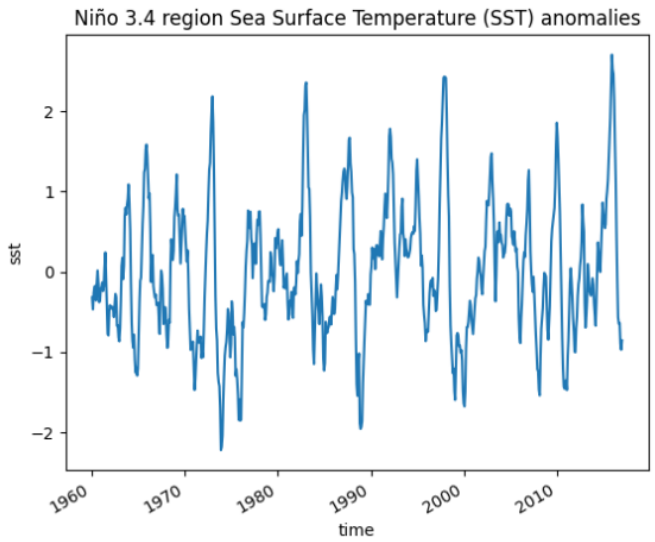
REPORT

1、

1.1、提取数据后，找到 Niño 3.4 所处位置，由于经度范围为 0-360，所以经度需要在基础上加上 360，计算气候根据时间的平均值，月度气候减去气候平均值得出异常值，再求出纬度经度的平均画出时间序列图

```
SST_nino34 = ds['sst'].sel(lon=slice(-170 + 360, -120 + 360), lat=slice(-5, 5))
# 计算该区域的月度气候学
climate = SST_nino34.groupby('time.month').mean('time')
# 计算异常值
anomalies = SST_nino34.groupby('time.month') - climate
# 可视化异常值
anomalies.mean(['lon', 'lat']).plot()  # 对经纬度取平均，只显示时间序列
plt.title('Niño 3.4 region Sea Surface Temperature (SST) anomalies')
plt.show()
在 2023.11.22 14:40:43 于 149ms内执行
```



1.2、使用 rolling 方法计算了海温数据的 3 个月滑动平均。time=3 表示使用 3 个月的窗口进行滑动平均，center=True 表示把计算的结果放在窗口的中间位置。再计算在 Niño 3.4 区域内（5N-5S，120-170W）的海温异常的时间平均值。dim=['lat', 'lon'] 表示对经度和纬度取平均值。

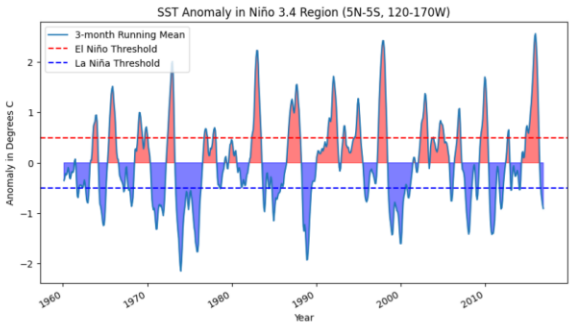
plt.axhline(0.5, color='red', linestyle='--', label='El Niño Threshold'): 添加水平线表示 El Niño 的阈值，使用红色虚线，并在图例中标记。0 以下的以此类推。

plt.fill_between(...): 使用 fill_between 方法填充图表的区域，用红色表示 El Niño，用蓝色表示 La Niña。

（代码和结果图如下）

```python
# 应用3个月滑动平均
anomalies_rolling = anomalies.rolling(time=3, center=True).mean()
# 计算异常值的时间平均值
time_avg_anomalies = anomalies_rolling.mean(dim=['lat', 'lon'])
# 绘制图表
plt.figure(figsize=(10, 5))
time_avg_anomalies.plot(label='3-month Running Mean')
plt.axhline(0.5, color='red', linestyle='--', label='El Niño Threshold')
plt.axhline(-0.5, color='blue', linestyle='--', label='La Niña Threshold')
plt.fill_between(time_avg_anomalies.time.values, 0, time_avg_anomalies, where=time_avg_anomalies > 0, color='red',
                 alpha=0.5)
plt.fill_between(time_avg_anomalies.time.values, 0, time_avg_anomalies, where=time_avg_anomalies < 0, color='blue',
                 alpha=0.5)
plt.title('SST Anomaly in Niño 3.4 Region (5N-5S, 120-170W)')
plt.xlabel('Year')
plt.ylabel('Anomaly in Degrees C')
plt.legend()
plt.show()

```

在 2023.11.22 14:40:43 于 176ms内执行



2、

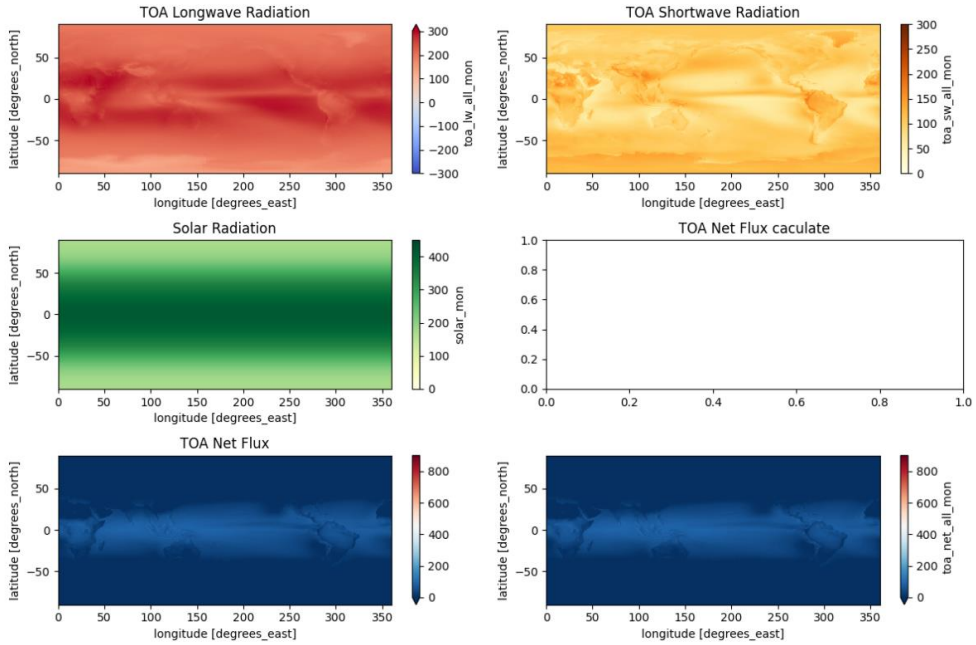
2.1

将长波、短波、太阳辐射根据时间来求平均，再用太阳辐射减去短波和长波辐射的到净辐射与测量出来的净辐射相比较。

```python
# 计算 TOA 长波、短波和太阳辐射的时间平均
toa_longwave_mean = ds1['toa_lw_all_mon'].mean(dim='time')
toa_shortwave_mean = ds1['toa_sw_all_mon'].mean(dim='time')
solar_radiation_mean = ds1['solar_mon'].mean(dim='time')
toa_net_all_mon_mean = ds1['toa_net_all_mon'].mean(dim='time')
# 计算 TOA 净辐射
toa_net_flux =  solar_radiation_mean-toa_longwave_mean -toa_shortwave_mean
# 绘制 2D 图，显示 TOA 长波、短波、太阳辐射和 TOA 净辐射
fig, axs = plt.subplots(3, 2, figsize=(12, 8))
# 绘制 TOA 长波辐射
toa_longwave_mean.plot(ax=axs[0, 0], cmap='coolwarm', vmin=-300, vmax=300)
axs[0, 0].set_title('TOA Longwave Radiation')
# 绘制 TOA 短波辐射
toa_shortwave_mean.plot(ax=axs[0, 1], cmap='YlOrBr', vmin=0, vmax=300)
axs[0, 1].set_title('TOA Shortwave Radiation')
# 绘制太阳辐射
solar_radiation_mean.plot(ax=axs[1, 0], cmap='YlGn', vmin=0, vmax=450)
axs[1, 0].set_title('Solar Radiation')
# 绘制计算出的 TOA 净辐射
toa_net_flux.plot(ax=axs[2, 0], cmap='RdBu_r', vmin=0, vmax=900)
axs[1, 1].set_title('TOA Net Flux caculate')
# 绘制 TOA 净辐射
toa_net_all_mon_mean.plot(ax=axs[2, 1], cmap='RdBu_r', vmin=0, vmax=900)
axs[2, 0].set_title('TOA Net Flux')
#可以看出，结果是一样的
plt.tight_layout()
plt.show()
```

在 2023.11.22 15:38:15 于 906ms内执行

得出结果



2.2

使用纬度（'lat'）和经度（'lon'）信息计算每个网格单元的面积。是通过考虑纬度的余弦值以及纬度和经度的差异来完成的，再通过单位面积辐射量×网格面积得出每个网格的辐射量，再对经纬度进行加和得出太阳辐射、长波、短波辐射的总量

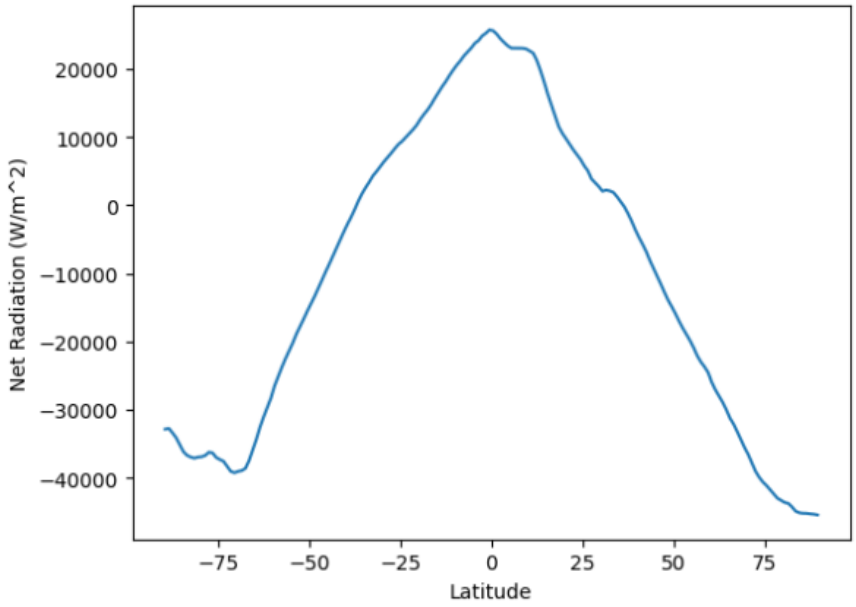
```
# 计算每个网格的面积，假设 'lat' 和 'lon' 是纬度和经度的坐标
grid_area = (
    np.cos(np.radians(ds1['lat'])) *
    np.radians(ds1['lat']).diff('lat') *
    np.radians(ds1['lon']).diff('lon')
)

# 计算 TOA 入射太阳辐射、出射长波辐射和出射短波辐射的全球总量
toa_incoming_solar_global = (ds1['solar_mon'] * grid_area).sum(dim=('lat', 'lon'))
toa_outgoing_longwave_global = (ds1['toa_lw_all_mon'] * grid_area).sum(dim=('lat', 'lon'))
toa_outgoing_shortwave_global = (ds1['toa_sw_all_mon'] * grid_area).sum(dim=('lat', 'lon'))

# 打印全球总量
print("TOA 入射太阳辐射（全球）:", toa_incoming_solar_global.values)
print("TOA 出射长波辐射（全球）:", toa_outgoing_longwave_global.values)
print("TOA 出射短波辐射（全球）:", toa_outgoing_shortwave_global.values)
```

2.3

计算每个纬度的净辐射，首先对时间求平均，再对经度求和，即可得出每个维度的辐射量

```python
# 选择 'toa_net_all_mon' 变量并按时间求平均
net_radiation = ds1['toa_net_all_mon'].mean(dim='time')
# 按纬度聚合数据，计算每个纬度带的净辐射总和
net_radiation_by_lat = net_radiation.sum(dim='lon')
# 绘图
net_radiation_by_lat.plot()
# 设置图表标题和标签
plt.xlabel('Latitude')
plt.ylabel('Net Radiation (W/m^2)')
plt.show()
```

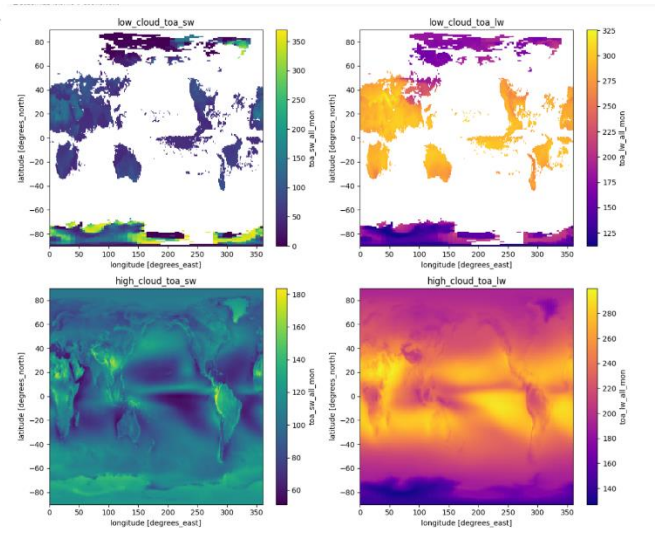
在 2023.11.22 15:50:05 于 107ms内执行



2.4、

首先对低云区高云区进行筛选，筛选完了以后对时间求平均，通过 subplots 对整个画板进行设置，通过 ax 确定图画位置

```python
# 定义低云和高云的阈值
low_cloud_threshold = 25
high_cloud_threshold = 7
# 根据云覆盖率定义低云和高云区域
low_cloud_area = ds1['cldarea_total_daynight_mon'] < low_cloud_threshold
high_cloud_area = ds1['cldarea_total_daynight_mon'] > high_cloud_threshold
# 计算低云和高云区域的时间平均出射短波和长波辐射
shortwave_low_cloud = ds1['toa_sw_all_mon'].where(low_cloud_area).mean(dim='time')
longwave_low_cloud = ds1['toa_lw_all_mon'].where(low_cloud_area).mean(dim='time')
shortwave_high_cloud = ds1['toa_sw_all_mon'].where(high_cloud_area).mean(dim='time')
longwave_high_cloud = ds1['toa_lw_all_mon'].where(high_cloud_area).mean(dim='time')
# 绘制低云和高云区域的出射短波和长波辐射组合
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
shortwave_low_cloud.plot(ax=axes[0, 0], cmap='viridis')
axes[0, 0].set_title('low_cloud_toa_sw')
longwave_low_cloud.plot(ax=axes[0, 1], cmap='plasma')
axes[0, 1].set_title('low_cloud_toa_lw')
shortwave_high_cloud.plot(ax=axes[1, 0], cmap='viridis')
axes[1, 0].set_title('high_cloud_toa_sw')
longwave_high_cloud.plot(ax=axes[1, 1], cmap='plasma')
axes[1, 1].set_title('high_cloud_toa_lw')
plt.tight_layout()
plt.show()
```



2.5

筛选出高云低云区域，然后计算低云和高云区域的全球平均出射短波和长波辐射，将全球每个网格单元的短波辐射乘以低云区域的面积，以获取低云区域的短波辐射。再对低云区域的短波辐射在经度、纬度和时间维度上进行求和，得到全球低云区域的总短波辐射。除号后面对低云区域的面积在经度、纬度和时间维度上进行求和，得到全球低云区域的总面积。最后，通过将全球低云区域的总短波辐射除以低云区域的总面积，计算得到了全球低云区域的平均短波辐射。

结果如下

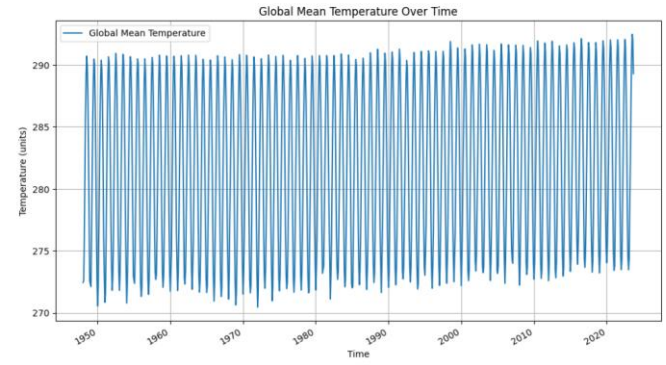
```
全球平均低云区域短波辐射： 97.11116175716484
全球平均低云区域长波辐射： 247.3310922454561
全球平均高云区域短波辐射： 111.76594124698039
全球平均高云区域长波辐射： 215.39049391670886
```

3、下载的 air.mon.mean.nc 文件

3.1

求出了月平均温度的时间序列图

```python
#月平均温度
air_T=ds2['air']
air_T
global_mean_temp=air_T.mean(dim=('lat','lon'))
plt.figure(figsize=(12,6))
global_mean_temp.plot(label="Global Mean Temperature")
global_mean_temp
plt.title("Global Mean Temperature Over Time")
plt.xlabel("Time")
plt.ylabel("Temperature (units)")  # 请替换为实际的温度单位
plt.legend()
plt.grid(True)
plt.show()
```

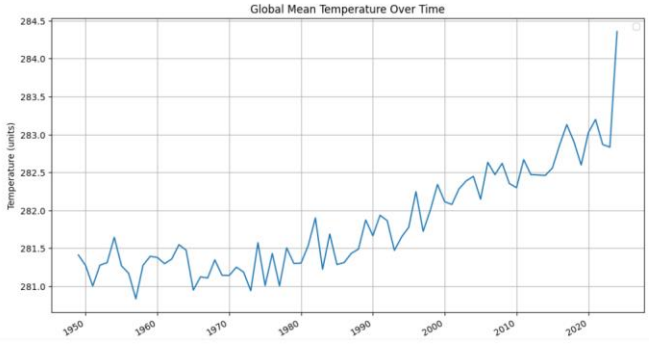


3.2

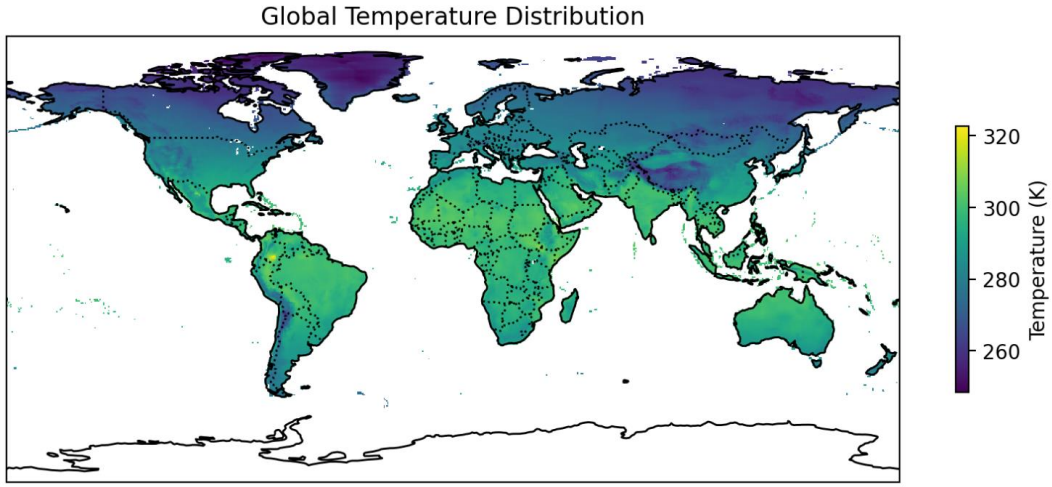
求出了年平均温度的时间序列图

```python
#年平均温度
annual_mean_temperature =air_T.resample(time='Y').mean(dim=('time'))
global_mean_temp_Y=annual_mean_temperature.mean(dim=('lat','lon'))
print(global_mean_temp_Y)
plt.figure(figsize=(12, 6))
global_mean_temp_Y.plot()
plt.title("Global Mean Temperature Over Time")
plt.xlabel("Time")
plt.ylabel("Temperature (units)")  # 请替换为实际的温度单位
plt.legend()
plt.grid(True)
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscor
<xarray.DataArray 'air' (time: 76)>\narray([281.41174, 281.2741 , 281.00327, 281.2777 , 281.31116, 281
```

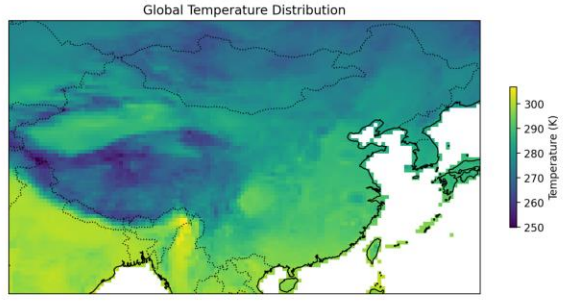


全世界的平均温度分布图

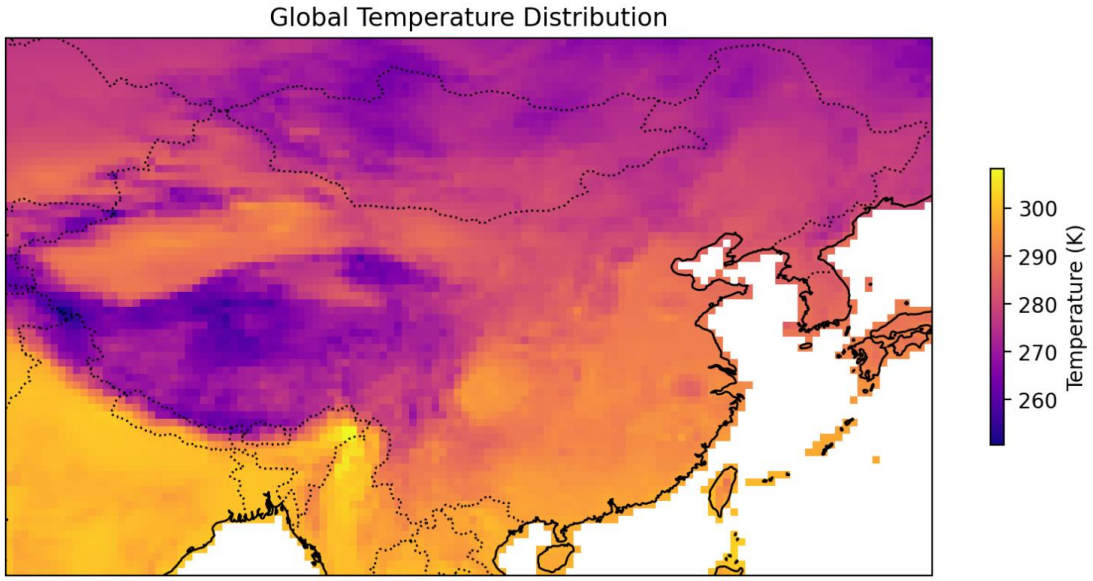


中国的平均温度分布图

```python
air_T_1=ds2['air']
temperature_china = air_T_1.sel(lon=slice(73, 135), lat=slice(54, 18))
temperature_china# 查看中国的子集温度数据
plt.figure(figsize=(10, 6),dpi=200)
ax = plt.axes(projection=ccrs.PlateCarree())
temperature_china.mean(dim='time').plot.pcolormesh(ax=ax, transform=ccrs.PlateCarree(), cmap='viridis', cbar_kwargs={'label': 'Temperature (K)','shrink':0.4})
ax.coastlines()
ax.add_feature(ccrs.cartopy.feature.BORDERS, linestyle=':')
plt.title('Global Temperature Distribution')
plt.show()
```



中国 2022 年的平均温度分布图



中国 1949 年的平均温度分布图