Author and Team Leader: Ozlem OZALP **Co-authors:** Antonio VECOLI and Jacopo ALESSANDRI

The Sea Surface Temperature (SST) is an important physical characteristic of the oceans and is the one of the vital component of the climate system . The Adriatic Sea is an elongated basin, located in the central Mediterranean, between the Italian peninsula and the Balkans.The CMEMS Mediterranean Sea Physics Reanalysis time series is provided since 01/01/1987. The SST is defined by selecting the first vertical level of the daily mean of Potential Temperature within the variable name “thetao”. The data is available and can be downloaded from the following link

Introduction to WEkEO Notebook

Data used

Product Description	Data Store collection ID	Product Navigator	WEkEO HDA ID	WEkEO metadata
CMEMS Reanalysis: Daily Mean Potential temperature from 1987 to 2019	MEDSEA_MULTIYEAR_PHY_006_004	link	EO:MO:DAT:MEDSEA_MULTIYEAR_PHY_006_004/med-cmcc-tem-rean-d	link

Learning outcomes

In this notebook the SST has been analysed in the Adriatic Sea from 01/01/1987 to 31/12/2019 therefore you will find;

- How to analyse the SST Time Series in the Adriatic Sea from CMEMS Mediterranean Sea Physics Reanalysis.
- How to generate plots for daily, monthly and annual trends in average temperature.
- How to visualize maps of the average seasonals and anomalies.

Contents

The CODE SECTION includes:

4. Section 5: The Seasonal Anomalies Map Visualizations.
5. Section 6: References.

Outline

The first section of the Jupyter Notebook describes how to get the data from CMEMS, then it provides the Python code for:

- Data Aggregation
- Time Series Analysis
- Data Visualization

The Code Section is based on the following elements

1. Required Python Modules

   The following libraries need to be installed:

   ```
   conda install -c conda-forge xarray dask netCDF4 -y
   conda install -c conda-forge cartopy
   pip install statsmodels
   pip install seaborn
   pip install regionmask
   pip install pygeos
   ```

2. "functions_cmems.py" called in the main programme covers:

   The file contains the calling functions to provide necessary methods for the data elaboration. Therefore, in the initial part, it contains the specific functions for the data aggregation, then continuing with time series analysis and visualization functions and finally providing routines for the calculation and visualization of seasonal anomalies. The function names are described in details in the code section step by step.

Section 1: Data Preparation

Back to top

The first part of the Notebook describes how to get the data from CMEMS. The name of the CMEMS DATA is **Mediterranean Sea Physics Reanalysis**. The dataset name is **med-cmcc-tem-rean-d** and contain 33 years daily information. Thereafter, the dataset is reseized for the Adriatic Sea:

```
Time= 1987-2019

Longitude: 12-22 E°  and  Latitude: 37-46 N°
```

Hence, the Dataset Dimension is:

```
Time: 12053
lat: 216
```

```
lon: 241
depth: 1.01823
```

And finally a unique output file "CMEMS_SST" in netCDF format is prepared for the data aggregation process and can easily downloaded from the following link. The file size is 2.34 GB. The spatial data in csv format is also present in the following link.

Note that:

The winter Season is defined from January to April, and the Summer Season is from July to October in the Adriatic Sea, REFERENCE LINK.

CODE SECTION

Section 2: Functions for the Data Aggregation process.

Importing libraries and functions

```
import xarray as xr
import pandas as pd
import numpy as np
import cartopy.feature as cfeature
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
import regionmask
import matplotlib.colors

import functions_cmems as fc

import warnings
warnings.resetwarnings()
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
/opt/conda/lib/python3.8/site-packages/geopandas/_compat.py:112:
UserWarning: The Shapely GEOS version (3.9.1-CAPI-1.14.2) is
incompatible with the GEOS version PyGEOS was compiled with (3.10.1-
CAPI-1.16.0). Conversions between both will be slow.
  warnings.warn(
```

Loading the Data

```
ncRawDataFileName =
"WEkEO_PART_1_Aggregations/WEkEO_SST_DATA/CMEMS_SST.nc"
fc.areaPerimeter =pd.read_csv
("WEkEO_PART_1_Aggregations/WEkEO_SST_DATA/areaAdriatic.csv")
dataOutput = "Aggregations/CMEMS_SST_clipped.nc"
```

"areaPerimeter" is delimiting function over the area of interest. In the spatial data, the 1st column is longitude, the 2nd column is latitude. The Clipped file is saved as "CMEMS_SST_clipped.nc".

```
rawData = xr.open_dataset(ncRawDataFileName)
clippedData = fc.ClipDataOnRegion(rawData,
fc.areaPerimeter,dataOutput)
```

```
CMEMS SST Dimension: <xarray.Dataset>
Dimensions:  (time: 12053, lat: 216, lon: 241)
Coordinates:
  * time     (time) datetime64[ns] 1987-01-01T12:00:00 ... 2019-12-
31T12:00:00
    depth    float32 ...
  * lat      (lat) float32 37.02 37.06 37.1 37.15 ... 45.85 45.9 45.94
45.98
  * lon      (lon) float32 12.0 12.04 12.08 12.12 ... 21.88 21.92
21.96 22.0
Data variables:
    thetao   (time, lat, lon) float32 ...
Attributes:
    Conventions:  CF-1.8
Clipped Area Dimensions:            LON        LAT
0    19.641391  39.744436
1    18.375273  39.798191
2    18.391109  39.816245
3    18.399336  39.899609
4    18.399436  39.936100
..         ...        ...
413  19.694582  39.794718
414  19.673055  39.793055
415  19.650836  39.772500
416  19.640000  39.756664
417  19.641391  39.744436

[418 rows x 2 columns]
Reseized Area: <xarray.Dataset>
Dimensions:  (time: 12053, lat: 146, lon: 188)
Coordinates:
  * time     (time) datetime64[ns] 1987-01-01T12:00:00 ... 2019-12-
31T12:00:00
    depth    float32 ...
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
Data variables:
    thetao   (time, lat, lon) float32 ...
Attributes:
    Conventions:  CF-1.8
saving to  Aggregations/CMEMS_SST_clipped.nc
finished saving
```

```
ncRawDataFileName_annual_mean = "Aggregations/CMEMS_SST_clipped.nc"
annualMapsNcFile = "Aggregations/CMEMS_SST_clipped_Annual_Mean.nc"

rawData_annual_mean = xr.open_dataset(ncRawDataFileName_annual_mean)
```

The clipped file has been indexed in "months" through the XARRAY library ,then averaged by year and finally saved as "CMEMS_SST_clipped_Annual_Mean.nc" by using the **GenerateAnnualMeanMaps** function

- am1 = t.sel(time=AM(t['time.month']))
- am2 = am1.groupby('time.year').mean('time')

```
clippedData_annual_mean = 
fc.GenerateAnnualMeanMaps(rawData_annual_mean, annualMapsNcFile)

ANNUAL MEAN for 33 years: <xarray.Dataset>
Dimensions:  (lat: 146, lon: 188, year: 33)
Coordinates:
    depth    float32 1.018
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69 
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88 
19.92 19.96
  * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017 
2018 2019
Data variables:
    thetao   (year, lat, lon) float32 19.11 19.11 19.12 19.13 ... nan 
nan nan
ANNUAL MEAN for 33 years: <xarray.Dataset>
Dimensions:  (lat: 146, lon: 188, year: 33)
Coordinates:
    depth    float32 1.018
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69 
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88 
19.92 19.96
  * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017 
2018 2019
Data variables:
    thetao   (year, lat, lon) float32 19.11 19.11 19.12 19.13 ... nan 
nan nan
Annual Mean minimum T: <xarray.DataArray 'thetao' ()>
array(15.038228, dtype=float32)
Coordinates:
    depth    float32 1.018
Annual Mean maximum T: <xarray.DataArray 'thetao' ()>
array(20.717798, dtype=float32)
Coordinates:
    depth    float32 1.018
saving to  Aggregations/CMEMS_SST_clipped_Annual_Mean.nc
finished saving
```

The Mean Seasonal Maps have been generated with dimensions (TIME, LAT and LON) from "CMEMS_SST_clipped.nc" file.

The new aggregated files have been saved as:

1. "CMEMS_SST_WINTER_SEASON.nc" and is defined between January and April by using **"GenerateSeasonalWinter"** function:

```
    def WINTER(month):

      return (month >= 1) & (month <= 4)

      seasonal_data_winter = t.sel(time=WINTER(t['time.month']))
```

2. "CMEMS_SST_SUMMER_SEASON.nc" through **"GenerateSeasonalSummer"** function between July and October:

```
    def SUMMER(month):

      return (month >= 7) & (month <= 10)

      seasonal_data_summer = t.sel(time=SUMMER(t['time.month']))
```

The Seasonal means have been calculated from each saved file:

```
   seasonal_data_winter1 =
seasonal_data_winter.groupby('time.year').mean()
   seasonal_data_summer1 =
seasonal_data_summer.groupby('time.year').mean()
```

The maximum and minimum Temperatures by Season are also printed:

```
   print("",seasonal_data_winter.thetao.min())
   print("",seasonal_data_winter.thetao.max())
   print("",seasonal_data_summer.thetao.min())
   print("",seasonal_data_summer.thetao.max())

winter_output= "Aggregations/CMEMS_SST_WINTER_SEASON.nc"
summer_output="Aggregations/CMEMS_SST_SUMMER_SEASON.nc"

SeasonWinter =
fc.GenerateSeasonalWinter(rawData_annual_mean,winter_output)

Reseized Area: <xarray.Dataset>
Dimensions:  (lat: 146, lon: 188, year: 33)
Coordinates:
    depth    float32 1.018
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
  * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017
2018 2019
```

```
Data variables:
    thetao   (year, lat, lon) float32 13.69 13.7 13.72 13.75 ... nan
nan nan nan
WINTER SEASON MINIMUM TEMPERATURE AT SEA SURFACE: <xarray.DataArray
'thetao' ()>
array(7.6017065, dtype=float32)
Coordinates:
    depth    float32 1.018
WINTER SEASON MAXIMUM TEMPERATURE AT SEA SURFACE: <xarray.DataArray
'thetao' ()>
array(16.397697, dtype=float32)
Coordinates:
    depth    float32 1.018
saving to  Aggregations/CMEMS_SST_WINTER_SEASON.nc
finished saving
```

```
SeasonSummer =
fc.GenerateSeasonalSummer(rawData_annual_mean,summer_output)
```

```
Reseized Area: <xarray.Dataset>
Dimensions:  (lat: 146, lon: 188, year: 33)
Coordinates:
    depth    float32 1.018
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
  * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017
2018 2019
Data variables:
    thetao   (year, lat, lon) float32 25.26 25.27 25.28 25.29 ... nan
nan nan
SUMMER SEASON MINIMUM TEMPERATURE AT SEA SURFACE: <xarray.DataArray
'thetao' ()>
array(19.326422, dtype=float32)
Coordinates:
    depth    float32 1.018
SUMMER SEASON MAXIMUM TEMPERATURE AT SEA SURFACE: <xarray.DataArray
'thetao' ()>
array(26.531475, dtype=float32)
Coordinates:
    depth    float32 1.018
saving to  Aggregations/CMEMS_SST_SUMMER_SEASON.nc
finished saving
```

The following 1D outputs are aggregated SST over the Adriatic Sea.

```
NcFile1Doutput = "Aggregations/CMEMS_SST_clipped_1D_FIXED_DIM.nc"

clippedfix1=fc.Generate1DFixDim(rawData_annual_mean,NcFile1Doutput)
```

```
saving to  Aggregations/CMEMS_SST_clipped_1D_FIXED_DIM.nc
finished saving
File Dimension: <xarray.Dataset>
Dimensions:  (time: 12053)
Coordinates:
  * time     (time) datetime64[ns] 1987-01-01T12:00:00 ... 2019-12-31T12:00:00
    depth    float32 1.018
Data variables:
    thetao   (time) float32 13.98 13.95 13.93 13.85 ... 16.13 15.95 15.8 15.69
```

```
NcFile1DoutputCSV= "Aggregations/CMEMS_SST_clipped_1D_FIXED_DIM.csv"
```

```
clippedData1Dcsv = fc.Generate1DFixDimCSV(NcFile1Doutput,NcFile1DoutputCSV)
```

```
clippedData1Dcsv
```

```
1987-01-01 12:00:00    13.981805
1987-01-02 12:00:00    13.951108
1987-01-03 12:00:00    13.931457
1987-01-04 12:00:00    13.849789
1987-01-05 12:00:00    13.725996
                         ...
2019-12-27 12:00:00    16.223177
2019-12-28 12:00:00    16.133165
2019-12-29 12:00:00    15.948340
2019-12-30 12:00:00    15.800557
2019-12-31 12:00:00    15.685452
Length: 12053, dtype: float32
```

Section 3: The SST Data Analysis and Plots.

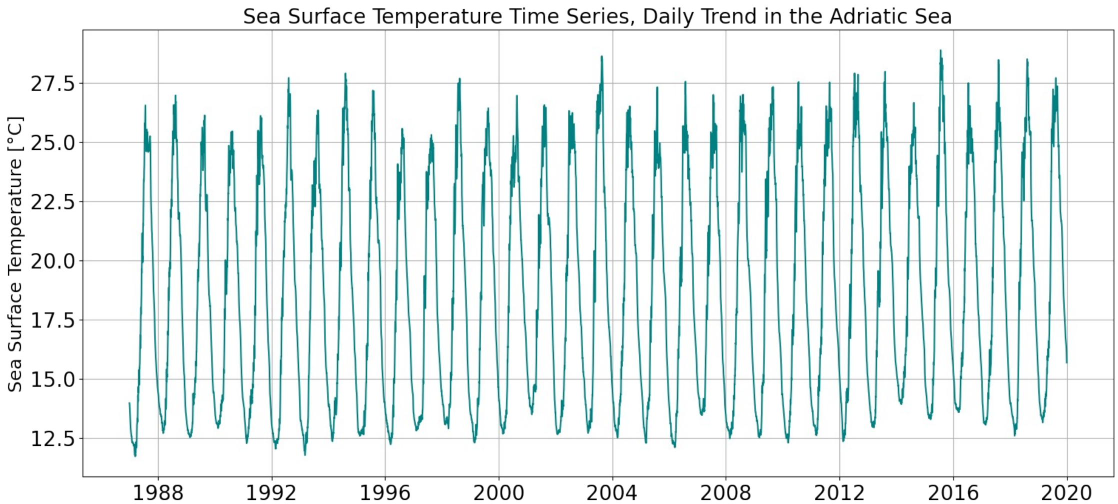
Back to top

The following plots will visualize: SST Time Series, Daily Trend in the Adriatic Sea,

The Time Series analysis have been generated from the previously clipped "CMEMS_SST_clipped_1D_FIXED_DIM" file in CSV format. Therefore, The **"faGenerateDailyTimeSeries"** function reads the Daily Mean file through pandas and parse dates with taking the list of "DATE" column.

```
NcFile1DoutputCSV= "Aggregations/CMEMS_SST_clipped_1D_FIXED_DIM.csv"
```

```
ts=fc.GenerateDailyTimeSeries(NcFile1DoutputCSV)
```



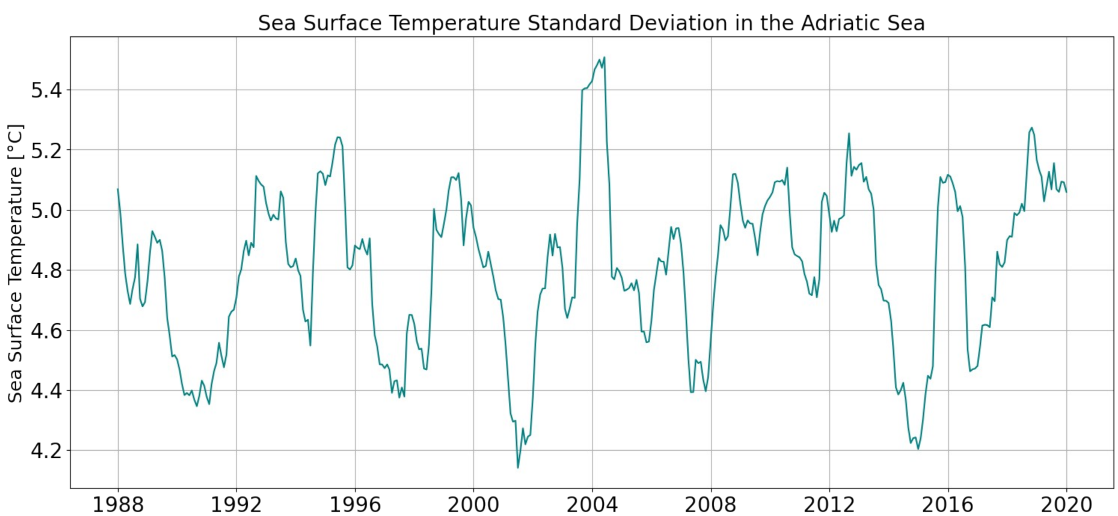
The **"GenerateDailyTimeSeriesSTD"** function starts with the **groupby** method to provide information on data in the "DATE" column in "Monthly Mean".

```
fy_dt = file2.groupby(pd.Grouper(freq='M')).mean()
```

The window size equal to 12 has been choosen for the moving average calculation to calculate standard deviation by year.

```
 daily_sdT = fy_dt.rolling(window = 12).std()
```

```
ts1=fc.GenerateDailyTimeSeriesSTD(NcFile1DoutputCSV)
```

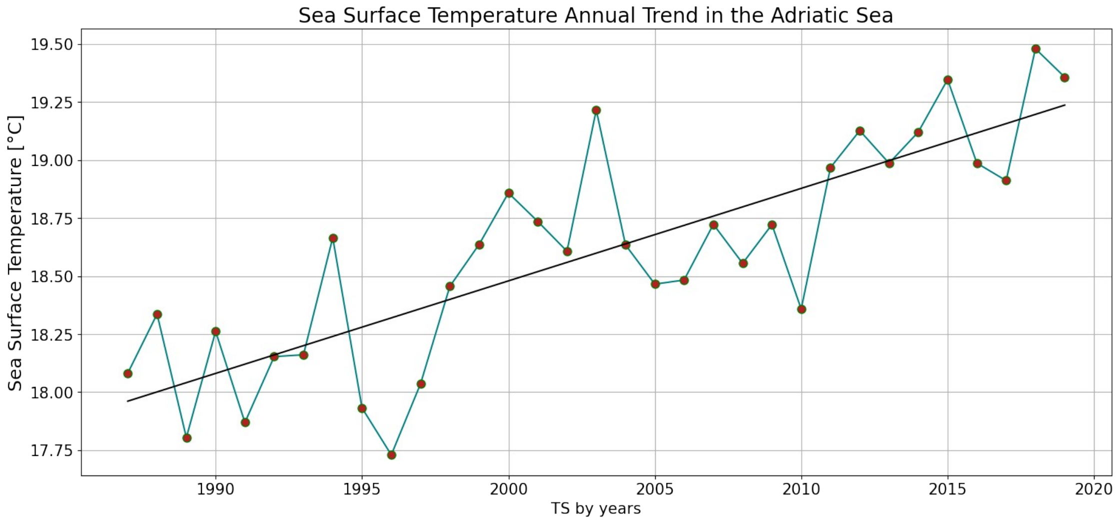


The **"Generate1DTendency"** function shows the Annual Trend in the Adriatic Sea. The input file is a 1-Dimensional file in netCDF format. Hence, the Linear Regression has been calculated, once the data frame for the Annual Mean has been created.

```
fy_1D= t.mean(dim=(lat_name, lon_name), skipna=True)
fy_dt = fy_1D.groupby('time.year').mean()
df = fy_dt.to_dataframe().reset_index().set_index('year')
```

The horizontal axis has "df.index" by year while vertical axis has the Temperature with the variable name "thetao" in the Cartesian coordinate system.

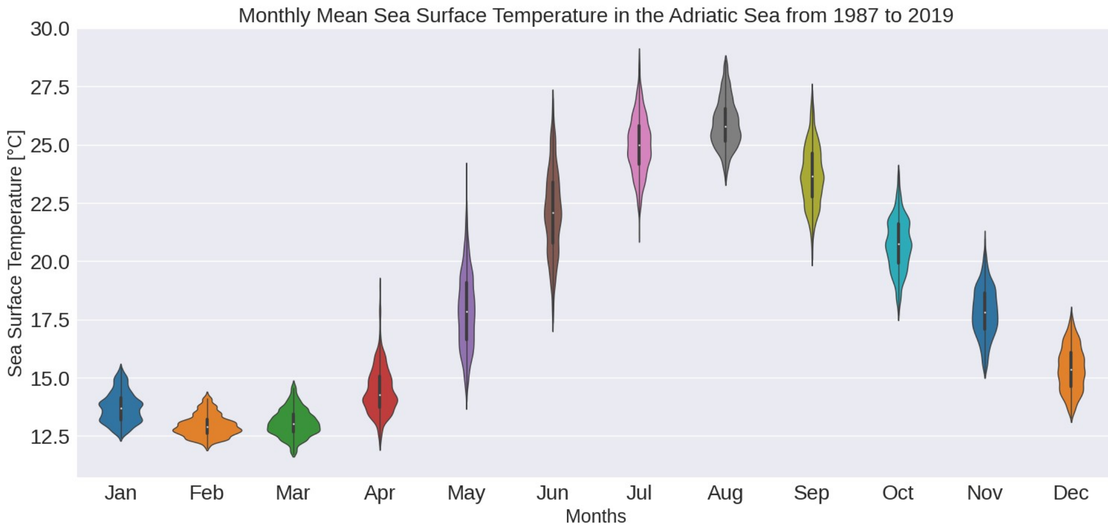
```
ncRawDataFileName_clipped = "Aggregations/CMEMS_SST_clipped.nc"
rawData1_clipped = xr.open_dataset(ncRawDataFileName_clipped)

NcFile1Doutput = "Aggregations/CMEMS_SST_clipped_1D_FIXED_DIM.nc"
clippedfix1Tendency=fc.Generate1DTendency(rawData1_clipped,NcFile1Dout
put)
```



The **"GenerateDailyTimeSeriesPLOT"** function shows the Monthly Mean with Violin Plot in the Adriatic Sea. The input file is a 1-Dimensional file in CSV format. The Data distribution is shown by using the Seaborn Library. The month names are converted in their full names with:

```
 file2['month'] = [d.strftime('%b') for d in file2.DATE]
```

```
ts_monthly=fc.GenerateDailyTimeSeriesPLOT(NcFile1DoutputCSV)
```



Section 4: The Seasonal and Annual Mean Map Visualizations

Back to top

Each map has been displayed with the following steps:

1. Loading the data variable for each selected period of time both seasonal and annual mean with dimensions of (time, lat, lon) and the spatial data to mask out the missing values.
2. The Mean Temperatures for each time coverage have been calculated.
3. Data masking for the area of interest through the vectorized library pygoes.
4. The Plate Carrée projection with the coastline has been selected for the map.
5. A heatmap generation for maximum and minimum temperature toghether with its contour line.
6. The colour bar has been set.

```
t_summer = xr.open_dataset('Aggregations/CMEMS_SST_SUMMER_SEASON.nc')
t_winter = xr.open_dataset('Aggregations/CMEMS_SST_WINTER_SEASON.nc')
t_annual_mean =
xr.open_dataset('Aggregations/CMEMS_SST_clipped_Annual_Mean.nc')

t_summer,t_winter,t_annual_mean
```

```
(<xarray.Dataset>
 Dimensions:  (lat: 146, lon: 188, year: 33)
 Coordinates:
     depth    float32 ...
   * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
   * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
   * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017
2018 2019
 Data variables:
```

```
    thetao   (year, lat, lon) float32 ...,
 <xarray.Dataset>
 Dimensions:  (lat: 146, lon: 188, year: 33)
 Coordinates:
     depth    float32 ...
   * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
   * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
   * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017
2018 2019
 Data variables:
     thetao   (year, lat, lon) float32 ...,
 <xarray.Dataset>
 Dimensions:  (lat: 146, lon: 188, year: 33)
 Coordinates:
     depth    float32 ...
   * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
   * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
   * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017
2018 2019
 Data variables:
     thetao   (year, lat, lon) float32 ...)
```

```
file_csv_area_1=
pd.read_csv('WEkEO_PART_1_Aggregations/WEkEO_SST_DATA/areaAdriatic.csv
')
```

The xarray.DataArray with Summer Period and geographical coordinates is shown:

```
temp_summer = t_summer['thetao'][:,:,:]
temp_summer
```

```
<xarray.DataArray 'thetao' (year: 33, lat: 146, lon: 188)>
[905784 values with dtype=float32]
Coordinates:
    depth    float32 ...
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
  * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017
2018 2019
```

The xarray.DataArray with Winter Period and geographical coordinates is shown:

```
temp_winter = t_winter['thetao'][:,:,:]
temp_winter
```

```
<xarray.DataArray 'thetao' (year: 33, lat: 146, lon: 188)>
[905784 values with dtype=float32]
```

```
Coordinates:
    depth    float32 ...
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
  * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017
2018 2019
```

And finally, the xarray.DataArray with Annual Mean and geographical coordinates is as follows: (the Time period is 33 years as noted before).

```
temp_annual_mean = t_annual_mean['thetao'][:,:,:]
temp_annual_mean
```

```
<xarray.DataArray 'thetao' (year: 33, lat: 146, lon: 188)>
[905784 values with dtype=float32]
Coordinates:
    depth    float32 ...
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
  * year     (year) int64 1987 1988 1989 1990 1991 ... 2015 2016 2017
2018 2019
```

Maximum and Minimum Mean Temperatures for each time coverage follow:

```
temp_summer_av= np.mean(temp_summer[:],axis = 0)
temp_summer_av.min(),temp_summer_av.max()
```

```
(<xarray.DataArray 'thetao' ()>
 array(20.6577, dtype=float32)
 Coordinates:
     depth    float32 1.018,
 <xarray.DataArray 'thetao' ()>
 array(25.221352, dtype=float32)
 Coordinates:
     depth    float32 1.018)
```

```
temp_winter_av= np.mean(temp_winter[:],axis = 0)
temp_winter_av.min(),temp_winter_av.max()
```

```
(<xarray.DataArray 'thetao' ()>
 array(9.44431, dtype=float32)
 Coordinates:
     depth    float32 1.018,
 <xarray.DataArray 'thetao' ()>
 array(15.3477745, dtype=float32)
 Coordinates:
     depth    float32 1.018)
```

```
temp_annual_mean_av= np.mean(temp_annual_mean[:],axis = 0)
temp_annual_mean_av.min(),temp_annual_mean_av.max()
```

```
(<xarray.DataArray 'thetao' ()>
 array(16.09838, dtype=float32)
 Coordinates:
     depth    float32 1.018,
 <xarray.DataArray 'thetao' ()>
 array(20.021091, dtype=float32)
 Coordinates:
     depth    float32 1.018)
```

```
lon_name_summer   = temp_summer.lon[:]
lat_name_summer   = temp_summer.lat[:]

lon_name_winter   = temp_winter.lon[:]
lat_name_winter   = temp_winter.lat[:]

lon_name_annual_mean   = temp_annual_mean.lon[:]
lat_name_annual_mean   = temp_annual_mean.lat[:]
```

Masking the area for the Adriatic Sea during the Summer Season

```
outline_adriatic = np.array(file_csv_area_1)

region_area_adriatic = regionmask.Regions([outline_adriatic])

mask_pygeos_area_summer = region_area_adriatic.mask(t_summer.thetao,
method="pygeos")
LON, LAT = np.meshgrid(lon_name_summer, lat_name_summer)
```

```
/opt/conda/lib/python3.8/site-packages/pygeos/io.py:85: UserWarning:
The shapely GEOS version (3.9.1-CAPI-1.14.2) is incompatible with the
PyGEOS GEOS version (3.10.1-CAPI-1.16.0). Conversions between both
will be slow
  warnings.warn(
```

```
thetao_area_summer = temp_summer_av.values
thetao_area_summer[np.isnan(mask_pygeos_area_summer)] = np.nan

thetao_area_summer
```

```
array([[      nan,       nan,       nan, ...,       nan,       nan,
              nan],
       [      nan,       nan,       nan, ...,       nan,       nan,
              nan],
       [      nan,       nan,       nan, ...,       nan, 23.820696,
        23.96131 ],
       ...,
       [      nan,       nan,       nan, ...,       nan,       nan,
              nan],
       [      nan,       nan,       nan, ...,       nan,       nan,
```

```
         nan],
       [      nan,       nan,       nan, ...,       nan,       nan,
         nan]], dtype=float32)
```

Masking the area for the Adriatic Sea during the Winter Season

```
mask_pygeos_area_winter = region_area_adriatic.mask(t_winter.thetao,
method="pygeos")
LON1, LAT1 = np.meshgrid(lon_name_winter, lat_name_winter)

thetao_area_winter = temp_winter_av.values
thetao_area_winter[np.isnan(mask_pygeos_area_winter)] = np.nan

thetao_area_winter
```

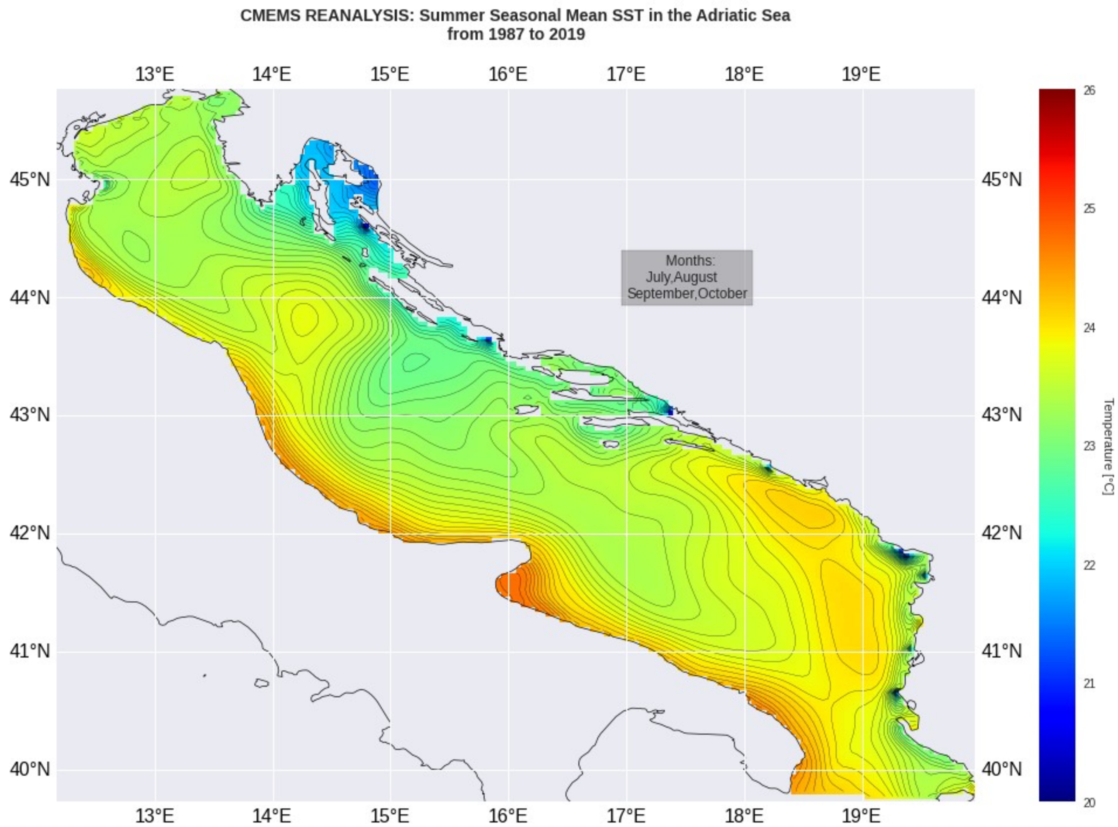
```
array([[      nan,       nan,       nan, ...,       nan,       nan,
         nan],
       [      nan,       nan,       nan, ...,       nan,       nan,
         nan],
       [      nan,       nan,       nan, ...,       nan, 14.702645,
        14.63934 ],
       ...,
       [      nan,       nan,       nan, ...,       nan,       nan,
         nan],
       [      nan,       nan,       nan, ...,       nan,       nan,
         nan],
       [      nan,       nan,       nan, ...,       nan,       nan,
         nan]], dtype=float32)
```

Masking the area for the Adriatic Sea for the Annual Mean

```
mask_pygeos_area_annual_mean =
region_area_adriatic.mask(t_annual_mean.thetao, method="pygeos")
LON2, LAT2 = np.meshgrid(lon_name_annual_mean, lat_name_annual_mean)

thetao_area_annual_mean = temp_annual_mean_av.values
thetao_area_annual_mean[np.isnan(mask_pygeos_area_annual_mean)] =
np.nan

thetao_area_annual_mean
```

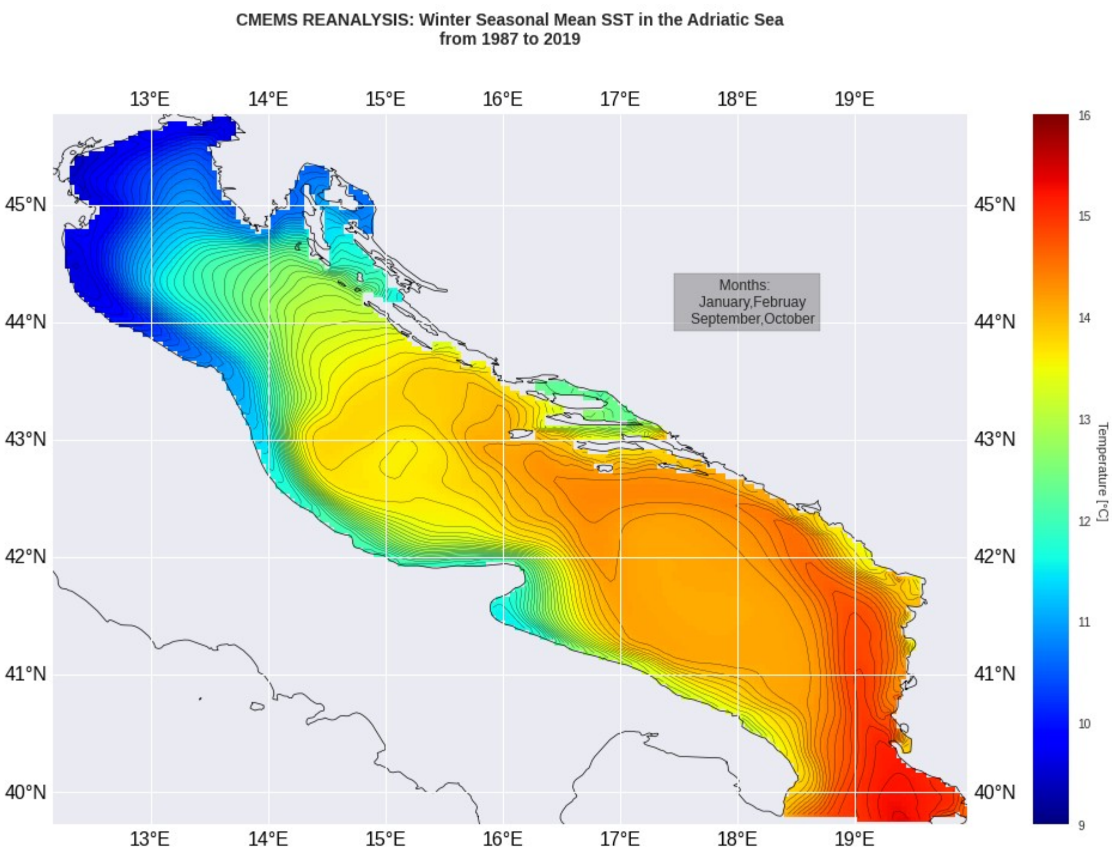
```
array([[      nan,       nan,       nan, ...,       nan,       nan,
         nan],
       [      nan,       nan,       nan, ...,       nan,       nan,
         nan],
       [      nan,       nan,       nan, ...,       nan, 19.334784,
        19.379007],
       ...,
       [      nan,       nan,       nan, ...,       nan,       nan,
         nan],
       [      nan,       nan,       nan, ...,       nan,       nan,
         nan],
```

```
       [      nan,       nan,       nan, ...,       nan,       nan,
              nan]], dtype=float32)

fig = plt.figure(figsize=(16, 10))
ax = plt.subplot(projection=ccrs.PlateCarree())

heatmap=temp_summer_av.plot(
    ax=ax,
    x="lon",
    y="lat",
    transform=ccrs.PlateCarree(),
    cmap="jet",
    shading="auto",
    add_colorbar=False,
    vmin=20,
    vmax=26
)
lines=temp_summer_av.plot.contour(ax=ax,alpha=1,linewidths=0.3,colors
= 'k',linestyles='None',levels=60)
# the level of contour lines= (vmax-vmin)*10


g1 = ax.gridlines(draw_labels = True)
g1.xlabel_style = {'size': 16, 'color': 'k'}
g1.ylabel_style = {'size': 16, 'color': 'k'}
#add embellishment


ax.add_feature(cfeature.COASTLINE,linewidths=0.7,alpha=0.9999)



plt.title("CMEMS REANALYSIS: Summer Seasonal Mean SST in the Adriatic
Sea\nfrom 1987 to 2019\n",fontweight='bold', size=14)
cbar = plt.colorbar(heatmap)
cbar.ax.set_ylabel('Temperature [°C]',labelpad=+14, rotation=270)

plt.text(17,44,'           Months:\n     July,August\
nSeptember,October',fontsize=12,bbox = dict(facecolor = 'gray', alpha
= 0.5))
plt.tight_layout()
plt.savefig('image_outputs/SummerSeasonalMean.png')

plt.show()
```




```
fig = plt.figure(figsize=(16, 10))
ax = plt.subplot(projection=ccrs.PlateCarree())
heatmap=temp_winter_av.plot(
    ax=ax,
    x="lon",
    y="lat",
    transform=ccrs.PlateCarree(),
    cmap="jet",
    shading="auto",
    add_colorbar=False,
    vmin=9,
    vmax=16
)

lines=temp_winter_av.plot.contour(ax=ax,alpha=1,linewidths=0.3,colors
= 'k',linestyles='None',levels=70)
# the level of contour lines= (vmax-vmin)*10

g1 = ax.gridlines(draw_labels = True)
g1.xlabel_style = {'size': 16, 'color': 'k'}
g1.ylabel_style = {'size': 16, 'color': 'k'}
#add embellishment
```

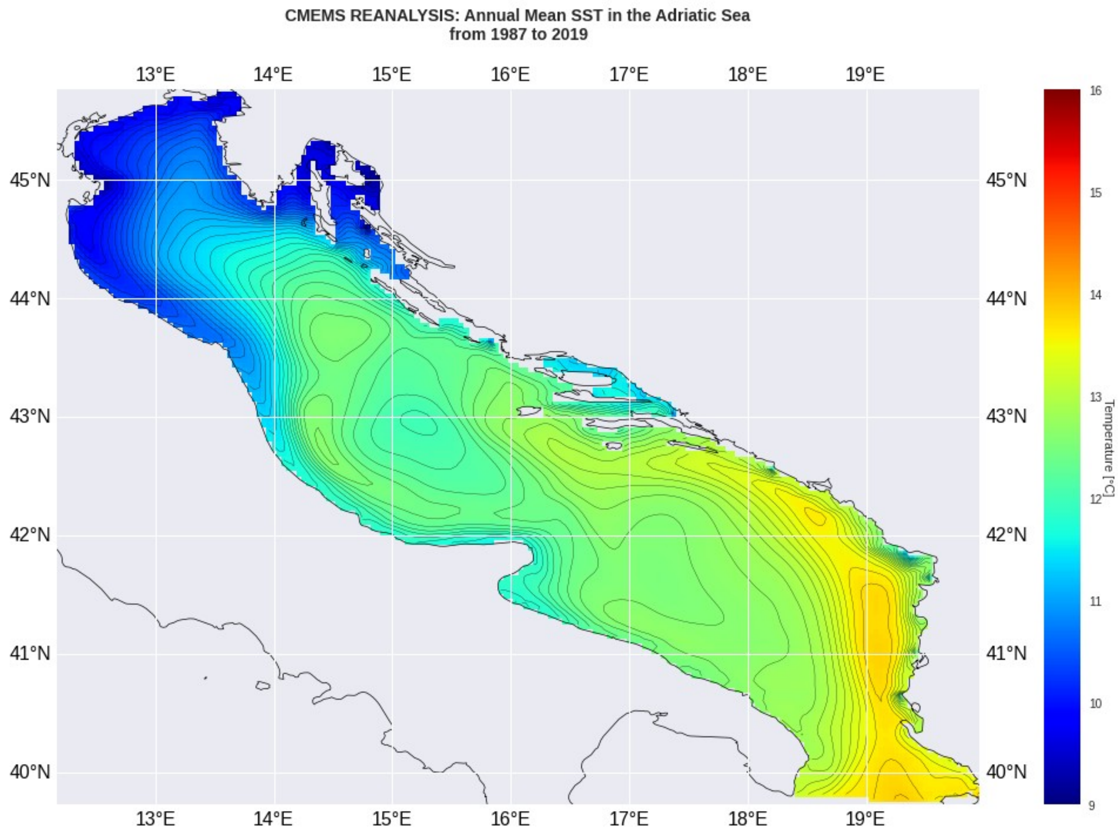
```
ax.add_feature(cfeature.COASTLINE,linewidths=0.7,alpha=0.9999)



plt.title("CMEMS REANALYSIS: Winter Seasonal Mean SST in the Adriatic Sea\nfrom 1987 to 2019\n\n",fontweight='bold', size=14)
cbar = plt.colorbar(heatmap)
cbar.ax.set_ylabel('Temperature [°C]', labelpad=+14,rotation=270)

plt.text(17.5,44,'          Months:\n     January,Februay\n September,October',fontsize=12,bbox = dict(facecolor = 'gray', alpha = 0.5))
plt.tight_layout()
plt.savefig('image_outputs/WinterSeasonalMean.png')

plt.show()
```




```
fig = plt.figure(figsize=(16, 10))
ax = plt.subplot(projection=ccrs.PlateCarree())
hetmap=temp_annual_mean_av.plot(
    ax=ax,
    x="lon",
    y="lat",
```

```
    transform=ccrs.PlateCarree(),
    cmap="jet",
    shading="auto",
    add_colorbar=False,
    vmin=16,
    vmax=21
)

lines=temp_annual_mean_av.plot.contour(ax=ax,alpha=1,linewidths=0.3,co
lors = 'k',linestyles='None',levels=50)
# the level of contour lines= (vmax-vmin)*10

g1 = ax.gridlines(draw_labels = True)
g1.xlabel_style = {'size': 16, 'color': 'k'}
g1.ylabel_style = {'size': 16, 'color': 'k'}
#add embellishment




ax.add_feature(cfeature.COASTLINE,linewidths=0.7,alpha=0.9999)

plt.title("CMEMS REANALYSIS: Annual Mean SST in the Adriatic Sea\nfrom
1987 to 2019\n",fontweight='bold', size=14)

cbar = plt.colorbar(heatmap)
cbar.ax.set_ylabel('Temperature [°C]', labelpad=+10, rotation=270)


plt.tight_layout()
plt.savefig('image_outputs/AnnualSeasonalMean.png')

plt.show()
```

```
/tmp/ipykernel_10558/3026980815.py:30: MatplotlibDeprecationWarning:
Starting from Matplotlib 3.6, colorbar() will steal space from the
mappable's axes, rather than from the current axes, to place the
colorbar.  To silence this warning, explicitly pass the 'ax' argument
to colorbar().
  cbar = plt.colorbar(heatmap)
```



Section 5: The Seasonal Anomalies Map Visualizations.

Back to top

Load previously generated files to calculate the Seasonal Anomalies:

1. "Aggregations/CMEMS_SST_SUMMER_SEASON.nc"
2. "Aggregations/CMEMS_SST_WINTER_SEASON.nc"

And,

1. "WEkEO_PART_1_Aggregations/areaAdriatic.csv" to Mask out the missing values.

The Seasonal Anomalies have been calculated through the **"computeSenSlopeMap"** function. "Sen Slope" is a method for robust linear regression. It computes the slope as the median of all slopes (in our case all seasonal mean) between paired values.

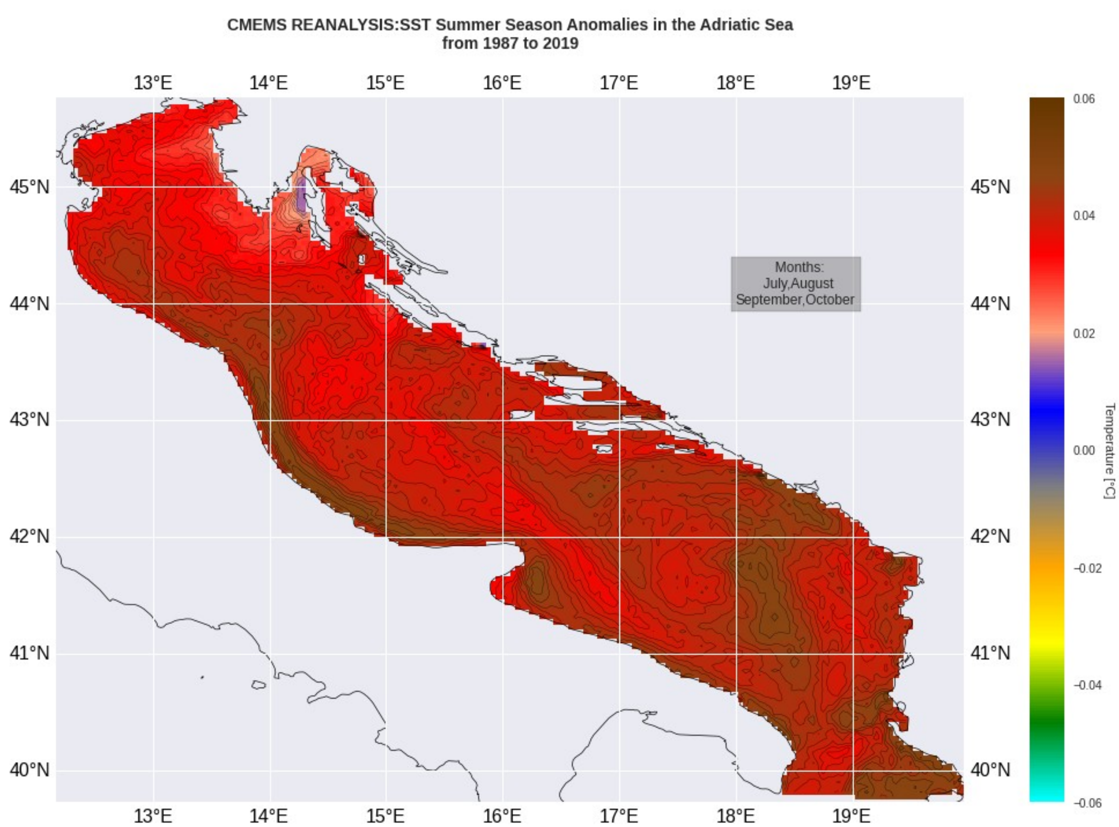
The "xr.apply_ufunc" is a vectorization function for unlabeled arrays on xarray objects and it is used to create seasonal maps.

```
ncRawDataFileName_summer = "Aggregations/CMEMS_SST_SUMMER_SEASON.nc"
ncRawDataFileName_winter = "Aggregations/CMEMS_SST_WINTER_SEASON.nc"

NcFileDoutput_summer_anomal = "Aggregations/SummerAnomalyOutput.nc"
NcFileDoutput_winter_anomal = "Aggregations/WinterAnomalyOutput.nc"
```

```
rawData_SUMMER_anomaly = xr.open_dataset(ncRawDataFileName_summer)
rawData_WINTER_anomaly = xr.open_dataset(ncRawDataFileName_winter)

summerAnomalyFile=fc.computeSenSlopeMap(ncRawDataFileName_summer,NcFil
eDoutput_summer_anomal)

Output: <xarray.Dataset>
Dimensions:  (lat: 146, lon: 188)
Coordinates:
    depth    float32 ...
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
Data variables:
    thetao   (lat, lon) float64 0.03278 0.0315 0.03187 0.03103 ... nan
nan nan
output  min: <xarray.DataArray 'thetao' ()>
array(0.01442564)
Coordinates:
    depth    float32 ...
output max: <xarray.DataArray 'thetao' ()>
array(0.0502005)
Coordinates:
    depth    float32 ...

winterAnomalyFile=fc.computeSenSlopeMap(ncRawDataFileName_winter,NcFil
eDoutput_winter_anomal)

Output: <xarray.Dataset>
Dimensions:  (lat: 146, lon: 188)
Coordinates:
    depth    float32 ...
  * lat      (lat) float32 39.73 39.77 39.81 39.85 ... 45.65 45.69
45.73 45.77
  * lon      (lon) float32 12.17 12.21 12.25 12.29 ... 19.83 19.88
19.92 19.96
Data variables:
    thetao   (lat, lon) float64 0.01877 0.01959 0.01985 0.01864 ...
nan nan nan
output  min: <xarray.DataArray 'thetao' ()>
array(0.01317847)
Coordinates:
    depth    float32 ...
output max: <xarray.DataArray 'thetao' ()>
array(0.05237167)
Coordinates:
    depth    float32 ...

summer_anomaly_vis =
xr.open_dataset("Aggregations/SummerAnomalyOutput.nc")
```

```
lon_name_summer   = summer_anomaly_vis.lon[:]
lat_name_summer  = summer_anomaly_vis.lat[:]
time_name_summer  = 'year'
depth_name_summer = 'depth'
temp_summer = summer_anomaly_vis.thetao[:]

outline_1 = np.array(file_csv_area_1)

region_area_1 = regionmask.Regions([outline_1])

mask_pygeos_area_1 = region_area_1.mask(summer_anomaly_vis.thetao,
method="pygeos")
LON, LAT = np.meshgrid(lon_name_summer, lat_name_summer)

thetao_area_1 = summer_anomaly_vis.thetao.values
thetao_area_1[np.isnan(mask_pygeos_area_1)] = np.nan

fig = plt.figure(figsize=(16, 10))
ax = plt.subplot(projection=ccrs.PlateCarree())
cmap = matplotlib.colors.LinearSegmentedColormap.from_list("",
["#653700","saddlebrown","red","lightsalmon","blue","gray","orange","y
ellow","green","cyan"])
#new colorbar generation

heatmap=summer_anomaly_vis.thetao.plot(
    ax=ax,
    x="lon",
    y="lat",
    transform=ccrs.PlateCarree(),
    cmap=cmap.reversed(),
    shading="auto",
    add_colorbar=False,
    vmin=-0.06,
    vmax=0.06
)

lines=summer_anomaly_vis.thetao.plot.contour(ax=ax,alpha=1,linewidths=
0.3,colors = 'k',linestyles='None',levels=30)

g1 = ax.gridlines(draw_labels = True)
g1.xlabel_style = {'size': 16, 'color': 'k'}
g1.ylabel_style = {'size': 16, 'color': 'k'}
#add embellishment




ax.add_feature(cfeature.COASTLINE,linewidths=0.7,alpha=0.9999)
```

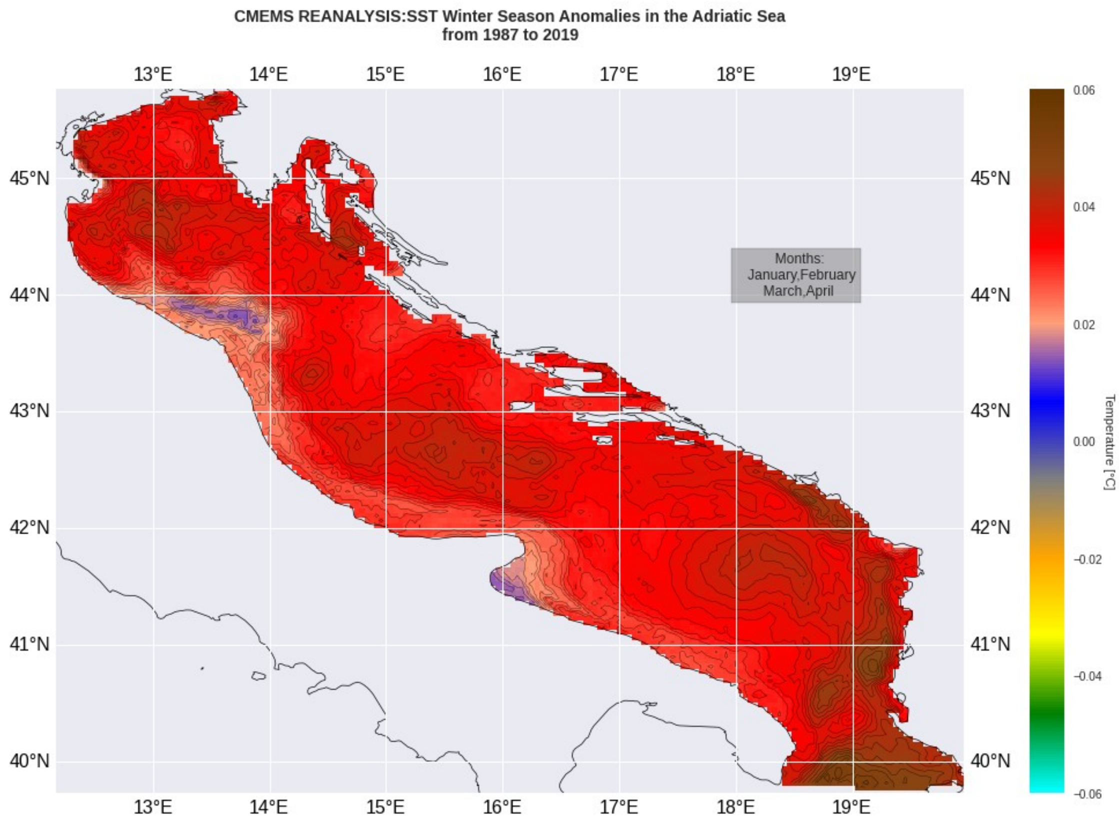
```
plt.title("CMEMS REANALYSIS:SST Summer Season Anomalies in the
Adriatic Sea\nfrom 1987 to 2019\n",fontweight='bold', size=14)

plt.tight_layout()
plt.savefig('image_outputs/SummerSEasonAnnomalies.png')

cbar = plt.colorbar(heatmap)
cbar.ax.set_ylabel('Temperature [°C]',labelpad=+10, rotation=270)

plt.text(18,44,'          Months:\n       July,August\
nSeptember,October',fontsize=12,bbox = dict(facecolor = 'gray', alpha
= 0.5))

plt.tight_layout()

plt.show()
```




```
winter_anomaly_vis =
xr.open_dataset("Aggregations/WinterAnomalyOutput.nc")

lon_name_winter   = winter_anomaly_vis.lon[:]
lat_name_winter  = winter_anomaly_vis.lat[:]
time_name_winter  = 'year'
depth_name_winter = 'depth'
temp_winter = winter_anomaly_vis.thetao[:]
```

```
outline_1 = np.array(file_csv_area_1)

region_area_1 = regionmask.Regions([outline_1])

mask_pygeos_area_1 = region_area_1.mask(winter_anomaly_vis.thetao, 
method="pygeos")
LON, LAT = np.meshgrid(lon_name_winter, lat_name_winter)

thetao_area_1 = winter_anomaly_vis.thetao.values
thetao_area_1[np.isnan(mask_pygeos_area_1)] = np.nan

fig = plt.figure(figsize=(16, 10))
ax = plt.subplot(projection=ccrs.PlateCarree())
cmap = matplotlib.colors.LinearSegmentedColormap.from_list("", 
["#653700","saddlebrown","red","lightsalmon","blue","gray","orange","y
ellow","green","cyan"])
#new colorbar generation

heatmap=winter_anomaly_vis.thetao.plot(
    ax=ax,
    x="lon",
    y="lat",
    transform=ccrs.PlateCarree(),
    cmap=cmap.reversed(),
    shading="auto",
    add_colorbar=False,
    vmin=-0.06,
    vmax=0.06
)

lines=winter_anomaly_vis.thetao.plot.contour(ax=ax,alpha=1,linewidths=
0.3,colors = 'k',linestyles='None',levels=30)

g1 = ax.gridlines(draw_labels = True)
g1.xlabel_style = {'size': 16, 'color': 'k'}
g1.ylabel_style = {'size': 16, 'color': 'k'}
#add embellishment




ax.add_feature(cfeature.COASTLINE,linewidths=0.7,alpha=0.9999)


plt.title("CMEMS REANALYSIS:SST Winter Season Anomalies in the 
Adriatic Sea\nfrom 1987 to 2019\n",fontweight='bold', size=14)

plt.tight_layout()
plt.savefig('image_outputs/WinterSEasonAnnomalies.png')
```

```
cbar = plt.colorbar(heatmap)
cbar.ax.set_ylabel('Temperature [°C]',labelpad=+10, rotation=270)

plt.text(18,44,'          Months:\n   January,February\n
March,April',fontsize=12,bbox = dict(facecolor = 'gray', alpha = 0.5))

plt.tight_layout()

plt.show()
```



Section 6: References

- WEkEO LINK
- Copernicus Marine LINK
- Mediterranean Sea Physics Reanalysis LINK
- WEkEO Data-- EO:MO:DAT:MEDSEA_MULTIYEAR_PHY_006_004 LINK
- Global Climate Observing Sysem - GCOS WMO: Sea Surface Temperature (SST) LINK
- "The Adriatic Sea General Circulation. Part I: Air–Sea Interactions and Water Mass Structure" DOI