Assignment 03

YuGuobin　　12332284

1. Niño 3.4 index

1.1. [10 points] Compute monthly climatology for SST from Niño 3.4 region, and subtract climatology from SST time series to obtain anomalies

a) First, read the nc file:

```
In [2]: ds= xr.open_dataset("NOAA_NCDC_ERSST_v3b_SST.nc")
```

```
In [3]: ds
Out[3]:
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 ... 354.0 356.0 358.0
    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
```

b) Select the areas of the South American coast ((5N-5S, 170W-120W).) and group them by month:

```
Sac= ds.sst.sel(lon=slice(190, 240), lat=slice(-5, 5)).groupby('time.month')
Sac
```

```
DataArrayGroupBy, grouped over 'month'
12 groups with labels 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.
```

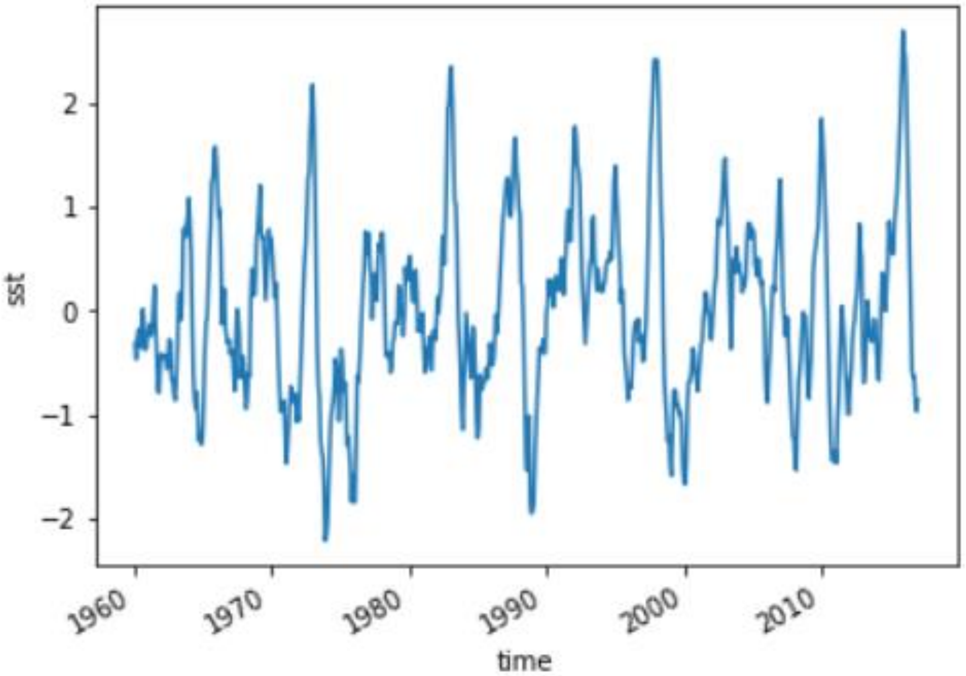
c) The monthly mean of SST is calculated first to obtain the monthly climatology, and then subtract climatology from SST time series to obtain anomalies:

```
Nino = Sac - Sac.mean(dim='time')
NinoMean = Nino.mean(dim=['lat', 'lon'])
NinoMean
```

d) Finally, plot the anomalies:

```
NinoMean.plot()
```

```
[<matplotlib.lines.Line2D at 0x2146149b220>]
```



1.2. [10 points] Visualize the computed Niño 3.4. Your plot should look similar to this one.

a) Use "plt.fill_between" to fill the parts of the anomalies that are greater than 0 and less than 0 respectively. This method needs to define the values of the x and y axes as the year and the anomalies respectively, and use "where" to define the parts that are greater than 0 or less than 0:

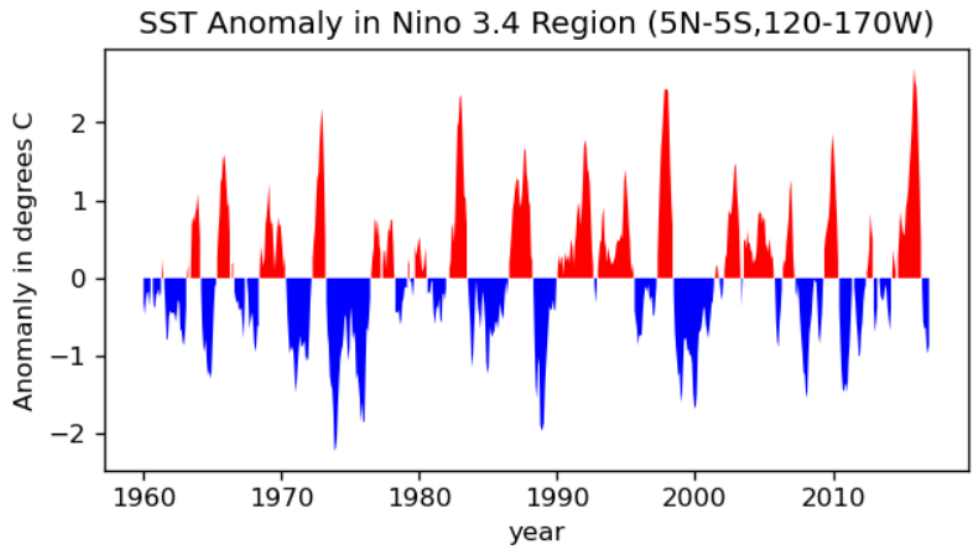
b) Finally, add the title and the x, y labels

I got inspired by reading:

https://blog.csdn.net/HHG20171226/article/details/101650909

```
plt.figure(figsize=(6,3), dpi=120)
plt.fill_between(NinoMean.time,NinoMean.data,where=(NinoMean>0),facecolor = 'red')
plt.fill_between(NinoMean.time,NinoMean.data,where=(NinoMean<0),facecolor = 'blue')
plt.title("SST Anomaly in Nino 3.4 Region (5N-5S,120-170W)")
plt.ylabel("Anomanly in degrees C")
plt.xlabel("year")
```

```
Text(0.5, 0, 'year')
```



2. Earth's energy budget

2.1. [5 points] Make a 2D plot of the time-mean TOA longwave, shortwave, and solar radiation for all-sky conditions. Add up the three variables above and verify (visually) that they are equivalent to the TOA net flux.

a) First read the nc file.

```
ds = xr.open_dataset("CERES_EBAF-TOA_200003-201701.nc")
ds
```

b) Select the TOA longwave, shortwave, and solar radiation for all-sky conditions from the dataset , and then calculate the average value is in time dimension:

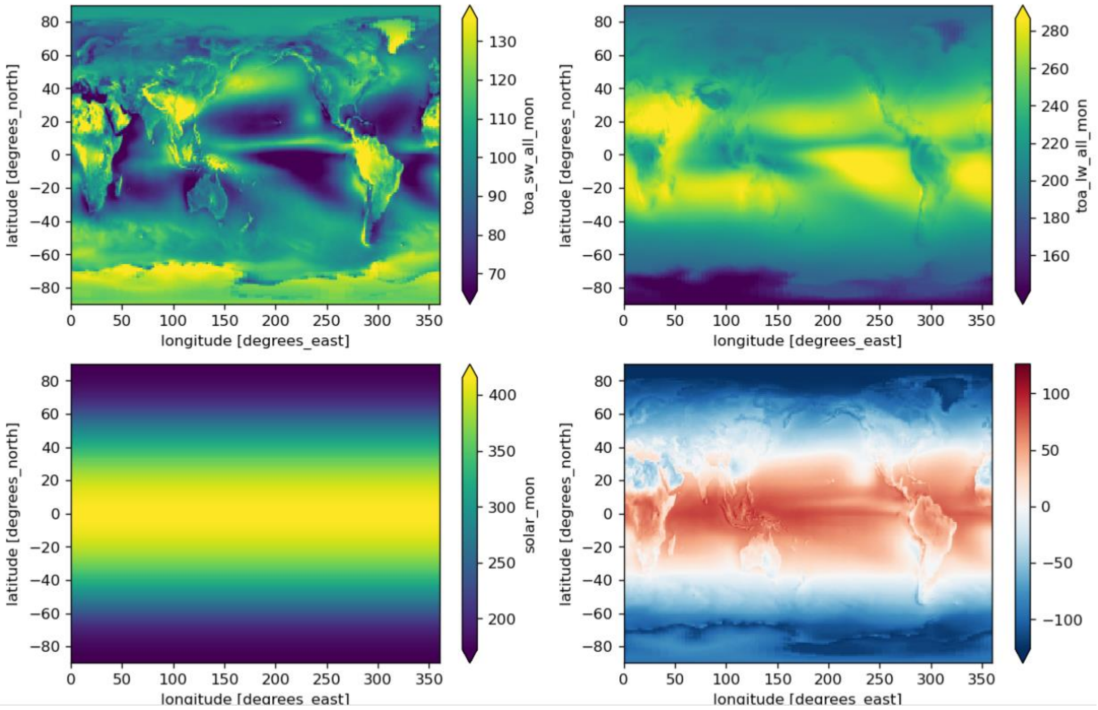
```
toa_sw_all_mean = ds.toa_sw_all_mon.mean(dim = 'time')
toa_lw_all_mean = ds.toa_lw_all_mon.mean(dim = 'time')
solar_mean = ds.solar_mon.mean(dim = 'time')
```

c) Subtract TOA longwave and shortwave from solar radiation to get TOA net flux

```
toa_net_flux_add = solar_mean-toa_sw_all_mean-toa_lw_all_mean
```

d) Define a 2*2 graph to draw the 4 graphs separately:

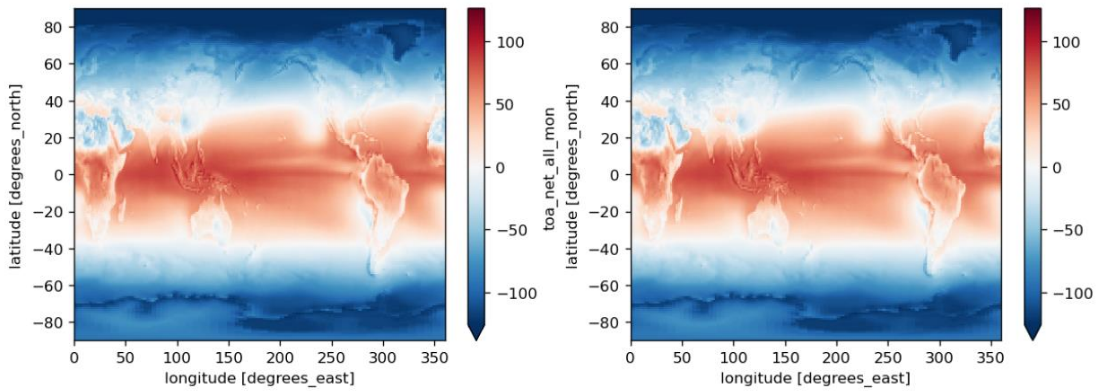
```
plt.figure(figsize=(12,8), dpi=120)
plt.subplot(2,2,1)
toa_sw_all_mean.plot(robust=True)
plt.subplot(2,2,2)
toa_lw_all_mean.plot(robust=True)
plt.subplot(2,2,3)
solar_mean.plot(robust=True)
plt.subplot(2,2,4)
toa_net_flux_add.plot(robust=True)
```



e) Create a 2*1 graph. In the left graph, select the TOA net flux variable in the data set and draw it; in the right graph, select the TOA net flux just calculated to draw it: Comparing two pictures can know they are equal:

```
toa_net_flux_mean = ds.toa_net_all_mon.mean(dim = 'time')
plt.figure(figsize=(12,4), dpi=120)
plt.subplot(1,2,1)
toa_net_flux_mean.plot(robust=True)
plt.subplot(1,2,2)
toa_net_flux_add.plot(robust=True)
```

<matplotlib.collections.QuadMesh at 0x2120cd4fc40>



2.2. **[10 points] Calculate and verify that the TOA incoming solar, outgoing longwave, and outgoing shortwave approximately match up with the cartoon above.**

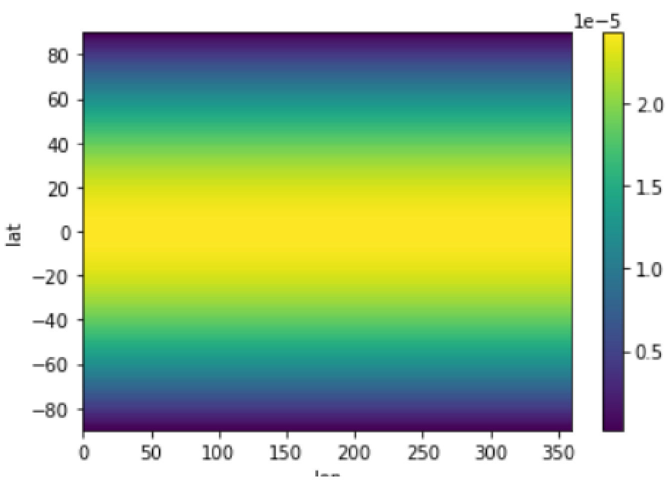
a) First, we need to calculate the area weight coefficient of each grid in the map. Here I import a package named "area", which contains a function "get_area_from_dataset" that directly returns the area of each grid of the input dataArray. The area of each grid is then divided by the total area to obtain the weight coefficient. 【**Note**: In order to import the ".ipynb" package, you need to install and import the "import_ipynb" package】

```
import import_ipynb
import area
```

```
importing Jupyter notebook from area.ipynb
```

```
grid_area = area.get_area_from_dataset(toa_net_flux_mean)
grid_area_weight = grid_area/grid_area.sum()
grid_area_weight.plot()
```

```
<matplotlib.collections.QuadMesh at 0x2146246a670>
```



I got inspired by reading:

https://seaflux.readthedocs.io/en/stable/_modules/pyseaflux/area.html

b) First, the average of TOA incoming solar over 10 years is calculated, then multiplied by the weight coefficient of the grid area, and finally summed over all the grids. The TOA incoming solar is 340.3, which approximately matches up with the cartoon's 340.4:

```
TOA_incoming_solar = ds.solar_mon.sel(time=slice("2000-01-01", "2010-12-31")).mean(dim = 'time')
TOA_incoming_solar_wMean = (TOA_incoming_solar*grid_area_weight).sum()
TOA_incoming_solar_wMean
```

```
xarray.DataArray

array(340.30208353)
▸ Coordinates: (0)
▸ Indexes: (0)
▸ Attributes: (0)
```

c) Using a similar method, the outgoing longwave value is calculated to be 240.3, which is approximately the value of 239.9 in the cartoon:

```
outgoing_longwave = ds.toa_lw_all_mon.sel(time=slice("2000-01-01", "2010-12-31")).mean(dim = 'time')
outgoing_longwave_wMean = (outgoing_longwave*grid_area_weight).sum()
outgoing_longwave_wMean
```

```
xarray.DataArray

array(240.29922999)
▸ Coordinates: (0)
▸ Indexes: (0)
▸ Attributes: (0)
```

d) Similarly, the outgoing shortwave value is calculated to be 99.2, which is approximately the value of 99.9 (77.0+22.9) in the cartoon:

```
outgoing_shortwave = ds.toa_sw_all_mon.sel(time=slice("2000-01-01", "2010-12-31")).mean(dim = 'time')
outgoing_shortwave_wMean = (outgoing_shortwave*grid_area_weight).sum()
outgoing_shortwave_wMean
```

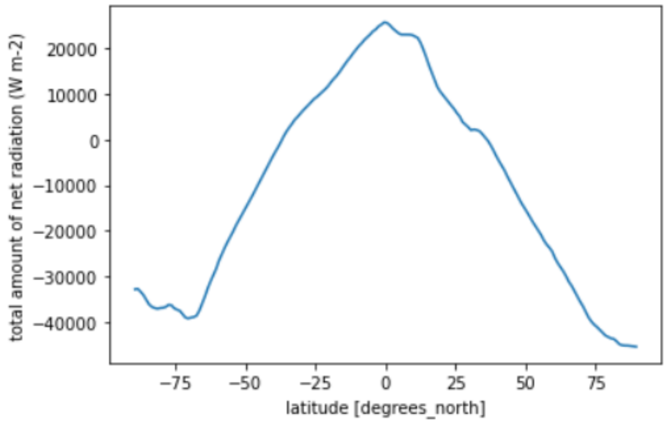
```
xarray.DataArray

array(99.24315248)
▸ Coordinates: (0)
▸ Indexes: (0)
▸ Attributes: (0)
```

2.3. [5 points] Calculate and plot the total amount of net radiation in each 1-degree latitude band. Label with correct units.

a) First select the variable "toa_net_all_mon" in the dataset, average it by “time”, and then sum it by “lon”. Then add y labels and units to the graph.

```
total_amount_of_net_radiation =ds.toa_net_all_mon.mean(dim='time').sum(dim = 'lon').plot()
plt.ylabel('total amount of net radiation (W m-2)')
```

```
Text(0, 0.5, 'total amount of net radiation (W m-2)')
```

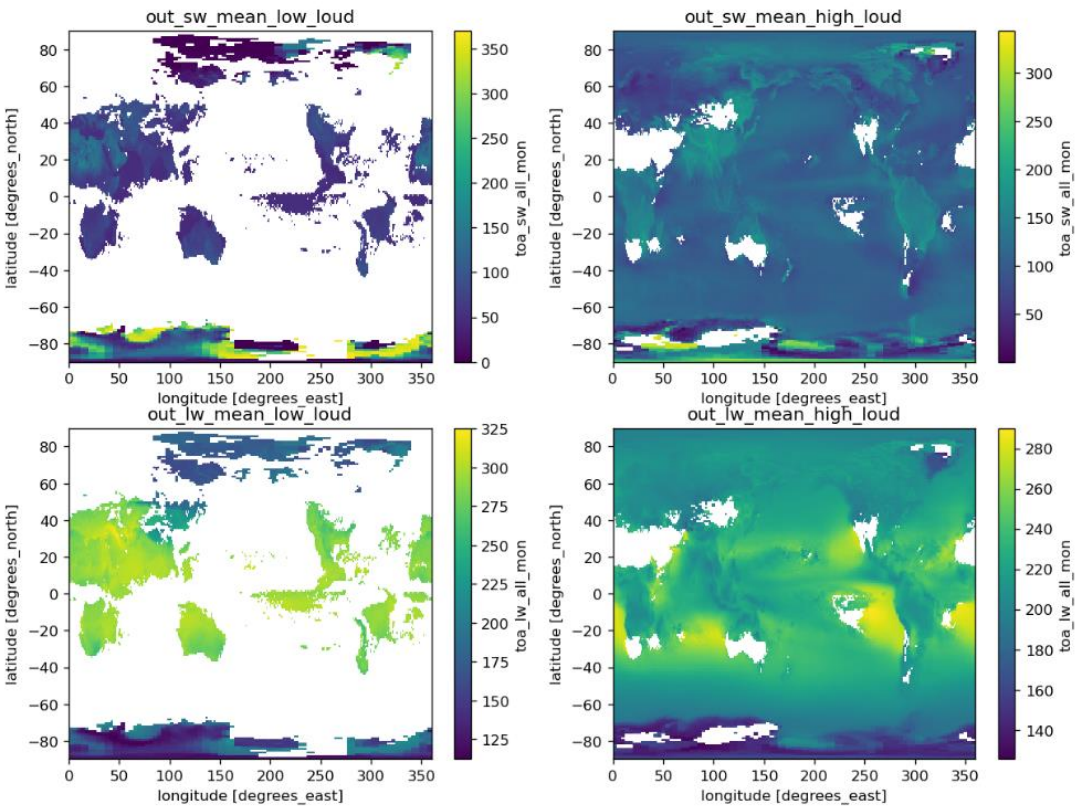


2.4. [5 points] Calculate and plot composites of time-mean outgoing shortwave and longwave radiation for low and high cloud area regions. Here we define low cloud area as ≤25% and high cloud area as ≥75%. Your results should be 2D maps.

a) For example, for outgoing shortwavelow in low cloud area regions, first use the “where” method and select variable "cldarea_total_daynight_mon" to filter the cloud amount (**≤25%**). then, average the time dimension of variable "toa_sw_all_mon.mean".

b) In a similar way, several other variables are filtered, and finally a 2*2 image is drawn, and the corresponding title is added.

```
out_sw_mean_low_loud = ds.where(ds.cldarea_total_daynight_mon<=25).toa_sw_all_mon.mean(dim = 'time')
out_sw_mean_high_loud = ds.where(ds.cldarea_total_daynight_mon>=75).toa_sw_all_mon.mean(dim = 'time')
out_lw_mean_low_loud = ds.where(ds.cldarea_total_daynight_mon<=25).toa_lw_all_mon.mean(dim = 'time')
out_lw_mean_high_loud = ds.where(ds.cldarea_total_daynight_mon>=75).toa_lw_all_mon.mean(dim = 'time')
plt.figure(figsize=(12,9), dpi=120)
plt.subplot(2, 2, 1)
out_sw_mean_low_loud.plot()
plt.title('out_sw_mean_low_loud')
plt.subplot(2, 2, 2)
out_sw_mean_high_loud.plot()
plt.title('out_sw_mean_high_loud')
plt.subplot(2, 2, 3)
out_lw_mean_low_loud.plot()
plt.title('out_lw_mean_low_loud')
plt.subplot(2, 2, 4)
out_lw_mean_high_loud.plot()
plt.title('out_lw_mean_high_loud')
```



2.5. [5 points] Calculate the global mean values of shortwave and longwave radiation, composited in high and low cloud regions. What is the overall effect of clouds on shortwave and longwave radiation?

a) Using the longwave and shortwave radiation screened in the previous question under different cloud cover conditions, multiplied by the weight coefficient of the global grid and then summed, the global weighted average can be obtained.

```
out_sw_mean_low_loud_wMean = (out_sw_mean_low_loud * grid_area_weight).sum()
out_sw_mean_high_loud_wMean = (out_sw_mean_high_loud * grid_area_weight).sum()
out_lw_mean_low_loud_wMean = (out_lw_mean_low_loud * grid_area_weight).sum()
out_lw_mean_high_loud_wMean = (out_lw_mean_high_loud * grid_area_weight).sum()
print("shortwave in low cloud regions:",out_sw_mean_low_loud_wMean)
print("shortwave in high cloud regions:",out_sw_mean_high_loud_wMean)
print("longwave in low cloud regions:",out_lw_mean_low_loud_wMean)
print("longwave in high cloud regions:",out_lw_mean_high_loud_wMean)
```

```
shortwave in low cloud regions: <xarray.DataArray ()>
array(19.65385802)
shortwave in high cloud regions: <xarray.DataArray ()>
array(104.55015333)
longwave in low cloud regions: <xarray.DataArray ()>
array(70.55932454)
longwave in high cloud regions: <xarray.DataArray ()>
array(208.55489987)
```

b) The global mean value of shortwave in low cloud regions is 19.65, in high cloud regions is 104.55; The longwave in low cloud regions is 70.56, in high cloud regions is 208.55. It can be seen that both shortwave and longwave, the radiation in the high cloud area is higher than the low cloud area, so the clouds have a positive effect on the short wave and long wave radiation.

3. Explore a data set

3.1. [5 points] Plot a time series of a certain variable with monthly seasonal cycle removed.

a) I downloaded the monthly rainfall flux data at 1°*1° resolution for 2010-2020 from the GES DISC website. Then used the program "combineNC.ipynb" to combine the 132 nc files and named it “Rainf2010_2020.nc”. Load the data as follows:

```
ds = xr.open_dataset('Rainf2010_2020.nc')
ds
```

xarray.Dataset

▸ Dimensions: (**time**: 132, bnds: 2, **lon**: 360, **lat**: 151)

▾ Coordinates:

time	(time)	datetime64[ns]	2010-01-01 ... 2020-12-01
lon	(lon)	float64	-180.0 -179.0 ... 178.0 179.0
lat	(lat)	float64	-60.0 -59.0 -58.0 ... 89.0 90.0

▾ Data variables:

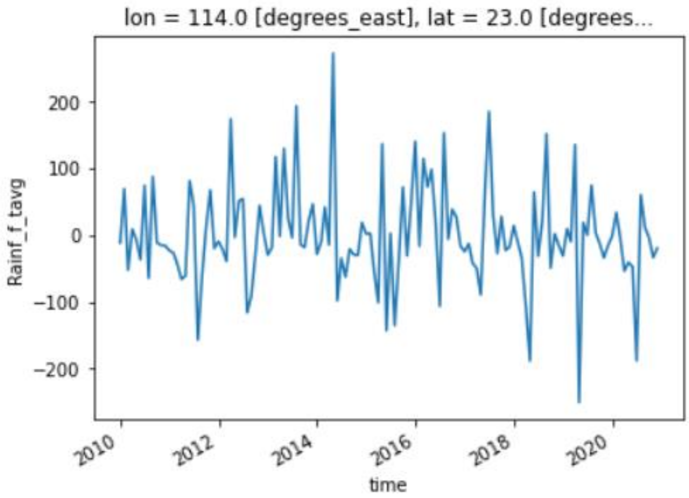
time_bnds	(time, bnds)	datetime64[ns]	...
Rainf_f_tavg	(time, lat, lon)	float32	...
standard_name :	rainfall_flux		
long_name :	rainfall flux		
units :	kg m-2 s-1		
cell_methods :	time:mean		
vmin :	-0.000118062664		
vmax :	0.00024219579		

▸ Indexes: (3)

▸ Attributes: (13)

b) Remove the seasonal cycle. Firstly, group data by month. Then, apply mean to grouped data, and then compute the anomaly, finally the longitude and latitude of Shenzhen are selected for plotting.

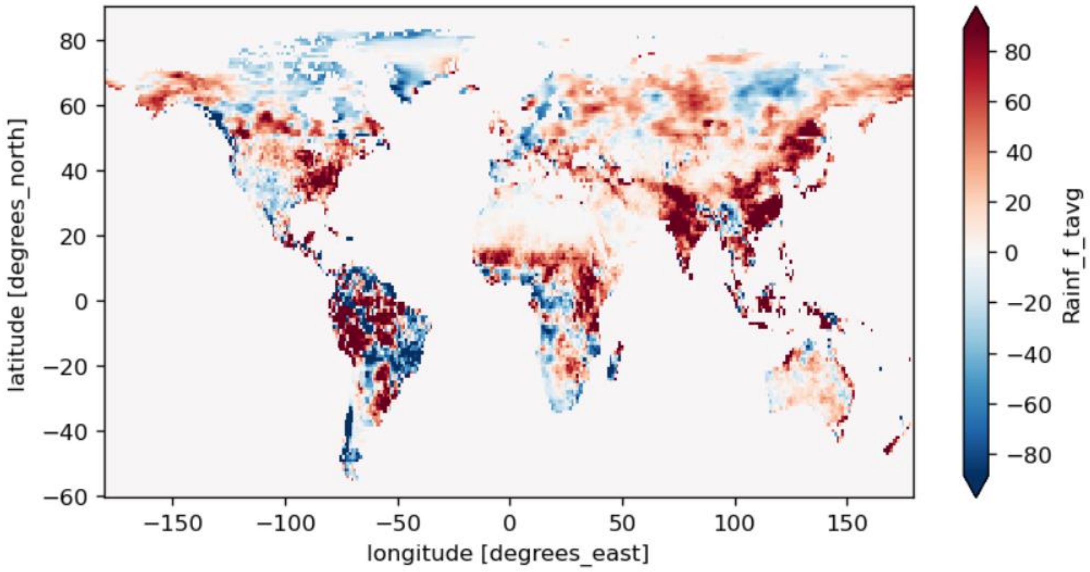
```
In  [26]: ds_rain = ds.Rainf_f_tavg*60*60*24*30
          ds_rain_group = ds_rain.groupby('time.month')
          anomalies = ds_rain_group - ds_rain_group.mean(dim = 'time')
          anomalies.sel(lon=114.1, lat=22.5, method='nearest').plot()

Out[26]: [<matplotlib.lines.Line2D at 0x1a60bd31160>]
```



3.2. [5 points] Make at least 5 different plots using the dataset.

a) Plot the global average rainfall flux over a 10-year period:

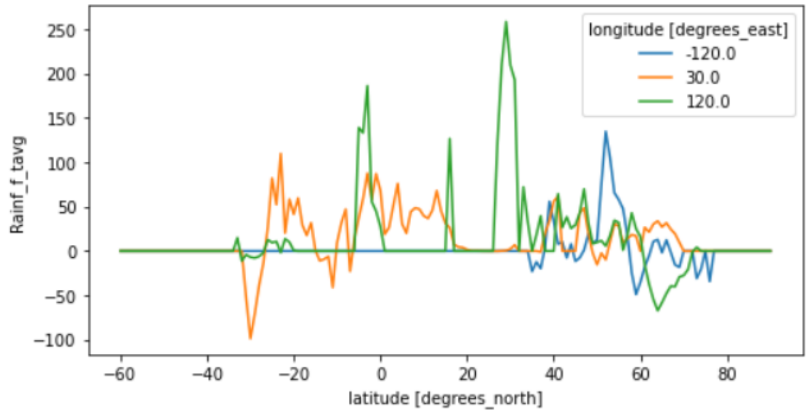
```
precip_mean = ds_rain.groupby('time.year').sum().mean('year')
plt.figure(figsize=(8,4), dpi=120)
precip_mean.plot(robust=True)

<matplotlib.collections.QuadMesh at 0x2120034b310>
```



b) Plot the time series of the global average rainfall flux at 3 longitudes:

```
precip_mean.sel(lon = [-120,30,120],method="nearest").plot(x = 'lat',hue ='lon',figsize=(8,4))
```

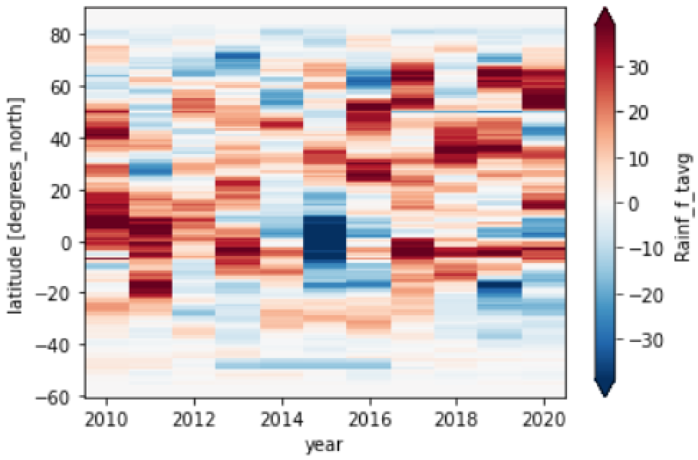
```
[<matplotlib.lines.Line2D at 0x2120040bc10>,
 <matplotlib.lines.Line2D at 0x2120040bc70>,
 <matplotlib.lines.Line2D at 0x2120040bd90>]
```



c) Plot zonal mean rainfall flux:

```
ds_rain.groupby('time.year').sum().mean(dim='lon').transpose().plot(robust = True)
```

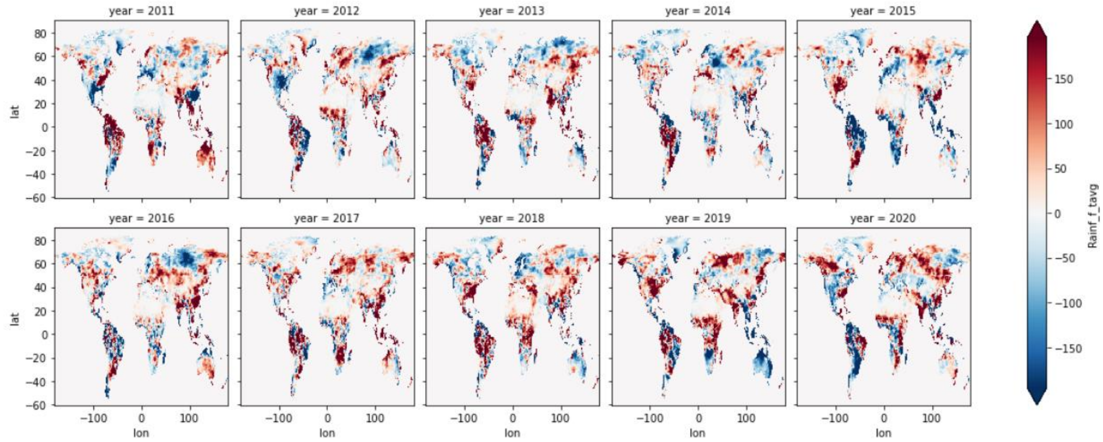
```
<matplotlib.collections.QuadMesh at 0x21200490d60>
```



d) Plot annual mean rainfall flux from 2010 to 2020:

```
ds_rain.groupby('time.year').sum().sel(year = slice("2011","2020")).plot(col = 'year',col_wrap=5, robust=True)
```

```
<xarray.plot.facetgrid.FacetGrid at 0x21200770d90>
```



e) Plot monthly mean rainfall flux over the 10 years:

```
ds_rain.groupby('time.month').mean().plot(col = 'month',col_wrap=6, robust=True)
```

```
<xarray.plot.facetgrid.FacetGrid at 0x212027c8c40>
```