

datos_temporales

October 17, 2025

0.0.1 Datos temporales con Xarray

Manejo y representación de datos temporales en Python

Guillermo Martínez Flores

Script original proporcionado por Yeray Santana

Librerías principales

1. [Pandas](#)
2. [Xarray](#)

Usamos **Xarray** para datos visualizar datos espaciotemporales (en 3 o más dimensiones).

Datos

Están disponibles en el directorio de datos en formato NetCDF.

Archivo: ERA5_Coarse.nc

Instalación

Como se menciona en el README.md podemos instalar las librerías necesarias (**pandas** y **xarray**, entre otras) utilizando el archivo `environment.yml`:

```
conda env create -f environment.yml -n geospatial
```

Notebook

Este notebook se basa en el script compartido por Yeray Santana para introducir el uso de **Python** para visualizar datos temporales.

0.0.2 Importar librerías

```
[102]: import numpy as np
import pandas as pd
import xarray as xr
import matplotlib.pyplot as plt
from matplotlib.dates import date2num, num2date, datetime
import cmocean as cmo
import subprocess
import scipy.stats as sstat
```

```
from scipy.interpolate import UnivariateSpline
import seaborn as sns
```

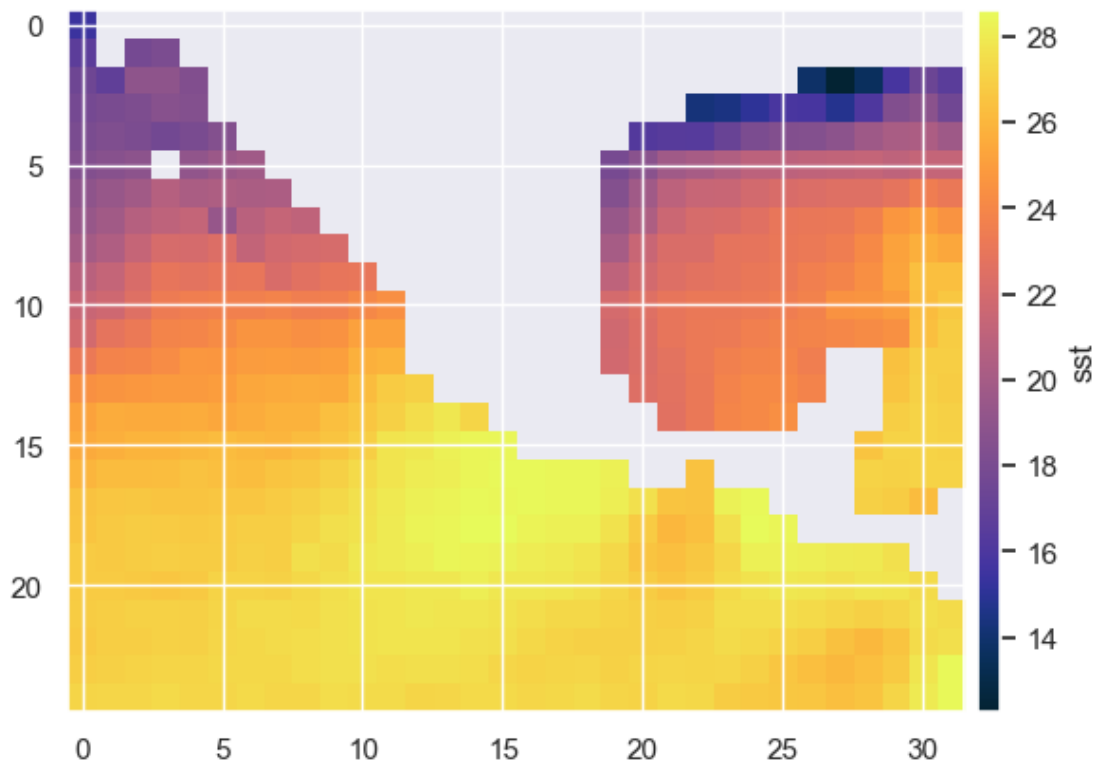
0.0.3 Abrir archivo NetCDF con Xarray

```
[103]: roi = [8,33,-117+360,-85+360] # region de interés
ds = xr.open_dataset('data/ERA5_Coarse.nc')
sst = ds.sst
sst = sst - 273.15 # convirtiendo a Celsius de Kelvin
sst = sst.sel(latitude=slice(roi[1],roi[0]), longitude=slice(roi[2],roi[3]))
```

0.0.4 Mapa de TSM de la primera capa temporal

```
[104]: cmap = cmo.cm.thermal
fig, ax = plt.subplots()
image = plt.imshow(sst[0],cmap=cmap)

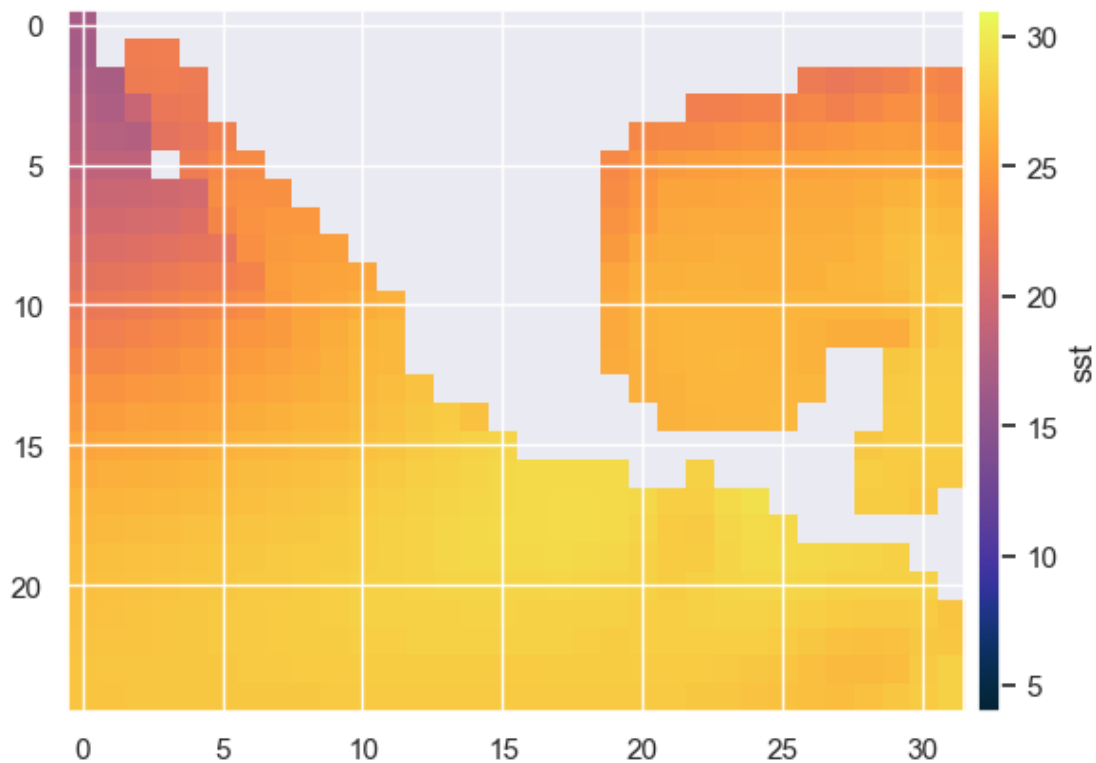
cax = fig.add_axes([ax.get_position().x1+0.01, ax.get_position().y0,0.02, ax.
    ↪get_position().height])
cbar= plt.colorbar(image, cax=cax)
cbar.set_label('sst')
plt.show()
```



0.0.5 Mapa de TSM media

```
[106]: sst_mean = sst.mean(dim='time')
fig, ax = plt.subplots()
image = plt.imshow(sst_mean, cmap=cmap, vmin=4, vmax=31)

cax = fig.add_axes([ax.get_position().x1+0.01, ax.get_position().y0, 0.02, ax.
    ↪ get_position().height])
cbar= plt.colorbar(image, cax=cax)
cbar.set_label('sst')
plt.show()
```



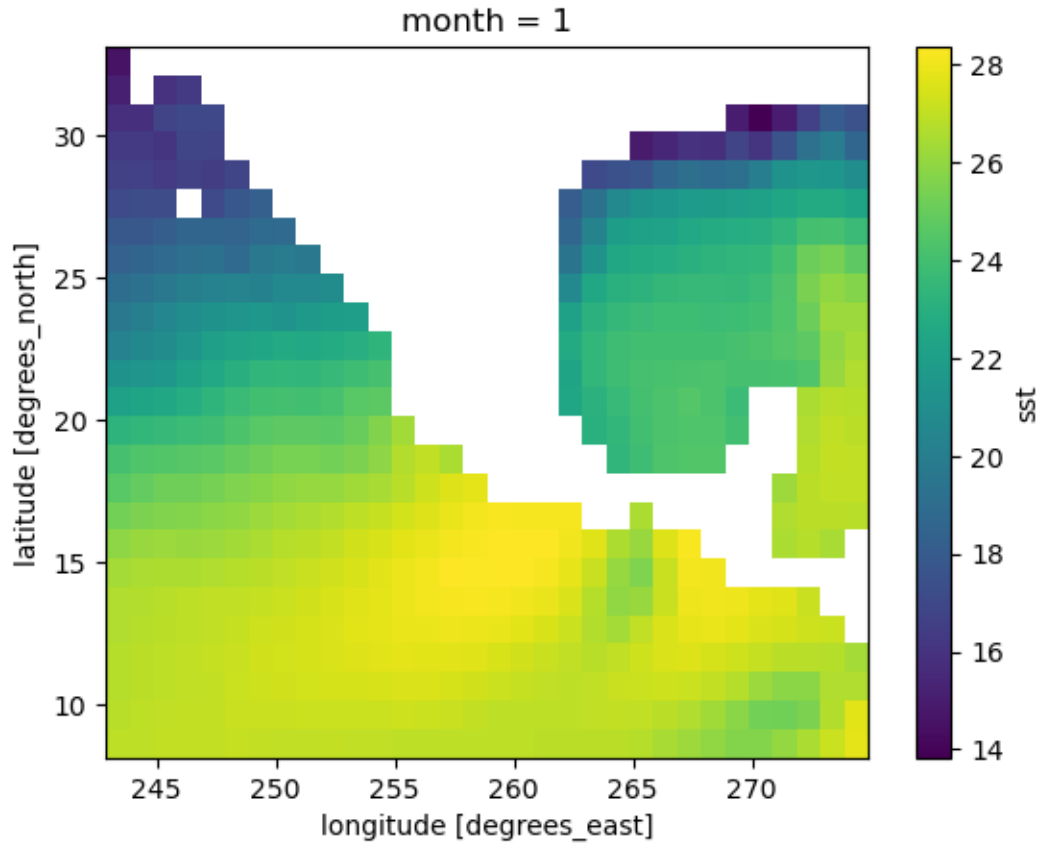
0.0.6 Seleccionar datos para años específicos

```
[63]: # Seleccionar un año concreto
print(sst.sel(time='1965').squeeze().time.values)
# Seleccionar un periodo concreto
print(sst.sel(time=slice('1965','1995')).time.values)
# Seleccionar un mes concreto
print(sst.isel(time=(sst.time.dt.month == 1)).time.values)

1965-09-01T00:00:00.000000000
['1965-09-01T00:00:00.000000000' '1969-01-01T00:00:00.000000000'
 '1972-05-01T00:00:00.000000000' '1975-09-01T00:00:00.000000000'
 '1979-01-01T00:00:00.000000000' '1982-05-01T00:00:00.000000000'
 '1985-09-01T00:00:00.000000000' '1989-01-01T00:00:00.000000000'
 '1992-05-01T00:00:00.000000000' '1995-09-01T00:00:00.000000000']
['1959-01-01T00:00:00.000000000' '1969-01-01T00:00:00.000000000'
 '1979-01-01T00:00:00.000000000' '1989-01-01T00:00:00.000000000'
 '1999-01-01T00:00:00.000000000' '2009-01-01T00:00:00.000000000'
 '2019-01-01T00:00:00.000000000']
```

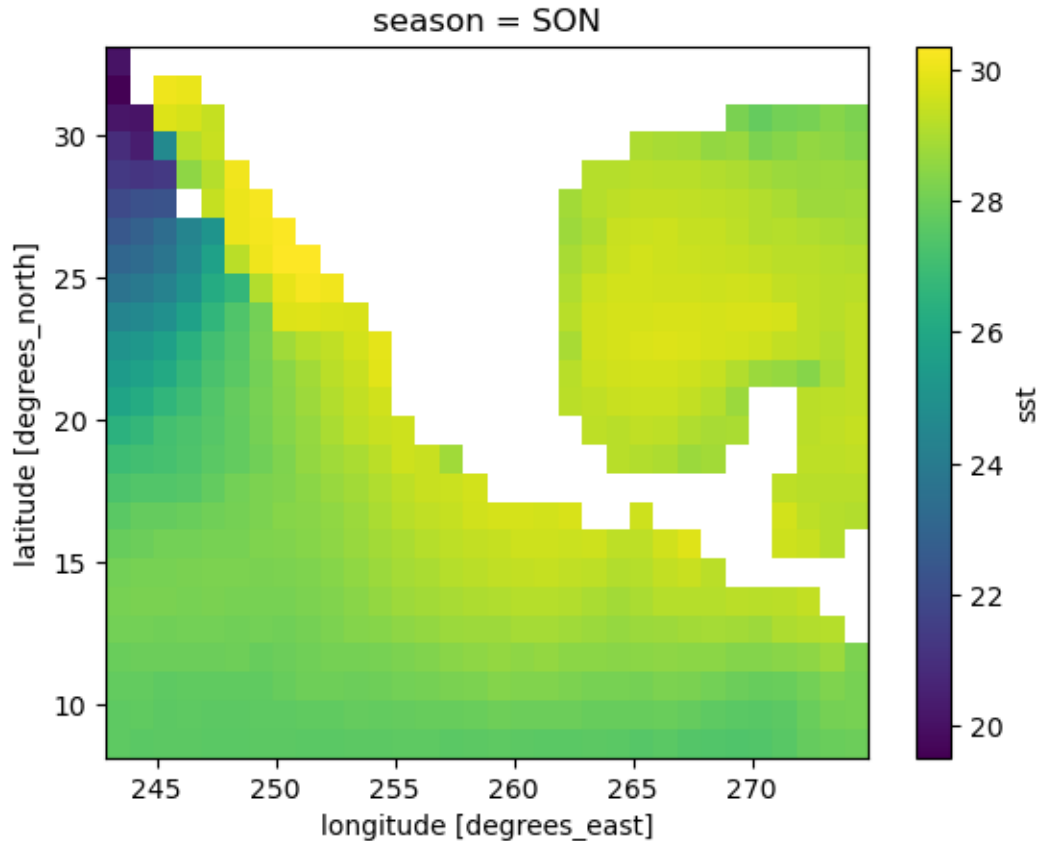
0.0.7 Agrupar por mes

```
[72]: sst.groupby("time.month") # Lo que significa es que tenemos datos de los meses ↪  
      ↪ 1, 5 y 9  
      #sst.groupby("time.year")  
      sst.groupby("time.month").mean().sel(month=1).plot(); # Así graficamos los ↪  
      ↪ datos del mes
```



0.0.8 Agrupar por estación del año

```
[71]: sst.groupby("time.season").mean().isel(season=2).plot();
```



0.0.9 Función resample

```
[73]: sst.resample(time='1YE').mean() # 'S' usa el inicio del año, y 'E' el final
```

```
[73]: <xarray.DataArray 'sst' (time: 61, latitude: 25, longitude: 32)> Size: 390kB
array([[[[15.43694278,      nan,      nan, ...,      nan,
          nan,      nan],
        [16.63696362,      nan, 17.77837162, ...,      nan,
          nan,      nan],
        [17.45384156, 16.81218517, 19.00245457, ..., 15.84476477,
          17.32119144, 16.54688494],
        ...,
        [26.70294818, 26.91518837, 26.89667905, ..., 26.53204547,
          27.42913046, 27.5747371 ],
        [26.93184676, 26.9960124 , 27.08300619, ..., 26.69616143,
          27.80116777, 28.51994631],
        [27.18357349, 27.224294  , 27.2458882 , ..., 27.18357349,
          27.99983445, 28.528584  ]],
```

```

[[      nan,      nan,      nan, ...,      nan,
      nan,      nan],
 [      nan,      nan,      nan, ...,      nan,
      nan,      nan],
 [      nan,      nan,      nan, ...,      nan,
      nan,      nan],
...
 [      nan,      nan,      nan, ...,      nan,
      nan,      nan],
 [      nan,      nan,      nan, ...,      nan,
      nan,      nan],
 [      nan,      nan,      nan, ...,      nan,
      nan,      nan]],
[[15.08835061,      nan,      nan, ...,      nan,
      nan,      nan],
 [15.74481445,      nan, 16.18471926, ...,      nan,
      nan,      nan],
 [16.62770896, 16.42657436, 17.39893058, ..., 16.28898843,
 17.95667805, 17.14103407],
...,
 [26.97935401, 27.07930433, 27.13236438, ..., 26.13471209,
 27.0990476 , 26.6208902 ],
 [27.07128363, 27.15642649, 27.19591304, ..., 26.57214899,
 27.68517603, 28.40395458],
 [27.22799586, 27.2594617 , 27.28105591, ..., 28.07942453,
 28.47367302, 28.74267512]]], shape=(61, 25, 32))

```

Coordinates:

```

* latitude (latitude) float32 100B 32.62 31.62 30.62 ... 10.62 9.625 8.625
* longitude (longitude) float32 128B 243.4 244.4 245.4 ... 272.4 273.4 274.4
* time      (time) datetime64[ns] 488B 1959-12-31 1960-12-31 ... 2019-12-31

```

```
[74]: sst.resample(time='1MS').mean().time # ¿En qué se diferencia de groupby?
```

```
[74]: <xarray.DataArray 'time' (time: 721)> Size: 6kB
array(['1959-01-01T00:00:00.000000000', '1959-02-01T00:00:00.000000000',
      '1959-03-01T00:00:00.000000000', ..., '2018-11-01T00:00:00.000000000',
      '2018-12-01T00:00:00.000000000', '2019-01-01T00:00:00.000000000'],
      shape=(721,), dtype='datetime64[ns]')
```

Coordinates:

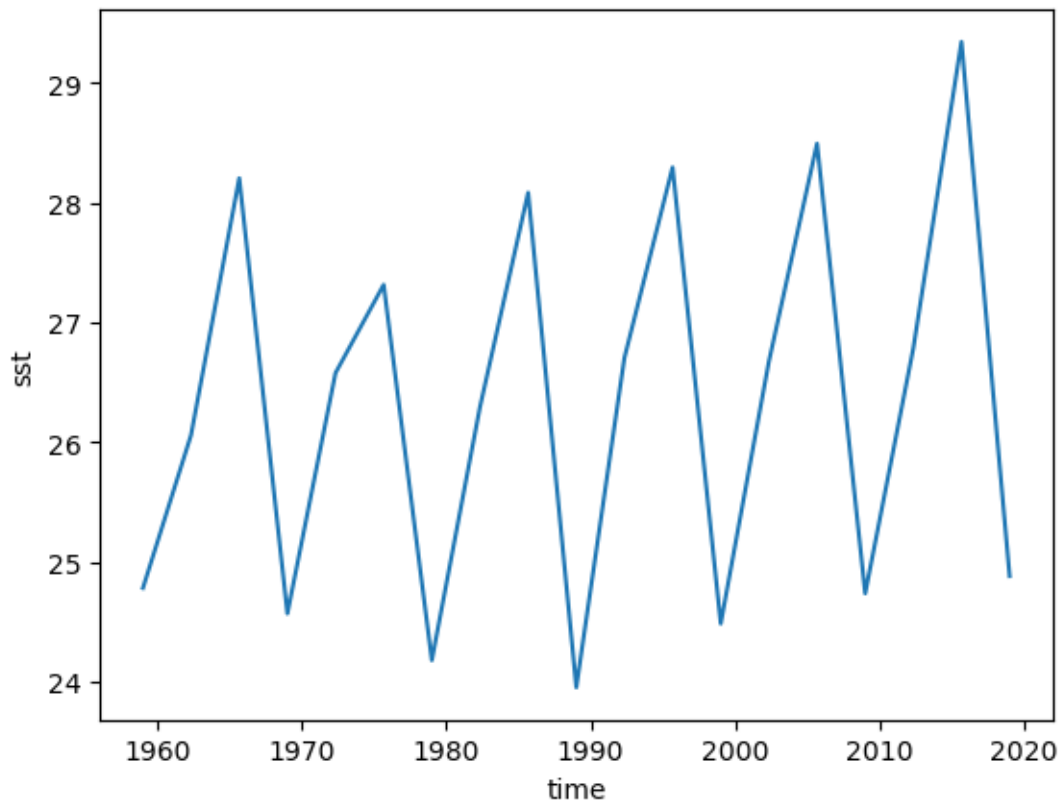
```
* time      (time) datetime64[ns] 6kB 1959-01-01 1959-02-01 ... 2019-01-01
```

Attributes:

```
long_name: time
```

0.0.10 Grafica la SST media (anual) en el periodo

```
[77]: sst_media = sst.mean(dim=['latitude','longitude'])
sst_media.plot();
```



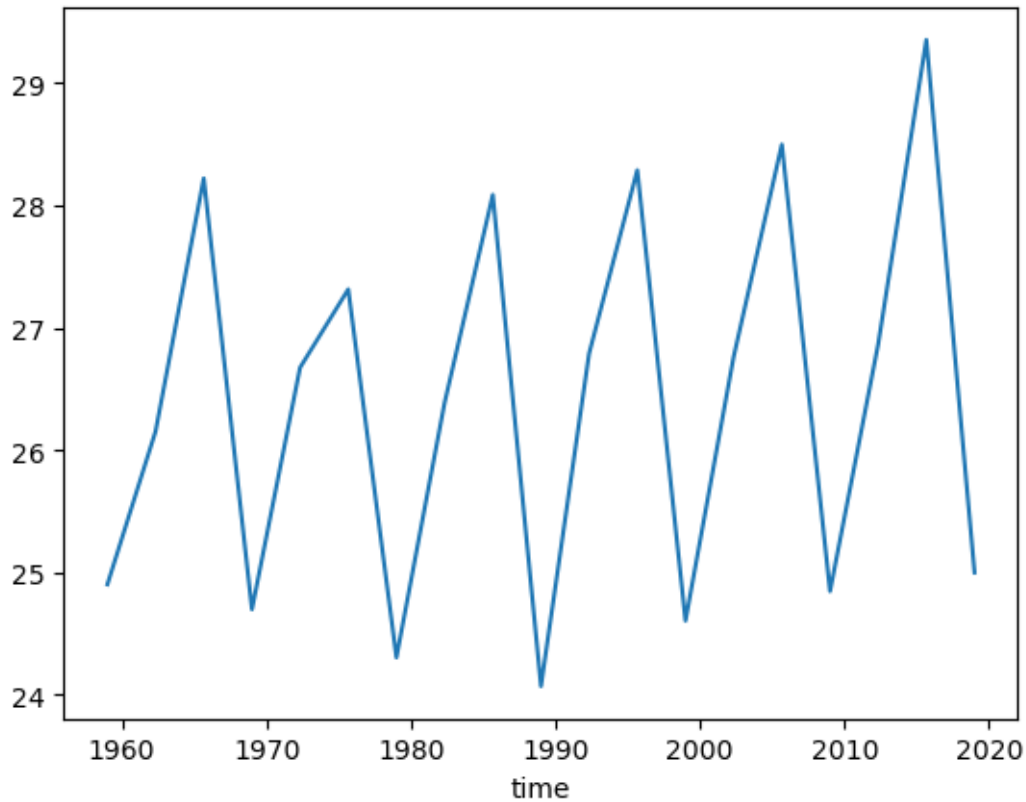
0.0.11 La media ponderada (tiene en cuenta el tamaño de cada pixel)

```
[ ]: try:
    sst.to_netcdf('data/sst.nc')
    result = subprocess.run(["cdo", "gridarea", 'data/sst.nc', 'data/sst_grid.
    nc'],capture_output=True, text=True, check=True)
    if result.returncode == 0:
        print("CDO ejecutado exitosamente")
except subprocess.CalledProcessError as e:
    print(f"Error en CDO: {e.stderr}")
except Exception as e:
    print(f"Error: {e}")
```

CDO ejecutado exitosamente


```
[90]: sst_area = xr.open_dataset("data/sst_grid.nc")
areacello = (sst_area.cell_area * sst[0].notnull())
w = areacello/areacello.sum()
```

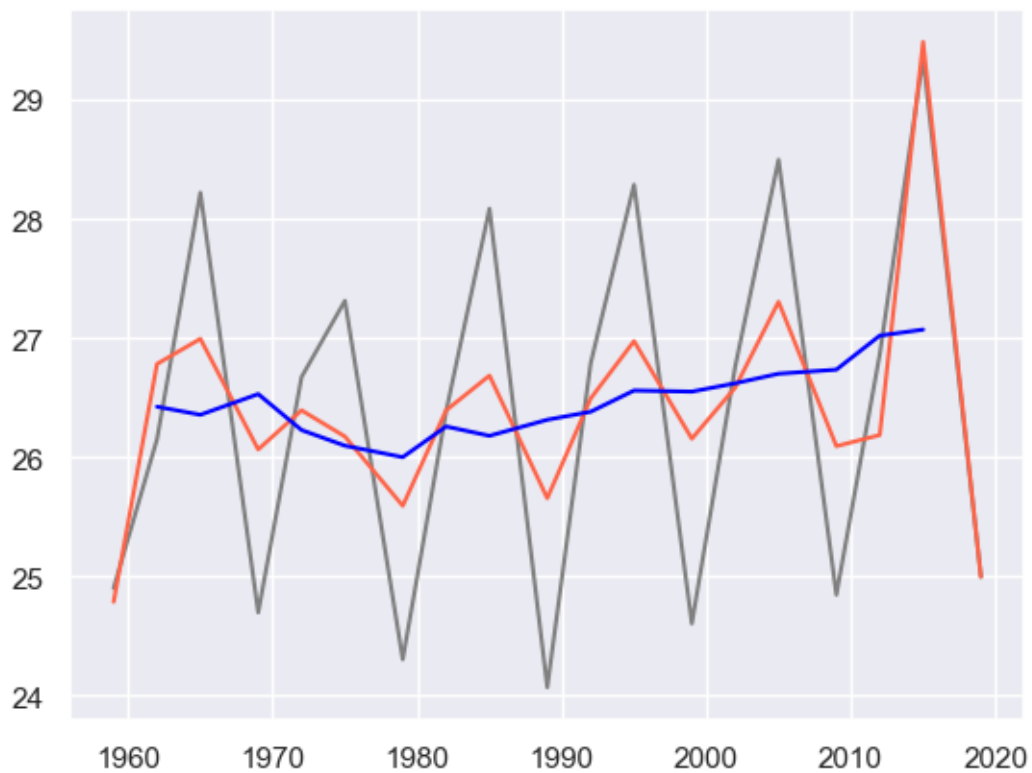
```
[91]: sst_media_pond = (sst * w).sum(dim=['latitude','longitude'])
sst_media_pond.plot();
```



0.0.12 Suavizar la serie temporal

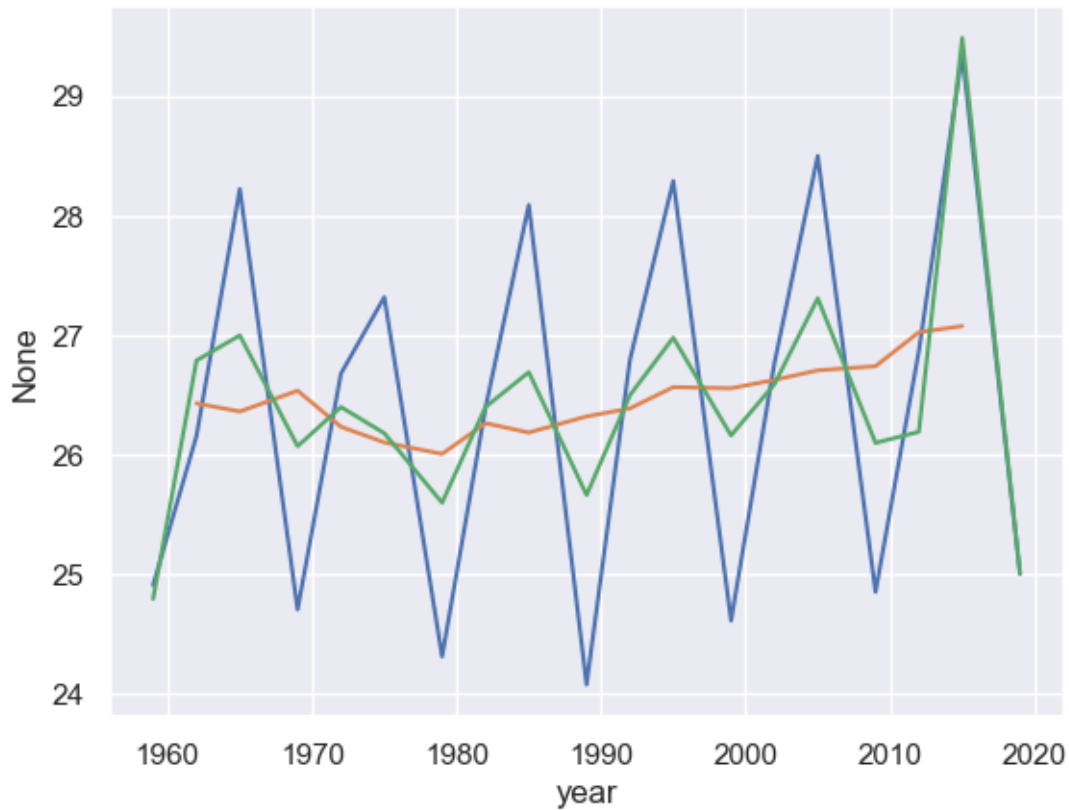
```
[ ]: xs = sst_media_pond.time.dt.year # vector of time in years
movavg_sst_media_pond = sst_media_pond.rolling(time=3,center=True).mean()
    ↪ #Elijo 3 ya que cada punto se diferencia unos 3 años
# Otros métodos que exigen interpolación: ejemplo de la Spline
# Creating a spline regression model of current Tmax
spl_sst_media_pond = UnivariateSpline(xs, sst_media_pond.values,k=4) # 4
    ↪ Degrees of freedom
fig, ax = plt.subplots()
plt.plot(xs,sst_media_pond,'grey')
plt.plot(xs,spl_sst_media_pond(xs),'tomato')
```

```
plt.plot(xs,movavg_sst_media_pond,'blue')  
plt.show()
```



0.0.13 Gráfica con Seaborn

```
[96]: sns.set_theme(style="darkgrid")  
sns.lineplot(y=sst_media_pond,x=xs)  
sns.lineplot(y=movavg_sst_media_pond,x=xs)  
sns.lineplot(y=spl_sst_media_pond(xs),x=xs);
```



0.0.14 Crear dataframe con pandas y graficar

```
[100]: values = {'sst_media_pond': sst_media_pond,
                 'movavg_sst_media_pond': movavg_sst_media_pond,
                 'spl_sst_media_pond': spl_sst_media_pond(xs),
                 }
df = pd.DataFrame(values, index=xs)
df.head()
fig, ax = plt.subplots()
sns.set_theme(style="darkgrid")
sns.lineplot(data=df, palette="tab10", linewidth=2.5,)
plt.ylabel('°C')
plt.show()
```

