PS3_2

November 10, 2021

[1]:
```
import numpy as np
import pandas as pd
import xarray as xr
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
%matplotlib inline
```

[2]:
```
ds = xr.open_dataset("CERES_EBAF-TOA_200003-201701.nc", engine="netcdf4")
ds
```

[2]:
```
<xarray.Dataset>
Dimensions:                     (lon: 360, time: 203, lat: 180)
Coordinates:
  * lon                         (lon) float32 0.5 1.5 2.5 … 357.5 358.5 359.5
  * 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: (12/14)
    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_net_mon             (time, lat, lon) float32 …
    solar_mon                   (time, lat, lon) float32 …
    cldarea_total_daynight_mon  (time, lat, lon) float32 …
    cldpress_total_daynight_mon (time, lat, lon) float32 …
    cldtemp_total_daynight_mon  (time, lat, lon) float32 …
    cldtau_total_day_mon        (time, lat, lon) float32 …
Attributes:
    title:              CERES EBAF (Energy Balanced and Filled) TOA Fluxes. Mo…
    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
```

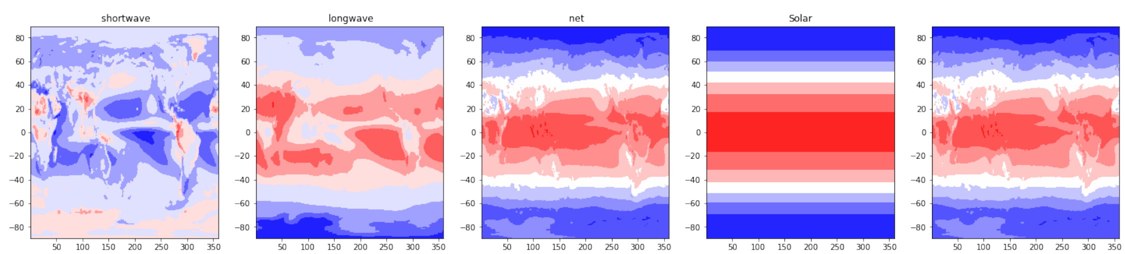
```
    DOI:                 10.5067/TERRA+AQUA/CERES/EBAF-TOA_L3B.004.0
    Production_Files:   List of files used in creating the present Master netC…
```

[3]:
```python
plt_ds=[ds.toa_sw_all_mon.mean(dim='time'),
     ds.toa_lw_all_mon.mean(dim='time'),
     ds.toa_net_all_mon.mean(dim='time'),
     ds.solar_mon.mean(dim='time'),]
labels=[' shortwave',' longwave',' net','Solar']

fig,axs=plt.subplots(1,5,figsize=(25,5))
lon=plt_ds[0].lon
lat=plt_ds[0].lat

for ds,ax,label in zip(plt_ds[:4],axs[:4],labels[:4]):
    ax.contourf(lon,lat,ds,cmap='bwr',)
    ax.set_title(label)
axs[4].contourf(lon,lat,plt_ds[3]-(plt_ds[0]+plt_ds[1]),cmap='bwr')
```

[3]:
```
<matplotlib.contour.QuadContourSet at 0x225b2369af0>
```



[4]:
```python
weights = np.cos(np.deg2rad(ds.lat))
TOA_SW_Weighted = ds.toa_sw_all_mon.weighted(weights)
TOA_SW_Weighted.mean(dim=('lat', 'lon', 'time'))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-4-72dc8bb6c42a> in <module>
      1 weights = np.cos(np.deg2rad(ds.lat))
----> 2 TOA_SW_Weighted = ds.toa_sw_all_mon.weighted(weights)
      3 TOA_SW_Weighted.mean(dim=('lat', 'lon', 'time'))

D:\tiankuan\anaconda3\lib\site-packages\xarray\core\common.py in␣
 ↪__getattr__(self, name)
    244                 with suppress(KeyError):
    245                     return source[name]
```

```
--> 246         raise AttributeError(
    247             "{!r} object has no attribute {!r}".format(type(self).
 ↪__name__, name)
    248         )

AttributeError: 'DataArray' object has no attribute 'toa_sw_all_mon'
```

[ ]:
```
TOA_LW_Weighted = ds.toa_lw_all_mon.weighted(weights)
TOA_LW_Weighted.mean(dim=('lat', 'lon', 'time'))
```

[ ]:
```
Solar_Weighted = ds.solar_mon.weighted(weights)
Solar_Weighted.mean(dim=('lat', 'lon', 'time'))
```

[ ]:
```
weights=np.cos(np.deg2rad(ds.lat))
weights.plot()
plt.show()
```

[ ]:
```
Cloud_Mean = ds.cldarea_total_daynight_mon.mean(dim='time')
Low_Cloud_Area = Cloud_Mean.where(Cloud_Mean <= 25.0)
High_Cloud_Area = Cloud_Mean.where(Cloud_Mean >= 75.0)


TOA_SW_Mean = ds.toa_sw_all_mon.mean(dim='time')
TOA_LW_Mean = ds.toa_lw_all_mon.mean(dim='time')

Low_Cloud_SW = TOA_SW_Mean * (Low_Cloud_Area / Low_Cloud_Area)
Low_Cloud_LW = TOA_LW_Mean * (Low_Cloud_Area / Low_Cloud_Area)

High_Cloud_SW = TOA_SW_Mean * (High_Cloud_Area / High_Cloud_Area)
High_Cloud_LW = TOA_LW_Mean * (High_Cloud_Area / High_Cloud_Area)
```

[ ]:
```
plt.figure(figsize=(14, 8))
ax = plt.subplot(2, 2, 1)
Low_Cloud_SW.plot(robust=True)
ax.set_title('Low Cloud Area outgoing Shortwave')

ax = plt.subplot(2, 2, 2)
Low_Cloud_LW.plot(robust=True)
ax.set_title('Low Cloud Area outgoing Longwave')

ax = plt.subplot(2, 2, 3)
High_Cloud_SW.plot(robust=True)
ax.set_title('High Cloud Area outgoing Shortwave')

ax = plt.subplot(2, 2, 4)
High_Cloud_LW.plot(robust=True)
```

```python
ax.set_title('High Cloud Area outgoing Longwave')

plt.show()
```

[ ]:
```python
Cloud_Mean = ds.cldarea_total_daynight_mon.mean(dim='time')
Low_Cloud_Area = Cloud_Mean.where(Cloud_Mean <= 25.0)
High_Cloud_Area = Cloud_Mean.where(Cloud_Mean >= 75.0)

TOA_SW_Mean = ds.toa_sw_all_mon.mean(dim='time')
TOA_LW_Mean = ds.toa_lw_all_mon.mean(dim='time')

Low_Cloud_SW = TOA_SW_Mean * (Low_Cloud_Area / Low_Cloud_Area)
Low_Cloud_SW.mean()
```

[ ]:
```python
Low_Cloud_LW = TOA_LW_Mean * (Low_Cloud_Area / Low_Cloud_Area)
Low_Cloud_LW.mean()
```

[ ]:
```python
High_Cloud_SW = TOA_SW_Mean * (High_Cloud_Area / High_Cloud_Area)
High_Cloud_SW.mean()
```

[ ]:
```python
High_Cloud_LW = TOA_LW_Mean * (High_Cloud_Area / High_Cloud_Area)
High_Cloud_LW.mean()
```