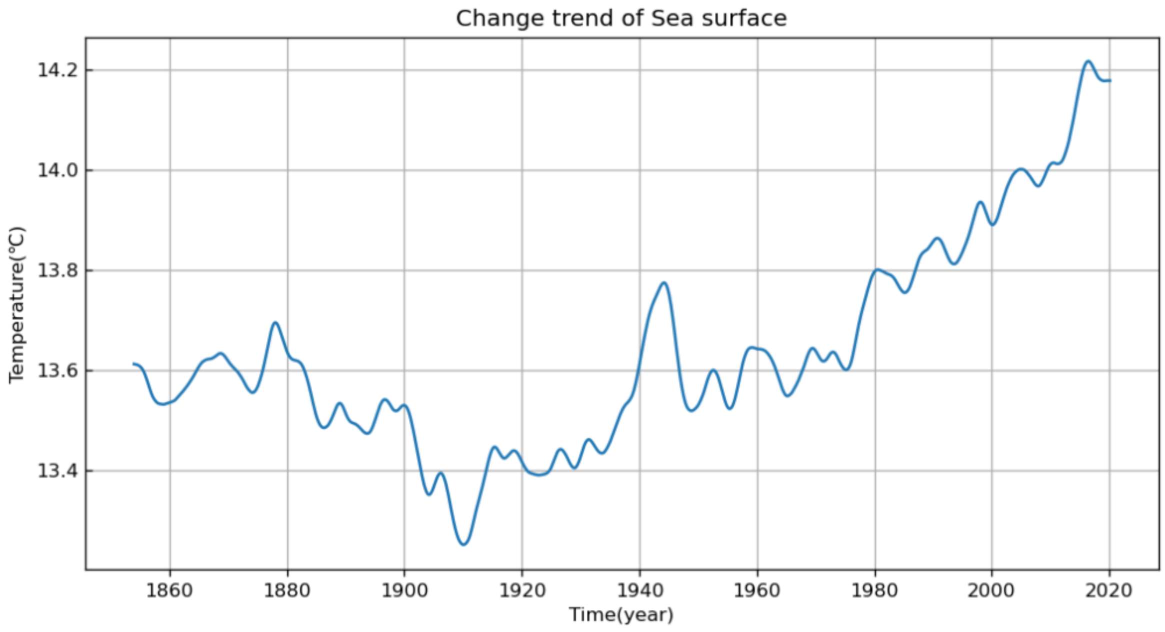


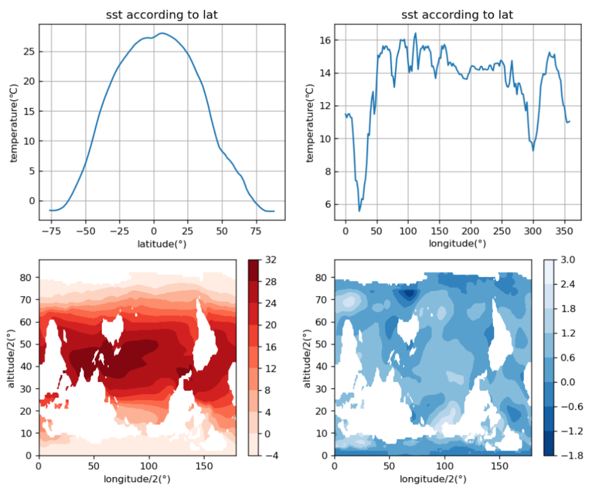
Figure 10: Change trend of Sea surface

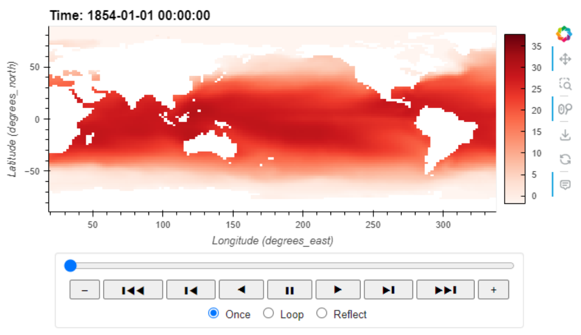
3.2

```
#3.2
#check the data
data
#%%
mean_lat = data.mean(dim='time').mean(dim='lon')
mean_lon = data.mean(dim='time').mean(dim='lat')
mean_all = data.mean(dim='time')
data1 = data.where(data['time'].dt.year==2019).mean(dim='time') - mean_all
#%%
plt.figure(figsize=(10,8),dpi=120)

# plot
# plot the sst according to lat
plt.subplot(2,2,1)
plt.plot(mean_lat['lat'],mean_lat['sst'])
plt.title('sst according to lat')
plt.xlabel('latitude(°)')
plt.ylabel('temperature(℃)')
plt.tick_params(axis='y',left=True,direction='in',which='both')
plt.grid()

# plot the sst according to lon
plt.subplot(2,2,2)
plt.plot(mean_lon['lon'],mean_lon['sst'])
plt.title('sst according to lat')
plt.xlabel('longitude(°)')
plt.ylabel('temperature(℃)')
plt.tick_params(axis='y',left=True,direction='in',which='both')
plt.grid()

# plot the 2d time -mean surface
plt.subplot(2,2,3)
plt.contourf(mean_all['sst'],cmap='Reds')
plt.xlabel('longitude/2(°)')
plt.ylabel('altitude/2(°)')
plt.colorbar()

# plot the change of sst between 2019 and the mean_surface
plt.subplot(2,2,4)
plt.contourf(data1['sst'],cmap='Blues_r')
plt.xlabel('longitude/2(°)')
plt.ylabel('altitude/2(°)')
plt.colorbar()
#%%
# hvplot
data.sst.hvplot(groupby='time',clim=(data.sst.min(), data.sst.max()),cmap='Reds',widget_type='scrubber
```

output:

```
<matplotlib.colorbar.Colorbar at 0x17329204350>
```


sst according to lat
temperature(°C)
latitude(°)
sst according to lat
temperature(°C)
longitude(°)
altitude/2(°)
longitude/2(°)
altitude/2(°)
longitude/2(°)


Time: 1854-01-01 00:00:00
Latitude (degrees_north)
Longitude (degrees_east)
Once
Loop
Reflect